

APPLICATION OF MODERN CONTROL THEORY TO THE
CONTROL OF A NUCLEAR POWER REACTOR

by

Luis Tepper

A thesis submitted for the degree of Doctor of Philosophy

Department of Computing and Control
Imperial College of Science and Technology
University of London

August 1972

For Luisa, with thanks.

ABSTRACT

Concepts of modern control theory and in particular of optimal control have been used to develop a general and systematic method of regulator design starting from the non-linear mathematical model of the plant. This method was applied to the study of the control of a nuclear reactor plant of the SGHWR type (steam generating heavy water reactor), of 105 Mw electric power, working at the A.E.A. Establishment, Winfrith.

The method yielded a regulator of reasonably satisfactory performance, according to the control objectives. The plant with the controller incorporated was simulated on a large digital computer as well as on an analog computer, for different cases of design. Only measurable state variables carrying large amounts of feedback control (compared with the rest of the variables) were used on the regulator, with no significant deterioration of performance compared with that obtained feeding back the complete state vector.

In the course of this work a new method of optimal linear regulator design for external disturbances was developed. It is computationally very simple and also allows a simple way of calculating transients by using the same eigenvector matrix regardless of the disturbance.

Five programs have been written for this thesis: the solution of the set of non-linear equations for the steady state, the construction of the linear model, the design of the optimal linear regulator, the free simulation of the

system and lastly the simulation with the control inputs constrained to its limit values.

The complete method, the general description of the programs and the results are reported in this thesis. The appendices include documentation of the more important routines in the various programs, and plots of the simulation runs for the plant with a recommended regulator.

ACKNOWLEDGMENTS

I would like to thank Professor J.H. Westcott for his continued interest, assistance and encouragement. I am very deeply indebted to Professor D.Q. Mayne for his guidance and help throughout the project.

United Kingdom Atomic Energy Authority, Winfrith, have supported the major part of this project; Israel Atomic Energy Commission have given personal financial support in the final stages of the work. Acknowledgment is gratefully made to both institutions.

Thanks are due to Mr. M.H. Butterfield and Dr. J.D. Cummins, from A.E.A., Winfrith, who have contributed by setting the control objectives, explaining the physics of the plant and patiently discussing the work all along the project. Dr. Cummins helped with the running of the analog simulations at Winfrith.

Dr. J.C. Allwright contributed with many fruitful discussions, particularly concerning Chapter 4 of the thesis. Other colleagues from the Control Systems Section helped with stimulating discussions on various aspects of the work, notably H.H. Johnson, J. Katzberg, Y. Shamash and R.W. Wilde. Mrs. M. Bicknell typed part of the manuscript. To all the above mentioned I am grateful.

CONTENTS

<u>Abstract</u>	3
<u>Acknowledgments</u>	5
<u>1. Introduction</u>	10
1.1 Object of the work described.	10
1.2 Description of the system and its existing (conventional) controls.	13
1.3 Control objectives.	20
1.4 Outline of the remainder of the thesis.	21
<u>2. Mathematical description of the system</u>	24
2.1 Lists of equations and parameters.	27
2.1.1 Model X (non-linear, basic).	27
2.1.2 Model X1 (linear, equivalent).	36
2.1.3 Model X2 (linear, comprehensive).	36
<u>3. Outline of the methods of computation</u>	38
3.1 Steady state solution of the non-linear set.	40
3.2 Linear model. Pure time delays. Approximations. Additional equations.	41
3.2.1 Model X (linear, basic).	42
3.2.2 Model X1 (linear, equivalent).	44
3.2.3 Model X2 (linear, comprehensive).	48
3.3 The eigenvalues/eigenvectors problem.	53
3.3.1 Motivation.	53
3.3.2 The old routine.	55
3.3.3 The new routine.	60
3.4 The optimal linear regulator problem. Performance criteria. Constraints.	62
3.4.1 Algorithm for solving the optimal regulator problem.	69

3.5	Optimal regulator for external disturbances.	76
3.6	Free transients computation. Evaluation of the optimal regulator under no constraints.	79
3.7	Saturated transients computation. Constraints on the control inputs.	82
3.8	Reduced set of feedback variables.	87
4.	<u>A new method of optimal linear regulator design for external disturbances</u>	90
4.1	The generalized method for finite terminal time T.	90
4.2	Simplification for infinite terminal time.	96
4.3	Advantages of the method.	97
4.4	Normalization conditions for the eigenvectors of the Hamiltonian matrix.	101
5.	<u>Description of digital and analog programs</u>	103
5.1	SSNL2D (Steady state).	103
5.1.1	Main program.	104
5.1.2	List of subroutines.	105
5.1.3	Description of subroutines.	105
5.1.4	Output.	107
5.2	MXEV2X1 (Linear model).	108
5.2.1	Main program.	108
5.2.2	List of subroutines.	109
5.2.3	Description of subroutines.	110
5.2.4	Output.	116
5.3	REGULA (Optimal linear regulator).	117
5.3.1	Main program.	118
5.3.2	List of subroutines.	120
5.3.3	Description of subroutines.	121
5.3.4	Output.	127

5.4	SIMULA (Free transients computation and display).	128
5.4.1	Main program	128
5.4.2	List of subroutines.	134
5.4.3	Description of subroutines.	136
5.4.4	Output.	142
5.5	SIMSAT (Saturated transients computation and display).	144
5.5.1	Main program.	144
5.5.2	List of subroutines.	149
5.5.3	Description of subroutines.	150
5.5.4	Output.	150
5.6	Description of analog programs.	151
6.	<u>Results</u>	153
6.1	The steady state.	155
6.2	The linear model.	157
6.3	The optimal regulator.	159
6.3.1	Controllability, observability, precision and speed.	161
6.3.2	REGULA with SIMULA	164
6.3.3	REGULA with SIMSAT	169
6.3.4	Reduced set of feedback variables. Recommended filter.	173
6.3.5	The recommended case. Discussion of results. Comments on the analog runs.	181
6.4	Model checking.	188
7.	<u>Conclusions and recommendations for future research</u>	190
	<u>References</u>	196

Appendices

A. Listing of the more important routines in SSNL2D.	199
B. Listing of the more important routines in MXEV2X1.	217
C. Listing of the more important routines in REGULA.	245
D. Listing of the more important routines in SIMULA.	290
E. Listing of the more important routines in SIMSAT.	324
F. List of results of SIMULA for cases of REGULA.	343
G. Simulation plots as produced by SIMULA and SIMSAT.	354
G.1 Case 0126 as plotted by SIMULA.	356
G.2 Case 0126 as plotted by SIMSAT (full set of feedback variables)	365
G.3 Case 0126 as plotted by SIMSAT (reduced set of feedback variables and D.C. compensation).	374
G.4 Checking of the linear model against the analog record of the non-linear model.	383

CHAPTER 1

INTRODUCTION

1.1 Object of the work described

The object of the work described in this thesis is to apply modern control strategies, and in particular concepts of optimal control, to the control of a nuclear power station working at a level of full load or nearly full load.

The control objectives will be described further on. The purpose of the work is to test the problems encountered in using the above mentioned strategies to a large plant, described mathematically by a set of non-linear differential and algebraic equations, including a number of pure transport delays.

The mathematical model, derived from the physical properties of the plant, has not been modified nor reduced. Several reasons contributed to this initial decision:

- (i) The declared intention of tackling the problem of a plant as it stands, existing and working properly controlled in a conventional manner. It was decided not to modify the plant equations in order to find out the practical difficulties encountered in attacking the problem head on.
- (ii) The availability of a large digital computer^(*), capable of handling the controller design and system simulation in a relatively short time. This

(*) The computer used in this study is a CDC 6600 with a store of 131 K of 60 bit-long words.

fact sets the system in the category of medium (rather than large) size. As will be seen, the state matrix is of order 30.

- (iii) The said mathematical model has been and is now being simulated on a large analog computer at the plant site. The dynamic responses obtained follow closely the actual performance of the plant.
- (iv) The mathematical model of the plant was constructed taking due account of the necessity of keeping the number of equations to a minimum, discarding minor side-effects from the outset. It is felt that such an approach to plant-modelling, if it succeeds in keeping the resulting number of equations reasonably small, is more realistic than the reduction by mathematical means - using any of the available methods of model reduction^[1,2] - of a model which has been constructed on an exceedingly large scale in the first place. It could be called a "model reduction" on physical grounds.

In applying the (modern) control strategies to this plant, a number of features are worthwhile mentioning:

- (i) A new method was developed for the design of the optimal linear regulator in the presence of external disturbances and for simulating the resulting system response.
- (ii) The controller designed with the above mentioned method is computed entirely off-line. It can be physically realized either by a small digital computer or by an analog device.

- (iii) The computational methods and programs were designed in such a manner as to be useful for any other plant, provided the mathematical description is given.
- (iv) As will be seen later, the body of programs is divided into four major parts: the steady state, the linear model, the regulator design and the simulation.
- (v) There are two simulation programs: the free simulation in which no constraints are attached and the saturated simulation in which the control inputs are allowed to move only within their permitted constraints.
- (vi) It has not escaped the notice of this author that a number of articles and books^[3,4] have been written on the subject of modern control applied to nuclear reactors. The emphasis is in general on a small number of key equations to be solved. By contrast, the present work makes no attempt at picking the most important equations, using the whole model as it was constructed at the plant site with the help of experts in the various related fields such as neutron physics, thermodynamics, etc. - with the criterion mentioned before (of keeping the resulting model reasonably small) in mind. (*)

(*) At the time of typing this thesis a preprint of a paper for the 5th IFAC Congress became available [5] in which a complete model for a pressurized heavy water reactor is dealt with. This model starts with 220 equations and after two successive reductions ends up with an 18th order state matrix. The regulator design is also based on optimal control type strategies, but the computational methods used are different from those used in this thesis.

1.2 Description of the system and its existing (conventional) controls

The case involves the SGHWR (steam generating heavy water reactor) installed at AEE, Winfrith. The mathematical model was constructed by the Control and Instrumentation Division of that establishment [6,7,8].

The plant, the main dynamic interactions and the existing control system are described in [9,10]. Briefly, the reactor has a nominal output of 300 Mw (thermal) with 105 Mw net electrical output (Fig.1.1). The fuel is slightly enriched Uranium, the moderator in the calandria is heavy water and the coolant flowing through the 104 boiling channels is plain water. The equations describing the dynamics of the reactor assume it to be a point model, wherefore the thermal output is roughly proportional to the average neutron flux and the moderator height.

The reactivity is dependent mainly on the fuel temperature (negative coefficient), coolant voidage (neg. coeff.), moderator height (positive non-linear coeff.), moderator temperature (pos. coeff.) and coolant temperature (neg. coeff.).

The thermal power is directly converted into steam flow to the turbine, which in turn drives the generator. To keep the water level in the steam drum a flow of feed-water is necessary. The generator is connected to the electrical grid at Winfrith but not so in the model of this study, which is meant to drive a local varying load. Figure (1.2) shows the principal dynamic interactions in the plant [10]. Figure (1.3) gives a schematic diagram of

the existing control systems^[6], although not all of them are actually implemented. It is possible to distinguish three main loops plus various emergency controls and logical set-ups (*). The three subsystems are:

- (i) The reactor power output is controlled mainly by moderator height, through its two effects on power and reactivity. At full power this height is 11.5 ft. above core bottom.
- (ii) The steam pressure is controlled by the turbine valve. It stands at 940 p.s.i.a. and the specified allowed variations ought not to be greater than ± 40 p.s.i.a.
For constant pressure the steam flow - and therefore the mechanical power - is of course proportional to the turbine valve opening.
- (iii) The drum-water level is controlled by the feed-water flow valve.

(*) The relevant symbols in Fig.(1.3) are:

W_s	$\equiv$	Mass flow of steam from drum to turbine.
p_d	$\equiv$	Pressure in the steam drum.
l_w	$\equiv$	Level of water in the drum.
W_f	$\equiv$	Mass flow rate of feedwater.
ϕ_i	$\equiv$	Neutron flux as measured by an ionization chamber.
T_{ss}	$\equiv$	Superheat temperature.

This system of control is of the so called "decoupled" type, as against the "coupled control" in which the pressure is controlled by altering reactivity, whereas the power demand is controlled by the steam valve position; this obviates the need for a power controller. In the "decoupled control" used, the response to station load changes is faster.

Table (1.1) gives a list of the main characteristics of the SGHWR at Winfrith, as well as the present values of reactivity coefficients and others.

Table 1.1Characteristics and coefficients of the SGHWR, Winfrith,
at present operating values

Nominal output (thermal)	300 Mw
Electrical output (net)	105 Mw
Fuel	slightly enriched UO_2
Moderator in calandria	heavy water
Coolant flowing through boiling channels	plain water
Number of boiling channels	104
Core diameter	10.25 ft.
Core height	10.5 ft.
Steam pressure at turbine control valve	940 p.s.i.a.
Coolant inlet temperature	527°F
Coolant outlet temperature	538°F
Steam exit quality (maximum channel)	11.3%
Maximum can temperature	554°F
Steam temperature at turbine control valve	532°F
Steam flow rate to turbine	330 lb/sec
Final feedwater temperature	390°F
Drum water level	44 inches

$\bar{f}$ $\equiv$ channel average void fraction

k_v $\equiv$ voidage coefficient of reactivity - 0.02

$\delta k_M / \delta T_M$ $\equiv$ moderator temperature coeff. of reactivity + 15 mN/°C

$\delta k_f / \delta T_f$ $\equiv$ fuel temperature coeff. of reactivity -0.877 mN/°F

$\delta k_c / \delta T_c$ $\equiv$ coolant temperature coeff. of reactivity -2.778 mN/°F

mN $\equiv$ milinile $\equiv 10^{-5}$

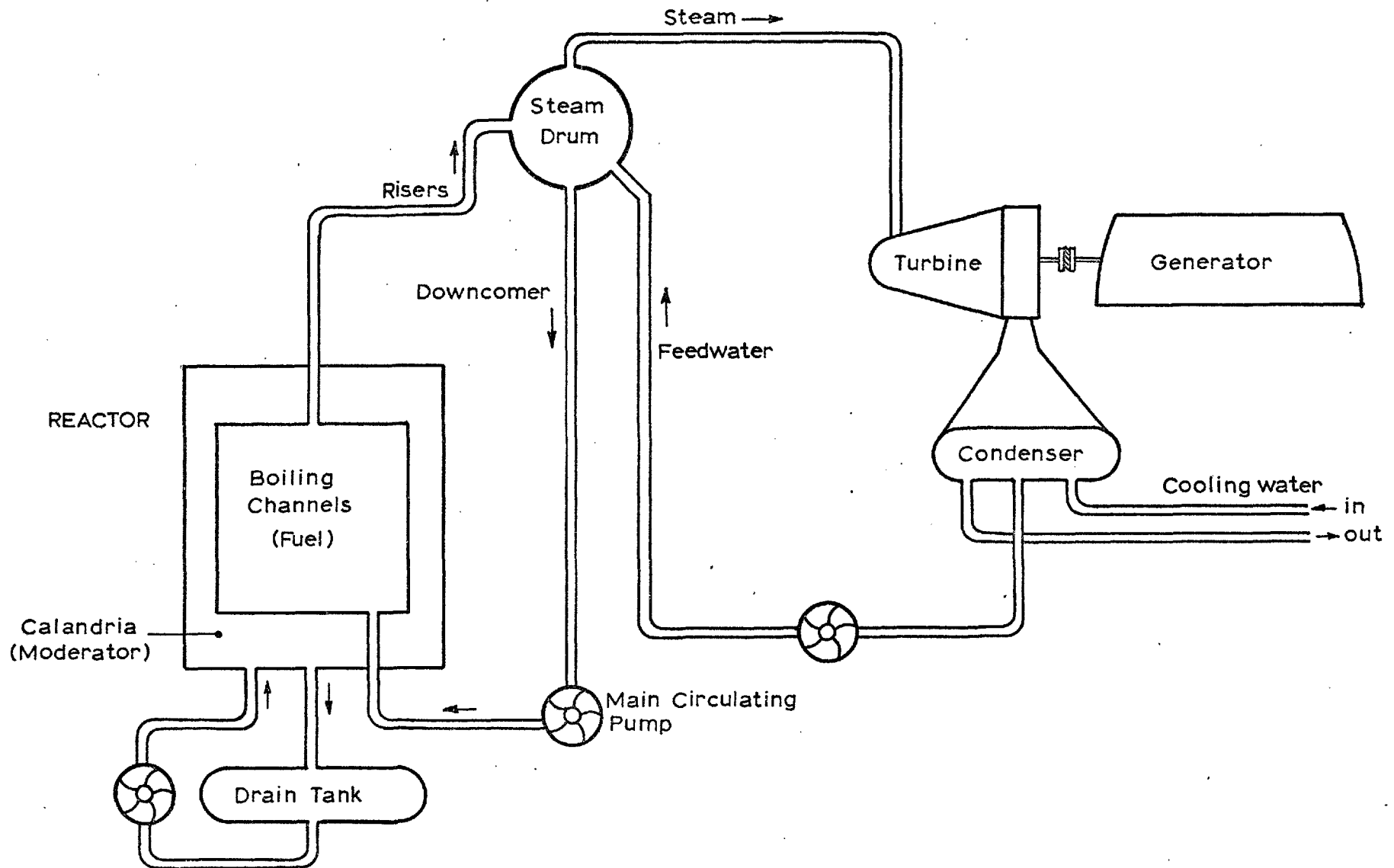


Fig.1-1 Diagram of the basic features of the SGHWR

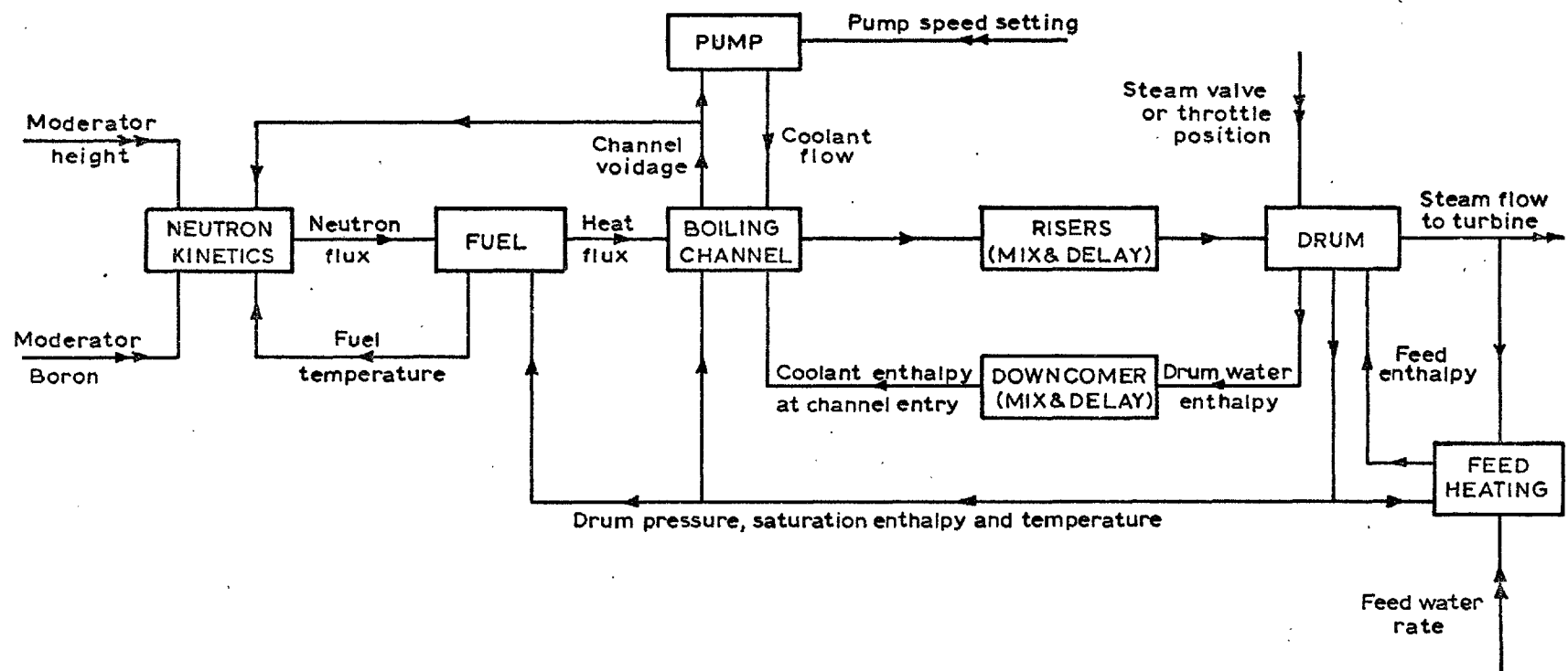


Fig. 1.2 Principal dynamic interactions in SGHWR system

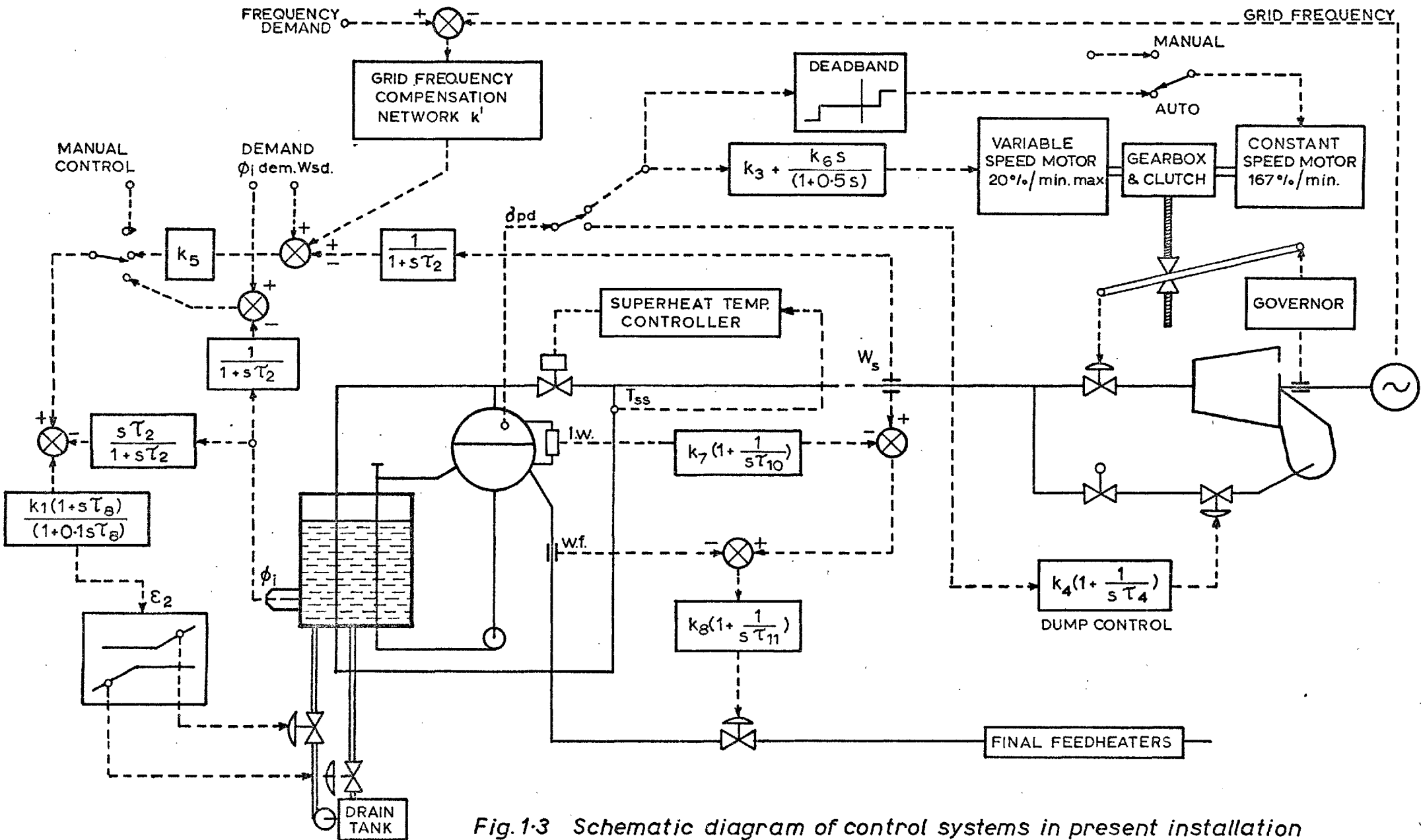


Fig.1-3 Schematic diagram of control systems in present installation at Winfrith.

1.3 Control objectives

The main objectives can be summarized as follows:

- (i) Minimize frequency variations due to rapid changes in load. The droop is 4%, i.e. 2 c/s variation for 100% change in load (normal frequency 50 c/s). The constraint here is not clear cut, it being desirable to keep the variations within 12.5% of the droop (i.e. ± 0.25 c/s) but it certainly should not be larger than 50% (± 1 c/s).

- (ii) Minimize the time involved in power changes.

Specifications of the Central Electricity Generating Board require that 60% of the spinning reserve (i.e. the difference between full power and the operating level) be turned on within 2.5 seconds, the remainder within 2 minutes. For the usual operating level of 75% the above two quantities represent 15% and 10% respectively.

- (iii) Constraint of "dry-out margin": the peak transient power must not pass a certain level, due to safety and engineering considerations. Details will be discussed later, but let it be known that this level is in the region of 110%. If overshoots can be minimized as a consequence of the present study, the operating level - now at 75% - could be raised, with a better economic performance as a result.
- (iv) Minimize frequency variations and overshoots in the presence of plant disturbances.

It will be seen later in the chapter on the optimal regulator problem, that in order to pursue the above mentioned

objectives it is necessary to minimize a performance index, i.e. a functional of the state variables and the control inputs. The objectives must be achieved without violating a number of constraints, some of them of a physical character such as valve openings, others specified for various reasons such as safety and engineering considerations.

The actual plant at Winfrith includes a number of control devices for emergencies as well as for start-up and shut-down procedures. These involve special features for preventing Xenon-poisoning, pump cavitation and for dealing with other problems. The study undertaken here does not cover any of these special schemes and assumes the reactor working at a steady level in the order of full power. In short, it aims at obtaining a performance such that the system recovers as fast and as smoothly as possible from external and plant disturbances.

It has been decided from the outset to work along the lines of what is called "modern" control theory and in particular "optimal control", in order to explore the practical difficulties encountered in applying those concepts to a real, large, working plant.

1.4 Outline of the remainder of the thesis

Chapter 2 deals with the mathematical description of the system as constructed by the Control and Instrumentation Division of AEE, Winfrith. Later the successive versions of the linear model used for the regulator design are described. A complete list of the differential and algebraic

equations, as well as lists of variables, are included.

In Chapter 3 the methods of computation are explained for the successive steps in the solution of the problem posed: the steady state, the linear model, the regulator design, the free simulation (no constraints) and the saturated simulation.

Eigenvalues and eigenvectors were profusely used as a means of computation, hence special attention is focused on the selection of the most appropriate methods for calculating them.

Various methods for the solution of the matrix Riccati equation are reviewed and the one used is described in detail.

Finally the method of selection of the more important feedback variables is described; this leads to the implementation of the "reduced" set and the construction of an appropriate, simple filter.

Chapter 4 is devoted entirely to the new method developed for the design of the linear optimal regulator with external disturbances, and in particular for step-type disturbances (most frequently encountered in the plant under discussion).

In Chapter 5 the various digital programs are described in some detail. This is a rather long chapter and frankly a bit boring; the reason for this is that it comprises a major part of the work on which this thesis is based and may be used as a kind of users manual by anyone using these programs for a different problem or for exploring the same problem more in depth. Unnecessary details, however, have

not been included, or at least tried to be kept to a minimum. The chapter concludes with a brief description of the analog simulations run on the Winfrith computer.

In Chapter 6 the results of the investigation are given and analyzed. This refers to both the digital and analog results as well as the relationship between some of them. Conclusions are drawn and final recommendations given for an implementable controller. This includes of course a "reduced set" of feedback variables and it will be seen that the set comprises only measurable quantities.

Chapter 7 contains the more general conclusions of this research work as well as recommendations for future work.

Chapter 8 shows the references and Chapter 9 the appendices. These are mainly the listings of the more important parts of the digital programs as well as selected parts of their outputs. It would have proved impossible to print the whole programs, for sheer size.

CHAPTER 2

MATHEMATICAL DESCRIPTION OF THE SYSTEM

The basic model includes the equations representing the dynamics of the system at normal operation, excluding the existing controls at the actual plant and excluding load following (i.e. the changes in frequency created by a mis-match between the mechanical power delivered and the electrical power solicited).

This model, which shall be called X, is non-linear and has 23 differential equations, 17 implicit algebraic and 19 explicit algebraic equations. It also includes 4 pure time delays. The equations, the variables and the delays will all be listed below. They were directly transcribed from the Winfrith model^[8] with the exception of a few equations which were rearranged but not intrinsically changed. It should be said at this stage that these and other equations are implemented on an analog computer at Winfrith which is reported to yield transients and general behaviour closely resembling those of the actual plant. The analog computer and the complete set of equations are colloquially called "the simulator" and this word will be referred to whenever the need arises.

Model X (non-linear, basic)

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}, \underline{x}_E, \underline{x}_A, \underline{u}, t) \quad (2.1)$$

$$0 = \underline{g}(\underline{x}, \underline{x}_E, \underline{x}_A, \underline{u}, t) \quad (2.2)$$

$$\underline{x}_E(t) = \underline{h}(\underline{x}, \underline{x}_E, \underline{x}_A, \underline{u}, \underline{x}_D, \underline{x}_{AD}, t) \quad (2.3)$$

where

$\underline{x}$ = vector of 23 state variables

$\underline{x}_A$ = vector of 17 implicit algebraic variables

$\underline{x}_E$ = vector of 19 explicit algebraic variables

$\underline{u}$ = vector of 7 inputs

$\underline{x}_D$ = vector of 3 delayed x variables

$\underline{x}_{AD}$ = vector of 1 delayed x_A variable

Before writing the equations and parameters in detail, let us explain briefly the growth of the original system of 23 state variables into one of 30. Model X (non-linear) was linearized around its steady state values. Details of the linearization process will be explained later. Suffice it now to write the general set of equations:

Model X (linear, basic)

$$\dot{\underline{x}}(t) = A \cdot \underline{x}(t) + A_1 \cdot \underline{x}(t-t_{A1}) + B \cdot \underline{x}_A(t) + C \cdot \underline{u}(t) \quad (2.4)$$

$$\begin{aligned} 0 &= D \cdot \underline{x}(t) + D_1 \cdot \underline{x}(t-t_{D1}) + D_2 \cdot \underline{x}(t-t_{D2}) \\ &\quad + E \cdot \underline{x}_A(t) + E_1 \cdot \underline{x}_A(t-t_{E1}) + F \cdot \underline{u}(t) \end{aligned} \quad (2.5)$$

where $A, \dots, C$ = matrices of 23 rows each

$D, \dots, F$ = matrices of 17 rows each

$t_{A1}, t_{D1}, t_{D2}, t_{E1}$ = pure time delays

$\underline{x}, \underline{x}_A, \underline{u}$ = vectors with the same dimensions as above, representing linear deviations.

This model was constructed and tested, and was intended to be used in an integration routine to compute the time response (details later). But it was decided, in order to be able to tackle the optimal regulator problem with the method of the matrix Riccati equation, to replace the pure time delays with differential equation approximations of the first order. This yielded Model X1 (details later), the final equivalent equation of which is:

Model X1 (linear, equivalent)

$$\dot{\underline{x}}_1(t) = A_s \cdot \underline{x}_1(t) + B_s \cdot \underline{u}(t) \quad (2.6)$$

where A_s, B_s are matrices of 27 rows each computed in a manner to be described later

$\underline{x}_1$ = vector of 27 state variables, 4 of which represent the 1st order differential equations approximating the delays

$\underline{u}$ = vector as above (7 inputs)

Model X1 was further increased to include a number of differential equations existing in the "simulator" and representing some controls and the load following mentioned above. The routine involving this change, called FDBCK1, is open-ended in the sense that it is possible to include any number of new equations and variables. As will be described later, case (3) of FDBCK1 includes feedwater control and load following, involving 3 additional differential equations. Thus we arrive at the final, or comprehensive, Model X2.

Model X2 (linear, comprehensive)

$$\dot{\underline{x}}_2(t) = A_f \cdot \underline{x}_2(t) + B_f \cdot \underline{u}_b(t) \quad (2.7)$$

where A_f, B_f are matrices of 30 rows each, computed as described later

$\underline{x}_2$ = vector of 30 state variables, 3 of which represent the equations introduced in the FDBCK1 routine

$\underline{u}_b$ = vector of 10 inputs; 3 new inputs have been introduced through FDBCK1 - the remainder have the same meaning as the previous vector $\underline{u}_1$ except for 2 of them which are now included in feedback loops and therefore represent reference points.

2.1 Lists of equations and parameters

2.1.1 Model X (non-linear, basic)

The following list of equations, taken from a program listing (subroutine CALFUNO of SSNL2D), requires a few explanations which are also valid for subsequent lists:

- (i) The letter E after a number means 10 raised to the power of the following number.
- (ii) FUN1,...,FUN6 are empirical tables taken from the "simulator" and listed in programs SSNL2D and MXEV2X1.
- (iii) $\underline{x}_d$ = vector of delayed variables.

Explicit algebraic equations

$\underline{f}(\underline{x}, \underline{u})$

Reactivity

$$x_{E_1} = \text{FUN1}(x_{22} - 0.4) \quad (2.1.1)$$

Fuel pin temperature

$$x_{E_2} = 0.5x_8 + 0.5x_9 \quad (2.1.2)$$

$$x_{E_3} = (1.1931x_5 + 1.1569x_6 + 2.5958x_7)/x_{22} \quad (2.1.3)$$

Channel and riser

$$x_{E_4} = \text{FUN2}(x_{10}) \quad (2.1.4)$$

$$x_{E_5} = \text{FUN3}(x_{10}) \quad (2.1.5)$$

$$x_{E_6} = -3.99896E-2x_{13} + 5.18E1 \quad (2.1.6)$$

Steam flow

$$x_{E_7} = 0.3862x_{10}x_{23} \quad (2.1.7)$$

Feedwater

$$x_{E_8} = \text{FUN5}(x_{19}) \quad (2.1.8)$$

Feedheat

$$x_{E_9} = 9.67E-4x_{16} - 0.1082 \quad x_{E_9} \geq 0 \quad (2.1.9)$$

Recirculation

$$x_{E10} = -5.31E-3(x_{14})^2 + 3.578E-3(x_{15})^2 + 1.266E-3x_{14}x_{15} \quad (2.1.10)$$

$$x_{E11} = \text{FUN6}(u_3) \quad (2.1.11)$$

$$\underline{f}(\underline{x}, x_{E1}, \dots, x_{E11}, \underline{u})$$

Recirculation

$$x_{E12} = x_{E6}x_{14} \quad (2.1.12)$$

Feedheat

$$x_{E13} = -0.408654x_{E7}x_{E9} + 0.408654x_{E7} - 1.42749E2 \quad (2.1.13)$$

$$\underline{f}(\underline{x}, x_{E1}, \dots, x_{E11}, \underline{x}_D, \underline{x}_{AD}, \underline{u})$$

Feedheat

$$x_{E14} = x_{20}(t - \text{TAUFW}) \quad (2.1.14)$$

Channel and riser

$$x_{E15} = x_{13}(t - \text{TAUW}) \quad (2.1.15)$$

$$x_{E16} = x_{13}(t - \text{TAURW}) + x_{A4}(t - \text{TAUR}) - x_{E4} \quad (2.1.16)$$

$$\underline{f}(\underline{x}, x_{E1}, \dots, x_{E17}, \underline{u})$$

Channel and riser

$$x_{E17} = 4.329E-7x_{E16}x_{10} + 1.11E-3x_{E16} \quad (2.1.17)$$

$$x_{E18} = 2.1E4x_{E17}/x_{10} \quad (2.1.18)$$

$$x_{E19} = 0.31x_{E18} + 1.2 \quad (2.1.19)$$

Differential equations

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{x}_E, \underline{x}_A, \underline{u})$$

Neutron kinetics

$$\dot{x}_1 = f_1 = (2.5E3x_{A_1} - 1.3E1)x_1 + 3.9x_2 + 9.1x_3 \quad (2.1.20)$$

$$\dot{x}_2 = f_2 = 2.4E-2x_1 - 2.4E-2x_2 \quad (2.1.21)$$

$$\dot{x}_3 = f_3 = 0.15x_1 - 0.15x_3 \quad (2.1.22)$$

Moderator temperature

$$\dot{x}_4 = f_4 = 6.929E-2x_1x_{22} - 3.03E-3x_4 + 3.03E-3u_2 \quad (2.1.23)$$

$$0 \leq x_4 \leq 100$$

Decay heat

$$\dot{x}_5 = f_5 = 9.53E-2x_1x_{22} - 9.53E-2x_5 \quad (2.1.24)$$

$$\dot{x}_6 = f_6 = 1.395E-2x_1x_{22} - 1.395E-2x_6 \quad (2.1.25)$$

$$\dot{x}_7 = f_7 = 4.0E-4x_1x_{22} - 4.0E-4x_7 \quad (2.1.26)$$

Fuel pin temperature

$$\dot{x}_8 = f_8 = 6.736E1x_1 - 0.2564x_8 + 0.2564x_{E_2} + x_{E_3} \quad (2.1.27)$$

$$\dot{x}_9 = f_9 = 6.736E1x_1 - 0.6051x_9 + 0.2564x_{E_2} + 0.3487x_{A_3} \quad (2.1.28)$$

Channel and riser

$$\dot{x}_{11} = f_{11} = -1.43x_{11} + 1.43x_{E_6} (1.0497E3/(1.+x_{A_{14}}) + 1.2597E3) \quad (2.1.29)$$

Steam drum

$$\dot{x}_{12} = f_{12} = -f_{11} - x_{A_{15}} + x_{16} \quad x_{12} \geq 0 \quad (2.1.30)$$

$$\dot{x}_{10} = f_{10} = (1.16E-5x_{10}f_{12} + 0.2639(x_{A_{15}} - x_{E_7}))/ (1.-1.E-5x_{12}) \quad (2.1.31)$$

$$\dot{x}_{13} = f_{13} = (x_{E4}x_{A16} - x_{13}x_{A16} - x_{13}x_{16} + x_{E14}x_{16})/x_{12} \quad (2.1.32)$$

Recirculation

$$\dot{x}_{14} = f_{14} = 4.444x_{E10} - 4.444x_{A17} \quad (2.1.33)$$

$$\begin{aligned} \dot{x}_{15} = f_{15} = & -7.197E-4(x_{15})^2 - 1.29525E1x_{15} - 0.8903x_{E10} \\ & + 2.03145E3 \quad x_{15} \geq 0 \end{aligned} \quad (2.1.34)$$

Feedwater

$$\begin{aligned} \dot{x}_{16} = f_{16} = & -5.2962E-2(x_{16})^2 - 3.704E1x_{10} - 1.85185x_{16} \\ & - 3.704E1x_{17} + 5.055555E4 \end{aligned} \quad (2.1.35)$$

$$\dot{x}_{17} = f_{17} = 3.2E1x_{16} - 2.08E3x_{E8}\sqrt{x_{17}} \quad (2.1.36)$$

$$\begin{aligned} \dot{x}_{19} = f_{19} = & -x_{19} + u_4 \quad -0.1 \leq \dot{x}_{19} \leq 0.1 \quad (2.1.37) \\ & 0 \leq x_{19} \leq 1 \end{aligned}$$

Feedheat

$$\dot{x}_{20} = f_{20} = -2.339E-4x_{16}x_{20} + 2.339E-4x_{E13}x_{16} \quad (2.1.38)$$

Power controller

$$\begin{aligned} \dot{x}_{21} = f_{21} = & -x_{21} + u_5 \quad -0.2 \leq \dot{x}_{21} \leq 0.2 \quad (2.1.39) \\ & -1 \leq x_{21} \leq 1 \end{aligned}$$

$$\dot{x}_{22} = f_{22} = 8.56E-4x_{21} \quad (2.1.40)$$

Turbine

$$\begin{aligned} \dot{x}_{23} = f_{23} = & -4.0x_{23} + 4.0u_6 - 2.0u_7 - 2.0 \quad (2.1.41) \\ & 0 \leq x_{23} \leq 1 \\ & -1 \leq u_6 \leq 1 \end{aligned}$$

Turbine and alternator

$$\dot{x}_{18} = f_{18} = -2.0x_{18} + 0.6378x_{E7} \quad (2.1.42)$$

Implicit algebraic equations

$$g(\underline{x}, \underline{x}_E, \underline{x}_A, \underline{u}) = 0$$

Reactivity

$$\begin{aligned} 0 = g_1 = & -x_{A1} - 2.0E-2x_{A2} - 2.778E-5x_{A3} + 1.5E-4x_4 \\ & + x_{E1} - 8.77E-6x_{E2} + u_1 \end{aligned} \quad (2.1.43)$$

Channel and riser

$$0 = g_2 = -x_{A2} + 0.988x_{A13}x_{A8}/x_{A4} \quad (2.1.44)$$

$$0 = g_3 = -x_{A3} - 0.5(x_{A8})^2/x_{A4} + 0.5x_{A4} + x_{E15} \quad (2.1.45)$$

$$0 = g_4 = -x_{A4} + 2.81E5x_{A5}/x_{E12} \quad (2.1.46)$$

$$0 = g_5 = -x_{A5} - 2.5118E-3x_{A3}x_{22} + 2.5118E-3x_9x_{22} \quad (2.1.47)$$

$$0 = g_6 = -x_{A6} + x_{E5}x_{A7} + x_{E4} - x_{E15} \quad x_{A6} \geq 0 \quad (2.1.48)$$

$$0 = g_7 = -x_{A7} + x_{A10} + 0.5x_{A11} \quad (2.1.49)$$

$$0 = g_8 = -x_{A8} + x_{A4} - x_{A6} \quad x_{A8} \geq 0 \quad (2.1.50)$$

$$0 = g_9 = -x_{A9} + x_{A7} + x_{10} \quad (2.1.51)$$

$$0 = g_{10} = -x_{A10} + 1.085E-3(x_{14})^2(x_{A14}+1.0) + 7.0/(x_{A14}+1.0) \quad (2.1.52)$$

$$0 = g_{11} = -x_{A11} - 4.16x_{A2} + 1.01314E-3(x_{14})^2x_{A12} + 7.05E-4(x_{14})^2 + 4.16 \quad (2.1.53)$$

$$0 = g_{12} = -x_{A12} + 2.7514E1x_{A8}/x_{A9} \quad x_{A12} \geq 0 \quad (2.1.54)$$

$$\text{slip ratio} = \gamma = 1.$$

$$0 = g_{13} = -x_{A13} + \text{FUN4}(x_{A12}) \quad (2.1.55)$$

$$0 = g_{14} = -x_{A14} + 0.547645x_{A12}/x_{E19} + 0.5x_{E18}/x_{E19} \quad (2.1.56)$$

$$x_{A14} \geq 0$$

Steam drum

$$0 = g_{15} = -x_{A15} + x_{E17}(x_{A15}+x_{A16}) \quad (2.1.57)$$

$$0 = g_{16} = -x_{A16} - x_{A15} + x_{E12} - f_{11} \quad (2.1.58)$$

Recirculation

$$0 = g_{17} = -x_{A17} + x_{A10} + x_{A11} + 2.1225E-2(x_{14})^2x_{E11} + 3.13E-3(x_{14})^2 - 1.116E1 \quad (2.1.59)$$

Time delays

$\underline{f}_\tau(\underline{x}, \underline{x}_E, \underline{x}_A)$

Recirculation

$$\text{TAUW} = 5.675E2/x_{14} \quad (2.1.60)$$

Feedheat

$$\text{TAUFW} = 8.55\text{E}3/x_{16} \quad (2.1.61)$$

Channel and riser

$$\text{TAUR} = 1.048\text{E}3/(x_{E19}x_{14} + x_{E19}x_{14}x_{A14}) \quad (2.1.62)$$

$$\text{TAURW} = \text{TAUR} + \text{TAUW} \quad (2.1.63)$$

Output variables (feedback to no other variables)

$f_0(\underline{x}, \underline{u})$

Ion chamber flux

$$y_1 = -0.8729x_1x_{22} + 1.838x_1 \quad (2.1.64)$$

Moderator temperature

$$y_2 = 2.0x_4 - u_2 \quad \begin{aligned} 0 \leq x_4 \leq 100^\circ\text{C} \\ 0 \leq y_2 \leq 100^\circ\text{C} \\ u_2 = 27.7^\circ\text{C} \end{aligned} \quad (2.1.65)$$

Feedwater level

$$y_3 = 1.268\text{E}-3x_{12} \quad (2.1.66)$$

List of parameters, names corresponding to the "simulator"

Names	Units	Description
<u>Inputs</u>		
DKBAL	u_1	Reactivity adjustment to balance reactivity
TMIN	u_2 °C	Moderator inlet temperature
YDV	u_3 normalized	Discharge valve position from fully open (6 inches)
YRDEM	u_4 normalized	Demanded position of YR (100 pc feedwater valve)
YMDEM	u_5 normalized	Demanded position of YM (Moderator valve)
YTCVDEM	u_6 normalized	Speeder motor demanded position of YTCV (Turbine control valve)
DF	u_7 c/s	Frequency error

Note: if $u_5 \neq 0$, moderator height changes

State variables x_1 to x_{23}

PHIDASH	x_1	normalized	Neutron flux
X1BARDASH	x_2	normalized	1st group delayed neutrons effect
X2BARDASH	x_3	normalized	2nd group delayed neutrons effect
TMAV	x_4	$^{\circ}\text{C}$	Mean moderator temperature
A1	x_5	normalized	1st group decay heat term
A2	x_6	normalized	2nd group decay heat term
A3	x_7	normalized	3rd group decay heat term
TF1STAR	x_8	$^{\circ}\text{F}$	Mean temperature of fuel centre zone (offset 400)
TF2STAR	x_9	$^{\circ}\text{F}$	Mean temperature of fuel outer zone (offset 400)
PD	x_{10}	p.s.i.a.	Steam drum pressure
M	x_{11}	lb	Total mass of water and two phase mixture in primary circuit
MW	x_{12}	lb	Mass of water in steam drum
HDSTAR	x_{13}	btu/lb	Enthalpy of steam drum water
QW	x_{14}	ft^3/sec	Volumetric flow of water through main circulators
OMEGA	x_{15}	rad/sec	Circulator pump speed
WFW	x_{16}	lb/sec	Mass flow rate of water into steam drum
DPREG	x_{17}	p.s.i.a.	Pressure drop across feedwater regulator valves
PMECH	x_{18}	Mw(el.)	Mechanical power produced by turbine
YR	x_{19}	normalized	100 pc feedwater valve position
HFHSTAR	x_{20}	btu/lb	Enthalpy of feedwater leaving feedheaters (offset 400)
YM	x_{21}	normalized	Moderator valve position
HDASH	x_{22}	normalized	Moderator height from core bottom (normalized to 12.5 feet)
YTCV	x_{23}	normalized	Turbine control valve position

Implicit algebraic variables

DK	x_{A1}		Excess reactivity
FBAR	x_{A2}		Channel average void fraction
TCOOLSTAR	x_{A3}	$^{\circ}\text{F}$	Channel average coolant temperature (offset 400)
DH	x_{A4}	btu/lb	Enthalpy rise through core
QDASH	x_{A5}	normalized	Thermal power output from fuel to coolant (normalized to 296 Mw(th.))
DHSC	x_{A6}	btu/lb	Enthalpy of subcooling
DPSC	x_{A7}	p.s.i.a.	Pressure difference between channel and steam drum
DHB	x_{A8}	btu/lb	Enthalpy rise of coolant through boiling region
PC	x_{A9}	p.s.i.a.	Mean channel pressure
DPR	x_{A10}	psia	Pressure drop through risers
DPC	x_{A11}	psia	Pressure drop through channel
KXCOUT	x_{A12}		K.quality at channel outlet
F4	x_{A13}		Function of x_{A12}
KXRBAR	x_{A14}		K.average riser quality
WSRISER	x_{A15}	lb/sec	Mass flow rate of steam from risers into drum
WWRISER	x_{A16}	lb/sec	Mass flow rate of water from risers into drum
DPREC	x_{A17}	psia	Pressure drop in the recirculation loop

Explicit algebraic variables

DKM	x_{E1}		Reactivity effect of moderator height
TFAVSTAR	x_{E2}	$^{\circ}\text{F}$	Mean temperature of fuel (offset 400)
HGTF	x_{E3}		Function of the 3 decay heat terms and normalized moderator height
HWSTAR	x_{E4}	btu/lb	Enthalpy of water at saturation
DHW/DP	x_{E5}	btu/lb/psia	F3 a function of pressure

ROW	x_{E6}	lb/ft ³	Density of water at mean channel temper.
WSDRUM	x_{E7}	lb/sec	Mass flow rate of steam from drum into turbine (dumps not considered here)
F5	x_{E8}		Function of YR
F(WFW)	x_{E9}		Function of WFW
DPP	x_{E10}	psia	Pressure rise through main circulators
FUNCTYDV	x_{E11}		F6, discharge valve characteristic
WW	x_{E12}	lb/sec	Mass flow rate of water in the downcomers
HFHOSTAR	x_{E13}	btu/lb	Steady state value of HFHSTAR (not reference)
HFWSTAR	x_{E14}	btu/lb	Enthalpy of feedwater at drum inlet (offset 400, delayed value of HFHSTAR)
HISTAR	x_{E15}	btu/lb	Enthalpy of coolant at core inlet (offset 400, delayed value of HDSTAR)
B	x_{E16}	btu/lb	Function of HWSTAR and delayed values of HISTAR and DH
XR	x_{E17}		Riser exit quality
KXROUT	x_{E18}		K.Riser exit quality
GAMMAR	x_{E19}		Riser slip ratio

Transport delay times

TAUFW	t_{A1}	seconds	From last feedheater exit to drum inlet (feedwater)
TAUW	t_{D1}	sec	From drum outlet to core inlet (downcomer)
TAURW	t_{D2}	sec	Sum of TAUW and TAUR
TAUR	t_{E1}	sec	From core outlet to drum inlet (riser)

Output variables

PHIION	y_1	normalized	Ion chamber indication of neutron flux
TMOUT	y_2	°C	Moderator outlet temperature
L	y_3	inches	Steam drum level

2.1.2. Model X1 (linear, equivalent)

The appropriate equations will be found in section (3.2.2).
The list of additional state variables adopted (first order approximation for each delay) is:

$x_{24} = z_1 = \text{DELAYVAR1} = \text{variable for approximating delay } t_{A1}$

$x_{25} = z_2 = \text{DELAYVAR2} = \text{variable for approximating delay } t_{D1}$

$x_{26} = z_3 = \text{DELAYVAR3} = \text{variable for approximating delay } t_{D2}$

$x_{27} = z_4 = \text{DELAYVAR4} = \text{variable for approximating delay } t_{E1}$

2.1.3. Model X2 (linear, comprehensive)

The appropriate equations leading to the final model will be found, in their general form, in section (3.2.3).

The list of additional parameters for "case 3" used for the optimal linear regulator problem is:

State variables

$x_{28} = w_1 = \text{IF1} = \text{feedwater controller variable 1}$

$x_{29} = w_2 = \text{IF2} = \text{feedwater controller variable 2}$

$x_{30} = w_3 = \text{DF} = \text{frequency error}$

Inputs

$u_{28} = \text{PE} = \text{Electric power delivered}$

$u_{29} = \text{WSDDEM} = \text{Demanded WSDDRUM (normalized to 344.91 lb/sec)}$

$u_{210} = \text{L1DEM} = \text{Steam drum demanded level for YR controller (normally 44 inches)}$

Outputs

$y_4 = \text{PMECH} = \text{Mechanical power produced by turbine } (x_{18})$

$y_5 = \text{WSDRUM} = \text{Mass flow rate of steam from drum } (x_{E7})$

$y_6 = \text{WFW} = \text{Mass flow rate of water into drum } (x_{16})$

The list of additional equations for case(3) is as follows:

Load following

$$y_4 = x_{18} \quad (2.1.67)$$

$$\dot{w}_3 = TF_1 y_4 - TF_1 u_{28} \quad (2.1.68)$$

$$u_{27} = w_3 \quad (2.1.69)$$

where TF_1 = Transfer function gain (integrator) from
 accelerator torque to DF
 = 0.0515

u_{27} is an input with feedback.

Feedwater control

$$y_{3(err)} = y_3 - u_{210} \quad (2.1.70)$$

$$y_3 = 0.1268E-2x_{12} \quad (2.1.71)$$

$$y_6 = x_{16} \quad (2.1.72)$$

$$\begin{aligned} \dot{w}_2 = & - (TK_{14}/TAU_{11})w_1 - (TK_{14}/TAU_{11})y_3 + (0.0029TK_{15}/TAU_{11})y_5 \\ & - (0.0029/TAU_{11})y_6 + (TK_{14}/TAU_{11})u_{210} \end{aligned} \quad (2.1.73)$$

$$\dot{w}_1 = (1./TAU_{10})y_3 - (1./TAU_{10})u_{210} \quad (2.1.74)$$

$$\begin{aligned} u_{24} = & TK_{13}w_2 - TK_{13}TK_{14}w_1 - TK_{13}TK_{14}y_3 + 0.0029TK_{13}TK_{15}y_5 \\ & - 0.0029TK_{13}y_6 + TK_{13}TK_{14}u_{210} \end{aligned} \quad (2.1.75)$$

where TK_{13} = Transfer function gain for YRDEM
 = 5.0

TK_{14} = First transfer function gain for YR controller
 = 0.025 (critical)

TK_{15} = Set ratio in YR controller
 = 1.0

TAU_{10} = Time constant for YR controller (1st transf. function)
 = 200.0 sec

TAU_{11} = Time constant for YRDEM
 = 100.0 sec

u_{24} is an input with feedback.

CHAPTER 3

OUTLINE OF THE METHODS OF COMPUTATION

The solution of the problem posed in this study is broken into five main components, to be explained here. The pertinent programs are described in detail in Chapter 5. They are:

- (i) Solution of the steady state of the set of non-linear equations
- (ii) Construction of the linear model
- (iii) Solution of the optimal linear regulator problem
- (iv) Transients computation (or "free simulation") for the original system and those computed via (iii)
- (v) "Saturated simulation", i.e. the transients computation described in (iv) with the control inputs constrained to move only within their physical limits.

Perhaps it would be worthwhile, before going into detail, to explain why each of these steps were taken.

(i) The steady state calculation is necessary as it yields the point of departure for the (linear) deviations on which this work is based. Actually this computation is necessary not only for one but for various levels of operations. It is conceivable to envisage, as a sequel to this study, a piece-wise linear model of the system, switching from one set of steady states to another as the amplitude of the transients grow.

Also, the non-linear model had to be constructed in order to check the correctness of the linear model.

(ii) The reasons for working with a linear rather than with

a non-linear model are, in general, the usual ones advanced for this approach, namely the more or less close approximation for relatively small signals and the vast amount of knowledge developed for linear systems. This is of course particularly true for a system of high order. In addition, the methods of control, including optimal control, which it is the purpose of this study to apply in a straightforward manner, cater for linear dynamical systems.

(iii) The optimal regulator problem is the design of a "feedback controller such as to keep a stationary system within an acceptable deviation from a reference condition using acceptable amounts of control".^[11] Our study deals with two kinds of perturbation:

- (a) state perturbation (e.g. a change in the drum steam pressure), in which case the system must eventually settle down at the level where it was before the disturbance.
- (b) external disturbance, such as a change in the electric power required; if this change is a step function or any other function with a permanent off-set, the state will settle at a different level from the one it was forced to depart, a level consistent with the new set of inputs.

This problem is tackled by solving the steady state matrix Riccati equation, which minimizes a performance index made up of an integral of quadratic forms in the state and the control.

(iv) The transients computation or linear free simulation is done for a dual purpose:

- (a) To check the "correctness" of the model by comparing it with transients produced on the Winfrith simulator
- (b) to evaluate the quality of the "optimal" controllers found by the method outlined in (iii), i.e. violation of constraints, magnitude of overshoots, time constants, etc.
- (v) The saturated simulation is done for purpose (b) of the preceding paragraph, the control inputs being forbidden to exceed their physical limits. This simulation is thus much closer to the real working of the plant.

3.1 Steady State Solutions of the non-linear set

The object is to find the solution to the set of non-linear equations consisting of 23 steady-state differential equations and 17 implicit algebraic equations, described as Model X (non-linear, basic) and listed in Section 2.1.

On the face of it, by setting all derivatives to zero we have a set of 40 non-linear equations with 40 unknown variables to solve. This was indeed the first, albeit naive, approach. It led us into obtaining multiple solutions in the vicinity of a certain point. It proved necessary to inspect closely the differential equations (with derivatives equal to zero) to find that some of them are linear combinations of others, while yet others are entirely irrelevant, i.e. redundant. Thus a set of only 35 equations and variables was arrived at (it will be explained later how the re-programming was done). From the point of view of the original set, this means that some of the variables are actually constants, as far as the

steady state is concerned. Conversely, some of the inputs are actually unknowns, i.e. outputs of the computation (*).

The central algorithm used for this problem is the Powell method^[12] for minimizing a sum of squares of non-linear functions, those representing the 35 equations containing initially chosen values in place of the unknown quantities. This by now well known method is similar to the classical "generalized least squares" method, but, unlike it, does not require any derivatives to be computed. It is therefore less cumbersome and more efficient.

Since Powell's method works best when the values of the variables have the same order of magnitude, a routine for scaling and descaling them to unit order before and after the minimization procedure has been developed. The orders are known from similar steady states. The same is true for the numerical values of the functions, although there is a difference because they keep changing their order by the very process of minimization. Also, it is necessary that these values be small. Therefore another routine was developed, for calculating the scaling factors of the functions after the first call for the functions evaluation. As usual, details of the programming will be given later.

3.2 Linear Model. Pure Time Delays. Approximations.

Additional Equations

As mentioned in Chapter 2, three successive linear models were constructed (Models X, X1, and X2) in order

(*) The names inputs and variables apply to the dynamic model only.

to be used for the control study. The method of linearization common to all models is the well-known one of expanding the non-linear system in a Taylor series about the steady state and dropping all but the first order terms, i.e. in general, to linearize a non-linear system

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}) \quad (3.2.1)$$

about an equilibrium state $\underline{x}_0, \underline{u}_0$, that is

$$\dot{\underline{x}} = \underline{f}(\underline{x}_0, \underline{u}_0) = 0 \quad (3.2.2)$$

and obtain a linear system of the form

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u} \quad (3.2.3)$$

The matrices A and B are determined such that

$$[a_{ij}] = \frac{\partial f_i}{\partial x_j} \quad \underline{x}_0, \underline{u}_0 \quad (3.2.4)$$

$$\text{and} \quad [b_{ij}] = \frac{\partial f_i}{\partial u_j} \quad \underline{x}_0, \underline{u}_0 \quad (3.2.5)$$

3.2.1 Model X (linear, basic)

This model will be discussed only in so far as it was constructed and tested, with the intention of calculating the time response of the system by considering the delayed-variable-terms as inputs. But work on it was not carried on to further stage, for the reasons advanced in Chapter 2, namely the unsuitability of this model for a Riccati-type solution of the optimal control problem. Let us repeat equations (2.4), (2.5)

$$\dot{\underline{x}}(t) = \underline{A} \underline{x}(t) + \underline{A}_1 \underline{x}(t-t_{A1}) + \underline{B} \underline{x}_A(t) + \underline{C} \underline{u}(t) \quad (3.2.6)$$

$$0 = \underline{D} \underline{x}(t) + \underline{D}_1 \underline{x}(t-t_{D1}) + \underline{D}_2 \underline{x}(t-t_{D2}) + \underline{E} \underline{x}_A(t)$$

$$+ \underline{E}_1 \underline{x}_A(t-t_{E1}) + \underline{F} \underline{u}(t) \quad (3.2.7)$$

From (3.2.6) and (3.2.7)

$$\dot{\underline{x}} = \underline{A}_s \underline{x}(t) + \underline{D}_s(t) + \underline{B}_s \underline{u}(t) \quad (3.2.8)$$

where

$$\underline{A}_s = \underline{A} - \underline{B}\underline{E}^{-1}\underline{D} \quad (3.2.9)$$

$$\underline{B}_s = \underline{C} - \underline{B}\underline{E}^{-1}\underline{F} \quad (3.2.10)$$

$$\begin{aligned} \underline{D}_s(t) = & \underline{A}_1 \underline{x}(t-t_{A1}) - \underline{B}\underline{E}^{-1}\{\underline{D}_1 \underline{x}(t-t_{D1}) + \underline{D}_2 \underline{x}(t-t_{D2}) \\ & + \underline{E}_1 \underline{x}_A(t-t_{E1})\} \end{aligned} \quad (3.2.11)$$

The solution of (3.2.8) over a time period Δ is

$$\underline{x}(t+\Delta) = \phi(\Delta) \underline{x}(t) + \int_t^{t+\Delta} \phi(t+\Delta-\tau) [\underline{D}_s(\tau) + \underline{B}_s \underline{u}(\tau)] d\tau \quad (3.2.12)$$

where $\phi(t)$ is the state-transition matrix

$$\phi(t) = e^{\underline{A}_s t} \quad (3.2.13)$$

Equation (3.2.12), with suitable use of a numerical integration routine for integrating over period Δ (suitably chosen), should have provided a reasonably fast simulation of the system. But as was mentioned before, this integration was not carried out and a new model was constructed in order to enable the optimal control problem to be solved straightforwardly.

A considerable amount of soul searching was done before going ahead with model X1, which comprises approximation of the pure time-delays, on the grounds that even the crudest approximation increases the state-vector by four components, while a better one would increase it by, say, 8 to 24 components. The fact that first order approximation was also used at the Winfrith analog simulator with apparently satisfactory results, eventually helped in the decision.

3.2.2 Model X1 (linear, equivalent)

As mentioned in Chapter 2 and par. (3.2.1) this model includes approximations to the four pure time delays of the system. The basis of the approximations are tables of values calculated from orthogonal functions such as Legendre polynomials or trigonometric functions,^[13] which Reeve (1968) showed to be a more general approach than the Padé approximations or some generalizations of them.^[14] The advantages of Reeve's method lie more in higher order approximations, and since it was decided to use a low order in the first place, there is not really much difference either way. The 1st order trigonometric function approximation seemed the best "value for money" choice, but the routine written (DELAY in MXEV2X1, see section 5.2.3) is capable of dealing with higher orders and different functions as well.

For this choice, the approximation for each delay is:

$$x_d = -x + 2\tau^{-\frac{1}{2}}z \quad (3.2.14)$$

$$\dot{z} = 2\tau^{-\frac{1}{2}}x - 2\tau^{-1}z \quad (3.2.15)$$

where x_d is the variable x delayed by τ and z is a new state variable introduced.

This gives a step response of the shape of a negative unity-amplitude drop, followed by an exponential rise towards the input amplitude, with a time constant equal to one half of the transport delay.

For convenience, equations (3.2.14,15) are changed, without affecting the response (this is easily verifiable) into:

$$\underline{x}_d = -\underline{x} + 4.\tau^{-1}\underline{z} \quad (3.2.16)$$

$$\dot{\underline{z}} = \underline{x} - 2.\tau^{-1}\underline{z} \quad (3.2.17)$$

Before writing the generalized equations for the time delay approximations, let us write the basic equations of the system, as obtained from linearizing the N/L model (they may be compared with equations (3.2.6,7)):

$$\dot{\underline{x}}(t) = \underline{A}\underline{x}(t) + \underline{B}\underline{x}_A(t) + \underline{B}_D\underline{x}_D(t) + \underline{C}\underline{u}(t) \quad (3.2.18)$$

$$0 = \underline{D}\underline{x}(t) + \underline{E}\underline{x}_A(t) + \underline{E}_D\underline{x}_D(t) + \underline{F}\underline{u}(t) \quad (3.2.19)$$

where $\underline{x}, \underline{x}_A, \underline{u} \equiv$ vectors as dimensioned before, representing linear deviations

$\underline{x}_D \equiv$ vector of 4 delayed variables (3 state variables, 1 implicit algebraic variable)

$\underline{A}, \underline{B}, \underline{B}_D, \underline{C} \equiv$ matrices of 23 rows each

$\underline{D}, \underline{E}, \underline{E}_D, \underline{F} \equiv$ matrices of 17 rows each

Now the pure time delays are approximated in the following general way;

$$\dot{\underline{z}}(t) = \underline{P}\underline{x}(t) + \underline{P}_A\underline{x}_A(t) + \underline{Q}\underline{z}(t) \quad (3.2.20)$$

$$\underline{x}_D(t) = \underline{G}\underline{x}(t) + \underline{G}_A\underline{x}_A(t) + \underline{H}\underline{z}(t) \quad (3.2.21)$$

where $\underline{z}$ is a vector of L new state variables

(L = 4 for 1st order approximation in all the delays)

$$\underline{z} \equiv [\underline{z}_1 \ \underline{z}_2 \ \underline{z}_3 \ \underline{z}_4]^T$$

$\underline{z}_i$ has a dimension equal to the order of approximation for $\underline{x}_{Di}$

$\underline{P}, \underline{P}_A, \underline{Q} \equiv$ matrices of L rows each

$\underline{G}, \underline{G}_A, \underline{H} \equiv$ matrices of 4 rows each

The following set holds for each time delay (compare it with the specific 1st order chosen case of equations (3.2.16,17)):

$$\dot{\underline{z}}_i = \underline{p}_i x_{ni} + \underline{Q}_i \underline{z}_i \quad (3.2.22)$$

$$x_{Di} = g_i x_{ni} + \underline{h}_i^T \underline{z}_i \quad (3.2.23)$$

where $i = 1, 2, 3, 4$

$$x_{n1} = x_{20}$$

$$x_{n2} = x_{13}$$

$$x_{n3} = x_{13}$$

$$x_{n4} = x_{A4}$$

and

$L_i \equiv$ order of approximation of x_{Di}

$\underline{z}_i \equiv$ vector of L_i new state variables

$\underline{p}_i \equiv$ vector of L_i components set according to
desired type of approximation

$\underline{h}_i \equiv$ vector of L_i components set as for $\underline{p}_i$

$\underline{Q}_i \equiv$ matrix of $L_i \times L_i$ elements set as for $\underline{p}_i$

$g_i \equiv$ scalar set as for $\underline{p}_i$

$$L = L_1 + L_2 + L_3 + L_4$$

The matrices in (3.2.20,21) are formed by the components in (3.2.22,23) in the following manner:

$$P = \begin{bmatrix} & & [\underline{p}_1] & \\ 0 & [\underline{p}_2] & & 0 \\ & [\underline{p}_3] & & \\ & & & \end{bmatrix}, \quad P_A = \begin{bmatrix} & & & 0 \\ & & & \\ & & & \\ [\underline{p}_4] & & & \end{bmatrix}$$

$\begin{matrix} \uparrow & \uparrow & & \uparrow \\ 13 & 20 & & 4 \end{matrix}$

$$Q = \begin{bmatrix} [\underline{Q}_1] & & & 0 \\ & [\underline{Q}_2] & & \\ & & [\underline{Q}_3] & \\ 0 & & & [\underline{Q}_4] \end{bmatrix}$$

$(L \times L)$

$$G = \begin{bmatrix} & s_1 & \\ 0 & s_2 & \\ & s_3 & 0 \\ & 0 & \end{bmatrix}, \quad G_A = \begin{bmatrix} & & \\ 0 & & 0 \\ & s_4 & \\ \uparrow & & \\ 13 & 20 & 4 \end{bmatrix}$$

$$H = \begin{bmatrix} [h_1^T] & & & \\ & [h_2^T] & & 0 \\ & & [h_3^T] & \\ 0 & & & [h_4^T] \end{bmatrix}$$

Now we define the composite vector $\underline{x}_1$ with $(23+L)$ state variables

$$\underline{x}_1 = [\underline{x} \ \underline{z}]^T$$

and obtain the following equations which combine (3.2.18) to (3.2.21)

$$\dot{\underline{x}}_1 = A_1 \underline{x}_1 + B_1 \underline{x}_A + C_1 \underline{u} \quad (3.2.24)$$

$$0 = D_1 \underline{x}_1 + E_1 \underline{x}_A + F \underline{u} \quad (3.2.25)$$

where (we leave the matrix manipulations out)

$$A_1 = \begin{bmatrix} A+B_D G & B_D H \\ P & Q \end{bmatrix}, \quad B_1 = \begin{bmatrix} B \\ P_A \end{bmatrix}$$

$$C_1 = \begin{bmatrix} C \\ 0 \end{bmatrix}, \quad D_1 = \begin{bmatrix} D+E_D G & E_D H \end{bmatrix}$$

$$E_1 = E+E_D G_A$$

Finally, eliminating $\underline{x}_A$ from (3.2.24,25), the equivalent linear model X_1 is

$$\boxed{\dot{\underline{x}}_1(t) = A_s \underline{x}_1(t) + B_s \underline{u}(t)} \quad (3.2.26)$$

$$\text{where } A_s = A_1 - B_1 E_1^{-1} D_1$$

$$B_s = C_1 - B_1 E_1^{-1} F$$

Equation (3.2.26) is of course the same as eqn. (2.6) for the 1st order approximation which we have chosen for all 4 delays; $L = 4$ and therefore A_s, B_s are matrices of $(27 \times 27), (27 \times 7)$ respectively. The details will be found in the routine DELAY1 of MXEV2X1 (see section 5.2.3 and Appendix B).

3.2.3 Model X2 (linear, comprehensive)

As mentioned in Chapter 2 this final model incorporates a number of additional equations, involving new variables, inputs and outputs, equations which represent certain controls and the load following existing in the Winfrith simulator. These new equations have been incorporated into a separate routine, called FDBCK1 in the matrix evaluation program MXEV2X1 (see section 5.2). It may be asked why these equations have not been directly incorporated into the basic model or the linear versions of it. The reason is that the basic model represents the dynamics of the system without any controls or load following, and that those controls vary for different versions of the Winfrith simulator. Indeed, they may vary in the controls implementation of the reactor itself. Thus, while the basic dynamics is contained in the basic model and unaltered, the FDBCK1 routine allows us to alter controls and load following at will.

Thus it will be seen in section (5.2.3) that there are seven different versions of FDBCK1, numbered "Case" (1) to (7). The modifications are readily done, without upsetting the general structure. Case (7), it will be seen

later, corresponds to the model which has been set up on the Winfrith simulator with the sole purpose of checking the correctness of the model constructed in this study.

Case (3), on the other hand, corresponds to the model which was considered the most suitable to be used for calculating the "optimally controlled" system.

Model X2 is built up from Model X1 in the following way: starting with equation (3.2.26)

$$\dot{\underline{x}}_1(t) = A_{s-1} \underline{x}_1(t) + B_{s-1} \underline{u}(t) \quad (3.2.26)$$

The newly introduced equations are

$$\underline{y}(t) = C_{s-1} \underline{x}_1(t) \quad (3.2.27)$$

$$\dot{\underline{w}}(t) = R_w \underline{w}(t) + S_y \underline{y}(t) + D_{d-2} \underline{u}_2(t) \quad (3.2.28)$$

$$\underline{u}_2(t) = T_w \underline{w}(t) + V_y \underline{y}(t) + E_e \underline{u}_b \quad (3.2.29)$$

where $\underline{y} \equiv$ vector of n_y outputs (6 for cases (1) to (7))

$\underline{w} \equiv$ vector of n_w new state variables (3 for case (3))

$\underline{u}_n \equiv$ vector of n_{un} newly introduced inputs

(3 for cases (1) to (7))

$\underline{u}_2 \equiv$ vector of n_{u2} inputs (10 for cases (1) to (7))

$$\underline{u}_2 = [\underline{u} \ \underline{u}_n]^T, \quad n_{u2} = 7 + n_{un}$$

$\underline{u}_b \equiv$ vector of n_{u2} input reference points for $\underline{u}_2$ or additional control laws to be found in subsequent computations. (10 for cases (1) to (7)).

$C_s \equiv$ matrix of n_y rows (6 for cases (1) to (7))

$R, S, D_d \equiv$ matrix of n_w rows (3 for case 3)

$T, V, E_e \equiv$ matrix of n_{u2} rows (10 for cases (1) to (7))

Now incorporate the new inputs into model X1. From (3.2.26)

$$\begin{aligned} \dot{\underline{x}}_1 &= A_{s-1} \underline{x}_1 + B_{s-1} \underline{u} + B_{n-n} \underline{u}_n \\ &= A_{s-1} \underline{x}_1 + B_{sn} \underline{u}_2 \end{aligned} \quad (3.2.30)$$

where $B_{sn} = [B_s \ B_n]$, $B_n = [0]$

$B_n \equiv$ matrix of $(23+L)$ rows

Replacing (3.2.27,29) into (3.2.28,30)

$$\dot{\underline{x}}_1 = [A_s + B_{sn} VC_s] \underline{x}_1 + B_{sn} T \underline{w} + B_{sn} E \underline{u}_b \quad (3.2.31)$$

$$\dot{\underline{w}} = [SC_s + D_d VC_s] \underline{x}_1 + [R + D_d T] \underline{w} + D_d E \underline{u}_b \quad (3.2.32)$$

i.e.

$$\boxed{\dot{\underline{x}}_2(t) = A_f \underline{x}_2(t) + B_f \underline{u}_b(t)} \quad (3.2.33)$$

where $\underline{x}_2 = [\underline{x}_1 \ \underline{w}]^T$

$\equiv$ vector of $(n_{x2} = 23+L+n_w)$ components

$(n_{x2} = 30 \text{ for case (3)})$

and

$A_f, B_f \equiv$ matrices of n_{x2} rows each (30 in case (3))

$$A_f = \begin{bmatrix} A_s + B_{sn} VC_s & B_{sn} T \\ SC_s + D_d VC_s & R + D_d T \end{bmatrix} \quad B_f = \begin{bmatrix} B_{sn} E \\ D_d E \end{bmatrix}$$

$(n_{x2} \times n_{x2})$ $(n_{x2} \times n_{u2})$

Equation (3.2.33) represents the final comprehensive model of the matrix evaluation program. It constitutes the data for the next program, (see section 3.4) which, as we shall see, aims at obtaining an "optimal" control law for the system. With that control law incorporated, a new system will emerge with a performance such as to satisfy the requirements of the specifications. The detailed equations for the different "cases" of FDBCK1 have not been written here. The list of additional parameters can be found in section (2.1) together with the equations for case (3) only.

Before finishing this section, it ought to be mentioned that the "correctness" of the linearization of the model, as well as all the manipulations conducing to the final comprehensive model, have been checked against the basic non-linear model. The details will not be explained here; they are in the routine TESTMX and TESTMX2 in MXEV2X1 (see section 5.2.3).

There are some further points which might as well be mentioned here, concerning the eigenvalues and the transients computations of the linear model:

- (i) The eigenvalues of model X1 spread over a range of 10^5 (from -30.0 to -0.0004) and are all negative except one value at the origin and a second one very near to it. This fact imprints its mark on all subsequent computations, since the precision required for reasonable results is much higher than it would be for eigenvalues of a similar order.
- (ii) It was thought to compute the transients of model X1 in order to see, first of all, if the model works and how the transients compare with the original Winfrith model. A straightforward way of doing this is by computing the state transition matrix, as explained in (3.2.1)

$$\phi(t) = e^{A^T s} \quad (3.2.34)$$

but of course it was soon realised that the two eigenvalues at the origin cause the system to be unstable. The reason is that in the Jordan canonical form elements of the type

$$\begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

appear, which increase with time^(*).

- (iii) The conclusion of (ii) was an additional reason, among other reasons given previously, to incorporate FDBCK1 into the program, and so avoid having 2 repeated eigenvalues at the origin. In physical terms, this means that the loop of the feed-water circulation (which when open represented a missing column in the matrix A_s of model X1) was closed.
- (iv) It was decided to calculate eigenvectors as well, for the dual purpose of using them for transients computation and for the solution of the matrix Riccati equation, subjects to be dealt with at greater length in sections (3.3), (3.4), (3.6). The routine used in MXEV2X1 for eigenvalues/eigenvectors, employing a modified SHARE library program, was not entirely satisfactory, particularly the computation of eigenvectors, for which great precision was needed. A new routine based on Wilkinson's inverse iteration method^[16] was then used, with the added bonus of higher precision for the eigenvalues.

(*) It is easily shown that ^[15] if $J = \begin{bmatrix} \lambda & 1 & 0 \\ & \lambda & 1 & 0 \\ & 0 & & \lambda \end{bmatrix}$ then

$e^{Jt} = e^{\lambda t} \begin{bmatrix} 1 & t & t^2/2! \\ & 1 & t \\ & 0 & 1 \end{bmatrix}$ and for second order $e^{Jt} = e^{\lambda t} \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$

Therefore, for $\lambda = 0$ we have

$$e^{Jt} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

The whole problem of eigenvalues/eigenvectors will be described in detail in section 3.3, but it is mentioned here in order to explain why there are two different eigenvalues-routines in the programs of this study. MXEV2X1 (construction of the linear model, see section 5.2) uses the "old" routine, which fails for certain conditions. On the other hand, REGULA (optimal regulator problem, see section 5.3) and SIMULA and SIMSAT (transients computation, see sections 5.4 and 5.5) use the "new" routine. The first one of the three programs mentioned was written earlier and it was not considered necessary to change it, in view of the fact that the eigenvalues and eigenvectors must be computed anyway in the last two ones, hence only the eigenvalues of A_s (i.e. model X1) are computed in MXEV2X1, for the purpose of checking the model.

3.3 The eigenvalues/eigenvectors problem

3.3.1 Motivation

The computation of eigenvalues/eigenvectors is necessary for two main reasons: transients computation and the solution of the optimal regulator problem. Also, it may be stated rather loosely that the eigenvalues can tell us a good deal about a dynamical system.

- (i) Transients computation: it may be argued that by calculating the state transition matrix $\phi(t)$ with any of the routines available for series expansion, no eigenvalues/eigenvectors are needed. While this is correct, it is also true that care has to be taken to check convergence and set the right number of

terms for different time intervals to obtain reasonable accuracy. Different systems resulting from different control laws obtained by "optimal" control, must then be adjusted each time^(*).

On the other hand, if a good eigenvalues/eigenvectors routine is available, the transients computation becomes a mere algebraic problem:

$$e^{At} = We^{\Lambda t}V \quad (3.3.1)$$

where

$\Lambda \equiv$ diagonalized transform of matrix A

$W \equiv$ eigenvector matrix of A

$V \equiv W^{-1}$

W, V are computed once and equation (3.3.1) implemented for any desired t .

(ii) Solution of the matrix Riccati equation:

section (3.4) will deal at length with this question.

Briefly, the complete steady state equation is

$$0 = Q + A^T P + PA - PBR^{-1}B^T P \quad (3.3.2)$$

while the linear or "Liapunov" part of it is

$$0 = Q + A^T P + PA \quad (3.3.3)$$

There are, generally speaking, two numerical approaches for solving (3.3.2):

- (a) Solve the whole equation by eigenvector/eigenvalue methods or by integration
- (b) Solve (3.3.3) and iterate by various methods.

(*) It should be added here that there exist algorithms such as MEXP7T (Lund)^[17] which according to its author's claim, avoid that pitfall by scaling the matrix and scaling and storing the intermediate values in such a way that the number of terms is fixed and the accuracy reportedly good.

For solving (3.3.3) it is possible to apply straight algebraic solution of the set of linear equations, use eigenvectors/eigenvalues methods, integrate or use other special methods for special circumstances. Integration is a particularly widely used method. As for (a), if a reliable "eigen" routine is available, and if computer storage is not a too stringent limitation, this is the most straightforward way of solving (3.3.2).

3.3.2 The old routine

The first routine used (let it be called the "old EIGSYS") was a modified version of EIGSYS,^[18] which in turn is a modified version of the original EIGSYS employing SHARE library programs numbered 300601 and 3385. It consists of a number of subroutines which will be described in section (5.2.3). In it all the eigenvalues are calculated and optionally, column eigenvectors of a square symmetrical or non-symmetrical matrix with real elements. The eigenvalues are found using the QR transform method^[19]. The eigenvectors are found by solving the resulting sets of homogeneous simultaneous equations.

For the routine to be useful, it has to fulfil two conditions:

- (i) All eigenvalues must be computed (self-evident)
- (ii) The precision of the computed eigenvectors as well as of the eigenvalues must be good enough so that the following equation holds:

$$A = W \Lambda V \quad (3.3.4)$$

By "good enough" it is meant that all elements of the matrices on both sides of equation (3.3.4) should be equal within several decimal places.

In some cases of very ill-conditioned matrices, condition (i) was not fulfilled. Before explaining what was done to get the routine working, it might be useful to outline briefly the fundamentals of the QR method and the steps preceding it:

Before applying the QR-transform, the A matrix is transformed into upper Hessenberg form, that is all the elements below the first subdiagonal are zero, using a series of similarity transforms: [20,21]:

```

X  X  X  X  X  X
X  X  X  X  X  X
0  X  X  X  X  X
0  0  X  X  X  X
0  0  0  X  X  X
0  0  0  0  X  X

```

The purpose of the QR-transform is to cause elements on the subdiagonal of the upper Hessenberg matrix to vanish. If the upper Hessenberg matrix can be transformed into the form

```

X  X  X  X  X  X
X  X  X  X  X  X
0  X  X  X  X  X
0  0  X  X  X  X
0  0  0  X  X  X
0  0  0  0  0  α

```

then the matrix has a real eigenvalue of α and the last row and column can be dropped, reducing the order of the

matrix by one. On the other hand, if the upper Hessenberg matrix is transformed into the form

```

X X X X X X
X X X X X X
0 X X X X X
0 0 X X X X
0 0 0 0 X X
0 0 0 0 X X

```

then a pair of eigenvalues, possibly complex, can be found directly from the lower right 2x2 submatrix, and the order of the original matrix reduced by two. If any element on the subdiagonal is zero, the matrix is divided into two submatrices and these handled independently. This feature makes the method very powerful with large systems by breaking it up into smaller matrices, but it becomes a failure when confronted with some peculiar matrices, as for instance when the uppermost element in the first subdiagonal is zero and the lower right 2x2 submatrix has an incomplete diagonal, for instance

```

0 0 0 0 0 X
0 X X X X 0
0 X X X X 0
0 0 X X X 0
0 0 0 X X 0
0 0 0 0 X 0

```

In such cases the problem is overcome by reordering rows and columns in such a way that the lower right 2x2 submatrix is full or has at least the two diagonal elements. This was done successfully in the work described in this

thesis with a new subroutine REORDR. The program works in such a way that once the lower right 2×2 submatrix has been handled, the algorithm keeps "climbing up" the sub-diagonal, undisturbed at finding additional zeros on the diagonal in its way.

Condition (ii) is more tricky, since it deals with precision. The results of the computation of equation (3.3.4) yielded elements on both sides of it which are equal down to at least 5 decimal figures, usually at least 8. However there is a snag in this "precision". When the elements of the left hand side are 0, it means that the R.H. elements may be as high as 10^{-5} . This number has to be compared with significant (i.e. non-zero) elements in the same row and columns where it occurs. It turned out that with the exception of one row, all the left-hand zeros had corresponding "good" zeros on the right hand. But one row had most L.H. zeros corresponding to numbers of various orders on the R.H., the highest being 10^{-5} , only one order smaller than the smallest "non-zero" in the same row. The reason for this imprecision, so it seemed by inspection, is that W contains many eigenvectors which are very close (in the space) to each other, nearly equal. The probable reason for this fact, in turn, is the wide spread of eigenvalues of the system.

It is noteworthy that 3 different approaches to the construction of W and V in equation (3.3.4) produced similar results.

- (a) Real matrix associated with a block-diagonalized λ .
- (b) Same as (a) using double precision.

(c) Complex matrix with diagonal λ . For this purpose an inversion routine for complex matrices was written as a modified version of the routine for real matrices (based on SHARE library number SDA3180) in the Control Section Library of C.C.D., Imperial College.

A few words should be said at this stage on the construction of a real eigenvector matrix associated with a real block diagonalized transform of A:

W contains complex conjugate vectors if Λ contains complex conjugate values.

$$\text{Let } \Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_n \end{bmatrix} \quad W = \begin{bmatrix} \underline{w}_1 & \underline{w}_2 & \dots & \underline{w}_n \end{bmatrix} \quad (3.3.5)$$

Suppose that λ_1, λ_2 and therefore $\underline{w}_1, \underline{w}_2$ are complex conjugate (the conclusions we shall arrive at apply of course to matrices where several or all of the eigenvalues are paired into complex conjugates).

Then

$$\begin{aligned} \lambda_1 &= \alpha_1 + j \beta_1 \\ \lambda_2 &= \alpha_1 - j \beta_1 \end{aligned} \quad (3.3.6)$$

and

$$\begin{aligned} \underline{w}_1 &= \underline{r}_1 + j \underline{i}_1 \\ \underline{w}_2 &= \underline{r}_1 - j \underline{i}_1 \end{aligned} \quad (3.3.7)$$

It can be shown^[22] that, constructing Λ_r and W_r in the following way

$$\Lambda_r = \begin{bmatrix} \alpha_1 & -\beta_1 & & 0 \\ \beta_1 & \alpha_1 & & \\ & & \lambda_3 & \\ 0 & & & \lambda_n \end{bmatrix} \quad W_r = \begin{bmatrix} \underline{r}_1 & \underline{i}_1 & \underline{w}_3 & \dots & \underline{w}_n \end{bmatrix} \quad (3.3.8)$$

Then

$$A = W_r \Lambda_r V_r \quad (3.3.9)$$

where $V_r = W_r^{-1}$

Equation (3.3.9) is equivalent to (3.3.4) and all the usual analytical operations on A , such as equation (3.3.1), may be performed with the real rather than with the complex matrices.

3.3.3 The new routine

Due to the nonfulfillment of conditions (i) and (ii) in section 3.3.2 a new routine was looked into and the inverse iteration method adopted^[16] for the computation of the eigenvectors. This method gives far more precise eigenvectors and also more precise eigenvalues as a by-product.

The inverse iteration method is based on the fact that the eigenvalues calculated by any numerical method, including the powerful QR transformation, are not, in general, accurate. Indeed, when they are, the inv. iteration method must be bypassed and the set of homogeneous linear equations

$$[A - I \lambda_i] \underline{w}_i = 0 \quad (3.3.10)$$

solved, where the matrix in brackets is singular (and therefore the magnitude of the eigenvectors free to choose from).

If the calculated eigenvalue p is close enough to λ_i , but not equal, then the iteration for eigenvector $\underline{w}_i$ can be summarized by the equations

$$[A - Ip] \underline{x}_{s+1} = \underline{z}_s \quad (3.3.11)$$

$$\underline{z}_{s+1} = \underline{x}_{s+1} / \max(x_{s+1}) \quad (3.3.12)$$

where $\underline{z}_0$, the initial value of $\underline{z}$, is an arbitrary vector, usually chosen to be all ones, while $\max(x_{s+1})$ is the element of $\underline{x}_{s+1}$ with the largest magnitude. The iteration process continues until the change in $\underline{z}$ is less than a prescribed value. To examine this process, expand $\underline{z}_0$ in terms of the precise eigenvectors of A

$$\underline{z}_0 = \sum_i c_i \underline{w}_i \quad (3.3.13)$$

After s iterations, and neglecting the normalization process of equation (3.3.12)

$$\underline{z}_s = \sum_i \frac{c_i \underline{w}_i}{(\lambda_i - p)^s} \quad (3.3.14)$$

(this is not difficult to prove with some matrix manipulations). From equation (3.3.14) it is clear that $\underline{w}_i$ will predominate in the sum if p is close enough to λ_i , hence $\underline{z}_s$ is the desired eigenvector $\underline{w}_i$.

It is seen that the accuracy of $\underline{w}_i$ does not depend on the accuracy of p with respect to λ_i . On the other hand, with eigenvectors as accurate as wished, the eigenvalue p calculated in the first place can be made as accurate as wished by the method of the Rayleigh quotient^[16], which can be described by the equations (to be iterated)

$$p_{s+1} = p_s + \underline{v}^T [A - I p_s] \underline{w} / \underline{v}^T \underline{w} \quad (3.3.15)$$

where p_s is the current estimate of the eigenvalue and $\underline{w}$, $\underline{v}$ are respectively the eigenvectors of A, A^T .

The new program which will be described later (see section 5.3.3) employs the inverse iteration method and the Rayleigh quotient and was obtained from Prof. J.E. Van Ness^[23], to whom this author is indebted. The original

program is also called EIGSYS^(*) (let the new program be called "new" EIGSYS to distinguish it from the old EIGSYS) and after some modification by this author it is called EIGVVI.

In this program the precision of the eigenvectors obtained is at least two orders of magnitude better than that obtained with the old EIGSYS. The program did not fail for any of the many cases in which the old EIGSYS did. Two kinds of failures must be mentioned here:

- (i) The failure of computing eigenvalues. This problem had been overcome when still using the old EIGSYS, with the reordering operation mentioned in par.(3.3.2).
- (ii) The eigenvectors could not be calculated with the old EIGSYS for many matrices, due to their near-singularity. The new EIGSYS worked, as mentioned before. In fairness, though, it should be added that in a small number of cases (out of hundreds computed) the program failed to compute correct eigenvectors when some eigenvalues were extremely close to each other. Of course a proper message was then printed. The reason for the failure is plain from eq.(3.3.14).

3.4 Optimal Regulator Problem. Performance criteria.

Constraints.

In section 3 (iii) an outline of the optimal regulator problem was given, i.e. a feedback controller designed to keep a stationary system within an acceptable deviation from a reference condition using acceptable amounts of control.

(*) It was converted for the CDC-6400 by L.H. Weiner, Vogelback Computing Center, Northwestern University, Evanston, Illinois.

The problem is tackled by solving the steady state Matrix Riccati equation.

Let us say a few words on the differential Riccati equation:

The basic, general problem of optimal control may be defined as the minimisation of a performance index

$$J = \int_{t_0}^{t_f} L(\underline{x}, \underline{u}, t) dt \quad (3.4.1)$$

for a system described by

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}, t) \quad (3.4.2)$$

where both $\underline{f}$ and L are non-linear and time-varying. The control law obtained will be

$$\underline{u}^0 = \underline{u}^0(\underline{x}, t) \quad (3.4.3)$$

One approach for dealing with this problem is the Hamilton Jacobi approach^[24] which will not be described here except by stating that it involves the solution of a partial differential equation known as the "Hamilton-Jacobi" equation, which it is almost impossible to solve except for some special or trivial cases.

But for a linear system with a quadratic performance index which can be described by the following equations

$$\dot{\underline{x}}(t) = A(t) \underline{x}(t) + B(t) \underline{u}(t) \quad (3.4.4)$$

$$J = \frac{1}{2} \underline{x}_f^T Q_f \underline{x}_f + \frac{1}{2} \int_{t_0}^{t_f} (\underline{x}^T Q \underline{x} + \underline{u}^T R \underline{u}) dt \quad (3.4.5)$$

where $Q(t) \equiv$ Positive semidefinite cost matrix
penalizing the state $\underline{x}$

$Q_f \equiv$ Positive semidefinite cost matrix
penalizing the final state $\underline{x}_f$

$R(t) \equiv$ Positive definite cost matrix

penalizing the control $\underline{u}$

the Hamilton-Jacobi approach is a way to lead us to the general Matrix Riccati equation [22]

$$\dot{P}(t) = -PA - A^T P + PBR^{-1}B^T P - Q \quad (3.4.6)$$

$$P(t_f) = Q_f$$

$P(t)$ is obtained by integrating equation (3.4.6) backwards in time from the known terminal condition over the time interval of interest. It can be shown that $P(t)$ must be a positive definite, symmetric matrix. It can also be shown that the optimal control law is linear and given by

$$\underline{u}^0(t) = -K(t) \underline{x}(t) \quad (3.4.7)$$

where $K(t)$ is a time-varying feedback gain matrix given by

$$K(t) = R^{-1}(t)B^T(t)P(t) \quad (3.4.8)$$

Furthermore it can be shown [11] that the minimal performance index at any time is

$$J^0 = \frac{1}{2} \underline{x}^T P(t) \underline{x} \quad (3.4.9)$$

For a stationary system, i.e. A, B constant and if the matrices Q, R are constant, consider the case $t_f - t_0 \rightarrow \infty$. Examining the Riccati equation (3.4.6) it is possible that a steady state, finite solution exists as

$$\dot{P} = 0 = -PA - A^T P + PBR^{-1}B^T P - Q \quad (3.4.10)$$

implies that $P(t) \rightarrow P^0$ as $t_f \rightarrow \infty$

with the conditions that the pair A, B is stabilizable and the pair A, C_q observable, where $C_q^T C_q = Q$.

If so, the feedback gain matrix of equation (3.4.8) is constant

$$K \Rightarrow K^0 = R^{-1}B^T P^0 \quad (3.4.11)$$

Furthermore, from equation (3.4.9) the minimal performance index at time t_0 is

$$J^0(t_0) = \frac{1}{2} \underline{x}^T(t_0) P^0 \underline{x}(t_0) \text{ as } t_f \rightarrow \infty \quad (3.4.12)$$

which is independent of time. Thus if a steady-state finite solution P^0 exists and is positive definite, it follows that $\underline{x}(t)$ and $\underline{u}(t)$ are bounded and

$$\underline{u}^0(t) = -K^0 \underline{x}(t) \quad (3.4.13)$$

is an asymptotically stable regulator [11]

Equation (3.4.10) is sometimes called the "reduced" or "degenerate" Riccati equation, or just "the Riccati equation".

Our chief problem in the context of this section was to select a reliable method for solving (3.4.10) where A, B are the matrices A_f, B_f from equation (3.2.33) and Q, R are matrices set at will in order to satisfy certain performance criteria outlined broadly in section (1.3). Of course the proof of "goodness" of the "optimal" control law thus found, can be obtained by simulating the new system, i.e. computing its transients.

If such a reliable method could be found, it would be possible to feed the program (REGULA, see section 5.3) with different elements for the cost matrices Q, R and simulate the resulting "optimal" system (SIMULA, see section 5.4), all on a trial-and-error basis.

A good review on the main methods available has been written by Hagander [25]. Although it concentrates on the "Liapunov" equation,

$$Q + A^T P + P A = 0 \quad (3.4.14)$$

which is the linear part of the Riccati equation:

- (i) many conclusions can be extended to eqn. (3.4.10)
- (ii) It can be solved and iterated to satisfy equation (3.4.10).

Broadly speaking, there are algebraic methods and integration algorithms or a combination of both. The algebraic methods generally need much more store than the integration routines, in particular the straightforward solution of the set of linear equations (3.4.14) which, once obtained, still has to be iterated to solve the non-linear Riccati equation^(*).

The application of eigenvector/eigenvalue techniques to solve the Liapunov equation or directly the Riccati equation, takes up less store but carries many problems typical of this type of computations. Therefore it is not particularly recommended by Hagander, but in our own experience it proved to be a fairly reliable method, if time-expensive. Since it is the one we adopted, it will be discussed later, after finishing this brief review.

Apart from the eigenvalues/eigenvectors method the only other approach known to this author for solving the Riccati equation directly, i.e. without breaking it into its linear and non-linear components, is to integrate it backwards with boundary conditions $P(t_f) = 0$ until $\dot{P} \approx 0$. In this case Kalman^[26] has shown that $P(t) \rightarrow P^0$.

The remainder of the techniques solve the Liapunov equation and iterate, such as Kleinman's method^[27]. The Liapunov equation itself can be solved, as mentioned before, by algebraic methods or by integration, the latter having preference when computer storage is at a premium. The

(*) Hagander^[25] quotes 45K of store for a 20x20 matrix, plus a further 45K to improve precision.

integration method of solving equation (3.4.14) is based on the fact^[28] that the steady state solution to the equation

$$\dot{P} = Q + A^T P + P A \quad (3.4.15)$$

is

$$P = \int_0^{\infty} e^{A^T t} Q e^{A t} dt \quad (3.4.16)$$

if A is stable. Among the many papers published on this integration Davison's^[29] and Hagander's^[25] algorithms will be mentioned, both based on an Euler approximation of the integral (3.4.16), Hagander's algorithm being a refinement of Davison's.

Equation (3.4.16) can also be solved with eigenvectors/eigenvalues by using equation (3.3.1) i.e.

$$P = \int_0^{\infty} V^T e^{A^T t} W^T Q W e^{A t} V dt \quad (3.4.17)$$

Developing this equation, the result is

$$P = [p_{ij}] = -V^T \left[\frac{m_{ij}}{\lambda_i + \lambda_j} \right] V \quad (3.4.18)$$

where $[m_{ij}] = M = W^T Q W$

λ_i, λ_j = eigenvalues of A

There are other methods^[25] for solving the Liapunov equation for special circumstances.

The approach adopted in this work was the direct application of eigenvectors/eigenvalues to the Riccati equation. This technique was proposed by MacFarlane^[30], Potter^[31] and extended by Vaughan^[32], Fath^[33]. Potter's paper is based on three simple theorems, the proof of which will not be given here (also, real matrices only will be dealt with here, unlike Potter's treatment).

Consider the Riccati equation

$$0 = Q + A^T P + PA - PBR^{-1}B^T P \quad (3.4.10)$$

where Q, A and $BR^{-1}B^T$ are $n \times n$ matrices.

Construct a matrix M ($2n \times 2n$)

$$M = \begin{bmatrix} A^T & Q \\ BR^{-1}B^T & -A \end{bmatrix} \quad (3.4.19)$$

Call the eigenvectors of M

$$\underline{a}_i = \begin{bmatrix} \underline{b}_i \\ \underline{c}_i \end{bmatrix}, \quad i = 1, \dots, 2n$$

where $\underline{b}_i$ are the vectors of the upper n elements

and $\underline{c}_i$ are the vectors of the lower n elements

Theorem (i) : every solution of (3.4.10) has the form

$$P = [b_1 \dots b_n][c_1 \dots c_n]^{-1} \quad (3.4.20)$$

so, if $[c_1 \dots c_n]$ is nonsingular, P of eqn.(3.4.20)

satisfies eqn. (3.4.10).

Theorem (ii) : if Q and $BR^{-1}B^T$ are symmetric then $a_1, \dots, a_n$ are eigenvectors of M corresponding to eigenvalues $\lambda_1, \dots, \lambda_n$ such that $\lambda_i \neq -\lambda_j^*$ (conjugate) and if $[c_1 \dots c_n]$ is nonsingular, P of eqn. (3.4.20) is symmetric.

Theorem (iii) : if Q and $BR^{-1}B^T$ are positive semidefinite symmetric and $a_1, \dots, a_n$ are eigenvectors of M corresponding to $\lambda_1, \dots, \lambda_n$ and these $\lambda_1, \dots, \lambda_n$ have all positive real parts and $[c_1 \dots c_n]$ is nonsingular, then P of eqn.(3.4.20) is positive semidefinite.

The 3 theorems above, coupled with the fact that it can be shown^[34] that the complex eigenvalues of M are

symmetric with respect to the real and imaginary axes of the complex plane (the real eigenvalues are symmetric w.r.t. the imag. axis only), suggests the following procedure which was followed in a straightforward manner^(*):

3.4.1 Algorithm for solving the optimal regulator problem

- (i) Construct M (sometimes called the Hamiltonian or state-costate matrix) crisscrossing the submatrices of equation (3.4.19) and giving the opposite sign.

$$M = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \quad (3.4.21)$$

- (ii) Calculate the eigenvectors of M. Precisely one half of them should correspond to stable eigenvalues.
- (iii) Reorder the eigenvector matrix thus obtained into four equally dimensioned submatrices such that

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \quad (3.4.22)$$

where the columns of W_{11}, W_{21} are the eigenvectors corresponding to the stable eigenvalues.

- (iv) Compute (if W_{11} is nonsingular) the Riccati matrix P

$$P = W_{21} W_{11}^{-1} \quad (3.4.23)$$

(*) Fath's^[33] procedure does not require working with complex eigenvector matrices. Such a technique would have been preferable, but since a program could not be obtained at the time, this author wrote his own program, less complicated but which involves complex arithmetic.

- (v) Check if P fulfils each of the following properties, within reasonable bounds of numerical accuracy
- (a) P should be real
 - (b) P should be symmetric
 - (c) P should be positive definite
 - (d) P should satisfy equation (3.4.10)
- (vi) Compute the feedback gain matrix

$$K = R^{-1} B^T P \quad (3.4.11)$$

to be inserted into the optimal control law

$$\underline{u}^0(t) = -K \underline{x}(t) \quad (3.4.13)$$

- (vii) Since the system is time-invariant, compute the "optimally controlled" state matrix from equations (3.4.4), (3.4.11) and (3.4.13)

$$A_{oc} = A - BR^{-1}B^TP \quad (3.4.24)$$

The matrices A_{oc} and K are fed into SIMULA (see section 5.4) for the purpose of computing the transients of $\underline{x}$ and $\underline{u}$, evaluate them and test if they violate any constraints.

Before solving the matrix Riccati equation, two steps have to be taken

- (a) Set Q and R of eqn. (3.4.5)
- (b) Test if the system of eqn. (3.4.4) is completely controllable, and if the pair A, C_q is completely observable, where

$$C_q^T C_q = Q \quad (3.4.25)$$

These are conditions^[35] for obtaining an optimal control law via the Riccati equation.

(a) The question of selecting appropriate values for Q and R is not trivial, and goes to the core of the optimal regulator problem. Which are the state variables to be penalized through Q and by what amount? By what amount are the control variables to be penalized through R ?

The second question is relatively easier than the first, because R should be such as to have an acceptable amount of control, i.e. u must not violate any constraints. The first question is more difficult, because it is necessary to start selecting which are the variables that merit any penalizing at all, and it seems that if fewer variables are selected, then fewer conflicts will arise in the "tug of war" which the solution of the Riccati equation generates. By "tug of war" it is meant that by penalizing x_A it is possible to cause x_B to become larger than it was expected before penalizing x_A . Also, by penalizing x_A and x_B , the result may be that neither will be substantially reduced. All this will be discussed in more detail in dealing with results (see Chapter 6), but it should be stated here that the selection of Q and R was done, first, by way of "educated guesses" and later by comparing the results obtained with the transients computation.

Many runs of the program REGULA (see section 5.3) were done and some results as well as their relative merits will be discussed in Chapter 6. Suffice it here to state that three stages could be discerned in the series of runs performed.

(a.i) It was decided that rather than penalizing x it

would be more reasonable to penalize $\underline{y}$ (see eqn. (3.2.27) where $\underline{y}$ are the outputs (measurable) encountered in the basic model X and the additional FDBCK1 routine.

i.e.

$$\underline{y} = C \underline{x} \quad (3.4.26)$$

and minimize the performance index

$$J = \frac{1}{2} \int_0^{\infty} [\underline{y}^T Q_y \underline{y} + \underline{u}^T R \underline{u}] dt \quad (3.4.27)$$

rather than the original equation (3.4.5). Expanding eqn. (3.4.27) with eqn. (3.4.26) and replacing in eqn. (3.4.5) we arrive at

$$Q = C^T Q_y C \quad (3.4.28)$$

In this case, and if Q_y is a diagonal matrix (which seems the easiest and most sensible way of constructing it), then, from eqn. (3.4.25) it is seen that the pair of matrices which has to undergo the observability test is A, C_q where

$$C_q = Q_y^{\frac{1}{2}} \cdot C \quad (3.4.29)$$

- (a.ii) It was found that on many runs of REGULA (in which each "case" is characterized by a different set of Q and R) matrix W_{11} of eqn. (3.4.22) was singular, hence the Riccati equation could not be solved. The main cause of the inclination to singularity of W_{11} (and of course also W_{21} and the rest of W) is obviously the wide spread in eigenvalues, which makes several groups of eigenvectors have nearly the same direction. In view of this, it was decided to try to penalize $\underline{x}$ directly, so as to be able to choose exactly which

variables are to be penalized and by what amount. Still, only measurable state variables were penalized. The number of "cases" in which W_{11} was singular dropped. In order not to change the routine, the arrangement of (a.i) was kept, but new outputs y defined such that there is a one-to-one relationship between each output variable and the state-variable penalized.

(a.iii) Although the following explanation should really be given in the program's description (Chapter 5), it ought to be noted here that even with the arrangement of (a.ii) there were still "cases" of singularity of W_{11} . This was irritating because in the process of searching for a better control, certain cases had to be dropped not because the results were going in the wrong direction but because the computation tool did not work.

It was at this stage that a new inversion routine was utilized, instead of the old routine (which was a modified version for complex matrices of the routine for real matrices in the Control Section library of I.C., which in turn is a much modified version of the SDA3180 SHARE library routine). The new routine INCOMA is based on converting a complex $n \times n$ matrix $C = R + jI$ into a real $2n \times 2n$ matrix CR , where

$$CR = \begin{bmatrix} R & -I \\ I & R \end{bmatrix} \quad (3.4.30)$$

Inverting CR and calling $V = CR^{-1}$

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \quad (3.4.31)$$

It can be shown that (*)

$$C^{-1} = V_{11} + jV_{21} \quad (3.4.32)$$

This program was taken from the CERN library [36].

(b) Controllability and observability:

We used here a tool which is, admittedly, stronger than needed, namely the diagonalization of the matrix A of eqn. (3.4.4) and the computation of eigenvectors. It was done so because a good, reliable eigenvalues/eigenvectors routine existed in the program anyway and so further store space was saved, as well as programming effort.

Controllability has to be tested once only for the system, since the different settings of Q and R do not affect it, as can be seen from the equations below. Observability, on the other hand, is affected by Q, but it is easily seen how it changes with variations in Q, and therefore it is not necessary to test it on each run.

In short, if the system

$$\dot{\underline{x}} = A \underline{x} + B \underline{u} \quad (3.4.33)$$

$$\underline{y}_q = C_q \underline{x} \quad (3.4.34)$$

(*) To find the inverse of a matrix C means that given

$$C \underline{x} = \underline{b} \quad (3.4.37)$$

Find C^{-1} such that $\underline{x} = C^{-1} \underline{b}$

Expanding (3.4.37) in its real and imaginary terms we arrive at

$$CR \begin{bmatrix} R(\underline{x}) \\ I(\underline{x}) \end{bmatrix} = \begin{bmatrix} R(\underline{b}) \\ I(\underline{b}) \end{bmatrix} \quad (3.4.38)$$

i.e.

$$\begin{bmatrix} R(\underline{x}) \\ I(\underline{x}) \end{bmatrix} = V \begin{bmatrix} R(\underline{b}) \\ I(\underline{b}) \end{bmatrix} \quad (3.4.39)$$

where CR and V are described in eqn.(3.4.30), (3.4.31). Expanding now eqn.(3.4.39) and with some manipulation we arrive at eqn.(3.4.32).

has to be tested for controllability, observability^(*), then if transformed into the canonical system

$$\dot{\underline{z}} = \underline{\Lambda} \underline{z} + \underline{E} \underline{u} \quad (3.4.35)$$

$$\underline{y}_q = \underline{F}_q \underline{z} \quad (3.4.36)$$

where

$$\underline{\Lambda} = \underline{V} \underline{A} \underline{W}$$

$$\underline{z} = \underline{V} \underline{x}$$

$$\underline{E} = \underline{V} \underline{B}$$

$$\underline{F}_q = \underline{C}_q \underline{W}$$

$$\underline{W} = \text{eigenvector matrix of } \underline{A}$$

$$\underline{V} = \underline{W}^{-1}$$

The two conditions are:

1. The system is completely controllable if and only if $\underline{E}$ has at least one non-zero element in each row.
2. The system is completely observable if and only if $\underline{F}_q$ has at least one non-zero element in each column.

Dealing as we do with numerical computation, it has to be defined what is a non-zero. It was decided to consider 10^{-2} (rather loosely) as a non-zero element.

One further point before finishing this section. The model (X2) finally used has a vector $\underline{u}$ of 10 inputs. Among these, only two are used as control inputs and are fed by the "optimal" controller. Instead of breaking up the model into controlled and fixed inputs, the diagonal property of the cost matrix $\underline{R}$ (and therefore $\underline{R}^{-1}$) was used in such a way as to make the elements r_{dd}^{-1} equal to 0, where u_d are the fixed inputs. In these circumstances a proviso

(*) Note that eqn.(3.4.34) employs $\underline{y}_q$ and $\underline{C}_q$ as distinct from $\underline{y}$ and $\underline{C}$ in eqn.(3.4.26). This is because the observability condition mentioned in connection with eqn.(3.4.25) must take the cost matrix $\underline{Q}$ into account.

must be added to the condition (1) mentioned above, namely: the non-zero must be found in one of the two columns corresponding to the control inputs.

3.5 Optimal Regulator for external disturbances

In the previous section the method used for designing the linear optimal regulator was described. It was mentioned that of the 10 available inputs, only two are used for controlling the system. The remainder may stay fixed, in which case they are taken care of by setting the input cost matrix R so that the corresponding elements r_{dd}^{-1} are nil. The controller will then be designed so as to bring the state $\underline{x}$ to zero from whatever initial excitation the system starts. However if one or more of the inputs u_d are used for external disturbance, the controller cannot just be left in the way described. It would obviously be incorrect to assign, say, a step excitation to any u_d without further changes, since from eqn.(3.4.11) the same u_d after applying the optimal control law would be nil, which it is not.

Various methods are available for dealing with external disturbances, notably the one [37] which introduces an additional component to the state vector, for each disturbance input. A new method was developed for the work described in this thesis. It will be described in detail in Chapter 4, but the final mathematical expressions derived will be written in this section so as to enable us to go on with the description of the methods of computation.

Salient features of the new method are:

- (a) The D.C. gain of the system is computed and controlled simultaneously with its dynamic behaviour.
- (b) In computing the transients, it is sufficient once only to calculate the transition matrix of the optimally controlled system. The expressions obtained are capable of accommodating any number of disturbances.

Without elaborating, the equations for the original system, the optimal control law and the optimally controlled system are, respectively

$$\dot{\underline{x}} = A\underline{x} + B_c \underline{u}_c + B_d \underline{u}_d \quad (3.5.1)$$

$$\begin{aligned} \underline{u}_c^o &= -K_r \underline{x} - K_{dg} \underline{u}_d \\ &= -K_r \underline{x} + \underline{k}_{du} \end{aligned} \quad (3.5.2)$$

$$\dot{\underline{x}} = A_{oc} \underline{x} + B_c \underline{k}_{du} + B_d \underline{u}_d \quad (3.5.3)$$

where

$\underline{u}_c \equiv$ vector of the control inputs

$\underline{u}_d \equiv$ vector of the disturbance inputs

$B_c \equiv$ driving matrix for the control inputs, a submatrix of B in the general linear system equation (3.4.4)

$B_d \equiv$ driving matrix for the disturbance inputs, a submatrix of B

$\underline{u}_c^o \equiv$ optimal control vector

$K_r \equiv$ feedback gain matrix from $\underline{x}$ to $\underline{u}_c$

$K_{dg} \equiv$ "disturbance gain" matrix from $\underline{u}_d$ to $\underline{u}_c$

$$\underline{k}_{du} \equiv -K_{dg} \underline{u}_d \quad (3.5.4)$$

$A_{oc} \equiv$ state matrix of the optimally controlled system

$$= A - B_c K_r \quad (3.5.5)$$

K_r is a submatrix of K from eqn.(3.4.11) repeated here

$$K = R^{-1} B^T P \quad (3.5.6)$$

K_r is the submatrix of K consisting of the rows of K corresponding to $\underline{u}_c$.

Let us repeat the equation of P as computed with the method of eigenvectors of the Hamiltonian matrix M of eqn.(3.4.21)

$$P = W_{21} W_{11}^{-1} \quad (3.5.7)$$

where the meaning of W_{21} , W_{11} can be seen in eqn.(3.4.22). It is easily seen that

$$K_r = R_c^{-1} B_c^T W_{21} W_{11}^{-1} \quad (3.5.8)$$

where R_c is the control cost matrix, a submatrix of the input cost matrix R with columns and rows corresponding to $\underline{u}_c$.

The expression for K_{dg} is

$$K_{dg} = K_{dr} B_d + [K_{dr} B_c - I] K_{cd} \quad (3.5.9)$$

where

$$K_{dr} = R_c^{-1} B_c^T (W_{11}^T)^{-1} \Lambda^{-1} W_{21}^T \quad (3.5.10)$$

$\Lambda \equiv$ diagonal matrix of the positive eigenvalues of the Hamiltonian matrix M of eqn.(3.4.21)

K_r and K_{dr} are computed in REGULA.

K_{cd} is the gain matrix from $\underline{u}_d$ to $\bar{\underline{u}}_c$, i.e.

$$\bar{\underline{u}}_c = K_{cd} \underline{u}_d \quad (3.5.11)$$

where $\bar{\underline{u}}_c$ is the settling-down value of $\underline{u}_c$ for a final off-set value $\underline{u}_d$. K_{cd} is set from values taken from the linearized model of the original system of equations.

Equations (3.5.2) and (3.5.3) will be used in SIMULA and SIMSAT, the programs for the transients computation

without and with constraints on the controls, respectively. The vector $\underline{k}_{du}$ of eqn.(3.5.4) may be considered as a "fixed added" control in the case of a step disturbance, or just as an additional control (computed with matrix K_{dg} which is unique to the controller) due, not to feedback from $\underline{x}$, but rather as a feedforward from $\underline{u}_d$. The block diagram of the optimal controller is seen in Fig. (3.1).

3.6 Free Transients Computation. Evaluation of the optimal regulator under no constraints.

As mentioned in the introduction to this chapter, the transients computation (or free simulation) is necessary in order to

- (a) assess the correctness of the linear model and
- (b) evaluate the quality of the "optimal" controllers computed in the manner outlined in sections (3.4) and (3.5) with the program REGULA (see 5.3).

The results will be discussed in Chapter 6.

The time response of a linear time invariant system

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) \quad (3.6.1)$$

for an initial time $t_i = 0$ and an initial state $\underline{x}(0) = \underline{x}_0$ is

$$\underline{x}(t) = e^{At} \underline{x}_0 + \int_0^t e^{A(t-\tau)} B \underline{u}(\tau) d\tau \quad (3.6.2)$$

For $\underline{u}$ constant this equation becomes

$$\underline{x}(t) = e^{At} \underline{x}_0 + \left[\int_0^t e^{A\tau} d\tau \right] B \underline{u} \quad (3.6.3)$$

$$= e^{At} \underline{x}_0 + A^{-1} [e^{At} - I] B \underline{u} \quad (3.6.4)$$

Equation (3.6.4) can be computed by transition matrix methods. On the grounds explained in section (3.3.1) it was decided

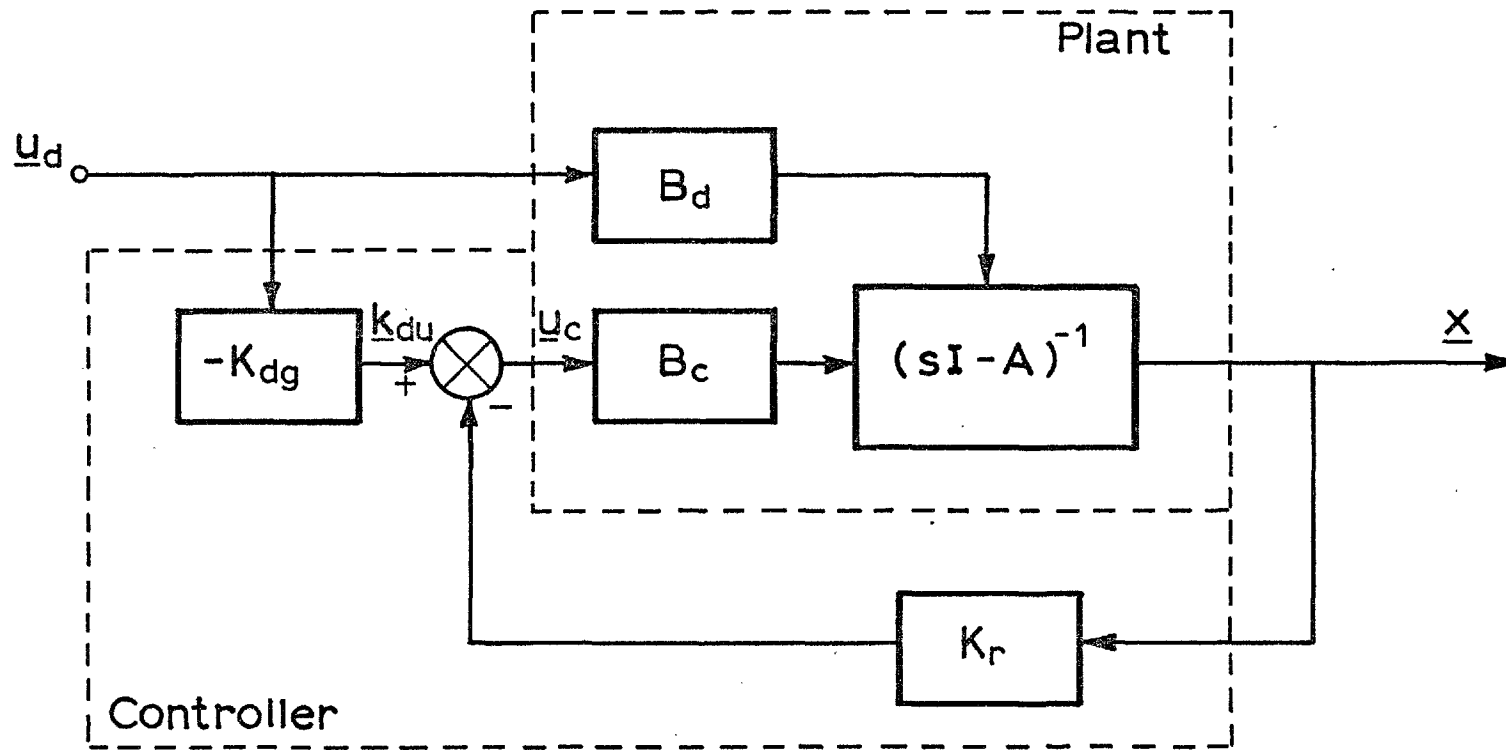


Fig.3.1 Block diagram of optimal controller for internal and external disturbances.

$\underline{u}_d$:vector of external disturbances

K_r :feedback gain matrix

K_{dg} :feedforward matrix

$\underline{k}_{du}$:correcting control vector (for external disturbances)

to use the algebraic eigenvalue/eigenvector method, hence

$$\underline{x}(t) = W e^{\Lambda t} V \underline{x}_0 + W \Lambda^{-1} [e^{\Lambda t} - I] V B \underline{u} \quad (3.6.5)$$

where Λ, W, V are the eigenvalues, the eigenvector matrix and its inverse, respectively.

Equation (3.6.5) is computed in the program SIMULA (see section 5.4) for various times, and graphically displayed. The vector $\underline{u}$ is considered constant because the excitations dealt with in this work are either state excitation ($\underline{x}_0 \neq [0]$) or step response to input $\underline{u}$. For responses other than step type (e.g. impulse or ramp) different equations must be established, but the kind of matrix manipulations involved are of a very similar nature. The program SIMULA can therefore easily be changed.

The system to be "optimally" controlled is the comprehensive model (X2) summarized in eqn.(3.2.33). In the previous section it was explained how the matrix B is broken up into two submatrices B_c and B_d , driving the control input vector and the disturbance input vector respectively. Let us repeat equations (3.5.1 to 3) for:

a. The uncontrolled system

$$\dot{\underline{x}} = A \underline{x} + B_c \underline{u}_c + B_d \underline{u}_d \quad (3.6.6)$$

b. The optimally controlled system

$$\dot{\underline{x}} = A_{oc} \underline{x} + B_c \underline{k}'_{du} + B_d \underline{u}_d \quad (3.6.7)$$

$$\underline{u}_c = -K_r \underline{x} + \underline{k}_{du} \quad (3.6.8)$$

The symbols are described in the previous section.

The basic simulation equation (3.6.5) thus becomes:

a. For the uncontrolled system:

$$\underline{x}(t) = W e^{\Lambda t} V \underline{x}_0 + W \Lambda^{-1} [e^{\Lambda t} - I] V [B_c \underline{u}_c + B_d \underline{u}_d] \quad (3.6.9)$$

b. For the optimally controlled system:

$$\underline{x}(t) = W_{oc} e^{A_{oc} t} V_{oc} \underline{x}_0 + W_{oc} A_{oc}^{-1} [e^{A_{oc} t} - I] V_{oc} [B_{c-du} + B_{d-du}] . \quad (3.6.10)$$

where the subscript (_{oc}) refers to the state matrix A_{oc} of the optimally controlled system.

Equation (3.6.9) computed over a number of time intervals t , is used in one run only for an uncontrolled system, whereas eqn. (3.6.10) is used in as many runs of SIMULA as there are "cases" of optimal control computed in REGULA. Details of the procedure will be given in the program's description (see 5.4), as well as means of graphical display, comparison of an original matrix A and the "reconstructed" matrix WAV , peak values of variables and how they approach or violate the constraints, enumeration of the variables which carry a dominant weight in the feedback to $\underline{u}_c$, etc.

In SIMULA the state variables $\underline{x}$ are computed, as well as the control variables $\underline{u}_c$ and some additional parameters the behaviour of which it is important to know. By inspecting the results of SIMULA for each "case" of REGULA it is possible to evaluate to what degree the computed regulator approaches the control objectives outlined in section (1.3), concerning time constants, overshoots or constraints.

3.7 Saturated transients computation. Constraints on the control inputs.

The free simulation of the system, uncontrolled or optimally controlled, as described in the previous section, allows control and state variables to move freely according

with the dynamics of the system. Whenever a parameter violates a constraint, a clear warning is printed and the designer subsequently changes the elements of the cost matrices Q , R for the next "case" of optimal regulator. In the real working of the plant, however, certain parameters cannot violate their constraints, due to physical limitations. These are the so called "hard" constraints^(*). Other constraints are only given by specifications from engineering and performance criteria, such as the limitations on frequency and pressure, for instance. Yet a third category of constraints, such as speeds of valve movements, are engineering constraints, which at least in theory could be changed by applying higher gains to amplifiers or other driving elements.

It was decided to construct a program of saturated simulation which actually implements the "hard" constraints. It turns out that of the 3 variables with such constraints, two are control variables, namely the inputs to the moderator and the turbine control valves. The third variable is the feedwater valve, which it was decided from the onset to leave with its existing conventional controller (see Fig. 1.3). As will be seen in the chapter on results, in none of the many cases run did this variable violate the constraint. From the above it is clear that the saturated simulation was to be computed with constraints on the control variables only. This decision was both practical and necessary: Practical because, as will immediately be described, the

(*) which include valves that cannot possibly open to more than 100% or close to less than 0% .

mathematical manipulations are much easier for constraints on the controls than for constraints on the state variables. Necessary because to constrain a variable which has no actual physical limitation does not correspond to the real operation of the plant. Again for example, the steam pressure may indeed violate the prescribed constraints and nothing will happen except that it would be unwise from the engineering point of view to let it be so. It should be added at this stage that there are a number of hard constraints among the state variables, such as the total mass of water in the drum or in the whole primary circuit, which obviously cannot become negative. But these constraints, as will be seen in the results, never come anywhere near to being violated. Hence the fortunate coincidence of having the real hard constraints on the control variables only.

The method for the constrained simulation is thus based on checking the controls $\underline{u}_c$ after each integrating interval: if one or both of the controls violates the constraints, it is set back at its limit (saturation point) and the feedback action on it made inoperative for the next interval. Since the state matrix of the controlled system is (eqn. 3.5.5)

$$A_{oc} = A - B_c K_r \quad (3.7.1)$$

it is clear that there will be four possible state matrices according to the saturation conditions. Also, $\underline{u}_c$ will assume 9 possible values, since the upper and lower constraints must be taken into account. Specifically, and starting from eqn.(3.5.3) for the free simulation

$$\dot{\underline{x}} = A_{oc} \underline{x} + B_{c-du} \underline{k} + B_d \underline{u}_d \quad (3.7.2)$$

$$= (A - B_c K_r) \underline{x} + B_{c-du} \underline{k} + B_d \underline{u}_d \quad (3.7.3)$$

Break up matrices B_c , K_r into vectors, and vector $\underline{k}_{du}$ into components, then:

For non-saturated case:

$$\dot{\underline{x}} = [A - \underline{b}_c^1 \underline{k}_r^1]^T \underline{x} + \underline{b}_c^1 \underline{k}_{du1} + \underline{b}_c^2 \underline{k}_{du2} + B_d \underline{u}_d \quad (3.7.4)$$

If u_{c2} saturates, call it $\tilde{u}_{c2}$, then

$$\dot{\underline{x}} = [A - \underline{b}_c^1 \underline{k}_r^1]^T \underline{x} + \underline{b}_c^1 \underline{k}_{du1} + \underline{b}_c^2 \tilde{u}_{c2} + B_d \underline{u}_d \quad (3.7.5)$$

If u_{c1} saturates

$$\dot{\underline{x}} = [A - \underline{b}_c^2 \underline{k}_r^2]^T \underline{x} + \underline{b}_c^2 \underline{k}_{du2} + \underline{b}_c^1 \tilde{u}_{c1} + B_d \underline{u}_d \quad (3.7.6)$$

If $\underline{u}_c$ saturates

$$\dot{\underline{x}} = A \underline{x} + B_{c-c} \tilde{\underline{u}}_c + B_d \underline{u}_d \quad (3.7.7)$$

Thus there are 4 possible A_{oc_l} and $\sum_l \underline{b}_c^i \underline{k}_{du_i}$

as well as 9 possible $\sum_q \underline{b}_c^j \tilde{u}_{c_j}$

The general expressions for eqs.(3.7.4) to (3.7.7) are:

$$A_{oc_l} = A - \sum_l \underline{b}_c^i \underline{k}_r^i{}^T \quad \left\{ \begin{array}{l} l = 1, \dots, 4 \\ q = 1, \dots, 9 \end{array} \right. \quad (3.7.8)$$

$$\dot{\underline{x}} = A_{oc_l} \underline{x} + \sum_l \underline{b}_c^i \underline{k}_{du_i} + \sum_q \underline{b}_c^j \tilde{u}_{c_j} + B_d \underline{u}_d \quad \left\{ \begin{array}{l} i, j = 1, 2 \\ i \neq j \end{array} \right. \quad (3.7.9)$$

A saturation code table for A_{oc_l} , $\sum_l$ and $\sum_q$ is printed on the program SIMSAT (see section 5.5). As the digital simulation is discrete, it is necessary to apply $\tilde{u}_c$ during a complete interval, regardless of the precise crossing point of u_c and the saturation line. Among several possible

approaches, the choice arrived at is to check the saturation status at the beginning of each time interval: accordingly, eqn.(3.7.9) is integrated over an interval $\delta t = \delta t(k)$ applying the appropriate $A_{oc\ell}$ and $\sum q$

$$\underline{x}_{k+1} = e^{A_{oc\ell} \delta t} \underline{x}_k + \left[\int_0^{\delta t} e^{A_{oc\ell} \tau} d\tau \right] \left[\sum_{\ell} \underline{b}_{-c}^i k_{du_i} + \sum_{q-c} \underline{b}_{-c}^j \tilde{u}_{c_j} + B_{d-d} \underline{u}_d \right] \quad (3.7.10)$$

Using eigenvalues/vectors Λ , W and inverses Λ^{-1} , V

$$\underline{x}_{k+1} = W_{\ell} e^{\Lambda_{\ell} \delta t} V_{\ell} \underline{x}_k + W_{\ell} \Lambda_{\ell}^{-1} [e^{\Lambda_{\ell} \delta t} - I] V_{bu_q} \quad (3.7.11)$$

where

$$V_{bu_q} = V_{\ell} \left[\sum_{\ell} \underline{b}_{-c}^i k_{du_i} + \sum_{q-c} \underline{b}_{-c}^j \tilde{u}_{c_j} + B_{d-d} \underline{u}_d \right] \quad (3.7.12)$$

There remain two observations by comparing eqs.(3.7.11) and (3.6.10) for saturated and free simulation, respectively:

- a. For the free simulation only a limited number of computations need be done, since the integration, which is done analytically, takes always $\underline{x}_0$ as the starting point. For the saturated simulation on the other hand, the integration must be done over each interval, due to the different saturation status possible for different intervals, and hence the need of using different integration formulae.

The smaller the interval, the more precise will the computation be, if during that interval the control crosses the saturation line. Hence the program takes much longer to execute.

- b. Some Λ_{ℓ} may contain zero eigenvalues (this is not so for the optimally controlled system) hence it is

necessary, when applying eqn.(3.7.11) to check if there is a $\lambda = 0$, in which case the factor

$$\lambda_i^{-1}(e^{\lambda_i \delta t} - 1) \text{ becomes } \delta t$$

3.8 Reduced set of feedback variables

Having designed an acceptable controller for the plant, it will be based on feedback from all the state variables. Not all of these variables are measurable, hence the need of applying techniques of deterministic state reconstruction of the observer type^[38] or estimation by stochastic filtering of the noise-disturbed plant^[39]. It was decided in this work to find out first which are the more important feedback variables i.e. what variables carry the dominant "weights" in the feedback control. The method of weight evaluation will be explained immediately. The point of the exercise was to see if the dominant variables are measurable, in which case the need for an estimator vanishes. As it turned out (details will be discussed in the chapter on results) the dominant variables are indeed measurable, some of them directly and others indirectly.

The criterion applied to separate the dominant from the non-dominant variables is as follows: take the peak absolute values $x_{j_{pk}}$ of all the state variables in the simulation; multiply these values by the elements of the feedback gain matrix $k_{r_{i,j}}$ (see eqn. 3.5.2). The products thus obtained form a "weighted gain" matrix of elements $k_{r_{i,j}} x_{j_{pk}}$, each row of which feeds one of the control inputs u_{c_i} . Now determine the maximum absolute value of

each row: the "dominant" variables are defined as those elements with absolute values greater or equal to one-tenth of the maximum value. This criterion seems - and is - rather subjective. However, when the reduced set of variables thus obtained was applied to the feedback and the system simulated, it was found that the transient performance was remarkably similar to that of the system with the full set. Furthermore, the "weighted gain" matrix was computed for a number of cases and the dominant variables were generally the same. The "filter" adopted for the dominant variables was designed so as to let "pass" those variables which were dominant in some of the cases, even if not in all of them.

On the other hand, the settling-down values for permanent disturbances (step-input, in our case) were different from the values they ought to be according to the system equations, values which the variables indeed assumed when the complete set was fed back, in all cases. To overcome this problem, a D.C. compensation scheme was set up such as to force two of the state variables to their proper settling-down value. Only two variables can be forced in such a manner because only two control inputs are available. The logical choice was to select the two most important variables, steam pressure and frequency error. An additional bonus for this choice is the fact that these two variables must return to zero regardless of the disturbance input; thus an on-line computation of the desired values is obviated. The scheme is as follows:

For the steady state, eqn.(3.5.3) becomes

$$\dot{\underline{x}} = A_{oc}\underline{x} + B_{c-du}\underline{k} + B_{d-d}\underline{u} = 0, \quad A_{oc} = A - B_c K_r \quad (3.8.1)$$

$$\underline{x} = -A_{oc}^{-1}(B_{c-du}\underline{k} + B_{d-d}\underline{u}) \quad (3.8.2)$$

Define an output vector $\underline{y}$ such that

$$\underline{y} = C \underline{x} \quad (3.8.3)$$

It was said before that frequency and steam pressure were chosen as the variables to be forced to 0. However the pertinent routine DCCOMP (see section 5.4.3) is such that any two variables may be chosen, their respective numbers being read in as data. In the particular case of pressure and frequency and according to the mathematical model of the system (see sections 2.1.1 and 2.1.3) the vector $\underline{y}$ is

$$\begin{bmatrix} x_{10} \\ x_{30} \end{bmatrix}$$

In general it is desired that $\underline{y}$ be 0, hence:

$$\underline{y} = -CA_{oc}^{-1} B_c \tilde{\underline{k}}_{du} - CA_{oc}^{-1} B_{d-d} u_d = 0 \quad (3.8.4)$$

where $\tilde{\underline{k}}_{du}$ is the forced added control computed instead of $\underline{k}_{du}$ of eqs. (3.5.4), (3.5.9)

$$\boxed{\tilde{\underline{k}}_{du} = -[CA_{oc}^{-1} B_c]^{-1} CA_{oc}^{-1} B_{d-d} u_d} \quad (3.8.5)$$

The results obtained with this formula were highly satisfactory, as will be discussed in the chapter on results. Not only did x_{10}, x_{30} converge to 0 but all the other variables settled at values within a small fraction of one % of the values arrived at with the complete (unfiltered) set of feedback variables.

CHAPTER 4

A NEW METHOD OF OPTIMAL LINEAR REGULATOR DESIGN FOR EXTERNAL DISTURBANCES

The need for a controller to accommodate external disturbances was explained in section (3.5) where the final expressions are also listed. In this chapter it is proposed to describe the method in detail, as well as to show the advantages it claims over other methods.

4.1. The generalized method for finite terminal time T

The conventional "linear quadratic" problem for which the Riccati solution is readily available is given by the system

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) \quad (4.1.1.)$$

where the object is to minimize a cost function

$$V(T) = \frac{1}{2} \int_0^T \left\{ \langle \underline{x}(t), Q\underline{x}(t) \rangle + \langle \underline{u}(t), R\underline{u}(t) \rangle \right\} dt \quad (4.1.2.)$$

$Q \geq 0, R > 0$

For the generalized problem of external disturbance, define:

$\underline{u}_c(t) \equiv$ control input vector

$\underline{u}_d(t) \equiv$ disturbance input vector

$\underline{u}_c(t) \equiv$ value of $\underline{u}_c(t)$ as affected by $\underline{u}_d(t)$

$B_c \equiv$ driving matrix for $\underline{u}_c(t)$

$B_d \equiv$ driving matrix for $\underline{u}_d(t)$

The generalized system and cost function are then, respectively

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B_c \underline{u}_c(t) + B_d \underline{u}_d(t) \quad (4.1.3.)$$

$$V(T) = \frac{1}{2} \int_0^T \left\{ \langle \underline{x}(t), Q\underline{x}(t) \rangle + \langle (\underline{u}_c(t) - \underline{u}_c), R(\underline{u}_c(t) - \underline{u}_c) \rangle \right\} dt \quad (4.1.4.)$$

(From here on, (T) and (t) will be dropped for more convenient writing).

The proposed technique for solving the problem for the optimal control $\underline{u}_c^\circ$ follows the conventional line starting with the expression for the Hamiltonian:

$$H = \frac{1}{2} \langle \underline{x}, Q\underline{x} \rangle + \frac{1}{2} \langle (\underline{u}_c - \bar{\underline{u}}_c), R(\underline{u}_c - \bar{\underline{u}}_c) \rangle + \langle (A\underline{x} + B_c \underline{u}_c + B_d \underline{u}_d), \underline{\lambda} \rangle \quad (4.1.5.)$$

From here:

$$\dot{\underline{x}} = \frac{\partial H}{\partial \underline{\lambda}} = A\underline{x} + B_c \underline{u}_c + B_d \underline{u}_d \quad (4.1.6.)$$

$$-\dot{\underline{\lambda}} = \frac{\partial H}{\partial \underline{x}} = Q\underline{x} + A^T \underline{\lambda} \quad (**) \quad (4.1.7.)$$

$$\frac{\partial H}{\partial \underline{u}_c} = R(\underline{u}_c - \bar{\underline{u}}_c) + B_c^T \underline{\lambda} \quad (4.1.8.)$$

$$\underline{u}_c^\circ \Rightarrow \frac{\partial H}{\partial \underline{u}_c} = 0 \Rightarrow \underline{u}_c^\circ = \bar{\underline{u}}_c - R^{-1} B_c^T \underline{\lambda} \quad (4.1.9.)$$

The optimal control problem is now reduced to solving the equations

$$\begin{cases} \dot{\underline{x}} = A\underline{x} + B_c \bar{\underline{u}}_c - B_c R^{-1} B_c^T \underline{\lambda} + B_d \underline{u}_d \end{cases} \quad (4.1.10.)$$

$$\begin{cases} -\dot{\underline{\lambda}} = Q\underline{x} + A^T \underline{\lambda} \end{cases} \quad (4.1.11.)$$

With boundary conditions^[22] (*)

$$\begin{cases} \underline{x}(0) = \underline{x}_0 \\ \underline{\lambda}(T) = 0 \end{cases}$$

Call $[\tilde{B}] \equiv [B_c \ B_d]$, $\bar{\underline{u}} \equiv \begin{bmatrix} \bar{\underline{u}}_c \\ \underline{u}_d \end{bmatrix}$ for convenience

Rewrite (4.1.10.), (4.1.11.) as:

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{\lambda}} \end{bmatrix} = \begin{bmatrix} A & -B_c R^{-1} B_c^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} \underline{x} \\ \underline{\lambda} \end{bmatrix} + \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \bar{\underline{u}} \quad (4.1.12.)$$

and attempt to solve for $\lambda(0) = \lambda_0$

(*) The more general expression for eq. (4.1.2) includes a second term $L_f(\underline{x}(T))$. It can be shown that $\underline{\lambda}(T) = \partial L_f / \partial \underline{x} |_{\underline{x} = \underline{x}(T)}$. In our case $L_f = 0$, hence $\underline{\lambda}(T) = 0$.

(**) ()^T = () transpose.

The Hamiltonian matrix can be decomposed as

$$M = \begin{bmatrix} A & -B_c R^{-1} B_c^T \\ -Q & -A^T \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} -\Lambda & 0 \\ 0 & \Lambda \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \quad (4.1.13.)$$

where Λ is the positive half of the canonical form of M , the eigenvalues of which [34] are symmetrical with respect to the imaginary axis, W is the eigenvector matrix and V its inverse.

Let $\phi(t, \tau)$ be the state transition matrix for eq. (4.1.12.)

$$\phi(t, \tau) = e^{M(t-\tau)} = W \begin{bmatrix} e^{-\Lambda(t-\tau)} & 0 \\ 0 & e^{\Lambda(t-\tau)} \end{bmatrix} V \quad (4.1.14.)$$

The solution of eq. (4.1.12) at terminal time T is

$$\begin{bmatrix} \underline{x}(T) \\ \underline{\lambda}(T) \end{bmatrix} = \phi(T, t) \begin{bmatrix} \underline{x}(t) \\ \underline{\lambda}(t) \end{bmatrix} + \int_t^T \phi(T, \tau) \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \underline{u} d\tau \quad (4.1.15.)$$

We shall later generalize for $\underline{u}_d$ being any given function. Let us continue now with the assumption that $\underline{u}_d$ is a constant, i.e. a step function, in which case $\begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \underline{u}$ may be removed from the integral.

From eq. (4.1.14.)

$$\int_t^T \phi(T, \tau) d\tau = W \begin{bmatrix} -\Lambda^{-1} (e^{-\Lambda(T-t)} - I) & 0 \\ 0 & \Lambda^{-1} (e^{\Lambda(T-t)} - I) \end{bmatrix} V \quad (4.1.16.)$$

Hence

$$\begin{aligned} \begin{bmatrix} \underline{x}(T) \\ \underline{\lambda}(T) \end{bmatrix} &= \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \left\{ \begin{bmatrix} e^{-\Lambda(T-t)} & 0 \\ 0 & e^{\Lambda(T-t)} \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} \underline{x}(t) \\ \underline{\lambda}(t) \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} -\Lambda^{-1} (e^{-\Lambda(T-t)} - I) & 0 \\ 0 & \Lambda^{-1} (e^{\Lambda(T-t)} - I) \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \underline{u} \right\} \\ &= \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} e^{-\Lambda(T-t)} V_{11} & e^{-\Lambda(T-t)} V_{12} \\ e^{\Lambda(T-t)} V_{21} & e^{\Lambda(T-t)} V_{22} \end{bmatrix} \begin{bmatrix} \underline{x}(t) \\ \underline{\lambda}(t) \end{bmatrix} \\ &\quad + \begin{bmatrix} -\Lambda^{-1} (e^{-\Lambda(T-t)} - I) V_{11} & -\Lambda^{-1} (e^{-\Lambda(T-t)} - I) V_{12} \\ \Lambda^{-1} (e^{\Lambda(T-t)} - I) V_{21} & \Lambda^{-1} (e^{\Lambda(T-t)} - I) V_{22} \end{bmatrix} \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \underline{u} \end{aligned} \quad (4.1.17.)$$

$$\begin{aligned} \underline{\lambda}(T) = & \left[W_{21} e^{-\Lambda(T-t)} V_{11} + W_{22} e^{\Lambda(T-t)} V_{21} \right] \underline{x}(t) + W_{21} \left[e^{-\Lambda(T-t)} V_{12} + W_{22} e^{\Lambda(T-t)} V_{22} \right] \underline{\lambda}(t) \\ & + \left[W_{21} \Lambda^{-1} (I - e^{-\Lambda(T-t)}) V_{11} + W_{22} \Lambda^{-1} (e^{\Lambda(T-t)} - I) V_{21} \right] \tilde{B} \underline{u} \end{aligned} \quad (4.1.18.)$$

For the boundary condition $\underline{\lambda}(T) = 0$ the expression for $\underline{\lambda}(t)$ is

$$\begin{aligned} \underline{\lambda}(t) = & - \left[W_{21} e^{-\Lambda(T-t)} V_{12} + W_{22} e^{\Lambda(T-t)} V_{22} \right]^{-1} \left\{ \left[W_{21} e^{-\Lambda(T-t)} V_{11} + W_{22} e^{\Lambda(T-t)} V_{21} \right] \underline{x}(t) + \right. \\ & \left. + \left[W_{21} \Lambda^{-1} (I - e^{-\Lambda(T-t)}) V_{11} + W_{22} \Lambda^{-1} (e^{\Lambda(T-t)} - I) V_{21} \right] \tilde{B} \underline{u} \right\} \end{aligned} \quad (4.1.19.)$$

Let us now use certain properties of matrix inversion^[34] for the above eigenvectors under certain normalization conditions which will be described in section (4.4)

$$W^{-1} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}^{-1} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} = \begin{bmatrix} W_{22}^T & -W_{12}^T \\ -W_{21}^T & W_{11}^T \end{bmatrix} \quad (4.1.19.a)$$

Equation (4.1.19) then becomes

$$\begin{aligned} \underline{\lambda}(t) = & - \left[-W_{21} e^{-\Lambda(T-t)} W_{12}^T + W_{22} e^{\Lambda(T-t)} W_{11}^T \right]^{-1} \left\{ \left[W_{21} e^{-\Lambda(T-t)} W_{22}^T - W_{22} e^{\Lambda(T-t)} W_{21}^T \right] \underline{x}(t) + \right. \\ & \left. + \left[W_{21} \Lambda^{-1} (I - e^{-\Lambda(T-t)}) W_{22}^T - W_{22} \Lambda^{-1} (e^{\Lambda(T-t)} - I) W_{21}^T \right] \tilde{B} \underline{u} \right\} \end{aligned} \quad (4.1.20.)$$

From eq. (4.1.9.)

$$\begin{aligned} \underline{u}_c(t) = & \underline{u}_c + R^{-1} B^T \left[-W_{21} e^{-\Lambda(T-t)} W_{12}^T + W_{22} e^{\Lambda(T-t)} W_{11}^T \right]^{-1} \left\{ \left[W_{21} e^{-\Lambda(T-t)} W_{22}^T - W_{22} e^{\Lambda(T-t)} W_{21}^T \right] \underline{x}(t) + \right. \\ & \left. + \left[W_{21} \Lambda^{-1} (I - e^{-\Lambda(T-t)}) W_{22}^T - W_{22} \Lambda^{-1} (e^{\Lambda(T-t)} - I) W_{21}^T \right] \tilde{B} \underline{u} \right\} \end{aligned} \quad (4.1.21.)$$

Now call $K(T, t)$ the feedback gain matrix from $\underline{x}$ to $\underline{u}_c$ and

$K_d(T, t)$ the gain matrix from $\tilde{B} \underline{u}$ to $\underline{u}_c$

$$K(T, t) = R^{-1} B^T \left[W_{21} e^{-\Lambda(T-t)} W_{12}^T - W_{22} e^{\Lambda(T-t)} W_{11}^T \right]^{-1} \left[W_{21} e^{-\Lambda(T-t)} W_{22}^T - W_{22} e^{\Lambda(T-t)} W_{21}^T \right] \quad (4.1.22.)$$

$$\begin{aligned} K_d(T, t) = & R^{-1} B^T \left[-W_{21} e^{-\Lambda(T-t)} W_{12}^T + W_{22} e^{\Lambda(T-t)} W_{11}^T \right]^{-1} \left[W_{21} \Lambda^{-1} (e^{-\Lambda(T-t)} - I) W_{22}^T \right. \\ & \left. + W_{22} \Lambda^{-1} (e^{\Lambda(T-t)} - I) W_{21}^T \right] \end{aligned} \quad (4.1.23.)$$

Equation (4.1.21.) now becomes

$$\underline{u}_c^o(t) = -K(T,t)\underline{x}(t) - K_d(T,t)\widetilde{B}\underline{\bar{u}} + \underline{\bar{u}}_c \quad (4.1.24.)$$

$$= -K(T,t)\underline{x}(t) - K_d(T,t)[B_c \underline{\bar{u}} + B_d \underline{u}_d] + \underline{\bar{u}}_c$$

$$\underline{u}_c^o(t) = -K(T,t)\underline{x}(t) - [K_d(T,t)B_c - I]\underline{\bar{u}}_c - K_d(T,t)B_d \underline{u}_d \quad (4.1.25.)$$

From the steady state equations a relationship between $\underline{\bar{u}}_c$ and $\underline{u}_d$

must be found such that

$$\underline{\bar{u}}_c = K_{cd} \underline{u}_d \quad (4.1.26.)$$

Hence

$$\begin{aligned} \underline{u}_c^o(t) &= -K(T,t)\underline{x}(t) - \left\{ K_d(T,t)[B_c K_{cd} + B_d] - K_{cd} \right\} \underline{u}_d \\ &= -K(T,t)\underline{x}(t) - K_{dg}(T,t)\underline{u}_d \end{aligned} \quad (4.1.27.)$$

$$\boxed{\underline{u}_c^o(t) = -K(T,t)\underline{x}(t) + \underline{k}_{du}(T,t)} \quad (4.1.28.)$$

where K_{dg} and $\underline{k}_{du}$ have been introduced for more convenient writing.

From the original equations (4.1.9.), (4.1.10.) and the result in

(4.1.28.), the optimal system is now

$$\boxed{\dot{\underline{x}}(t) = [A - B_c K(T,t)]\underline{x}(t) + B_c \underline{k}_{du}(T,t) + B_d \underline{u}_d} \quad (4.1.29.)$$

$$\dot{\underline{x}}(t) = A_{oc}(T,t)\underline{x}(t) + B_c \underline{k}_{du}(T,t) + B_d \underline{u}_d \quad (4.1.30.)$$

For a more closed expression we can sum up the last two terms of

eq. (4.1.30.) such that

$$B_{dg}(T,t) = B_d - B_c [K_d(T,t)[B_c K_{cd} + B_d] - K_{cd}] \quad (4.1.31.)$$

whence

$$\boxed{\dot{\underline{x}}(t) = A_{oc}(T,t)\underline{x}(t) + B_{dg}(T,t)\underline{u}_d} \quad (4.1.32.)$$

This sums up the controller design for finite terminal time T and

constant (step function) disturbance $\underline{u}_d$. Its relevance is mainly

theoretical, since the gain matrices $K(T,t)$ and $K_d(T,t)$ must be

computed for every t . At its best, a discrete control could be

envisaged with a permanent updating of the controller elements.

In fact its main purpose will be seen in the next section, where a great simplification is performed for $T \rightarrow \infty$. This will yield a controller that can be computed once and for all.

Before finishing this section, let us look at the implications for the case when $\underline{u}_d(t)$ is any given function, as opposed to the step function we have dealt with so far.

The second term of eq. (4.1.15.) will now be

$$\int_t^T \phi(T, \tau) \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \underline{\bar{u}}(\tau) d\tau = \underline{z}(t) \quad (4.1.33.)$$

where $\underline{z}(t)$ is the result of the integration performed for every t .

Equation (4.1.17.) becomes now

$$\begin{bmatrix} \underline{x}(T) \\ \underline{\lambda}(T) \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} e^{-\Lambda(T-t)} & 0 \\ 0 & e^{\Lambda(T-t)} \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} \underline{x}(t) \\ \underline{\lambda}(t) \end{bmatrix} + \begin{bmatrix} \underline{z}^1(t) \\ \underline{z}^2(t) \end{bmatrix} \quad (4.1.34.)$$

$$\underline{\lambda}(T) = [W_{21} e^{-\Lambda(T-t)} V_{11} + W_{22} e^{\Lambda(T-t)} V_{21}] \underline{x}(t) + [W_{21} e^{-\Lambda(T-t)} V_{12} + W_{22} e^{\Lambda(T-t)} V_{22}] \underline{\lambda}(t) + \underline{z}^2(t) \quad (4.1.35.)$$

$$\underline{\lambda}(t) = -[-W_{21} e^{-\Lambda(T-t)} W_{12}^T + W_{22} e^{\Lambda(T-t)} W_{11}^T]^{-1} \left\{ [W_{21} e^{-\Lambda(T-t)} W_{22}^T - W_{22} e^{\Lambda(T-t)} W_{21}^T] \underline{x}(t) + \underline{z}^2(t) \right\} \quad (4.1.36.)$$

$K(T, t)$ retains the value obtained in equation (4.1.22.)

$$K_z(T, t) = R^{-1} B_c^T [W_{21} e^{-\Lambda(T-t)} W_{12}^T - W_{22} e^{\Lambda(T-t)} W_{11}^T]^{-1} \quad (4.1.37.)$$

$$\underline{u}_c^0(t) = -K(T, t) \underline{x}(t) - K_z(T, t) \underline{z}^2(t) + \underline{u}_c(t) \quad (4.1.38.)$$

There is not much point in going further towards a closed expression for the control and the whole (optimally controlled) system, since any expression will include $\underline{z}^2(t)$ which is an integration result anyway. The conclusion is that the controller can still be designed, although the continuous heavy on-line computing renders the design all but useless, except for a possible theoretical follow-up.

If the disturbance input is a known and analytically integrable function, then a closed expression can be obtained.

4.2. Simplification for infinite terminal time

In the case of $\underline{u}_d$ constant, the expressions for $K(T)$, $K_d(T)$ can be simplified for $T \rightarrow \infty$.

Inspecting eq. (4.1.22.)

$$K(T \rightarrow \infty) = R^{-1} B_c^T [W_{11}^T]^{-1} W_{21}^T \quad (4.2.1.)$$

On the other hand, the solution of the matrix Riccati equation via eigenvectors of the Hamiltonian matrix, for the special case with no external disturbance and infinite terminal time^[32] yields

$$\underline{\lambda}(t) = P \underline{x}(t) \quad (4.2.2.)$$

and

$$\underline{u}_c^o(t) = -R^{-1} B_c^T \underline{\lambda}(t) = -R^{-1} B_c^T P \underline{x}(t) \quad (4.2.3.)$$

where P is the Riccati matrix which must be symmetric.

The general case discussed in the present method includes the D.C. gain due to the external disturbance, yielding (see eqs. 4.1.28. and 4.2.1.)

$$\underline{u}_c^o(t) = -K \underline{x}(t) + \underline{k}_{du} \quad (4.2.4.)$$

$$= -R^{-1} B_c^T [W_{11}^T]^{-1} W_{21}^T \underline{x}(t) + \underline{k}_{du} \quad (4.2.5.)$$

Comparing equations (4.2.3.) and (4.2.5.) it is clear that the first term of eq. (4.2.5.), representing the dynamic component of the control, is equal to eq. (4.2.3.) which handles the dynamic performance only. Hence $[W_{11}^T]^{-1} W_{21}^T = P$, and since P is symmetric, eq. (4.2.1.) becomes

$$K(T \rightarrow \infty) = R^{-1} B_c^T W_{21} W_{11}^{-1} \quad (4.2.6.)$$

which is the formula arrived at in^[32].

Now, inspecting eq. (4.1.23.)

$$K_d(T \rightarrow \infty) = R^{-1} B_c^T [W_{22} e^{\Lambda^T W_{11} T}]^{-1} [W_{22} \Lambda^{-1} e^{\Lambda^T W_{21} T}]$$

$$K_d(T \rightarrow \infty) = R^{-1} B_c^T (W_{11}^T)^{-1} \Lambda^{-1} W_{21}^T \quad (4.2.7.)$$

Before finishing this section, a word of caution is necessary: the disturbance input $\underline{u}_d$ was assumed to be a step function. The controller thus designed is therefore strictly optimal only for step functions. For any other disturbance function, if the disturbance settles at a steady value $\underline{u}_d$, the design will still be correct towards the settling time. This is an important conclusion, in particular for the case studied in this thesis, where one of the main objectives is to obtain a performance such that the plant recovers fast and smoothly from an external disturbance such as a sudden increase or decrease in the demand of power. For any disturbance other than a step function, the second term in equation (4.1.15) must be integrated with $\underline{u}$ and no closed expression will be found for the regulator.^(*) On the other hand, if the disturbance function is known and analytically integrable, such as for instance a sinusoid, the computation will still yield a closed expression.

Finally, the matrices called K and K_d in this section and in the previous one, are called K_r and K_{dr} , respectively, in section (3.5) and in the computer programs. This is so due to the particular matrix manipulations of the programs, where K_r and K_{dr} are submatrices of larger matrices computed for the case when all the inputs are used for control. Also, R is called R_c in section (3.5), where R is used for the case of all inputs used for control.

4.3. Advantages of the method

This method yields the D.C. gain due to the steady disturbance simultaneously with the dynamic optimal regulator. Both features are incorporated in the controller. This in itself is not considered

(*) In using the closed expressions derived above, the controller will behave with an approximation comparable to that of decomposing any function into elementary steps.

a major advantage, since the DC gain can be computed by other means as well (see section 3.8). Even so, however, this method ensures the exact DC gain for all variables, whereas other methods such as the one described in section (3.8) determines the DC values of only part of the state vector, depending on the number of control inputs available.

The important advantages are:

- (i) Theoretically the method allows any type of disturbance, as was shown in the closing paragraphs of section (4.1) (see eq. 4.1.33. and onwards).
- (ii) Computationally the method is fairly simple for the regulator design and transients calculation. Let us compare the new method presented here with the fairly well known one^[37] which consists of adding one state for each disturbance input and will be briefly described here.^(*)

Let us repeat eqs. (4.1.1.) and (4.1.3.) for the non-disturbed and disturbed linear dynamic system, respectively

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B\underline{u}(t) \quad (4.3.1.)$$

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B_c \underline{u}_c(t) + B_d \underline{u}_d(t) \quad (4.3.2.)$$

For one specific set of disturbance inputs, eq. (4.3.2.) can be rewritten as follows, for $\underline{u}_d$ constant.

$$\dot{\underline{x}}(t) = A\underline{x}(t) + B_c \underline{u}_c(t) + k \underline{d} \quad (4.3.3.)$$

(*) The description does not follow exactly the presentation in^[37], where the magnitude of $\underline{u}_d(t)$ is unknown. Also, for the case of several disturbance inputs, if their magnitudes change in equal proportion, only one additional state is needed.

where $\underline{d}$ is an n -component vector (as $\underline{x}$) and k a scalar which may change for different disturbances. As observed in the footnote, if there are several disturbances which do not change in the same proportion, then $\underline{d}_1, \dots, \underline{d}_n$ are needed for n inputs. Add now one additional state x_{n+1} to $\underline{x}$, call the enlarged vector $\underline{z}$ and define new matrices $\bar{A}, \bar{B}_c$

$$\underline{z} = \begin{bmatrix} \underline{x} \\ x_{n+1} \end{bmatrix}, \quad \bar{A} = \begin{bmatrix} A & \underline{d} \\ 0 & 0 \end{bmatrix}, \quad \bar{B}_c = \begin{bmatrix} B_c \\ 0 \end{bmatrix} \quad (4.3.4.)$$

The new system is now

$$\dot{\underline{z}} = \bar{A}\underline{z} + \bar{B}_c u_c \quad (4.3.5.)$$

which, if decomposed, yields

$$\left. \begin{aligned} \dot{\underline{x}} &= A\underline{x} + \underline{d} x_{n+1} + B_c u_c \\ \dot{x}_{n+1} &= 0 \end{aligned} \right\} \quad (4.3.6.)$$

Comparing eqs. (4.3.3.) and (4.3.6.) it is seen that they are identical if

$$\left. \begin{aligned} x_{n+1} &= k \\ \dot{x}_{n+1} &= 0 \end{aligned} \right\} \quad (4.3.7.)$$

which is acceptable. The new system (4.3.5.) is now dealt with, for optimal control, in the usual manner via the solution of the Riccati equation.

The advantage of the method developed for this thesis lies in obviating the need of creating a new state (or several) and hence the need of recomputing the eigenvalues/eigenvectors of the system for each new disturbance. Perhaps it should be added in fairness that when only one new state is needed, the additional eigenvalue is

zero and the eigenvector matrix is fairly similar to the original one, consisting of the original eigenvector matrix enlarged by one new pair of row and column. (*) Having said that, there are still two advantages to our method:

- a. In the new method there is no difference whether there are one or more different disturbance inputs.
- b. Even for one disturbance only, the "conventional" method yield a zero eigenvalue. If the original system A has no zero eigenvalue nothing more has to be said. However if A does have one pole at the origin - which in the author's experience is not uncommon for physical systems, and certainly is the case for the system investigated in this thesis - then the additional zero root turns the system unstable. Although such a system would become stable with optimal control introduced, the problem remains when it is being run - or simulated - with constraints on its control inputs, for then the system runs uncontrolled during the time intervals when all the control inputs lie on their saturation lines, as was shown in section (3.7).

(*) It can easily be shown that if $A = W\Lambda V$, an $n \times n$ matrix, and $\bar{A}$ of eqs. (4.3.4.) is $\bar{A} = R^T S$, the first n eigenvalues of $\bar{A}$ are equal to those of A , the new γ is nil and the new eigenvectors are as follows:

$$\underline{r}^1, \dots, \underline{r}^n = \begin{bmatrix} W^1 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} W^n \\ 0 \end{bmatrix}$$

The last eigenvector $\underline{r}^{n+1}$ is calculated as follows:

$$\underline{r}^{n+1} = \frac{h}{1}, \quad \bar{A} \underline{r}^{n+1} = 0 \quad \underline{r}^{n+1} = [0], \quad A \underline{h} + \underline{d} = [0], \quad \underline{h} = -A^{-1} \underline{d} \quad (4.3.8.)$$

4.4. Normalization conditions for the eigenvectors of the Hamiltonian matrix

In section (4.1) certain properties of matrix inversion for eigenvectors were used for conditions of normalization not discussed there. The properties, found by O'Donnell^[34] are repeated here

$$W^{-1} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}^{-1} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} = \begin{bmatrix} W_{22}^T & -W_{12}^T \\ -W_{21}^T & W_{11}^T \end{bmatrix} \quad (4.4.1.)$$

As Walter^[40] rightly pointed out, the scaling is not the expected one of normalizing to unit length, but instead involves the following procedure:

- a. For the $2n \times 2n$ eigenvector matrix, the first n columns (W_{11}, W_{21}) are scaled arbitrarily.
- b. The remaining n columns are first scaled arbitrarily (U_{12}, U_{22}) and then rescaled using the following formula which it is fairly easily shown to be correct.

The result will be the final matrix W

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} W_{11} & U_{12} \\ W_{21} & U_{22} \end{bmatrix} \begin{bmatrix} I & O \\ O & G \end{bmatrix} \quad (4.4.2.)$$

$$\text{where} \quad G = [W_{11}^T \ U_{22} \ -W_{21}^T \ U_{12}]^{-1} \quad (4.4.3.)$$

is shown to be a diagonal matrix.

It is interesting to note that Walter's remark, while being correct does not invalidate the expression for the solution of the matrix Riccati equation $(P = W_{21} \ W_{11}^{-1})$ for a system with no permanent external disturbances and infinite terminal time, as Vaughan^[41] in turn showed: the reason is of course that only W_{11}, W_{21} are needed.

However, in the method developed here for external disturbance, for the case of finite terminal time all four quarters

of W are used, as can be seen in eqs. (4.1.22.), (4.1.23.), hence Walter's normalization procedure is needed. On the other hand, for infinite terminal time the normalization is again not needed, as can be seen in eqs. (4.2.6.), (4.2.7.).

CHAPTER 5

DESCRIPTION OF DIGITAL AND ANALOG PROGRAMS

It was stated at the outset of chapter 3 that the computations of this study are broken into five main components. The following digital programs contain these computations.

- (i) SSNL2D : steady state solution of the
non-linear set of equations
describing the system.
- (ii) MXEV2X1 : Matrix evaluation, i.e. construction
of the linear model(s).
- (iii) REGULA : Solution of the optimal linear regulator
problem.
- (iv) SIMULA : Transients computation (free simulation)
- (v) SIMSAT : Saturated simulation (constraints on the
control inputs).

The programs will be detailed below, but the description in each section will refer to the corresponding section in chapter 3, without repetitions, if possible.

As an appendix to this thesis, the important parts of the programs as run on the CDC 6600 are listed. There are quite a number of comments in the programs, which make it superfluous to write them down in the following sections. The descriptions here are therefore intended to be brief.

5.1. SSNL2D (Steady State)

See section 3.1.

On the CDC 6600 the compilation time is less than 10 seconds. The execution time varies with the number of iterations that are necessary, a typical figure being 2 - 3 seconds. The store needed is about 16K, excluding the system (about 3K).

5.1.1 Main program

The dimensioning and initialization of the program is described in the program itself. It is only necessary to add a few definitions. The key routine called from the main program for least squares, is VAO2A. For this routine:

M $\equiv$ number of functions F

N $\equiv$ number of variables X

NW $\equiv$ size of auxilliary array W

$$\geq N + (M + \frac{3N+1}{Z}) (N+1)$$

NA $\equiv$ number of algebraic variables XA

MA $\equiv$ number of algebraic functions G

An equivalences was established between the arrays of X,XA on the one hand and F,G on the other, to help the handling of them.

E $\equiv$ array of the accuracy of the variables

ESCALE $\equiv$ limits the step size of the search.

Normally x_i will not be changed by more than
ESCALE* E_i in a single step.

MAXFUN $\equiv$ maximum number of calls for the evaluation of functions

(CALFUN), set in order to stop the program if it does not converge reasonably fast.

IPRINT $\equiv$ periodical numbers of iterations after which the values of the variables and the functions are printed. No printout if IPRNT = 0

ITAUW, etc. are set to 1 for the steady state program. Recall

that the decks of cards were initially intended to be used also for non-linear simulation.

The most important points to remember are:

- (i) The scaling of functions and variables, done in order to enable VAO2A to work better.

The is done with routines SCLEVA, SCALEF, SCALEV.

- (ii) The functions evaluation. This is done in two separate routines: CALFUNO for the original 40 functions, including 23 differential and 17 algebraic equations. CALFUN for the 35 functions remaining for the solution of the steady-state set of equations.
- (iii) The routine REPLACE which changes the set in CALFUNO to the set in CALFUN and vice versa.

5.1.2 List of subroutines

CALFUNO	YTAU
REPLACE	EXPL
CALFUN	CALCLT
SCALEV	FUN1
SCALEF	FUN2
SCLEVA	FUN3
READ	FUN4
OINPUT	FUN5
INITIAL	FUN6
OUTPUT	INTPOL
OPUNCH	VAO2A
ULN	VDOL1A
STAAL	

5.1.3 Description of subroutines

CALFUNO : Carries the complete list of equations described as Model X(basic) in chapter 3, written as 40 functions to be evaluated.

CALFUN : Carries 35 functions only, as explained in section (3.1). This is the set to be minimized by VAO2A.

REPLACE : Is the bridge between CALFUNO and CALFUN, replacing several variables which in the steady state are constants, by other variables or inputs, or vice versa.

SCALEV : Scales variables to unit order in three stages:

1. The variables are scaled to unit order before calling the minimisation routine; 2. The individual terms of each function in CALFUN are scaled at each call with the inverse of the scaling factor; 3. The variables are scaled back to their original values after completing the minimization routine.

SCLEVA : Tests the value of each function after the first call of CALFUN and sets a scaling-down factor if the value is larger than specified (otherwise the factor is set to one). This operation is done once only.

SCALEF : Multiplies all functions evaluated in CALFUN, by the scaling factor computed in SCLEVA. This operation is done at the end of each CALFUN call.

READ : Reads in initial values as data.

OLINPUT : Prints initial values, calling sub-routines UIN and STAAL for formats, etc.

INITIAL : Reads and prints comments on level of power and initial values.

OUTPUT : Prints final values, calling UIN, STAAL, YTAU and EXPL.

OPUNCH : Punches final values, to be used as input data for the next program, MXEV2XL.

CALCLT : Calculates time delays and output variables.

FUN1 to FUN6 : Tables of empirical functions which are used in

some of the equations of the non-linear set.

INTPOL : Straight interpolation routine for use with FUN1 to FUN6.

VAO2A : This is the Powell routine for minimizing a sum of squares of the form

$$f(x_1, x_2, \dots, x_n) = \sum_{k=1}^m f_k^2(x_1, \dots, x_n) \quad (5.1.1.)$$

where $m \geq n$

The restrictions on this routine are that $m, n \geq 2$ and that the matrix H must resemble G, where

$$H = [h_{ij}] = \left[\sum_{k=1}^m \frac{\partial f_k}{\partial x_i} \cdot \frac{\partial f_k}{\partial x_j} \right] \quad (5.1.2.)$$

$$G = [g_{ij}] = \left[\sum_{k=1}^m \left(\frac{\partial f_k}{\partial x_i} \cdot \frac{\partial f_k}{\partial x_j} + f_k \frac{\partial^2 f_k}{\partial x_i \partial x_j} \right) \right] \quad (5.1.3.)$$

It can be shown that G indeed resembles H when the value of f is small at the minimum, hence the need for routines SCLEVA and SCALEF. VAO2A calls VDOLA, the routine in which the search is done for one function of one variable at a time.

5.1.4 Output

- (i) List of initial values for all variables.
- (ii) Function values of the 40 original equations calculated in CALFUNO (no scaling).
- (iii) Function values of the 35 remaining equations to be solved for the steady state (no scaling).
- (iv) Values of variables and functions calculated after each IPRINT iterations, with variables and functions scaled.

- (v) Repetition of (iii), after minimization is ended.
- (vi) Repetition of (ii), after minimization is ended.
- (vii) List of final values for all variables.
- (viii) Punch out of (vii). The first 6 items are only printed.

5.2 MXEV2X1 (Linear Model)

See section 3.2.

On the CDC 6600 the compilation time is under 18 seconds.

The execution time varies according to the requests, a typical case taking about 7 seconds. The storage needed is 41K, system excluded.

5.2.1 Main program

As in the previous program, the dimensioning and initialization is explained in the program itself.

Only the first few parameters need to be explained:

N , NA , NU , NE , NY are the dimensions of the vectors $\underline{x}$, $\underline{x}_A$, $\underline{u}$, $\underline{x}_E$, $\underline{y}$ of the basic model, the steady state values of which are the output of program SSNL2D and the input of MXEV2X1. This means that for each level of operation a different linear model is obtained in MXEV2X1.

ND is the number of pure time delays to be approximated.

The comments explain quite clearly the meaning of each of the matrices evaluated, and how the model progresses from the basic one (X), to the equivalent one which includes delay approximations ($X1$), to the comprehensive one which includes the new equations in FDBCK1.

5.2.2 List of subroutines

DELAY1	EDMX	SLOPE5
FDBCK1	FMX	SLOPE6
CALFUN	PMX	INTPOL
TESTMX	PAMX	
TESTMX2	QMX	CAUTION
COMPARE	GMX	IMBEDV
NEWXU	GAMX	IMBED
XAX1U	HMX	EQUVRR
WRIMX	AlMX	UNIT
DRAWMX	B1MX	CARVUP
DRAWLM	C1MX	MULMXV
SCAN	D1MX	ADDV
READ	ElMX	MULTI
OINPUT	ASMX	ADD
UIN	BSMX	SUBTR
STAAL	FUN1	NULL
YTAU	FUN2	NEGATE
EXPL	FUN3	RORDRM
PCH2O13	FUN4	INVERT
AMX	FUN5	XEIGEN
BMX	FUN6	EIG6
BDMX	SLOPE1	DETMT
CMX	SLOPE2	HESSEN
DMX	SLOPE3	QREIG
EMX	SLOPE4	QRT

5.2.3 Description of subroutines

DELAY1 : In this routine the order and type of approximation for each of the four delays are set independently. For the present choice, see section (3.2.2). New equations and variables are introduced.

FDBCK1 : In this routine the additional equations mentioned in section (3.2.3) are set and incorporated into the equivalent model (X1) to form the final comprehensive model (X2) which represents the uncontrolled system. There is a great deal of matrix manipulation which is done by calling specific matrix routines, briefly described later. The routine is kept in such a way as to enable the user to change, add or remove equations without upsetting the structure of the program. Each change embodies a new "case". There are 7 cases which have been produced in chronological order, and among which only two are at present relevant:

Case 7 was constructed specifically to meet the model set up on the analog "simulator" at Winfrith in order to test the correctness of the linear model. The responses obtained subsequently with the SIMULA program closely matched the "simulator" response. This case excludes moderator control and load following and includes feedwater control (the one existing at the plant) only.

Case 3 is as case 7 but includes load following. It is the model finally adopted for our study of the optimal regulator problem.

CALFUN : This routine contains the basic non-linear model tested in CALFUNO (in SSNL2D) and its purpose is to test the correctness of the linear model by comparing computed values in the linear and the non-linear models.

TESTMX : This routine test the correctness of the linear model

by comparing the results obtained using the linear and the non-linear models. It sets deviations from the steady state, such that the linear model operates on the deviations while the N/L model does it on the values of the steady state plus the deviations. If the linearity were perfect, the results would have to be equal. As it is, when deviations of up to 2% were used, the differences between the results obtained using the two models, are well within one per cent for the most part, although a few results are further apart, one as far as 10%. Using deviations of up to 5% from the steady state values, some errors are as large as 20% and 30%. The "bad behaved" equations are in general those including functions with tabulated values, which suggests that polynomial fitting might be necessary for better approximation.

$\bar{f}$, $\bar{g}$ are the values obtained from the non-linear set (CALFUN) (see eqs. (2.1) and (2.2)).

$$\bar{f} = f(\underline{x} + \delta\underline{x}, \underline{x}_A + \delta\underline{x}_A, \underline{u} + \delta\underline{u}) - f(\underline{x}, \underline{x}_A, \underline{u}) \quad (5.2.1.)$$

$$\bar{g} = g(\underline{x} + \delta\underline{x}, \underline{x}_A + \delta\underline{x}_A, \underline{u} + \delta\underline{u}) - g(\underline{x}, \underline{x}_A, \underline{u}) \quad (5.2.2.)$$

These values are compared with δf , δg , from the linear model (see eqs. 3.2.18 and 3.2.19)

$$\delta f = A\delta\underline{x} + B\delta\underline{x}_A + B_D \delta\underline{x}_D + C\delta\underline{u} \quad (5.2.3.)$$

$$\delta g = D\delta\underline{x} + E\delta\underline{x}_A + E_D \delta\underline{x}_D + F\delta\underline{u} \quad (5.2.4.)$$

$\underline{x}_D$ in eqn. (5.2.3.) and (5.2.4.) is replaced by the expression obtained when solving the pair of equations (3.2.20), (3.2.21) for $\dot{\underline{z}} = 0$ and eliminating $\underline{z}$. Having done so, the two equations (5.2.3.) and (5.2.4.) become

$$\delta f = [A+B_D(G-HQ^{-1}P)]\delta\underline{x} + [B+B_D(G_A-HQ_A^{-1}P_A)]\delta\underline{x}_A + C\delta\underline{u} \quad (5.2.5.)$$

$$\delta \underline{g} = [D + E_D(G - HQ^{-1}P)]\delta \underline{x} + [E + E_D(G_A - HQ^{-1}P_A)]\delta \underline{x}_A + F\delta \underline{u} \quad (5.2.6.)$$

On the whole, the conclusion from this test is that the linearization is reasonably acceptable. While there is no absolute criterion of "good" linearity (this must be tested dynamically), gross errors can be detected.

TESTMX2 : Tests the correctness of the equivalent linear model (X1) against the basic model (X). It really is only a test of the correct manipulation of matrices, and not of the "goodness" of linearity as in TESTMX. In essence it is not more than a different road - from the beginning - arriving at the equivalent model, and then the elements of the final matrices A_s and B_s in eq. (3.2.26) are compared. The pertinent equations are printed in the program, both in the routine and in the output.

COMPARE : Compares the elements of one matrix with those of submatrices of another, partitioned, matrix.

NEWXU : Punches out, in alphameric characters, the names, units, steady state values of all the state variables and inputs of the final comprehensive linear model (X2) as well as those of some additional parameters, all of them required in the SIMULA and SIMSAT programs (see sections 5.4, 5.5).

XAX1U : Computes and punches out the matrices XAX1 and XAU necessary for calculating $\underline{x}_A$ from $\underline{x}_1$ and $\underline{u}$, needed in SIMULA and SIMSAT. From eq. (3.2.25)

$$\begin{aligned} \underline{x}_A(t) &= -E_1^{-1} D_1 \underline{x}_1(t) - E_1^{-1} F \underline{u}(t) \\ &= XAX1 \underline{x}_1(t) + XAU \underline{u}(t) \end{aligned} \quad (5.2.7.)$$

There is an option to punch out only a reduced matrix XAX1R for SIMULA and SIMSAT in order to save storage in these two programs.

XAXLR includes only the rows necessary for those components of x_A needed in SIMULA and SIMSAT.

WRIMX : Prints any real matrix arranging it in rows.

DRAWMX : Draws any real matrix up to 31 columns (no limit in number of rows) with the following symbols:

X for a non-zero element

. for a zero element

where "zero" is defined as 10^{-10}

DRAWLM : Draws (with the same symbols as DRAWMX) the complete linear models evolving in MXEV2X1:

(i) the basic model, i.e. eqs. (3.2.18) to (3.2.21)

(ii) the equivalent linear model (X1), i.e. eqn. (3.2.26)

(iii) the comprehensive linear model (X2), i.e. eqn. (4.2.33)

SCAN : Routine called by DRAWLM.

READ : Reads in as input the data punched out by the previous program, SSNL2D.

OINPUT : Prints out the data read in, which is a list of the state variables, inputs, algebraic variables (implicit and explicit) time delays and outputs. It also prints a brief explanation of the time delay approximation chosen.

UIN, STAAL, YTAU, EXPL : routines called by OINPUT, identical to those in SSNL2D.

PCH2013 : Punches out the two matrices A_f and B_f from the final comprehensive model (eq. 3.2.33), to be used as input data for the next program, REGULA.

AMX, BMX, BDMX, CMX, DMX, EMX, EDMX, FMX : In these routines the matrices of the basic model corresponding to eqs. (3.2.18, 3.2.19) are set. The individual elements have been calculated by hand according with the linearization procedure described by eqs.

(3.2.4., 3.2.5). Printing and drawing is done on request.

PMX, PAMX, QMX, GMX, GAMX, HMX : the matrices of the time delay approximations corresponding to eqs. (3.2.20., 3.2.21) are set in this routine.

AlMX, BlMX, ClMX, DlMX, ElMX : compute the matrices corresponding to eqs. (3.2.24., 3.2.25) which is an intermediate stage towards the equivalent linear model (X1).

ASMX, BSMX : compute the final matrices for model X1 (eq. 3.2.26)

FUN1 to FUN6 : Tables of empirical functions, identical to those in SSNL2D.

SLOPE1 to SLOPE6 : Derivatives of the functions tabulated in FUN1 to FUN6 with respect to the independent variables in each function. The derivation was done in a separate program by simple interpolation, and the results tabulated again.

INTPOL : straight interpolation routine for use with FUN1 to FUN6 and SLOPE1 to SLOPE6, identical to the one used in SSNL2D.

CAUTION : Prints a message at the end of the output, warning the reader that $\underline{u}_0$ in the final comprehensive model (eq. 3.2.33) is not equal to $\underline{u}$ in the previous model because:

- (i) it incorporates several new inputs entered in FDBCK1
- (ii) some of the inputs $\underline{u}_i$ are involved in feedback loops incorporated in FDBCK1, and therefore their counterpart $\underline{u}_{bi}$ must be considered as an additional reference point, and is therefore different from $\underline{u}_i$.

IMBEDV to INVERT : matrix manipulation routines to be described below. Several of them were taken from the library of the Control Section (I.C.), the remainder written in the course of this work.

They all refer to real matrices and vectors.

IMBEDV : Transfer of a vector (column or row) to a partitioned matrix.

IMBED : Transfer of a submatrix to a partitioned matrix.

EQUVRR : Equivalence of two vectors.

UNIT : Produces the identity matrix.

CARVUP : Transfer of a submatrix from a partitioned matrix.

MULMXV : Multiplication of a matrix and a column vector.

ADDV : Addition of two vectors.

ADD : Addition of two matrices.

MULTI : Multiplication of two matrices.

SUBTR : Matrix subtraction.

NULL : Produces the null matrix.

NEGATE : Negative of a matrix.

RORDRM : Switches column or row vectors within a matrix.

INVERT : Given a real matrix B, it finds a matrix A such that $AB = I$. If the matrix B is singular a message is produced. The rank of B is always printed. This is a modified version of the existing routine at the Control Section library, which in turn is a modified version of the SDA3180 routine of the SHARE library (I.B.M.).

XEIGEN to QRT : package for calculating and printing the eigenvalues of a real square matrix. It is a modified version of EIGSYS^[18] or rather of that part of it dealing with eigenvalues, which is based on the 300601 program of the SHARE library. The problems with this program have been discussed in Section (3.3.2).

XEIGEN : Calling routine from the main program.

$N \equiv$ order of square matrix AS(input)

$WNN1, WNN2 \equiv$ auxiliary square matrices of order N

RTASR $\equiv$ real part of eigenvalues computed.

RTASI $\equiv$ imaginary part of eigenvalues computed.

EIG6 : Control routine calling the remainder of the package and also establishing the following checks:

- (i) Trace of input matrix equal to trace of upper Hessenberg matrix equal to sum of eigenvalues.
- (ii) Determinant equal to product of eigenvalues.

DETMT : Computes the determinant of a matrix.

HESSEN : Transforms the input matrix into upper Hessenberg form.

QREIG : Finds the eigenvalues of the upper Hessenberg matrix by calling QRT and testing the convergence of the iterative method, thus deciding when to accept the values found.

QRT : This is the central part of the method. It performs the QR transformation explained in Section (3.2.2). It is a FORTRAN version of the second ALGOL program given in Francis' paper^[19].

5.2.4. Output

This varies according to the requests made through various control points in the main program. The following can be printed (and drawn, in the case of matrices). They will also be punched if it is specifically requested.

- (i) List of steady state values (output from SSNL2D) at the level of power indicated.
- (ii) A brief description of the time-delays approximation.
- (iii) All the matrices computed in routines AMX to EIMX.
- (iv) Matrices XAXL and XAU intended for the computation of $\underline{x}_A(t)$ in SIMULA and SIMSAT (see eq. 5.2.7).

Both matrices are printed and punched.

There is an option that a reduced matrix XAXLR be punched out instead of XAXL and XAU, in order to

save storage space in SIMULA and SIMSAT.

- (v) Matrices A_s and B_s of the equivalent model X_1 (see eq. (3.2.26)).
- (vi) Eigenvalues of A_s and B_s .
- (vii) The drawing of the complete basic linear model and the time delay approximations.
- (viii) The drawing of the equivalent linear model X_1 (eq. 3.2.26).
- (ix) The test of correctness of the basic linear model (see TESTMX in 5.2.3).
- (x) The test of correctness of the equivalent linear model (see TESTMX2).
- (xi) All the matrices computed in FDBCK1, leading to the final comprehensive model X_2 (eq. 3.2.33).
- (xii) Matrices A_f and B_f of eq. (3.2.33). They are also punched for input data to REGULA (see section 5.3).
- (xiii) A warning message described briefly in CAUTION.
- (xiv) The drawing of the comprehensive model (eq. 3.2.33).
- (xv) Steady state values, names and units of the state variables and inputs corresponding to the comprehensive model, as well as of some additional parameters. They are punched and used as input data in SIMULA and SIMSAT (see sections 5.4 and 5.5).

5.3 REGULA (Optimal linear regulator)

See sections (3.4.1) and (3.5).

The compilation time is under 15 seconds on the CDC 6600 and a typical execution time 90 to 100 seconds. The store needed is about 53K (system excluded).

5.3.1 Main program

The input data are the matrices A and B, which correspond to A_f and B_f in eq. 3.2.33, punched in the previous program, MXEV2X1. The dimensions here are:

$$A(N \times N) \quad , \quad B(N \times NU)$$

The principal arrays and their dimensions are explained in the program (*). An important parameter is NY, which is the order of the cost matrix Q_y , with $C(NY \times N)$ (see eqs. 3.4.26 to 3.4.29). The program starts by reading and writing the matrices A and B. In previous versions it also reads C_s (see eq. 3.2.27) but this is not done now, as explained in section (3.4.1, a.ii). It also reads the eigenvalues of A, as calculated in a previous run of the same program. It then proceeds by constructing the state-costate (Hamiltonian) matrix AM. Later it checks controllability and observability when requested to do so. For this purpose it needs to calculate eigenvalues and eigenvectors of A calling EIGVV1 (the "new" routine). The computed eigenvalues are compared with the read-in values and the program ejected if a discrepancy occurs. Next it calculates eigenvalues and eigenvectors of AM in order to implement the algorithm as explained in Section (3.4.1) and calls routine RICCAT in order to perform the remainder of the algorithm. The feedback gain matrix K_r (a reduced version of K) and the matrix K_{dr} from eqs. (3.5.8) and (3.5.10) respectively, are computed, printed and punched for use in SIMULA and SIMSAT, according to eqs. (3.5.2) to (3.5.5) and (3.5.9). The optimally controlled system A_{oc} (eq. 3.5.5) is computed and printed, as is P (eq. 3.5.7). Two matrices may optionally

(*) Many equivalences of arrays are found; an effort was done to save memory space, since this is a rather store-expensive program (about 52K excluding system space).

be punched out: P and Q_{oc} where

$$Q_{oc} = Q + K^T R K \quad (5.3.1)$$

It is easily shown that the Liapunov equation (3.4.14) holds

if Q is replaced by Q_{oc} . (*)

The eigenvalues of P are then computed in order to assess if the matrix is indeed positive-definite. Finally the eigenvalues of A_{oc} are computed and then compared with those of A in the routine COMPEI.

This comparison gives a quick idea of what has changed in the "optimally controlled" system from one case to the next.

The product of the eigenvalues of A_{oc} and the costing elements of Q and R are punched for use in SIMULA and SIMSAT.

(*) This optional facility has been incorporated in order to provide data for other work which is not directly linked with this study.

5.3.2 List of subroutines

SCSMX	COMPEI
RICCAT	
PCHKKD	MULTI
CTROBS	ADD
EIGVVL	NULL
HESSEN	NEGATE
QREIG	IMBED
QRT	CARVUP
ASQR	UNIT
EIGVEC	TRANSP
DELS	EQUVC
WRCMX	EQUVC
DRCMX	EQUVCX
WRIMX	TRNSPC
DRRMX	MULCOM
WRIEIG	MLMXDC
ORM2013	
IRM2013	
ICV2013	
PCM2013	MULCR
RCM2013	MULRC
PRM2013	INVCOM
RRM2013	INCOMA
PCV2013	MATIN1
RCV2013	

5.3.3 Description of subroutines

SCSMX : In this routine the Hamiltonian or state-costate matrix M (eq. 3.4.21) is constructed.

The choice of Q and R is read in as data.

Only two inputs are used for control, u_5 and u_6 . The state variables costed are at present $x_1, x_4, x_{10}, x_{18}, x_{21}, x_{22}, x_{23}, x_{30}$. For their meaning see section (2.1). Of course SCSMX is constructed in such a way that any new choices are feasible. In the first series of runs the eight variables mentioned were costed, but this number was gradually reduced to two variables only ($x_{10} = p_d, x_{30} = \partial f$) after it was found that the trial-and-error procedure of computing the "optimal regulator" and then simulating the new system obtained became easier with less states being costed, and easiest with two states only, controlled by two inputs ($u_5 = Y_M^1, u_6 = Y_{TCV}^1$).

Some of the necessary matrix manipulations are performed inside the routine, with short, special formulas, instead of calling the more general routines which are in the program. This was done in order to save storage (see footnote in 5.3.1).

RICCAT : Performs all the operations indicated in the algorithm of section (3.4.1) starting with step (iii) and computes K_{dr} (eq. (3.5.10) of section (3.5)). The routine receives as arguments, the matrix M (constructed in SCSMX) and its eigenvectors W .

There are several control points which, if requested to do so from the main program, print or punch various matrices such as W_{11}, W_{21} (eq. 3.4.22), the Riccati matrix P (eq. 3.4.23), the matrix SSO resulting from computing the Riccati equation (3.4.10) which should be nil. Three of the four tests indicated in the

algorithm are performed inside the routine, namely:

- (i) P should be real : those elements which have a ratio of imaginary over real part greater than 10^{-7} are printed, and if the ratio is greater than 10^{-4} a special warning is printed.
- (ii) P should be symmetric : the absolute values of symmetric elements are compared and if their fractional difference is greater than 10^{-7} , they are printed. Same special warning as in (i).
- (iii) SSO (i.e. the matrix Riccati equation) should be nil. Elements greater than 10^{-7} are printed, with the same special warning as above.

The real matrix P is computed by averaging the symmetric elements of the complex matrix and neglecting the imaginary part.

The feedback gain matrix K is computed (eq. (3.4.11) or (3.5.6)) and the reduced matrix K_r (eq. (3.5.8)) printed and punched. So is K_{dr} (eq. 3.5.10).

The optimally controlled matrix A_{oc} is computed and printed.

As discussed in section (3.4.1), the complex matrix W_{11} which has to be inverted is sometimes numerically singular (this depends on the precision of the inversion routine). When W_{11} is singular, both W_{11} and W_{21} are printed, regardless of the request from the main program as expressed in the corresponding control point.

RICCAT calls INCOMA for inversion.

PCHKKD : This routine picks the rows of K and K_d corresponding to the control inputs, thus producing the reduced matrices K_r and K_{dr} (eqs. (3.5.8), (3.5.10)). They are printed and punched for use in SIMULA and SIMSAT.

CTROBS : The routine tests controllability and observability (see section 3.4.1(b)).

It computes the matrices E and F_q of equations (3.4.35, 3.4.36) and then tests if E has at least one non-zero element in each row (controllability) and if F_q has at least one non-zero element in each column (observability).

Furthermore, since not all inputs are controlled, the test of E is confined to the columns corresponding to the controls only. This is done automatically, and no change is needed if the controls are changed in SCSMX.

The matrices E and F_q are printed and so are the numbers of the E -rows or F_q -columns, which lack elements greater than 10^{-2} . (This value has somewhat arbitrarily been defined as a "non-zero" element).

EIGVV1 to DELSP : package for calculating and printing eigenvalues and eigenvectors of a real square matrix. This is the "new" routine discussed in section (3.3.3). It is a modified version of EIGSYS^[23] from Northwestern University, Evanston, Illinois.

EIGVV1 : This is the control routine calling the remainder of the package, and performs all the operations which are not specifically executed in those routines. It also constructs the complete eigenvector-matrix, either regular (right hand) or transposed (left hand) and prints it out if so requested by the main program. It prints out all the final eigenvalues.

Briefly: EIGVV1 calls HESSEN and QREIG to calculate the eigenvalues, DELSP for the determinant, and ASQR and EIGVEC to calculate the eigenvectors. The trace and the determinant of the input matrix A is compared with the sum and product, respectively, of the eigenvalues. The right hand eigenvectors $\underline{w}_i$ and the left

hand eigenvectors $\underline{t}_j$ are computed, using the inverse iteration method and, if so requested, printed out along with their respective residual vectors $(\underline{A}\underline{w}_i - \lambda_i \underline{w}_i)$ and $(\underline{t}_j \underline{A} - \lambda_j \underline{t}_j)$. For the same request, the squares of the residual vectors are printed for convenience in checking the accuracy of the eigensystem. After finding each eigenvector, the Rayleigh quotient scheme is used to correct the original eigenvalue, which, in turn, produces a corrected E-vector along with new residuals. A second correction is applied which produces a further refinement of the E-value only. The sum and product of the corrected E-values are printed.

HESSEN : Transforms the input matrix into upper Hessenberg form. This routine is almost identical to HESSEN of the "old" EIGSYS (See sections 3.3.2 and 5.2.3). Maximum dimension 200.

QREIG, QRT : Almost identical to QREIG, QRT of the "old" EIGSYS (see sections 3.3.2 and 5.2.3).

ASQR : To square matrix A. This routine is needed for the EIGVEC routine to compute the eigenvectors^(*).

EIGVEC : To find the real or complex eigenvectors of a real square matrix. A control parameter allows the user to call for either the right-hand or the left-hand eigenvectors or both, the difference in time between finding both eigenvectors or one being not great. The method is a modified version of Wilkinson's inverse iteration method.

DELSF : Find the determinant of a matrix.

(*) ASQR is more complicated than need be, because it is still the unmodified version of the subroutine of EIGSYS from Northwestern University, where the program is on a disc.

WRCMX : Prints any matrix of complex vectors, column-wise.

It optionally prints the eigenvalues at the bottom of the corresponding eigenvector, if eigenvectors are printed.

DRCMX : Draws complex matrix with dots (zeros) and crosses (non-zeros). Maximum number of columns 31.

WRIMX : Prints any real matrix, row-wise.

DRRMX : Draws real matrix with dots and crosses.

Maximum number of columns 99.

WRIEIG : Prints complex eigenvalues.

ORM2013 : Writes a real matrix on an output file numbered as in arguments list. Intended to write on tape, punch or paper, as requested. Format is (4E20.13), each row commencing on a new card.

IRM2013 : Reads a real matrix from an input file, selected as above. Format as above.

ICV2013 : Reads a complex vector from an input file, selected as above. Format as above.

PCM2013 : Punches a complex matrix, format as above.

RCM2013 : Reads a complex matrix, format as above.

PRM2013 : Punches a real matrix, format as above.

RRM2013 : Reads a real matrix, format as above.

PCV2013 : Punches a complex vector, format as above.

RCV2013 : Reads a complex vector, format as above.

COMPEI : Compares the eigenvalues of the original matrix A with the "optimally" controlled A_{oc} , as well as the two products of E-values.

MULTI, ADD, NULL, NEGATE, IMBED, CARVUP, UNIT : Manipulation routines for real matrices, identical to those described in section (5.2.3).

TRANSP : Finds the transpose of a real matrix.

EQUMC : Equivalence of two complex matrices.

EQUVC : Equivalences of two complex vectors.

EQURCX : Given a real matrix, returns a complex matrix with zero imaginary part.

TRNSPC : Finds the transpose of a complex matrix.

MULCOM : Product of two complex matrices.

MLMXDC : product of a square complex matrix A and a diagonal complex matrix Λ , with the two options $A \Lambda$ and ΛA .

MULCR : Multiplies a complex matrix by a real one, result is a complex matrix.

MULRC : Multiplies a real matrix by a complex one, result is a complex matrix.

INVCOM : Inversion of a complex matrix. This is a modified version of INVERT (inversion of a real matrix) described in Section (5.2.3).

INCOMA : Inversion of a complex matrix and solution of a set of linear (complex) equations. This routine is better than INVCOM in the sense that for matrices which are nearly singular INCOMA is successful in cases where INVCOM fails^(*). The method of INCOMA is described in section (3.4.1(a.111)) and was taken from the CERN library [36]. INCOMA calls MATINI.

MATINI : Inversion of a real matrix and solution of a set of linear equations. This routine was taken from the CERN library [36] and is written in COMPASS language instead of FORTRAN. It is suitable for use on CDC computers (6000 series) where it is read in the same way as a FORTRAN routine^(**).

(*) INVCOM is not as good as INCOMA, but is still remaining in the program because it is called from a number of routines which were not changed for that purpose.

(**) A FORTRAN version might be available directly from CERN.

5.3.4 Output

Varies according to the requests made through various control points in the main program.

Printed (and drawn, in the case of matrices):

- (i) Matrices A and B. They correspond to A_f and B_f in eqs. (3.2.33), i.e. the uncontrolled system.
- (ii) Complex eigenvalues of A, computed and punched out in a previous run, read in for the current run.
- (iii) Matrix C (eq. (3.4.26)) constructed in SCSMX.
- (iv) Matrix M (called AM) of eq. (3.4.21) constructed in SCSMX.
- (v) Eigenvalues of A, computed in the current run and punched out.
- (vi) Matrices E and F_q of eq. (3.4.35, 3.4.36) and the tests of controllability and observability.
- (vii) Eigenvalues of the state-costate matrix AM.
- (viii) Matrices W_{11} and W_{21} (eq. 3.4.22).
- (ix) Test of the real condition of matrix P (eq. 3.4.10).
- (x) Test of the symmetric condition of matrix P.
- (xi) Matrix P.
- (xii) Matrix SSO resulting from computing equation 3.4.10.
- (xiii) Test of the null condition of matrix SSO.
- (xiv) Matrix KR, i.e. linear feedback gain (eq. 3.5.8), punched out for use in SIMULA and SIMSAT.
- (xv) Matrix KDR (eq. 3.5.10), punched out for use in SIMULA and SIMSAT.
- (xvi) Matrix A_{oc} , i.e. the optimally controlled system.
- (xvii) Eigenvalues of P to test positive-definiteness.

- (xviii) Eigenvalues of A_{oc} .
- (xix) A comparative table of eigenvalues of A and A_{oc} .
- (xx) PROAOC, the product of the real parts of the eigenvalues of A_{oc} , punched out to be used in SIMULA and SIMSAT for checking.
- (xxi) The costs set for Q and R , punched out for use in SIMULA and SIMSAT.
- (xxii) A further option which is never needed in this program but which it is possible to use due to the existence of EIGV1, is printing out the final eigenvector matrix of any input matrix, as well as the "current" eigenvectors individually computed and improved after the Rayleigh quotient refinement.

5.4 SIMULA (Free transients computation and display)

See section (3.6).

The compilation time is about 15 second on the CDC 6600.

The execution time is 10 seconds plus 10 seconds for each simulation, typical total is 30 sec. Store needed, 35K (system excluded).

5.4.1 Main Program

The purpose is to compute $\underline{x}(t)$ for a given initial excitation $\underline{x}(0)$ or a step input (*) of $\underline{u}_d(t)$. Equations (3.6.9) and (3.6.10), using eigenvectors/eigenvalues, are repeated here:

(a) for uncontrolled system A

$$\underline{x}(t) = W e^{\Lambda t} V \underline{x}_0 + W \Lambda^{-1} [e^{\Lambda t} - I] V [B_c \underline{u}_c + B_d \underline{u}_d] \quad (5.4.1.)$$

(*) For input different than step, such as impulse or ramp, the equations (5.4.1, 5.4.2) are slightly different.

- (b) for optimally controlled system A_{oc} , and writing,
 for the sake of brevity, W, V, Λ instead of $W_{oc}, V_{oc},$
 Λ_{oc}

$$\underline{x}(t) = We^{\Lambda t} V \underline{x}_0 + W \Lambda^{-1} [e^{\Lambda t} - I] V (B_{cdu} \underline{k} + B_{dd} \underline{u}_d) \quad (5.4.2.)$$

For each t , $\underline{x}(t)$ is computed and then $\underline{u}_c(t)$ with eq. (3.6.8).

Then several additional parameters are computed called $\underline{x}_{tr}(t)$.

All the computed parameters ($\underline{u}_c(t)$ is not computed in case(a)) are then printed out and most of them displayed graphically along the length of the computer paper.

Then a table of constraints is printed, where the peak values and the bounds of the constrained parameters or of their derivatives are printed. A warning message is printed when a constraint is violated.

Finally the "weighted-gain" matrix with the dominant values indicated is printed (see section 3.8).

A number of options are open as read in on a "keys" card before the data. Other options are inserted into the main program. Since many runs are made with this program, as well as with REGULA and SIMSAT, they are written in binary records on a tape and therefrom called, and they cannot be changed unless the main routines are run in source language. But this is to be avoided in order to reduced the printout and so enable the program to run on a higher priority turn-around in the computer. Most of the options are also geared towards reducing the printout.

The options in the "keys" card for SIMULA are:

- (i) IPD = 0, XAXIR, B, A, AOC, BD, BC not printed.
- (ii) IQUICK = 1, KR and KDR not printed, simulation for excitation No. 5 not plotted.

- (iii) $MI = 5$, KR , KDR , $PROAOC$ (product of eigenvalues of A_{oc}) and costs read from cards.
- $MI = 3$, KR , KDR , $PROAOC$, costs read from tape No. 3.
- (iv) $IKDU = 1$, k_{du} to be read in, to test influence of fixed added control.
- (v) $MP = 5$, $XAXIR$, list of steady state values, names and units of parameters, B , A , read from cards. These items represent the "permanent" data in the sense that they do not change for different "cases" of the regulator. They are "permanent" for a fixed level of power and for a fixed case of $FDBCK1$ in $MXEV2X1$ (see section 5.2.3).

$MP = 2$, "permanent" data read from tape No. 2.

- (vi) $KRED = 1$, not all state variables are fed back, hence read in filter pattern $KRFILT$ (see section 3.8).
- (vii) $IDC = 1$, D.C. compensation computed such that NC (number of control inputs) variables settle at 0. This option is to be used only when $KRED = 1$ (see above).

Before the input data are read in, and for the case $IDC = 1$, the numbers of the variables chosen to be forced to 0 by D.C. compensation are read in.

The input data are, in the following order:

- (i) Matrices $KR(NC,N)$ and $KDR(NC,N)$, produced by $PCHKKD$ in $REGULA$ (see section 5.3.4).
- (ii) Matrix $XAX1R (NAR,NL)$, produced by $XAX1U$ in $MXEV2X1$ (see section 5.2.4).
- (iii) List of steady state values, names and units of $\underline{x}_{tr}(NXTR)$, $\underline{x}(N)$ and $\underline{u}(NU)$, in that order, produced by $NEWXU$ in $MXEV2X1$ (5.2.4).

- (iv) Matrices $B(N,NU)$, $A(N,N)$ produced as B_f , A_f by
MXEV2X1 (5.2.4).
- (v) PROAOC (product of eigenvalues of matrix A_{oc})
produced by REGULA (see section 5.3.4).
- (vi) Costs set for the matrices Q , R in REGULA, produced
by REGULA (see section 5.3.4).
- (vii) KRFILT, filter pattern set to return only a reduced
number of state variables for feedback (optional).
- (viii) KDU, values of k_{du} set to test their influence
(optional).

After reading in the above data, the program proceeds to:

- (i) The matrix B is broken up into its components B_c and
 B_d . For this purpose it is necessary first to reorder
 B such that the columns of B_c (corresponding to $\underline{u}_c$) be
at one end of B . Routines RORDRM and CARVUP are used.
- (ii) As a consequence of (i) it is also necessary to
reorder the steady state values, names and units of $\underline{u}$.
This is done in RORDUS, where the s.s. values, names
and units of $\underline{u}_c$ are generated as well.
- (iii) Construct $A_{oc} = A - B_c K_r$ if the controlled system is
to be simulated (K_r may be filtered if $KRED = 1$).
- (iv) Calculate eigenvalues/eigenvectors of A_{oc} (or A , if
the uncontrolled system is to be simulated) i.e. Λ ,
 W , V . (Λ is called RTC).
- (v) Compute $W\Lambda V$ and compare the resulting complex matrix
with the original A or A_{oc} using routine COMPCR. This
comparison was felt to be necessary as giving an idea
of the accuracy of the simulation. Compute Λ^{-1}
(RTCIN).

- (vi) The type of excitation is set by assigning a certain number to it. From here on a loop starts which begins with a different excitation at each turn. This is done so as to enable the program to simulate the system for several excitations, without having to do all the preliminary work each time.

Set initial excitation $\underline{x}_0$ and step (or other) inputs $\underline{u}_d$ with routine EXCITE.

In the same routine EXCITE:

- a. Set KCD (see eq. 3.5.11) for a special type of excitation.
- b. Compute $\bar{\underline{u}}_c = K_{cd} \underline{u}_d$.
- c. Compute $V \underline{x}_0$ for all cases.
- d. Compute $V \underline{B} \underline{u}$ for uncontrolled case (eq. 5.4.1).
- e. For optimally controlled case compute K_{dg} (eq. 3.5.9)

$$\underline{k}_{du} = -K_{dg} \underline{u}_d$$

$$V \underline{B} \underline{u} = V(B_c \underline{k}_{du} + B_d \underline{u}_d) \text{ for eq. (5.4.2)}$$
- f. If IKDU = 1, read in values of $\underline{k}_{du}$ to test their influence. Call KDUCHG for this step.
- g. If IDC = 1, compute $\tilde{\underline{k}}_{du}$ (see eq. 3.8.5), different from $\underline{k}_{du}$, to insure DC compensation. Call DCCOMP for this step.

- (vii) Compute additional parameters with EXTRA.

- (viii) Print initial values of $\underline{x}$, $\underline{u}$ and $\underline{x}_{tr}$.

Print $\bar{\underline{u}}_c$, $\underline{k}_{du}$ and constraints for $\underline{u}_c$.

- (ix) Set maximum time, TMAX, maximum number of time intervals NLIM and initial time.

Set all arrays of routine STORE (to be described later) to zero and initialize.

- (x) From here on, until (xviii), all steps are repeated

for each time.

Set $\underline{x}$ (response to $\underline{u}_d$) and $\underline{x}_1$ (response to $\underline{x}_0$) to zero. Print time.

- (xi) Compute $e^{\Lambda t}$.
- (xii) Compute $S = \Lambda^{-1} (e^{\Lambda t} - I)$.
- (xiii) If there is an input disturbance, compute second term of eq. (5.4.1) or (5.4.2). Check real condition. If requested, print response to input.
- (xiv) If there is a state excitation, compute first term of eqs. (5.4.1) or (5.4.2). Check real condition. If requested, print response.
- (xv) Add both terms of eqs. (5.4.1) or (5.4.2). If requested, print response.
- (xvi) Compute additional parameters with EXTRA. If requested, print.
- (xvii) Store the response for time t calling sub-routine STORE.
- (xviii) Set new time t and go to (x).
- (xix) Print final values of $\underline{x}$, $\underline{u}$ and $\underline{x}_{tr}$.
- (xx) Print all values in matrix form, each column headed by the corresponding time t , if requested.
- (xxi) Display graphically the response of certain selected parameters $\underline{x}$, $\underline{u}_c$ and $\underline{x}_{tr}$.
Print the table of constraints and peak values, mentioned before in this section.
Compute and print the "weighted gain" matrix (see section 3.8) determining the maximum and dominant values of the feedback variables for each control input.

(xxii) Set a new type of excitation and go to (vi).

5.4.2 List of subroutines

FILTER	NULL
KDUCHG	MULTI
CHDATA	SUBTR
DCCOMP	ADD
EXCITE	RORDRM
EXTRA	NEGVR
CONSTR	ADDV
PLOTS	NULLRV
GPH	MULCOM
SCALPL	INVCOM
STORE	MLMXDC
LISTVS	INVERT
TRFVCR	MULCMRV
COMPCR	MULCMCV
WRRMX	MULDVC
WRCMX	EIGVV1
WRRMXT	HESSEN
IRM2013	QREIG
IRV2013	QRT
IIV8A10	ASQR
RCM2013	EIGVEC
RRM2013	DEISP
RCV2013	SLOPE1
RRV2013	INTPOL
RIV8A10	RORDUS
CARVUP	
MULMXV	

5.4.3 Description of subroutines

FILTER : This routine is active when KRED = 1 in the "keys" card (see section 5.4.1). It reduces the number of variables fed back selecting those that yield "dominant" values (see section 3.8). First the filter pattern is read in, then the matrix K_r is changed so that the non-dominant elements of it are nil. The filter-pattern is hand-punched as data, and is the result of observations of a number of cases run. It will be seen that the routine PLOTS yields a table at the end of the program's output, where the maximum value of the "weighted-gain" matrix $[k_{ri,j} \times j_{pk}]$ and the dominant values ($\geq 1/10$ maximum value) are determined.

KDUCHG : This routine is active when IKDU = 1 in the "keys" card (see section 5.4.1). It reads and applies the values of $\underline{k}_{du}$ (fixed added control for a given $\underline{u}_d$) chosen differently from the values computed in EXCITE, in order to test their influence.

CHDATA : Checks particular elements of data, as a final check that the data has been correctly read in. For certain changes in the "permanent" data (see section 5.4.1), this routine must be changed.

DCCOMP : Computes $\tilde{\underline{k}}_{du}$ for D.C. compensation such that a number NC (number of control inputs) of variables chosen settle at

zero value for a step input.

$$\tilde{k}_{du} = -[C A_{oc}^{-1} B_c]^{-1} C A_{oc} B_d u_d \quad (3.8.5.)$$

EXCITE : The initial excitation $\underline{x}(0)$ for the state and the step inputs $\underline{u}_c$ (control) and $\underline{u}_d$ (disturbance) are set in this routine. There can be a step input $\underline{u}_c$ for the uncontrolled system only (eq. 5.4.1). The argument ITYPE in the routine selects automatically, as requested from the main program, 6 types of excitation. Obviously it is possible to insert as many different types of excitation as one wishes.

At present the 6 types are:

(1) $\underline{x}(0) = 0, \underline{u}_c = 0, \underline{u}_d = 0$

This is also the initialization in advance of all other types of excitation.

(2) $x_{10}(0) = 20$

$x_{10} \equiv \text{PD (p.s.i.a.)}$

$u_{d8} = + 10\% \cdot u_{s8}$

$u_{d8} \equiv \text{PE (MW)}$

$u_{s8} \equiv \text{steady state value of } u_{d8}$

(3) $u_{d1} = 10^{-5}$

$u_{d1} \equiv \text{DKBAL } (10^{-6} \equiv 1 \text{ milinile})$

(4) $u_{d8} = + 10\% \cdot u_{s8}$

(5) $x_{10}(0) = 20$

(6) $u_{d8} = - 10\% \cdot u_{s8}$

Types (1) and (2) were used for trying the program.

Type (3) was used to compare the transients with those produced on a similar model with the Winfrith analog "simulator". The

similarity of the results provided a good check for the correctness of the model.

Type (4) should be used with steady state values at least 10% below full power.

Type (6) was used on the model with steady state values at full power, and given the fact that a linear model is used, the results of types (4) and (6) should be closely similar, or rather closely symmetric.

Types (5) and (6) were used on all cases of simulation. As explained in step (vi) of section (5.4.1), in this routine the matrix KCD is set and the values of $\bar{u}_c$, Vx_o , VBu (eq. 5.4.1), K_{dg} , k_{du} , VBu (eq. 5.4.2) are computed.

EXTRA : Computes and prints additional desired parameters, which are not state variables. This routine may need adjustment for different cases of MXEV2X1. At present it is set for cases (1) to (7) in FDBCK1 of MXEV2X1 (see section 5.2.3). The parameters computed at present are:

- (i) $\bar{u}_c(t)$, for controlled system only (eq. 5.4.2), using eq. (3.5.2) with the reduced matrix KR described above. KR is produced in PCHKKD of REGULA (see section 5.3.3).
- (ii) various components of $\bar{x}_A(t)$ using eq. (5.2.7) with the reduced matrix XAXLR instead of the matrices XAX1 and XAU. This matrix XAXLR is produced in the routine XAX1U of MXEV2X1 (see section 5.2.3).
- (iii) several parameters belonging to $\bar{x}_E$, $\bar{y}$ and $\bar{u}_2$ in MXEV2X1. For these parameters the corresponding equations were written by hand, as it was felt to be easier to do so.

One part of the routine is devoted to reading in matrices KR, XAXLR and the steady state values, names and units of all the

components of $\underline{x}_{tr}$.

RORDUS : The steady state values, names and units of $\underline{u}$ are reordered, as explained in step (ii) of the main program.

CONSTR : This rather elaborate routine consists of two main bodies:

- (i) it sets constraints, where they exist, on parameter values and/or derivatives
- (ii) for those parameters "set" in (i), it prints a table of the peak values, the constraints, the steady state values and messages of warning if a constraint has been violated.

Here an important fact must be stressed, namely that the linear model and the simulation of it refer to linear deviations from the steady state, hence to check if a constraint has been violated the following two tests must be performed for each constrained parameter or derivative.

$$(a) \quad UB_i \geq SS_i + pp_i \quad (5.4.3.)$$

$$(b) \quad LB_i \leq SS_i + np_i \quad (5.4.4.)$$

where

$UB_i \equiv$ upper bound of parameter i

$LB_i \equiv$ lower bound of parameter i

$SS_i \equiv$ steady state value of parameter i

$pp_i \equiv$ positive peak of linear deviation of parameter i

$np_i \equiv$ negative peak of linear deviation of parameter i

For certain parameters, the constraint is peak to peak, hence the test to be performed is

$$(c) \quad PPB_j \geq |pp_j - np_j| \quad (5.4.5.)$$

If test (a) or (b), or alternatively (c), fails, the constraint has

been violated. This test is done not in CONSTR but in routine SCALPL to be described later, for convenience. The printing is done in CONSTR with variable format, set through a chain of logical operators which are first set in CONSTR (part (i)), then tested in SCALPL, both routines being called by PLOTS, to be described later. The actual list of values of the constraints will be written in chapter (6) of results.

PLOTS : Plots selected parameters from $\underline{x}$, $\underline{x}_{tr}$ and $\underline{u}_c$ on a scale such as to fit into the range -0.2 to +1.0. The time axis runs along the print-out paper length. The selection of parameters is fixed manually. The scaling is done automatically calling SCALPL. It calls CONSTR at the beginning of the routine in order to set the constraints on certain parameters or their derivatives and calls it again at the end in order to print the constraints-table described in CONSTR.

After calling CONSTR, this routine calls SCALPL, where the scale of each parameter is set according to its positive and negative peak values, and where the violation of constraints is tested. Having done that, the remainder of the routine is used for plotting the selected parameters, in batches of 2 to 5 at a time, each batch headed by a listing of the steady state values and the scales used for each parameter.

Finally this routine computes (in SCALPL) the absolute peak values and prints the "weighted gain" matrix $[k_{ri,j} \ x_{jpk}]$ mentioned in section (3.8), determining the dominant variables for each control input.

GPH : Routine called by PLOTS for each line of printing (i.e. for each time interval) and which performs the actual printing of the symbols once the scaling has been done in PLOTS. This is a

very modified version of the GPH routine in the Control Systems library (Imperial College).

SCALPL : This routine computes the scale of each parameter as asked by PLOTS, in such a way as to fit into the range -0.2 to +1.0. The largest of the absolute values of the maximum positive and negative values fits into the positive (+1.0) part of the plot, setting 1.0 equal to 1.0, 0.5 or 0.2 times the order of the peak value. As mentioned before, the test of violation of constraints is done in this routine.

STORE : stores the values of $\underline{x}$, $\underline{u}_c$, $\underline{x}_{tr}$ and t for each t , in order to plot them after having completed the computation for all times t .

LISTVS : Prints the lists of initial and final values of $\underline{x}_i$ and $\underline{u}_c$, as well as their steady state values. Prints $\bar{\underline{u}}_c$ and $\underline{k}_{du}$.

TRFVCR : Tests if a complex vector is real (i.e. imaginary part relatively negligible) and constructs the real vector.

COMPCR : Compares a complex matrix with a real one, specifically it tests how well, numerically, the following equation holds

$$A = W \wedge V \quad (5.4.6.)$$

The test can of course be applied to any pair of a complex and a real matrix.

The end-product of this routine is a table in which the elements of the complex matrix C and the real matrix R are printed where at least one of the following situations apply (absolute values considered):

- (i) $\text{Real}(C) > 10^{-7}$ and $\frac{\text{Real}(C) - R}{\text{Real}(C)} > 10^{-7}$
- (ii) $(\text{Real}(C) > 10^{-7} \text{ or } \text{Imag}(C) > 10^{-7})$ and $R < 10^{-7}$

(iii) $R > 10^{-7}$ and $\text{Real}(C) < 10^{-7}$

The ratio of condition (i) is printed, and a warning message is printed when the ratio in (i) or the absolute values in (ii) are greater than 10^{-4} .

WRRMX : Prints any real matrix columnwise. No limits in dimensions.

WRCMX : Prints any complex matrix columnwise.

See further description in section (5.3.3.).

WRRMXT : This is an extension of WRRMX, with the time t printed at the top of each column and the names of the parameters printed in place of the first column on the left of each submatrix. In this way a compact print-out of the simulation results is achieved.

IRM2013 : Identical to the routine described in section (5.3.3.).

IRV2013 : Reads a real vector from an input file numbered as in argument list, format (4E20.13).

IIV8A10 : Reads an integer vector from an input file as above, format (8A10).

RCM2013, RRM2013, RCV2013 : Identical to the routines in section (5.3.3.).

RRV2013 : Reads a real vector, format (4E20.13).

RIV8A10 : Reads an integer vector, format (8A10).

CARVUP, MULMXV, NULL, MULTI, SUBTR, ADD, RORDRM : Identical to routines in section (5.2.3).

NEGVR : Negative of a real vector.

ADDV : Adds two real vectors.

NULLRV : Sets real vector to zero.

MULCOM, INVCOM, MLMXDC : identical to routines in section (5.3.3).

INVERT : inverts a real square matrix.

MULCMRV : Multiply complex matrix by real vector, result complex.

MULCMCV : Multiply complex matrix by complex vector.

MULDVC : Multiply complex diagonal matrix by complex vector.

EIGVV1, HESSEN, QREIG, QRT, ASQR, EIGVEC, DELSP : Routines identical to those described in section (5.3.3). Package for the computation of eigenvalues and eigenvectors, discussed in section (3.3.3).

SLOPE1, INTPOL : Identical to routines in section (5.2.3).

5.4.4 Output

(i) Table of options as read in from the "keys" card (see section 5.4.1).

(ii) Optional. Matrix KR for eq. (3.5.2), i.e.

$$\underline{u}_c(t) = -K_r \underline{x}(t) + \underline{k}_{du} \quad (5.4.7.)$$

(iii) Optional. Matrix KDR (eq. 3.5.10) for computing $\underline{k}_{du}$.

(iv) Optional. Matrix XAXLR for computing selected components of $\underline{x}_A(t)$ with a reduced version of eq. (5.2.7), i.e.

$$\underline{x}_{Ar}(t) = XAXLR \cdot \underline{x}(t) \quad (5.4.8.)$$

(v) Optional. Matrix B, before reordering and breaking it up into B_c and B_d for use in eqs. (5.4.1) and (5.4.2).

(vi) Optional. State matrix A of the uncontrolled system.

(vii) For KRED = 1. Filter pattern KRFILT.

Matrix KR after being filtered by KRFILT.

Table of variables fed back to each control input.

(viii) Optional. Matrices B_d and B_c .

- (ix) Optional. Controlled state matrix $A_{oc} = A - B K_c r$.
- (x) Eigenvalues of A_{oc} .
- Product of eigenvalues of computed A_{oc} .
- Product of eigenvalues as produced by REGULA.
- Fractional difference of the above two products. (If larger than 10^{-7} , run is aborted, unless filtering was used).
- (xi) Optional. Table of comparisons between the elements of matrices A_{oc} and $W \Lambda V$. See details in subroutine COMPCR described in section (5.4.3).
- (xii) For the case $IDC = 1$, the values of $\underline{k}_{du}$ and $\tilde{\underline{k}}_{du}$ computed. For the case $IKDU = 1$, the values of $\underline{k}_{du}$ computed and these read-in to test their influence.
- (xiii) List of initial values of all the component parameters of $\underline{u}_d$, $\underline{u}_c$, $\underline{x}$, $\underline{x}_{tr}$ as well as $\bar{\underline{u}}_c$, $\underline{k}_{du}$ and the constraints on $\underline{u}_c$.
- (xiv) Here the actual simulation starts:
- For each time t , the following are printed out.
- (a) time t
- (b) elements of the vector $\underline{x}(t)$, when $\text{Imag}(x_i) > 10^{-7}$ and $\left\{ \frac{\text{Imag}(x_i)}{\text{Real}(x_i)} > 10^{-4} \text{ or } \text{Real}(x_i) = 0 \right\}$.
- The source of excitation (initial state or input disturbance) is pointed out.
- (c) The complete vector $\underline{x}(t)$, $\underline{u}_c(t)$ and $\underline{x}_{tr}(t)$, written now in real form. This option is redundant, and therefore only used when testing the program.
- (xv) List of final values of $\underline{u}_d$, $\underline{u}_c$, $\underline{x}$, $\underline{x}_{tr}$.

(xvi) Optional. Complete results of the simulation, written in compact matrix form, with each column headed by the corresponding time.

(xvii) Plots of the simulation results, in batches of 2 to 5 parameters at a time, as described in routine PLOTS (5.4.3).

(xviii) A table of constraints and their possible violations, as described in routine CONSTR (5.4.3).

(xix) The "weighted-gain" matrix $[k_{r_{i,j}} \ x_{j_{pk}}]$ and a tabulation of the maximum and dominant values.

(xx) Steps (xii) to (xix) are repeated for any other type of excitation requested by the main routine.

5.5 SIMSAT (saturated transients computation and display)

See section (3.7)

5.5.1 Main program

This program is at first sight very similar to SIMULA. Indeed it uses most of the same subroutines. However there are two important points which make it substantially different from SIMULA:

- a. The integration is performed over each time interval in sequence, hence the starting point for $\underline{x}_{k+1}$ is $\underline{x}_k$ instead of $\underline{x}(0)$ as in SIMULA.
- b. The saturation status is checked after each integration interval and the equations changed accordingly.

We shall repeat the system equations (3.7.8), (3.7.9)

$$A_{ocL} = A - \sum_{L-c-r} b_{L-c-r}^{i,iT} \quad \left. \begin{array}{l} L = 1, \dots, 4 \\ q = 1, \dots, 9 \end{array} \right\} \quad (5.5.1.)$$

$$\dot{\underline{x}} = A_{ocL} \underline{x} + \sum_{L-c} b_{L-c}^{i,k} \underline{u}_{dui} + \sum_{q-c} b_{q-c}^{j,u} \underline{u}_{cj} + B_{d-d} \underline{u}_d \quad \left. \begin{array}{l} i, j = 1, 2 \\ i \neq j \end{array} \right\} \quad (5.5.2.)$$

(The numbers quoted are for $NC = 2$ control inputs, as is the case here; in general if $i, j = 1, \dots, NC \geq L = 2^{NC}$, $q = 3^{NC}$).

The recurrence equations (3.7.11), (3.7.12) are repeated here

$$\underline{x}_{k+1} = W_L e^{A_L \delta t} V_L \underline{x}_k + W_L A_L^{-1} [e^{A_L \delta t} - I] \underline{v}_{buq} \quad (5.5.3.)$$

$$\underline{v}_{buq} = V_L [\sum_{L-c} b_{L-c}^{i,k} \underline{u}_{dui} + \sum_{q-c} b_{q-c}^{j,u} \underline{u}_{cj} + B_{d-d} \underline{u}_d] \quad (5.5.4.)$$

As in the case of SIMULA, the parameters $\underline{x}$, $\underline{u}_c$ and $\underline{x}_{tr}$ are computed for each t and stored. At the end of the computation, the final values are printed; the parameters plotted and a table of constraints and peak values printed, as well as the "weighted gain" matrix with the dominant values.

Again as in the case of SIMULA a number of options are open and read in on a "keys" card before the data, whereas other options are inserted into the main program. The options in the "keys" card are:

- (i) IPD, IQUICK, MI, IKDU, MP, KRED, IDC as in SIMULA (see section 5.4.1).
- (ii) ISKIP5 = 1, excitation No. 5 not run at all.
- (iii) TMAX, simulation time limit in seconds.
- (iv) RTMLIM, real time limit in seconds. This is necessary as the length of computation is not exactly predictable, and rather long. Hence a call asking for the time is inserted in the main program, and if it reaches RTMLIM (which is set at 5 seconds less than the time limit on the job card) the

simulation finishes leaving enough time for the printing and plotting of results.

- (v) DT10, DT100, DT1000, DT10000, respectively the integration interval δt after 10, 100, 1000, 10000 seconds of simulation time. Up to 10 seconds the δt 's are set in the main program, and are as follows:

Up to .3 seconds of simulation time, $\delta t = 10$ m sec.. Up to 1 sec, $\delta t = 50$ m sec.. Up to 10 sec, $\delta t = .1$ sec. These intervals can of course be changed at will, but from experience they seem the right ones. It must be recalled that since the integration is performed from t_k to t_{k+1} and the control values may cross the saturation line at any time, the shorter δt is the more accurate the simulation will be. On the other hand, economy in computing time has to be taken into account: for each δt the computing time is about 60 m sec.

The input data are as in SIMULA (see section 5.4.1). After reading them in, the program proceeds to:

- (i) Break up B into B_c and B_d as in (i) of SIMULA (section 5.4.1).
- (ii) Reorder the values of $\underline{u}$ as in (ii) of SIMULA.
- (iii) Set code number 1 to KASAT and KSUMB, which correspond to L and q of eq. (5.5.3) and are numbered as follows:

Saturation code table: UBUC(1) $\equiv$ upper bound of u_{c_1}

LBUC(1) $\equiv$ lower " " "

UBUC(2) $\equiv$ upper " " u_{c_2}

LBUC(2) $\equiv$ lower " " "

V $\equiv$ constraint violated

UBUC(1)	LBUC(1)	UBUC(2)	LBUC(2)	KASAT	KSUMB
-	-	-	-	1	1
-	-	V	-	2	2
-	-	-	V	2	3
V	-	-	-	3	4
-	V	-	-	3	5
V	-	V	-	4	6
V	-	-	V	4	7
-	V	V	-	4	8
-	V	-	V	4	9

Compute $\Lambda_1, W_1, \dots, \Lambda_k, W_k$ for A_{ocL} (eq. 5.5.1).

Compute Λ_1^{-1}, V_1 . Store $\Lambda_1, W_1, \Lambda_1^{-1}, V_1$ in RTC, W, RTCIN, V, respectively. Call EIGSAT for this step.

(iv) Set type of excitation, initial $\underline{x}(0)$ and disturbance input

$\underline{u}_d$ calling routine EXCITS and:

a. Set KCD (eq. 3.5.11)

b. Compute $\underline{\bar{u}}_c = K_{cd} \underline{u}_d$

c. Compute K_{dg} (eq. 3.5.9), $\underline{k}_{du} = -K_{dg} \underline{u}_d$

d. For IDC = 1, compute $\tilde{\underline{k}}_{du}$ (eq. 3.8.5)

e. Compute $\underline{v}_{bu_1} = V_1 (B\underline{u})_1 = V_1 [\underline{b}_c^1 \underline{k}_{du_1} + \underline{b}_c^2 \underline{k}_{du_2} + B_d \underline{u}_d]$ (eq. 5.5.4)

This is for the non-saturating case, i.e. $L=q=1$.

(v) Compute additional parameters with EXTRA.

(vi) Check if $\underline{u}_c(0)$ violates constraints. If so, set new code numbers and change $\underline{u}_c$ to $\tilde{\underline{u}}_c^{(*)}$. Call SATCOD for this step.

(vii) Print initial values of $\underline{x}, \underline{u}, \underline{x}_{tr}$. Print $\underline{\bar{u}}_c, \underline{k}_{du}$ and constraints for $\underline{u}_c$.

(viii) Set TMAX (maximum time), NLIM (max. number of time intervals

(*) $\tilde{\underline{u}}_c \equiv$ value of $\underline{u}_c$ saturated.

to be plotted), initial time t , initial interval δt . Set all arrays in STORE to zero.

- (ix) From here on, until (xix) all steps are repeated for each time interval.

Check if new code number KASNEW is equal to the previous one (KASAT). If not, update eigenvalues/vectors and inverses calling UPDEIG.
- (x) Check if new code number KSBNEW is equal to the previous one (KSUMB). If not, update $\underline{v}_{buq}$ to satisfy eq. (5.5.4).
- (xi) Compute $e^{\Lambda \delta t}$.
- (xii) Compute $\underline{s} = \Lambda^{-1} (e^{\Lambda \delta t} - I) \underline{v}_{bu}$. If $\lambda_i = 0 \Rightarrow s_i = \delta t \cdot v_{bu_i}$.
- (xiii) Compute second term of eq. (5.5.3). Check real condition. Print if requested.
- (xiv) Compute first term of eq. (5.5.3). Check real condition. Print if requested.
- (xv) Add both terms of eq. (5.5.3). Print if requested.
- (xvi) Compute additional parameters with EXTRA. Print if requested.
- (xvii) Check if $\underline{u}_c$ violates constraints. If so, proceed as in step (vi). Print $\underline{u}_c$, $\underline{x}_{tr}$ if requested.
- (xviii) If the time interval coincides with the preestablished times of plotting, store the response for time t .
- (xix) Set new δt and go to (ix).
- (xx) Print final values of $\underline{x}$, $\underline{u}$ and $\underline{x}_{tr}$.
- (xxi) Print all values in matrix form if requested, each column headed by the corresponding t .
- (xxii) As in step (xxi) of SIMULA.
- (xxiii) Set a new type of excitation and go to (iv).

The store needed is about 45K (system excluded). The execution time varies with the width of the integrating interval δt , typical cases varying between 100 and 800 seconds on the CDC 6600, for a simulation time of 10000 seconds.

5.5.2 List of subroutines

EIGSAT	CARVUP
SATC0D	MULMXV
UPDEIG	NULL
UPDVBU	MULTI
EXCITS	SUBTR
EQU MC	ADD
EQUVC	RORDRM
FILTER	NEGVR
KDUCHG	ADDV
CHDATA	NULLRV
DCCOMP	MULCOM
EXTRA	INVCOM
RORDUS	MLMXDC
CONSTR	INVERT
PLOTS	MULCMRV
GPH	MULCMCV
SCALPL	MULDVC
STORE	EIGVVI
LISTVS	HESSEN
TRFVCR	QREIG
WRRMX	QRT
WRCMX	ASQR
WRRMXT	EIGVEC
IRM2013	DEISP
IRV2013	SLOPE1
IIV8A10	INTPOL
RCM2013	
RRM2013	
RCV2013	
RRV2013	
RIV8A10	

5.5.3 Description of subroutines

EIGSAT : Computes the four possible A_{ocL} (eq. 5.5.1), for each of them the eigenvalues/vectors Λ_L , W_L , and for $A_{oc_1} = A - B_c K_r$ the inverses Λ_1^{-1} , W_1^{-1} . Stores all the Λ_L , W_L . In addition it stores Λ_1 , Λ_1^{-1} , W_1 , W_1^{-1} in Λ , Λ^{-1} , W and V respectively.

SATCOD : Checks if $\underline{u}_c$ violates any constraint and if so, changes it to the corresponding saturated value $\tilde{\underline{u}}_c$ sets corresponding code numbers to KASNEW and KSBNEW according to the saturation code table shown in step (iii) of the main program.

UPDEIG : Updates eigenvalues/vectors and their inverses according to the saturation code KASAT, i.e. stores Λ_L , W_L into Λ , W and computes the new Λ^{-1} , W^{-1} .

UPDVBU : Updates the new vector $\underline{v}_{buq}$ (eq. 5.5.4) according to the new values of $L(KASAT)$ and $q(KSUMB)$.

EXCITS : This routine is very similar to EXCITE (see section 5.4.3). On top of setting the initial excitation $\underline{x}(0)$ and $\underline{u}_d$, it sets KCD and computes $\bar{\underline{u}}_c$, KDG, $\underline{k}_{du}$, $\underline{v}_{bu_1}$, as explained in step (iv) of the main program.

EQUVC, EQUVC : Identical to the routines described in section (5.3.3).

FILTER, ..., INTPOL (Remainder of routines in the list of section 5.5.2) : Identical to the routines described in section (5.4.3).

5.5.4 Output

It is the same as for SIMULA (see section 5.4.4) with the following additions:

- (i) The table of options as read from the "keys" cards include ISKIP5, TMAX, RTMLIM, DT10, DT100, DT1000, DT10000, explained in section (5.5.1).

- (ii) Optional. The four matrices A_{ocL} instead of A_{oc} only.
- (iii) The eigenvalues and ranks of the four matrices A_{ocL} .

5.6 Description of analog programs

During the course of this research a number of simulation runs were done on the analog computer at the plant site. This computer is used at Winfrith for the alternate objectives of general purpose and in the role of "simulator" of the plant, in which case it fulfils the following functions:^[6]

- (i) As a device for giving the plant operators initial training.
- (ii) As a device on which theoretical transient experiments can be carried out prior to practical dynamics experiments on the SGHWR itself.
- (iii) As a device which can easily be modified for studying control system design modifications to the present SGHWR or for design of control systems on future reactors.

It is in this last capacity that the simulator was used for the purposes of this study.

The "simulator" comprises:

- (i) A general purpose 2-Console 231R Pace analog computer.
- (ii) A reactor control desk with instruments and controls and a separate panel attached to it.
- (iii) A logic and interface unit.

The Pace complement used for the simulator comprises: 200 amplifiers, 300 potentiometers, 17 quarter square multipliers, 13 servo-multipliers, 17 comparators and 7 function generators. In addition there are three Elliot transportation delay drums and

two 8-channel chart recorders.

As a whole the simulator represents the complete non-linear model of the plant, including the logical units for emergency trips and other conditions; these units were not used in the runs done for this study.

Two different "programs" were used on the simulator for our purposes:

- (i) The plant stripped to its essentials, in order to compare the correctness of the linear model before attempting to add control systems to it. This was case (7) in FDBCK1 (see section 5.2.3) which includes feedwater control only.
- (ii) The plant incorporating the controller for various "cases" of regulator design, including free or saturated simulation, and complete or reduced set of feedback variables. The non-controlled model here was case (3) of FDBCK1 which includes conventional (existing) feedwater control and load-following.

CHAPTER 6

RESULTS

What results are we really looking for? Ideally we want to obtain the feedback gain matrix K for a large (infinite?) number of different levels of operation. Even so we would then only be obtaining an approximation to an optimal controller because we are employing a set of linearized models, while the plant is really non-linear. Since linearity is a "reasonable" approximation for small variations, it follows that a large number of steady states ought to be calculated and an equal number of linear models computed for each steady state. Each model would then have an "optimal" dynamic performance if a suitable matrix K were applied to it. The "optimal" performance would then hold for small variations around each steady state. But since the variations are - or may be - large, the approximations are necessarily rough^(*). The most one can hope for is to obtain "optimal" regulators which when applied to the real non-linear system will give it a better performance, according to the control objectives outlined in section (1.3). The procedures of computation, for each level of operation, are as follows:

- (i) Find the steady state. This is done by running SSNL2D once (if the run is successful).
- (ii) Find the linear model by running MXEV2X1 once with the output of SSNL2D as input data.

(*) One way out of this problem is a piecewise linear approach, switching from one model to the other as the signals grow.

- (iii) Solve the optimal regulator problem by running REGULA with the output of MXEV2X1 as input data and setting cost matrices Q for the state and R for the control.
- (iv) Simulate the optimally controlled system by running SIMULA with the output of REGULA as input data. Judge the dynamic performance according to the control objectives and repeat steps (iii) and (iv) as many times as is necessary to obtain acceptable responses.
- (v) Run on SIMSAT selected cases of REGULA which are "acceptable" by SIMULA standards. This being the most realistic simulation the system is subjected to, the best response of SIMSAT will indicate the controller to be adopted.

A major part of the work done in this study consisted in writing, testing and improving the five programs which form the procedure just described.

The steady state was computed for 100% and 75% full power. The linear model was constructed for 100% only and a number of runs were made of REGULA, SIMULA and SIMSAT to find an appropriate feedback gain matrix K for that level. The system was excited with a negative step demand of 10% electric power and separately with an alteration of the steam pressure by 20 p.s.i. The reasoning behind this is that a negative step at 100% should produce a performance roughly symmetric to that corresponding to a positive 10% step at a lower level. The approximation is of course rough, as explained above.

6.1. The steady state

(i) For the 100% level, 7 sets of different initial values were given as input data to SSNL2D, all with successful results^(*), according to table (6.1.1). NI is the number of iterations and NC the number of calls for the function evaluation before the program converges, i.e. satisfies the conditions imposed upon it. SI is the sum of the squares of functions with the initial values and SF with the final values of the variables. The first set of initial values corresponds to the data obtained from the plant site. The other sets are deviations from the first one.

(ii) For the 75% level, the set of steady state values was found, but it was not further used.

(iii) Appendix (A) is a listing of the more important routines of SSNL2D.

(*) Two runs with sets having 5 and 10 initial values with opposite sign, respectively, did not converge after 1000 calls, and gave no evidence of any pronounced minimization taking place.

Set	Initial Values	NI	NC	SI	SF
1	All equal to the values obtained with STEAD6 (Winfrith)	4	45	1.10^{+3}	2.10^{-17}
2	All $\pm 3\%$ off (some values + 3%, some - 3%)	8	55	2.10^{+8}	1.10^{-15}
3	All $\pm 10\%$ off	11	64	1.10^{+9}	4.10^{-14}
4	All $\pm 30\%$ off	18	87	3.10^{+10}	3.10^{-12}
5	From x_1 to x_{18} , $\pm 50\%$ off	44	206	7.10^{+10}	4.10^{-9}
6	All $\pm 50\%$ off	43	183	1.10^{+11}	2.10^{-9}
7	Two values with opposite sign	4	45	2.10^{+11}	3.10^{-15}

Table 6.1.1. Some results of SSNL2D (100% power level)

NI : Number of iterations before the program converges.

NC : Number of calls for the function evaluations before the program converges.

SI : Sum of the squares of functions with the initial values of the variables.

SF : " " " " " " " " final " " " "

6.2 The Linear Model

Only the 100% level has been processed by MXEV2X1. Seven different "cases" in FDBCK1 (see section 5.2.3) have been executed, the output being the complete printout with the checking feature described in TESTMX (see section 5.2.3 for results of linearization), and the punchout of the corresponding matrices A_f , B_f , C_s , as well as a list of steady state values, etc. The results of the linearization are quite acceptable, as can be seen in section (5.2.3) in the description of TESTMX. There is no point in listing the print-output here. The more important routines of program MXEV2X1 are listed in appendix (B). Case (3) in FDBCK1 (i.e. load following and feedwater control but no moderator control) has been used for the optimization and subsequent simulation procedure: the output of MXEV2X1 for case (3) was fed as input to REGULA, together with set data elements of the matrices Q and R for obtaining the optimal regulator for each of many cases of REGULA (*).

(*) The cases in FDBCK1 of MXEV2X1 should not be confused with the cases of REGULA. The cases in FDBCK1 (seven altogether) refer to a particular model, whereas the cases in REGULA are different regulators obtained for different cost matrices Q and R , all for the same linear model (Case 3 in FDBCK1) at a level of operation of 100% full power.

Case (7) was used directly in SIMULA in order to check the transients against those obtained with the Winfrith simulator^(*).

- (*) Case (7) as mentioned in section (5.2.3) is the basic model plus feedwater control but excluding load following and moderator control. It would have been preferable to use case (4) for checking the model: this case includes load following, feedwater control and moderator (or power) control, as existing at Winfrith. This would have been sensible because a comparison of the performance with the existing controls and that with the "optimal" controller could have been made. But unfortunately case (4) yielded an unstable system in our model, and this was confirmed by Winfrith, where the plant with the existing moderator control is indeed unstable when load following is the operating mode (it is stable when connected to the grid, i.e. when the frequency is not a state but an input to the plant).

Following are the equations for the moderator control, which were not listed in section (2.1.3).

$$y_1 = (1.838 - 0.8729\bar{x}_{22})x_1 - 0.8729\bar{x}_1 x_{22} \quad (6.2.1.)$$

$$y_5 = 0.3862\bar{x}_{23}x_{10} + 0.3862\bar{x}_{10}x_{23} \quad (6.2.2.)$$

$$\dot{w}_4 = \frac{1}{\tau_f} w_4 + \frac{1}{\tau_f} y_1 - \frac{0.0029}{2\tau_f} y_5 \quad (6.2.3.)$$

$$u_{25} = K_f w_4 - K_f y_1 - \frac{K_f}{4} u_{27} + \frac{K_f}{2} u_{29} \quad (6.2.4.)$$

Where the meanings of most variables are found in section (2.1) and:

$\bar{x} \equiv$ steady state values of x

$w_4 \equiv$ IPC, power controller variable

$K_f \equiv$ transfer function gain for moderator valve
 $= 30$

$\tau_f \equiv$ moderator valve lag (power controller)
 $= 40$ seconds

6.3 The optimal regulator

The linear model at 100% power level obtained from MXEV2XL (Case 3 of FDBCK1) was applied to REGULA in order to find the best "optimal" regulator to suit the control objectives. Nearly 200 cases were run and the resulting systems A_{oc} applied to SIMULA for transients computation and evaluation. In this way the elements of the matrices Q and R were set at different values on each new case, judging by the performance of the last one^(*). The number of cases would have been appreciably smaller if it were not for the long "turn-around" mentioned in the footnote. Alternatively, the regulator could possibly have been refined further with the same number of cases run. In the end acceptable cases for the controller were found, and one or two will be especially recommended.

The complete list of cases run will be written in appendix F.

(*) In actual fact, due to the exceedingly long "turn-around" time of the computer, several new "cases" were fed into REGULA simultaneously, without waiting for the results. The evaluation procedure mentioned above was thus done on a batch basis, rather than case-after-case.

The purpose of appendix F is to indicate the search directions towards an optimal controller, to be used if and when an extension of the present work takes place. The proper procedure would have been to set systematically series of cases changing one element of cost at a time and repeating for different combinations of the rest of the elements. This would yield "families" of sensitivity tables for each parameter. Needless to say, such a procedure would be unwieldy and enormously expensive in computer time, due to the large number of possible permutations, even if these were restricted to what is considered reasonable.

On the other hand, the procedure used and described above has the disadvantages of a search which is not systematic and a bit haphazard. It is very time consuming for the designer, who finds himself as a kind of human element inside an interactive process which has a very long period due to the long "turn-around". It seems that this procedure would be ideal if used on-line with the computer (possibly combined with visual display).

In this chapter only a few selected cases will be tabulated for judgment. The tables include the following:

The case number, the cost elements q_{ii} , r_{ii} (Q and R were made diagonal to facilitate the search for a better control), the product of the eigenvalues λ of the controlled system, the time constant τ of the mechanical power x_{1s} , and the peak values of the most important parameters involved. In some instances the positive and negative peaks are given, whereas for other parameters only one peak is given, implying that the peak value with the opposite sign is one or two orders smaller and hence irrelevant. The value is underlined whenever a constraint is violated.

As was explained in the description of SCSMX (see section 5.3.3) the number of state variables to be costed was reduced gradually from 8 measurable states to only two, having found that for two costed control inputs it is easiest to search for a better control by costing two states. These were selected to be x_{10} and x_{30} , the steam drum pressure and frequency, respectively. The two control inputs are u_{c_1} (also called u_5) which is the demanded position of the moderator control valve and u_{c_2} (u_6), the speeder motor demanded position of the turbine control valve.

The peak values listed are those of the costed parameters, and in addition x_1 (the normalized neutron flux), x_{21} (the moderator control valve) and the derivative of u_{c_2} . These last 3 parameters are not listed in the complete tables of appendix F.

6.3.1 Controllability, observability, precision and speed

(i) The controllability is not affected by changes in Q or R , hence it needed not to be tested for each case. The test, as described in CTROBS (see section 5.3.3 and eq. 3.4.35) was done and the system is controllable in the sense that all rows of E have at least one element greater than 10^{-2} in columns 5 and 6 (the controlled inputs).

(ii) The observability does depend on Q (see eqs. 3.4.36 and 3.4.29) and hence needs to be checked for each case. In fact from the equations quoted it can be seen that the values of the elements in the rows i of Fq are proportional to the square root of the corresponding cost element q_{ii} , hence they may be forecast. Row number 3 corresponds to the cost of x_{10} (p_d) and it turns out that in this row are the critical elements of Fq , critical in the sense that many of the row's elements are the largest elements

(*) (in eq. 3.4.36)

in their respective columns. It follows that the cost corresponding to x_{10} must have a value high enough for F_q having at least one element greater than 10^{-2} in each column, according to the definition of non-zero given arbitrarily in sections (3.4.1, 5.3.3). The cost thus necessary for x_{10} turned out to be larger than 10^4 . During the first stages of our trial-and-error runs of REGULA-SIMULA a high value was given to $q_{10,10}$ but it was soon found that although x_{10} was thus drastically reduced, it was exceedingly difficult to balance the cost pattern so as to avoid too high values on the other important parameters such as u_{c_1} (YMDEM), u_{c_2} (YTCVDEM) and x_{30} (DF). Therefore most of the cases have a lighter penalty on x_{10} , implying that the "non-zero" element in some of the columns of F_q was of order 10^{-3} rather than 10^{-2} ; but since the value 10^{-2} was chosen arbitrarily anyway, this is no serious setback.

(iii) The precision of the solution of the matrix Riccati equation implies (see section 3.4.1) that P should be real, symmetric, positive definite and satisfy eq. (3.4.10) within reasonable bounds of numerical accuracy. The ways of testing precision was explained in RICCAT (see section 5.3.3). It is not considered desirable to present here extensive tables of accuracy, on the grounds that it would not illuminate the problem more than a qualifying comment would do. On the other hand, while running the programs, the precision was inspected in order to acquire a feeling of the usefulness of going ahead with the simulation. On these general premises then, the following should be said: The ratio of imaginary to real part of the elements of P was smaller than 10^{-7} with the exception of a number (the order of several

tens) of elements where this ratio was larger but rarely more than 10^{-4} . Roughly the same precision occurred with the symmetrical condition of P. The matrix SSO representing eq. (3.4.10) which should be 0, had a very considerable number of elements greater than 10^{-7} and a smaller but significant number of elements greater than 10^{-4} (usually the order of several tens). Occasionally the precision was worse than the above figures, in which cases a closer inspection revealed that the eigenvalues of the Hamiltonian matrix were not all symmetrical with respect to the imaginary axis, i.e. the procedure of solution of the Riccati equation was imprecise from the beginning. Such cases were, of course, dismissed; they usually occurred for very heavy penalties on all the costed parameters.

Finally the test of positive definiteness usually yielded two to four negative eigenvalues of very small magnitude, of an order at least 10^{10} smaller than the largest positive eigenvalues. Summarizing, the precision obtained is reasonably satisfactory for the cases used for simulation. Also, the real proof of quality of the regulator designed does not lie in the accuracy of computation but in the performance yielded in the simulation runs.

(iv) If we define the "speed" of the controlled system as the product of its eigenvalues, it is seen that it increases approximately with the square root of the product of the elements of the diagonal matrix R^{-1} and also with the square root of the product of the elements of the diagonal matrix Q_y . In other words, the "speed" increases by giving lighter penalties to the control inputs and heavier penalties to the state variables.

On the other hand, the constraints listed at the bottom of table (6.3.2), to be compared with the peak values in that same table, are the bounds to the deviations from the particular steady state values (100% of full power) for which table (6.3.2) is valid. Therefore, only in the case of those parameters which happen to have a zero steady state value, are the constraints listed in table (6.3.1) equal to those listed at the bottom of table (6.3.2).

(ii) As mentioned in section (3.7) there are three types of constraints: the so called "hard" constraints such as valve openings, specified constraints such as frequency and pressure deviations and lastly engineering type constraints such as speeds of valves, which could in theory be changed. In the present work the emphasis was concentrated in the first two categories, hence table (6.3.2) lists only a few of the total number of constrained parameters. In the discussion of results it will be seen that this choice was justified.

(iii) In the cases of x_1 (ϕ' , normalized neutron flux) and x_{A_5} (Q' , normalized thermal power) the constraints are peak to peak instead of positive and negative bounds separately. They were calculated with the safety constraints at present active at the plant site: a tripping device operates when the power output exceeds $\pm 13\text{Mw}(\text{elec.})$ from a set reference level. From the steady state equations it emerges that the equivalent of $26\text{Mw}(\text{elec.})$ peak to peak is approximately 0.20 for the normalized flux and 0.24 for the normalized thermal output. These were the constraints adopted, and in all the cases run the two parameters remained well within these bounds.

Let us now look at tables (6.3.1) and (6.3.2) and discuss the results.

Table 6.3.1 List of constrained parameters and/or derivatives
(The steady state values are for 100% power level)

		NAME	UNIT	S.STATE	CONSTRAINTS	
					UPPER	LOWER
CONTROL INPUTS (UC)						
DERIVATIVE OF	UC(1) ==	YMDEN	NORMALIZED	0.E+00	1.000E+00	-1.000E+00
	UC(2) ==	YTCVDEM	NORMALIZED	1.450E+00	3.250E+00	0.E+00
	UC(2) ==	YTCVDEM	NORMALIZED/SEC		3.300E-03	-3.300E-03
STATE VARIABLES						
DERIVATIVE OF	X(1) ==	PHIDASH	NORMALIZED	1.058E+00	2.000E-01	PEAK TO PK
	X(10) ==	PD	PSIA	9.400E+02	9.800E+02	9.000E+02
	X(10) ==	PD	PSIA /SEC			-1.000E+00
	X(11) ==	M	LB	8.452E+04		0.E+00
DERIVATIVE OF	X(12) ==	MW	LB	3.470E+04		0.E+00
	X(15) ==	OMEGA	RAD/SEC	1.516E+02		0.E+00
	X(16) ==	WFW	LB/SEC	3.449E+02		0.E+00
	X(19) ==	YP	NORMALIZED	4.641E-01	1.000E+00	0.E+00
DERIVATIVE OF	X(19) ==	YR	NORMALIZED/SEC		3.300E-02	-3.300E-02
	X(21) ==	YM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00
	X(21) ==	YM	NORMALIZED/SEC		2.000E-01	-2.000E-01
	X(22) ==	HDASH	NORMALIZED	9.200E-01	9.700E-01	
	X(23) ==	YTCV	NORMALIZED	9.501E-01	1.000E+00	0.E+00
	X(28) ==	IF1	INCHES	0.E+00	1.000E+00	
	X(29) ==	IF2		0.E+00	1.000E+00	
	X(30) ==	DF	C/S	0.E+00	1.000E+00	-1.000E+00
ADDITIONAL PARAMETERS						
DERIVATIVE OF	Y(3)-UD(6) ==	DL	INCHES	0.E+00	8.000E+00	-8.000E+00
	U2(4) ==	YRDEM	NORMALIZED	4.641E-01	1.000E+00	0.E+00
	XE(1) ==	DKM	/SEC		3.000E-05	-3.000E-05
	XA(5) ==	QDASH	NORMALIZED	1.014E+00	2.400E-01	PEAK TO PK
	XA(6) ==	DHSC	BTU/LB	1.857E+01		0.E+00
	XA(8) ==	DHB	BTU/LB	4.945E+01		0.E+00
	XA(12) ==	KXCOUT		1.402E+00		0.E+00
	XA(14) ==	KXRBAR		9.535E-01		0.E+00

Case	Costs				Results								
	x_{10}	x_{30}	u_{c1}	u_{c2}	Product of eigenvs.	τ (sec) x_{18} P_{MEC}	Peak values of deviations from s. state						
	p_d	δf	Y_M	Y_{TCV}			u_{c1}	u_{c2}	x_{30}	x_{10}	$\frac{x_1}{\phi}$	$\dot{u}_{c2}$	x_{21} Y_M
0012	10	E4	E4	E2	8.8E-17	.16	<u>+.36</u> <u>-3.4</u>	<u>+.056</u> <u>-1.5</u>	.29	36.	-.12	<u>+18.</u> <u>-.6</u>	<u>-2.9</u>
0029	10	E5	E4	E2	3.E-16	.1	<u>+.36</u> <u>-3.4</u>	<u>+.39</u> <u>-3.4</u>	.04	37.	-.12	<u>+49.</u> <u>-3.2</u>	<u>-2.9</u>
0046	5E2	E6	E6	E3	2.E-16	.1	<u>+.29</u> <u>-2.8</u>	<u>+.36</u> <u>-3.8</u>	.18	<u>42.</u>	-.12	<u>+78.</u> <u>-3.9</u>	<u>-2.4</u>
0049	5E3	E6	E6	E3	5.8E-16	.55	<u>+.51</u> <u>-4.9</u>	<u>+.47</u> <u>-3.9</u>	.81	20.	-.12	<u>+150.</u> <u>-8.5</u>	<u>-4.</u>
0084	1	1	E5	E6	3.E-20	.51	<u>+.06</u> <u>-1.</u>	0. -.09	.23	<u>110.</u>	-.12	<u>+5E-4</u> <u>-7E-5</u>	<u>-.93</u>
0085	10	1	E5	E6	1.1E-19	.51	<u>+.19</u> <u>-1.8</u>	0. -.09	.23	<u>62.</u>	-.12	<u>+5E-3</u> <u>-7E-4</u>	<u>-1.6</u>
0092	2.1	1	E6	E5	1.6E-20	.49	<u>+.07</u> <u>-.65</u>	0 -.098	.43	<u>160.</u>	-.12	<u>+.01</u> <u>-.001</u>	<u>-.62</u>
0096	30	1	E5	E6	2.3E-19	.51	<u>+.26</u> <u>-2.5</u>	0. -.09	.23	<u>48.</u>	-.12	<u>+.014</u> <u>-.002</u>	<u>-2.2</u>
0098	20	1	E5	E5	4.2E-19	.53	<u>+.08</u> <u>-2.2</u>	<u>+.08</u> <u>-.09</u>	.44	<u>52.</u>	-.12	<u>+.07</u> <u>-.003</u>	<u>-1.9</u>
0100	20	1	E5	E6	1.7E-19	.51	<u>+.23</u> <u>-2.2</u>	0. -.09	.23	<u>53.</u>	-.12	<u>+.01</u> <u>-.001</u>	<u>-2.</u>
0110	3E3	1	E7	E7	6.7E-19	.56	<u>+.25</u> <u>-2.4</u>	<u>+.13</u> <u>-.1</u>	.54	<u>46.</u>	-.12	<u>+.088</u> <u>-.005</u>	<u>-2.1</u>
0120	10	E2	E4	E5	6.3E-19	.52	<u>+.36</u> <u>-3.4</u>	0. -.096	.23	36.	-.12	<u>+.041</u> <u>-.004</u>	<u>-2.9</u>
0123	E2	E2	E4	E6	1.8E-18	.52	<u>+.72</u> <u>-6.3</u>	0. -.1	.23	22.	-.12	<u>+.041</u> <u>-.005</u>	<u>-4.9</u>
0126	5E3	1	E8	E8	1.E-19	.52	<u>+.16</u> <u>-1.5</u>	0. -.093	.27	<u>73.</u>	-.12	<u>+.022</u> <u>-.002</u>	<u>-1.4</u>

Table 6.3.2. Some results of REGULA with SIMULA (free simulation)

Case (3) of FDBCK1 at 100% of full power

Excitation (6): $u_{d8} = -.1u_{s8}$ ($u_{d8} = P_E$, step disturbance,
 $u_{s8} = P_E$, steady state)Constraints on deviations: $-1. \leq u_{c1} \leq +1.$
 $-1.45 \leq u_{c2} \leq +1.8$
 $-1. \leq x_{30} \leq +1.$ c/sec
 $-40. \leq x_{10} \leq +40.$ psia
 $-0.0033 \leq \dot{u}_{c2} \leq +0.0033$ /sec
 $-1. \leq x_{21} \leq +1.$ x_1 (peak to peak) $\leq .20$

Notes: (i) Values underlined violate constraints.

(ii) $E2 = 10^2$

The results actually available to the designer are of course more complete than those listed in table (6.3.2). They include the plots of the responses of the most important variables. These plots are to be found in appendix(G) for one case (0126) of SIMULA. In the same appendix are the plots for the same case of SIMSAT, as well as SIMSAT with a reduced set of feedback variables. From table (6.3.2) it is seen that the time constant of the mechanical power delivered is usually about .5 seconds, i.e. well within the specifications set in the control objectives. On the other hand, it proved quite impossible to manage so that all the variables could stay within their constraints. As can be seen in appendix(F) (where all the cases run are listed), by penalizing one variable the usual result is that one or more of the other variables yield higher peak values.

A situation of "tug of war" arises which leads to ever increasing penalties all around, without achieving the desired aim of getting the peak values to remain within the constraints. In table (6.3.2) only two cases (84 and 92) are written for illustration of what happens when the peak of u_{c_1} (Y_M demanded) is forced to remain within its constraints: the pressure rises to peaks 3 or 4 times

as high as the values specified in the control objectives.

The peak of x_1 (normalized flux) has approximately the same value for all cases indicating that the adjustment of reactor power remains relatively detached from the conflicting relationship of the other important parameters. The speed of the demanded position of Y_{TCV} ($\dot{u}_{c_2}$) is almost invariably greater than the specified value. Although it has been stated before that the specified speed could be altered, at least in theory, it should be obvious that preference is given to the cases where the speed is within its constraint or not much outside it.

Penalties on u_{c_2} and on x_{30} have normally the same result of reducing the excursion of Y_{TCV} and of the frequency error.

This can be seen better from the complete tables in appendix(F). But the advantage of penalizing u_{c_2} lies in the fact that alone among the 4 costed parameters it has little influence in the peak values of the other 3. Hence it is more amenable to "tuning".

12 of the cases in table 6.3.2 have been selected from the complete tables (in appendix F) because they provide a reasonable basis for being run on SIMSAT, having one or both of the peak values of the control variables violating their constraints, mostly by something of the order 100-200% but not as much as a full order of magnitude.

6.3.3 REGULA with SIMSAT

It should be clear by now why the 12 cases mentioned in table (6.3.2) have been selected to be run on SIMSAT. By forcing the control inputs to remain within their constraints it was expected that the frequency error and pressure would go up, and it was hoped that the displacement would be acceptable. Let us look at

table (6.3.3) and discuss it.

The most important conclusion is that it proved impossible to force x_{10} (p_d) to remain within its constraints of ± 40 p.s.i.a. However, the figure is normally around the 80 p.s.i.a. which is only twice the constraint. Now this constraint was an engineering specification, as will be recalled from section (3.7). It is not a hard constraint, and the specification may be changed, as was found in consultation with the experts at the plant site. (*)

Next we come to $\dot{u}_{c_2}$, the speed of Y_{TCV} , which in all cases violates the constraints, but comes most closely to them in case (0085).

Cases (0096), (0100) and (0126) also yield $\dot{u}_{c_2}$ quite close to the constraints, though outside them. It was already discussed that the speed constraint could be changed, and the cases mentioned seem acceptable.

Next we come to the neutron flux, which for SIMULA was always $\approx .12$ but for SIMSAT increases and in one case even violates the constraint. From the above mentioned cases, (0085) and (0126) are the most preferable.

In only one case (0049) does the frequency violate its constraints of ± 1 c/s. It is of course desirable to choose among the cases with the smallest frequency deviations, and again cases (0085), (0100), (0096) and (0126) are the best.

It is at this stage that a controller has to be recommended, and cases (0085), (0126) seem the most appropriate choices. There is

(*) The purpose of the bar on too large deviations from the steady state pressure is, for negative deviations, the danger of pump cavitation should the pressure fall to a drastically low value, and for positive deviations, danger of the pump stalling.

surely room for more refined optimization, but it seems to be that the pressure deviation must be high in order to accommodate the remainder of the parameters within their constraints. Physical changes spring to mind as possible ways out of this problem, and most probably a different choice of k_v (voidage coefficient of reactivity) could lead to a solution (see table 1.1 in section 1.2) but any physical change was not contemplated in this study. (*)

In all the cases run on SIMSAT, the control vector entered the zone of no saturation at times shorter than 30 seconds. This allowed us to set relatively long integration intervals, namely 10 seconds long after 100 sec. of simulation. The intervals up to the first 10 sec. of simulation are built into the program - i.e. they are not data - and can be found in section (5.5.1). The final settling values of all the variables were equal for all cases of SIMSAT and SIMULA, as they should be. The settling times for the control inputs are of the order of 200 - 300 seconds, whereas the remainder of the parameters settle anywhere between 100 and 1000 seconds. This result is not surprising, as the eigenvalues of the controlled system yield natural modes of oscillation ranging from 40 milliseconds to 2500 seconds.

(*) Also, in changing k_v , careful consideration must be given to keep it within the constraints of a reasonable Uranium fuel economy.

Case	Costs				Results							
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ	Peak values						
	p_d	δf	y_M	y_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{40}	x_{11} Φ	$\dot{u}_{c_2}$	x_{21} y_M
0012	10	E4	E4	E2	.16	+ .65 -1.	+ .15 -1.45	.64	<u>78.</u>	-.17	+17. - <u>65</u>	-1.
0029	10	E5	E4	E2	.12	+ .67 -1.	+ .44 -1.45	.68	<u>83.</u>	-.18	+34. - <u>3.5</u>	-1.
0046	5E2	E6	E6	E3	.12	+ .49 -1.	+ .38 -1.45	.33	<u>81.</u>	-.16	+42. - <u>4.</u>	-1.
0049	5E3	E6	E6	E3	.56	+1. -1.	+1.1 -1.45	<u>2.5</u>	<u>63.</u>	-.18	+68. - <u>8.6</u>	-1.
0085	10	1	E5	E6	.51	+ .24 -1.	0. -.09	.23	<u>83.</u>	-.14	+ .005 - .0007	-1.
0096	30	1	E5	E6	.51	+ .38 -1.	0. -.09	.25	<u>82.</u>	-.15	+ .014 - .002	-1.
0098	20	1	E5	E5	.54	+ .32 -1.	+ .17 -.09	.67	<u>78.</u>	-.14	+ .068 - .005	-1.
0100	20	1	E5	E6	.51	+ .33 -1.	0. -.09	.23	<u>82.</u>	-.15	+ .009 - .0013	-1.
0110	3E3	1	E7	E7	.57	+ .37 -1.	+ .28 -.09	.9	<u>76.</u>	-.15	+ .09 - .008	-1.
0120	10	E2	E4	E5	.52	+ .66 -1.	+ .037 -.096	.42	<u>80.</u>	-.17	+ .042 - .0034	-1.
0123	E2	E2	E4	E6	.51	+1. -1.	+ .033 -.12	.41	<u>80.</u>	-.21	+ .042 - .0043	-1.
0126	5E3	1	E8	E8	.52	+ .18 -1.	0. -.094	.31	<u>85.</u>	-.13	+ .023 - .002	-1.

Table 6.3.3. Some results of SIMSAT (saturated simulation)
for cases of REGULA

Same comments as for table (6.3.2)

6.3.4 Reduced set of feedback variables. Recommended filter.

As explained in section (3.8), a reduced set of feedback variables was found and tried successfully in a number of cases. To determine the "dominant" values, the "weighted gain" matrix of elements $k_{r,i,j} x_{jpeak}$ was computed for a number of cases, and the incidence of these dominant values inspected.^(*) Tables (6.3.4) and (6.3.5) are summaries for excitations type (6) and (5) respectively. The tables present the dominant values as feedback to each control input separately. From the 15 parameters which contribute more heavily to the feedback, the following are directly or almost directly measurable: x_4 , x_{10} , x_{18} , x_{21} , x_{22} , x_{23} , x_{30} . On the other hand x_2 , x_3 , x_8 are not measurable. This leaves x_{11} , x_{13} , x_{25} , x_{26} , x_{27} to be discussed here.

$x_1 = M$ (total mass of water and two-phase mixture in primary circuit) follows closely the transients of $x_{20} = h_{fh}^*$ (enthalpy of feedwater leaving the feedheaters), which is measurable and hence could be taken - with an appropriate scaling factor - in lieu of x_{11} for feedback purposes.

$x_{13} = h_d^*$ (enthalpy of steam drum water) also at present not measurable, could be measured by the temperature of the steam drum water. x_{25} , x_{26} , x_{27} are the delayed variables introduced in section (2.1.2) and described by equations (3.2.14), (3.2.15) in section (3.2.2). These equations could be simulated in a fairly simple analog device and the delay-variables x_{25} , x_{26} , x_{27} (which correspond to z in the equations aforementioned) could be

(*) The "weighted gain" matrix option is now standard on the program, but since it was introduced at a late stage, only a limited number of cases are available for inspection.

obtained from the measurement of the delayed parameters x_d , i.e. x_{13} , x_{20} , x_{A_h} delayed, as seen in the part of section (3.2.2) which follows the equations mentioned. We have seen that x_{13} and x_{20} are measurable; so is $x_{A_h} = \delta h$ (enthalpy rise through core), albeit indirectly.

Of the three unmeasurable variables x_2 , x_3 , x_8 the following can be said: for excitation type (6) - the most important - x_2 and x_8 are non-dominant, whereas x_3 is a dominant value in a small number of cases, and then at the lower end of the upper decade. For excitation type (5) - 20 p.s.i.a. of initial pressure deviation - there were some cases where x_2 and x_8 are dominant, and many cases where x_3 is dominant. Again, the three of them are on the lower side of the upper decade. It was decided to experiment with a filter which does not include x_2 , x_3 , x_8 and see if there are important differences in the transients. Various cases of SIMULA and SIMSAT were tried and the results were highly satisfactory: the difference in the transients between the cases run with the complete set of feedback variables and those run with the above-mentioned filter, is hardly noticeable at all.

We shall give peak and final values of two cases (0085 and 0126); the simulation plots for case (0126) can be seen in appendix (G).

The filter used and recommended is as follows:

$$\text{Feedback to } u_{c_1} : x_4 - x_{10} - x_{11} - x_{18} - x_{21} - x_{22} - x_{25} - x_{26} - x_{27} - x_{30} .$$

$$\text{Feedback to } u_{c_2} : x_{10} - x_{11} - x_{13} - x_{18} - x_{22} - x_{23} - x_{25} - x_{26} - x_{27} - x_{30} .$$

It has been explained in section (3.8) that although the transients with the filtered controller were almost undistinguishable from the transients with the complete set, the final settling-down values

were different, unless a D.C. - compensation scheme explained in the same section was used. This scheme consists of forcing two variables to settle at zero value (which is the proper one), changing the forced added control $\underline{k}_{du}$ of eqs. (3.5.4), (3.5.9) into another vector of two components, computed with eq. (3.8.5) and repeated here

$$\tilde{\underline{k}}_{du} = -[CA_{oc}^{-1} B_c]^{-1} CA_{oc}^{-1} B_d u_d \quad (6.3.1.)$$

or

$$\tilde{\underline{k}}_{du} = -K_{DC} u_d \quad (6.3.2.)$$

In those cases where DC compensation was used, the difference between $\underline{k}_{du}$ and $\tilde{\underline{k}}_{du}$ was only fractional; however, it is this small difference which is responsible for bringing the parameters to their proper settling-down value. Put in a different way, this is also saying that without D.C. compensation, the very small amounts of feedback which can easily be neglected as far as the transients are concerned, when cancelled do not allow the parameters to settle down at their proper values.

Let us tabulate the values for $\underline{k}_{du}$ and $\tilde{\underline{k}}_{du}$ for three cases:

Case		$\underline{k}_{du}$	$\tilde{\underline{k}}_{du}$
0085	k_{du_1}	-1.839	-1.651
	k_{du_2}	- .09084	- .09136
0108	k_{du_1}	-1.723	-1.553
	k_{du_2}	- .04271	- .09051
0126	k_{du_1}	-1.525	-1.386
	k_{du_2}	- .09140	- .09410

Let us now tabulate comparative results for two cases, 0085 and 0126, with peak and final values for the four possible ways of simulation: free, saturated (both with complete set of feedback variables), saturated with reduced set of feedback variables (or filter) but no D.C. compensation and finally saturated, filtered and D.C. compensated. These tables are (6.3.6) and (6.3.7). From them and measuring the peak value, it can be concluded that the transients are indeed very similar in the filtered and the unfiltered cases.

A more detailed inspection may be done on the simulation plots for case (0126) in appendix(G).

The final values of a number of important variables for the different types of simulation of cases (0085), (0126) can be compared in table (6.3.7) where it is seen that for the case of filter with D.C. compensation the values are quite acceptable.

Same comments as for table (6.3.2)

Table 6.3.6. Comparative results (peak values) for various types of simulation.

Case	Type of simulation	Costs				Results								
		x_{10} p_d	x_{30} δf	u_{c1} Y'_M	u_{c2} Y'_{TCV}	τ (sec) x_{18} P_{MEC}	Peak values							
							u_{c1}	u_{c2}	x_{30}	x_{10}	x_1 Φ'	$\dot{u}_{c2}$	x_{21} Y_M	x_{A5} Q'
0085	SIMULA	10	1	E5	E6	.51	+0.19 -1.8	0. -0.09	.23	62.	-0.12	+0.005 -0.0007	-1.6	-0.13
	SIMSAT					.51	+0.24 -1.	0. -0.091	.23	83.	-0.14	+0.005 -0.0007	-1.	-0.15
	SIMSAT & FILTER					.51	+0.23 -1.	0. -0.094	.23	84.	-0.14	+0.005 -0.0007	-1.	-0.16
	SIMSAT & FILTER & D.C. COMPENS.					.51	+0.19 -1.	0. -0.091	.23	85.	-0.128	+0.005 -0.0007	-1.	-0.143
0126	SIMULA	5E3	1	E8	E8	.52	+0.16 -1.5	0. -0.09	.27	73.	-0.12	+0.022 -0.002	-1.4	-0.13
	SIMSAT					.52	+0.18 -1.	0. -0.094	.31	85.	-0.13	+0.023 -0.002	-1.	-0.14
	SIMSAT & FILTER					.52	+0.16 -1.	0. -0.11	.32	85.	-0.13	+0.023 -0.002	-1.	-0.15
	SIMSAT & FILTER & D.C. COMPENS.					.51	+0.14 -1.	0. -0.098	.32	89.	-0.123	+0.023 -0.003	-1.	-0.128

Case	Type of simulation	Results: final (settling-down) values									
		u_{c_1} Y'_M	u_{c_2} Y'_{TCV}	x_{10} p_d	x_{13} h_d^*	x_{21} Y_M	x_{23} Y_{TCV}	x_{30} δf	x_{E_2} T_f^*	x_1 Φ'	x_{A5} Q'
0085	SIMULA	-.38E-5	-.09047	.34E-6	.93	-.38E-5	-.09047	.43E-6	-50.8	-.0729	-.0893
	SIMSAT	-.25E-5	-.09047	.25E-6	.93	-.25E-5	-.09047	.43E-6	-50.8	-.0729	-.0893
	SIMSAT, FILTER	-.24E-5	-.09336	-20.3	-2.03	-.24E-5	-.06993	-.047	-53.1	-.0717	-.0885
	SIMSAT, FILTER, D.C. COMPENS.	-.24E-5	-.09047	-.35E-2	.93	-.24E-5	-.09046	-.78E-5	-50.8	-.0729	-.0893
0126	SIMULA	-.38E-5	-.09047	.34E-6	.93	-.38E-5	-.09047	.43E-6	-50.8	-.0729	-.0893
	SIMSAT	-.25E-5	-.09047	.46E-6	.93	-.25E-5	-.09047	.43E-6	-50.8	-.0729	-.0893
	SIMSAT, FILTER	-.24E-5	-.105	-20.1	-2.08	-.24E-5	-.06957	-.071	-53.1	-.0717	-.0885
	SIMSAT, FILTER, D.C. COMPENS.	-.24E-5	-.09047	-.42E-2	.93	-.24E-5	-.09047	-.14E-4	-50.8	-.0729	-.0893

Table 6.3.7. Comparative results (final values) for various types of simulation.

Same comments as for table (6.3.2)

6.3.5 The recommended case. Discussion of results. Comments on the analog runs.

From the inspections of tables (6.3.2), (6.3.6), (6.3.7), which in turn are a selection of much more complete tables, it becomes apparent that the best choice is probably case (0085) or (0126). We shall write here, for the benefit of possible implementation at the plant site - partly or totally - the matrices necessary for the constructions of the controller for both cases. The scheme to be implemented is that of figure (3.1) in section (3.5). The matrices to be written here are:

- (i) $K_r \equiv$ Gain matrix for the complete set of feedback variables from $\underline{x}$ to $\underline{u}_c$.
- (ii) $K_{dg} \equiv$ Gain matrix for the complete set, from the disturbance vector $\underline{u}_d$ to $\underline{k}_{du}$ which feeds $\underline{u}_c$.

For the reduced set of feedback variables - and hence liable to actual implementation - the matrix K_r must be filtered with the filter recommended in the previous section and K_{dc} used instead of K_{dg} . Hence

- (iii) $K_{dc} \equiv$ Gain matrix for the filtered case - using D.C. compensation to force $x_{40}(p_d)$ and $x_{30}(\delta f)$ to 0 - from $\underline{u}_d$ to $\tilde{\underline{k}}_{du}$ which (in place of $\underline{k}_{du}$) feeds $\underline{u}_c$.

Only 5 decimal places will be written, as greater accuracy will not be possible to be implemented anyway ($E+2 = 10^2$).

- (i) Matrix K_r (the two rows are written columnwise for convenience)

Case (0085)

Feedback to:	$\underline{u}_{c1} = \text{YMDDEM}$	$\underline{u}_{c2} = \text{YTCVDEM}$
From $x_1 = \text{PHIDASH}$	.16755E-1	-.67275E-5
$x_2 = \text{X1BARDASH}$	.15069E+1	-.13517E-2
$x_3 = \text{X2BARDASH}$	.95610E+0	-.16211E-2
$x_4 = \text{TMAV}$	.22770E+0	-.13481E-3

$x_5 = A1$	.62053E-2	-.26428E-4
$x_6 = A2$	.26588E-1	-.52033E-4
$x_7 = A3$	.89290E-1	-.12417E-3
$x_8 = TF1STAR$	-.53360E-3	-.56275E-6
$x_9 = TF2STAR$	.87331E-3	-.18234E-6
$x_{10} = PD$	.35006E-2	-.11694E-3
$x_{11} = M$	.16716E-3	-.13370E-5
$x_{12} = MW$	-.52890E-5	-.14034E-7
$x_{13} = HDSTAR$	.53673E-2	.17924E-3
$x_{14} = QW$	.29300E-3	.89524E-5
$x_{15} = OMEGA$	.10949E-3	.33139E-5
$x_{16} = WFW$	.17903E-7	-.50005E-8
$x_{17} = DPREG$	-.28509E-7	.11440E-7
$x_{18} = PMECH$	.21568E-2	.78609E-4
$x_{19} = YR$	.11579E-2	-.16740E-3
$x_{20} = HFHSTAR$	.70906E-3	.22236E-5
$x_{21} = YM$	.52264E-1	-.29786E-4
$x_{22} = HDASH$	.65784E+2	-.37990E-1
$x_{23} = YTCV$	-.78187E-4	.85998E-2
$x_{24} = DELAYVAR1$	.12192E-3	-.78934E-6
$x_{25} = DELAYVAR2$	-.15702E-1	-.66777E-4
$x_{26} = DELAYVAR3$	.16880E-1	-.51216E-4
$x_{27} = DELAYVAR4$	.17194E-1	-.12816E-3
$x_{28} = IF1$	-.40127E-2	-.11174E-4
$x_{29} = IF2$	.14446E+0	.44742E-3
$x_{30} = DF$	.83842E-1	-.70244E-3

(ii) Matrix K_{dg} (rows written columnwise)

Case (0085)

Feed to:	u_{c_1}	u_{c_2}
From $u_{d_1} = \text{DKBAL}$	.16055E+4	-.89309E+0
$u_{d_2} = \text{TMIN}$	.13119E-1	.85003E-6
$u_{d_3} = \text{YDV}$	-.19313E+1	.10570E-2
$u_{d_4} = \text{YRDEM(REF)}$	.28892E-1	.89484E-4
$u_{d_5} = \text{WSDDEM}$	.0	.0
$u_{d_6} = \text{LIDEM}$	.41711E-2	.11068E-4
$u_{d_7} = \text{DF(REF)}$	.83839E-1	-.70440E-3
$u_{d_8} = \text{PE}$	-.17563E+0	-.86731E-2

(iii) Matrix K_{dc} (rows written columnwise)

Case (0085)

Feed to:	u_{c_1}	u_{c_2}
From u_{d_1}	.16733E+4	-.96632E+0
u_{d_2}	.23293E-1	-.14495E-3
u_{d_3}	-.20389E+1	.93893E-3
u_{d_4}	.0	.0
u_{d_5}	.0	.0
u_{d_6}	.0	.0
u_{d_7}	.0	-.50000E+0
u_{d_8}	-.15760E+0	-.87228E-2

For excitation type (6) the last column only is necessary for either K_{dg} or K_{DC} , since u_{dg} is the only external disturbance.

As mentioned in section (6.3.3), Case (0085) is not entirely within its constraints, notably the pressure x_{10} which has a peak value of 85 p.s.i.a. for excitation type (6), i.e. about twice the constraint value of ± 40 p.s.i.a. This is still acceptable from the engineering point of view.

The other constraints which are violated for excitation type (6) are (see table 6.3.1 for comparison): speed of demanded opening of the turbine control valve ($\dot{u}_{c_2}$) is 0.0051 instead of 0.0033; rate of decrease of pressure ($\dot{x}_{10}$) is -1.3 psia/sec instead of -1.0 psia/sec; speed of opening (closing) of the feedwater valve position ($\dot{x}_{19}$) is 0.057 (-.14) instead of ± 0.033 ; speed of closing of moderator valve position ($\dot{x}_{21}$) is 1.0 instead of 0.2.

For excitation type (5) there are only three violations:

$\dot{u}_{c_2}$ is -0.0057 instead of -0.0033

$\dot{x}_{10}$ is -5.5 instead of -1.0

$\dot{x}_{19}$ is 0.23 instead of 0.033

All the above mentioned constraint violations (except pressure) are of speed, and hence admissible in principle, as explained in section (3.7). However, in making the choice between cases (0085) and (0126), preference was given to the lower speed of Y'_{TCV} in case (0085) as against the slightly lower excursion of $\bar{\Phi}'$ in case (0126).

The final value of $\bar{\Phi}'$ is -.0729 (see table 6.3.7); the peak values of -.128 and -.123 for cases (0085) and (0126) mean that the overshoots are 76% and 69% respectively. The thermal power output $Q' (x_{A_5})$ has a final value of -.0893 (table 6.3.7), whereas the

peak values are $-.143$ and $-.128$ for cases (0085) and (0126) respectively. Therefore the overshoots of Q' are 60% and 55% for the above two cases. Hence, if the speed $\dot{u}_{c_2}$ can be allowed to be higher, case (0126) may be a better choice.

The three matrices K_r , K_{dg} , K_{dc} will be listed as before:

Matrix K_r (Case 0126)

Feedback to:	u_{c_1}	u_{c_2}
From x_1	.12021E-1	-.49547E-4
x_2	.11853E+1	-.70209E-2
x_3	.69546E+0	-.83795E-2
x_4	.19101E+0	-.66792E-3
x_5	.33611E-2	-.13618E-3
x_6	.17698E-1	-.27497E-3
x_7	.65216E-1	-.64816E-3
x_8	-.42946E-3	.28537E-5
x_9	.59034E-3	-.30868E-5
x_{10}	.24788E-2	-.58011E-3
x_{11}	.11665E-3	-.68224E-5
x_{12}	-.53851E-5	-.71194E-7
x_{13}	.29690E-2	.88361E-3
x_{14}	.21003E-3	.43529E-4
x_{15}	.78496E-4	.16034E-4
x_{16}	.13689E-7	-.25019E-7
x_{17}	-.21806E-7	.56982E-7
x_{18}	.15249E-2	.34423E-3
x_{19}	.89214E-3	-.82686E-3
x_{20}	.48550E-3	.10048E-4
x_{21}	.44242E-1	-.14684E-3

x_{22}	.55469E+2	-.18890E+0
x_{23}	-.38546E-4	.39422E-1
x_{24}	.72744E-4	-.39333E-5
x_{25}	-.11366E-1	-.32823E-3
x_{26}	.12037E-1	-.26806E-3
x_{27}	.12204E-1	-.65545E-3
x_{28}	-.40378E-2	-.56490E-4
x_{29}	.14088E+0	.22217E-2
x_{30}	.59259E-1	-.35241E-2

Matrix K_{dg} (Case 0126)

Feed to:	$\bar{u}_{c_1}$	u_{c_2}
From u_{d_1}	.13607E+4	-.43974E+1
u_{d_2}	.13097E-1	.83184E-5
u_{d_3}	-.16471E+1	.51087E-2
u_{d_4}	.28175E-1	.44433E-3
u_{d_5}	.0	.0
u_{d_6}	.42469E-2	.56147E-4
u_{d_7}	.59259E-1	-.35241E-2
u_{d_8}	-.14563E+0	-.87273E-2

Matrix K_{dc} (Case 0126)

Feed to:	u_{c_1}	u_{c_2}
From u_{d_1}	.14109E+4	-.48049E+1
u_{d_2}	.20629E-1	-.72074E-3
u_{d_3}	-.17230E+1	.46001E-2
u_{d_4}	.0	.0
u_{d_5}	.0	.0

u_{d_6}	.0	.0
u_{d_7}	.0	-.50000E+0
u_{d_8}	-.13234E+0	-.89851E-2

For case (0126) the constraint violations are:

Excitation type (6):

$x_{10}(p_d)$ is 89 psia instead of 40 psia
 $\dot{x}_{10}$ is -1.2 psia/sec instead of -1. psia/sec
 $\dot{u}_{c_2}(\dot{Y}_{TCV})$ is .023 instead of .0033
 $\dot{x}_{19}(\dot{Y}_R)$ is .052 and -.14 instead of $\pm .033$
 $\dot{x}_{21}(\dot{Y}_M)$ is -1. instead of -.2

Excitation type (5):

$\dot{u}_{c_2}$ is -.029 instead of -.0033
 $\dot{x}_{10}$ is -5.5 instead of -1.
 $\dot{x}_{19}$ is .23 instead of .033

To summarize, cases (0085) and (0126) are both good choices, although in none we manage to have all the peak values within the specified constraints. The violations, where they occur, are on "soft" constraints which may be changed by different engineering specifications. The hard constraints, on the other hand, are well under control. The reduced set of feedback variables, coupled with the D.C. compensation scheme discussed, offers a performance remarkably close to that of the complete set, with all selected variables measurable, either directly or indirectly.

It is at this stage that some comments are due on the analog runs on the Winfrith simulator. The details of the runs are not discussed because the digital simulation is more accurate and thus

yields more definite results. But the series of analog runs were very helpful in giving a quick and clear picture of the controlled plant dynamics, in free or saturated simulation, with complete and reduced set of feedback variables. This also enabled us to work with confidence towards the completion of the digital programs. It should also be added that although the accuracy of an analog computer is inherently lower than that of a digital computer, the Winfrith simulator contains the original non-linear model and hence is more faithful to the actual plant-performance than the linear model used for the digital simulation programs.

Finally, the analog simulator was useful in order to test if the linear model is correct in the first place, as will be detailed in the next section. (*)

6.4 Model checking

It has been explained in section (5.2.3) that routine FDBCK1 was constructed in such a way as to accommodate various different linear models ("cases") chosen by the user. The model of case (3) was used for the optimal regulator problem and described in section (6.2). The model of case (7) was constructed in order to match the non-linear model set up on the Winfrith simulator with the specific purpose of comparing the transients and so to test the correctness of the linear model.

The model of case (7) is almost identical to that of case (3), except that it does not include load following (see footnote in section 6.2).

(*) In this section the matrices given for implementation in the construction of a controller represent, for the complete set of feedback variables, 74 lines of communication (matrices K_r and K_{dg}). For the reduced set with DC compensation 29 lines

The model of case (7) was simulated with excitation type (3), which is a step response for a disturbance of 10 miliniles in u_{d_1} = DKBAL. The plot is included in appendix (G). The transients were compared with the analog plots from the Winfrith simulator and found to be closely similar, generally within 10 to 30%. The final (settling-down) value were also similar, as seen below:

Final (settling-down) values:

	Digital	Analog ~
x_1 = PHIDASH	.12E-1	.11E-1
x_4 = TMAV	.25E+0	.20E+0
x_{10} = PD	.12E+2	.85E+1
x_{16} = WFW	.43E+1	.30E+1
x_{19} = YR	.13E-1	.95E-2
x_{21} = YM	-.1E-25	.0
x_{22} = HDASH	-.2E-14	.0
x_{23} = YTCV	-.5E-14	.0
x_{A_2} = FBAR	-.61E-4	Noise
x_{A_3} = TCOOLSTAR	.18E+1	.13E+1
x_{E_2} = TFAVSTAR	.10E+2	.90E+1
x_{E_7} = WSDRUM	.43E+1	.30E+1

Only a limited number of parameters were recorded on the simulator as can be seen from the list above and in the plots in appendix (G). In assessing the similarity of the analog and digital runs, it is necessary to bear in mind the non-linearity of the analog model.

/cont.

(matrices K_r filtered and K_{dc}). In either case it is necessary to consider the cost effectiveness of investing in additional instrumentation as against the advantages in the performance with this controller compared with the conventional set-up in existence now.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

Having completed the chapter on results, it is appropriate to ask what this work has achieved, in the light of what its author has set out to do.

In the first place it has been shown that a controller for an existing large plant can be designed using tools of optimal control theory. Although this in itself is by no means unique, it is well known that a tremendous gap exists between the many sophisticated methods developed in several branches of modern control theory and the relatively few published practical applications of multivariable processes. Hence one more application, even if modest, can claim credit for its contribution towards the bridging of that gap.

In the second place, and at least during the time when the work for this thesis was in progress, no other work tackling the problem of control of a complete nuclear power plant and using methods of modern (optimal) control theory was known to this author. (*) Hence this contribution.

In the third place, a new method was developed for dealing with external measurable disturbances, which is computationally simple for both the regulator design and the analytic simulation, and which computes the correction to the disturbance simultaneously with the dynamic regulator.

Let us summarize the succession of steps taken for the controller design:

(i) Construction of the (non-linear) mathematical model of the plant^[6,8]. This is a most important part of the problem.

(*) See footnote in section (1.1).

If appropriately selected, a medium-size model may be constructed which contains all the important information, without the need for mathematical model reduction.

(ii) Solution, for the steady state, of the set of non-linear differential and algebraic equations of the model. This can be done for as many levels of operations as desired, and was done for 100% and 75% in this work.

(iii) Construction of the linear model for operation around the level of 100% computed in (ii) above. The transport-delay times which are very common in large processes, have been approximated by first order differential equations of the Padé type.

(iv) The mathematical model of (i) is simulated on an analog computer at the plant site and reflects the real operation quite faithfully. The transport delays were first simulated by computer delay drums and later with first order Padé approximations: in both cases the responses were similar to the real ones at the plant.

(v) The linear model of (iii) was compared with the non-linear model and the responses were fairly similar.

(vi) A regulator was designed with the data provided by the program for the linear model, plus elements of the cost matrices selected as data for each successive run of the program for the regulator design (REGULA).

(vii) The plant dynamics with the successive regulators designed as in (vi) were simulated (SIMULA) and the responses evaluated according to the control objectives and the constraints upon the system.

(viii) A selection of cases simulated as in (vii) were run on the saturated simulation program SIMSAT and the results

evaluated as in (vii), but more realistically since the hard constraints were not violated due to the nature of the program.

(ix) A reduced set of feedback variables was obtained, selected by their relative weight on the amount of control feedback. Nearly all of these variables are measurable. A few among the selected cases from (viii) were simulated on SIMSAT with that reduced set, choosing the measurable variables only. The transients thus obtained are hardly distinguishable from those obtained using the complete set. The settling-down values are different, therefore a D.C. compensation scheme was introduced such as to force at least two variables to settle at their proper values, having only two inputs. The results of the D.C. compensation were very good, as not only the two selected variables settled at their correct values, but the remainder also settled very near to their own correct values.

(x) Two cases were chosen as recommendable for possible implementation, they are cases (0085) and (0126). The matrices necessary for the physical realization of these two regulators were given. Although the search for a better controller has not been exhausted, it is worthwhile to compare the results obtained with the control objectives, before further discussion:

(x.a) Frequency variations: for a step disturbance of -10% of the full power load, the peak frequency changes are 0.23 c/s and 0.32 c/s for the two cases, respectively. Since the maximum admissible change is ± 1 c/s and the desirable less than ± 0.25 c/s, the cases described offer a reasonably good result.

- (x.b) The time involved in power changes is shorter than the specified requirement. The time constant of the mechanical power delivered is of the order of 0.5 seconds.
- (x.c) On the dry-out margin problem it is best to examine the behaviour of Q' (the thermal power output) in the two cases mentioned. For a change in load of 10%, the overshoots are 60% and 55% respectively. It is unsafe to extrapolate for changes in load of 25% because the plant is non-linear whereas our model is linear. For straight extrapolation the result would not be entirely satisfactory, since the overshoot exceeds 40%, the level at which a switch in load from 75% to 100% would reach the 110% of full power which is the safety limit explained in section (1.3). This comment is of course true for the paragraphs (x.a) and (x.b) above as well. The simulation runs in this work were done for 10% step changes, considering that the model is linear and therefore a larger step could not any more be considered a "small signal".
- (x.d) For plant disturbance the results are likely to be better than for internal disturbances, if excitation type (5) is anything to judge by (recall that this is an initial departure of 20 p.s.i.a. from the nominal pressure).

Steps (i) to (x) indicate the method of controller design, and can be applied to any large-scale system. A proper check on the

complete non-linear model is strongly recommended. This will probably be done if a "modern" controller is to be implemented at the nuclear power plant which was the object of this study, as a sequel to it.

The feature of the reduced set of feedback variables should not be overlooked. It is an important stage in the development of the controller, since all the states fed back are measurable, directly or indirectly. It may be agreed that this was only a fortunate coincidence. However, there are reasons for suspecting that the most important variables - as far as weight of feedback is concerned - are measurable due to physical reasons. Although it is impossible for us to prove this, the fact still remains that with this approach an appreciably large number of feedback variables will be measurable, leaving only a few to be estimated if they cannot be neglected; this would make the computation necessary for estimation considerably easier. It is interesting to note how little difference for the transients is caused by neglecting all but the most "weight-carrying" feedback state variables.

It is hoped that the report of this study has conveyed the difficulties associated with this kind of work on a large system, in at least the following aspects:

- (a) The large range of eigenvalues of the system (order of 10^5) which was the reason for the lengthy trials of different methods of computation for eigenvalues and eigenvectors.
- (b) The trial-and-error procedure of selecting suitable cost matrices.
- (c) The painstaking nature of attention to detail which is necessary due to the sheer size of the system.

Following are some recommendations for future work:

- (i) It is felt that if the means were available, it might be advisable to set up an on-line scheme with a (large) computer, with programs and main data stored on disk. The delay between the times of changing parameters and obtaining the complete simulation is of order less than two minutes on a machine with the power of a CDC 6600.
- (ii) A more advanced and systematic method of choosing the cost matrices.
- (iii) A reliable program for non-linear simulation, either on a digital or on a hybrid digital-analog computer.
- (iv) Study the effect of noise on the plant, i.e. apply stochastic filtering and control to the study of this or another similarly large plant.
- (v) Investigate the stability conditions of the plant with the optimal controller incorporated, for the case when one or more lines of information fail.
- (vi) Develop a piece-wise linear model switching from one level of operation to another as the transients grow.
- (vii) Make a model reduction using one of the several methods available, apply optimal control, replace in the original model and compare results with the regulator obtained using a non-reduced model.

References:

- [1] E. J. DAVISON, A method of simplifying linear dynamic systems, IEEE Trans. Automatic Control, AC-11, (1966), p.93.
- [2] M. R. CHIDAMBARA, Two simple techniques for the simplification of large dynamic systems, Proc. J.A.C.C., (1969), p.669.
- [3] E. DUNCOMBE & D. E. RATHBONE, Optimization of the response of a nuclear reactor plant to changes in demand, IEEE Trans. Aut. C., AC-14, (1969), p.277.
- [4] R. R. MOHLER & C. N. SHEN, Optimal control of nuclear reactors, Academic Press, 1970.
- [5] J. ATARI & M. M. SHAH, Modelling and analytical control system design of a complete nuclear power plant prototype, IFAC 5th World Congress (1972), paper 6.1.
- [6] C. B. GUPPY, A systems analysis of the SGHWR dynamics and its application to the operations simulator. Report AEEW-R 501 (1967).
- [7] P. K. M'PHERSON, G. B. COLLINS, C. B. GUPPY, A. SUMNER, Dynamics analysis of a nuclear boiler. Proc. Inst. Mec. Eng., 180, (1965-66), Part I, No. 17, p. 417.
- [8] SGHWR Simulator, lists of equations, parameters and constants, Internal document, Computing Section (231R), AEE Winfrith, 20.7.67.
- [9] R. V. MOORE and J. E. R. HOLMES, The SGHWR system, Proc. of Conference on Steam Generating and other heavy water reactors, British Nuclear Energy Society, May 1968, paper A:1.
- [10] D. WRAY, M. H. BUTTERFIELD, R. N. M. McMILLAN, Control of SGHWR's, id., paper B:6.
- [11] See, for example, A. E. Bryson and Y. C. Ho, Applied Optimal Control, Blaisdell Publ. Company (1969).
- [12] M. J. D. POWELL, A method for minimizing a sum of squares of non-linear functions without calculating derivatives, Computer Journal, 7 (1965), p.303.
- [13] P. J. REEVE, A method of approximating to pure time delay, International Journal of Control, 8 (1968), p.53.
- [14] J. L. STEWART, Generalized Padé approximations, Proc. I.R.E., 48 (1960) p.2003.
- [15] See, for example, K. Ogata, State Space analysis of control systems, Prentice-Hall (1967) p.287.
- [16] J. H. WILKINSON, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford (1965).

- [17] K. MARTENSSON, Linear quadratic control package, part I, Report 6802, Lund Institute of Technology (1968).
- [18] W. N. CALDWELL, EIGSYS, FORTRAN IV Subroutine, internal document, library of Control Systems Section, Dept. of Computing and Control, Imperial College, 1967.
- [19] J. G. F. FRANCIS, The QR Transformation, Part I: The Computer Journal, 4 (1961), p.265, Part II: *ibid*, 4 (1962), p.332.
- [20] J. H. WILKINSON, Stability of the reduction of a matrix to almost triangular and triangular forms by elementary similarity transformations, J. ACM, 6 (1959), p.336.
- [21] E. R. HANSEN, On the Danilewski Method, J. ACM, 10 (1963), p.102.
- [22] D. Q. MAYNE, Lecture Notes, Control Systems Section, Department of Computing and Control, Imperial College.
- [23] J. E. VAN NESS, PALS, A program for analysing linear systems, a report from Northwestern University, Evanston, Illinois, 1969.
- [24] See, for example, D. G. Schultz and J. L. Melsa, State functions and linear control systems, McGraw Hill (1967).
- [25] P. HAGANDER, Numerical Solution of $A^T S + SA + Q = 0$, Report 6920, Lund Institute of Technology, (1969).
- [26] R. E. KALMAN, Contributions to the theory of Optimal Control, Boletín de la Sociedad Matemática Mexicana (1960) p.102.
- [27] D. L. KLEINMAN, On an interative technique for Riccati Equation Computations, IEEE Trans. Automatic Control, AC-13, (1968), P.114.
- [28] R. BELLMAN, Introduction to Matrix Analysis, McGraw Hill, (1960), p. 231.
- [29] E. J. DAVISON and F. T. MAN, The Numerical Solution of $A^T Q + QA = -C$, IEEE Trans. Automatic Control, AC-13, (1968), p.448.
- [30] A. G. J. MacFARLANE, An eigenvector solution to the optimal linear regulator problem, J. Electronics and Control, 14 (1963), p.643.
- [31] J. E. POTTER, Matrix Quadratic Solutions, J. SIAM, Appl. Math., 14 (1966), p.496.
- [32] D. R. VAUGHAN, A negative exponential solution for the Matrix Riccati Equation, IEEE Trans. Autom. Control, AC-14, (1969), p.72.

- [33] A. F. FATH, Computational Aspects of the linear optimal Regulator Problem, IEEE Trans. Autom. Control, AC-14, (1969), p.546.
- [34] J. J. O'DONNELL, Asymptotic solution of the Matrix Riccati Equation of Optimal Control, Proc. 4th annual Allerton Conf. Circuit and System Theory (1966), p.577.
- [35] See, for example, M. Athans and P. F. Falb, Optimal Control, McGraw-Hill (1966), chapter 9.
- [36] C.E.R.N. library routines No. F303 (INCOMA) and F100 (MATIN1), (1969).
- [37] C. D. JOHNSON, Optimal control of the linear regulator with constant disturbances, IEEE Trans. Autom. Control, AC-13, (1968), p.416.
- [38] D. G. LUENBERGER, Observers for multivariable systems, IEEE Trans. Autom. Control, AC-11, (1966), p.190.
- [39] R. E. KALMAN and R. BUCY, New results in linear filtering and prediction, Trans. ASME, 83D, (1961), p.95.
- [40] O. H. D. WALTER, Eigenvector scaling in a solution of the matrix Riccati equation, IEEE Trans. Autom. Control, AC-15, (1970), p.486.
- [41] D. R. VAUGHAN, Further comments on "Eigenvector Scaling in a solution of the matrix Riccati equation", IEEE Trans. Autom. Control, AC-16 (1971), p.211.

APPENDIX A.

Listing of the more important routines in SSNL2D.

SSNL2D

CALFUNO

SCALEV

SCALEF

SCLEVA

CALCLIT

FUN1

FUN2

FUN3

FUN4

FUN5

FUN6

INTPOL

VAO2A

VDOLA

PROGRAM SSNL2D (INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT)

3.11.70 (L. TEPPER, CONTROL SYSTEMS SECTION, DEPARTMENT OF
COMPUTING AND CONTROL, IMPERIAL COLLEGE)

STEADY STATE SOLUTION OF N/L EQUATIONS OF SGHWR MODEL

TVA02A (CH.E.) USED

THE SET OF N/L EQUATIONS INCLUDE =

A) $F(X, XA, X0, XA0, U) = 0$, IDENTICAL TO $XDOT = F()$ IN DYNAMIC SET
B) $G(X, XA, X0, XA0, U) = 0$, ... $G() = 0$

WHERE

VECTOR	F (DIFF.EQUATIONS),	23	COMPONENTS
..	X (STATE VARIABLES)	23	..
..	G (IMPL.ALG.EQUNS.)	17	..
..	XA(.. .. VARS.)	17	..
..	U (INPUT VARIABLES)	7	..

AND

X13D(ITA0W), X13D(ITA0RW), X200(ITA0FW), X040(ITA0P) ARE DELAYED VALUES
OF X(13), X(13), X(20), XA(4) RESPECTIVELY. DELAYS ARE ZERO FOR S.S.
FURTHERMORE THE FOLLOWING ARE CALCULATED

TAU (3 TIME DELAYS AND 1 LINEAR CORR.)

Y (3 OUTPUT VARIABLES (FEEDBACK TO NO OTHER VARIABLES))

XE (19 EXPLICIT VARIABLES FOR CHECKOUT PURPOSE)

EQUATIONS (61), (63), (64), (65), (67), (55) IN WINFRITH SIMULATOR

HAVE NOW BEEN CHANGED INTO EQUATIONS

(61.), (63.), (64.), (65.), (67.), (55.)

THIS CHANGE ENTAILS SOME MANIPULATION AND THE INTRODUCTION OF A TIME
CONSTANT OF 0.7 SECONDS FOR X(11), (M, MASS OF WATER AND 2-PHASE
MIXTURE IN PRIMARY CKT). THIS TIME CONSTANT APPEARS IN EQ. (55) OF THE
SIMULATOR.

THE FUNCTIONS AFFECTED ARE

F(10), F(11), F(12), F(13), G(15), G(16)

ALSO

F(10), F(13) DROP DENOMINATORS INVOLVING X(12) IN THE STEADY STATE
(THEY WILL BE REINTRODUCED FOR THE DYNAMIC SIMULATION)

THE SCALING FACTORS SCALF OF FUNCTIONS ARE CALCULATED IN SCLEVA AFTER THE
FIRST FUNCTIONS EVALUATION, INSTEAD OF BEING FIXED AS DATA. THE REASON IS
THAT SMALL CHANGES IN INITIAL VALUES MAY ENTAIL LARGE CHANGES IN F(I).
AT THE END OF EACH CALL OF CALFUN THE FUNCTIONS ARE MULTIPLIED BY SCALF,
CALLING SCALEF.

PROCEDURE FOR SCALING VARIABLES IN SCALEV (SCALING FACTORS ARE DATA HERE).

1. SCALE VARIABLES TO UNITY OPDER BEFORE CALLING MINIMIZATION ROUTINE
2. SCALE TERMS IN CALFUN WITH THE INVERSE OF THE SCALING FACTOR
3. SCALE VARIABLES BACK TO ORIGINAL VALUES AFTER COMPLETING MINIMIZATION

NOTE

C 4. CALL CALFUN (NO SCALING) AND WRITE VALUES BEFORE AND AFTER THE PROCED.

C
C
C THE NEW NONLINEAR SET CONSISTS OF 40 EQUATIONS (AND VARIABLES) ONLY.
C THE STEADY STATE SET HAS BEEN REDUCED TO 35 EQUATIONS (AND VARIABLES).
C THE PRINTOUT STILL CONSISTS OF ALL VARIABLES (AS WELL AS XE,Y AND TAU),
C SUCH THAT THE PUNCHOUT IS SUITED FOR DATA INPUT TO MXEV2Y1 (LINEAR MODEL).
C ACCORDINGLY, THE CALFUN ROUTINE CONTAINS 35 FUNCTIONS ONLY, BUT AT THE
C BEGINNING OF THE PROGRAM AND AT THE END, THE ORIGINAL SET OF 40 FUNCTIONS
C (CALFUNO) IS CALCULATED FOR CHECKING PURPOSES.
C SOME OF THE INPUTS (U(1),U(4) AND U(6)) ARE TREATED AS VARIABLES IN THE
C STEADY STATE, WHEREAS SOME OF THE VARIABLES (X(1),X(10) AND X(22)) ARE
C TREATED AS INPUTS AND X(12) IS AN ALTOGETHER IRRELEVANT CONSTANT.
C A SUBROUTINE ..REPLACE.. EXCHANGES THESE NAMES OF VARIABLES AND INPUTS AT
C THE START AND END OF THE PROGRAM, AND ALSO CALCULATES SOME VARIABLES FROM
C WHICH THE FUNCTIONS WERE REMOVED.
C IN PARTICULAR, FUNCTIONS F(1) AND F(22) ARE JUST LINEAR COMBINATIONS.
C F(19),F(21) AND F(23) HAVE BEEN REMOVED,THE VARIABLES COMPUTED SEPARATELY.

C
C
C SUBROUTINE READ
C .. OINPUT CALLS UIN AND STAAL
C .. CALFUN .. FUN1 TO FUN6 AND SCALEV, SCALEF
C .. IT IS REDUCED TO 35 FUNCTIONS AND SOME VARIABLES AND
C .. INPUTS CHANGE NAME
C .. CALFUNO CALLS FUN1 TO FUN6, IS THE ORIGINAL N/L SET FROM SIMULATOR
C .. REPLACE EXCHANGES ORIGINAL NAMES OF SOME VARIABLES AND INPUTS
C .. SCALEVA
C .. SCALEV
C .. VAG2A CALLS CALFUN AND V001A
C .. FUN1 TO FUN6 .. INTPOL
C .. CALCLT
C .. OUTPUT .. STAAL, YTAU, AND EXPL AND OPUNCH
C
C PEAL X(40),F(35),N(4370),E(35),XA(17),G(17),U(7),X130(1),X200(1),
1 XA40(1),XE(19),Y(3),XR(35)
C REAL SCALE(35)
C EQUIVALENCE (XR(19),XA(1)),(F(19),G(1))
C EQUIVALENCE (X(13),X130(1)),(X(20),X200(1)),(XA(4),XA40(1))
C EXTERNAL CALFUN
C M = 35
C N = 35
C NW = 4370
C NA = 17
C NA = 17
C NE = 19
C NU = 7
C NY = 3
C ESCAL = 1.E8
C DO 10 I=1,N
10 E(I) = 1.E-8
C IPPINT = 5
C MAXFUN = 1000
C ITAHW = 1
C ITAHFW = 1
C ITAIR = 1
C ITAURW = 1
C CALL READ (U,NA,NU,X,XA,U)
C CALL OINPUT (U,NA,NU,X,XA,U)
C CALL INITIAL
C CALL CALFUNO (X,XA,U,XE,ITAHW,ITAHFW,ITAIR,ITAURW,
1 X130,X200,XA40,M+5)
C CALL REPLACE (X,U,XR, 1,XE,NU,NE,XA,
R X1,X10,X22)
C CALL CALFUN(M,N,F,XR,NA,NA,NU,NE,G,XA,U,XE,X130,X200,XA40,

```

1          ITAUN,ITAUFW,ITAUP,ITAUW,SCAF,SSQ,4,

P          X1,X10,X22)
  WRITE (6,100) SSQ
100 FORMAT (1H0,* FIRST FUNCTION EVALUATION (NO SCALING)*,7X,*F=*,
1          E24.14//)
  WRITE (6,111) F
101 FORMAT (1X,7E17.8)
  CALL SCLEVA (F,M,SCAF)
  DUMMY = 1.
  CALL SCALEV(XR,XA,N,NA,DUMMY,DUMMY,1)
  CALL VAO2A(M,N,F,XR,E,ESCAL,IPRINT,MAXFUN,CALFUN,W,NH,MA,NA,NU,NE,
1          G,XA,U,XE,X13D,X20D,XA4D,
2          ITAUN,ITAUFW,ITAUW,ITAUW,SCAF,
P          X1,X11,X22)
  CALL SCALEV(XR,XA,N,NA,DUMMY,DUMMY,3)
  CALL CALFUN(M,N,F,XR,NA,NU,NE,G,XA,U,XE,X13D,X20D,XA4D,
1          ITAUN,ITAUFW,ITAUW,ITAUW,SCAF,SSQ,4,
P          X1,X10,X22)
  WRITE (6,200) SSQ
200 FORMAT (1H0,* FINAL FUNCTION EVALUATION (NO SCALING)*,7X,*F=*,
1          E24.14//)
  WRITE (6,101) F
  CALL REPLACE (X,U,XD, 2,XE,NU,NE,XA,
P          X1,X10,X22)
  CALL CALFUNO (X,XA,U,XE,ITAUN,ITAUFW,ITAUW,ITAUW,
1          X13D,X20D,XA4D,M+5)
  CALL CALCLT (N,NA,NU,NE,NY,X,XA,U,XE,Y,TAUW,TAUFW,TAUR,TAURW,X13D,
1          X20D,XA4D)
  CALL OUTPUT (N,NA,NU,NE,NY,X,XA,U,XE,Y,TAUW,TAUFW,TAUR,TAURW)
  CALL OPUNCH(N,NA,NU,NE,NY,X,XA,U,XE,Y,TAUW,TAUFW,TAUR,TAURW)
  STOP
  END

C
C
C
C SUBROUTINE CALFUNO (X,XA,U,XE,ITAUN,ITAUFW,ITAUW,ITAUW,
1          X13D,X20D,XA4D,M)
C
C ORIGINAL SET OF 40 EQUATIONS AS TAKEN FROM THE WINFRITH SIMULATOR.
C NUMBERS FOLLOWED BY A DOT INDICATE REARRANGEMENTS.
C SEE OUTPUT FOR MEANINGS OF VARIABLES
C
C REAL F(40),X(23),G(17),XA(17),U(7),XE(19),X13D(1),X20D(1),XA4D(1)
C EQUIVALENCE (F(24),G(1))
C
C EXPLICIT ALGEBRAIC EQUATIONS, THESE
C ARE NOT COORDINATES IN MINIMIZATION ROUTINE
C
C FN (X,U)
C
C REACTIVITY
C
C XE(1) = FUN1 (X(22) - 0.4)
C
C FUEL PIN TEMPERATURE
C
C XE(2) = 0.5*X(8) + 0.5*X(9)
C XE(3) = (1.1931*X(5) + 1.1569*X(6) + 2.5958*X(7))/X(22)
C
C CHANNEL AND RISER
C
C XE(4) = FUN2 (X(10))
C XE(5) = FUN3 (X(10))
C XE(6) = -3.99696E-2*X(13) + 5.1851

```

SIMUL.
EQ. NUMB.

3,9,135

29

30

69

33

53

```

C
C
C STEAM FLOW
C
C   XE(7) = 0.3862*X(10)*X(23) 83
C
C FEEDWATER
C
C   XE(8) = FUN5 (X(19)) 108
C
C FEEDHEAT
C
C   XE(9) = 9.67E-4*X(16) - 0.1082 123
C                                     XE(9)GE0
C
C RECIRCULATION
C
C   XE(10) = -5.31E-3*(X(14)*X(14)) + 3.578E-3*(X(15)*X(15)) 81
C   1      + 1.256E-3*(X(14)*X(15))
C   XE(11) = FUN6 (U(3)) 75
C
C FN (X,XE(1),...,XE(11),U)
C
C RECIRCULATION
C
C   XE(12) = XE(6)*X(14) 74
C
C FEEDHEAT
C
C   XE(13) = -0.408654*XE(7)*XE(9) + 0.408654*XE(7) - 1.42749E2 122
C
C FN (X,XE(1),...,XE(11),X0,XA0,U)
C
C FEEDHEAT
C
C   XE(14) = X200 (ITAUFW) 119
C
C CHANNEL AND RISER
C
C   XE(15) = X130 (ITAUN) 38
C   XE(16) = X130 (ITAURW) + XA40 (ITAUR) - XE(4) 49
C
C FN (X,XE(1),...,XE(17),U)
C
C CHANNEL AND RISER
C
C   XE(17) = 4.329E-7*(XE(16)*X(10)) + 1.11E-3*XE(16) 46
C   XE(18) = 2.1E4*XE(17)/X(10) 50
C   XE(19) = 0.31*XE(18) + 1.2 58
C
C DIFFERENTIAL EQUATIONS
C
C XDOT = F (X,XE,XA,U)
C
C NEUTRON KINETICS
C
C   F(1) = (2.5E3*XA(1) - 1.3E1)*X(1) + 3.9*X(2) + 9.1*X(3) 15
C   F(2) = 2.4E-2*X(1) - 2.4E-2*X(2) 16
C   F(3) = 0.15*X(1) - 0.15*X(3) 17
C

```

C MODERATOR TEMPERATURE

$$F(4) = 6.929E-2 * X(1) * X(22) - 3.03E-3 * X(4) + 3.03E-3 * U(2) \quad 22$$

X(4) GE0, LE100

C DECAY HEAT

$$F(5) = 9.53E-2 * X(1) * X(22) - 9.53E-2 * X(5) \quad 24$$

$$F(6) = 1.395E-2 * X(1) * X(22) - 1.395E-2 * X(6) \quad 25$$

$$F(7) = 4.0E-4 * X(1) * X(22) - 4.0E-4 * X(7) \quad 26$$

C FUEL PIN TEMPERATURE

$$F(8) = 6.736E1 * X(1) - 0.2564 * X(8) + 0.2564 * XE(2) + XE(3) \quad 27$$

$$F(9) = 6.736E1 * X(1) - 0.6051 * X(9) + 0.2564 * XE(2) + XE(3) \quad 28$$

1 + 0.3407 * X4(3)

C CHANNEL AND RISER

$$F(11) = -1.43 * X(11) + 1.43 * XE(6) * (1.0497E3 / (1. + XA(14))) + 1.2597E3 \quad 55.$$

C STEAM DRUM

$$F(12) = -F(11) - XA(15) + X(16) \quad 64.$$

X(12) GE0

$$F(10) = 1.16E-5 * X(10) * F(12) + 0.2639 * (XA(15) - XE(7)) \quad 61.$$

C INSERT THE FOLLOWING CARD FOR THE DYNAMIC SIMULATION

$$F(10) = F(10) / (1.0 - 1.0E-5 * X(12)) \quad 61.$$

$$F(13) = XE(4) * XA(16) - X(13) * XA(16) \quad 67.$$

$$1 - X(13) * X(16) + XE(14) * X(16)$$

C INSERT THE FOLL. CARD FOR THE DYN. SIM.

$$F(13) = F(13) / X(12) \quad 67.$$

C RECIRCULATION

$$F(14) = 4.444 * XE(10) - 4.444 * XA(17) \quad 77$$

NO DIRECT FEEDBK FROM STATE VAR.

$$F(15) = -7.197E-4 * (X(15) * X(15)) - 1.29525E1 * X(15) - 0.8903 * XE(10) \quad 82$$

1 + 2.03145E3

X(15) GE0

C FEEDWATER

$$F(16) = -5.2962E-2 * (X(16) * X(16)) - 3.704E1 * X(10) - 1.85195 * X(16) \quad 95$$

$$1 - 3.704E1 * X(17) + 5.05555E4$$

$$F(17) = 3.2E1 * X(16) - 2.09E3 * XE(8) * SQRT(X(17)) \quad 96$$

$$F(19) = -X(19) + U(4) \quad 107$$

X(19) GE0, LE1

XDOT(19) GE-0.1, LE0.1

C FEEDHEAT

$$F(20) = -2.339E-4 * X(16) * X(20) + 2.339E-4 * XE(13) * X(16) \quad 117$$

C POWER CONTROLLER

$$F(21) = -X(21) + U(5) \quad 133$$

$$F(22) = 8.56E-4 * X(21) \quad 134$$

X(21) GE-1, LE1

XDOT(21) GE-0.2, LE0.2

C TURBINE

$$F(23) = -4.0 * X(23) + 4.0 * U(6) - 2.0 * U(7) - 2.0 \quad 140$$

X(23) GE0, LE1

U(6)GE-1,LE1

TURBINE AND ALTERNATOR

$$F(18) = - 2.0 * X(18) + 0.6378 * XE(7)$$

175

IMPLICIT ALGEBRAIC EQUATIONS

$$G(X, XE, XA, U) = 0$$

REACTIVITY

$$G(1) = - XA(1) - 2.0E-2 * XA(2) - 2.778E-5 * XA(3) + 1.5E-4 * X(4) + XE(1) - 8.77E-6 * XE(2) + U(1)$$

14

CHANNEL AND RISER

$$G(2) = - XA(2) + 0.986 * XA(13) * XA(8) / XA(4)$$

44

$$G(3) = - XA(3) - 0.5 * XA(9) * XA(8) / XA(4) + 0.5 * XA(4) + XE(15)$$

36

$$G(4) = - XA(4) + 2.81E5 * XA(5) / XE(12)$$

31

$$G(5) = - XA(5) - 2.5118E-3 * XA(3) * X(22) + 2.5118E-3 * X(9) * X(22)$$

37

$$G(6) = - XA(6) + XE(5) * XA(7) + XE(4) - XE(15)$$

32

$$G(7) = - XA(7) + XA(10) + 0.5 * XA(11)$$

41

$$G(8) = - XA(8) + XA(4) - XA(6)$$

34

$$G(9) = - XA(9) + XA(7) + X(19)$$

40

$$G(10) = - XA(10) + 1.085E-3 * X(14) * X(14) * (XA(14) + 1.0)$$

54

$$G(11) = - XA(11) - 4.16 * XA(2) + 1.01314E-3 * X(14) * X(14) * XA(12)$$

42

$$G(12) = - XA(12) + 2.7514E1 * XA(8) / XA(9)$$

43

$$G(13) = - XA(13) + FUN4(XA(12))$$

45

$$G(14) = - XA(14) + 0.547645 * XA(12) / XE(19) + 0.5 * XE(19) / XE(19)$$

57

$$G(15) = - XA(15) + XE(17) * (XA(15) + XA(16))$$

63.

$$G(16) = - XA(16) - XA(15) + XE(12) - F(11)$$

65.

$$G(17) = - XA(17) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(18) = - XA(18) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(19) = - XA(19) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(20) = - XA(20) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(21) = - XA(21) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(22) = - XA(22) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(23) = - XA(23) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(24) = - XA(24) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(25) = - XA(25) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(26) = - XA(26) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(27) = - XA(27) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(28) = - XA(28) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(29) = - XA(29) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(30) = - XA(30) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(31) = - XA(31) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(32) = - XA(32) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(33) = - XA(33) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(34) = - XA(34) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(35) = - XA(35) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(36) = - XA(36) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(37) = - XA(37) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(38) = - XA(38) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(39) = - XA(39) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(40) = - XA(40) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(41) = - XA(41) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(42) = - XA(42) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(43) = - XA(43) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(44) = - XA(44) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

$$G(45) = - XA(45) + XA(10) + XA(11) + 2.1225E-2 * X(14) * X(14) * XE(11)$$

72

$$G(46) = - XA(46) + 3.13E-3 * X(14) * X(14) - 1.116E1$$

72

C

C THE SCALING FACTORS LISTED DIVIDE THE VARIABLES VALUES TO OBTAIN UNITY ORDER

C

```

DATA (SCAFX(I), I=1,19)/5.E 1, 1.E 0, 1.E 0, 5.E 1, 1.E 0, 1.E 0,
1      1.E 0, 1.E 3, 5.E 2, 5.E-1, 1.E 5, 1.E 0,
2      1.E 2, 1.E 2, 1.E 2, 5.E 2, 1.E 2, 1.E 2/,
4      (SCAFXA(I), I=1,17)/
5      1.E-2, 1.E-1, 1.E 2, 5.E 1, 1.E 0, 1.E 1,
6      5.E 1, 5.E 1, 1.E 3, 1.E 1, 1.E 1, 1.E 0,
7      5.E-1, 1.E 0, 5.E 2, 5.E 3, 5.E 1/
GO TO (10,20,30,40),ISCA

```

C

C

1. SCALE VARIABLES TO UNITY ORDER BEFORE CALLING MINIMIZATION ROUTINE

C

```

10 DO 101 T=1,18
101 X0(I) = X0(T)/SCAFX(I)
    DO 102 I=1,NA
102 XA0(I)= XA0(I)/SCAFXA(I)
    RETURN

```

C

C

2. SCALE TERMS IN CALFUN WITH THE INVERSE OF THE SCALING FACTOR

C

```

20 DO 201 T=1,18
201 X(I) = X0(I)*SCAFX(I)
    DO 202 I=1,NA
202 XA(I) = XA0(I)*SCAFXA(I)
    RETURN

```

C

C

3. SCALE VARIABLES BACK TO ORIGINAL VALUES AFTER COMPLETING MINIMIZATION

C

```

30 DO 301 I=1,18
301 X0(I) = X0(I)*SCAFX(I)
    DO 302 I=1,NA
302 XA0(I)= XA0(I)*SCAFXA(I)
    RETURN

```

C

C

4. CALL CALFUN(NO SCALING) AND WRITE VALUES BEFORE AND AFTER THE PROCEDURE

C

```

40 DO 401 I=1,18
401 X(I) = X0(I)
    DO 402 I=1,NA
402 XA(I) = XA0(I)
    RETURN
END

```

C

C

C

. SUBROUTINE SCALEF (F,M,SCAF)

C

C

SCALES FUNCTIONS AT THE END OF EACH CALFUN

C

```

.. REAL F(M),SCAF(M)
   DO 1 I=1,M
1 F(I) = F(I)*SCAF(I)
   RETURN
END

```

C

C

C

. SUBROUTINE SDALEVA (F,M,SCAF)

C

C

CALCULATES SCALING FACTORS OF FUNCTIONS ONCE, AFTER 1ST FUNCTION EVALUATION

C

```

. REAL F(M),SCAF(M)

```

```
DO 1 I=1,M
```

```
SCAF(I) = 1.
IF (ABS(F(I)) - .2E-2)1,1,2
2 SCAF(I) = .2E-2/ABS(F(I))
1 CONTINUE
RETURN
END
```

```
SUBROUTINE CALCLT (N,NA,NU,NE,NY,X,XA,U,XE,Y,TAUW,TAUFW,TAUR,TAURW
1 ,X130,X200,XA40)
```

```
COMPUTES VALUES NOT NEEDED FOR MINIMIZATION ROUTINE
```

```
REAL X(N),XA(NA),U(NU),XE(NE),Y(NY),X130(1),X200(1),XA40(1)
```

```
TIME DELAYS
```

```
FN (X,XE,XA)
```

```
RECIRCULATION
```

EQ. NUMB.

```
TAUW = 5.675E2/X(14)
```

73

```
FEEDHEAT
```

```
TAUFW = 8.55E3/X(16)
```

121

```
CHANNEL AND RISER
```

```
TAUP = 1.048E3/(XE(19)*X(14) + XE(19)*X(14)*XA(14))
TAUPW = TAUR + TAUP
```

52

```
OUTPUT VARIABLES (FEEDBACK TO NO OTHER VARIABLES)
```

```
FN (X,U)
```

```
ION CHAMBER FLUX
```

```
Y(1) = - 0.8727*X(1)*X(22) + 1.838*X(1)
```

18

```
MODERATOR TEMPERATURE
```

```
Y(2) = 2.0*X(4) - U(2)
```

23

```
U(2) = 27.7 DEGC
X(4)GE0,LE100 ""
Y(2)GE0,LE100 ""
```

```
FEEDWATER LEVEL
```

```
Y(3) = 1.268E-7*X(12) - 4.4E1
```

112

```
RETURN
END
```

```
FUNCTION FUN1 (Z)
```

C DKM(Reactivity Effect of Moderator Height) as Function of

C HDASH(Normalized Moderator Height) Minus 0.4

REAL X(13),Y(13)

DATA (X(I),I=1,13)/0.500,0.056,0.120,0.160,0.208,0.256,0.304,

1 0.352,0.400,0.448,0.496,0.544,0.600 / ,

2 (Y(I),I=1,13)/-0.02300,-0.31300,-0.00340,0.00165,0.00665,

3 0.01070, 1.01395, 0.01665,0.01910,0.02130,

4 0.02300,0.02455,0.02615/

L = 13

C FUN1 IS GIVEN A VALUE FOR COMPILATION REQUIREMENTS ONLY

FUN1 = 1.0

CALL INTPOL (L,X,Y,FUN1,7)

RETURN

END

C

C

C

FUNCTION FUN2 (7)

C HWSTAP(Enthalpy of Water at Saturation(Offset 400))

C AS FUNCTION OF P0(Steam Drum Pressure)

REAL X(13),Y(13)

DATA (X(I),I=1,13)/ 0.0,100.0,150.0,200.0,250.0, 300.0,400.0,

1 500.0,600.0,700.0,800.0,900.0,1000.0/,

2 (Y(I),I=1,13)/-198.195,-101.595,- 69.495,- 44.595,- 24.0,

3 - 6.195, 24.0 , 49.395, 71.595, 91.5,

4 109.695,126.6 , 142.395/

L = 13

FUN2 = 1.

CALL INTPOL (L,X,Y,FUN2,2)

RETURN

END

C

C

C

FUNCTION FUN3 (7)

C DHW/DP AS FUNCTION OF PRESSURE

REAL X(12),Y(12)

DATA (X(I),I=1,12)/ 0.0, 80.0,100.0,150.0,200.0, 250.0,

1 300.0,350.0,400.0,450.0,500.0,1000.0/,

2 (Y(I),I=1,12)/1.5 ,0.831,0.72,0.55,0.450,0.3834,

3 0.33,0.289,0.26,0.24,0.228,0.151/

L = 12

FUN3 = 1.

CALL INTPOL (L,X,Y,FUN3,7)

RETURN

END

C

C

C

FUNCTION FUN4 (7)

C A FUNCTION OF KXCOU(K * QUALITY AT CHANNEL OUTLET)

REAL X(13),Y(13)

DATA (X(I),I=1,13)/0. ,0.1 ,0.2 ,0.325,0.47 ,0.625,0.8125,

1 1.025,1.275,1.61 ,1.975,2.45 ,3.0 / ,

2 (Y(I),I=1,13)/0. ,0.0469 ,0.03885,0.13475,0.1803 ,

3 0.22325,0.2631 ,0.31165,0.3553 ,0.40415,

4 0.44795,0.49455,0.5379 /

L = 13

FUN4 = 1.

CALL INTPOL (L,X,Y,FUN4,7)

RETURN

END

C

C

```

C
      FUNCTION FUN5 (Z)
C A FUNCTION OF YR(100 PG FEEDWATER VALVE POSITION)
      REAL X(6),Y(6)
      DATA (X(I), I= 1,6)/0.0,0.76,0.14, 0.22 ,0.32 ,1.0/,
1      (Y(I), I =1,6)/0.0,0.012,0.039,0.082,0.168,1.0/
      L = 6
      FUN5 = 1.
      CALL INTPOL (L,X,Y,FUN5,Z)
      RETURN
      END

C
C
C
      FUNCTION FUN6 (Z)
C FUNCTION OF YDV, DISCHARGE VALVE CHARACTERISTIC
      REAL X(12),Y(12)
      DATA (X(I),I=1,12)/0. ,0.1333,0.25 ,0.3333,0.4167,0.5,
1      0.5833,0.6667,0.75 ,0.8333,0.9167,1.0/,
2      (Y(I),I=1,12)/0.02 ,0.0653,0.124 ,0.1733,0.2353,0.3067,
3      0.384 ,0.4667,0.5587,0.6567,0.811 ,1.0/
      L = 12
      FUN6 = 1.
      CALL INTPOL (L,X,Y,FUN6,Z)
      RETURN
      END

C
C
C
      SUBROUTINE INTPOL (L,X,Y,FUN,Z)
C INTERPOLATION ROUTINE
      REAL X(L), Y(L)
      L1 = L-1
      IF (Z-X(1))1,1,3
3 IF (Z-X(L ))2,4,4
2 DO 5 J=1,L1
      K=J+1
      DZ=Z-X(J)
      DX=X(K)-X(J)
      IF(DZ-DX) 6,6,5
5 CONTINUE
6 FUN = Y(J) + DZ*(Y(K)-Y(J))/DX
      RETURN
1 FUN = Y(1)
      RETURN
4 FUN = Y(L )
      RETURN
      END

C
C
C
      SUBROUTINE V002A (N,NF,X,E,ESCALE,IPRINT,MAXFUN,CALFUN,W,NM3N,MA,
1      NA,NU,NE,G,XA,U,XE,X13D,X20D,XA4D,ITAUW,ITAUFW,
2      ITAUR,ITAURW,SCAF,X1,X10,X22)

C
C MINIMIZATION OF A SUM OF SQUARES OF NONLINEAR FUNCTIONS (POWELL ALGORITHM),
C CALLS V001A.
C
C F(N) = FUNCTIONS
C X(N) = VARIABLES
C E(I) = ACCURACY OF VARIABLE X(I)
C ESCALE = LIMITS THE STEP SIZE OF THE SEARCH. NORMALLY X(I) WILL NOT BE
C CHANGED BY MORE THAN ESCALE*E(I) IN A SINGLE STEP.
C IPRI NT = PERIODICAL NUMBER OF ITERATIONS AFTER WHICH THE VALUES OF VARIABLES

```

```

C      AND FUNCTIONS ARE PRINTED. NO PRINTOUT FOR IPRINT = 0.

C      CALFUN = ROUTINE WHERE THE FUNCTIONS ARE EVALUATED
C      MAXFUN = MAXIMUM NUMBER OF CALLS OF CALFUN
C      W(NM3N) = AUXILIARY ARRAY WHERE NM3N.GE. (N + (M + (3*N+1)/2)*(N+1))
C      G(MA),XA(NA),U(NU),XE(NE),X13D,X20D,XA4D,ITAUW,ITAUFW,ITAUW,ITAURW,SCAF,X1,
C      X10,X22 ARGUMENTS FOR CALFUN, DUE TO THE PARTICULAR STRUCTURE OF THAT
C      ROUTINE AND EXPLAINED ELSEWHERE IN THIS PROGRAM. THEY ARE NOT
C      INHERENT TO VAG2A.
C
      REAL F(M),X(N),E(N),G(MA),XA(NA),U(NU),X13D(1),X20D(1),XA4D(1),
1      XE(NE),SCAF(M)
      REAL W(NM3N)
      EXTERNAL CALFUN
      IF(NM3N-(N+(M+(3*N+1)/2)*(N+1))) 8000,8002,8002
8000 WRITE(6,8001)
8001 FORMAT(26H DIMENSION OF W TOO SMALL )
      STOP
8002 CONTINUE
100 FORMAT (1H ,I10)
      MPLUSN=M+N
      KST=N+MPLUSN
      NPLUS=N+1
      KINV=NPLUS*(MPLUSN+1)
      KSTORE=KINV-MPLUSN-1
      CALL CALFUN (M,N,F,X,MA,NA,NU,NE,G,XA,U,XE,X13D,X20D,XA4D,
1      ITAUW,ITAUFW,ITAUW,ITAURW,SCAF,SSQ,2,
R      X1,X10,X22)
      NN=N+N
      K=NN
      DO 1 I=1,M
      K=K+1
      W(K)=F(I)
1 CONTINUE
      IINV=2
      K=KST
      I=1
2 X(I)=X(I)+E(I)
      CALL CALFUN (M,N,F,X,MA,NA,NU,NE,G,XA,U,XE,X13D,X20D,XA4D,
1      ITAUW,ITAUFW,ITAUW,ITAURW,SCAF,SSQ,2,
R      X1,X10,X22)
      X(I)=X(I)-E(I)
      DO 3 J=1,N
      K=K+1
      W(K)=F(J)
      W(J)=0.
3 CONTINUE
      SUM=0.
      KK=NN
      DO 4 J=1,M
      KK=KK+1
      F(J)=F(J)-W(KK)
      SUM=SUM+F(J)*F(J)
4 CONTINUE
      IF (SUM) 5,5,6
5 WRITE(6,7) I
7 FORMAT (5X,3HVA02A E(,I3,20H) UNREASONABLY SMALL)
      DO 8 J=1,M
      NN=NN+1
      F(J)=W(NN)
8 CONTINUE
      GO TO 10
6 SUM=1./SQRT (SUM)
      J=K-N+1
      W(J)=E(I)*SUM

```

```

      DO 9 J=1,M

      K=K+1
      W(K)=F(J)*SUM
      KK=NN+J
      DO 11 II=1,T
      KK=KK+MPLUSN
      W(II)=W(II)+W(KK)*W(K)
11  CONTINUE
      9  CONTINUE
      ILESS=I-1
      IGAMAX=N+I-1
      INCINV=N-ILESS
      INCINP=INCINV+1
      IF (ILESS) 13,13,14
13  W(KINV)=1.
      GO TO 15
14  R=1.
      DO 16 J=NPLUS,IGAMAX
      W(J)=0.
16  CONTINUE
      KK=KINV
      DO 17 II=1, ILESS
      IIP=II+N
      W(IIP)=W(IIP)+W(KK)*W(II)
      JL=II+1
      IF (JL-ILESS) 18,18,19
18  DO 20 JJ=JL, ILESS
      KK=KK+1
      JJP=JJ+N
      W(IIP)=W(IIP)+W(KK)*W(JJ)
      W(JJP)=W(JJP)+W(KK)*W(II)
20  CONTINUE
19  R=R-W(II)*W(IIP)
      KK=KK+INCINP
17  CONTINUE
      R=1./R
      KK=KINV
      DO 21 II=NPLUS,IGAMAX
      RB=-R*W(II)
      DO 22 JJ=II,IGAMAX
      W(KK)=W(KK)-3R*W(JJ)
      KK=KK+1
22  CONTINUE
      W(KV)=RB
      KK=KK+INCINV
21  CONTINUE
      W(KK)=R
15  GO TO (27,24),IINV
24  I=I+1
      IF (I-N) 2,2,25
25  IINV=1
      FF=0.
      KL=NN
      DO 26 I=1,M
      KL=KL+1
      F(I)=W(KL)
      FF=FF+F(I)*F(J)
26  CONTINUE
      ICONT=1
      ISS=1
      MC=N+1
      IPP=IPRINT*(IPRINT-1)
      ITC=0
      IPS=1

```

IPC=0

```

27 IPC=IPC-IPRINT
   IF (IPC) 23,29,29
28 WRITE (6,30) ITC, MC, EF
30 FORMAT (//5X,94ITERATION,I4,I9,16H CALLS OF CALFUN,5X,2HF=,E24.14)
   WRITE(6,31) X
31 FORMAT(10H VARIABLES/(1X,7E17.8))
   WRITE(6,32) F
32 FORMAT(10H FUNCTIONS/(1X,7E17.8))
   CALL SECONO(S)
   IF (S.GE.59.0) GO TO 33
   IPC=IPP
   GO TO (29,33),IPS
29 GO TO (34,35),ICONT
35 IF (CHANGE-1.) 10,10,36
10 IF (IPRINT) 33,33,37
37 WRITE (6,38)
38 FORMAT (//5X,45HVA02A FINAL VALUES OF FUNCTIONS AND VARIABLES)
   IPS=2
   GO TO 28
33 RETURN
36 ICONT=1
34 ITC=ITC+1
   K=N
   KK=KST
   DO 39 I=1,N
   K=K+1
   W(K)=0.
   KK=KK+N
   W(I)=0.
   DO 40 J=1,M
   KK=KK+1
   W(I)=W(I)+W(KK)*F(J)
40 CONTINUE
39 CONTINUE
   OM=0.
   K=KINV
   DO 41 II=1,N
   IIP=II+N
   W(IIP)=W(IIP)+W(K)*W(II)
   JL=II+1
   IF (JL-N) 42,42,43
42 DO 44 JJ=JL,N
   JJP=JJ+N
   K=K+1
   W(IIP)=W(IIP)+W(K)*W(JJ)
   W(JJP)=W(JJP)+W(K)*W(II)
44 CONTINUE
   K=K+1
43 IF (OM-ABS (W(II)*W(IIP))) 45,41,41
45 OM=ABS (W(II)*W(IIP))
   KL=II
41 CONTINUE
   II=N+MPLUSN*KL
   CHANGE=0.
   DO 46 I=1,N
   JL=N+I
   W(I)=0.
   DO 47 J=NPLUS,NM
   JL=JL+MPLUSN
   W(I)=W(I)+W(J)*W(JL)
47 CONTINUE
   II=II+1
   W(II)=W(JL)

```



```

      W(JL)=X(I)

      IF (ABS (E(I)*CHANGE)-ABS (W(I))) 48,48,46
48  CHANGE=ABS (W(I)/E(I))
46  CONTINUE
      DO 49 I=1,M
      II=II+1
      JL=JL+1
      W(II)=W(JL)
      W(JL)=F(I)
49  CONTINUE
      FC=FF
      ACC=0.1/CHANGE
      IT=3
      XC=0.
      XL=0.
      IS=3
      XSTEP=-AMIN1(0.5,ESCALE/CHANGE)
      IF (CHANGE-1.) 50,50,51
50  ICOUT=2
51  CALL V001A (IT,XC,FC,6,ACC,0.1,XSTEP)
      GO TO (52,53,53,53),IT
52  MC=MC+1
      IF (MC-MAXFUN) 54,54,55
55  WRITE (6,56) MAXFUN
56  FORMAT (5X,5HVA02A,I6,16H CALLS OF CALFUN)
      ISS=2
      GO TO 53
54  XL=XC-XL
      DO 57 J=1,N
      X(J)=X(J)+XL*W(J)
57  CONTINUE
      XL=XC
      CALL CALFUN (M,N,F,X,MA,NA,NU,NE,G,XA,U,XE,X130,X200,XA40,
1      ITAUN,ITAUFW,ITAUW,SCAF,SSQ,2,
R      X1,X10,X22)
      FC=P.
      DO 58 J=1,M
      FC=FC+F(J)*F(J)
58  CONTINUE
      GO TO (59,59,60),IS
60  K=N
      IF (FC-FF) 61,51,62
61  IS=2
      FMIN=FC
      FSEC=FF
      GO TO 63
62  IS=1
      FMIN=FF
      FSEC=FC
      GO TO 63
59  IF (FC-FSEC) 64,51,51
64  K=KSTORE
      GO TO (75,74),IS
75  K=N
74  IF (FC-FMIN) 65,51,66
66  FSEC=FC
      GO TO 63
65  IS=3-IS
      FSEC=FMIN
      FMIN=FC
63  DO 67 J=1,N
      K=K+1
      W(K)=X(J)
67  CONTINUE

```

```

DO 68 J=1,M

    K=K+1
    W(K)=F(J)
68 CONTINUE
    GO TO 51
53 K=KSTORE
    KK=N
    GO TO (69,70,69),IS
70 K=N
    KK=KSTORE
69 SUM=0.
    DM=0.
    JJ=KSTORE
    DO 71 J=1,N
        K=K+1
        KK=KK+1
        JJ=JJ+1
        X(J)=W(K)
        W(JJ)=W(K)-W(KK)
71 CONTINUE
    DO 72 J=1,M
        K=K+1
        KK=KK+1
        JJ=JJ+1
        F(J)=W(K)
        W(JJ)=W(K)-W(KK)
        SUM=SUM+W(JJ)*W(JJ)
        DM=DM+F(J)*W(JJ)
72 CONTINUE
    GO TO (73,10),ISS
73 J=KINV
    KK=NPLUS-KL
    DO 76 I=1,KL
        K=J+KL-I
        J=K+KK
        W(I)=W(K)
        W(K)=W(J-1)
76 CONTINUE
    IF (KL-N) 77,78,78
77 KL=KL+1
    JJ=K
    DO 79 I=KL,N
        K=K+1
        J=J+NPLUS-I
        W(I)=W(K)
        W(K)=W(J-1)
79 CONTINUE
    W(JJ)=W(K)
    R=1./W(KL-1)
    W(KL-1)=W(N)
    GO TO 88
78 R=1./W(N)
88 K=KINV
    DO 80 I=1,ILESS
        BB=R*W(I)
        DO 81 J=I,ILESS
            W(K)=W(K)-BB*W(J)
            K=K+1
81 CONTINUE
        K=K+1
80 CONTINUE
    IF (FMIN-FF) 82,83,83
83 CHANGE=0.
    GO TO 84

```

82 FF=FMIN

```

      CHANGE=ABS (XC)*CHANGE
84  XL=-DM/FMIN
      SUM=1./SQRT (SUM+DM*XL).
      K=KSTOPE
      DO 85 I=1,N
      K=K+1
      W(K)=SUM*W(K)
      W(I)=0.
85  CONTINUE
      DO 86 I=1,M
      K=K+1
      W(K)=SUM*(W(K)+XL*F(I))
      KK=NN+I
      DO 87 J=1,N
      KK=KK+4*PLUSN
      W(J)=W(J)+W(KK)*W(K)
87  CONTINUE
86  CONTINUE
      GO TO 14
      END

```

C
C
C

```

      SUBROUTINE VD01A (ITEST,X,F,MAXFUN,ABSACC,RELACC,XSTEP)
      ABSF(ABC123)=ABS(ABC123)
      GO TO (1,2,2),ITEST
2   IS=6-ITEST
      ITEST=1
      IINC=1
      XINC=XSTEP*XSTEP
      MC=IS-3
      IF (MC) 4,4,15
3   MC=MC+1
      IF (MAXFUN-MC) 12,15,15
12  ITEST=4
43  X=DR
      F=FR
      IF (FB-FC) 15,15,44
44  X=DC
      F=FC
15  RETURN
      1 GO TO (5,6,7,8),IS
      8 IS=3
      4 DC=X
      FC=F
      X=X+XSTEP
      GO TO 3
      7 IF (FC-F) 9,10,11
10  X=X+XINC
      XINC=XINC+XINC
      GO TO 3
      9 DR=X
      FR=F
      XINC=-XINC
      GO TO 13
11  DB=DC
      FR=FC
      DC=X
      FC=F
13  X=DC+DC-DR
      IS=2
      GO TO 3
      6 DA=DR

```

```

DB=DC

FA=FB
FR=FC
32 DC=X
FC=F
GO TO 14
5 IF (FR-FC) 16,17,17
17 IF (F-FR) 18,32,32
18 FA=FR
DA=DB
19 FR=F
DB=X
GO TO 14
16 IF (FA-FC) 21,21,20
20 XINC=FA
FA=FC
FC=XINC
XINC=DA
DA=DC
DC=XINC
21 XINC=DC
IF ((D-DB)*(D-DC)) 32,22,22
22 IF (F-FA) 23,24,24
23 FC=FB
DC=DB
GO TO 19
24 FA=F
DA=X
14 IF (FB-FC) 25,25,29
25 IINC=2
XINC=DC
IF (FB-FC) 29,45,29
29 D=(FA-FB)/(DA-DB)-(FA-FC)/(DA-DC)
IF (D*(DB-DC)) 33,37,37
37 D=0.5*(DB+DC-(FB-FC)/D)
IF (ABSF(D-X)-ABSF(ABSACC)) 34,34,35
35 IF (ABSF(D-X)-ABSF(D*PELACC)) 34,34,36
34 ITEST=2
GO TO 43
36 IS=1
X=D
IF ((DA-DC)*(DC-D)) 3,26,38
38 IS=2
GO TO (39,40),IINC
39 IF (ABSF(XINC)-ABSF(DC-D)) 41,3,3
33 IS=2
GO TO (41,42),IINC
41 X=DC
GO TO 10
40 IF (ABSF(XINC-X)-ABSF(X-DC)) 42,42,3
42 X=0.5*(XINC+DC)
IF ((XINC-X)*(X-DC)) 26,26,3
45 X=0.5*(DB+DC)
IF ((DB-X)*(X-DC)) 26,26,3
26 ITEST=3
GO TO 43
END

```

APPENDIX B.

Listing of the more important routines in MXEV2X1

(Routines listed in Appendix A are not repeated. CALFUN is CALFUNO in SSNL2D).

MXEV2X1

DELAY1

FDBCK1

TESTMX

TESTMX2

COMPARE

DRAWMX

DRAWIM

SCAN

SLOPE1

SLOPE2

SLOPE3

SLOPE4

SLOPE5

SLOPE6

INVERT

PROGRAM MXEV2X1 (INPUT,OUTPUT,PUNCH,TAPES=INPUT,TAPES=OUTPUT)

3.11.70 (L. TEPPER, CONTROL SYSTEMS SECTION, DEPARTMENT OF
COMPUTING AND CONTROL, IMPERIAL COLLEGE)

LINEAR MODEL CONSTRUCTION

MATRIX EVALUATION PROGRAM FOR LINEAR MODEL

THE INPUT VALUES FOR THIS PROGRAM ARE THE STEADY STATE VALUES COMPUTED IN
PROG. SSNL20, X,XA,XE,U. THE DAP(FOR STEADY STATE) HAS BEEN REMOVED.

THE BASIC EQUATIONS, DERIVED BY LINEARIZATION OF THE N/L MODEL, ARE

$$\dot{X}(T) = A \cdot X(T) + B \cdot XA(T) + D \cdot XD(T) + C \cdot U(T) \quad (1)$$

$$0 = D \cdot X(T) + E \cdot XA(T) + ED \cdot XD(T) + F \cdot U(T) \quad (2)$$

WHERE

X IS THE VECTOR OF 23 STATE VARIABLES

XA 17 ALGEBRAIC VARIABLES

U 7 INPUTS

XD 4 DELAYED VARIABLES (3 STATE , 1 ALGEBR.)

A,B,D,C ARE MATRICES OF 23 ROWS EACH

D,E,ED,F 17

THE PURE TIME DELAYS ARE APPROXIMATED IN THE FOLLOWING GENERAL WAY

$$\dot{Z}(T) = P \cdot X(T) + PA \cdot XA(T) + Q \cdot Z(T) \quad (3)$$

$$XD(T) = G \cdot X(T) + GA \cdot XA(T) + H \cdot Z(T) \quad (4)$$

WHERE Z IS A VECTOR OF L NEW STATE VARIABLES (L = 4 FOR FIRST ORDER APPROX.)

Z = [Z1 Z2 Z3 Z4]T

T = TRANSPOSE

P,PA,Q = MATRICES OF L ROWS EACH

G,GA,H = ... 4

THE FOLLOWING SET HOLDS FOR EACH TIME DELAY

$$\dot{Z}_{OTI} = P_I \cdot X_{NI} + Q_I \cdot Z_I \quad (5) \quad I = 1,2,3,4$$

$$X_{DI} = G_I \cdot X_{NI} + H_I \cdot Z_I \quad (6) \quad T = TRANSPOSE$$

XN1 = X(20)

XN2 = X(13)

XN3 = X(13)

XN4 = XA(4)

WHERE

LI = ORDER OF APPROXIMATION OF XDI

ZT = VECTOR OF LI STATE VARIABLES

PI = ... ELEMENTS SET ACCORDING TO DESIRED TYPE OF APPROXIMATION

HI = ...

OT = MATRIX (LI,LI), ...

GI = SCALAR

L = L1 + ... + L4

THE MATRICES ON (3),(4) ARE FORMED BY MATRICES AND VECTORS FROM (5),(6) IN
THE FOLLOWING MANNER

$$P = \begin{bmatrix} P_1 & \dots & P_L \end{bmatrix} \quad PA = \begin{bmatrix} P_1 & \dots & P_L \end{bmatrix} \quad Q = \begin{bmatrix} Q_1 & \dots & Q_L \end{bmatrix} \quad (3A)$$

C (L,23) (0.P3*XX.. .0) (L,17) (0... 0) (L,L) (0Q3*XX 0)

C [0.. P*XX.. .0] [0... 0] [0. P*XX 0]
 C [0... .0] [P4*XX.. 0] [0. Q4*XX]
 C [0... .0] [0.P*XX.. 0] [0. P*XX]

↑ ↑
 13 20 4

C [0.. G1*XX.0] [0... .0] HIT*(XX.....0)
 C G = [0.G2*XX. .0] GA = [0... .0] H = [0.XX...0] (4A)
 C (4,23) [0.G3*XX.. .0] (4,17) [0... .0] (4,L) [0...XX.0]
 C [0.. .0] [G4*XX.. .0] [0.H4T*XX]

↑ ↑
 13 20 4

C NOW, IF

C $X_1 = [X \ 7]T$ WITH $23+L$ STATE VARIABLES

C WE OBTAIN THE FOLLOWING EQUATIONS WHICH INCLUDE (1),(2),(3),(4)

$$C \quad \dot{X}_1(t) = A_1 * X_1(t) + B_1 * X_A(t) + C_1 * U(t) \quad (7)$$

$$C \quad 0 = D_1 * X_1(t) + E_1 * X_A(t) + F * U(t) \quad (8)$$

C WHERE $A_1(23+L, 23+L)$, $B_1(23+L, 17)$ AND $C_1(23+L, 7)$

$$C \quad \begin{bmatrix} A_1 & B_1 \\ D_1 & E_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (7A)$$

C $D_1(17, 23+L)$ AND $E_1(17, 17)$ ARE

$$C \quad \begin{bmatrix} D_1 & E_1 \end{bmatrix} = \begin{bmatrix} D & E \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (8A)$$

C FINALLY, ELIMINATING X_A FROM (7),(8), THE EQUIVALENT LINEAR MODEL IS

$$C \quad \begin{aligned} &***** \\ &* \quad \dot{X}_1(t) = AS * X_1(t) + BS * U(t) \quad * \\ &* \quad ***** \end{aligned} \quad (9)$$

$$C \quad \text{WHERE } AS = A_1 - B_1 * E1V * D_1 \quad (10)$$

$$C \quad BS = C_1 - B_1 * E1V * F \quad (11)$$

C $E1V = \text{INVERSE OF MATRIX } E_1$

C 28.1.70

C COMPREHENSIVE MODEL, ADDING (EXISTING) CONTROLS TO THE SYSTEM.
 C FORCK1 IS THE NEW ROUTINE INTRODUCED FOR THAT PURPOSE.

C THE COMPREHENSIVE MODEL IS NOW

$$C \quad \begin{aligned} &***** \\ &* \quad \dot{X}_2(t) = AF * X_2(t) + BF * U_2 \quad * \\ &* \quad ***** \end{aligned} \quad (13)$$

```

C *****
C
C WHEPE
C
C X2 = (X1,UIT) STATE VECTOR OF LENGTH NX2 = NL + NW
C W = NEW STATE VARIABLES INTRODUCED IN FDBCK, LENGTH NW
C UR = INPUT VECTOR OF REFERENCE POINTS, LENGTH NU2 = NU + NUN
C (INPUTS RECEIVING FEEDBACK ARE NIL IN THIS VECTOR)
C
C SEE ALL FURTHER COMMENTS IN FDBCK ROUTINE
C
C 30.12.69
C
C TO AVOID REPEATED EIGENVALUES, TRY SMALL CHANGES IN CALFUN
C
C F(19) = - .95*X(19) + .95*U(4) INSTEAD OF TIME CONSTANT 1 SEC
C F(21) = -1.05*X(21) + 1.05*U(5) INSTEAD OF TIME CONSTANT 1 SEC
C
C AND THE CORRESPONDING ELEMENTS IN AMX AND CMX
C
C A(19,19) = - .95 C(19,4) = .95
C A(21,21) = - 1.05 C(21,5) = 1.05
C
C IF SUCCESSFULL, RECALL CHANGING CALFUN AND CALFUND IN SSNL2
C
C LIST OF DIMENSIONS AND ARRAYS
C
C N = NUMBER OF STATE VARIABLES, BEFORE INTRODUCING THE DELAYS APPROXIMATION
C NA = NUMBER OF ALGEBRAIC VARIABLES
C NU = NUMBER OF INPUTS
C NE = NUMBER OF EXPLICIT VARIABLES (IN SSNL20)
C NY = NUMBER OF OUTPUTS
C (N TO NY ARE CARRIED OVER FROM SSNL20)
C ND = NUMBER OF DELAYED VARIABLES
C L = NUMBER OF STATE VARIABLES ADDED BY DELAYS APPROXIMATION
C NL = NUMBER OF STATE VARIABLES (TOTAL) AFTER DELAYS APPROXIMATION
C NUN = NUMBER OF INPUTS INTRODUCED IN FDBCK1 FOR THE COMPREHENSIVE MODEL
C NW = NUMBER OF STATE VARIABLES INTRODUCED IN FDBCK1
C NX2 = NUMBER OF STATE VARIABLES (TOTAL) OF COMPREHENSIVE MODEL
C NU2 = NUMBER OF INPUTS (TOTAL) OF COMPREHENSIVE MODEL
C NY = NUMBER OF OUTPUTS (TOTAL) OF COMPREHENSIVE MODEL
C
C X(N),XA(NA),U(NU),
C XE(NE),Y(NY) STEADY STATE VALUES CALCULATED IN SSNL2
C T1A,T1D,T2D,T1E PURE TIME DELAYS .. .. .
C A(N,N),R(N,NA),B(N,ND),
C C(N,NU),D(NA,N),E(NA,NA)
C ED(NA,ND),F(NA,NU) ELEMENTS OF MATRICES OF BASIC LINEAR MODEL (1,2)
C AS(NL,NL),G(NL,NL) EQUIVALENT LINEAR MODEL MATRICES (9)
C
C P1(L1),P2(L2),P3(L3),P4(L4),
C H1(L1),H2(L2),H3(L3),H4(L4),
C O1(L1,L1),O2(L2,L2),O3(L3,L3),
C O4(L4,L4),G1,G2,G3,G4 ELEMENTS SET TO DESIRED TYPE OF DELAY APPROX. (5,6)
C
C P(L,N),PA(L,NA),Q(L,L)
C G(ND,N),GA(ND,NA),H(ND,L) MATRICES COMPOSED OF PREVIOUS ELEMENTS (3,4)
C
C A2(N,N),P2(N,L),E2(NA,L)
C D2(NA,N) WORKING MATRICES (7A,8A)
C
C A1(NL,NL),P1(NL,NA),O1(NL,NU),
C D1(NA,NL),E1(NA,NA) MATRICES IN STEP TOWARDS FINAL MODEL (7,8)
C

```



```

C      E1V(NA,NA),EEV(NA,NA)      INVERSE OF E1 AND PRODUCT E1*E1V (10,11)

C      QV(L,L),QQV(L,L)           INVERSE OF Q AND PRODUCT Q*QV
C      WAL(NA,NL),WAN(NA,NU)      WORKING MATRICES (10,11)
C
C      WNL1(NL,NL),WNL2(NL,NL)    WORKING MATRICES FOR EIGENVALUE CALCULATION
C      RTASP(NL),RTASI(NL)        REAL AND IMAGINARY ROOTS OF AS
C      RF(N),RG(NA),RX(N),RYA(NA),RU(NH),
C      FBAP(N),GBAP(NA),VX(NA),VNA(NA),
C      HQV(NH,L),GT(NH,N),GAT(NH,NA),
C      WNN(N,N),WNA(N,NA),
C      WAN(NA,N),WAA(NA,NA),
C      ERPF(N),ERPG(NA),
C      ERPFPC(N),ERPGPC(NA)        VARIABLES FOR MATRIX TESTING ROUTINE
C      AT(N,N),AZ(N,L),GT(N,NU),
C      PT(L,N),QT(L,L),FT(L,NU),
C      EDV(NA,NA),DBA(N,NA),WNO(N,NO),
C      WLA(L,NA),WLO(L,NO)        WORKING MATRICES FOR TESTMX(PART II)
C      M1(N),MA1(NA),MU1(NH),
C      NO1(NO),LL1(L),NL1(NL)    VARIABLES FOR DRAWLM ROUTINE
C      IPPNT1                      PRINT AND DRAW MATRICES IN MXEVA2 (EXCEPT AS AND
C                                  RS) FOR ANY VALUE OTHER THAN 0
C      IPPNT2                      ID FOR AS AND RS
C      IPPNT3                      ID FOR ALL NEW MATRICES IN FORCK
C      IPPNT4                      ID FOR INTERMEDIATE SUBMATRICES OF FORCK
C      IPPNT5                      ID FOR AE AND OF IN FORCK
CC
C LIST OF SUBROUTINES
C
C      READ                        READS STEADY STATE VALUES
C      QINPUT                      WRITE INPUT VALUES( RESULTS FROM SSNL2)
C      UIN,STAAL,EXPL,YTAU
C
C      AMX,PMX,DMX,CMX            EVALUATE MATRICES OF EQ (1)
C      EMX,EMX,EDMX,FMX          .. .. .. (2)
C      DELAY                      SETS ORDERS OF DELAY APPROX. AND VALUES (5,6)
C      PMX,PAMX,OMX              EVALUATE MATRICES OF EQ (3) AND QV,QQV
C      GMX,GAMX,HMX              .. .. .. (4)
C      AIMX,BIMX,CIMX,DIMX,EIMX .. .. .. (7,8) AND E1V, EEV
C
C
C      WRIMX                      WRITE ELEMENTS OF MATRICES
C      DRAWMX                     DRAWS .. .. ..
C      DRAWLM                     DRAWS COMPLETE LINEAR MODEL
C      SCAN                       AUXILIARY TO DRAWLM
C      OPUNCH                     PUNCH ELEMENTS OF EQUIVALENT L. M. MATRICES
C      CALFUN                     EVALUATES FUNCTIONS OF N/L MODEL
C      FUN1,...,FUN6              TABLES OF EMPIRICAL FUNCTIONS
C      SLOPF1,...,SLOPF6          TABLES OF FIRST DERIVATIVES
C      INTPOL                     SIMPLE INTERPOLATION
C      MULTT                      MATRIX MULTIPLICATION
C      INVERT                     .. INVERSION
C      NULL                       .. SET TO NULL
C      ADD                        .. ADDITION
C      SUBTR                       .. SUBTRACTION
C      NEGATE                     .. NEGATIVE OF
C      IMBED                      TRANSFER OF SUBMATRIX TO PARTITIONED MATRIX
C      IMREDV                     TRANSFER OF VECTOR TO MATRIX
C      XEIGEN                     CALCULATES AND WRITES EIGENVALUES OF AS
C      ETGB                       /
C      DETMT                      /
C      HESSEN                     / EIGENVALUES PACKAGE
C      OPEIG                      /
C      ORT                       /
C      TESTMX                     TESTS MATRICES BY COMPARISON WITH NONLINEAR MODEL

```

```

C      TESTMX2                TESTS CORRECTNESS OF EQUIVALENT LINEAR MODEL

CC
C      FDBCK1                 NEW ROUTINE CHANGING MXEVA2 INTO MXEVA2X
C                             ADDING NEW (FEEDBACK) EQUATIONS TO THE SYSTEM
C
C LIST OF VARIABLES FOR FDBCK1 ROUTINE
C
C      BN(NL,NUIN),BSN(NL,NU2)  EXTENSION OF MATRIX BS                (9,9A)
C      CS(NY,NL),R(NW,NW),S(NW,NY),
C      DD(NW,NU2),T(NU2,NW),
C      V(NU2,NY),EE(NU2,NU2)    NEW MATRICES INTRODUCED BY FDBCK      (14,15,16)
C      AF(NX2,NX2),BF(NX2,NU2)  COMPREHENSIVE MODEL MATRICES          (13)
C
C
C      REAL X(23),XA(17),U(7),XE(19),Y(3)
C      REAL A(23,23),B(23,17),B2(23,4),C(23,7),
1      D(17,23),E(17,17),ED(17,4),F(17,7)
C
C      ASSUME MAXIMUM ORDER 3 FOR DELAY APPROXIMATION
C
C      REAL P1(3),P2(3),P3(3),P4(3),H1(3),H2(3),H3(3),H4(3),
1      Q1(3,3),Q2(3,3),Q3(3,3),Q4(3,3)
C      REAL P(12,23),PA(12,17),Q(12,12),G(4,23),GA(4,17),H(4,12)
C      REAL A2(23,23),B2(23,12),D2(17,23),E2(17,12)
C DIMENSIONS REDUCED TO ENVISAGE 1ST ORDER DELAY APPROXIMATIONS ONLY
C      REAL A1(27,27),B1(27,17),C1(27,7),D1(17,27),E1(17,17)
C      REAL E1V(17,17),EEV(17,17),QV(12,12),QOV(12,12)
C      REAL AS(27,27),BS(27,7),WAL(17,27),WAV(17,7)
C      REAL WNL1(31,71),WNL2(31,31),PTAS2(31),PTAS1(31),WNL3(31,31)
C      REAL DF(23),DG(17),DX(23),DXA(17),DU(7),FRAP(23),GRAP(17),VN(23),
1      VNA(17),ERRF(23),ERRG(17),ERRGPC(23),ERRGPC(17),
2      HQV(4,12),GT(4,23),GAT(4,17),
3      WNN(23,23),WNA(23,17),WAN(17,23),WAA(17,17)
C      REAL XTST(23),XATST(17),UTST(7),XETST(19)
C
C      REAL AT(23,23),AZ(23,12),OT(23,7),PT(12,23),OT(12,12),FT(12,7),  ARRAYS/
1      FQV(17,17),BPA(23,17),WNO(23,4),WLA(12,17),WLD(12,4)          TESTMX2
C      INTEGER N1(23),NA1(17),NU1(7),NO1(4),LL1(12),NL1(35),NAME(8)
C
C      DEFINITIONS FOR FDBCK ROUTINE (ASSUMING MAXIMUM VALUES OF
C      NW = 10, NUIN = 8, NY = 10)
C
C      DIMENSIONS REDUCED TO ENVISAGE 1ST ORDER DELAY APPROXIMATIONS ONLY
C
C      REAL BN(27,8),BSN(27,15),AF(37,37),BF(37,15),RTAFR(37),RTAFI(37),
1      CS(10,27),R(10,10),S(10,10),DD(10,15),T(15,10),V(15,10),
2      EF(15,15)
C      INTEGER NX21(45),NU21(15)
C
C      WORKING MATRICES USE ARRAYS FROM PREVIOUS SUBROUTINES
C
C      DATA INRAS,INRAF/3H AS,3H AF/
C      N = 23
C      NA = 17
C      NU = 7
C      NF = 19
C      NY = 3
C      NO = 4
C      IPPNT1 = 0
C      IPPNT2 = 1
C      IPPNT3 = 0
C      IPPNT4 = 0
C      IPPNT5 = 1

```

```
CALL READ (N,NA,NU,NE,X,XA,U,XE,NY,Y,T1A,T1D,T2D,T1E,NAME,8)
```

```

C
C 22.4.70
C
C DIMENSIONS FOR DELAY1
C
C   L1 = L2 = L3 = L4 = 1
C
C   CALL DELAY1 (L1,L2,L3,L4,P1,P2,P3,P4,H1,H2,H3,H4,Q1,Q2,Q3,Q4,
1   G1,G2,G3,G4,T1A,T1D,T2D,T1E,1)
C   L = L1 + L2 + L3 + L4
C   NL = N + L
C   CALL QINPUT (N,NA,NU,NE,X,XA,U,XE,NY,Y,T1A,T1D,T2D,T1E,NAME,8)
C   CALL AMX (N,NA,NU,NE,X,XA,U,XE,A,IPRNT1)
C   CALL BMX (N,NA,NU,NE,X,XA,U,XE,B,IPRNT1)
C   CALL CMX (N,NA,NU,NE,X,XA,U,XE,C,IPRNT1)
C   CALL DMX (N,NA,NU,NE,X,XA,U,XE,D,IPRNT1)
C   CALL EMX (N,NA,NU,NE,X,XA,U,XE,E,IPRNT1)
C   CALL FMX (N,NA,NU,NE,X,XA,U,XE,F,IPRNT1)
C
C   CALL PMX (N,NA,NU,L,L1,L2,L3,L4,P,P1,P2,P3,P4,IPRNT1)
C   CALL PAMX (N,NA,NU,L,L1,L2,L3,L4,PA,P1,P2,P3,P4,IPRNT1)
C   CALL QMX (N,NA,NU,L,L1,L2,L3,L4,Q,Q1,Q2,Q3,Q4,QV,QOV,IPRN
1T1)
C   CALL GHX (N,NA,NU,L,L1,L2,L3,L4,G,G1,G2,G3,G4,IPRNT1)
C   CALL GAMX (N,NA,NU,L,L1,L2,L3,L4,GA,G1,G2,G3,G4,IPRNT1)
C   CALL HMX (N,NA,NU,L,L1,L2,L3,L4,H,H1,H2,H3,H4,IPRNT1)
C
C   CALL A1MX (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,A,QD,A2,B2,A1,IPRN
1T1)
C   CALL B1MX (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,B,B1,IPRNT1)
C   CALL C1MX (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,C,C1,IPRNT1)
C   CALL D1MX (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,D,ED,D2,E2,Q1,IPRN
1T1)
C   CALL E1MX (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,E,ED,E1,E1V,EEV,
1   IPRNT1)
C
C COMPUTE MATRICES FOR CALCULATING XA(T) FROM X1(T) AND U(T). PRINT AND PUNCH
C FROM (8)
C
C   XA(T) = - E1V*D1*X1(T) - E1V*F*U(T)
C           = XAX1*X1(T) + XAU*U(T) (14)
C
C   CALL XAX1U (NA,NL,NU,E1V,Q1,F,WNL1,WNL2,WNL3,7,1)
C
C   CALL ASMX (NL,NA,NU,P1,E1V,A1,D1,AS,WAL,IPRNT2)
C   CALL BSMX (NL,NA,NU,B1,E1V,C1,F,BS,WAU,IPRNT2)
C
C   CALL XEIGEN (NL,AS,WNL1,WNL2,RTASR,PTASI)
C   CALL OPJHCH (NL,NU,AS,BS,RTASR,PTASI)
C   CALL GRAHLM (N,NA,NU,NU,L,NL,P,PA,Q,G,GA,H,A,B,BD,C,D,E,ED,F
1   ,AS,BS,N1,NA1,NU1,NU1,L1,1,NL1,
2   NX2,NU2,AF,BF,NX21,NU21,1)
C
C 30.4.70
C
C BEFORE TESTING LINEARIZATION REPLACE THE INPUT VARIABLES, AS SOME ARE MODIFIED
C IN TESTMX AND USED AGAIN LATER
C
C   CALL EQUVR2 (X,XTST,N)
C   CALL EQUVR2 (YA,XATST,NA)
C   CALL EQUVR2 (U,UTST,NU)

```

CALL EQUVR2 (XE,XETST,NE)

```

C      CALL      TESTMX (N,NA,NU,NE,XTST,XATST,UTST,XETST,
1      A,B,BD,C,D,E,ED,F,VN,VNA,ERRF,ERRG,ERRFRC,ERRGRC,
2      NO,L,G,GA,H,P,PA,QV,HQV,GT,GAT,WNN,WNA,WAN,WAA)
      CALL      TESTMX2 (N,NA,NU,A,B,BD,C,D,E,ED,F,NO,L,G,GA,H,P,PA,
1      WNN,WNA,WAN,WAA,
2      NL,AS,BS,Q,AT,AZ,OT,PT,QT,FT,EDV,BRA,
3      WNL1,WNL2,WNL3,WLA,WLO)

```

C THE FOLLOWING PARAMETERS FOR THE F O B C K ROUTINE ARE INSERTED HERE
C AND WILL BE CHANGED AS THE NEED ARISES

```

C      28.4.70      (NUN = 3,      NW = 3,      NY = 6)
C      NUN = 3
C      NW = 3
C      NY = 6

```

C NOTE THAT NY HAS NOT THE SAME VALUE AS ABOVE, WHERE IT WAS USED
C FOR CHECKOUT OF THE STEADY STATE (SSNL2) ONLY

```

C      NU2 = NU + NUN
C      NX2 = NL + NW
C      CALL FORCK1 (N,NL,NU,NUN,NU2,NW,NY,NX2,AS,BS,X,3N,BSH,CS,R,S,DD,
1      T,V,AF,BF,WNL1,WNL2,B1,WAL,A1,E1,IPRNT3,IPRNT4,
2      IPRNT5,EE,WNL3,F1V)
      CALL ORANLM (N,NA,NU,NO,L,NL,P,PA,Q,G,GA,H,A,B,BD,C,D,E,ED,F
1      ,AS,BS,W1,NA1,NU1,NQ1,LL1,NL1,
2      NX2,NU2,AF,BF,NX21,NU21,2)
      CALL CAUTION

```

```

C      CALL      NEWXU (X,U,N,NU,XA,XE,NA,NE,NX2)
C
C      CALL PCH2013 (BF,NX2,NU2,RTAFR,RTAFI,1)
C      CALL PCH2013 (AF,NX2,NX2,RTAFP,RTAFI,1)
C      CALL PCH2013 (CS,NY,NL,RTAFR,RTAFI,1)
C      STOP
C      END

```

```

C      SUBROUTINE DELAY1 (L1,L2,L3,L4,P1,P2,P3,P4,H1,H2,H3,H4,Q1,Q2,Q3,Q4,
1      G1,G2,G3,G4,T1A,T1D,T2D,T1E,TDLY)

```

```

C      22.4.70
C      IN THIS ROUTINE THE ORDERS OF APPROXIMATION OF THE 4 DELAYS ARE SET AT WILL,
C      AS WELL AS THE VALUES FOR EQ (5,6).
C      3 IS THE MAXIMUM ORDER DIMENSIONED IN MAIN ROUTINE.  BEWARE.
C      SEE EXPLANATIONS IN MAIN ROUTINE
C      REAL P1(L1),P2(L2),P3(L3),P4(L4),H1(L1),H2(L2),H3(L3),H4(L4),
1      Q1(L1,L1),Q2(L2,L2),Q3(L3,L3),Q4(L4,L4)
C      GO TO (100,200),IDLY
C      FIRST ORDER APPROXIMATION
C      THIS MODEL HAS A RESPONSE (TO A STEP INPUT) OF THE SHAPE OF A NEGATIVE-UNITY-
C      AMPLITUDE-DROP, FOLLOWED BY AN EXPONENTIAL RISE TOWARDS THE INPUT AMPLITUDE,
C      WITH A TIME CONSTANT EQUAL TO HALF THE DELAY TIME.

```

```

C
C
C      G = -1
C      H = 4/TAU      (TAU = TIME DELAY)
C      P = 1
C      Q = -2/TAU
C
100  L1 = L2 = L3 = L4 = 1
      G1 = G2 = G3 = G4 = -1.
      P1(1) = P2(1) = P3(1) = P4(1) = 1.
      Q1(1,1) = - 2./T1A
      Q2(1,1) = - 2./T1D
      Q3(1,1) = - 2./T2D
      Q4(1,1) = - 2./T1E
      H1(1) = - 2.*Q1(1,1)
      H2(1) = - 2.*Q2(1,1)
      H3(1) = - 2.*Q3(1,1)
      H4(1) = - 2.*Q4(1,1)
      RETURN
C
200  WRITE (6,201)
201  FORMAT (//////31X,*THIS MODEL HAS A FIRST ORDER APPROXIMATION FOR
1    ALL TIME DELAYS,*/
7      31X,*WITH TIME CONSTANTS EQUAL TO HALF THE VALUES OF
2    EACH DELAY*//31X,*SPECIFICALLY, IN EQUATIONS (5,6)*//63X,*PI = 1*/
363X,*OI = -2/TAU*/63X,*GI = -1*/63X,*HI = 4/TAU      WHERE TAU I
4    S THE TIME DELAY IN EACH CASE*/)
      RETURN
      END
C
C
C      SURROUTINE FORCK1( N,NL,NU,MUN,NU2,NW,NY,NX2,AS,BS,X,BN,BSN,CS,R,
1      S,DD,T,V,AF,DF,VGS,AVG,BT,SCS,DVC,DOT,IPRNT3,
2      IPRNT4,IPRNT5,EE,BSE,ODE)
C
C      28.1.70, 15.9.70
C
C      IN THIS ROUTINE ADDITIONAL EQUATIONS ARE ATTACHED TO THE LINEAR MODEL,
C      SUCH THAT THERE IS FEEDBACK TO SOME OF THE INPUTS, THE REST BEING
C      CONSIDERED REFERENCE POINTS.
C      NEW STATE VARIABLES ARE INCORPORATED, CALLED W.
C      THE NEW EQUATIONS ARE KEPT IN THIS SEPARATE ROUTINE SO AS TO ENABLE
C      US TO CHANGE THEM AT WILL.
C
C      FROM MXEV2X1
C
C      X1DOT(T) = AS*X1(T) + BS*U(T)      X1 = [X Z]T      (9)
C
C      NEW EQUATIONS
C
C      Y(T) = CS*X1(T)      (14)
C
C      WDOT(T) = R*W(T) + S*Y(T) + DD*U2(T)      (15)
C
C      U2(T) = T*W(T) + V*Y(T) + EE*UB      (16)
C
C      WHERE
C
C      Y = VECTOR OF NY OUTPUTS
C      W = .. .. NEW STATE VARIABLES
C      UN = .. .. MUN NEWLY INTRODUCED INPUTS
C      U2 = .. .. MU2 INPUTS
C
C      U2 = [U UN]T

```

```

NU2 = NU + NUN

C
C UR = .. .. NU2 INPUT REFERENCE POINTS(OR ADDITIONAL CONTROL LAW FOR)U2
C EE = (NU2,NU2) MATRIX WHICH IS BASICALLY THE IDENTITY MATRIX,
C SUPPLEMENTED BY ELEMENTS (IF ANY) REPRESENTING CONNECTIONS
C BETWEEN INPUTS.
C
C NOW INCORPORATE NEW INPUTS INTO ORIGINAL SYSTEM (9)
C
C X1DOT = AS*X1 + BS*U + RN*UM
C          = AS*X1 + BSN*U2
C                                     BSN = [BS BN],   BN = [0]
C                                     (9A)
C
C REPLACING (14,16) INTO (9A) AND (15) WE OBTAIN
C
C X1DOT = [AS+BSN*V*CS]*X1 + BSN*T*W + BSN*EE*UB
C                                     (17)
C WDOT = [S*CS+DD*V*CS]*X1 + [P+DD*T]*W + DD*EE*UB
C                                     (18)
C I.E.
C *****
C *
C * X2DOT(T) = AF*X2(T) + BF*UB
C *
C *****
C                                     (13)
C
C WHERE
C
C X2 = [X1 W]T WITH (NX2 = NL + NW) COMPONENTS.
C
C      [[           ] [           ]]      [[           ]]
C      [ AS+BSN*V*CS  BSN*T ]      [ BSN*EE ]
C AF = [           ], BF = [           ]
C      [           ]      [           ]
C(NX2,NX2) [[           ]] (NX2,NU2) [[           ]]
C          [[           ]]      [[           ]]
C          [ S*CS+DD*V*CS R+DD*T ]      [ DD*EE ]
C          [[           ]]      [[           ]]
C
C REAL AS(NL,NL),BS(NL,NU),X(N),RN(NL,NUM),BSN(NL,NU2),CS(NY,NL),
1 R(NW,NW),S(NW,NY),DD(NW,NU2),T(NU2,NW),V(NU2,NY),EE(NU2,NU2),
2 AF(NX2,NX2),BF(NX2,NU2)
C
C WORKING MATRICES (DIFFERENT NAMES IN MAIN ROUTINE)
C
C REAL VCS(NU2,NL),DVC(NL,NL),DT(NL,NW),SCS(NW,NL),DVC(NW,NL),
1 DT(NW,NU),DSE(NL,NU2),DOE(NW,NU2)
C
C DATA FOR WRINX AND DRAWMX
C
C DATA IWRSN,IWRCS,IWRB,IWS,IWRDD,IWRT,IWRV,IWRAF,IWRRF/
1 3HASN,3H CS,3H R,3H S,3H DD,3H T,3H V,3H AF,3H BF/,
2 IWRBVC,IWRBT,IWRSOS,IWRVOC,IWRASB,IWRSD,IWRDT,IWRDZ/
3 3H BVC,3H BT,3H SOS,3H VOC,3H ASB,3H SD,3H DDT,3H RDZ/,
4 IWREE,IWRDSE,IWRDDE/3H EE,3H RSE,3H DE/
C
C INITIALIZE NEW MATRICES
C
C CALL NULL (BN,NL,NUM)
C CALL NULL (CS,NY,NL )
C CALL NULL ( R,NU,NW)
C CALL NULL ( S,NW,NY)

```

CALL NULL (TD,NW,NU2)

CALL NULL (T,NU2,NW)
CALL NULL (V,NU2,NY)
CALL UNIT (EE,NU2)

```

C
C
C ***** 28.1.70
C *      *
C * CASE 1 * APPLY CONTROLS EXISTING (OR SUGGESTED) IN SIMULATOR
C *      *
C *****
C
C ***** 20.4.70
C *      *
C * CASE 2 * APPLY FEEDWATER CONTROL (3) ONLY
C *      *
C *****
C
C ***** 17.8.70
C *      *
C * CASE 4 * APPLY ALL CONTROLS AS IN (1) BUT MOVE LAG ON WSDERROR BACK ON WSD
C *      *
C *****
C
C ***** 17.8.70
C *      *
C * CASE 5 * AS CASE (4) BUT WITHOUT MODERATOR CONTROL (2)
C *      * I.E. LEAVE EQ. (14,15) FOR Y, WDOT, BUT REMOVE EQ. (16) FOR U2
C ***** ( CTEMP IS PUNCHED ON THE CARDS CORRESPONDING TO U2)
C
C ***** 9.9.70
C *      *
C * CASE 6 * AS CASE (5) BUT WITHOUT LOAD FOLLOWING, I.E. FEEDWATER CONTR. ONLY
C *      *
C *****
C
C ***** 10.9.70
C *      *
C * CASE 7 * AS CASE (6) BUT REMOVING ALL EQUATIONS (14,15,16), NOT AS
C *      * IN CASES (5,6) WHERE ONLY THE FEEDBACK WAS REMOVED.
C ***** SUMMARIZING, THIS CASE IS AS CASE (2), BUT WITH SOME MODIFICATIONS
C          IN THE MODERATOR CONTROL WHICH IS NOT ACTIVE ANYWAY.
C          THIS IS THE MODEL COMPARED WITH THE WINFRITH TRANSIENTS (O.K.)
C
C ***** 28.4.70
C *      *
C * CASE 3 * APPLY LOAD FOLLOWING(1) AND FEEDWATER CONTROL (3) ONLY
C *      *
C ***** NOTE..
C          A) CONTROL (2) IS NOT REMOVED FROM THIS PROGRAM
C             ALTOGETHER, BUT COMMENT CARDS CTEMP PUNCHED INSTEAD
C             OF LEGITIMATE STATEMENTS FOR W AND U2 (THE CORRESPONDING
C             OUTPUTS Y, I.E. ELEMENTS OF MATRIX OS, REMAIN AS VALID
C             STATEMENTS.
C          B) SEE OUTPUT Y(2) (MODERATOR TEMPERATURE) AFTER
C             CONTROLS (1) TO (3)
C ***** PAY ATTENTION TO THE SPECIAL COMMENT ON THE COST FUNCTION
C          IN THAT CHAPTER.
C
C          EQ. NUMB.
C (1) LOAD FOLLOWING          (SIMUL.)
C
C      Y(4) = X(18)
C      WDOT(3) = TF1*Y(4) - TF1*U2(3)
C      U2(7) = W(3)
C
C          176. AND
C          OTHERS

```

```

C
C
C WHERE (NEW PARAMETERS)
C
C Y(4) = PMECH      MW(F)  MECH POWER PRODUCED BY TURBINE (X18)
C W(3) = DF         C/S    FREQUENCY ERROR
C U2(8) = PE        MW(E)   ELEC POWER DELIVERED TO GRID
C
C AND (OLD PARAMETERS)
C
C X(18) = PMECH
C U2(7) = DF
C
C AND (COEFFICIENTS)
C
C TF1 = TRANSF FUNCTION GAIN(INTEGR) FROM ACC TORQUE TO DF
C      = 0.0515
C
C DATA
C
C TF1      = .0515
C GS(4,18) = 1.
C S(3,4)   = TF1
C DD(3,8)  = - TF1
C T(7,3)   = 1.
C
C U2(7) IS AN INPUT WITH FEEDBACK
C
C (2) MODERATOR CONTROL          (CHANGED ON 17.8.70 ACCORDING TO WINFRITH, 15.4)
C
C Y(1) = (1.838 - 0.9729*X(22)PAR)*X(1) - 0.9729*X(1)PAR*X(22)      18
C Y(5) = 0.3862*X(23)PAR*X(1) + 0.3862*X(10)PAR*X(23)              83
C WDOT(4) = -(1./TAUF)*W(4) + (1./TAUF)*Y(1) - (.0029/(2.*TAUF))*Y(5) 125,131
C U2(5) = TKF*W(4) - TKF*Y(1) - (TKF/4.)*U2(7) + (TKF/2.)*U2(9)    129,130
C
C WHERE (NEW PARAMETERS)
C
C Y(5) = WSDRUM      LBS/SEC MASS FLOW RATE OF STREAM FROM DRUM (XE7)
C W(4) = IPC         POWER CONTROLLER VARIABLE
C U2(9) = WSDDEM     NORMAL. DEMANDED WSDRUM (NORMALIZED TO 344.91 LBS/SEC)
C                      (1./344.91 = .0029)
C
C AND (OLD PARAMETERS)
C
C Y(1) = PHION      NORM.   ION CHAMBER INDICATION OF FLUX (NORMAL.)
C U2(5) = YHDEM     NORM.   DEMANDED YM (MODERATOR VALVE POSITION)
C X(1) = PHIDASH    NORM.   NEUTRON FLUX
C X(10) = PD        PSTA    STEAM DRUM PRESSURE
C X(22) = HDASH     NORM.   MODERATOR HEIGHT
C X(23) = YTCV      NORM.   TURBINE CONTROL VALVE POSITION
C
C X( )PAR = X( ) IN STEADY STATE
C
C AND (COEFFICIENTS)
C
C TKF = TRANSF FUNCTION GAIN FOR MODERATOR VALVE
C      = 30
C TAUF = MODERATOR VALVE LAG (POWER CONTROLLER)
C       = 40 SEC
C
C DATA
C
C TKF      = 30.
C TAUF     = 40.
C
C GS(1,1) = 1.838 - .9729*X(22)

```


CS(1,22) = - .4729*X(1)

CS(5,10) = .3862*X(23)

CS(5,23) = .3862*X(10)

C
CTEMP P(4,4) = - 1./TAUF
C
CTEMP S(4,1) = 1./TAUF
CTEMP S(4,5) = - .0029/(2.*TAUF)

C
C
CTEMP T(5,4) = TKF
CTEMP V(5,1) = - TKF
C
CTEMP EE(5,7) = - TKF/4.
CTEMP EE(5,9) = TKF/2.

C
G*** U2(5) IS AN INPUT WITH FEEDBACK (NOT IN RUN (3))*****

C
C (3) FEEDWATER CONTROL

C
C Y(3)ERR = Y(3) - U2(10) 112,113
C Y(3) = 0.1258E-2*X(12) 112,113

C
C Y(6) = X(16)
C WDOT(2) = - (TK14/TAU11)*W(1) - (TK14/TAU11)*Y(3)
C + (.0729*TK15/TAU11)*Y(5) - (.0029/TAU11)*Y(6) 100,101,
C + (TK14/TAU11)*U2(10) 103,104
C WDOT(1) = (1./TAU10)*Y(3) - (1./TAU10)*U2(10) 106

C
C U2(4) = TK13*W(2) - TK13*TK14*W(1) - TK13*TK14*Y(3)
C + .0029*TK13*TK15*Y(5) - .0029*TK13*Y(6) 99,103,
C + TK13*TK14*U2(10) 104

C
C WHERE (NEW PARAMETERS)

C
C Y(3) = L INCHES STEAM DRUM LEVEL (IN PREVIOUS DEFINITIONS IN
C XEVA2 AND SSNLF, Y(3) WAS THE ERROR FROM
C NORMAL WORKING LEVEL)
C Y(6) = WFW LBS/SEC MASS FLOW RATE OF WATER INTO DRUM (X16)
C W(2) = IF2 FEEDWATER CONTROLLER VARIABLE
C W(1) = IF1 INCHES
C U2(10) = L1DEM INCHES STEAM DRUM DEMANDED LEVEL FOR YR CONTROLLER
C (NORMAL WORKING LEVEL IS 44 INCHES)

C
C AND (OLD PARAMETERS)

C
C U2(4) = YRDEM NORMAL. DEMANDED YR (FEEDWATER VALVE POSITION)
C X(12) = MW LB MASS OF WATER IN STEAM DRUM
C X(16) = WFW LBS/SEC MASS FLOW RATE OF WATER INTO DRUM

C
C AND (COEFFICIENTS)

C
C TK13 = TRANSFER FUNCTION GAIN FOR YRDEM
C = 5.
C TK14 = FIRST TRANSF FUNCTION GAIN FOR YR CONTROLLER
C*** = 0.025 (CRITICAL)
C TK15 = SET RATIO IN YR CONTROLLER
C = 1.
C TAU10 = TIME CONSTANT FOR YR CONTROLLER (FIRST TRANSF FUNCTION)
C = 200 SEC
C TAU11 = TIME CONSTANT FOR YRDEM
C = 100 SEC

C
C . DATA

```

C
TK13      = 5.
TK14      = .025
TK15      = 1.

C
TAU10     = 200.
TAU11     = 100.

C
CS(3,12)  = .1268E-2
CS(6,16)  = 1.

C
R(2,1)    = - TK14/TAU11

C
S(2,3)    = - TK14/TAU11
S(2,5)    = .0029*TK15/TAU11
S(2,6)    = -.0029/TAU11
S(1,3)    = 1./TAU10

C
DD(1,10)  = - S(1,3)
DD(2,10)  = - S(2,3)

C
T(4,2)    = TK13
T(4,1)    = - TK13*TK14

C
V(4,3)    = - TK13*TK14
V(4,5)    = .0029*TK13*TK15
V(4,6)    = -.0029*TK13

C
EE(4,10)  = - V(4,3)

C
U2(4) IS AN INPUT WITH FEEDBACK

C (4) MODERATOR TEMPERATURE (NO CONTROL SUGGESTED HERE. INTRODUCED ONLY FOR
C      THE SAKE OF COMPLETENESS OF THE MATRIX CS)

C
C      Y(2) = 2*X(4) - U2(2) 23

C
C      WHERE (OLD) PARAMETERS)

C
C      Y(2) = TMOU      DEG C    MODERATOR OUTLET TEMPERATURE
C      X(4) = TMAV      DEG C    MEAN MODERATOR TEMPERATURE
C      U2(2) = TMIN      DEG C    MODERATOR INLET TEMPERATURE

C
C***** SPECIAL COMMENT *****
C
C      IT WILL BE SEEN IN SUBROUTINE SCOSM (STATE-COSTATE MATRIX) THAT WE DEFINE
C      A COST MATRIX QY FOR THE OUTPUTS Y RATHER THAN FOR THE STATE VARIABLES
C      THEREFORE THE EQUATION (23) FOR Y(2) IS INCONVENIENT SINCE IT WOULD
C      PRODUCE MIXED TERMS (IN X AND U) IN THE COST FUNCTION.
C      IN CONSEQUENCE WE SHALL KEEP U(2) CONSTANT UNLESS AND UNTIL WE CHANGE
C      THE STRUCTURE OF THE COST MATRIX. THEREFORE.....
C*****
C
C      DATA
C
C      CS(2,4) = 2.

C
C
C
C      SET AND PRINT MATRIX      RSN = [BS,PN1 (9A)

C
C      CALL IMBED (BS,RSN,NL,NU,NL,NU2,1,1)
C      CALL IMBED (TN,RSN,NL,NUN,NL,NU2,1,NU+1)
C      IF (IPRNT3) 111,222,111

. 111 CONTINUE PRINT
C      CALL WRIMX (RSN,NL,NU2,IWRBSN)

```

CALL DRAWHX(RSN,NL,NU2,IWRBSN,7)

C			
C	PRINT ALL NEWLY INTRODUCED MATRICES		(14,15,
C			16)
	CALL WPTMX (CS,NY,NL,IWPCS)		
	CALL DRAWHX(CS,NY,NL,IWPCS,6)		
	CALL WPTMX (R,NW,NW,IWPR)		
	CALL DRAWHX(R,NW,NW,IWPR,8)		
	CALL WPTMX (S,NW,NY,IWPS)		
	CALL DRAWHX(S,NW,NY,IWPS,9)		
	CALL WPTMX (DD,NW,NU2,IWPD)		
	CALL DRAWHX(DD,NW,NU2,IWPD,7)		
	CALL WPTMX (T,NU2,NW,IWRT)		
	CALL DRAWHX(T,NU2,NW,IWRT,8)		
	CALL WPTMX (V,NU2,NY,IWV)		
	CALL DRAWHX(V,NU2,NY,IWV,9)		
	CALL WPTMX (EE,NU2,NU2,IWREE)		
	CALL DRAWHX (EE,NU2,NU2,IWREE,1)		
	222 CONTINUE		NOPRINT
C			
C	CALCULATE PARTIAL MATRICES		(17,18)
C			
C	ASB = RSN*V*CS + AS		
C			
	CALL MULTI (V,CS,VCS,NU2,NY,NL)		
	CALL MULTI (RSN,VCS,BVC,NL,NU2,NL)		
	IF (IPRNT4) 301,302,301		
301	CONTINUE		PRINT
	CALL WPTMX (BVC,NL,NL,IWBVC)		
	CALL DRAWHX (BVC,NL,NL,IWBVC,6)		
302	CONTINUE		NO PRINT
	CALL ADD (AS,BVC,BVC,NL,NL)		
	IF (IPRNT4) 311,312,311		
311	CONTINUE		PRINT
	CALL WPTMX (BVC,NL,NL,IWRASB)		
	CALL DRAWHX (BVC,NL,NL,IWRASB,6)		
312	CONTINUE		NO PRINT
C			
C	BT = RSN*T		
C			
	CALL MULTI (RSN,T,BT,NL,NU2,NW)		
	IF (IPRNT4) 321,322,321		
321	CONTINUE		PRINT
	CALL WPTMX (BT,NL,NW,IWRT)		
	CALL DRAWHX (BT,NL,NW,IWRT,8)		
322	CONTINUE		NO PRINT
C			
C	SD = S*CS + DD*V*CS		
C			
	CALL MULTI (S,CS,SCS,NW,NY,NL)		
	IF (IPRNT4) 331,332,331		
331	CONTINUE		PRINT
	CALL WPTMX (SCS,NW,NL,IWRSOS)		
	CALL DRAWHX (SCS,NW,NL,IWRSOS,6)		
332	CONTINUE		NO PRINT
	CALL MULTI (DD,VCS,BVC,NW,NU2,NL)		
	IF (IPRNT4) 341,342,341		
341	CONTINUE		PRINT
	CALL WPTMX (BVC,NW,NL,IWRDVC)		
	CALL DRAWHX (BVC,NW,NL,IWRDVC,6)		
342	CONTINUE		NO PRINT
	CALL ADD (SCS,BVC,BVC,NW,NL)		
	IF (IPRNT4) 351,352,351		
351	CONTINUE		PRINT

```

CALL WRIMX (DVC,NW,NL,IWRSO)

CALL DRAWMX (DVC,NW,NL,IWRSO,6)
352 CONTINUE NO PRINT
C
C RD = R + DT*T
C
CALL MULTI (DD,I,DDT,NW,NU2,NW)
IF (IPRNT4) 361,362,361
361 CONTINUE PRINT
CALL WRIMX (DDT,NW,NW,IWRODT)
CALL DRAWMX (DDT,NW,NW,IWRODT,8)
362 CONTINUE NO PRINT
CALL ADD (R,DDT,DDT,NW,NW)
IF (IPRNT4) 371,372,371
371 CONTINUE PRINT
CALL WRIMX (DDT,NW,NW,IWROD)
CALL DRAWMX (DDT,NW,NW,IWROD,8)
372 CONTINUE NO PRINT
C
C BSE = RSN*EE
C DDE = DD*EE
C
CALL MULTI (RSN,EE,BSE,NL,NU2,NU2)
CALL MULTI (DD,EE,DDE,NW,NU2,NU2)
IF (IPRNT4) 381,382,381
381 CONTINUE
CALL WRIMX (BSE,NL,NU2,IWROSE)
CALL DRAWMX (BSE,NL,NU2,IWROSE,1)
CALL WRIMX (DDE,NW,NU2,IWRODE)
CALL DRAWMX (DDE,NW,NU2,IWRODE,1)
382 CONTINUE
C
C CALCULATE FINAL MATRICES AF AND PF (13A)
C
CALL IMBED (BVC,AF,NL,NL,NX2,NX2,1,1)
CALL IMBED (DT,AF,NL,NW,NX2,NX2,1,NL+1)
CALL IMBED (DVC,AF,NW,NL,NX2,NX2,NL+1,1)
CALL IMBED (DDT,AF,NW,NW,NX2,NX2,NL+1,NL+1)
C
CALL IMBED (BSE,PF,NL,NU2,NX2,NU2,1,1)
CALL IMBED (DDE,PF,NW,NU2,NX2,NU2,NL+1,1)
IF (IPRNT5) 333,444,333
333 CONTINUE PRINT
CALL WRIMX (AF,NX2,NX2,IWRAF)
CALL DRAWMX (AF,NX2,NX2,IWRAF,10)
CALL WRIMX (PF,NX2,NU2,IWRPF)
CALL DRAWMX (PF,NX2,NU2,IWRPF,7)
444 CONTINUE NOPPRINT
RETURN
END
C
C
C
SUBROUTINE TESTMX (N,NA,NU,NE,X,XA,U,XE,DE,DG,DX,DXA,DU,FRAR,GBAR,
1 A,B,DD,C,D,E,ED,F,VN,VNA,ERRF,ERRG,ERRFPC,ERRGPC,
2 ND,L,G,GA,H,P,PA,QV,HCV,GT,GAT,WNN,WNA,WAN,WAA)
C
C TESTS MATRICES CALCULATED IN MXEVA BY GIVING VALUES (SMALL BUT NOT ZERO)
C TO X,XA,U AND CALCULATING XDOT=F AND G
C X,XA,U ARE CALLED DX,DXA,DU TO DISTINGUISH THEM FROM THE STEADY STATE VALUES
C F,G .. .. DE,DG
C THE RESULTS OUGHT TO BE CLOSE TO THE VALUES FRAR AND GBAR OBTAINED FROM THE
C NONLINEAR EQUATIONS (CALFUN) FOR VALUES OF X,XA,U EQUAL TO X+DX,XA+DXA, U+DU

```

C

```

      REAL X(N),XA(NA),U(NU),XE(NE),DX(N),DXA(NA),DU(NU),DE(N),DS(NA),
1      ERAP(N),G3AP(NA),ERRF(N),ERRG(NA),ERRFPC(N),ERRGPC(NA),
2      A(N,N),B(N,NA),BT(N,NO),C(N,NU),
3      D(NA,N),E(NA,NA),ED(NA,NO),F(NA,NU),
4      VN(N),VNA(NA),G(NO,N),GA(NO,NA),H(NO,L),P(L,N),PA(L,NA),
5      OV(L,L),HOV(NO,L),GT(NO,N),GAT(NO,NA),
6      WNN(N,N),WNA(N,NA),WAN(NA,N),WAA(NA,NA)
      REAL S(11)

```

C

C GENERATE DX,DXA,DU

C

```

      READ (5,1111) (S(I),I=1,11)
1111 FORMAT (7(E6.1,4X))
      READ (5,1112) IL,IP
1112 FORMAT (A3,A5)
      DO 1 I=1,N,4
1    DX(I) = X(I)*S(1)
      DO 1001 I=2,N,4
1001 DX(I) = X(I)*S(2)
      DO 2001 I=3,N,4
2001 DX(I) = X(I)*S(3)
      DO 3001 I=4,N,4
3001 DX(I) = X(I)*S(4)
      DO 2 I=1,NA,4
2    DXA(I) = XA(I)*S(5)
      DO 1002 I=2,NA,4
1002 DXA(I) = XA(I)*S(6)
      DO 2002 I=3,NA,4
2002 DXA(I) = XA(I)*S(7)
      DO 3002 I=4,NA,4
3002 DXA(I) = XA(I)*S(8)
      DO 3 I=1,NU,3
3    DU(I) = U(I)*S(9)
      DO 1003 I=2,NU,3
1003 DU(I) = U(I)*S(10)
      DO 2003 I=3,NU,3
2003 DU(I) = U(I)*S(11)

```

C

C DF = A*DX + B*DXA + D*DXD + C*DU

(1)

C = [A+BT*(G-H*OV*P)]*DX + [D+BT*(GA-H*OV*PA)]*DXA + C*DU

(1A)

C

C NOTE.. (1A) HAS BEEN OBTAINED BY MAKING ZOOT=0 IN (3) (STEADY STATE)
 C AND FINDING XD FROM (3,4), REPLACING INTO (1)

C

```

      CALL MULTI (H,OV,HOV,NO,L,L)
      CALL MULTI (HOV,P,GT,NO,L,N)
      CALL SURTR (G,GT,GT,NO,N)
      CALL MULTI (BT,GT,WNN,N,NO,N)
      CALL ADD (A,WNN,WNN,N,N)
      CALL MULMXV (WNN,DX,DE,N,N)
      CALL MULTI (HOV,PA,GAT,NO,L,NA)
      CALL SURTR (GA,GAT,GAT,NO,NA)
      CALL MULTI (BT,GAT,WNA,N,NO,NA)
      CALL ADD (C,WNA,WNA,N,NA)
      CALL MULMXV (WNA,DXA,VN,N,NA)
      CALL ADDV (DE,VN,DE,N)
      CALL MULMXV (C,DU,VN,N,NU)
      CALL ADDV (DE,VN,DE,N)

```

C

C DG = D*DX + E*DXA + ED*DXD + F*DU

(2)

C = [D+ED*(G-H*OV*P)]*DX + [E+ED*(GA-H*OV*PA)]*DXA + F*DU

(2A)

C

```

      CALL MULTI (ED,GT,WAN,NA,NO,N)

```

```

CALL ADD      (D,WAN,WAN,NA,N)

CALL MULMXV   (WAN,DX,DG,NA,N)
CALL MULTI    (EO,GAT,WAA,NA,NO,NA)
CALL ADD      (E,WAA,WAA,NA,NA)
CALL MULMXV   (WAA,DXA,VNA,NA,NA)
CALL ADDV     (DG,VNA,DG,NA)
CALL MULMXV   (F,DU,VNA,NA,NU)
CALL ADDV     (DG,VNA,DG,NA)

C
C GENERATE X+DX, XA+DXA, U+DU FOR CALFUN
C
DO 4 I=1,N
4 X(I) = X(I) + DX(I)
DO 5 I=1,NA
5 XA(I) = XA(I)+DXA(I)
DO 6 I=1,NU
6 U(I) = U(I)+DU(I)

C
C CALCULATE VALUES OF FBAR AND GBAR (EQUIVALENT TO F AND G IN CALFUN)
C
CALL CALFUN(N,NA,NU,NE,X,XA,U,XE,FBAR,GBAR,X13D,X20D,XA4D,1,1,1,1)

C
C CALCULATE ERRORS (DF-FBAR) AND (DG-GBAR), AS WELL AS THE PERCENTUAL ERRORS
C (RELATED TO FBAR AND GBAR)
C
DO 7 I=1,N
7 ERRE(I) = DF(I) - FBAR(I)
DO 8 I=1,NA
8 ERRG(I) = DG(I) - GBAR(I)
DO 20 I=1,N
IF(FBAR(I))100,101,100
101 ERREPC(I) = 0.
GO TO 20
100 ERREPC(I) = 100.*ERRE(I)/FBAR(I)
20 CONTINUE
DO 21 I=1,NA
IF(GBAR(I))200,201,200
201 ERRGPC(I) = 0.
GO TO 21
200 ERRGPC(I) = 100.*ERRG(I)/GBAR(I)
21 CONTINUE

C
C WRITE TABLES OF DF,DG SIDE BY SIDE WITH FBAR AND GBAR FOR COMPARISON
C
WRITE (6,10) IL,IP
10 FORMAT (1H1,31X,*TEST OF CORRECTNESS OF NXEVA*///31X,*FBAR AND GBAR
11 ARE THE VALUES OBTAINED FROM THE NONLINEAR SET (CALFUN)*//41X,
2*WITH X = X+DX, XA = XA+DXA, U = U+DU*/46X,*FBAR = F(X+DX,XA+D
3A,U+DU) - F(X,XA,U)*//46X,*GBAR = G(X+DX,XA+DXA,U+DU) - G(X,XA,U)*//
4/31X,
5DF AND DG ARE OBTAINED FROM THE LINEAR SET*//41X,
633HDF = A*DX + B*DXA + CD*DXD + E*DU,33X,*(1)*//44X,66H= (A+CD*(G-H
7*OV*P))*DX + (D+ED*(GA-H*OV*PA))*DXA + E*DU (1A)//41X,
833HDG = D*DX + E*DXA + ED*DXD + F*DU,33X,*(2)*//44X,66H= (D+ED*(G-H
9*OV*P))*DX + (E+ED*(GA-H*OV*PA))*DXA + F*DU (2A)//
A1X,*VALUES OF DX,DXA,DU ARE SCATTERED OVER THE RANGE *,A3,* TO *,
B A5,* OF STEADY STATE VALUES.*//
C1X,*(1A,2A) HAVE BEEN OBTAINED BY MAKING ZDOT=0 IN (3), (S/STATE),
D AND FINDING XD FROM (3,4).*,22X,*PERCENTUAL ERROR*/
E1X,*(SEE MAIN PROGRAM)*,93X,*(I. = UNDEFINED)*//
WRITE (6,11) ((I,DF(I),I,FBAR(I),I,ERRE(I),ERREPC(I)),I=1,N)
11 FORMAT (10X,*DF(*,I2,*) =*,E15.8,4X,*FBAR(*,I2,*) =*,E15.8,10X,
1 *DF(*,I2,*)-FBAR(*,I2,*) =*,E15.8,9X,E10.3)
WRITE (6,12) ((I,DG(I),I,GBAR(I),I,ERRG(I),ERRGPC(I)),I=1,NA)

```

12 FORMAT (10X,*DG(*,I2,*) =*,E15.3,4X,*GBAR(*,I2,*) =*,E15.3,10X,

1 *DG(*,I2,*)-GBAR(*,I2,*) =*,E15.8,9X,E10.3)

RETURN
END

C
C
C

SUBROUTINE TESTHX2 (N,NA,NU,A,B,BD,C,D,E,ED,F,ND,L,G,GA,H,P,PA,
1 KNN,WNA,WAN,WAA,
2 NL,AS,BS,Q,AT,AZ,CT,PT,QT,FT,EDV,BBA,
3 WNL1,WNL2,WND,NLA,WLD)
REAL A(N,N),B(N,NA),BD(N,ND),C(N,NU),D(NA,N),E(NA,NA),ED(NA,ND),
1 F(NA,NU),G(ND,N),GA(ND,NA),P(ND,L),P(L,N),PA(L,NA),
2 WNN(N,N),WNA(N,NA),WAN(NA,N),WAA(NA,NA)
REAL AS(NL,NL),BS(NL,NU),Q(L,L),WNL1(NL,NL),WNL2(NL,NL),
1 AT(N,N),AZ(N,L),CT(N,NU),PT(L,N),QT(L,L),FT(L,NU),
2 EDV(NA,NA),BBA(N,NA),WND(N,ND),WLA(L,NA),WLD(L,ND)

C
C

TEST PART II

C

TEST CORRECTNESS OF FINAL MODEL

C

X1DOT = AS*X1 + BS*U (9)

C

AGAINST BASIC LINEAR MODEL

C

XDOT = A*X + B*XA + BD*XD + C*U (1)

C

D = D*X + E*XA + ED*XD + F*U (2)

C

ZDOT = P*X + PA*XA + Q*Z (3)

C

XD = G*X + GA*XA + H*Z (4)

C

XDOT = A*X + B*XA + BD*(G*X+GA*XA+H*Z) + C*U

C

D = D*X + E*XA + ED*(G*X+GA*XA+H*Z) + F*U

C

D = (D+ED*G)*X + (E+ED*GA)*XA + F*U + ED*H*Z

C

XA = -EDV*((D+ED*G)*X + (E+ED*GA)*XA + F*U) EDV = INV(E+ED*GA)

C

XDOT = A*X - BD*EDV*((D+ED*G)*X+ED*H*Z+F*U) + BD*G*X + BD*H*Z + C*U

C

- BD*GA*EDV*((D+ED*G)*X+ED*H*Z+F*U)

C

ZDOT = P*X - PA*EDV*((D+ED*G)*X+ED*H*Z+F*U) + QZ

C

XDOT = AT*X + AZ*Z + QT*U (4)

C

ZDOT = PT*X + QT*Z + FT*U (5)

C

WHERE

C

AT = A + BD*G - (B+BD*GA)*EDV*(D+ED*G)

C

AZ = BD*H - (B+BD*GA)*EDV*ED*H

C

QT = C - (B+BD*GA)*EDV*F

C

PT = P - PA*EDV*(D+ED*G)

C

QT = Q - PA*EDV*ED*H

C

FT = - PA*EDV*F

C

CALCULATE EDV = INV(E+ED*GA)

C

CALL MULTI (ED,GA,WAA,NA,ND,NA)

CALL ADD (E,WAA,WAA,NA,NA)

CALL INVERT (WAA,EDV,NA,IRANK)

C

CALCULATE BBA = (B+BD*GA)*EDV

C

CALL MULTI (BD,GA,WNA,N,ND,NA)

CALL ADD (B,WNA,WNA,N,NA)

CALL MULTI (WNA,EDV,3BA,N,NA,NA)

C

C CALCULATE WAN = (D+ED*G)

C

CALL MULTI (ED,G,WAN,NA,ND,N)

CALL ADD (D,WAN,WAN,NA,N)

C

C CALCULATE AT = A + BD*G - BBA*WAN

C

CALL MULTI (BD,G,AT,N,ND,N)

CALL MULTI (BBA,WAN,WNN,H,NA,N)

CALL SUBTR (AT,WNN,WNN,N,N)

CALL ADD (A,WNN,AT,N,N)

C

C CALCULATE A7 = (BD - BBA*ED)H

C

CALL MULTI (BBA,ED,WND,N,NA,ND)

CALL SUBTR (BD,WND,WND,N,ND)

CALL MULTI (WND,H,A7,N,ND,L)

C

C CALCULATE CT = C - BA*F

C

CALL MULTI (BBA,F,CT,N,NA,NU)

CALL SUBTR (C,CT,CT,N,NU)

C

C CALCULATE PT = P - PA*EDV*(D+ED*G)

C

CALL MULTI (ED,G,WAN,NA,ND,N)

CALL ADD (D,WAN,WAN,NA,N)

CALL MULTI (PA,EDV,WLA,L,NA,NA)

CALL MULTI (WLA,WAN,PT,L,NA,N)

CALL SUBTR (P,PT,PT,L,N)

C

C CALCULATE QT = Q - PA*EDV*ED*H

C

CALL MULTI (PA,EDV,WLA,L,NA,NA)

CALL MULTI (WLA,ED,WLD,L,NA,ND)

CALL MULTI (WLD,H,QT,L,ND,L)

CALL SUBTR (Q,QT,QT,L,L)

C

C CALCULATE FT = PA*EDV*F

CALL MULTI (PA,EDV,WLA,L,NA,NA)

CALL MULTI (WLA,F,FT,L,NA,NU)

CALL NEGATE (FT,FT,L,NU)

C

C COMPARE AS AND BS FROM EQ (9) WITH THE MATRICES OF EQ (A),(B)

C

WRITE (6,500)

500 FORMAT (1H1,31X,*TEST OF CORRECTNESS OF FINAL MODEL**

131X,50H XDOT = AS*X1 + BS*U (9) //

231X,50H AGAINST THE BASIC LINEAR MODEL //

331X,50H XDOT = A*X + D*XA + DD*XD + C*U (1) /

431X,50H Q = D*X + E*XA + ED*XD + F*U (2) //

531X,50H ZDOT = P*X + PA*XA + Q*Z (3) /

631X,50H XZ = G*X + GA*XA + H*Z (4) ///

731X,50H (1,2,3,4) COMBINE TO GIVE //

831X,50H XDOT = AT*X + AZ*Z + CT*U (A) /

931X,50H ZDOT = PT*X + QT*Z + FT*U (B) //

A31X,50H WHERE //

B31X,50H AT = A + BD*G - (D+BD*GA)*EDV*(D+ED+G) /

C31X,50H AZ = BD*H - (D+BD*GA)*EDV* ED*H /

D31X,50H CT = C - (B+BD*GA)*EDV*F /

E31X,50H PT = P - PA*EDV*(D+ED*G) /

F31X,50H QT = Q - PA*EDV*ED*H /

G31X,504 FT = - PA*EDV*F

////

H31X,50H THE ELEMENTS OF AS AND BS ARE COMPARED TO THE /
I31X,52H CORRESPONDING ELEMENTS IN THE SUBMATRICES OF (AF,BF) ///
J31X,50H ELEMENTS WITH ERROR GREATER THAN 1.E-15 ARE LISTED ///

WRITE (6,511)

501 FORMAT (/12X,*ELEMENT OF MATRIX AS*,10X,*ELEMENT OF MATRIX AT*,
113X,*ERROR*//)

CALL COMPARE (AT,AS,N,N,NL,NL,1,1)

WRITE (6,502)

502 FORMAT (/12X,*ELEMENT OF MATRIX AS*,10X,*ELEMENT OF MATRIX AZ*,
113X,*ERROR*//)

CALL COMPARE (AZ,AS,N,L,NL,NL,1,N+1)

WRITE (6,503)

503 FORMAT (/12X,*ELEMENT OF MATRIX AS*,10X,*ELEMENT OF MATRIX PT*,
113X,*ERROR*//)

CALL COMPARE (PT,AS,L,N,NL,NL,N+1,1)

WRITE (6,504)

504 FORMAT (/12X,*ELEMENT OF MATRIX AS*,10X,*ELEMENT OF MATRIX QT*,
113X,*ERROR*//)

CALL COMPARE (QT,AS,L,L,NL,NL,N+1,N+1)

WRITE (6,505)

505 FORMAT (/12X,*ELEMENT OF MATRIX BS*,10X,*ELEMENT OF MATRIX CT*,
113X,*ERROR*//)

CALL COMPARE (CT,BS,N,NU,NL,NU,1,1)

WRITE (6,506)

506 FORMAT (/12X,*ELEMENT OF MATRIX BS*,10X,*ELEMENT OF MATRIX FT*,
113X,*ERROR*//)

CALL COMPARE (FT,BS,L,NU,NL,NU,N+1,1)

RETURN

END

C
C
C

SUBROUTINE COMPARE (A,B,M,N,M1,N1,K1,K2)

C
C
C

COMPARES MATRICES WITH SUBMATRICES IN PARTITIONED MATRIX

REAL A(M,N),B(M1,N1)

DO 1 I=1,M

II = I+K1-1

DO 1 J=1,N

JJ = J+K2-1

ERROR = B(II,JJ) - A(I,J)

IF (ERROR.LT.1.E-15) GO TO 1

WRITE (6,2) II,JJ,3(II,JJ),I,J,A(I,J),ERROR

1 CONTINUE

RETURN

2 FORMAT (10X,*(*,I2,*,*,I2,*)*,3X,E15.8,5X,*(*,I2,*,*,I2,*)*,3X,
1 E15.8,5X,E15.8)

END

C
C
C

SUBROUTINE DRAWMX(A,M,N,IWRA,LOLLY)

C
C
C
C
C
C

DRAWS MATRIX, X FOR NONZERO ELEMENTS
. FOR ZERO ..

ZERO DEFINED AS LE.1.E-10

REAL A(M,N)

INTEGER DRAW (31,31),IFRMT(31)

DATA IZ,N7/1H.,14X/

DATA (IFRMT(I),I=1,31)/

```

1      10H(31X, 1A1), 10H(31X, 2A1), 10H(31X, 3A1), 10H(31X, 4A1),

2      10H(31X, 5A1), 10H(31X, 6A1), 10H(31X, 7A1), 10H(31X, 8A1),
3      10H(31X, 9A1), 10H(31X, 10A1), 10H(31X, 11A1), 10H(31X, 12A1),
4      10H(31X, 13A1), 10H(31X, 14A1), 10H(31X, 15A1), 10H(31X, 16A1),
5      10H(31X, 17A1), 10H(31X, 18A1), 10H(31X, 19A1), 10H(31X, 20A1),
6      10H(31X, 21A1), 10H(31X, 22A1), 10H(31X, 23A1), 10H(31X, 24A1),
7      10H(31X, 25A1), 10H(31X, 26A1), 10H(31X, 27A1), 10H(31X, 28A1),
8      10H(31X, 29A1), 10H(31X, 30A1), 10H(31X, 31A1)/
WRITE(6,1) IMPA,M,N
1 FORMAT (//31X,*MATRIX *,A3,* (*,I2,* *,I2,*) IN DRAWING*//
1      31X,*1234567890123456789012345678901*//)
ZERO = 1.E-10
DO 2 I=1,M
DO 2 J=1,N
DRAW (I,J) = IZ
IF (ABS(A(I,J)) - ZERO) 2,2,3
3 DRAW (I,J) = NZ
2 CONTINUE
LFRMT = IFRMT(N)
WRITE (6,LFRMT) ((DRAW(I,J),J=1,N),I=1,M)
WRITE (6,4)
4 FORMAT (//31X,*ELEMENTS OF VALUE GREATER THAN 1.E-10 DRAWN AS X*//)
RETURN
END

```

C
C
C

```

SURROUTINE DPAHLM (N,NA,NU,N0,L,NL,P,PA,Q,G,GA,H,A,B,DD,C,D,E,ED,F
1      ,AS,BS,N1,NA1,NU1,N01,LL1,NL1,
2      ,NX2,NU2,AF,BF,NX21,NU21,ICOM)
REAL A(N,N),B(N,NA),C(N,NU),D(NA,NA),E(NA,NU),ED(NA,N0),
1      F(NA,NU),AS(NL,NL),BS(NL,NU),P(L,N),PA(L,NA),Q(L,L),G(N0,N),
2      GA(N0,NA),H(N0,L)
INTEGER N1(N),NA1(NA),NU1(NU),N01(N0),LL1(L),NL1(NL)
REAL AF(NX2,NX2),BF(NX2,NU2)
INTEGER NX21(NX2),NU21(NU2)
DATA IZ,NZ/14.,14X/
GO TO (100,200,300),ICOM

```

C
C
C

DRAWS COMPLETE LINEAR MODEL

```

100 WRITE (6,1)
1 FORMAT (1H1,31X,*COMPLETE LINEAR MODEL, BASIC EQUATIONS (XD(T) IS
1THE VECTOR OF APPROX. DELAYED VARIABLES)*///21X,
270HXDOT(T) = A*X(T) + B*XA(T) + DD*XD(T) + C*U(T)
3      (1)///
4345678901234567*,20X, 5X,*12345678901234567890123*,20X,*12
4345678901234567*,20X, *1234*, 20X,*1234567*//)
DO 2 I=1,N
CALL SCAN (A ,N,N ,N1 ,I7,N7,I)
CALL SCAN (B ,N,NA,NA1,I7,N7,I)
CALL SCAN (DD,N,N,N01,I7,N7,I)
CALL SCAN (C ,N,NU,NU1,I7,N7,I)
WRITE (6,11)I,(N1(J),J=1,N),(NA1(J),J=1,NA),(N01(J),J=1,N0),
1      (NU1(J),J=1,NU)
11 FORMAT (1X,I2,2X,23A1,20X,17A1,20X,4A1,20X,7A1)
2 CONTINUE
WRITE (6,12)
12 FORMAT (//27X,64H0 = D*X(T) + E*XA(T) + ED*XD(T) +F*U(T)
1      (2)///
23*,20X,*12345678901234567*, 5X,*1234567890123456789012
20X,*1234*,20X,*1234567*//)
DO 13 I=1,NA
CALL SCAN (D ,NA,N ,N1 ,I7,N7,I)
CALL SCAN (E ,NA,NA,NA1,I7,N7,I)
CALL SCAN (ED,NA,N0,N01,I7,N7,I)

```

```
CALL SCAN (F ,NA,NU,NU1,IZ,N7,I)
```

```
WRITE (6,11) I, (N1(J), J=1, N), (NA1(J), J=1, NA), (NU1(J), J=1, NU),  
1 (NU1(J), J=1, NU)  
13 CONTINUE
```

C

```
WRITE (6,40)  
40 FORMAT (1H1,31X,*TIME DELAY APPROXIMATIONS*////21X,36H2DOT(T) = P  
1*X(T) + PA*XA(T) + Q*Z(T),31X,*(7)*///21X,*THE LENGTH OF ? IS L*//  
2/5X,*12345678901234567890123*,20X,*12345678901234567*,20X,*1234(TH  
3E ROW LENGTH OF Q IS L)*//  
DO 41 I=1,L  
CALL SCAN (P ,L,N ,N1 ,IZ,N7,I)  
CALL SCAN (PA,L,NA,NA1,I7,N7,I)  
CALL SCAN (Q ,L,L ,LL1,I7,N7,I)  
WRITE (6,42) I, (N1(J), J=1, N), (NA1(J), J=1, NA), (LL1(J), J=1, L)  
42 FORMAT (1X,I2,2X,27A1,20X,17A1,20X,4A1)  
41 CONTINUE
```

```
WRITE (6,50)  
50 FORMAT (////23X,34HXDOT(T) = G*X(T) + GA*XA(T) + H*Z(T),31X,*(4)*//  
1/5X,*12345678901234567890123*,20X,*12345678901234567*,20X,*1234(TH  
2E ROW LENGTH OF H IS L)*//  
DO 51 I=1,L  
CALL SCAN (G ,L,N ,N1 ,I7,N7,I)  
CALL SCAN (GA,L,NA,NA1,I7,N7,I)  
CALL SCAN (H ,L,L ,LL1,I7,N7,I)  
WRITE (6,42) I, (N1(J), J=1, N), (NA1(J), J=1, NA), (LL1(J), J=1, L)  
51 CONTINUE
```

C

C DRAWS EQUIVALENT LINEAR MODEL

C

```
300 WRITE (6,20)  
20 FORMAT (1H1,31X,*EQUIVALENT LINEAR MODEL*////20X,29HX1DOT(T) = AS  
1*X1(T) + BS*U(T),39X,*(9)*///20X,*X1 IS THE COMPOSITE VECTOR OF X  
2AND ? , ITS LENGTH IS NL=N+L*///5X,*123456789012345678901234567(R  
3OW LENGTH NL)*,30X,*1234567*//)
```

C

C

```
21.4.70
```

C

C

```
CHECK IF NL AND NU ARE EQUAL TO 27 AND 7 RESPECTIVELY
```

C

```
IF (NL - 27) 34,33,34
```

```
33 IF (NU - 7) 34,36,34
```

```
34 WRITE (6,35)
```

```
35 FORMAT (////10X,*NL IS NOT EQUAL TO 27 AND/OR NU NOT EQUAL TO 7*//  
1///10X,*CHANGE DRAWLH AND PERHAPS DELAY*)
```

```
STOP
```

C

```
36 DO 21 I=1,NL
```

```
CALL SCAN (AS,NL,NL,NL1,I7,N7,I)
```

```
CALL SCAN (BS,NL,NU,NU1,I7,N7,I)
```

```
WRITE (6,22) I, (NL1(J), J=1, NL), (NU1(J), J=1, NU)
```

```
22 FORMAT (1X,I2,2X,27A1,45X,7A1)
```

```
21 CONTINUE
```

```
RETURN
```

C

C DRAWS COMPREHENSIVE LINEAR MODEL

C

```
200 WRITE (6,60)
```

```
60 FORMAT (1H1,31X,*COMPREHENSIVE LIN.MODEL*////20X,28HX2DOT(T) = AF  
1*X2(T) + BC*U2,39X,*(13)*///20X,*X2 IS THE COMPOSITE VECTOR OF X1  
2 AND W, ITS LENGTH IS NX2=NL+NU*///5X,*12345678901234567890123456  
378901(ROW LENGTH NX2)*,25X,*1234567890(ROW LENGTH NU2)*//)
```

C

C

```
CHECK IF NX2 AND NU2 ARE EQUAL TO 30 AND 10 RESPECTIVELY
```

```
*** (CASE 3) ***
```

```

C CHECK IF NX2 AND NU2 ARE EQUAL TO 31 AND 10 RESPECTIVELY ***CASES 4,5,6**
C CHECK IF NX2 AND NU2 ARE EQUAL TO 29 AND 10 RESPECTIVELY ***CASES 7 ***
C IF (NX2 - 30) 4,3,4 CASE 4,5
3 IF (NU2 - 10) 4,6,4
4 WRITE (6,5)
5 FORMAT (1H1,10X,*NX2 IS NOT EQUAL TO 30 AND/OR NU2 NOT EQUAL TO 10 CASE 3
1*////10X,*CHANGE OPACLM AND PERHAPS FDBCK1*)
STOP
C
6 DO 61 I=1,NX2
CALL SCAN (AF,NX2,NX2,NX21,IZ,NZ,I)
CALL SCAN (BF,NX2,NU2,NU21,IZ,NZ,I)
WRITE (6,62) I, (NX21(J),J=1,NX2), (NU21(J),J=1,NU2)
62 FORMAT (1X, I2, 2X, 30A1, 42X, 10A1) CASE 3
61 CONTINUE
RETURN
END
C
C
C SUBROUTINE SCAN (A,M,N,NN,IZ,NZ,I)
REAL A(M,N)
INTEGER NN(N)
ZERO = 1.E-10
DO 1 J=1,N
NN(J) = I7
IF (ABS(A(I,J)) - ZERO) 1,1,2
2 NN(J) = NZ
1 CONTINUE
RETURN
END
C
C
C FUNCTION SLOPE1 (Z)
C DERIVATIVE OF XE(1) (DKM=REACTIVITY EFFECT OF MODERATOR HEIGHT)
C W.R.T. X(22) (HDASH=MOD. HEIGHT FROM CORE BOTTOM NORMAL. TO 12.5 FT)
C NOTE
C THE VALUES FOR DATA WERE CALCULATED IN SLOPES FROM DATA OF FUN
C (TAKEN FROM STEAD6) BY SIMPLE INTERPOLATION.
C SEE SLOPES (UMAC03JA,3.9.69) FOR DETAILS
C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE.
C
REAL X(14),Y(14)
DATA (X(I),I=1,14)/
1 0., .2800E-01, .8800E-01, .1400E+00, .1840E+00,
2 .2320E+00, .2800E+00, .3280E+00, .3760E+00, .4240E+00,
3 .4720E+00, .5200E+00, .5720E+00, .6000E+00/,
4 (Y(I),I=1,14)/
5 .1919E+00, .1786E+00, .1500E+00, .1262E+00, .1042E+00,
6 .8437E-01, .6771E-01, .5625E-01, .5104E-01, .4533E-01,
7 .3542E-01, .3229E-01, .2857E-01, .2657E-01/
L = 14
SLOPE1 = 1.
CALL INTPOL (L,X,Y,SLOPE1,Z)
RETURN
END
C
C
C

```

FUNCTION SLOPE2 (7)

```

C
C DERIVATIVE OF XE(4) (HWSTAR=ENTHALPY OF WATER AT SATURATION)
C W.P.T. X(10) (PD=STEAM DRUM PRESSURE)
C
C NOTE
C THE VALUES FOR DATA WERE CALCULATED IN SLOPES FROM DATA OF FUN
C (TAKEN FROM STEAD6) BY SIMPLE INTERPOLATION.
C SEE SLOPES (UMACQJA, 8.9.69) FOR DETAILS
C
C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE
C
REAL X(14),Y(14)
DATA (X(I),I=1,14)/
1 0. , .5000E+02, .1250E+03, .1750E+03, .2250E+03,
2 .2750E+03, .3500E+03, .4500E+03, .5500E+03, .6500E+03,
3 .7500E+03, .8500E+03, .9500E+03, .1000E+04/,
4 (Y(I),I=1,14)/
5 .1182E+01, .9660E+00, .6420E+00, .4980E+00, .4119E+00,
6 .3561E+00, .3019E+00, .2539E+00, .2220E+00, .1991E+00,
7 .1820E+00, .1690E+00, .1580E+00, .1524E+00/
L = 14
SLOPE2 = 1.
CALL INTPOL (L,X,Y,SLOPE2,7)
RETURN
END

```

FUNCTION SLOPE3 (7)

```

C
C DERIVATIVE OF XE(5) (DHW/DP=FUN3) W.P.T. X(10) (PD=STEAM DRUM PRESSURE)
C
C NOTE
C THE VALUES FOR DATA WERE CALCULATED IN SLOPES FROM DATA OF FUN
C (TAKEN FROM STEAD6) BY SIMPLE INTERPOLATION.
C SEE SLOPES (UMACQJA, 8.9.69) FOR DETAILS
C
C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE
C
REAL X(13),Y(13)
DATA (X(I),I=1,13)/
1 0. , .4000E+02, .9000E+02, .1250E+03, .1750E+03,
2 .2250E+03, .2750E+03, .3250E+03, .3750E+03, .4250E+03,
3 .4750E+03, .7500E+03, .1000E+04/,
4 (Y(I),I=1,13)/
5 -.1061E-01, -.8362E-02, -.5550E-02, -.3400E-02, -.2000E-02,
6 -.1332E-02, -.1068E-02, -.8200E-03, -.5800E-03, -.4000E-03,
7 -.2400E-03, -.1540E-03, -.7582E-04/
L = 13
SLOPE3 = 1.
CALL INTPOL (L,X,Y,SLOPE3,7)
RETURN
END

```

FUNCTION SLOPE4 (7)

```

C
C DERIVATIVE OF FUN4(KXCOUT) W.P.T. KXCOUT
C
C THE VALUES FOR DATA WERE CALCULATED IN SLOPES FROM DATA OF FUN
C (TAKEN FROM STEAD6) BY SIMPLE INTERPOLATION.
C SEE SLOPES (UMACQJA, 8.9.69) FOR DETAILS

```

C

C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE

C

```

REAL X(14),Y(14)
DATA (X(I),I=1,14)/
1 0., .5000E-01, .1500E+00, .2625E+00, .3975E+00,
2 .5475E+00, .7188E+00, .9187E+00, .1150E+01, .1442E+01,
3 .1792E+01, .2212E+01, .2725E+01, .3900E+01/,
4 (Y(I),I=1,14)/
5 .4937E+00, .4697E+00, .4195E+00, .3640E+00, .3169E+00,
6 .2771E+00, .2392E+00, .2049E+00, .1746E+00, .1458E+00,
7 .1200E+00, .9911E-01, .7842E-01, .6847E-01/
L = 14
SLOPE4 = 1.
CALL INTPOL (L,X,Y,SLOPE4,7)
RETURN
END

```

C

C

C

FUNCTION SLOPE5 (7)

C

C DERIVATIVE OF XE(3) (FUNG = FUNCTION OF YR)

C

C W.R.T. X(19) (YR=100°C FEEDWATER VALVE POSITION)

C

C NOTE

C

C THE VALUES OF DATA ARE TAKEN FROM SLOPE5 IN STEAD6

C

C

C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE

C

```

REAL X(9),Y(9)
DATA (X(I),I=1,9)/.0 ,.050,.100,.150,.200,.250,.300,.350,1. /,
1 (Y(I),I=1,9)/.130,.235,.313,.426,.578,.770,.960,1.232,1.232/
L = 9
SLOPE5 = 1.
CALL INTPOL (L,X,Y,SLOPE5,7)
RETURN
END

```

C

C

C

FUNCTION SLOPE6 (7)

C

C DERIVATIVE OF XE(11) (FUNCTYDV = F6,DISCHARGE VALVE CHARACTERISTICS)

C

C W.R.T. U(3) (YDV = DISCHARGE VALVE POSTION FROM FULLY OPEN)

C

C NOTE

C

C THE VALUES FOR DATA WERE CALCULATED IN SLOPES FROM DATA OF FUN

C

C (TAKEN FROM STEAD6) BY SIMPLE INTERPOLATION.

C

C SEE SLOPES (UMACJJA,8.9.63) FOR DETAILS

C

C

C POLYNOMIAL FITTING MIGHT BE CONSIDERED AT A LATER STAGE

C

```

REAL X(13),Y(13)
DATA (X(I),I=1,13)/
1 0., .5665E-01, .1917E+00, .2916E+00, .3750E+00,
2 .4583E+00, .5417E+00, .6250E+00, .7083E+00, .7917E+00,
3 .8750E+00, .9583E+00, .1000E+01/,
4 (Y(I),I=1,13)/
5 .2528E+00, .3399E+00, .5030E+00, .5918E+00, .7434E+00,
6 .8571E+00, .9280E+00, .9916E+00, .1104E+01, .1297E+01,
7 .1730E+01, .2269E+01, .2538E+01/
L = 13
SLOPE6 = 1.

```

```
CALL INTPOL (L,X,Y,SLOPE,Z)
```

```
RETURN  
END
```

```
SUBROUTINE INVERT (B,A,M,IRANK)
```

```
INVERTS REAL MATRICES
```

```
REAL A(M,M),B(M,M)  
DIMENSION ISWCH(100),JSWCH(100)  
IRANK = M  
DO 10 I1=1,M  
DO 10 I2=1,M  
10 A(I1,I2)=B(I1,I2)
```

```
INITIALISE PIVOT SWITCHES
```

```
DO 1 I=1,M  
ISWCH(I)=0  
JSWCH(I)=0  
1 CONTINUE
```

```
MAIN LOOP IS EXECUTED M TIMES. FIND PIVOT AND INTERCHANGE CORRESPONDING X AND  
Y, IF NO PIVOT MATRIX IS SINGULAR
```

```
DO 2 I=1,M
```

```
SEARCH FOR PIVOT
```

```
AMAX=0.0  
DO 3 I1=1,M  
DO 4 I2=1,M  
IF (ISWCH(I1).NE.0.OR.JSWCH(I2).NE.0) GO TO 4  
IF (ABS(A(I1,I2)).LE.AMAX) GO TO 4  
IPIVOT=I1  
JPIVOT=I2  
AMAX=ABS(A(I1,I2))  
4 CONTINUE  
3 CONTINUE  
IF (AMAX.LE.0.000001) GO TO 999  
APIVOT=1.0/A(IPIVOT,JPIVOT)  
A(IPIVOT,JPIVOT)=APIVOT  
ISWCH(IPIVOT)=JPIVOT  
JSWCH(JPIVOT)=IPIVOT
```

```
PIVOT FOUND REDUCE ELEMENTS
```

```
REDUCE ELEMENTS OFF PIVOT ROW AND PIVOT COLUMN
```

```
DO 5 I1=1,M  
DO 6 I2=1,M  
IF (I1.EQ.IPIVOT.OR.I2.EQ.JPIVOT) GO TO 6  
A(I1,I2)=A(I1,I2)-A(IPIVOT,I2)*A(I1,JPIVOT)*APIVOT  
6 CONTINUE  
5 CONTINUE
```

```
REDUCE ELEMENTS ON PIVOT ROW AND COLUMN
```

```
DO 99 I1=1,M  
IF (I1.EQ.IPIVOT) GO TO 99  
A(I1,JPIVOT)=A(I1,JPIVOT)*APIVOT
```

```
66 IF(I1.EQ.JPIVOT) GO TO 99
```

```
    A(IPIVOT,I1)=-A(IPIVOT,I1)*APIVOT
```

```
99 CONTINUE
```

```
2 CONTINUE
```

```
C
C
C
```

```
    INTERCHANGE ROWS AND COLUMNS IF NECESSARY
```

```
    DO 77 I1=1,M
```

```
11 IF(JSWCH(I1).EQ.I1) GO TO 77
```

```
    K=ISWCH(I1)
```

```
    DO 88 J=1,M
```

```
        TEMP=A(I1,J)
```

```
        A(I1,J)=A(K,J)
```

```
        A(K,J)=TEMP
```

```
88 CONTINUE
```

```
    ITEMP=ISWCH(I1)
```

```
    ISWCH(I1)=ISWCH(K)
```

```
    ISWCH(K)=ITEMP
```

```
    GO TO 11
```

```
77 CONTINUE
```

```
    DO 55 I=1,M
```

```
12 IF (JSWCH(I).EQ.I) GO TO 55
```

```
    K=JSWCH(I)
```

```
    DO 44 J=1,M
```

```
        TEMP=A(J,I)
```

```
        A(J,I)=A(J,K)
```

```
        A(J,K)=TEMP
```

```
44 CONTINUE
```

```
    ITEMP=JSWCH(I)
```

```
    JSWCH(I)=JSWCH(K)
```

```
    JSWCH(K)=ITEMP
```

```
    GO TO 12
```

```
55 CONTINUE
```

```
    RETURN
```

```
C
C
C
```

```
    MATRIX SINGULAR
```

```
999 CALL SLITE(1)
```

```
    WRITE(6,200)
```

```
200 FORMAT(1H ,15HMATRIX SINGULAR)
```

```
    ITRANK=I-1
```

```
    RETURN
```

```
END
```


APPENDIX C.

Listing of the more important routines in REGULA

REGULA

SCSMX2

RICCAT2

CTROBS1

EIGVV1

HESSEN

QREIG

QRT

ASQR

EIGVEC

DEISP

WRCMX

DRRMX

CMPEI1

INCOMA

MATINI (in COMPASS language, usable as FORTRAN in CDC machines).

PROGRAM REGULA (INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT,
1 DATOU,TAPE3=DATOU,PERMOTA,TAPE2=PERMOTA)

1ST CARD (KEYS).. HI(I/P MEDIUM FOR PERMANENT DATA), MO(O/P MEDIUM
FOR DATA INTO SIMLIX), ISHORT(A,B,AOC NOT PRINTED)
DATA IS A, B, RTC, PROA, NCASE, COSTS. N O NAME CARDS.
OUTPUT DATA IS KR, KDR, PROAOC, COSTS. OCCASIONALLY RTC AND/OR PROA
KR, KDR, PROAOC, COSTS, WRITTEN ON TAPE 3 IF REQUIR.

19.1.72 (L. TEPPER, CONTROL SYSTEMS SECTION, DEPARTMENT OF
COMPUTING AND CONTROL, IMPERIAL COLLEGE)

OPTIMAL LINEAR REGULATOR PROBLEM.

SOLUTION OF THE STEADY STATE (ALGEBRAIC) MATRIX RICCATI EQUATION.
SUITABLE FOR EXTERNAL DISTURBANCE INPUT UD AND HENCE FOR CONTROL
INPUT SETTLING DOWN VALUE UGBAR DIFFERENT FROM 0.

THIS PROGRAM HAS EVOLVED FROM MXANL1, IN WHICH THE CHIEF PURPOSE WAS
COMPUTING EIGENVALUES AND EIGENVECTORS OF A REAL SQUARE MATRIX.
THE MAIN SUBROUTINE FOR MXANL1 WAS EIGVV1.
THE SUBROUTINES SSCSX AND RICCAT ARE CENTRAL TO THE SOLUTION OF
THE RICCATI EQUATION. CTRORS TESTS CONTROLLABILITY AND OBSERVABILITY

21.5.70

INTRODUCE ROUTINES RICCAT AND SSCSX (STATE-COSTATE MATRIX AM)
WHERE AM IS PARTITIONED INTO 4 DIMENSIONALLY EQUAL QUARTERS (N,N)

$$AM = \begin{bmatrix} \begin{bmatrix} 1 & 0 \\ A_{M11} & A_{M12} \end{bmatrix} & \begin{bmatrix} 0 \\ -B^*[R]^{-1}B^*[Q] \end{bmatrix} \\ \begin{bmatrix} 0 \\ A_{M21} & A_{M22} \end{bmatrix} & \begin{bmatrix} 0 \\ -A^T \end{bmatrix} \end{bmatrix} \quad (2N, 2N)$$

THE PURPOSE IS TO CALCULATE P(N,N) FROM THE MATRIX RICCATI EQUATION

$$\dot{P} = Q + [A]^T P + P A - P B [R]^{-1} B^T P$$

AND THE CONTROL LAW

$$U = -[R]^{-1} B^T P X \quad (\text{FOR SYSTEM WITH ZERO STEADY STATE DISTURBANCE INPUT}).$$

FOR THE SYSTEM

$$\begin{aligned} / & \\ / & \dot{X} = A X + B U \\ / & \\ / & Y = C X \end{aligned}$$

12.4.70

MATRIX ANALYSIS

CALCULATES EIGENVALUES AND EIGENVECTORS OF A SQUARE AND REAL INPUT MATRIX A
(OR SEVERAL) AND PERFORMS VARIOUS OTHER MANIPULATIONS AND OPERATIONS.

THE MAIN SUBROUTINE IS * EIGVV1 * WHICH IS A MODIFIED VERSION OF EIGSYS

(L. WEINER/J. VAN NESS, VOGELBACK COMPUTER CENTER, NORTHWESTERN UNIVERSITY,

C EVANSTON, ILLINOIS) WHICH IN TURN IS BASED ON WILKINSON'S INVERSE ITERATION

C METHOD FOR CALCULATING EIGENVECTORS AND THE USUAL QR ALGORITHM (FRANCIS)
C FOR EIGENVALUES. THE EIGENVALUES ARE CORRECTED BY THE RAYLEIGH QUOTIENT
C SCHEME.

C SUBROUTINES USED BY EIGVV1

C HESSEN, GREIG, QPT, ASQP, EIGVEC, DEISP (ALL SUBROUTINES OF EIGSYS)
C WRCMX PRINTS A COMPLEX MATRIX
C DPCMX DRAWS A COMPLEX MATRIX WITH DOTS AND CROSSES

C OTHER SUBROUTINES

C SCSMX2 STATE-COSTATE MATRIX CONSTRUCTION
C RICCAT1 SOLUTION OF THE RICCATI EQUATION.
C PCHKKD PRODUCES KR, KPR (REDUCED FEEDBACK MXS)
C CTORBS1 CHECKS CONTROLLABILITY AND OBSERVABILITY
C QPCMX1 DRAWS COMPLEX MX (MAX PRESENT DIM. IS 31)
C WPMX WRITES REAL MATRIX
C DRRMX DRAWS A REAL MATRIX WITH DOTS AND CROSSES
C WPIEIG WRITES EIGENVALUES
C PCM2013 PUNCHES COMPLEX MATRIX WITH FORMAT (4E20.13)
C PCM2013, PRM2013, RRM2013, PCV2013, RCV2013
C READS COMPLEX MATRIX, PUNCHES REAL MATRIX.
C READS REAL MATRIX, PUNCHES, READS COMPLEX VECTOR.
C CMPEI COMPARES EIGENVALUES OF TWO MATRICES
C COMPARES READ-IN PRODUCT OF EIGENVALUES
C WITH COMPUTED VALUE, STOPS PROGRAM IN
C CASE OF DISAGREEMENT.
C MULTI, ADD, NULL, NEGATE, TRANSP, UNIT
C OPERATIONS ON REAL MATRICES
C IMRFD TRANSFERS SUBMATRIX INTO PARTITIONED MATRIX
C CARVHD TRANSFERS SUBMATRIX FROM PARTITIONED MATRIX
C EQUHMC, EQUHVC EQUIVALENCE OF COMPLEX MATRICES, VECTORS
C EQUIPCX EQUIVALENCE OF A REAL AND A COMPLEX MATRIX
C TRANSPC TRANSPOSES COMPLEX MATRIX
C MULCOM MULTIPLIES COMPLEX MATRICES
C MLMXDC MULTIPLIES A SQUARE MATRIX BY A DIAGONALIZED
C MATRIX WITH 2 OPTIONS
C OPTION 1 $\rightarrow A \cdot D$
C OPTION 2 $\rightarrow D \cdot A$
C MULCR, MULRC MULTIPLIES COMPLEX BY REAL MATRIX, REAL BY COMPLEX.
C INVCOM INVERTS COMPLEX MATRICES
C INCCMA INVERTS COMPLEX MATRICES, MORE PRECISE THAN INVCOM
C MATIN1 INVERTS REAL MATRIX

C LIST OF DIMENSIONS

C N = NUMBER OF STATE VARIABLES
C NU = NUMBER OF INPUTS
C NCTR = NUMBER OF INPUTS TO BE CONTROLLED (SET IN SCSMX)
C NY = NUMBER OF OUTPUTS (CONSTRUCTED IN SCSMX)
C NYS = NUMBER OF OUTPUTS IN PROGRAM MHEV2X1
C NL = DIMENSION OF X1 IN PROGRAM MHEV2X1 (1ST NL COMPONENTS OF X)
C NCASFR = CASE-NUMBER (OF FDBCK1 IN PROGRAM MHEV2X1) OF INPUT MATRIX A

C LIST OF MOST IMPORTANT ARRAYS

C A = STATE MATRIX, REAL (OF IN MHEV2X1) (N,N)
C B = DRIVING MATRIX, REAL (OF IN MHEV2X1) (N,NU)
C C = OUTPUT MATRIX, REAL (CONSTRUCTED HERE, AS EXTENSION OF CS) (NY,N)
C CS = OUTPUT MATRIX, REAL (CS IN MHEV2X1) (NYS,NL)
C RD = DIAGONAL LIST OF COST MATRIX R (FOR U) (NU)

```

C   RQINV = INVERSE OF RQ                               (NU)

C   QYD = DIAGONAL LIST OF COST MATRIX QY (FOR Y) (NY)
C   AM = STATE-COSTATE MATRIX, REAL                     (2N,2N)
C   AMC = STATE-COSTATE MATRIX, COMPLEX                 (2N,2N) (NOT USED YET)
C           (ALSO USED AS WORKING MATRIX IN EIGVV1 AND DESTROYED WHEN IVC = NOT 0)
C   RTR = EIGENVALUES, REAL PART                         (2N)
C   RTI = EIGENVALUES, IMAG PART                         (2N)
C   RTC = EIGENVALUES, COMPLEX                           (2N)
C   EVC = EIGENVECTORS MATRIX, COMPLEX                   (2N,2N)
C           (CONTAINS R.H. EIGENVECTORS FOR OPTION IVC = 1,3)
C           (CONTAINS L.H. EIGENVECTORS FOR OPTION IVC = 2)
C           (CONTAINS RUBRISH FOR OPTION IVC = 0)
C   EVCIN = INVERSE OF EVC
C   PC = COMPLEX P MATRIX OF RICCATI EQUATION            (N,N)
C   P = REAL P MATRIX OF RICCATI EQUATION               (N,N)

C   SUBROUTINE EIGVV1 (IVC,IPR1,IPR2,IPR3,IPR4,N,A,B,C,Z,Y,RTCX,ECX,
C   1 X,XI,V,VI,W,IROW,S,H,DA,DX,DV,CC,DD,AD,XD,VD,
C   2 GQ,HQ)

C   CONTROL POINTS OF EIGVV1

C   IVC = 0 → NO EIGENVECTOR CALCULATION
C   IVC = 1 → CALCULATES REGULAR (RIGHT HAND) EIGENVECTORS
C   IVC = 2 → CALCULATES TRANSPOSED (LEFT HAND) EIGENVECTORS
C   IVC = 3 → CALCULATES BOTH

C   ****WARNING
C           FOR THE TIME BEING, OPTIONS 1 AND 2 ARE FORBIDDEN. FURTHER CHANGES IN
C           EIGVV1 ARE NECESSARY.

C   IPR1 = 0 → VALUES OF CURRENT EIGENVECTORS (IN EIGVEC ROUTINE) NOT PRINTED
C   IPR2 = 0 → VALUES OF CORRECTED EIGENVALUES, RESIDUAL VECTORS, ETC., NOT PRIN.
C   IPR3 = 0 → FINAL EIGENVECTOR MATRIX NOT PRINTED
C   IPR4 = 0 → INPUT MATRIX A NOT PRINTED

C   NOTE
C           FOR IPR3 = NOT 0, THE R.H. EIGENVECTOR MATRIX IS PRINTED FOR IVC = 1,3,
C           WHEREAS THE L.H. EIGENVECTOR MATRIX IS PRINTED FOR IVC = 2
C           (SEE TEMPORARY WARNING ABOVE)

C   I.E.
C   CALL
C           W = R.H. EIGENVECTOR MATRIX
C           IVIT = L.H. EIGENVECTOR MATRIX
C           DIA = DIAGONALIZED TRANSFORM OF A
C           INV = INVERSE OF EITHER MATRIX
C   THEN
C           A = W * DIA * INV(W)
C           A = INV(IVIT) * DIA * IVIT

C   ****THE MAXIMUM DIMENSION NOW IS N = 30 (2N = 60), NU = 10, NYS = 6 *****
C   FOR HIGHER ORDER IT IS NECESSARY TO INCREASE THE DIMENSIONS IN THIS MAIN
C   PROGRAM ONLY, NOT IN THE SUBROUTINES. THE EXCEPTIONS ARE HESSEN(HIGHEST
C   ORDER 200), DRRMX(99), DRGMX(31)

C   *** DIMENSION OF C IS 10 ONLY *****
C   REAL A(30,30),B(30,10),C(10,30),CS(6,27),PD(15),PQINV(15),QYD(30)
C   REAL AM(60,60),BM(60,60),CM(60,60),RTR(60),RTI(60)
C   REAL W(60,4),X(60),XI(60),V(60),VI(60),G(60),H(60)
C   INTEGER IROW(60,2),DRAW(60,60),ICTR(15),ICOL(15),KOMENT(14)

```

COMPLEX AMC(60,60),RTC(60),EVC(60,60),EVCINV(30,30),PC(30,30)

COMPLEX DA(60,60),DX(60),DV(60),DC(60),DD(60)
DOUBLE AD(60,60),XD(60),VD(60),GD(60),HD(60)

C
C
C DEFINITIONS FOR SUBROUTINE RIGCAT

COMPLEX AMC1(30,30),AMC2(30,30),AMC3(30,30),AMC4(30,30),RTCA(30)
REAL BM1(30,30),BM2(30,30),BM3(30,30),BM4(30,30),KD(15,30),
1 CM1(30,30),CM2(30,30),CM3(30,30),CM4(30,30)
INTEGER INX(60)
EQUIVALENCE (AMC(1,1),AMC1),(AMC(1,16),AMC2),(AMC(1,31),AMC3),
1 (AMC(1,46),AMC4)
EQUIVALENCE (BM(1,1),BM1),(BM(1,16),BM2),(BM(1,31),BM3),
1 (BM(1,46),BM4),(CM(1,1),CM1),(CM(1,16),CM2),
2 (CM(1,32),CM3),(CM(32,47),CM4),(AM,KD)
EQUIVALENCE (EVCINV,PC)
EQUIVALENCE (INX,IROW)

C
C EQUIVALENCE (AMC,DA,AD),(DX,XD),(DV,VD),(DC,GD),(DD,HD)
C INTEGER DATCASE(2)
C DATA IHPA,IWPCIN,IWRAE/10H A ,10HEIGVG*INV.,10HEV*RIA*INV/
C DATA ICASE,ICOM,IWO,IVO,ISIO/
C 15HCASE,10H COMPLEX,10HW(AOC) ,10HV(AOC) ,10HEIGVA(AOC)/,
C 2 IPR,IPEAL/10HP(RIGCATI),10H REAL /,IKR/10HK(REDUCED)/
C 3 ,IKOR/10HKO(REDUC.)/

C
C ISHORT = 1 → A,R,AOC NOT PRINTED
C MI = 5 → READ PERMANENT DATA (A,...,PPQA) FROM CARD
C MI = 2 → READ PERMANENT DATA (A,...,PPQA) FROM TAPE 2
C MO = 6 → WRITE O/P DATA ON PAPER
C MO = 3 → WRITE O/P DATA ON TAPE 3
C

READ 117, MI, MO, ISHORT
READ (MI, 118) DATCASE
PRINT 119, DATCASE
NCASEQ = 3
N = 30
NU = 10
NYS = 6
NY = 8
NL = 27
NN = 2*N
IVC = 3

FN9CK1 3

C
C***WARNING OPTIONS 1 AND 2 FOR IVC ARE FORBIDDEN FOR THE TIME BEING
C

C
C READ INPUT MATRICES A,R,CS, PRINT AND DRAW
C

C
C CALL IRM2013 (MI,A,N,N)
C CALL IRM2013 (MI,R,N,NU)
C CALL IRM2013 (CS,NYS,NL)
C IF (ISHORT.EQ.1) GO TO 3
C CALL WRIMX (A,N,N,3H A)
C CALL DRRMX (A,N,N,DRAW,30H A)
C CALL WRTMX (R,N,NU,3H R)
C CALL DRRMX (R,N,NU,DRAW,30H R)
C CALL WRIMX (CS,NYS,NL,3H CS)
C CALL DRRMX (CS,NYS,NL,DRAW,30H CS)
C

C
C READ COMPLEX EIGENVALUES OF A (IF AVAILABLE)
C

3 CALL ICV2013 (MI,RTC,N)
CALL WRIEIG (RTC,N,10H A)

```

C
C READ PRODUCT OF REAL PARTS OF EIGENVALUES OF A (IF AVAILABLE)
C
  READ (MI,109) PROA
  PRINT 110, PROA,MI,H0,ISHORT
C
C READ NCASE AND COSTS FOR Y(3), Y(7), U(5), U(6) (THE 2 OUTPUTS
C AND 2 INPUTS WEIGHTED IN MATRICES QY AND R IN ROUTINE SCSMX2)
C
  READ 115, NCASE,(KOMENT(I),I=1,6)
  READ 116, (KOMENT(I),I=7,14)
  READ 113, QYD3,QYD7,RD5,RD6
  PRINT 114, NCASE,KOMENT,QYD3,QYD7,RD5,RD6
C
  IF (EOF,5) 1,2
1 PRINT 100
  GO TO 9999
2 PRINT 106
C
C CONSTRUCT STATE-COSTATE MATRIX CALLING SCSMX2 (THE FIRST STEPS ARE
C THE CONSTRUCTION OF MATRICES G, Q, R). USE PARAMETER 2.
C
  CALL SCSMX2 (N,NU,NY,NN,NYS,NL,A,B,C,CS,AM,RD,RTINV,QYD,ICTR,NCTR,
1 NCASE,DRAW,NCASEB,2,QYD3,QYD7,RD5,RD6,MO)
  CALL DRRMX (AM,NN,NN,DRAW,30H AM, STATE-COSTATE MATRIX )
C
C CALCULATE EIGENVECTORS OF A AND CHECK CONTROLLABILITY AND OBSERVABILITY
C
C ICTORS = 0 * CONTROLLABILITY AND OBSERVABILITY IS NOT CHECKED
C EIGENVALUES READ IN FROM A PREVIOUS PROGRAM'S OUTPUT
  ICTORS = 1
  IF (ICTORS) 4,5,4
4 PRINT 105
  CALL EIGVV1 (IVC,0,0,0,0,N ,A ,BM,CM,RTR,RTI,RTC,EVC,X,XI,V,VI,W,
1 IPON,G,H,DA,DX,DV,DC,DD,AD,XD,VD,GD,HD)
  CALL INVCOM (EVC,EVCINV,N,IRANK)
  PRINT 102, IVP4,IRANK
C
C THE FOLLOWING STATEMENT ONLY IF PROA HAS BEEN READ IN.
C
  CALL CMPEIL (RTCA,RTC,N,PROA,1,NCASE,NCASEB,MO)
  CALL MULCOM (EVC,EVCINV,DA,N,N,N)
  CALL DRCHX1 (DA,N,N,INRCIN,DRAW,NN)
  CALL CTROBS1 (B,C,N,NU,NY,EVC,EVCINV,QYD,AMC,DA,RTC,ICTR,NCTR,
1 ICOL,DRAW,NN)
C
  CALL PCV2013 (RTC,N)
5 CALL EQUVC (RTC,RTCA,N)
C
C CALCULATE EIGENVALUES AND (IF IVC = NOT 0) EIGENVECTORS OF AM
C
  PRINT 108
  CALL EIGVV1 (IVC,0,0,0,0,NN,AM,BM,CM,RTR,PTI,RTC,EVC,X,XI,V,VI,W,
1 IPON,G,H,DA,DX,DV,DC,DD,AD,XD,VD,GD,HD)
C
C PICK UP THE EIGENVECTORS CORRESPONDING TO STABLE EIGENVALUES OF AM AND
C CONSTRUCT MATRICES W11, W21, WHERE W11 CONSISTS OF THE UPPER HALVES
C AND W21 OF THE LOWER HALVES OF THE STABLE EIGENVECTORS OF AM.
C
C CALCULATE RICCATI MATRIX P = W21*INV(W11)
C
C IPRW = 0 * W11,W21 NOT PRINTED
C IPRR = 0 * P(RICCATI) NOT PRINTED
C IPRS = 0 * SSP(STEADY STATE) NOT PRINTED
C IPAOC = 0 * AOC(OPT. CONTROLLED) NOT PUNCHED

```

```

C   IPQOC = 0 * QOC (0 + (K)*P*K)   NOT PUNCHED

C
C   IPRW = IPPR = IPRS = 0
C   IPAOC = 0
C   IPQOC = 0
C   CALL RICCAT2 (M,NN,AM,EVC,AMC1,AMC2,AMC3,RC ,RM1,RM2,A ,RM4,
1      CM1,CM2,CM3,RTC,DX,DRAW,3,RM3,CM4,ROINV,NU,IPRW,
2      IPRR,IPRS,IPAOC,NCASE,IPQOC,NCASF9,INX,DV,KD,ISHORT)

C   RM1 = P,          RM3 = RT,  CM2 = AXP1 = AOC,  CM3 = AXP2 = SS0,
C                                     (WHEN IPQOC.NE.0, THEN AXP2 = QOC),
C   CM4 = DK * K = (RIINV*(Q)*T)*P
C   KD = (RIINV*(Q)*T*(IN11)INVIT*(- RTCS)INV*(M21)T
C   WHERE  RTCS = STABLE EIGENVALUES OF AM
C   DX = RTCS
C   DV = (- RTCS)INV
C
C   THE CONTROL LAW IS  $U(T) = -K*X(T)$ 
C   K IS A MATRIX (NU,N) WHICH IS EMPTY EXCEPT FOR THE ROWS CORRESPONDING TO
C   THE CONTROLLED INPUTS. WE PUNCH OUT THE CONTROLLED ROWS ONLY (KR)
C
C   PUNCH 112, ICASE,NCASE,IKR,IREAL,NCASF9
C   WRITE (NO,112) ICASE,NCASE,IKR,IREAL,NCASF9
C   CALL PCHKKD (N1,N,NCIR,CM4,CM3,ICIR,OPAW,1,NO)
C   PRINT 111, ICASE,NCASE,IKR,IREAL,NCASF9
C
C   FOR SYSTEMS WITH NONZERO DISTURBANCE INPUT THE OPTIMAL CONTROL IS
C
C    $UC(T) = -KR*X(T) - KDR*BD*UD + [I - KDR*BC]*UCBAR$ 
C
C   WHERE
C
C   UC      = CONTROL INPUTS VECTOR (PART OF U)
C   UCBAR   = SETTLING-DOWN VALUE OF UC
C   UD      = DISTURBANCE INPUTS VECTOR (PART OF U)
C   BC      = PART OF MATRIX B CORRESPONDING TO UC
C   BD      = PART OF MATRIX B CORRESPONDING TO UD
C   KR      = PART OF MATRIX K CORRESPONDING TO UC
C   KDR     = PART OF MATRIX KD CORRESPOND. TO UC
C
C   SEE DETAILS OF CONSTRUCTION OF KD IN ROUTINE RICCAT1
C
C   PUNCH 112, ICASE,NCASE,IKDR,IREAL,NCASF9
C   WRITE (NO,112) ICASE,NCASE,IKDR,IREAL,NCASF9
C   CALL PCHKKD (NU,N,NCIR,KD ,CM3,ICIR,DRAW,2,NO)
C   PRINT 111, ICASE,NCASE,IKDR,IREAL,NCASF9
C
C   IPCHP = 0 * P NOT PUNCHED
C
C   IPCHP = 0
C   IF (IPCHP) 7,8,7
C 7 PUNCH 112, ICASE,NCASE,IPR,IREAL,NCASF9
C   CALL PRM2013 (RM1,N,N)
C   PRINT 111, ICASE,NCASE,IPR,IREAL,NCASF9
C
C   CALCULATE EIGENVALUES OF P TO CHECK POSITIVE DEFINITENESS
C
C 8 PRINT 103
C   CALL EIGVV1 (0,0,0,0,1,N,RM1,RM3,CM3,RTD,RTI,RTC,EVC,X,XI,V,VI,W,
1      IROW,G,H,TA,DX,DV,DC,DD,AD,XD,VD,GD,HO)
C
C   CALCULATE EIGENVALUES OF AOC (A OPTIMALLY CONTROLLED) AND COMPARE WITH
C   EIGENVALUES OF A

```

I

C

C

C

c

c

C

1

END

C

C

C

—

3

5

C

6.8.71

C STATE-COSTATE MATRIX AM, PARTITIONED INTO 4 EQUALLY DIMENSIONED QUARTERS

$$AM = \begin{bmatrix} \begin{bmatrix} & & \end{bmatrix} & \begin{bmatrix} & & \end{bmatrix} \\ \begin{bmatrix} AM11 & AM12 & \end{bmatrix} & \begin{bmatrix} A & -P*(R)INV*(Q)^T & \end{bmatrix} \\ \begin{bmatrix} & & \end{bmatrix} & \begin{bmatrix} & & \end{bmatrix} \\ (2N,2N) \begin{bmatrix} & & \end{bmatrix} & \begin{bmatrix} & & \end{bmatrix} \\ \begin{bmatrix} AM21 & AM22 & \end{bmatrix} & \begin{bmatrix} -Q & -[A]^T & \end{bmatrix} \\ \begin{bmatrix} & & \end{bmatrix} & \begin{bmatrix} & & \end{bmatrix} \end{bmatrix}$$

C WHERE THE STEADY STATE RICCATI EQUATION IS

$$0 = Q + [A]^T P + P A - P^* Q^* (R) INV^* (Q)^T P \quad (1)$$

C AND THE OBJECTIVE IS TO MINIMIZE THE COST INTEGRAND

$$[X]^T Q X + [U]^T R U \quad (2)$$

C FOR THE SYSTEM

$$/ \quad \dot{X} = A X + B U \quad (3)$$

$$/ \quad Y = C X \quad (4)$$

C BY DEFINING A COST MATRIX QY FOR Y RATHER THAN FOR X, WE OBTAIN,
C FROM (2) AND (4)

$$Q = [C]^T QY C \quad (5)$$

C REAL AM(N,N),A(N,N),P(N,N),R(N,N),RINV(N,N),C(NY,N),
1 CS(NYS,NL),QYQ(NY)
C INTEGER ICTR(NN),DRAW(N,N)

C GO TO (20,20,30), ISQS

C (1) CONSTRUCTION OF Q

C C(NY,N) IS AN EXTENSION OF CS(NYS,NL) CALCULATED IN FORCK1 IN MXEV2X1

C N CORRESPONDS TO NX2 (MATRIX A) DEFINED IN FORCK1

C NL NL (MATRIX AS)

C NYS NY (OUTPUTS)

C NY = ACTUAL NUMBER OF OUTPUTS (MATRIX C)

C Y(1) = X(1) = PHIDASH

C Y(2) = X(4) = THAV

C Y(3) = X(10) = P0

C Y(4) = X(13) = PMECH

C Y(5) = X(22) = HDASH

C Y(6) = X(23) = YTCV

C Y(7) = X(30) = DF

C Y(8) = X(21) = YM

C SET C... UNIT VALUE TO ALL CORRESPONDING ELEMENTS

C 20 CALL NULL (C,NY,N)

C C(1,1)=C(2,4)=C(3,10)=C(4,18)=C(5,22)=C(6,23)=C(7,30)=C(8,21) = 1.

C CALL W2IMX (C,NY,N,3H C)

C CALL DRRMX (C,NY,N,DRAW,30H C

C PUNCH 101, NY,N

C CALL PRM2013 (C,NY,N)

C SET QY AS A LIST OF ELEMENTS OF A DIAGONAL MATRIX

C QYQ(3), QYQ(7) AND NCASE HAVE BEEN READ IN MAIN PROGRAM.

C THEY ARE PRINTED IN THE OUTPUT.

C
C DO 1 I=1,NY
1 QYD(I) = 0.
QYD(3) = QYD3 E QYD(7) = QYD7
C
C Q = [C]T*QY*C
C
C USE FOLLOWING FORMULA NY
C $Q(I,J) = \sum_{K=1}^{NY} C(K,I)*C(K,J)*QYD(K)$
C
C AND RECALL THAT AM21 = - Q

C
C DO 2 I=1,N
C II = I+N
C DO 2 J=1,N
C JJ = J
C AM(II,JJ) = 0.
C DO 2 K=1,NY
2 AM(II,JJ) = AM(II,JJ) - C(K,I)*C(K,J)*QYD(K)

C (2) CONSTRUCTION OF AM12 = - R*[R]INV*[R]T

C SET R AND [R]INV AS A LIST OF ELEMENTS OF A DIAGONAL MATRIX

C WHERE

C U(1) = QKRAL
C U(2) = TMIN
C U(3) = YDV
C U(4) = YPDDEM (YPDDEM, THE ORIGINAL U(4) IS ACTIVE IN CONTROL
C AS DESCRIBED IN FDBCK1)
C U(5) = YMDM
C U(6) = EDYTCVDEM
C U(7) = DFDDEM (DF, THE ORIGINAL U(7) IS ACTIVE IN FDBCK1)
C U(8) = PE
C U(9) = WSDDEM
C U(10) = LIDEM

C
C DO 3 I=1,NU
C ICTR(I) = 0
3 RD(I) = RD5
RD(6) = RD6
C DO 4 I=1,NU
4 RDINV(I) = 1./RD(I)
RDINV(2) = RDINV(3) = RDINV(4) = RDINV(7) = RDINV(9) = 0.
RDINV(1) = RDINV(8) = RDINV(10) = 0.
ICTR(5) = ICTR(6) = 1
NCTR = 2

C
C NU
C AM12(I,J) = - SUM B(I,K)*R(J,K)*RDINV(K)
C K=1

C DO 5 I=1,N
C II = I
C DO 5 J=1,N
C JJ = J+N
C AM(II,JJ) = 0.
C DO 5 K=1,NU
5 AM(II,JJ) = AM(II,JJ) - B(I,K)*R(J,K)*RDINV(K)

C (3) IMPED A AND - [A]T

```
CALL IMBED (A,AM,N,N,NN,NN,1,1)
```

```
DO 6 I=1,N
```

```
II = I+N
```

```
DO 6 J=1,N
```

```
JJ = J+N
```

```
6 AM(II,JJ) = - A(J,I)
```

```
PRINT 100
```

```
RETURN
```

```
30 PRINT 102, NCASE, NCASEF, (QYD(I), I=1, NY), RD(5), RD(6)
```

```
PUNCH 102, NCASE, NCASEF, (QYD(I), I=1, NY), RD(5), RD(6)
```

```
WRITE (MO,102) NCASE, NCASEF, (QYD(I), I=1, NY), RD(5), RD(6)
```

```
RETURN
```

```
100 FORMAT (//////////31X, *AM (STATE-COSTATE MATRIX) FOLLOWS*////////)
```

```
131X,58H      [[      ] [      ] [      ] [      ] [      ] [      ] /
```

```
231X,58H      [ AM11  AM12 ] [      ] [      ] [      ] [      ] [      ] /
```

```
331X,58H      AM E  [[      ] [      ] [      ] [      ] [      ] [      ] /
```

```
431X,58H(2N,2N) [[      ] [      ] [      ] [      ] [      ] [      ] /
```

```
531X,58H      [ AM21  AM22 ] [      ] [      ] [      ] [      ] [      ] /
```

```
631X,58H      [[      ] [      ] [      ] [      ] [      ] [      ] /
```

```
7////////)
```

```
101 FORMAT(13H MATRIX C (,I2,1H,,I2, 1H))
```

```
102 FORMAT (4X,*CASE*,I6,* FOR CASE *,I2,*(100 PD) OF MXEV2X1*//
```

```
1 9X,*Y(1) = X( 1) E PHIDASH, QYD(1) =*,1PE8.1/
```

```
2 9X,*Y(2) = X( 4) E TMAV , QYD(2) =*,1PE8.1/
```

```
3 9X,*Y(3) = X(10) E PD , QYD(3) =*,1PE8.1/
```

```
4 9X,*Y(4) = X(18) E PMECH , QYD(4) =*,1PE8.1/
```

```
5 9X,*Y(5) = X(22) E HDASH , QYD(5) =*,1PE8.1/
```

```
6 9X,*Y(6) = X(23) E YTCV , QYD(6) =*,1PE8.1/
```

```
7 9X,*Y(7) = X(30) E DF , QYD(7) =*,1PE8.1/
```

```
8 9X,*Y(8) = X(21) E YM , QYD(8) =*,1PE8.1//
```

```
9 9X,* U( 5) E YMDM , RD(5) =*,1PE8.1/
```

```
A 9X,* U( 6) E DYTCVDM, RD(6) =*,1PE8.1/
```

```
END
```

C
C
C

```
SUBROUTINE RICCAT2 (N,NN,AM,W,W11,W11INV,W21,PG,P,Q,A,AT,AM12,
```

```
1 AXR1,AXR2,RTC,RTCS,DRAH,R,RT,OK,RDINV,NU,
```

```
2 IPRW,IPRR,IPRS,IPCH,NCASE,IPCHQ,NCASEF,IX,RTCSIN,KD,ISHORT)
```

C
C

```
13.1.72
```

C
C

```
CALCULATE THE RICCATI MATRIX P AND CHECK IF THE STEADY STATE  
RICCATI EQUATION IS CORRECT
```

C
C

$$0 = Q + [A]T^*P + P^*A - P^*B^*[R]INV^*[B]T^*P \quad (1)$$

C
C

```
P IS CALCULATED AS FOLLOWS (POTTER,VAUGHAN,ETC.)
```

C
C

```
W E EIGENVECTOR MATRIX OF STATE-COSTATE MATRIX AM CALCULATED IN SCSMX  
PICK UP THE STABLE EIGENVECTORS FROM W (THEY MUST BE HALF THE TOTAL NUMBER)  
AND CONSTRUCT W11, W21, WHERE W11 CONSISTS OF THE UPPER HALVES OF  
THE STABLE EIGENVECTORS WHEREAS W21 CONSISTS OF THE LOWER HALVES.  
THEN ...
```

C
C

$$P = W21^*[W11]INV \quad (2)$$

C
C

```
P SHOULD BE REAL, SYMMETRIC AND POSITIVE DEFINITE. THE FIRST TWO PROPERTIES  
ARE CHECKED INSIDE THIS ROUTINE, THE THIRD IN THE MAIN PROGRAM.
```

C
C

```
FINALLY CHECK EQUATION (1), RECALLING THAT, FROM SCSMX
```

C
C

```
STATE-COSTATE MATRIX M (OR AM), PARTITIONED INTO 4 EQUAL QUARTERS
```

C
C

```
[[      ] [      ] [      ] [      ] [      ] [      ]
```

```

C      [ AM11  AM12 ]      [ A      - B*(RIINV*(RTIT )
C
C      AM = [[      ][      ][      ] = [[      ][      ][      ]
C      (2N,2N) [[      ][      ][      ] [[      ][      ][      ]
C      [ AM21  AM22 ]      [ - 0      - [A]T      ]
C      [[      ][      ][      ] [[      ][      ][      ]
C
C      COMPUTE  KD (DISTURBANCE GAIN MATRIX)
C
C      KD = [RIINV*(RTIT*([W11])INVIT*[-(RTCS)INV]*[W21]T
C
C      THIS IS TO BE USED FOR THE SIMULATION OF CASES WHERE THE DISTURBANCE
C      INPUTS AND THE STEADY-STATE VALUES OF THE CONTROL INPUTS ARE NOT 0.
C
C      UC(OPT)(T) = - KR*X(T) - KDR*BD*UD + [I - KDR*BC]*UCBAR
C      = - KR*X(T) + KDU
C
C      WHERE
C
C      UC      = CONTROL INPUTS VECTOR (PART OF U)
C      UGBAR   = SETTLING-DOWN VALUE OF UC
C      UD      = DISTURBANCE INPUTS VECTOR (PART OF U)
C      BC      = PART OF MATRIX B CORRESPONDING TO UC
C      BD      = PART OF MATRIX B CORRESPONDING TO UD
C      KR      = PART OF MATRIX K CORRESPONDING TO UC
C      KDR     = PART OF MATRIX K CORRESPOND. TO UC
C      RTCS    = DIAGONAL MATRIX OF THE STABLE EIGENVALUES
C
C      COMPLEX W(NN,NN),W11(N,N),W11INV(N,N),W21(N,N),PC(N,N),RTC(NN),
C      1      RTCS(N),RTCSIN(N)
C      REAL AM(NN,NN),P(N,N),Q(N,N),A(N,N),AT(N,N),AM12(N,N),AXR1(N,N),
C      1      AXR2(N,N),B(N,NU),BT(NU,N),ROINV(NU),DK(NU,N),KD(NU,N)
C      INTEGER DRAW(N,NN),IX(NN)
C      DATA ICASE,IAOC1,IAOC2/54CASE ,10HAOC(OPT. C,10HONTROLLED)/,
C      1      IQOC1,IQOC2/      10HQOC (0 + [,10HKIT*R*K) /
C
C      CONSTRUCT  W11 = UPPER HALVES OF STABLE EIGENVECTORS
C      ..      W21 = LOWER      ..      ..      ..
C
C      J = 0
C      DO 4 JJ=1,NN
C      IF (RTC(JJ)) 1,4,4
C      1 J = J+1
C      RTCS(J) = RTC(JJ)
C      DO 2 II=1,N
C      I = II
C      2 W11(I,J) = W(II,JJ)
C      N1 = N+1
C      DO 3 IT=N1,NN
C      I = IT-N
C      3 W21(I,J) = W(II,JJ)
C      4 CONTINUE
C      PRINT 101
C      IF (IPRW) 10,17,10
C      17 PRINT 111
C      GO TO 10
C      10 CALL WRCMX (W11,N,N,10HW11(UPSTR),RTCS,1)
C      CALL DRCMX1(W11,N,N,10HW11(UPSTR),DRAW,NN)
C      CALL WRCMX (W21,N,N,10HW21(LOSTR),RTCS,1)
C      CALL DRCMX1(W21,N,N,10HW21(LOSTR),DRAW,NN)
C
C      AM IS NEEDED AS AN AUXILIARY MATRIX IN THE INVERSION COMING NEXT.
C      THEREFORE, THE CARVUP OPERATIONS NECESSARY FOR CHECKING THE STEADY STATE
C      PICCATI EQUATION (SEE BELOW) ARE PERFORMED HERE.      20.10.70
C

```

19 CALL CARVUP (AM,AM12,NN,NN,N,N,1,N+1)

CALL CARVUP (AM,Q, NN,NN,N,N,N+1,1)

C
C
C

INVERT W11 AND CHECK IF PRODUCT IS I

CALL EQUML (W11,W11INV,N,N)

CALL INCOMA (W11INV,N,AM,NN,N,0,N,NN,NN,NERP,IX)

IW11 = 10HW11(UPSTB)

PRINT 117, NERP,IW11

IF (NERP) 14,15,14

14 IF (IPRW) 23,22,23

22 CALL WRCMX (W11,N,N,10HW11(UPSTB),RTCS,1)

CALL WRCMX (W21,N,N,10HW21(LOSTB),RTCS,1)

23 CALL INVCOM (W21,W11INV,N,IRANK)

IW21 = 10HW21(LOSTB)

PRINT 102, IW21,IRANK

CALL MULCOM (W21,W11INV,PC,N,N,N)

CALL DRCMX1 (PC,N,N,10HW21*W21INV,DRAW,NN)

STOP

15 CALL MULCOM (W11,W11INV,PC,N,N,N)

CALL DRCMX1 (PC,N,N,10HW11*W11INV,DRAW,NN)

C
C
C
C
C

P = W21*(W11)INV

(2)

CHECK IF P IS REAL AND SYMMETRIC. CONSTRUCT P(REAL) FROM PC(COMPLEX)

CALL MULCOM (W21,W11INV,PC,N,N,N)

IF (IPRW) 20,21,20

20 CALL WRCMX (PC,N,N,10HP(RICCATI),RTCS,0)

CALL DRCMX1 (PC,N,N,10HP(RICCATI),DRAW,NN)

21 PRINT 103

ZEPO = 1.E-7

DO 6 I=1,N

DO 6 J=1,N

IF (REAL(PC(I,J))) 32,30,32

30 IF (ABS(AIMAG(PC(I,J))) - ZERO) 6,6,31

31 RATIMR = ABS(AIMAG(PC(I,J)))

PRINT 115, I,J,PC(I,J),RATIMR

GO TO 6

32 RATIMR = ABS(AIMAG(PC(I,J)))/ABS(REAL(PC(I,J)))

IF (RATIMR - ZERO) 6,5,5

5 PRINT 104, I,J,PC(I,J),RATIMR

IF (RATIMR - 1.E-4) 6,6,13

13 PRINT 109

6 P(I,J) = (PC(I,J) + PC(J,I))/2.

PRINT 105

JT = 1

NMIN = N-1

DO 8 I=1,NMIN

JT = JT+1

DO 8 J=JT,N

IF (CABS(PC(I,J))) 35,33,35

33 IF (CABS(PC(J,I)) - ZERO) 3,3,34

34 ASYMF = CABS(PC(J,I))

PRINT 116, J,I,PC(J,I),I,J,PC(I,J),ASYMF,J,I

GO TO 8

35 ASYMF = CABS(PC(I,J)-PC(J,I))/CABS(PC(I,J))

IF (ASYMF - ZERO) 8,7,7

7 PRINT 106, J,I,PC(J,I),I,J,PC(I,J),ASYMF

IF (ASYMF - 1.E-4) 8,8,16

16 PRINT 110

8 CONTINUE

C

C RECALL JO CALCULATE EIGENVALUES OF P AFTER CALLING RICCAT, TO CHECK P.DEF.

C

C CHECK STEADY STATE RICCATI EQUATION

C CALL CARVUP (AM,AM12,NN,NN,N,N,1,N+1) OPERATION PERFORMED ABOVE
 C CALL CARVUP (AM,0, NN,NN,N,N,N+1,1) OPERATION PERFORMED ABOVE
 C CALL NEGATE (Q,Q,N,N)
 C CALL TRANSP (A,AT,N,N)
 C CALL DRRMX (A ,N,N,DRAW,30H A)
 C CALL DRRMX (AT ,N,N,DRAW,30H (AT)
 C CALL DRRMX (AM12,N,N,DRAW,30H AM12)
 C CALL DRRMX (Q ,N,N,DRAW,30H Q)

C

CALL MULTI (AM12,P,AXP1,N,N,N)
 CALL MULTI (P,AXR1,AXR2,N,N,N)
 CALL MULTI (Q,A,AXP1,N,N,N)
 CALL ADD (AXP1,AXR2,AXR2,N,N)
 CALL MULTI (AT,P,AXP1,N,N,N)
 CALL ADD (AXP1,AXR2,AXR2,N,N)
 CALL ADD (Q,AXR2,AXR2,N,N)
 IF (IPRS) 24,25,24
 24 CALL WPIHX (AXR2,N,N,3HSS0)
 25 CALL DRRMX (AXR2,N,N,DRAW,30H SS0(STEADY STATE RICCATI = 0))

C

C CHECK IF STEADY STATE RICCATI EQUATION IS ZERO

C

PRINT 107
 DO 10 I=1,N
 DO 10 J=1,N
 IF (ABS(AXR2(I,J)) - ZERO) 10,9,9
 9 PRINT 108, I,J,AXP2(I,J)
 IF (ABS(AXR2(I,J)) - 1.E-4) 10,10,12
 12 PRINT 109
 10 CONTINUE

C

C

C CONSTRUCT CONTROL LAW

C

C $K = [R]INV*[B]T^*P$ (3)

C SUCH THAT

C $U = - K*X$ (FOR THE UNDISTURBED CASE) (4)

C I.E.

C $XDOT = A*X + B*U$
 C $= A*X - B*[R]INV*[B]T^*P*X$ (5)

C HENCE THE OPTIMAL CONTROL MATRIX AOC IS

C

C $AOC = A - B*[R]INV*[B]T^*P$ (6)

C

CALL TRANSP (B,BT,N,NU)
 CALL MULTI (BT,P,BK,NU,N,N)
 DO 11 I=1,NU
 DO 11 J=1,N

C

11 BK(I,J) = BK(I,J)*BDINV(I)

C

CALL MULTI (B,BK,AXP1,N,NU,N)
 CALL NEGATE (AXP1,AXP1,N,N)
 CALL WRTMX (AXP1,N,N,3H-RK)
 CALL DRRMX (AXR1,N,N,DRAW,30H - B*K (ADDED CONTROL))

C

C COMPUTE (IF ASKED TO) QOC = Q + [K]T*R*K

C

C (THIS IS FOR CHECKING THE LIAPUNOV EQUATION WITH AOC AND QOC.
 C IT IS NOT NECESSARY FOR SOLVING THE OPTIMAL REGULATOR PROBLEM).

C

IF (IPCHQ) 28,29,28

28 PRINT 114

```
CALL MULTI (P,AXR1,AXR2,N,N,N)
CALL NEGATE (AXR2,AXR2,N,N)
CALL ADD (Q,AXR2,AXR2,N,N)
CALL WPIMX (AXR2,N,N,3H000)
CALL DPMX (AXR2,N,N,DRAW,3H000 (Q + (KIT*R*K)
PUNCH 112, ICASE,NCASE,I00C1,I00C2,NCASF8
CALL PPM2013 (AXR2,N,N)
PRINT 113, ICASE,NCASE,I00C1,I00C2,NCASF8
```

C

```
29 CALL ADD (A,AXR1,AXR1,N,N)
IF (ISHORT.EQ.1) GO TO 44
CALL WPIMX (AXR1,N,N,3H000)
CALL DPMX (AXR1,N,N,DRAW,3H000 (A OPTIMALLY CONTROLLED))
44 IF (IPCH) 26,27,26
26 PUNCH 112, ICASE,NCASE,IA0C1,IA0C2,NCASF8
CALL PPM2013 (AXR1,N,N)
PRINT 113, ICASE,NCASE,IA0C1,IA0C2,NCASF8
27 CONTINUE
```

C

RECALL TO CALCULATE EIGENVALUES OF AOC AFTER CALLING RICCAT)

C

CONSTRUCT $KD = [R]INV*[RIT]*[[W11]INV]*[[-RTCS]INV]*[W21]IT$

C

DO 36 I=1,N

36 RTCSIN(I) = - 1./RTCS(I)

CALL TRANSP (W21,PC,N,N)

C

PC IS DESTROYED, IS NOW [W21]IT

CALL MLMX02 (PC,W11,RTCSIN,N,2)

CALL TRANSP (W11INV,PC,N,N)

C

PC IS NOW [[W11]INV]IT

CALL MULCOM (PC,W11,W21,N,N,N)

C

W21 NOW STORES THE PRODUCT $[[W11]INV]IT*[-RTCS]INV*[W21]IT$

C

IT SHOULD BE A REAL MATRIX. CHECK IF IT IS AND CONSTRUCT REAL IN Q.

C

PRINT 118

DO 37 I=1,N

DO 37 J=1,N

IF (REAL(W21(I,J))) 41,39,41

39 IF (ABS(AIMAG(W21(I,J))) - ZERO) 37,37,40

40 RATIOIR = ABS(AIMAG(W21(I,J)))

PRINT 115, I,J,W21(I,J),RATIOIR

GO TO 37

41 RATIOIR = ABS(AIMAG(W21(I,J)))/ABS(REAL(W21(I,J)))

IF (RATIOIR - ZERO) 37,37,42

42 PRINT 104, I,J,W21(I,J),RATIOIR

IF (RATIOIR - 1.E-4) 37,37,43

43 PRINT 109

37 Q(I,J) = W21(I,J)

C

FINISH EXPRESSION KD

C

CALL MULTI (RT,Q,KD,NU,N,N)

DO 38 I=1,NU

DO 38 J=1,N

38 KD(I,J) = KD(I,J)*RTINV(I)

RETURN

101 FORMAT (141////////10X,*THE FOLLOWING TWO MATRICES ARE*////////10X,
1*W11 = UPPER HALVES OF THE STABLE EIGENVECTORS OF AM (STATE-COSTAT
2E MATRIX)*////////10X,

3*W21 = LOWER

4

102 FORMAT (////////10X,*RANK OF MATRIX *,A10,* IS*,I6////////)

103 FORMAT (1H1////10X,*THE MATRIX P (RICCATI) SHOULD BE REAL*///10X

1,*ELEMENTS WITH RATIO OF IMAG OVER REAL PART GT. 1.E-7, IF ANY, FOLLOW
2LOW*///20X,*REAL PART*,9X,*IMAG PART*,8X,*RATIO*//)

104 FORMAT (* P(*,I2,**,I2,*) = *,2E18.8, 5X,E8.1)

105 FORMAT (1H1////10X,*THE MATRIX P (RICCATI) SHOULD BE SYMMETRIC*
1// 10X,*THE PAIRS OF SYMMETRIC ELEMENTS WHICH SHOW A FRACTIONAL OF
2REFERENCE GT. 1.E-7, IF ANY, FOLLOW*///18X,*REAL PART*,7X,*IMAG
3PART*,21X,*REAL PART*,7X,*IMAG PART*,7X,*FRACT. ASSYMETRY*//)

106 FORMAT (2(* P(*,I2,**,I2,*) = *,E15.8,**,E15.8),7X,E10.2)

107 FORMAT (1H1////10X,*THE MATRIX SSO (STEADY STATE RICCATI EQUATION)
1 SHOULD BE ZERO*///10X,*P REAL HAS BEEN CONSTRUCTED BY AVERAGING
2THE SYMMETRIC ELEMENTS*///10X,*OF THE COMPLEX MATRIX AND NEGLECTING
3THE IMAGINARY PART)*///10X,*ELEMENTS GT. 1.E-7, IF ANY, FOLLOW*///
4//)

108 FORMAT (* SSO(*,I2,**,I2,*) = *,E18.8)

109 FORMAT (1H+,66X,*GT. 1.E-4*)

110 FORMAT (1H+,114X,*GT. 1.E-4*)

111 FORMAT (////10X,*THEY ARE NOT PRINTED IN THIS RUN, EXCEPT IF W11
1 IS SINGULAR*///)

112 FORMAT (A5,I3,5X,2A10,24X,* (FOR CASE *,I2,* OF ERROR1)*)

113 FORMAT (1X,A5,I3,5X,2A10,24X,* (FOR CASE *,I2,* OF ERROR1)*)

114 FORMAT (////10X,*MATRIX PRODUCT FOLLOWS*///10X,13H000 = 0 + [KIT*
1R*K/10X,18H = 0 + P*R*K //)

115 FORMAT (* P(*,I2,**,I2,*) = *,2E18.8, 5X,E8.1,5X,*RATIO IN THIS
1LINE IS REALLY IMAG(P)*)

116 FORMAT (2(* P(*,I2,**,I2,*) = *,E15.8,**,E15.8),7X,E10.2,5X,
1 *F.A. IS P(*,I2,**,I2,*)*)

117 FORMAT (////10X,*ERROR OF INVERSION = *,I3,* (IF 0, INVERSI
1ON OF *,A10,* IS SUCCESSFULLY COMPLETED)*///)

118 FORMAT (////10X,66HTHE MATRIX PRODUCT [(W11)INVT*[- RTCSINV*[W
121IT SHOULD BE REAL //10X,*ELEMENTS WITH RATIO OF IMAG OVER RE
2AL PART GT. 1.E-7, IF ANY, FOLLOW*///20X,*REAL PART*,9X,*IMAG PAR
3T*,8X,*RATIO*//)

END

SUBROUTINE CTORBS1 (P,C,N,NU,NY,W,WINV,DVD,AXC1,AXC2,RTC,ICTR,
1 NCTR,ICOL,OPAW,NN)

9.8.71

ROUTINE TO CHECK CONTROLLABILITY AND OBSERVABILITY OF SYSTEM

$$\dot{X} = A \cdot X + B \cdot U \quad (1)$$

$$Y = C \cdot X \quad (2)$$

IN CONJUNCTION WITH THE COST MATRIX Q OF THE COST INTEGRAND

$$[X]^T Q X + [U]^T R U \quad (3)$$

CONTROLLABILITY INVOLVES THE PAIR OF MATRICES A, B.

OBSERVABILITY IN THE CASE WE ARE INTERESTED IN, INVOLVES THE PAIR A, CQ

$$\text{WHERE } [CQ]^T CQ = Q \quad (4)$$

IN OUR CASE (SEE ROUTINE SCOSX) WE USE A COST MATRIX QY FOR Y RATHER
THAN Q FOR X. THEREFORE THE FIRST TERM OF (3) IS

$$[Y]^T QY = [X]^T [C]^T QY C X \quad (5)$$

HENCE

$$Q = [CQ]^T CQ = [C]^T QY C \quad (6)$$

C IF WE SET QY DIAGONAL (AND WE DO SO IN SCOMX), THEN

$$CQ = SORT(QY)*C \quad (7)$$

(CLEARLY IF QY HAS ALL DIAGONAL ELEMENTS NONZERO, IT IS ENOUGH WITH CHECKING OBSERVABILITY OF THE PAIR A, C)

C DIAGONALIZING THE SYSTEM (1),(2) INTO THE SYSTEM

$$\begin{matrix} / \\ / \end{matrix} \quad ZDOT = D*Z + E*U \quad (8)$$

$$\begin{matrix} / \\ / \end{matrix} \quad Y = F*Z \quad (9)$$

C WHERE

C WHERE

$$Z = [W]INV*X$$

$$D = [W]INV*A*W$$

$$E = [W]INV*B$$

$$F = C*W$$

$$W \equiv \text{EIGENVECTOR MATRIX}$$

(10)

(11)

C THE SYSTEM (8),(9) IS COMPLETELY CONTROLLABLE IFF

C 1. E HAS AT LEAST ONE NONZERO ELEMENT IN EACH ROW

C AND COMPLETELY OBSERVABLE IFF

C 2. F HAS AT LEAST ONE NONZERO ELEMENT IN EACH COLUMN

C SINCE WE ARE CHECKING OBSERVABILITY OF THE PAIR A, CQ, THE MATRIX
C F OF CONDITION (2.) SHOULD REALLY BE

$$FQ = CQ*W = SORT(QY)*C*W \quad (12)$$

C THE EIGENVECTOR MATRIX W OF A AND ITS INVERSE [W]INV MUST BE CALCULATED
C BEFORE CALLING CTRORS

C REAL B(N,NU),C(NY,N),QYD(NY)

C COMPLEX W(N,N),WINV(N,N),AXC1(N,NU),AXC2(NY,N),RTC(N)

C INTEGER ICTR(NU),ICOL(ICTR),DRAW(N,NN)

C PRINT 1

C (1) CONTROLLABILITY

$$E = [W]INV*B$$

C CALL MVLOR (WINV,7,AXC1,N,N,NU)

C CALL WPCMX (AXC1,N,NU,10H E = WINV*B,RTC,0)

C CALL DRCHX1(AXC1,N,NU,10H E = WINV*B,DRAW,NN)

C KCOL = 1

C DO 8 I=1,NU

C IF (ICTR(I) - 1) 8,9,8

C 9 ICOL(KCOL) = I

C KCOL = KCOL+1

C 8 CONTINUE

C PRINT 103,((ICOL(I)),I=1,ICTR)

C KCTR = 1

C DO 3 I=1,N

C DO 4 J=1,NU

C IF (ICTR(J) - 1) 4,7,4

C 7 IF (ABS(REAL(AXC1(I,J))) - 1.E-2) 4,3,3

C 4 CONTINUE

C PPINT 101, I

C KCTR = 0

C 3 CONTINUE

IF (KCTP) 10,11,10

10 PRINT 104

(2) OBSERVABILITY

FQ = SORT(QY)*C*W

11 CALL MULRC (C,W,AXC2,NY,N,N)

DO 2 I=1,NY

DO 2 J=1,N

2 AXC2(I,J) = AXC2(I,J)*SQRT(QY(I))

CALL WPCMX (AXC2,NY,N,10H FQ = CQ*W,RTC,0)

CALL ORCMX1(AXC2,NY,N,10H FQ = CQ*W,ORAW,NN)

KORS = 1

DO 5 J=1,N

DO 6 I=1,NY

IF (ABS(REAL(AXC2(I,J))) - 1.E-2) 6,5,5

6 CONTINUE

PRINT 102, J

KORS = 0

5 CONTINUE

IF (KORS) 12,13,12

12 PRINT 105

13 RETURN

1 FORMAT (1H1/////////31X,*C O N T R O L L A B I L I T Y AND 0
19 S E R V A B I L I T Y/////////31X,25HSYSTEM YDOT = A*X + B*U/31X,
219H Y = C*X//31X,*HAS BEEN DIAGONALIZED INTO*//40X, 16H
37DOT = D*Z + E*U/43X,7HY = F*Z,20X,20HWHERE E = (W)INV*B/78X, 7H
4F = C*W/73X,22HW = EIGENVECTOR MATRIX/31X,*THE COST INTEGRAND IS*//
5/40X,19H(X)T*Q*X + [UIT*P*U//31X,*WHERE Q CAN BE EXPRESSED AS*//
640X,27H(COIT*Q) OR AS [OIT*QY*C//31X,*WHERE QY IS THE COST MAT
7RIX OF THE OUTPUTS Y*//40X,9H(Y)T*QY*Y//31X,*FROM ABOVE*//40X,
8 15HCO = SORT(QY)*C//31X,*AND WE DEFINE*//40X,9HFC = CQ*W/////////
910X,*THE SYSTEM IS COMPLETELY CONTROLLABLE IFF THE MATRIX E THA
AT FOLLOWS HAS AT LEAST ONE NONZERO ELEMENT IN EACH ROW*//10X,*..

B OBSERVABLE FQ

C COLUMN*//)

101 FORMAT (10X,*ROW (*,I2,*) OF E HAS NO ELEMENT WITH ABS VALUE OF
1REAL PART GE. 1.E-2 IN THE COLUMNS MENTIONED ABOVE*)

102 FORMAT (10X,*COLUMN (*,I2,*) OF FQ HAS NO ELEMENT WITH ABS VALUE
1 OF REAL PART GE. 1.E-2*)

103 FORMAT (///10X,*COLUMNS CORRESPONDING TO THE CONTROLLED INPUTS ARE
1 *,10(I3,*,*)///)

104 FORMAT (10X,*EACH ROW OF E HAS AT LEAST ONE ELEMENT WITH ABS VAL
1UE *//10X,*OF REAL PART GE. 1.E-2 IN THE COLUMNS MENTIONED ABOVE*//)

105 FORMAT (///10X,*EACH COLUMN OF FQ HAS AT LEAST ONE ELEMENT WITH
1ABS VALUE OF REAL PART GE. 1.E-2*//)

END

SUBROUTINE EIGVV1 (IVC,IPR1,IPR2,IPR3,IPR4,N,A,B,C,Z,Y,RTCX,ECX,
1 X,XI,V,VI,W,IPON,G,H,DA,OX,OV,OC,OD,AD,XD,VD,
2 GQ,HQ)

10.4.70

ROUTINE FOR CALCULATING EIGENVALUES AND EIGENVECTORS OF A REAL SQUARE MATRIX
MODIFIED VERSION (TEPPER L.) OF EIGSYS FROM NORTHWESTERN UNIV. (WEINER L.H.)
USES SUBROUTINES HESSEN,ORIG,OPT,ASQR,EIGVEC,DEISP,WRCMX,ORCMX
EIGVEC IS BASED ON WILKINSON'S INVERSE ITERATION METHOD

C LIST OF MOST IMPORTANT ARRAYS AND CONTROL POINTS

C A = INPUT MATRIX, REAL (N,N)
 C B,C = WORKING MATRICES, REAL (N,N)
 C DA = WORKING MATRIX, COMPLEX (N,N)
 C Z = EIGENVALUES, REAL PART (N)
 C Y = .. IMAG .. (N)
 C PTOX = .. COMPLEX. (N)
 C ECX = COMPLEX EIGENVECTOR MATRIX (N,N)
 C
 C X = CURRENT R.H. EIGENVECTOR, REAL PART (N)
 C XI = ... IMAG .. (N)
 C V = .. L.H. .. REAL .. (N)
 C VI = ... IMAG .. (N)
 C
 C IVC = 0 → NO EIGENVECTOR CALCULATION
 C IVC = 1 → CALCULATE REGULAR (R.H.) EIGENVECTORS
 C IVC = 2 → .. TRANSPOSED(L.H.) ..
 C IVC = 3 → .. BOTH ..
 C
 C IPR1 = 0 → VALUES OF CURRENT EIGENVECTORS ARE NOT PRINTED
 C IPR2 = 0 → CORRECTED EIGENVALUES, RESIDUAL VECTORS, ETC., NOT PRINTED
 C IPR3 = 0 → EIGENVECTOR MATRIX (COMPLEX VERSION) NOT PRINTED
 C IPR4 = 0 → INPUT MATRIX A NOT PRINTED

REAL A(N,N), B(1,N), C(N,N), Z(N), Y(N), W(N,4), X(N), XI(N), V(N), VI(N),
 1 G(N), H(N)
 INTEGER IROW(N,2), T2

C T2 TO BE USED ONLY WHEN SYSTEM ON TAPE
 C

COMPLEX DA(N,N), DX(N), DV(N), DC(N), DD(N), ECX(N,N), PTOX(N)
 DOUBLE AD(N,N), XD(N), VD(N), GD(N), HD(N)
 COMPLEX DL, DU, DP, DPP, DRL, DLEW
 DOUBLE LD, SD, RD, RPP, RLQ
 EQUIVALENCE (DL, LD), (DU, SD), (DP, RD), (DPP, RPP), (DRL, RLQ)
 DATA IWRQ, IWRCL/10H.R.H. EIGVC, 10H.L.H. EIGVC/

C	T2 = 0	EIGSYS
1	M=N	EIGSYS
6	DO 12 I=1,N	EIGSYS
	DO 12 J=1,N	EIGSYS
12	B(I,J)=A(I,J)	EIGSYS
	IF (IPR4) 1022,1023,1022	
1022	PPRINT 902	EIGSYS
	DO 8 I=1,N	EIGSYS
8	PPRINT 9,I, (A(I,J),J=1,N)	EIGSYS
1023	CALL HESSEN (A,M,N)	
	CALL DREIG(A,M,Z,Y,1,N)	
	DO 20 I=1,N	EIGSYS
	DO 20 J=1,N	EIGSYS
20	A(I,J)=D(I,J)	EIGSYS
	CALL DE1SP (A,M,DET,IEXP,N)	
	NSW = 1	EIGSYS
80	TRACE = 0.0	EIGSYS
	TRICE = 0.0	EIGSYS
	PROD = 1.0	EIGSYS
	I = 0	EIGSYS
43	I = I + 1	EIGSYS
	IF (Y(I)) 41,42,41	EIGSYS
41	PROD = PROD * (Z(I)**2 + Y(I)**2)	EIGSYS
	I = I + 1	EIGSYS
	GO TO 40	EIGSYS
42	PROD = PROD * Z(I)	EIGSYS

```

40 IF(I - N)43,45,45
EIGSYS

45 DO 48 I = 1,N
TRACE = TRACE + B(I,I)
EIGSYS
48 TRICE = TRICE + Z(I)
EIGSYS
PPRINT 950,TRACE,DET, IEXP,TRICE,PROD
EIGSYS
IF (IVC) 1010,1000,1010
1010 GO TO (85,1000), NSW
85 DO 23 I=1,N
EIGSYS
DO 23 J=1,N
EIGSYS
23 A(I,J) = R(I,J)
EIGSYS
CALL ASQR ( T2,B,A,N,N,N)

C
C      A IS NOW A-SQUARE      R IS THE ORIGINAL MATRIX A
C
DO 21 K=1,N
IF (IPP1) 1037,1038,1037
1037 PPRINT 901
EIGSYS
1038 ISW = 0
EIGSYS
26 DO 22 I=1,N
EIGSYS
DO 22 J=1,N
EIGSYS
22 R(I,J)=A(I,J)
EIGSYS
CALL EIGVEC (IVC,B,C,W,IROW,X,XI,V,
1 VI,7(K),Y(K),N,N ,T2,SW1,ICOUNT,EPR,2 )
EIGSYS
IF (ICOUNT) 31,41,28
EIGSYS
81 PPRINT 915,Z(K),Y(K)
EIGSYS
GO TO 21
EIGSYS
28 IF(Y(K)) 27,29,27
EIGSYS
29 DO 70 I = 1,N
EIGSYS

C
C      XD IS D.P. R.H. E-VECTOR X
C
XD(I) = X(I)
EIGSYS
VD(I) = V(I)
EIGSYS

C
C      VD IS D.P. L.H. E-VECTOR V
C
DO 70 J = 1,N
EIGSYS
70 AD(I,J) = A(I,J)
EIGSYS

C
C      AD IS D.P. INPUT MATRIX A
C
LD = Z(K)
EIGSYS

C
C      LD IS D.P. CURRENT REAL EIGENVALUE
C
DO 71 I = 1,N
EIGSYS
GD(I) = 0.0
EIGSYS
DO 71 J = 1,N
EIGSYS
71 GD(I) = GD(I) + AD(I,J) * XD(J)
EIGSYS

C
C      GD IS D.P. PRODUCT A*X
C
DO 73 I = 1,N
EIGSYS

C
C      GD IS NOW D.P. RESIDUAL A*X - LAMBDA*X
C
73 GD(I) = GD(I) - LD * XD(I)
EIGSYS
SD = 0.0
EIGSYS
RD = 0.0
EIGSYS
RRD = RLD = 0.0
EIGSYS
DO 74 I = 1,N
EIGSYS

C
C      SD IS D.P. PRODUCT V-TRANSPOSE(A*X - LAMBDA*X)
C

```

```

SD = SD + VD(I) * GD(I)
EIGSYS

RRD = RRD + GD(I)**2
EIGSYS
C RRD IS DOUB.PREC. R.H. RESIDUAL VECTOR SQUARED
EIGSYS
C 74 RD = RD + VD(I) * XD(I)
EIGSYS
C RD IS D.P. VECTOR PRODUCT V-TRANPOSE * X
EIGSYS
C RR = RRD
EIGSYS
C DO 75 I = 1,N
EIGSYS
C G(I) = GD(I)
EIGSYS
C HD(I) = 0.0
EIGSYS
C DO 75 J = 1,N
EIGSYS
C 75 HD(I) = HD(I) + VD(J) * R(J,I)
EIGSYS
C HD IS THE D.P. PRODUCT V-TRANPOSE * A
EIGSYS
C DO 76 I = 1,N
EIGSYS
C HD(I) = HD(I) - V(I) * LD
EIGSYS
C 76 H(I) = HD(I)
EIGSYS
C H(I) IS NOW THE RESIDUAL (SINGLE PREC.) OF (V-TRANS*A - V-TRANS*LAMBDA
EIGSYS
C DO 78 I = 1,N
EIGSYS
C RLD IS DOUB.PREC. L.H. RESIDUAL VECTOR SQUARED
EIGSYS
C 78 RLD = RLD + HD(I) ** 2
EIGSYS
C ECX INCORPORATES COMPLEX EIGENVECTORS (FOR REAL EIGENVALUES)
C IVC = 1,3 * ECX(COLUMN K) IS R.H. EIGENVECTOR K
C IVC = 2 * ECX(ROW K) IS L.H. EIGENVECTOR K
C GO TO (1020,1025,1020), IVC
1020 DO 1024 I=1,N
1024 ECX(I,K) = X(I)
GO TO 1027
1025 DO 1026 I=1,N
1026 ECX(K,I) = V(I)
C 1027 IF (IPR1) 1012,1013,1012
1012 PRINT 905, K, Z(K), (X(I), V(I), G(I), H(I), I = 1,N)
EIGSYS
1013 Z(K) = LD + SD / RD
EIGSYS
C Z(K) IS NOW THE SINGLE PREC. IMPROVED EIGENVALUE
EIGSYS
C RL = PLD
EIGSYS
C IF (IPR2) 1018,1019,1019
1018 PRINT 910, SW1, ICOUNT, ERR,RR,RL
EIGSYS
PPRINT 916
EIGSYS
PRINT 961 , K, Z(K)
EIGSYS
1019 ISW = ISW + 1
EIGSYS
GO TO (26,21), ISW
EIGSYS
27 DL = CMPLX(Z(K),Y(K))
EIGSYS
C DL IS COMPLEX EIGENVALUE
EIGSYS
C DO 50 I=1,N
EIGSYS
C DC(I) = 0.0
EIGSYS
C DD(I) = 0.0
EIGSYS
C DX(I) = CMPLX(X(I),XI(I))
EIGSYS
C
EIGSYS

```

```

C   DX IS COMPLEX R. H. EIGENVECTOR                                EIGSYS
C
C   DV(I) = CMPLX(V(I),VI(I))                                     EIGSYS
C   DV IS COMPLEX L. H. EIGENVECTOR                               EIGSYS
C   ECX INCORPORATES COMPLEX EIGENVECTORS (FOR COMPLEX EIGENVALUES) EIGSYS
C   IVC = 1,3 → ECX(COLUMN K) IS R.H. EIGENVECTOR K
C   IVC = 2 → ECX(ROW K) IS L.H. EIGENVECTOR K
C
C   GO TO (1028,1029,1029), IVC
1028 ECX(I,K) = DX(I)
C   GO TO 1031
1029 ECX(K,I) = DV(I)
C
1031 DO 50 J = 1,N
C   50 DA(I,J) = B(I,J)
C   DA IS COMPLEX VERSION OF ORIGINAL INPUT MATRIX
C
C   DO 52 I = 1,N
C   DO 52 J = 1,N
C   DC(I) = DC(I) + DA(I,J) * DX(J)
52 DD(I) = DD(I) + DA(J,I) * DV(J)
C   DO 60 I = 1,N
C
C   DC(I) IS THE RESIDUAL OF A*X - LAMBDA*X
C
C   DC(I) = DC(I) - DL * DX(I)
60 DD(I) = DD(I) - DV(I) * DL
C
C   DD(I) IS THE RESIDUAL OF V*A - V*LAMBDA
C
C   DPR = DPL = 0.0
C   DP = 0.0
C   DU = 0.0
C   DO 54 I = 1,N
C   DPR = DPR + DC(I) ** 2
C   DPL = DPL + DD(I) ** 2
C
C   DPR IS COMPLEX R.H. RESIDUAL VECTOR SQUARED
C   DPL IS COMPLEX L.H. RESIDUAL VECTOR SQUARED
C
C   DU = DU + DV(I) * DC(I)
C
C   DU IS COMPLEX PRODUCT V-TRANS(A*X - LAMBDA*X)
C
54 DP = DP + DV(I) * DX(I)
C
C   DP IS COMPLEX VECTOR PRODUCT V-TRANS * X
C
C   DLNEW = DL + DU / DP
C
C   DLNEW IS THE COMPLEX IMPROVED VALUE FOR THE CURRENT EIGENVALUE
C
C   Y(K) = AIMAG(DLNEW)
C   Z(K) = REAL(DLNEW)
C
C   IF (IPR1) 1014,1016,1014
1014 PRINT 906, K, DL, (DX(I), DV(I), DC(I), DD(I), I = 1,N)
1016 IF (IPR2) 35,1017,35
C

```

EIGSYS

```

35 PRINT 911,SW1,ICOUNT,ERR,DRL,EIGSYS
PRINT 916,EIGSYS
PRINT 960, K, OLNEW,EIGSYS
1017 ISW = ISW + 1,EIGSYS
GO TO (26,21), ISW,EIGSYS
21 CONTINUE,EIGSYS
NSW = 2,EIGSYS
PPRINT 955,EIGSYS
GO TO 80,EIGSYS
901 FORMAT (1H1)
955 FORMAT(///// * MATRIX PROPERTIES REPRINTED HERE USING CORRECTED EIGENVALUES *)
915 FORMAT (///
1130H * * * * *
2 * * * * *
3 // * SURROUTINE EIGVEC CANNOT FIND E-VECTORS FOR LAMBDA = *
42E17.9, * FURTHER COMPUTATIONS FOR THIS LOOP ARE OMITTED **
5130H * * * * *
6 * * * * *
7 )
905 FORMAT(// 5X, * R. H. E-VECTOR OF A L. H. E-VECTOR OF A * 4X
1 16H(A*X - LAMBDA*X) 5X 16H(V*A - V*LAMBDA) 5X
2 * FOR LAMBDA(*I2, *) = * E17.9, //(2E22.9,2E21.9))
9 FORMAT(*RPN NO.*I7, / (10X,10E12.3))
910 FORMAT(/// * SW1 = *E17.9, * COUNT = * I3, * ERR = *E17.9, /
1* R.H. RESIDUAL VECTOR SQUARED = *E17.9,
2* L.H. RESIDUAL VECTOR SQUARED = *E17.9)
911 FORMAT(/// * SW1 = *E17.9, * COUNT = * I3, * ERR = *E17.9, /
1* R.H. RESIDUAL VECTOR SQUARED = * E17.9,E17.9 ,
2* L.H. RESIDUAL VECTOR SQUARED = * E17.9,E17.9 )
906 FORMAT(// , * FOR COMPLEX EIGENVALUE (*I2, *) = * E17.9,E17.9 //
1* R.H. COMPLEX EIGENVECTOR L.H. COMPLEX EIGENVECTOR
2 *32HCOMPLX RESID. OF (A*X - LAMBDA*X)
3 34H COMPLX RESID. OF (V*A - LAMBDA*V) / ( 4(E17.9,E17.9))
916 FORMAT( 5(/))
960 FORMAT(/// * THE CORRECTED COMPLEX EIGENVALUE(*I2, *) =
1 E17.9,E17.9 /)
902 FORMAT(5(/), * ... INPUT MATRIX FOLLOWS ...*)
961 FORMAT(/// * THE CORRECTED REAL EIGENVALUE (*I2, *) = *E17.9, /)
950 FORMAT( /// * TRACE OF INPUT MATRIX = *E17.9, 10X
1* DETERMINANT OF INPUT MATRIX = *E17.9, 2H * I4, /
2 * SUM OF EIGENVALUES = *E17.9, 10X
3* PRODUCT OF EIGENVALUES = *E17.9)
C
C CONSTRUCT AND WRITE VECTOR OF FINAL EIGENVALUES (COMPLEX VERSION)
C
1000 DO 1001 I=1,N
1001 RTGX(I) = CMPLX(Z(I),Y(I))
PRINT 1002
1002 FORMAT(1H1,31X,*FINAL EIGENVALUES*////23X,*REAL PART*,7X,*IMAG PAR
1T**//)
DO 1003 I=1,N
1003 PRINT 1004, I,RTGX(I)
1004 FORMAT (10X,*ROOT (*,I2,*)*,2E16.9)
C
C WRITE AND DRAW MATRIX OF COMPLEX EIGENVECTORS ** (NO DRAWING NOW) ***
C
IF (IVC) 1011,1015,1011
1011 IF(IPR3) 1021,1015,1021
1021 PPRINT 1032
1032 FORMAT (1H1//////////)
GO TO (1033,1035,1033), IVC
1033 PRINT 1034
1034 FORMAT (31X,*RIGHT HAND EIGENVECTORS MATRIX FOLLOWS*////31X,

```

```
1      38HA * EIGENVECTOR = EIGENVECTOR * LAMBDA)
```

```
      CALL WRCMX (ECX,N,N,IWRC,RTCX,1)
```

```
      GO TO 1015
```

```
1035 PRINT 1036
```

```
1036 FORMAT (31X,*LEFT-HAND EIGENVECTORS MATRIX FOLLOWS*////31X,
```

```
1      54EIGENVECTOR(TRANSP) * A = LAMBDA * EIGENVECTOR(TRANSP))
```

```
      CALL WRCMX (ECX,N,N,IWRC,RTCX,0)
```

```
C
```

```
C RECOVER ORIGINAL INPUT MATRIX A FOR FURTHER USE AFTER EIGVV1
```

```
C
```

```
1015 DO 1040 I=1,N
```

```
      DO 1040 J=1,N
```

```
1040 A(I,J) = B(I,J)
```

```
      RETURN
```

```
      END
```

```
EIGSYS
```

```
C
```

```
C
```

```
C
```

```
C
```

```
SUBROUTINE HESSEN(A,M,L)
```

```
HESSEN
```

```
SUBROUTINE TO PUT MATRIX IN UPPER HESSENBERG FORM.
```

```
HESSEN
```

```
DIMENSION A(L,L),B(200)
```

```
HESSEN
```

```
IF (M - 2) 30,30,32
```

```
HESSEN
```

```
32 DO 40 LC = 3,M
```

```
HESSEN
```

```
      N = M - LC + 3
```

```
HESSEN
```

```
      N1 = N - 1
```

```
HESSEN
```

```
      N2 = N - 2
```

```
HESSEN
```

```
      NI = N1
```

```
HESSEN
```

```
      DIV = ABS(A(N,N-1))
```

```
HESSEN
```

```
      DO 2 J = 1,N2
```

```
HESSEN
```

```
      IF(ABS(A(N,J))- DIV) 2,2,1
```

```
HESSEN
```

```
1      NI = J
```

```
HESSEN
```

```
      DIV = ABS(A(N,J))
```

```
HESSEN
```

```
2      CONTINUE
```

```
HESSEN
```

```
      IF(DIV) 3,40,3
```

```
HESSEN
```

```
3      IF(NI - N1) 4, 7,4
```

```
HESSEN
```

```
4      DO 5 J = 1,N
```

```
HESSEN
```

```
      DIV = A(J,NI)
```

```
HESSEN
```

```
      A(J,NI) = A(J,N1)
```

```
HESSEN
```

```
5      A(J,N1) = DIV
```

```
HESSEN
```

```
      DO 6 J = 1,M
```

```
HESSEN
```

```
      DIV = A(NI,J)
```

```
HESSEN
```

```
      A(NI,J) = A(N1,J)
```

```
HESSEN
```

```
6      A(N1,J) = DIV
```

```
HESSEN
```

```
7      DO 26 K = 1, N1
```

```
HESSEN
```

```
26      B(K) = A(N,K)/A(N,N-1)
```

```
HESSEN
```

```
      DO 45 J = 1,M
```

```
HESSEN
```

```
      SUM = 0.0
```

```
HESSEN
```

```
      IF (J - N1) 46,43,43
```

```
HESSEN
```

```
46      IF(B(J)) 41,43,41
```

```
HESSEN
```

```
41      A(N,J) = 0.0
```

```
HESSEN
```

```
      DO 42 K = 1,N1
```

```
HESSEN
```

```
      A(K,J) = A(K,J) - A(K,N1)*B(J)
```

```
HESSEN
```

```
42      SUM = SUM + A(K,J)*B(K)
```

```
HESSEN
```

```
      GO TO 45
```

```
HESSEN
```

```
43      DO 44 K = 1,N1
```

```
HESSEN
```

```
44      SUM = SUM + A(K,J)*B(K)
```

```
HESSEN
```

```
45      A(N1,J) = SUM
```

```
HESSEN
```

```
40      CONTINUE
```

```
HESSEN
```

```
30      RETURN
```

```
HESSEN
```

```
      END
```

```
HESSEN
```

```
SUBROUTINE QREIG(A,M,ROCTR,ROOTI,IPRNT,L)
```

```
QREIG
```


$$N=N-2$$

QREIG

```

1  IF(JJ) 1,177,177
26 IF(ABS(A(N,N-1)) .GT. 1.0E-10*ABS(A(N,N))) GO TO 50
29 IF(IPRNT) 85,87,86
86 PRINT 105, A(N,N), ZERO, ITER
87 ROOTR(N) = A(N,N)
   ROOTI(N) = 0.0
   N=N-1
   GO TO 177
50 IF(ABS(ABS(XNN/A(N,N-1))-1.0)-1.0E-6) 63,63,62
62 IF(ABS(ABS(XN2/A(N-1,N-2))-1.0)-1.0E-6) 63,63,700
63 VQ=ABS(A(N,N-1))-ABS(A(N-1,N-2))
   IF (ITEP-15) 53,164,64
164 IF(VQ) 165,165,166
165 R = A(N-1,N-2)**2
   SIG = 2.0*A(N-1,N-2)
   GO TO 60
166 R = A(N,N-1)**2
   SIG = 2.0*A(N,N-1)
   GO TO 60
64 IF(VQ) 67,67,66
66 IF(IPRNT) 88,85,88
88 PRINT 107, A(N-1,N-2)
   GO TO 84
67 IF(IPRNT) 89,87,89
89 PRINT 107, A(N,N-1)
   GO TO 86
700 IF(ITEP .GT. 50) GO TO 63
   IF(ITER .GT. 5) GO TO 53
701 ET = E*E + F*F
   IF(ET)702,601,702
702 GT = G*G + H*H
   IF(GT)703,601,703
703 Z1 = ((E-AA)**2 + (F-R)**2) / ET
   Z2 = ((G-C)**2 + (H-D)**2) / GT
   IF(Z1-0.25) 51,51,52
51 IF(Z2-0.25) 53,53,54
53 R=E*G-F*H
   SIG=E+G
   GO TO 60
54 R=E*E
   SIG=E+E
   GO TO 60
52 IF(Z2-0.25) 55,55,601
55 R=G*G
   SIG=G+G
   GO TO 60
601 R = 0.0
   SIG = 0.0
60 XNN=A(N,N-1)
   XN2=A(N-1,N-2)
   CALL CRT(A,N,R,SIG,D,L)
   AA=E
   R=F
   C=G
   D=H
   GO TO 12
104 FORMAT(///5X, 9HREAL PART 6X 14HIMAGINARY PART, 20X
1 13HTAKEN AS ZERO 6X *ITERATION NO.*/)
105 FORMAT(1X,E15.9,3X,E15.9, 42X I3)
107 FORMAT(52X, E17.8)
END

```

```

SUBROUTINE QRT(A,N,P,SIG,0,M)
DIMENSION A( M,M ),PSI(2),G(3)
N1 = N - 1
TA = N - 2
IP = IA
IF(N-3) 101,10,60
60 DO 12 J = 3,N1
J1 = N - J
IF(ABS(A(J1+1,J1))-0) 10,10,11
11 DEN = A(J1+1,J1+1)*(A(J1+1,J1+1)-SIG)+A(J1+1,J1+2)*A(J1+2,J1+1)+R
IF(DEN) 61,12,61
61 IF(ABS(A(J1+1,J1))*A(J1+2,J1+1)*(ABS(A(J1+1,J1+1)+A(J1+2,J1+2)
1-SIG)+ABS(A(J1+3,J1+2)))/DEN)-0) 10,10,12
12 IP=J1
10 DO 14 J=1,IP
J1=IP-J+1
IF(ABS(A(J1+1,J1))-0) 13,13,14
14 IQ=J1
13 DO 100 I=IP,N1
IF(I-IP) 16,15,16
15 G(1)=A(IP,IP)*(A(IP,IP)-SIG)+A(IP,IP+1)*A(IQ+1,IP)+R
G(2)=A(IP+1,IP)*(A(IP,IP)+A(IP+1,IP+1)-SIG)
G(3)=A(IP+1,IP)*A(IQ+2,IP+1)
A(IP+2,IP)=0.0
GO TO 19
16 G(1)=A(I,I-1)
G(2)=A(I+1,I-1)
IF(I-IA) 17,17,18
17 G(3)=A(I+2,I-1)
GO TO 19
18 G(3)=0.0
19 XK = SIGN(SQRT(G(1)**2 + G(2)**2 + G(3)**2), G(1))
22 IF(XK) 23,24,23
23 AL=G(1)/XK+1.0
PSI(1)=G(2)/(G(1)+XK)
PSI(2)=G(3)/(G(1)+XK)
GO TO 25
24 AL=2.0
PSI(1)=0.0
PSI(2)=0.0
25 IF(I-IQ) 26,27,26
26 IF(I-IP) 29,29,29
28 A(I,I-1)=-A(I,I-1)
GO TO 27
29 A(I,I-1)=-XK
27 DO 30 J=I,N
IF(I-IA) 31,31,32
31 C=PSI(2)*A(I+2,J)
GO TO 33
32 C=0.0
33 E=AL*(A(I,J)+PSI(1)*A(I+1,J)+C)
A(I,J)=A(I,J)-E
A(I+1,J)=A(I+1,J)-PSI(1)*E
IF(I-IA) 34,34,30
34 A(I+2,J)=A(I+2,J)-PSI(2)*E
30 CONTINUE
IF(I-IA) 35,35,36
35 L=I+2
GO TO 37
36 L=N
37 DO 40 J=IQ,L
IF(I-IA) 39,39,39
38 C=PSI(2)*A(J,I+2)

```



```

615 CERR = 0.0
EIGVEC

      IF(IVC2) 616, 619, 616
616 DO 618 I = 1, N
      XR(I) = -W(I,2)
      DO 617 J = 1, N
617 XR(I) = XR(I) + A(I,J)*XI(J)
618 XP(I) = XP(I)/ROOTI
      IF(IVC1) 619, 633, 619
619 DO 621 I = 1, N
      VR(I) = -W(I,4)
      DO 620 J = 1, N
620 VR(I) = VR(I) + A(J,I)*VI(J)
621 VP(I) = VR(I)/ROOTI
EIGVEC
C
C      SEARCH VECTORS FOR LARGEST ELEMENT AND NORMALIZE.
C
627 AMAX = 0.0
      DO 629 L = 1, N
      TEMP = VR(L)**2 + VI(L)**2
      IF(TEMP - AMAX) 629, 629, 629
628 AMAX = TEMP
      I2 = L
629 CONTINUE
      C1 = VR(I2)/AMAX
      C2 = -VI(I2)/AMAX
      DO 630 L = 1, N
      TEMP = VI(L)
      VI(L) = VP(L)*C2 + TEMP*C1
630 VP(L) = VR(L)*C1 - TEMP*C2
      IF(COUNT.EQ. 1) GO TO 632
      DO 631 LL = 1, N
631 CERR = AMAX1(CERR, ABS(VR(LL) - W(LL,3)), ABS(VI(LL) - W(LL,4)))
632 IF(IVC2) 633, 633, 633
633 AMAX = 0.0
      DO 635 L = 1, N
      TEMP = XR(L)**2 + XI(L)**2
      IF(TEMP - AMAX) 635, 635, 634
634 AMAX = TEMP
      I2 = L
635 CONTINUE
      C1 = XR(I2)/AMAX
      C2 = -XI(I2)/AMAX
      DO 636 L = 1, N
      TEMP = XI(L)
      XI(L) = XR(L)*C2 + TEMP*C1
636 XR(L) = XR(L)*C1 - TEMP*C2
      IF(COUNT.EQ. 1) GO TO 646
      DO 637 LL = 1, N
EIGVEC
C
C      TEST FOR CONVERGENCE.
C
637 CERR = AMAX1(CERR, ABS(XR(LL) - W(LL,1)), ABS(XI(LL) - W(LL,2)))
638 IF(COUNT.EQ. 1) GO TO 646
      IF(CERR.GE. 1.0E-4) GO TO 639
      IF(CERR.GE. CLIM) GO TO 648
      CLIM = CERR
      IF(CLIM.LE. 1.0E-9) GO TO 648
639 IF(COUNT.GE. 15) GO TO 68
647 COUNT = COUNT + 1
      IF(ROOTI) 642, 673, 642
642 IF(IVC2) 640, 644, 640
640 DO 641 LL = 1, N
      W(LL,1) = XR(LL)
641 W(LL,2) = XI(LL)
EIGVEC

```


2GENVECTOR(S) CONTINUES AT USER'S OPTION*//)

EIGVEC

[illegible]

CCC

```

SUBROUTINE DF1SP (A,N1,DET,IEXP,N2)
  DIMENSION A(N2,N2)
  NC=N1
  DET=1.0
  IEXP=0
  ASSIGN 41 TO IJK
8  DO 100 K=1,NC
  IF (K-NC) 11,9,11
9  ASSIGN 98 TO IJK
  GO TO 78
11 TEMP=0.0
  II=0
  DO 15 I=K,NC
  DO 15 J=K,NC
  IF (ABS(A(I,J)).LE.TEMP) GO TO 15
  TEMP=ABS(A(I,J))
  II=I
  JJ=J
15 CONTINUE
  IF (II.EQ.0) 20,21
20 DET=0.
  IEXP=0
  RETURN
21 IF (II-K) 25,29,25
25 DO 30 J=K,NC
  TEMP=A(II,J)
  A(II,J)=A(K,J)
30 A(K,J)=TEMP
  DET=-DET
29 IF (JJ-K) 35,39,35
35 DO 40 I=K,NC
  TEMP=A(I,JJ)
  A(I,JJ)=A(I,K)
40 A(I,K)=TEMP
  DET=-DET
38 SAVE=DET*A(K,K)
  IF (LEGVAR(SAVE)) 72,75
75 IF (ABS(SAVE).LT.1.E-250) 77,81
77 DET=(DET*1.E250)*A(K,K)
  IEXP=IEXP-250
  GO TO IJK,(41,98)
72 DET=(DET*1.E-250)*A(K,K)
  IEXP=IEXP+250
  GO TO IJK,(41,98)
81 DET=SAVE
  GO TO IJK,(41,98)
41 M=K+1
  DO 60 J=M,NC
  A(K,J)=A(K,J)/A(K,K)
  DO 60 I=M,NC
60 A(I,J)=A(I,J)-A(K,J)*A(I,K)
100 CONTINUE
98 RETURN

```

END

DE1SP

```

C
C
C
C      SUBROUTINE WRCHX (A, M,N,I4R,RTC,IRT)
C
C 18.3.70
C
C WRITE MATRIX OF COMPLEX VECTORS (OPTION IRT=0)
C OPTION IRT=1 INCLUDES THE EIGENVALUES WRITTEN AS BOTTOM ELEMENTS
C OF EACH CORRESPONDING EIGENVECTOR
C
      COMPLEX A(M,N),RTC(N)
      INTEGER JJ(4)
      K = 0
      NN = N
      KK = 1
      M1 = M
      IF (NN - 4) 4,4,5
5 K = NN - 4
      NN = 4
4 WRITE (6,100)
100 FORMAT (1H1)
      IF (IRT) 103,103,101
101 WRITE (6,102) M
102 FORMAT (31X,*EIGENVECTORS AND EIGENVALUES OF GIVEN MATRIX**//10X,
1*THE FIRST *,I2,* ELEMENTS ARE THE COMPONENTS OF THE EIGENVECTOR
2 AND THE LAST ELEMENT IS THE CORRESPONDING EIGENVALUE**//)
      M1 = M + 1
103 WRITE (6,12) I4R,M1,NN,KK
12 FORMAT (31X,*MATRIX *,A10,* IS PARTITIONED INTO DIMENSIONS *,
1I2,* BY *,I2//31X,*THE ELEMENTS OF THIS MATRIX STARTING AT ROW 1
2 AND COLUMN *,I2,* ARE AS FOLLOWS**//7X,*REAL PART*,5X,*IMAGINARY
3 PART*,4X,*REAL PART*,5X,*IMAGINARY PART*,4X,*REAL PART*,5X,*IMAGI
4NARY PART*,4X,*REAL PART*,5X,*IMAGINARY PART*)
      L = KK + NN - 1
      KI = 0
      DO 2 I=KK,L
      KI = KI + 1
2 JJ(KI) = I
      WRITE (6,3) ((JJ(I),JJ(I)),I=1,NN)
3 FORMAT (1H0,* COL NO*,3X,I3,7(13X,I3))
      WRITE (6,8)
8 FORMAT (2X,*ROW NO**//2X,*.*//2X,*.*//)
      DO 1 I=1,M
1 WRITE (6,11) I,(A(I,J),J=KK,L)
11 FORMAT (1X,I3,1X,8(E15.8,1X))
      IF (IRT) 105,105,104
104 WRITE (6,11) M1,(RTC(J),J=KK,L)
105 IF (K.LE.0) RETURN
      IF (K - 4) 6,6,7
6 NN = K
      KK = KK + 4
      K = 0
      GO TO 4
7 K = K - 4
      KK = KK + 4
      GO TO 4
      END

```

```

C
C
C
C      SUBROUTINE DRRX (A,M,N,DRAW,IWR)
C

```

C 13.5.70

C
C DRAWS MATRIX, X FOR NONZERO ELEMENTS
C . FOR ZERO ..
C

C ZERO DEFINED AS $1.E-10$

```

      REAL A(M,N)
      INTEGER DRAW (M,N), IWR(3), IFRMT(99), LFRMT(4), IN(99)
      DATA IZ,N7/14,,14X/
      DATA LFRMT/10H(10X,I2,2X,1H,,2H ,9HA1,1X,I2)/
      DATA IFRMT/
1     2H 1,2H 2,2H 3,2H 4,2H 5,2H 6,2H 7,2H 8,2H 9,2H10,2H11,2H12,
2     2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,
3     2H25,2H26,2H27,2H28,2H29,2H30,2H31,2H32,2H33,2H34,2H35,2H36,
4     2H37,2H38,2H39,2H40,2H41,2H42,2H43,2H44,2H45,2H46,2H47,2H48,
5     2H49,2H50,2H51,2H52,2H53,2H54,2H55,2H56,2H57,2H58,2H59,2H60,
6     2H61,2H62,2H63,2H64,2H65,2H66,2H67,2H68,2H69,2H70,2H71,2H72,
7     2H73,2H74,2H75,2H76,2H77,2H78,2H79,2H80,2H81,2H82,2H83,2H84,
8     2H85,2H86,2H87,2H88,2H89,2H90,2H91,2H92,2H93,2H94,2H95,2H96,
9     2H97,2H98,2H99/
      PRINT 1, M,N,IWR
      DO 9 I=1,N
        IN(I) = I
7     IF (IN(I) - 10) 9,9,9
8     IN(I) = IN(I) - 10
        GO TO 7
9     CONTINUE
      PRINT 2, (IN(I),I=1,N)
      PRINT 3
      ZERO = 1.E-10
      DO 4 I=1,M
        DO 4 J=1,N
          DRAW(I,J) = IZ
          IF (ABS(A(I,J)) - ZERO) 4,4,5
5     DRAW(I,J) = N7
4     CONTINUE
      LFRMT(3) = IFRMT(N)
      PRINT LFRMT, ((I,(DRAW(I,J),J=1,N),I),I=1,M)
      PRINT 3
      PRINT 2, (IN(I),I=1,N)
      PRINT 6
      RETURN
1     FORMAT (1H1,31X,*REAL MATRIX (*,I2,*,*,I2,*)      *,3A10//)
2     FORMAT (14X,99I1)
3     FORMAT (1H )
6     FORMAT (/31X,*ELEMENTS OF ABS VALUE GT 1.E-10 DRAWN AS X*/)
      END

```

C
C
C

SUBROUTINE CMPEI1 (RTA,RTAOC,N,PROAR,IC,NCASE,NCASEB,MO)

C 6.8.71

C COMPARES EIGENVALUES OF ORIGINAL SYSTEM WITH OPTIMALLY CONTROLLED.

C COMPARES PRODUCT OF COMPUTED EIGENVALUES WITH READ-IN PRODUCT.

```

      COMPLEX RTA(N),RTAOC(N)
      GO TO (30,20,20), IC
20  PRINT 100
      DO 7 I=1,N
        RATR = 0.
        RATI = 0.
        IF (REAL(RTA(I))) 1,2,1
1     RATR = REAL(RTAOC(I))/REAL(RTA(I))
2     IF (AIMAG(RTA(I))) 3,4,3

```

```

3  RATI = AIMAG(RTAOC(I))/AIMAG(RTA(I))

4  PRINT 101, I, RTA(I), RTAOC(I), I, RATR, RATI
   IF (REAL(RTA(I))) 5,6,5
5  IF (AIMAG(RTA(I))) 7,6,7
6  PRINT 102
7  CONTINUE
30 PROAOC = 1.
   DO 8 I=1,N
8  PROAOC = PROAOC*RTAOC(I)
   IF (IC - 1) 31,32,31
31 PROA = 1.
   DO 9 I=1,N
9  PROA = PROA*RTA(I)
   PRINT 103, PROAOC, PROA
   IF (PROA) 10,11,10
10 RATP = PROAOC/PROA
   PRINT 104, RATP
11 PRINT 105
   PUNCH 108, NCASE, NCASEB, PROAOC
   WRITE (MO,108) NCASE, NCASEB, PROAOC
   IF (IC - 3) 35,33,35
35 PUNCH 106, PROA
33 RETURN
32 RELPP = (PROAOC - PROAR)/PROAP
   PRINT 107, PROAOC, PROAR, RELPP
   IF (ABS(RELPP) - 1.E-10) 33,33,34
34 STOP
100 FORMAT (1H1,10X,*COMPARISON OF EIGENVALUES OF ORIGINAL MATRIX A
1AND OF OPTIMALLY CONTROLLED AOC* //13X,*EIGENVALUES OF A*,4X,*
2EIGENVALUES OF AOC*,9X,*RATIOS OF ROOT(AOC)/ROOT(A)* //12X,*REAL
3PART IMAG PART REAL PART IMAG PART*,7X,*RATIO OF REALS RATIO O
4F IMAGS*)
101 FORMAT (2X,*ROOT (*,I2,*) *,E9.2,*,*,E9.2,3X,E9.2,*,*,E9.2,3X,*(*,
1I2,*) *,E9.1,9X,E8.1)
102 FORMAT (1H+,31X,* (2. HERE MEANS INDEF. OR INF. RATIO)*)
103 FORMAT (/11X,*PRODUCT OF EIGENVALUES OF AOC =*,E10.2/1X,
1 *PRODUCT OF EIGENVALUES OF A =*,E10.2)
104 FORMAT (1X,*RATIO OF PRODUCTS =*,E10.2/)
105 FORMAT (1H0,*(REAL PARTS ONLY CONSIDERED)*//)
106 FORMAT (E20.13)
107 FORMAT (/10X,*PRODUCT OF COMPUTED EIGENVALUES OF A =*,E20.13/
1 10X,*READ-IN PRODUCT OF EIGENVALUES OF A =*,E20.13/
2 10X,*FRACTIONAL DIFFERENCE =*,E21.13)
108 FORMAT (*CASE*,I6,* (*,I2,* OF FORCK1)*,14X,*PROAOC = *,E20.13)
END

```

```

C
C
C
SUBROUTINE INCOMA(A,IR,C,IRPIR,N,M,NPM,NPN,NPNPM,NERROR,INDEX,IWP)F303CH 3
CERN F303 INCOMA FORT F303 2
C F303 4
C*-*- INVERSION OF A COMPLEX MATRIX, AND SET OF SOLUTIONS F303 5
C A= GIVEN COMPLEX ARRAY AND RETURNED INV(A) NEW 6A
C IR= COLUMN SIZE (I.E. ROW DIMENSION) OF A F303 7
C C= AUXILIARY REAL ARRAY, NEEDED FOR INVERSION F303 8
C IRPIR= COLUMN SIZE OF C, TWICE IR SAY. IN FACT, 2*N IS SUFFICIENT. F303 9
C N= ORDER OF THE MATRIX TO BE INVERTED F303 10
C M= NUMBER OF SETS OF SOLUTIONS = NUMBER OF SETS OF GIVEN CONSTANTS F303 11
C NPM=N+M F303 12
C NPN=N+M F303 13
C NPNPM=N+N+M F303 14
C NERROR= AS FOR MATIN1. IF NERROR=0 AT THE OUTPUT, PROBLEM IS SOLVED F303 15
C INDEX AS FOR MATIN1, FOR BOOK-KEEPING F303 16
C F303 17

```

COMPLEX	A(IR,NPM)	F303	18
	DIMENSION C(IRPIR,NPNM),INDEX(NPM)	F303	19
C	COMPLEX MATRIX A(1...N,1...N) TRANSFORMED INTO REAL MATRIX	F303	20
C	C(1...N+N,1...N+N)	F303	21
C	M SETS OF COMPLEX CONSTANTS A(1...N,N+1...N+M) TRANSFORMED	F303	22
C	INTO REAL MATRIX C(1...N+N,N+N+1...N+N+M)	F303	23
C		F303	24
	NP1=N+1	F303	25
	DO 1 I=1,N	F303	26
	DO 2 J=1,N	F303	27
	CR=PEAL (A(I,J))	F303	28
	CI=AIMAG(A(I,J))	F303	29
	C(I ,J)=CR	F303	30
	C(I+N,J+N)=CR	F303	31
	C(I+N,J)=-CI	F303	32
	2 C(I ,J+N)=-CI	F303	33
C	IF M=0, ONLY THE INVERSE IS COMPUTED	F303	34
	IF(M.EQ.0) GO TO 1	F303	35
	DO 5 J=NP1,NPM	NEW	36A
	C(I ,J+N)=PEAL (A(I,J))	F303	37
	5 C(I+N,J+N)=AIMAG(A(I,J))	NEW	38A
	1 CONTINUE	F303	39
C		F303	40
C	CALL MATIN1 FOR INVERSION AND M SETS OF SOLUTIONS IN REAL TERMS	F303	41
C		F303	42
	CALL MATIN1(C,IRPIR,NPN,0,M,INDEX,NERROR,DETERM)	F303	43
C		F303	44
C	INVERTED MATRIX C(1...N+N,1...N+N) , IN REAL TERMS	F303	45
C	M SETS OF SOLUTIONS C(1...N+N,N+N...N+N+M) , IN REAL TERMS	F303	46
C	FOR OUTPUT, RECOMBINE INTO COMPLEX FORM	F303	47
C	INVERTED MATRIX A(1...N,1...N)	F303	48
C	M SETS OF SOLUTIONS A(1...N,N+1...N+M)	F303	49
C		F303	50
	DO 3 I=1,N	F303	51
	DO 4 J=1,N	F303	52
	4 A(I,J)=CMPLX(C(I,J),C(I+N,J))	F303	53
	IF(M.EQ.0) GO TO 3	F303	54
	DO 6 J=NP1,NPM	NEW	55A
	6 A(I,J)=CMPLX(C(I,J+N),C(I+N,J+N))	NEW	56A
	3 CONTINUE	F303	57
	RETURN	F303	58
	END	F303	59

C				
C				
C				
	IDENT	MATIN1	F100	3
	ENTRY	MATIN1	F100	4
	EXT	OUTPIC	F100	5
PAPAM	ASS	2	F100	6
+	VED	36/0HMATIN1,24/3	F100	7
MATIN1	DATA	0	F100	8
	SA1	03 LOAD N	F100	9
	SA7	1	F100	10
	MX5	58	F100	11
	SA4	X1	F100	12
	BX0	X5*X1	F100	13
	GE	07,04,NE01 N.LE.1	F100	14
	NZ	X0,N7LP1	F100	15
	SA3	TSW1	F100	16
	SA4	ASW1	F100	17
	BX6	X3	F100	18
	LX7	X4,30	F100	19
	SA6	TSWICH1	F100	20
	JP	LP2	F100	21

NZLP1	SA3	TSW2		F100	22
	SA4	ASW2		F100	23
	IX6	X3+X0		F100	24
	BX7	X4+X0		F100	25
	SA6	TSWTOH1		F100	26
LP2	SA7	ASWTOH		F100	27
	BX0	-X5*X1	NUMBER OF LOOPS IN LOOP2	F100	28
	NZ	X0,NZLP2		F100	29
	SA3	TSW3		F100	30
	SA4	ASW1		F100	31
	BX6	X3		F100	32
	LX7	X4,B0		F100	33
	SA6	TSWTOH2		F100	34
	JP	LP3		F100	35
N7LP2	SA3	TSW4		F100	36
	SA4	ASW2		F100	37
	IX6	X3+X0		F100	38
	BX7	X4+X0		F100	39
	SA6	TSWTOH2		F100	40
LP3	SA7	ASWTOH		F100	41
	SB4	X1		F100	42
	BX6	X1		F100	43
	SA1	BIGLOOP		F100	44
	SB1	B1-B7		F100	45
	SX7	B1		F100	46
	SA6	N		F100	47
	SA7	PIVTCO		F100	48
	SA3	B2	GET ROW DIMENSION	F100	49
	MX0	42		F100	50
	BX1	X0*X1		F100	51
	SA4	SWAP-1		F100	52
	BX5	X0*X4		F100	53
	SA2	B5	GET M	F100	54
	IX6	X1+X3		F100	55
	SB2	B7		F100	56
	SB3	B0		F100	57
	BX7	X5+X3		F100	58
	SA5	OUT+1		F100	59
	SB5	Y2+B4	N+M	F100	60
	BX4	X0*X5		F100	61
	SA6	A1		F100	62
	SA7	A4		F100	63
	LX3	30		F100	64
	SX1	B5-B7		F100	65
	SA2	SWAP-2		F100	66
	BX5	X0*X2		F100	67
	IX6	X4+X1		F100	68
	BX7	X5+X1		F100	69
	SA6	A5		F100	70
	NO			F100	71
	SA7	A2		F100	72
	SA4	INDEX1	TO PLUG START OF ARRAY INDEX-1	F100	73
	SA5	SHUFFLE		F100	74
	SA2	TLOOP+1		F100	75
	BX4	X0*X4		F100	76
	SX6	B6-B7		F100	77
	BX5	X0*X5		F100	78
	IX7	X4+X6		F100	79
	LX0	30		F100	80
	BX6	X6+X5		F100	81
	SA7	A4		F100	82
	BX2	X6*X2		F100	83
	SA5	A5		F100	84
	NO			F100	85

SA1	=1.0		F100	86
RX7	X1		F100	87
IX6	X2+X3		F100	88
SA7	DETER	INITIALIZE DETERMINANT	F100	89
SA6	TL00P+1		F100	90
*			F100	91
* BIGLOOP TO TRANSFORM THE GIVEN SYSTEM			F100	92
*			F100	93
*			F100	94
* GET NEXT PIVOTELEMENT AND TRANSFORM ITS COLUMN			F100	95
*			F100	96
* B1 = A(0), B2 = 1, B3 = MAININDEX, B4 = ORDER OF MATRIX			F100	97
*			F100	98
*			F100	99
BIGLOOP SA2	PIVTOP	STARTADDRESS - 1 OF CURRENT PIVOT COLUMN	F100	100
SX3	0	LOGGED IN ROW DIMENSION OF A	F100	101
SX1	B4-B3	NUMBER OF ELEMENTS IN CURRENT PIVOTSUBCOL.	F100	102
SR5	X2	KEEP THE TOP ADDRESS - 1	F100	103
IX7	X3+X2	NEXT TOPADDRESS - 1	F100	104
MX4	57		F100	105
SX1	X1+B2	INCREMENT BY 1	F100	106
RX6	-X4*X1	NUMBER OF LOOPS IN PLOOP2	F100	107
SA7	A2		F100	108
AX1	3		F100	109
SA5	PSWTC41		F100	110
LX1	2	NUMBER OF LOOPS + 1 IN PLOOP1	F100	111
IX6	X6+X5		F100	112
SR6	X1-1		F100	113
SX0	30		F100	114
LX6	32		F100	115
*			F100	116
* SEARCH NEXT PIVOT ELEMENT			F100	117
*			F100	118
SA6	PSWITCH	STORE THE LOOPINDEX FOR PLOOP2	F100	119
SA1	X2+B3	FORCE A STORE LOAD CONFLICT. DUMMY LOAD TO	F100	120
SA5	A1+B2	SET A1, GOOD LOAD IN X5	F100	121
SR7	59		F100	122
GE	B6,B6,PSWITCH	LESS THAN 8 ELEMENTS IN SUBCOLUMN	F100	123
MX4	60	TO FORCE BOTH TESTS TO FAIL FIRST TIME	F100	124
RX2	X4		F100	125
PLOOP1 AX3	X5,B7	EXTEND SIGN	F100	126
RX6	X3-X5	ABSOLUTE VALUE	F100	127
SA1	A1+2	LOAD NEXT ELEMENT OF COLUMN	F100	128
* PL	X4,*+1		F100	129
RX0	X2	KEEP THE ABSOLUTE VALUE	F100	130
SA0	A5-B2	KEEP THE ADDRESS	F100	131
FX4	X0-X6		F100	132
AX3	X1,B7		F100	133
SA5	A5+2		F100	134
* PL	X4,*+1		F100	135
LX0	X5,B0		F100	136
SA0	A1-B2		F100	137
SR6	B5-B2	REDUCE LOOPINDEX	F100	138
RX2	X3-X1		F100	139
FX4	X0-X2		F100	140
NO			F100	141
N7	B6,PLOOP1		F100	142
AX3	X5,B7	HANDLE LAST TWO ELEMENTS	F100	143
RX6	X3-X5		F100	144
* PL	X4,*+1		F100	145
RX0	X2		F100	146
SA0	A1		F100	147
FX4	X0-X6		F100	148
SA1	A5	TO SET A1 FOR NEXT LOOP	F100	149

+	PL	X4,*+1		F100 150
	SA9	A5		F100 151
	RX0	X6		F100 152
PSWITCH	SB6	R6+0	PLUGGED IN LOOPINDEX	F100 153
	ZR	R6,POUT		F100 154
LOOP2	SA1	A1+R2	LOAD ONE ELEMENT AT A TIME	F100 155
	AX3	X1,R7		F100 156
	RX5	X3-X1		F100 157
	FX4	X0-X5		F100 158
+	PL	X4,*+1		F100 159
	SA0	A1		F100 160
	RX0	X5		F100 161
	SB6	R6-32		F100 162
	NO			F100 163
	N7	R6,LOOP2		F100 164
POUT	SX6	AC-R5	GET ROW IDENTIFIER INTO X6	F100 165
	SA1	A0	RELOAD THE PIVOT VALUE	F100 166
	ZR	X0,EPROP	PIVOT ELEMENT IS ZERO	F100 167
	SA5	ONE	LOAD (-1.0)	F100 168
	SA4	DETER		F100 169
	FX0	X5/X1		F100 170
	SR3	B3+32	INCREMENT MAIN INDEX	F100 171
	FX7	X1*X4		F100 172
	SA0	X6	KEEP ROW IDENTIFIER ALSO IN AC	F100 173
INDEX1	SA7	A4	STORE DETER	F100 174
	SB5	R5+32	NOW START ADDRESS OF PIVOT COLUMN	F100 175
	SA6	B3+0	PLUGGED ADDRESS-1 OF ARRAY INDEX	F100 176
*				F100 177
* TRANSFORM THE PIVOT COLUMN				F100 178
*				F100 179
TSWCH1	SA2	R5	FIRST ELEMENT OF PIVOT COLUMN	F100 180
	SA3	R5+32	ALTERNATE PLUG SA3 R5-32	F100 181
	SB6	0	PLUGGED LOOPINDEX EQ TSWCH2	F100 182
	FX4	X2*X0		F100 183
	FX5	X3*X0		F100 184
	SR7	R2+32	R7 = 2	F100 185
TRANSP1	SA2	A2+R7	LOAD NEXT PAIR	F100 186
	RX6	X4		F100 187
	SA3	A3+R7		F100 188
	LX7	X5,R0		F100 189
	FX4	X2*X0		F100 190
	SA6	A2-R7		F100 191
	SB6	R6-R7		F100 192
	FX5	X3*X0		F100 193
	SA7	A3-R7		F100 194
	NE	R7,R6,TRANSP1		F100 195
	RX6	X4		F100 196
	LX7	X5,R0		F100 197
	SA6	A2		F100 198
	NO			F100 199
	SA7	A3		F100 200
TSWCH2	SR7	R0		F100 201
	NO		ALTERNATE JP TOUT	F100 202
	SR6	0	PLUGGED LOOPINDEX FOR TRANSP2	F100 203
TRANSP2	SA3	A3+32	LOAD AND HANDLE ONE ELEMENT AT A TIME	F100 204
	FX6	X0*X3		F100 205
	SR6	R6-32		F100 206
	NO			F100 207
+	NO			F100 208
	SA6	A3		F100 209
	NZ	R6,TRANSP2		F100 210
TOUT	RX7	-X0		F100 211
	LX6	X7,R0		F100 212
	SR6	A0	ROW IDENTIFIER	F100 213

	SB4	R0		F100 214
	SA7	A1	STORE THE PIVOT INTO THE MATRIX	F100 215
	MX0	S9		F100 216
	SA6	PIVOT	KEEP THE PIVOT VALUE FOR LATER USE	F100 217
	SA1	R3+R1	FIRST ELEMENT OF PIVOT ROW	F100 218
	SA0	R1+R2	START OF MATRIX	F100 219
	E0	R6,R3,TLOOP		F100 220
	SA2	D1+R3	PIVOT NOT ON DIAGONAL	F100 221
	SA3	R1+R6		F100 222
	SX4	0	NUMBER OF COLUMNS IN SYSTEM	F100 223
	RX6	X3		F100 224
	LX7	X2,R0		F100 225
	SB6	0	ROW DIMENSION	F100 226
SWAP	SA6	A2		F100 227
	IX4	X0+X4	REDUCE LOOPINDEX	F100 228
	SA7	A3		F100 229
	NO			F100 230
	SA2	A2+R6		F100 231
	SA3	A3+R6		F100 232
	RX6	X3		F100 233
	LX7	X2,R0		F100 234
	N7	X4,SWAP		F100 235
	SA6	A2		F100 236
	SA7	A3		F100 237
	SA5	DETER	COMPLEMENT DETERMINAT	F100 238
	RX6	-X5		F100 239
	SA6	A5		F100 240
*				F100 241
* TRANSFORMATION LOOP FOR REST OF MATRIX				F100 242
*				F100 243
TLOOP	SA1	A1+R7	LOAD NEXT ELEMENT OF PIVOT ROW	F100 244
	SB4	R4+R2	REDUCE MAIN LOOPINDEX	F100 245
	RX7	X1		F100 246
	SA0	A0+R7	NEXT TRANS. COLUMN	F100 247
	SB7	0	PLUGGED ROW DIMENSION OF A	F100 248
	E0	R4,R3,OUT+1	SKIP THE PIVOT COLUMN	F100 249
	ZR	X1,OUT+1	PIVOTROW ELEMENT IS ZERO	F100 250
ASWITCH	SA3	R5	ALTERNATE SA5 R5-R2	F100 251
	SA5	R5+R2	SA4 AC-R2	F100 252
	SB6	0	LOOPIND. EQ ASWITCH	F100 253
	FX6	X3*X1	MULTIPLY FIRST PAIR OF PIVOTCOLUMN	F100 254
	FX7	X5*X1		F100 255
	SA2	A0		F100 256
	SA4	A0+R2	FIRST PAIR OF TRANS. COLUMN	F100 257
	FX0	X6+X2		F100 258
	RX5	X7		F100 259
	SB2	R2+R2	R2=2	F100 260
ALoop	NX6	X0,R0	FAST IN STACK LOOP TO TRANSFORM	F100 261
	FX4	X4+X5	A COLUMN AS FAR AS POSSIBLE	F100 262
	SA3	A3+R2	LOAD NEXT PAIR OF PIVOTCOLUMN	F100 263
	SA5	A5+R2		F100 264
	NX7	X4,R0		F100 265
	SA4	A4+R2	LOAD NEXT PAIR OF TRANS. COLUMN	F100 266
	SA2	A2+R2		F100 267
	SB6	R6-R2	REDUCE LOOPINDEX. CANNOT DO ANYTHING ANYHOW	F100 268
	FX3	X3*X1		F100 269
	SA6	A2-R2		F100 270
	FX5	X5*X1		F100 271
	FX0	X3+X2		F100 272
	SA7	A4-R2		F100 273
	NE	R2,R6,ALoop		F100 274
	FX4	X4+X5	HANDLE THE LAST PAIR	F100 275
	NX6	X0,R0		F100 276
	NX7	X4,R0	UNFORTUNATE CODE	F100 277

	SA6	A2		F100 278
	SA7	A4		F100 279
BSWITCH	S32	1	ALTERNATE SR2 1	F100 280
	S36	0	LOOPINDEX JP OUT	F100 281
ALoop	SA5	A5+B2	LOOPS MAXIMUM 3 TIMES	F100 282
	FX2	X5*X1	HANDLES THE REST OF THE COLUMN	F100 283
	SA4	A4+B2		F100 284
	FX3	X2+X4		F100 285
	MX6	X3,30		F100 286
	S36	B5-B2		F100 287
	NO			F100 288
	SA6	A4		F100 289
	N7	B5,ALoop		F100 290
OUT	SA2	PIVOT	TRANSFORM PIVOT ROW ELEMENT	F100 291
	FX7	X2*X1		F100 292
+	SA3	N		F100 293
	S36	0	NUMBER OF COLUMNS IN SYSTEM	F100 294
+	SA7	A1		F100 295
	GE	B6,B4,ALoop		F100 296
	S34	X3		F100 297
	NE	B4,B3,BIGLOOP		F100 298
SHUFFLE	SX2	B7	X2=0IM	F100 299
	MX4	59		F100 300
	S36	0	START OF ARRAY INDEX-1	F100 301
	S35	B4-B2	B5=N-1	F100 302
	PX0	X2		F100 303
	S31	B1+B2		F100 304
LOOP14	SA1	B6+B5	INDEX(MAIN)	F100 305
	SA3	N		F100 306
	IX5	X1+X4	LPIV-1	F100 307
	SX2	B5-B2	MAIN-1	F100 308
	IX6	X5-X2		F100 309
	PX7	X5,30		F100 310
	OX1	X7*X1		F100 311
	PX7	X2,30		F100 312
	OX5	X7*X0		F100 313
	ZR	X6,LOOP14		F100 314
	SA1	B1+X1		F100 315
	SA2	B1+X5		F100 316
	IX3	X3+X4	GET N - 1	F100 317
LOOP13	BX6	X1		F100 318
	LX7	X2,30		F100 319
	SA1	A1+B2		F100 320
	SA2	A2+B2		F100 321
	IX3	X3+X4		F100 322
+	SA6	A2-B2		F100 323
	SA7	A1-B2		F100 324
	PL	X3,LOOP13		F100 325
LOC14	S35	B5-B2		F100 326
	NO			F100 327
	N7	B5,LOOP14		F100 328
	SA1	DETER		F100 329
	SA2	MATIN1-7	ADDRESS OF NERROP	F100 330
	SA3	MATIN1-2	ADDRESS OF DETERM	F100 331
	OX6	X1		F100 332
	MX7	0		F100 333
	SA6	X3		F100 334
	SA7	X2		F100 335
	JP	MATIN1		F100 336
*				F100 337
*	HANDLE N=1			F100 338
*				F100 339
NE01	GE	B0,B4,MATIN1	N,LE,0	F100 340
	SA1	B1	N=1	F100 341

F100 406

	SR6	0	
TSW3	SR7	R0	F100 407
	JP	TOUT	F100 408
TSW4	SR7	R0	F100 409
	NO		F100 410
	SR6	0	F100 411
BSW1	SR2	1	F100 412
	JP	OUT	F100 413
BSW2	SR2	1	F100 414
	SR6	0	F100 415
PSWCH1	SR	B6,POUT	F100 416
	SR6	R6+0	F100 417
*			F100 418
*	LOCATIONS FOR INTERNAL USE		F100 419
*			F100 420
ONE	DATA	-1.0	F100 421
DETER	DATA	0	F100 422
PIVOT	DATA	0	F100 423
N	DATA	0	F100 424
PIVTOP	DATA	0	F100 425
	END		F100 426

APPENDIX D.

Listing of the more important routines in SIMULA

(Routines listed in APPENDIX C are not repeated).

SIMULA

DCCOMP

FILTER

EXCITED

EXTRA3

CONSTR1

PLOTS

GPH

SCALPL1

STORE

LISTVS

COMPCR

WRRMX

INVCOM

PROGRAM SIMULA (INPUT,OUTPUT,PERMDTA,TAPES=INPUT,
1 TAPE6=OUTPUT,TAPE2=PERMDTA)

ORDER OF DATA IS IPQ,IQUICK,HT,IKOU,MP,KREN,IDC(SET)
NCI(T) (SET IF IDC=1)
KR,KOP (REGULA),XAX1R(MXEIV2X1,XAX1U),SSXTR,SSXU
(MXEV2Y1,NEWXU), B,A (MXEV2X1),PROAOC,COSTS (REG.)
KRFILT(SET IF REQUIRED),KOU(SET IF REQ.).
KR,KOP,PROAOC,COSTS, READ FROM TAPE MI IF REQUIRED
XAX1R,SSXTR,SSXU,B,A READ FROM TAPE MP IF REQUIRED

24.5.72 (L. TEPPER, CONTROL SYSTEMS SECTION, DEPARTMENT OF
COMPUTING AND CONTROL, IMPERIAL COLLEGE)

SIMULATION (COMPUTATION AND PLOTTING) OF TRANSIENTS FROM THE EXCITATION OF
STATE VARIABLES OR DISTURBANCE INPUTS

PART (I)

1. COMPUTE EIGENVALUES AND EIGENVECTORS OF A OR AOC
2. CHECK (USING COMPCOR ROUTINE) THAT

$A = W \cdot RTC \cdot V$ WHERE RTC $\equiv$ COMPLEX EIGENVALUES (DIAGONAL)
 W $\equiv$ COMPLEX EIGENVECTOR MATRIX
 V $\equiv$ INVERSE OF W

PART(II) CALCULATE TRANSIENTS

IN REGULA WE HAD

$$\begin{aligned} \dot{X} &= A \cdot X + B \cdot U, & U &= [U(1), \dots, U(10)]^T & (1) \\ Y &= C \cdot X & & & (2) \end{aligned}$$

BUT ONLY 2 INPUTS ARE USED FOR CONTROL, THE REMAINDER BEING POSSIBLE
DISTURBANCES. HENCE THE CORRECT REPRESENTATION IS

$$\begin{aligned} \dot{X} &= A \cdot X + B \cdot UC + B \cdot UD, & (3) \\ UC &= [U(5), U(6)]^T \\ UD &= [U(1), \dots, U(4), U(9), U(10), U(7), U(8)]^T \end{aligned}$$

AFTER SOLVING THE RICCATI EQUATION FOR THE OPTIMAL LINEAR REGULATOR

$$\dot{X}(T) = AOC \cdot X(T) + BOC \cdot UD \quad (4)$$

$$\begin{aligned} &***** \\ * UC(T) &= -KR \cdot X(T) - KDG \cdot UD * & (5) \\ &***** \end{aligned}$$

WHERE

$$\begin{aligned} KR(NC,N) &\equiv \text{OPTIMAL FEEDBACK GAIN MATRIX (PRODUCED BY REGULA)} \\ AOC(N,N) &\equiv \text{OPTIMAL STATE MATRIX} \\ &= A - B \cdot KR & (6) \end{aligned}$$

$$\begin{aligned} BOC(N,ND) &\equiv \text{OPTIMAL DRIVING MATRIX} \\ &= [I - B \cdot KR] \cdot [B \cdot B + B \cdot KDG] & (7) \end{aligned}$$

$$\begin{aligned} KDG(NC,ND) &\equiv \text{DISTURBANCE GAIN MATRIX (OVERALL)} \\ &= KDR \cdot B + KDG \cdot B - I \cdot KDG & (8) \end{aligned}$$

$KDR(NC,N)$ $\equiv$ DISTURBANCE GAIN MATRIX (FROM B) (PRODUCED BY REGULA)
 $KDG(NC,ND)$ $\equiv$ GAIN MATRIX FROM UD TO UCPAR (TO BE SET FOR TYPES OF
EXCITATION. MAKING IT GENERAL IS TEDIOUS AND USELESS)

UCPAR $\equiv$ STEADY STATE DEVIATION (DUE TO UD) FROM STEADY STATE.
 $= KDG \cdot UD$

C FURTHER ON BELOW, WE SHALL USE FOR SIMPLICITY THE TERM

$$C \quad KDU \equiv -KDG*UD \quad (8A)$$

C THE SOLUTION OF (3) FOR STEP INPUTS UC, UD IS

$$C \quad X(T) = [EXP(A*T)]*X(0) + [INT(0-T)[EXP(A*T1)]*DT1]*[BC*UC + BD*UD] \quad (9)$$

C THE SOLUTION OF (4) FOR STEP INPUT UD (AND HENCE UC) IS

$$C \quad X(T) = [EXP(AOC*T)]*X(0) + [INT(0-T)[EXP(AOC*T1)]*DT1]*BDG*UD \quad (10)$$

C USING EIGENVALUES/EIGENVECTORS $RTC(A)$, $W(A)$, $V(A)$, AND

C $RTC(AOC)$, $W(AOC)$, $V(AOC)$

C THE SOLUTION (9) IS (OMITTING THE BRACKETED A)

$$C \quad * X(T) = W[EXP(RTC*T)]*V*X(0) + W*S(T)*V*[BC*UC + BD*UD] * \quad (11)$$

C WHERE $S(T) = [RTC*INV*[EXP(RTC*T) - I]]$

C THE SOLUTION (10) IS (OMITTING THE BRACKETED AOC)

$$C \quad * X(T) = W[EXP(RTC*T)]*V*X(0) + W*S(T)*V*BDG*UD * \quad (12)$$

C NOTE... EQ. (5) FOR UC(T) IS USED ONLY IN CONJUNCTION WITH EQ. (12)

C ADDITIONAL PARAMETERS XTR (SEE EXTRA) ARE COMPUTED AS FUNCTIONS OF X
C STEPS FOR IMPLEMENTATION

C READ IN

- C 1. IPD (SET TO 0 FOR NO PRINTOUT OF SOME INPUT DATA (SEE BELOW))
- C IQUICK (SET TO 1 TO FIT CATEGORY J6(COC), EXC(5) NOT PLOTTED
- C MI (SET TO 5 FOR READING KP,..., COSTS FROM CARDS, TO 3 FROM TAPE).
- C IKOU (SET TO 1 IF A VALUE OF KOU DIFFERENT FROM THE COMPUTED ONE
- C IS TO BE SET, TO TEST ITS INFLUENCE)
- C MP (SET TO 5 FOR READING XAX1R,..., A FROM CARDS, TO 2 FROM TAPE)
- C KRED (SET TO 1 IF NOT ALL THE STATE IS FED BACK)
- C IDC (SET TO 1 IF D.C. COMPENSATION IS TO BE USED)
- C 1A.NCI(I) (STATE VARIABLES TO BE FORCED TO SETTLE AT 0.)
- C (SET IF IDC = 1)
- C 2. KR(NG,N) AND KDR(NG,N), PRODUCED BY REGULA. XAX1R(NAR,NL), BY
- C MXYEV2X1. CALL EXTRA2 (1,...) FOR THIS PURPOSE.
- C 3. LISTS OF STEADY STATE VALUES, NAMES AND UNITS (IN THAT ORDER), OF
- C XTP(NXTP), X(N), U(NU) (IN THAT ORDER), BY MXYEV2X1.
- C 4. R(N,NU), A(N,N), PRODUCED BY MXYEV2X1.
- C 5. PROAOC (PRODUCT OF E/VALUES OF AOC(REAL PARTS)), COSTS, BY REGULA
- C 6. KPFIIT FILTER PATTERN / SET IF
- C KDU FORCED FIXED CONTROL/ REQUIRED

C COMPUTE

- C 1. BREAK UP B(N,NU) INTO BC(N,NC) AND BD(N,ND), CALLING RORDRM
- C AND CAPVUP
- C 2. REORDER THE STEADY STATE VALUES, NAMES AND UNITS OF U(NU) AND
- C GENERATE THE CORRESPONDING ITEMS OF UC(NG), CALLING RORDUS.
- C 2A. BUILD $AOC = A - BC*KR$
- C 3. RTC(N), W(N,N), V(N,N)
- C 4. W*RTC*V. COMPARE WITH ORIGINAL MATRIX, CALLING COMPCP
- C 5. RTCIN(N), THE INVERSE OF RTC.
- C 6. SET TYPE OF EXCITATION GIVING A TYPE NUMBER AND CALLING EXCITE.
- C IN THIS ROUTINE X(0) AND/OR U (STEP INPUT) ARE SET.

C KCD IS SET FOR SPECIAL TYPES OF EXCITATION AND UCBAR COMPUTED..

C (UCBAR = KCD*UD). ALSO COMPUTE IN THIS ROUTINE...

C 6A. $VX = V * X(0)$ FOR ALL CASES

C 6B. $VBU = V * R * U$ FOR OPEN LOOP CASE (EQ. (11))

C 6C. $KDG, KDU = -KDG * UD, VBU = V * BDG * UD$ FOR OPT. CON. CASE (EQ. (12))

C 6D. READ KDU IF IKDU = 1

C 6E. RECOMPUTE KDU IF IDC = 1 WITH NEW FORMULA IN ROUTINE DCCOMP

C 7. UC(0), XTR(0) CALLING EXTRA2 (2,...)

C 8. PRINT X(0), U(0), XTR(0), UCBAR AND KDU.

C 9. SET TMAX(MAX. TIME), NIMP(MAX. NUMBER OF TIME INTERVALS) AND INITIAL TIME

C SET ALL ARRAYS OF STOPE TO 0 AND INITIALIZE, CALLING STORE (1,...).

C 10. SET X(RESPONSE TO UD) AND X1(RESPONSE TO X0) TO 0. PRINT TIME.

C 11. $EXP(RTC * T)$

C 12. $S = RTCIN * (EXP(RTC * T) - 1)$

C 13. IF U EXC., COMP. 2ND TERM OF EQ(11) OR (12). CHECK PEAL, PRINT IF

C REQUIRED (IPR.NE.0)

C 14. IF X EXC., COMP. 1ST TERM OF EQ(11) OR (12). CHECK REAL, PRINT IF

C REQUIRED (IPR.NE.0)

C 15. ADD BOTH TERMS OF (11) OR (12). PRINT IF REQUIRED (IPR.NE.0)

C 16. UC AND XTR. PRINT IF IPR.NE.0, CALLING EXTRA2 (3,...)

C 17. STORE THE RESPONSE FOR T, CALLING STOPE (3,...)

C 18. SET NEW TIME T AND GO TO (10).

C 19. PRINT FINAL VALUES OF X, U, XTR, UCBAR, KDU.

C 20. PRINT ALL VALUES IN MATRIX FORM (EACH COLUMN HEADED BY TIME T),

C IF KWPSM.NE.0

C 21. PLOT SELECTED PARAMETERS X, UC AND XTR. PRINT TABLE OF CONSTRAINTS

C AND PEAK VALUES. PRINT MATRIX $[KR(I,J) * X(J) PEAK]$, TO OBTAIN

C DOMINANT VALUES FOR FEEDBACK.

C 22. SET NEW TYPE OF EXCITATION AND GO TO (6).

C Y = C * X IS NOT USED NOW (THE DEFINITIONS AND STATEMENTS ARE COMMENTED)

C LIST OF SUBROUTINES

C CHDATA	CHECKS INPUT DATA (MUST BE CHANGED FOR DIFFERENT
C	CASES OF FBCK1 OR STEADY STATE IN MXEV2X1.
C KDUCHG	CHANGES COMPUTED KDU (FIXED CONTROL ADDED FOR
C	EXTERNAL DISTURBANCES). CALLED FROM EXCITE1
C EXCITE1	SETS INITIAL EXCITATION X(0) AND/OR STEP INPUT U.
C	COMPUTES VARIOUS VALUES (INVARIANT DURING SIMULATION)
C EXTRA2	COMPUTES, PRINTS PARAM. OTHER THAN X. READS NEG. DATA
C ROPDUS	REORDERS STEADY STATE VALUES, NAMES AND UNITS OF U
C CONSTR	SETS CONSTRAINTS AND TESTS VIOLATIONS, PRINTS TABLE
C PLOTS	PLOTS SELECTED PARAM. FROM X, XTR, UC. CALLS CONSTR,
C	SCALPL
C GPH	CALLED FROM PLOT FOR EACH LINE OF PRINTING (I.E. EACH TIME T)
C SCALPL	COMPUTES THE SCALE OF EACH PARAMETER AND TESTS CONSTR. VIOL.
C STOPE	STORES X, UC, XTR AND T FOR ALL T
C LISTVS	PRINTS INITIAL, FINAL SS VALUES OF X, U, XTR.
C	PRINTS UCBAR, KDU, AND CONSTRAINTS OF UC
C TPEVCP	CHANGES COMPLEX VECTOR INTO REAL IF IMAG PART IS NEGLIGIBLE
C COMPCR	COMPARES COMPL. AND REAL MATRIX (TO TEST IF $A = W * RTC * V$)
C WRRMX	WRITES REAL MATRIX COLUMNWISE (NO LIMIT ON DIMENSIONS)
C WRCMX	WRITES COMPLEX MX COLUMNWISE (NO LIMIT ON DIMENSIONS)
C WRRMXT	EXTENSION OF WRRMX, T HEADING EACH COLUMN, NAMES OF
C	ALL PARAMETERS ON EACH SUBMATRIX
C RCM2013	READS COMPLEX MATRIX (4E20.13)
C RRM2013	READS REAL MATRIX (4E20.13)
C RCV2013	READS COMPLEX VECTOR (4E20.13)
C PRV2013	READS REAL VECTOR (4E20.13)
C RIV8A10	READS INTEGER VECTOR (8A10)
C CARVUP	TRANSFERS SUBMATRIX FROM PARTITIONED MATRIX
C MULMXV	MULTIPLIES REAL MATRIX AND REAL VECTOR
C NULL	SETS THE NULL MATRIX

```

C  RORDRM      SWITCHES ROWS OR COLUMNS IN REAL MATRIX

C  NEGV      NEGATIVE OF REAL VECTOR
C  ADDV      ADDS TWO REAL VECTORS
C  NULLRV    SETS REAL VECTOR TO ZERO
C  MULCOM     PRODUCT OF TWO COMPLEX MATRICES
C  INVCOM     INVERTS COMPLEX MATRIX
C  MLMXOC     PROD. OF A(SQ.) AND D(DIAG.) COMP. MATR. (A*D OR D*A)
C  MULCHCV    PROD. OF COMPL. MATRIX AND COMPL. COL. VECTOR
C  MULCHPV    PROD. OF COMPL. MATRIX AND REAL COL. VECTOR, RESULT COMPLEX
C  MULDVC     PROD. OF COMPL. DIAGONAL MATRIX AND COMPLEX VECTOR
C  EIGVV1     COMPUTES , PRINTS EIGENVALUES, EIGENVECTORS OF A REAL SQUARE
C             MATRIX, USING FRANCIS. OR ALGORITHM FOR E-VALUES, WILKINSON.S
C             INVERSE ITERATION METHOD FOR E-VECTORS AND THE RAYLEIGH QUOT.
C             SCHEME FOR E-VALUES CORRECTION.
C             THIS IS A MODIFIED VERSION OF EIGSYS (VAN NESS/WEINER, NORTH
C             WESTERN UNIV., EVANSTON, ILL.) AND CALLS THE FOLLOWING
C  HESSEN, OREIS, QRT, ASOP, EIGVEC, DEISP, WRCHX
C  SLOPE1     A NUMERICAL TABLE
C  INTPOL     STRAIGHT INTERPOLATION

C  LIST OF DIMENSIONS AND ARRAYS

C  N          NUMBER OF STATE VARIABLES
C  NU          .. .. INPUTS
C  NC          .. .. CONTROL INPUTS
C  ND          .. .. DISTURBANCE INPUTS
C  NY          .. .. OUTPUTS
C  NXTR        .. .. ADDITIONAL VARIABLES XTR

C  A(N,N)      STATE MATRIX
C  R(N,NU)     DRIVING MATRIX
C  RTC(N)      COMPLEX EIGENVALUES OF A
C  W(N,N)      COMPLEX EIGENVECTORS MATRIX
C  V(N,N)      INVERSE OF W
C  AC(N,N)     COMPLEX MATRIX FOR PRODUCTS
C  PTR(N)      REAL PART OF E-VALUES OF A
C  PTI(N)      IMAG PART OF E-VALUES OF A
C  A1(N,N),A2(N,N),RW(N,4)  AUXILIARY REAL MATRICES FOR EIGVV1
C  R1(N),...,R6(N)  AUXILIARY REAL VECTORS FOR EIGVV1
C  OA(N,N)     AUXILIARY COMPLEX MATRIX FOR EIGVV1
C  C1(N),...,C4(N)  AUXILIARY COMPLEX VECTORS FOR EIGVV1
C  AD(N,N)     AUX. DOUBLE PRECISION MATRIX FOR EIGVV1
C  Q1(N),...,Q4(N)  AUX. DOUBLE PRECISION VECTRS FOR EIGVV1
C  IPOW(N,2)   AUX. INTEGER MATRIX FOR EIGVV1
C  RTCIN(N)    COMPLEX, [RTC]INV
C  EXPRTC(N)   COMPLEX, EXP(RTC*T)
C  S(N)        COMPLEX, [RTC]INV*[EXP(RTC*T) - I]
C  RU(N)       REAL, R*U
C  VDU(N)      COMPLEX, V*R*U
C  VXO(N)      COMPLEX, V*X(1)
C  CM1(N,N)    COMPLEX AUX. MATRIX
C  RC(N,NC),BD(N,ND),UC(NC),UD(ND)  SEE EQUATION (3)
C  X0(N)       INITIAL VALUES OF X
C  X1(N)       CURRENT VALUES OF X, FOR X EXCITATION ONLY
C  X(N)        CURRENT VALUES OF X, FOR U DISTURBANCE ONLY
C             ALSO CURRENT VALUES OF X, FOR SIMULTANEOUS X EXC. AND U DIST.
C  XTR(NXTR)   CURRENT VALUES OF XTR, THE ADDITIONAL VARIABLES
C  XS(N),US(NU),UCS(NC),XTRS(NXTR)  STEADY STATE VALUES OF X,U,UC,XTR
C  NAMEX(N),NAMEU(NU),NAMEUC(NC),NAMEXTR(NXTR)  NAMES OF X,U,UC,XTR
C  UNITX(N),UNITU(NU),UNITUC(NC),UNITXTR(NXTR)  UNITS OF X,U,UC,XTR
C  NAMXTRV(NXTR)  VARIABLE TYPE AND NUMBER AS DEFINED IN PROGRAM MXEV2X1
C  ICASE(NW)     NUMBER OF CASE BEING SIMULATED
C  XAR(NAR)      XTR VARIABLES DEFINED AS XA IN PROGRAM MXEV2X1
C  XAX1R(NAR,NL)  MATRIX IN EQ. XAR=XAX1R*X1, WHERE X1(NL) IS THE VECTOR OF THE

```

```

C          1ST NL COMPONENTS OF X

C  KR(NC,N)      MATRIX IN EQ. 10C=KR*X
C  XP(N,NLIN),UCP(NC,NLIM),XTRP(NXTR,NLIM),TP(NLIM)
C          ARRAYS FOR STORING X, UC, XTR, T FOR ALL T
C  NAMCOST(65)    NAMES OF PENALIZED VARIABLES AND THEIR COSTS WRITTEN
C          ON 13 CARDS IN FORMAT(5A10)
C  UGBAR(NC)      SETTLING-DOWN VALUES OF UC (DEVIATIONS FROM STEADY ST)
C          DUE TO PERMANENT DISTURBANCE UD
C  KOU(NC)        FIXED ADDED CONTROL, DUE TO PERMAN. EXT. DISTURB. UD
C  KFOR(NC)       FIXED ADDED CONTROL, USED IN CONJUNCTION WITH KOU,
C          TO TEST ITS INFLUENCE
C  KRFILT(NC,N)   FILTER FOR CASE(KRED=1), I.E. NOT ALL ST.V. FED BACK
C  KOR(NC,N)      MATRIX OBTAINED FROM REGULA TO COMPUTE KOU
C  KRG(NC,ND)     SEE EQ. (8)
C  ACD(NC,ND)     AUXILIARY MATRIX FOR ABOVE
C
C  REAL A(31,31)
C  COMPLEX RTC(31),W(31,31),V(31,31),AC(31,31)
C  INTEGER INAME(8)
C  REAL A1(31,31),A2(31,31),PTR(31),RTI(31),RW(31,4),R1(31),R2(31),
1  R3(31),R4(31),R5(31),R6(31)
C  INTEGER IROW(31,2)
C  COMPLEX OA(31,31),C1(31),C2(31),C3(31),C4(31)
C  DOUBLE AO(31,31),O1(31),O2(31),O3(31),O4(31)
C  EQUIVALENCE (AC,OA,AD),(C1,O1),(C2,O2),(C3,O3),(C4,O4)
C
C  DEFINITIONS AND DATA FOR PART(II)
C
C  COMPLEX RTCIN(31),EXPTC(31),S(31),VBU(31),VXO(31),CN1(31,31)
C  REAL B(31,10),XO(31),UC(2),U1(8),JC(31,2),RO(31,8),BU(31),X(31),
1  Y1(31),US(10),XS(31),UCS(2),UGBAR(2),KOU(2),KFOR(2)
C  REAL C(10,31),Y(10),Y1(10)      ***FOR THE TIME BEING Y IS NOT USED***
C  INTEGER ICASE(3),NAMEX(31),NAMEU(10),UNITX(31),UNITU(10),NAMEUC(2)
C  INTEGER UNITUC(2),NAMCOST(65),DATCASE(2),KRFILT(2,31)
C  EQUIVALENCE (B,RO),(BU,R1),(X,R2),(X1,R3),(XO,R4)
C  EQUIVALENCE (VBU,C1),(VXO,C2)
C  REAL URUC(2),LRUC(2)
C  DEFINITIONS FOR EXTPAR AND EXCITE1
C  REAL XTR(14),XTRS(14),KP(2,31),XAR(7),XAX1R(7,27),KOR(2,31),
1  RC(2,2),KCD(2,8),KDG(2,8),ACD(2,8),PC1(2,2),AKOU(2)
C  KCD,KDG,ACD,RC,PC1,AKOU ARE AUXILIARY FOR ROUTINE EXCITED
C  EQUIVALENCE (A1,KCD),(A1(1,2),KDG),(A1(1,3),ACD)
C  INTEGER NAMXTR(14),NAMXTRV(14),UNTXTR(14)
C  INTEGER NOT(2)
C  DEFINITIONS FOR STORE
C  REAL XP(31,80),UCP(2,80),XTRP(14,80),TP(80)
C  EQUIVALENCE (XP,A1),(XP(1,32),A2),(XP(1,63),A),(A(1,19),XTRP)
C
C  DATA ICASE/30H      OF REGULA FOR CASE 3/
C  DATA ICASE/30H CASE 7 OF FORCK1 (MXEV2X1) /
C  IA = 1 - OPEN LOOP CASE (EQ.(11))
C  IA = 2 - CLOSED LOOP CASE (EQ.(12))
C  IA = 2
C
C  IPR = 0 - INDIVIDUAL VECTORS FOR EACH TIME NOT PRINTED
C  IPD = 0 - XAX1R,B,A,ACD,RO,PC NOT PRINTED
C  IQUICK = 1 - EXC(5) NOT PLOTTED, KR,KOR NOT PRINTED (TO FIT J6 ON CDC)
C  MI = 5 - KR,KOR,PROACD,COSTS, READ FROM CARDS.
C  MI = 3 - KR,KOR,PROACD,COSTS, READ FROM TAPE 3
C  IKOU = 1 - KOU TO BE READ IN, TO TEST INFLUENCES OF ADDED CONTROL.
C          THIS TEST MAY BE DONE ONLY AFTER HAVING CORRECT SIMULATION.
C  MP = 5 - XAX1R,SSXTR,SSXU,B,A, READ FROM CARDS
C  MP = 2 - XAX1R,SSXTR,SSXU,B,A, READ FROM TAPE 2
C  KRED = 1 - NOT ALL STATE VARIABLES FED BACK, READ IN FILTER KRFILT

```

C

```

READ 120, IPD, IQUICK, MI, IKOU, MP, KRED, IDC
IF (IQUICK.EQ.1) IPD = 0
IPR = 0
N = 30
NU = 10
NC = 2
ND = 8
NW = 3
NXTP = 13
NAR = 7
NL = 27
NOYD = 8
IF (IDC.NE.1) GO TO 46
READ 125, (NCI(I), I=1, NC)

```

C

C

C

C

```

IF MP = 5, A CARD MUST BE INSERTED (AFTER THE ..KEYS.. CARD),
IN FORMAT 122, WITH THE CASE (I) OF FDBCK1(MXEV2X1) AND
THE P.C. LEVEL OF POWER

```

```

46 READ(MP,122) DATCASE
PRINT 123, DATCASE
PRINT 124, IPD, IQUICK, MI, IKOU, MP, KRED, IDC
IF (IDC.NE.1) GO TO 45
PRINT 126, ((I,NCI(I)), I=1, NC)
45 PRINT 127

```

C

C

C

```

READ IN KR(MC,N), KOP(NC,N), XAX1R(NAR,NL), XTRS(NXTR),
NAMXTRV(NXTR), NAMXTR(NXTR), UNTXTR(NXTP).
CALL EXTPA3 (1, INAME, R, N, NU, X, XS, US, NC, ND, UC, UD, NXTP, XTR, XTRS,
1 NAMXTR, NAMXTRV, UNTXTR, NAMEU, KR, IA, IPR, NAR, NL, XAR, XAX1R, ICASE, NW,
2 IPD, KOR, KOU, MI, MP, IQUICK)

```

C

C

C

```

READ IN XS(N), NAMEX(N), UNITX(N), US(NU), NAMEU(NU), UNITU(NU)

READ (MP,101) INAME
CALL IPV2013 (XS, N, MP)
CALL IIV8A10 (NAMEX, N, MP)
CALL IIV8A10 (UNITX, N, MP)
PRINT 102, INAME
READ (MP,101) INAME
CALL IPV2013 (US, NU, MP)
CALL IIV8A10 (NAMEU, NU, MP)
CALL IIV8A10 (UNITU, NU, MP)
PRINT 102, INAME

```

C

C

C

```

READ IN B(N,NU), A(N,N).

```

```

READ (MP,101) INAME
CALL IRM2013 (B, N, NU, MP)
PRINT 102, INAME
IF (IPD) 29,30,29
29 CALL WPPMX (B, N, NU, 30H9 (BEFORE REORDERING) )
30 READ (MP,101) INAME
CALL IRM2013 (A, N, N, MP)
PRINT 102, INAME
IF (IPD) 31,32,31
31 CALL WPPMY (A, N, N, 30HSTATE (INPUT) )

```

C

C

C

```

READ IN PROAOC (1 CARD), COSTS (13 CARDS).

```

```

32 IF (IA - 1) 35,36,35
35 READ (MI,115) PROAOC
NNAMOST = 5 * (3 + NC + NOYD)

```

```
READ (MI,114) (NAMCOST(I),I=1,NNAMCOST)
```

```
36 IF (EOF,MI) 1,2
```

```
1 PRINT 103
```

```
STOP
```

```
2 PRINT 104
```

```
C
```

```
C CHECK IF A REDUCED SET OF FEEDBACK VARIABLES IS REQUIRED INSTEAD
```

```
C OF THE FULL SET. IF SO, CHANGE MATRIX KR
```

```
C
```

```
IF (KRED - 1) 42,41,42
```

```
41 CALL FILTER (NC,N,KR,KRFILT,NAMEX,1)
```

```
C
```

```
42 IS = 10HAGC(30,30)
```

```
IS = 10HA (30,30)
```

```
FOR A
```

```
C
```

```
C REORDER MATRIX R (CHANGE COLUMNS 5,6 FOR 9,10). 7PEAK UP 3 INTO R0
```

```
C AND R0. REORDER US (CHANGE 5,6 FOR 9,10)
```

```
C
```

```
CALL RORDRM (1,3,N,NU,5,9)
```

```
CALL RORDRM (1,3,N,NU,6,10)
```

```
CALL CAPVUP (3,30,N,NU,N,NO,1,1)
```

```
CALL CARVUP (3,30,N,NU,N,NO,1,9)
```

```
IF (IPD) 33,34,33
```

```
33 CALL WRRMX (R0,N,NO,30HRD (DISTURBANCE INPUT MATRIX) )
```

```
CALL WRRMX (30,N,NO,30HRC (CONTROL INPUTS MATRIX) )
```

```
34 CALL RORDUS (NU,US,NAMEL,UNITU,NAMEUC,NC,NO,UCS,UNITUC)
```

```
CHECK UC
```

```
C
```

```
THE FOLLOWING STATEMENT CHECKS DATA ELEMENTS.
```

```
C
```

```
THE SUBROUTINE CHDATA IS PARTICULAR TO THIS SYSTEM AT
```

```
C
```

```
THE PRESENT STEADY STATE. IT MUST BE CHANGED FOR ANY CHGE IN INPUT.
```

```
C
```

```
THE PURPOSE IS TO FLAG A MISTAKE IN DATA RATHER THAN HAVING LONG
```

```
C
```

```
PRINTOUTS, FOR PRINTOUT ECONOMY S SAKE.
```

```
C
```

```
IF (IA - 1) 37,38,37
```

```
37 CALL CHDATA (BC,N,NC,NAMCOST,NNAMCST)
```

```
C
```

```
BUILD MATRIX AOC = A - BC*KR
```

```
C
```

```
CALL MULTI (BC,KR,A1,N,NC,N)
```

```
CALL SUBTR (A,A1,A,N,N)
```

```
IF (IPD.EQ.0) GO TO 38
```

```
CALL WRRMX (A,N,N,30H AOC = A - BC*KR )
```

```
38 CALL EIGVVL (3,0,0,0,0,N,A,A1,A2,RTR,PTI,RTC,W,R1,R2,R3,R4,RW,
```

```
1 IPD,N,R5,P6,04,C1,C2,C3,C4,A0,01,02,D3,04)
```

```
C
```

```
COMPARE PRAOC (PRODUCT OF COMPUTED EIGENVALUES OF AOC) WITH
```

```
C
```

```
PROAOC (READ-IN PRODUCT OF EIGENVALUES OF AOC) AND PRINT.
```

```
C
```

```
IF (IA - 1) 39,40,39
```

```
39 PRAOC = 1.
```

```
DO 27 I=1,N
```

```
27 PRAOC = PRAOC*RTC(I)
```

```
RELPP = (PRAOC - PROAOC)/PROAOC
```

```
PRINT 119, PRAOC,PROAOC,RELPP
```

```
IF (KRED.EQ.1) GO TO 40
```

```
IF (ABS(RELPP) - 1.E-5 ) 28,28,25
```

```
C
```

```
25 PRINT 118
```

```
STOP
```

```
28 PRINT 104
```

```
40 CALL INVCOM (N,V,N,IPANK)
```

```
PRINT 105, IS, IRANK
```

```
C
```

```
. CALL MLMXDC (V,CM1,RTC,N,2)
```

CALL MULCOM (W,CM1,AC,N,N,N)

CALL WRCMX (AC,N,N,10HA(CALCUL.),RTC,0)

CALL COMPR (AC,A,N,N,10HA(CALCUL.),10HA(ORIG.))

DO 3 I=1,N

3 RTCIN(I) = 1./RTC(I)

SET TYPE OF EXCITATION AND TEST IF KOU IS REQUIRED TO BE CHANGED
FOR TESTING PURPOSES (IF SO, READ IN KOU).

ITYPE = 5

IF (IKOU.EQ.1) ITYPE = 6

4 CALL NULLRV (X,N)

CALL EXCITED(X,UC,UD,N,NC,ND,IX0,IUC,IUD,IS,ICASE,NW,BC,BO,BU,R2,
1 V,VBU,VX0,IEXC,XS,US,NU,ITYPE,KDP,UCBAR,IA,KOU,KQD,
2 KQG,ACD,PC,KDQOR,IKOU,A,A1,A2,PC1,NCI,AKOU,IDC)

CALL EXTRAP(2,TNAME,A,N,NU,X0,XS,US,NC,ND,UC,UD,NXTR,XTR,XTRS,

1 NAMXTR,NAMXTRV,UNTXTR,NAMEU,KP,IA,IPP,NAP,NL,XAR,XAX1R,ICASE,NW,

2 IPD,KOR,KOU,MI,MP,IQUICK)

CALL PLOTS (N,NC,NXTR,2, XP,UCP,XTRP,TP,TLAST,ND,NU,XS,UCS,

1 XTRS,NAMEX,NAMEUC,NAMXTR,NAMXTRV,UNITX,UNITUC,UNTXTR,IEXC,IS,

2 ICASE,NW,IA,ITYPE,IQUICK,KP,UBUC,LBUC,1)

PRINT 111, IEXC,IS,ICASE,ITYPE

PRINT 112

IF (IUC.EQ.0) GO TO 26

PRINT 121

26 CALL LISTVS (N,NU,NC,ND,X0,UC,UD,XS,US,NAMEX,NAMEU,UNITX,UNITU,

1 2HX0,NXTR,XTR,XTRS,NAMXTR,NAMXTRV,UNTXTR,UCBAR,KOU,

2 UBUC,LBUC,UCS,NAMEUC,UNITUC)

SET TIME INTERVALS (LONGEST TIME CONSTANT OF A IS OF ORDER 2000 SECONDS)

T = 0.

TMAX = 10000.

NLIM = 80

CALL STORE (1,N,NC,NXTR,NLIM,X,UC,XTR,XP,UCP,XTRP,T,TP,TLAST,NT,

1 NTLAST)

PROCEED COMPUTING FOR EACH T (STEPS 5 TO 10)

6 CALL NULLRV (X,N)

CALL NULLRV (X1,N)

IF (IPR) 16,12,16

12 IF (T) 15,13,15

13 PRINT 116, T

GO TO 17

15 IF (IQUICK.EQ.1) GO TO 17

PRINT 117, T

GO TO 17

16 PRINT 106, T

17 DO 7 I=1,N

EXPRIC(I) = 0EXP(RTC(I)*T)

7 S(I) = RTCIN(I)*(EXPRIC(I) - 1.)

CHECK IF U, X OR BOTH ARE EXCITED

IF (IUC.EQ.0.AND.IUD.EQ.0) GO TO 8

CALL MULDVC (S,VBU,G3,N)

CALL MULOMDV (U,G3,G4,N,N)

C4 IS NOW X(T) AND SHOULD BE REAL. CHECK IF IT IS AND CONSTRUCT REAL.

IF (IPR) 19,20,19

19 PRINT 107

20 CALL TRFVCR (C4,X,N,10HX(T) ,NAMEX,IPR)

C

8 IF (IX0) 9,10,9

9 CALL MULDOVC (EXPTC,VX0,C3,N)

CALL MULCMGV (W,C3,C4,N,N)

C

C

C4 IS NOW X1(T) AND SHOULD BE REAL. CHECK IF IT IS AND CONSTRUCT REAL

C

IF (IPP) 21,22,21

21 PRINT 108

22 CALL TRFVCR (C4,X1,N,10HX1(T) ,NAMEX,IPR)

10 CALL ADDV (X,X1,X,N)

IF (IX0.EQ.0.OR.IUC.EQ.0.AND.IUD.EQ.0) GO TO 11

IF (IPP) 18,11,18

18 PRINT 109, T

PRINT 110, ((I,X(T),NAMEX(I)),I=1,N)

11 CALL EXTRA3 (3,INAME,2,N,NL,X,XS,UC,NC,ND,UC,UD,NXTR,XTR,XTRS,

1 NAMXTP,NAMXTRV,UNTXTP,NAMEU,KR,IA,IPR,NAR,NL,XAR,XAX12,ICASE,NW,

2 IPD,KOR,KOU,HI,MP,IQUICK)

CALL STORE (2,N,NC,NXTR,NLIM,X,UC,XTR,XP,UCP,XTRP,T,TP,TLAST,NT,

1 NTLAST)

IF (NTLAST.GE.NLIM) GO TO 14

TE = T + 1.E-10

IF(TE.GE.TMAX) GO TO 14

IF(TE.GE. 0.) DT = .01

IF(TE.GE. .3) DT = .05

IF(TE.GE. 1.) DT = 1.

IF(TE.GE. 10.) DT = 10.

IF(TE.GE. 100.) DT = 100.

IF(TE.GE.1000.) DT = 1000.

T = T + DT

GO TO 6

C

C

PRINT FINAL VALUES OF X(T)

C

14 PRINT 111, IEXC,IS,ICASE,ITYPE

PRINT 113, T

CALL LISTVS (N,NU,NC,ND,X,UC,UD,XS,US,NAMEX,NAMEU,UNITX,UNITU,

1 2H X,NXTR,XTR,XTRS,NAMXTR,NAMXTRV,UNTXTR,UCBAR,KOU,

2 UBUC,LBUC,UCS,NAMEUC,UNITUC)

C

C

PRINT ALL VALUES IN MATRIX FORM

C

KWRSIM = 0 → RESULTS OF SIMULATION NOT PRINTED (PLOTTED ONLY)

C

KWRSIM = 1

IF (KWRSIM.EQ.0) GO TO 24

CALL WRRMXT (XP, N, NTLAST,30HX(T) FROM T = 0. TO T = , ..

1 TLAST,TP,10H SECONDS ,NAMEX,0)

CALL WRRMXT (UCP, NC, NTLAST,30HUC(T) FROM T = 0. TO T = ,

1 TLAST,TP,10H SECONDS ,NAMEUC,0)

CALL WRRMXT (XTRP,NXTR,NTLAST,30HXTR(T) FROM T = 0. TO T = ,

1 TLAST,TP,10H SECONDS ,NAMXTR,0)

24 CALL PLOTS (N,NC,NXTR,NTLAST,XP,UCP,XTRP,TP,TLAST,ND,NU,XS,UCS,

1 XTRS,NAMEX,NAMEUC,NAMXTR,NAMXTRV,UNITX,UNITUC,UNTXTR,IEXC,IS, * WITH *

2 ICASE,NW,IA,ITYPE,IQUICK,KR,IBUC,LBUC,0)

PRINT 114,(NAMCOST(I),I=1,NNAMCOST)

IF (KPED - 1) 44,43,44

43 CALL FILTER (NC,N,KP,KRFILT,NAMEX,2)

44 IF (ITYPE - 6) 23,1000,23

23 ITYPE = 6

GO TO 4

1000 STOP

100 FORMAT (1H1)

```

101 FORMAT      (9A10)

102 FORMAT (10X,8A10)
103 FORMAT (10X,*END OF FILE CARD READ AS DATA*//10X,*ONE OR MORE DATA
1 CARDS MISSING*)
104 FORMAT (///10X,*DATA C.K.*/////)
105 FORMAT (///10X,*RANK OF EIGENVECTOR MATRIX OF *,A10,* IS*,I6//)
106 FORMAT (1H1, 9X,*T = *,F11.3,* SECONDS*)
107 FORMAT (//10X,*THE FOLLOWING VECTOR X(T) IS THE RESPONSE TO THE E
1XCITATION OF THE INPUTS U ONLY.*//10X,*IF THE INITIAL STATE EXCIT
2ATION IS NIL, THEN X(T) IS ALL THE RESPONSE THERE IS*)
108 FORMAT (///10X,*THE FOLLOWING VECTOR X1(T) IS THE RESPONSE TO T
1HE INITIAL STATE EXCITATION ONLY.*//10X,*IF THE INPUTS EXCITATION I
2S NIL, THEN X1(T) IS ALL THE RESPONSE THERE IS*//)
109 FORMAT (///10X,*FOLLOWING IS THE RESPONSE X(T) FOR T =*,F11.3,
1* SECONDS.*//10X,*BOTH THE INITIAL STATE AND THE INPUTS HAVE BEEN E
2XCITED*//)
110 FORMAT (15X,*X(T) (*,I2,*) = *,E11.4,5X,A10)
111 FORMAT (1H1, 9X,A10,*RESPONSE OF SYSTEM *,A10,*, *,3A10,*, EXC T
1YPE (*,I2,*)*//)
112 FORMAT (10X,*FOLLOWING IS THE LIST OF VALUES OF THE INITIAL STATE
1EXCITATION (X1) AND THE DISTURBANCE INPUTS (U1)***//10X,*THE VALU
2ES OF THE CONTROL INPUTS (UC) AND ADDITIONAL PARAMETERS ARE THE
3RESULTS OF THE CONTROL LAW FOR THIS*//10X,*CASE AND OF THE ALGEBRAI
40 RELATIONS IN THE BASIC MODEL, RESPECTIVELY. THEY DO NOT RE
5PRESENT EXCITATION*//10X,*OF THE SYSTEM (UNLESS IT IS SPECIFICALLY
6STATED TO BE SO)*//10X,*THE STEADY STATE VALUES ARE ALSO PRINTED,
7 AS ARE UCRAT = STEADY STATE DEVIATION OF UC FROM S.S. VALUE,*//
8 63X,43HIN THE PRESENCE OF U1 (UCRAT = KCG*U1)//
9 63X,*KDU = FIXED AMOUNT OF CONTROL TO BE ADDED TO THE FEEDBACK
AK*//
69X,46H(UC(T) = - KR*X(T) + KDU = - KR*X(T) - KCG*U1)//
961X,*CONSTRAINTS ON U1*//)
113 FORMAT (10X,*F I N A L V A L U E S (T = *,F11.3,* SECONDS)*//)
114 FORMAT (5A10)
115 FORMAT (50X,E20.13)
116 FORMAT (1H1,9X,*SIMULATION COMPUTING STARTS. TIMES T ARE PRINTED
1.*//10X,*ALL ELEMENTS SHOULD BE REAL*//10X,*THOSE ELEMENTS (IF ANY)
2 WHICH HAVE IMAG PART GT.1.E-7 AND (IMAG/REAL RATIO GT.1.E-4 OR RE
3AL ZERO) ARE PRINTED*//10X,*X(T) ARE ELEMENTS RESULTING FROM THE EX
4CITATION OF U ONLY*//10X,*X1(T) ARE ELEMENTS RESULTING FROM THE E
5XCITATION OF X ONLY*//29X,*REAL PART*,32 X,*IMAG PART*//10X,
6*T = *,F11.3,* SEC*)
117 FORMAT (10X,*T = *,F11.3,* SEC*)
118 FORMAT (///10X,* DATA WRONG. RUN AGAIN WITH IPD = 1*)
119 FORMAT (///10X,*PRODUCT OF COMPUTED EIGENVALUES OF AOC =*,E20.13/
1 10X,*READ-IN PRODUCT OF EIGENVALUES OF AOC =*,E20.13/
2 10X,*FRACTIONAL DIFFERENCE =*,E20.13/)
120 FORMAT (7I1)
121 FORMAT (10X,*WARNING . . . THE CONTROL INPUTS (UC) HAVE BEEN E
1XCITED*//)
122 FORMAT (A2,A3)
123 FORMAT (1H1///10X,*DATA INPUT TO SIMULA , OUTPUT FROM 4XEV2X1*/
1/10X,*CASE (*,A2,*) OF FORCK1 , POWER LEVEL = *,A3,* PG.*)
124 FORMAT (////////10X,*IPD = *,I1,* + (0 IMPLIES XAX1P,B,A,AOC,BD,
1BC NOT PRINT.)*//10X,*IQUICK = *,I1,* + (1 IMPLIES EXC(5) NOT
2T PLOTTED)*//10X,*HI = *,I1,* + (5 IMPLIES KP,KDR,PROAOC,COSTS
3, READ IN FROM CARDS, 3 FROM TAPE 3)*//10X,*IKDU = *,I1,* + (1 IM
4PLIES KDU TO BE READ IN, TO TEST INFLUENCE OF FIXED ADDED CONTROL)
5*//10X,*MP = *,I1,* + (5 IMPLIES XAX1P,SSXTR,SSXU,B,A, READ IN
6FROM CARDS, 2 FROM TAPE 2)*//10X,*KRED = *,I1,* + (1 IMPLIES NOT
7T ALL STATE VARIABLES FED BACK, HENCE FILTER PATTERN TO BE READ
8IN)*//10X,*IDC = *,I1,* + (1 IMPLIES DC COMPENSATION USED, RECOM
9PUTING KDU, SUCH THAT X(NGI(I)) = 0)*//)
125 FORMAT (8(3X,I2))
126 FORMAT (10X,*NGI(*,I1,*) = *,I2,* (SEE EXPLANATION FOR IDC =

```


11)*)

127 FORMAT (//////)
END

C
C
C

SUBROUTINE DCCOMP (N,NC,ND,A,A1,A2,RC,PC1,NCI,KOUOR,KOU,BC,BD,UD,
1 ACD,AKOU)

C

24.5.72 * COMPUTE D.C. GAIN KOU SUCH THAT $X(NCI(I)) = 0$.
THIS COMPENSATION IS NECESSARY WHEN THE FEEDBACK GAIN
MATRIX KR (EQ.(6) IN MAIN PROGRAM) IS FILTERED AND HENCE
NOT ALL THE STATES ARE FED BACK. THE KOU COMPUTED IN
ROUTINE EXCITE MUST BE RECOMPUTED AS FOLLOWS..
FROM EQS. (4,6,7,8) IN MAIN PROGRAM

C

$XDOT = [A - RC*KR]*X + RC*KOU + BD*UD = 0$
 $X = -[A - RC*KR]INV*[RC*KOU + BD*UD]$
 $Y = C*X = [X(NCI(1)) \dots X(NCI(NC))]T$
 $= -C*[A - RC*KR]INV*RC*KOU - C*[A - RC*KR]INV*BD*UD = 0$
 $KOU = -[C*[A - RC*KR]INV*RC]INV*C*[A - RC*KR]INV*BD*UD$

C

THE EXPRESSION $C*[.]INV$ IS EQUIVALENT TO A SUBMATRIX OF $[.]INV$
WHERE THE ROWS(I) ARE THE ROWS(NCI(I)) OF $[.]INV$

C

REAL A(N,N),A1(N,N),A2(NC,N),RC(NC,NC),PC1(NC,NC),BC(N,NC),
1 BD(N,ND),UD(ND),KOU(NC),KOUOR(NC),ACD(NC,ND),AKOU(NC)
INTEGER NCI(NC)

C

DO 1 I=1,NC
1 KOUOR(I) = KOU(I)
CALL INVERT (A,A1,N,IRANK)
PRINT 100, IRANK
DO 2 I=1,NC
NI = NCI(I)
DO 2 J=1,N
2 A2(I,J) = A1(NI,J)
CALL MULTI (A2,RC,RC,NC,N,NC)
CALL INVERT (RC,PC1,NC,IRANK)
PRINT 101, IRANK
CALL MULTI (A2,BD,ACD,NC,N,ND)
CALL MULTI (PC1,ACD,A2,NC,NC,ND)
CALL WRMX (A2,NC,ND,3CHKDC (UG(T) = -KR*X(T) -KDC*UD))
CALL MULMXV (A2,UD,KOU,NC,ND)
CALL NEGVR (KOU,NC)
PRINT 102, (NCI(I),I=1,NC)
PRINT 103, ((I,KOUOR(I),KOU(I)),I=1,NC)
RETURN

100 FORMAT (1H1,9X,37H RANK OF MATRIX ACD = A - RC*KR IS ,I2)
101 FORMAT (10X,35H RANK OF MATRIX C*[ACD]INV*BC IS ,I2///
102 FORMAT (10X,*W A R N I N G ...THIS RUN INCLUDES D.C. COMPENSATION
1 SUCH THAT*/(10X,*X(*,I2,*) = 0.*/)
103 FORMAT (10X,*HENCE THE VALUE FIRST COMPUTED FOR KOU (CONSTANT CON
1TOL ADDED FOR EXTERNAL DISTURBANCE) HAS BEEN RECOMPUTED AS*//
210X,43HKOU = - [C*[ACD]INV*RC]INV*C*[ACD]INV*BD*UD///
316X,*FIRST COMPUTED VALUE*,9X,*RECOMPUTED VALUE*//
4(10X,*KOU(*,I1,*) = *,E13.6,16X,E13.6))
END

C
C
C

SUBROUTINE FILTER (NC,N,KR,KRFILT,NAMEX,IF)

C
C
C

31.1.72

C REDUCES THE NUMBER OF VARIABLES FED BACK, SELECTING THOSE THAT YIELD

C ..DOMINANT VALUES.. (I.E. THOSE ELEMENTS OF THE ..WEIGHTED-GAIN..
C MATRIX $KP(I,J)*X(J)$ PEAK) WHICH ARE WITHIN THE UPPER DECADE. SEE
C SUBROUTINE PLOT)

```

C
      INTEGER KRFILT(NC,N),NAMEX(N)
      REAL    KR      (NC,N)
      GO TO (10,20), IF
10  DO 11 I=1,NC
      READ 100, (KRFILT(I,J),J=1,N)
11  PRINT 101, I,(KRFILT(I,J),J=1,N)
      IF (EOF,5) 12,13
12  PRINT 102
      STOP
13  PRINT 103
      DO 15 I=1,NC
      DO 15 J=1,N
      IF (KRFILT(I,J) - 1) 14,15,14
14  KR(I,J) = 0.
15  CONTINUE
      CALL WRRMX (K2,NC,N,30HKR FILTERED (RED. SET FB VAR.))
20  PRINT 104
      DO 25 J=1,N
      IF (KRFILT(1,J).NE.1.AND.KRFILT(2,J).NE.1) GO TO 21
      IF (KRFILT(1,J).EQ.1.AND.KRFILT(2,J).NE.1) GO TO 22
      IF (KRFILT(1,J).NE.1.AND.KRFILT(2,J).EQ.1) GO TO 23
      IF (KRFILT(1,J).EQ.1.AND.KRFILT(2,J).EQ.1) GO TO 24
21  PRINT 105, J,NAMEX(J)          $          GO TO 25
22  PRINT 106, J,NAMEX(J)          $          GO TO 25
23  PRINT 107, J,NAMEX(J)          $          GO TO 25
24  PRINT 108, J,NAMEX(J)          $          GO TO 25
25  CONTINUE
      RETURN
100 FORMAT (30I1)
101 FORMAT (10X,*FILTER FOR UC(*,T1,*).. ONES REPRESENT VARIABLES FED
      1 BACK.. *,30I1)
102 FORMAT (/10X,*END OF FILE CARD READ AS DATA*//10X,*ONE OR MORE DAT
      1A CARDS MISSING*////10X,*THIS RUN MUST READ FILTER KRFILT*)
103 FORMAT (////10X,*DATA O.K., FILTER READ IN*//)
104 FORMAT (////10X,*THIS SIMULATION WAS RUN ON A FILTER. ONLY THE ..AC
      1TIVE.. VARIABLES WERE FED BACK*///33X,*FOR UC(1)*,5X,*FOR UC(2)*//
      2/)
105 FORMAT (10X,*X(*,I2,*) = *,A10)
106 FORMAT (10X,*X(*,I2,*) = *,A10, 5X,*FB ACTIVE*)
107 FORMAT (10X,*X(*,I2,*) = *,A10,19X,*FB ACTIVE*)
108 FORMAT (10X,*X(*,I2,*) = *,A10, 5X,*FB ACTIVE*,5X,*FB ACTIVE*)
      END

```

```

C
C
C
      SUBROUTINE EXCITED(X0,UC,UD,N,NC,N0,IX0,IUC,IUD,IS,ICASE,NH,RC,RD,
      1 RU,R2,V,V0U,VX0,TEXC,XS,US,UU,ITYPE,KOR,UCBAR,
      2 IA,KOU,KC0,K06,AC0,RC,KOUOR,IKOU,A,A1,A2,RC1,
      3 NCI,AKOU,IDC)

```

C 24.5.72

```

C
C SET INITIAL EXCITATION X0 FOR STATE VARIABLES
C SET STEP INPUTS UC (CONTROL) AND UD (DISTURBANCE)
C SET KOR (GOOD FOR CERTAIN TYPES OF EXCITATION ONLY. TO MAKE IT
C GENERAL WOULD BE TEDIOUS AND USELESS).
C COMP. UCBAR = SETTLING-DOWN VALUE OF UC
C           = KOR*UD
C           VX0 = V*X(0) FOR ANY CASE (EQS. (11,12) IN MAIN PROGRAM).

```

```

C      VRU = V*B*U  FOR OPEN-LOOP CASE (EQ. (11))

C      KDG = DISTURBANCE GAIN MATRIX (OVERALL) FOR O.C. CASE (12)
C      = KOR*BD + (KOR*BC - I)*KCD
C      VRU = V*B*G*UD .  FOR O.C. CASE (EQ. 12)
C      KDU = - KDG*UD  FOR O.C. CASE (EQ. 12)
C      FOR IKDU = 1 → READ IN FORCED KDU TO TEST ITS INFLUENCE
C      FOR IUG = 1 → COMPUTE AGAIN KDU ACCORDING TO THE EQUATIONS IN
C      ROUTINE DCGOMP, FOR DG COMPENSATION WHEN FILTER USED
C
C      INTEGER ICASE(NN),UCI(NC)
C      COMPLEX V(N,N),VRU(N),VXD(N)
C      REAL XD(N),UC(NC),UD(NC),AC(N,NC),BD(N,NC),BU(N),R2(N),
1      XS(N),US(NC),KOP(NC,N),UCBAR(NC),KDU(NC),KCD(NC,NC),
2      KDG(NC,NC),ACD(NC,NC),PC(NC,NC),KDUOR(NC)
C      REAL A(N,N),A1(N,N),A2(NC,N),PC1(NC,NC),AKDU(NC)
C
C      IX0 = 0      E      IUC = 0      E      IUD = 0
C
C      CALL NULLPV (XD,N)
C      CALL NULLRV (UC,NC)
C      CALL NULLRV (UD,NC)
C      CALL NULLRV (UCBAR,NC)
C      CALL NULLRV (KDU, NC)
C      CALL NULL (KCD,NC,NC)
C
C      (RECALL TO RESET IX0,ETC TO 1 WHERE NECESSARY)
C      EXCITATION TYPE (1)      X(0) = UC = UD = 0.
C      EXCITATION TYPE (2)      X0(10) = 20.  PSTA,  UD(8) = .1*US(8)
C      EXCITATION TYPE (3)      UD(1) = 10 MILLINILES = 10E-5
C      EXCITATION TYPE (4)      UD(8) = .1*US(8)
C      EXCITATION TYPE (5)      X0(10) = 20.  PSIA
C      EXCITATION TYPE (6)      UD(8) = -.1*US(8)
C
C      FROM THE STEADY STATE SET OF EQUATIONS (83,175,177 OF WINFRITH
C      SIMULATOR) AND DUE TO THE FACT THAT STEP INPUTS ARE USED, THE
C      FOLLOWING STATEMENT IS INTRODUCED FOR EXC. TYPES (1) TO (6).(CHCK 3)
C
C      KCD(2,8) = 8.1195/XS(10)
C
C      GO TO (9,2,3,4,5,6), ITYPE
C
C      2 IX0 = IUD = 1
C      X0(10) = 20.      E      UD(8) = .1*US(8)
C      GO TO 9
C      3 IUD = 1
C      UD(1) = 10.E-5
C      GO TO 9
C      4 IUD = 1
C      UD(8) = .1*US(8)
C      GO TO 9
C      5 IX0 = 1
C      X0(10) = 20.
C      GO TO 9
C      6 IUD = 1
C      UD(8) = - .1*US(8)
C
C      9 CALL MULMXV (KCD,UD,UCBAR,NC,NC)
C
C      IF (IUC.EQ.0.AND.TUD.EQ.0) GO TO 11
C      IF (IA = 1) 14,13,14
C
C      COMPUTE VRU  FOR OPEN LOOP CASE
C
C      13 CALL MULMXV (BD,UD,BU,N,NC)
C      . IF (IUC.EQ.0) GO TO 10

```

OKBAL

PE(MW)

PO(PSIA)

PE(MW)

CALL MULMXV (RC,UC,R2,N,NC)

CALL ADDV (RU,R2,RU,N)
GO TO 10

OPTIMALLY CONTROLLED CASE

KDU = - KDG*UD

KDG = KDR*BD + (KDR*BC - II*KCD)

14 CALL MULTI (KDR,BC,RC,NC,N,NC)

DO 15 I=1,NC

15 RC(I,I) = RC(I,I) - 1.

CALL MULTI (RC,KDG,ADD,NC,NC,ND)

CALL MULTI (KDR,BC,KDG,NC,N,ND)

CALL ADD (KDG,ADD,KDG,NC,ND)

CALL WRPMX (KDG,NC,ND,34KDG (UC(T) = -KR*X(T) -KDG*UD))

CALL MULMXV (KDG,UD,KDU,NC,ND)

CALL NEGVR (KDU,NC)

TEST IF KDU IS REQUIRED TO BE CHANGED TO EVALUATE ITS INFLUENCE
IF (IKDU.EQ.1) CALL KDUCHG (NC,KDU,KDUOP)

TEST IF KDU HAS TO BE RECOMPUTED, FOR D.C. COMPENSATION

IF (IDC.EQ.1) CALL DCCOMP (N,NC,ND,A,A1,A2,RC,RC1,NCI,KDUOR,KDU,
1 BC,BD,UD,ADD,AKDU)

VBU = V*RCG*UD

BDG = (I - BC*KDR)*(BD + BC*KDG) = BD - BC*KDG

HENCE VBU = V*(BD*UD + BC*KDU)

CALL MULMXV (BD,UD,RU,N,ND)

CALL MULMXV (RC,KDU,R2,N,NC)

CALL ADDV (RU,R2,RU,N)

10 CALL MULCMRV (V,BU,VBU,N,N)

11 IF (IXC.EQ.0) GO TO 12

CALL MULCMRV (V,X3,VX0,N,N)

12 IXC = 10HSTEP

RETURN

END

SUBROUTINE EXTRA3 (ICX,INAME,NNAM,N,NU,X,XS,US,NC,ND,UC,UD,NXTR,
1 XTR,XTRS,NAMXTR,NAMXTRV,UNTXTR,NAMU,KR,IA,IPR,NAR,NL,XAR,
2 XAX1R,ICASE,NW,IPD,KDR,KDU,MI,MP,IQUICK)

7.4.72

COMPUTES AND PRINTS EXTRA WANTED PARAMETERS WHICH ARE NOT STATE VARIABLES
THIS ROUTINE NEEDS ADJUSTMENT FOR DIFFERENT CASES OF MXEV2X1.

NOW IT IS SET FOR CASES (1) TO (7) OF FORCK1 (MXEV2X1).

REAL X(N),XS(N),US(NU),UC(NC),UD(ND),XTR(NXTR),XTRS(NXTR),

1 KR(NC,N),XAX1R(NAR,NL),XAR(NAR),KDR(NC,N),KDU(ND)

INTEGER NAMXTR(NXTR),NAMXTRV(NXTR),UNTXTR(NXTR),INAME(NNAM),

1 NAMU(NU),ICASE(NW)

FOR ANY CASE, THE FOLLOWING PARAMETERS FROM MXEV2X1 ARE DERIVED

XA(2) = FRAP

XA(3) = TCOOLSTR (DEGF)

XE(2) = TFAVSTAR (DEGF)

YE(7) = NSDPHM (LB/SEC)

XE(12) = NW (LB/SEC)

* XTR(1) = XAR(1)

* XTR(2) = XAR(2)

* XTR(3)

* XTR(4)

* XTR(5)

```

C Y(3)-U2(10)=

C Y(3)-U2(6) = DL (INCHES) XTR(6)
C XE(1) = 7KM XTR(8)
C XA(5) = 0DASH (NORMALIZED) XTR(9) = XAR(3)
C XA(6) = 7HSC (BTU/LB) XTR(10) = XAR(4)
C XA(9) = 0HR (BTU/LB) XTR(11) = XAR(5)
C XA(12) = KXCOHT XTR(12) = XAR(6)
C XA(14) = KXRRAR XTR(13) = XAR(7)
C FOR CASES (1) TO (7) OF FDBCK1 IN MXEV2X1
C U2(4) = YRDEM (NORMAL.) XTR(7)
C NOT TO BE CONFUSED WITH U2(4) = YRDEM(REF)
C
C GO TO (10,20,20), IEX
C
C DATA TAKEN FROM MXEV2X1 (UMAC06X, 16.9,70)
C
C COMPUTE XA(2,3,5,6,8,12,14) WHICH CORRESPOND TO XTR(1,2,9,10,11,12,13)
C
20 DO 27 I=1,NAR
  XAR(I) = 0.
  DO 27 J=1,NL
27 XAR(I) = XAR(I) + XAX1R(I,J)*X(J)
  DO 30 I=1,NAR
    IF (I - 2) 28,28,29
28 XTR(I) = XAR(I)
    GO TO 30
29 XTR(I+6) = XAR(I)
30 CONTINUE

C
C DATA TAKEN FROM SSNL2F (CALFUM, 18.11.69) (EASIER THAN PUNCHING OUT)
C
  XTR(3) = .5*X(8) + .5*X(9)
  XTP(4) = .3862*XS(23)*X(10) + .3862*XS(10)*X(23)
  XTR(5) = - .3999E-1*XS(14)*X(13) + (- .3999E-1*XS(13) + 51.8)*X(14)
  XTR(9) = SLOPE1(XS(22) - .4)*X(22)

C
C DATA TAKEN FROM FDBCK1 (CASES 3 TO 7)
C
  XTR(6) = .1268E-2*X(12) - U2(6)
  TK13 = 5. 2 TK14 = .025 E TK15 = 1.
  XTR(7) = TK13*X(29) - TK13*TK14*X(28) - TK13*TK14*XTR(6) + .0029*
  1 TK13*TK15*XTR(4) - .0029*TK13*X(16)

C
C IA = 1 -> A
C IA = 2 -> AOC
C
C UC(T) = - KR*X(T) - KOG*U2 = - KR*X(T) + KDU
C (KDU GOT NEGATIVE SIGN IN ROUTINE EXCITE)
C
  IF (IA - 1) 21,22,21
21 CALL MULMXV (KR,X,U2,NC,N)
  CALL NEGVR (UC,NC)
  CALL ADDV (UC,KDU,U2,NC)

C
22 IF (IEX.EQ.2) RETURN

C
C PRINT RESULTS IF REQUIRED (IPR.NE.0)
C
  IF (IPR.EQ.0) RETURN
  PRINT 100, NXTR
  DO 23 I=1,NXTR
23 PRINT 101, NAMXTRV(I),XTR(I),NAMXTR(I)
  IF (IA - 1) 24,26,24
24 PRINT 102, NC

```

DO 25 I=1,ND

IS = I+ND

25 PRINT 103, I, UC(I), NAMU(IS)

26 RETURN

100 FORMAT (/10X, *FOLLOWING ARE THE VALUES OF*, I3, * PARAMETERS WHICH ARE NOT STATE VARIABLES*/)

101 FORMAT (14X, A10, * = *, E11.4, 5X, A10)

102 FORMAT (/10X, *FOLLOWING ARE THE VALUES OF*, I3, * CONTROL INPUTS*/)

103 FORMAT (18X, *UC(*, I2, *) = *, E11.4, 5X, A10)

C
C
C
C
C
C

READ IN MATRIX KR (I.E. K(REDUCED)) AND KDR (KD(REDUCED)),

UC(T) = - KR*X(T) - KDR*UD + (I - KDR*PC)*UCPR

= - KP*X(T) - KDG*UD = - KR*X(T) + KDU

(SEE DEFINITIONS IN MAIN PROGRAM)

10 IF (IA - 1) 11, 12, 11

11 READ (MI, 200) INAME

CALL IPM2013 (KP, NC, N, MI)

PRINT 201, INAME

ICASE(1) = INAME(1)

READ (MI, 200) INAME

CALL IRM2013 (KDR, NC, N, MI)

PRINT 201, INAME

IF (IQWICK - 1) 14, 12, 14

14 CALL WRRMX (KP, NC, N, 30HKR (UC(T) = - KP*X(T) - KDG*UD))

CALL WRRMX (KDR, NC, N, 30HKDR (KDG=KDR*AD+(KDR*BC-I)*KCD))

C
C
C
C
C
C
C
C
C
C
C

READ IN MATRIX XAX1R (I.E. XAX1 REDUCED) FROM PUNCHED OUTPUT IN MXEV2X1,

XAR = XAX1R*X1

WHERE X1 IS THE VECTOR OF THE FIRST NL(27) ELEMENTS OF X

THE ORIGINAL EQUATION IN MXEV2X1 WAS

XA = XAX1*X1 + XAU*U

BUT FOR THE XTR VALUES NEEDED HERE, THE CORRESPONDING ELEMENTS IN XAU ARE NIL

AND ALSO THE MATRIX XAX1(17, 27) IS REDUCED TO XAX1R(7, 27) FOR THE SAME REASON

12 READ (MP, 200) INAME

PRINT 201, INAME

CALL IRM2013 (XAX1R, NAR, NL, MP)

READ (MP, 200) INAME

PRINT 201, INAME

IF (IPD.EQ.6) GO TO 13

CALL WRRMX (XAX1R, NAR, NL, 30HXAX1R (XAR(T) = XAX1R*X1(T)))

C
C
C
C

READ IN STEADY STATE VALUES, NAMES AND UNIT OF EXTRA PARAMETERS

DATA TAKEN FROM MXEV2X1 (NEWXII)

13 READ (MP, 200) INAME

CALL IRV2013 (XTR, NXTR, MP)

CALL IIV8A10 (NAMXTRV, NXTR, MP)

CALL IIV8A10 (NAMXTR, NXTR, MP)

CALL IIV8A10 (UNTXTR, NXTR, MP)

PRINT 201, INAME

RETURN

200 FORMAT (8A10)

201 FORMAT (10X, 8A10)

END

C
C
C
C

SUBROUTINE CONSTR1(IC, IP, N, UPPND, LORND, UPDER, LODER, PPK, NPK, PPDER, INPDER, LUPB, LLOR, LUPD, LLOD, VIOLB, VIOLD, SS, NAMV, NAM, UNT, PPBND, LPPB)

C
C
C
C

31.1.72

C IC = 1 → SETS CONSTRAINTS ON PARAMETER VALUES AND/OR DERIVATIVES

C IC = 2 → PRINTS CONSTRAINTS, PEAK VALUES AND VIOLATIONS

C IP = 1 → HANDLES UC PARAMETERS

C IP = 2 → HANDLES X PARAMETERS

C IP = 3 → HANDLES XTR PARAMETERS

```

C
  REAL UPRND(M), LORND(M), UPPER(M), LOWER(M), PK(M), NPK(M), PPDER(M),
1    NPDER(M), SS(M), PPRND(M)
  INTEGER NAMV(M), NAM(M), UNT(M), FM(12), FO(12), FP(13)
  LOGICAL LUPR(M), LLOP(M), LUPD(M), LLOD(M), VIOL3(M), VIOLD(M), LPPR(M)
  DATA FO/
1  60H(1X,*DERIVATIVE OF *,4X,      *UC(*,I2,*) = *,A10,2X,A10,      , ID TO FM
2  50H*/SEC*,9X,2X,      1PE10.3, 1PE12.3,5X,E3.1,      2X,E3.1) / ID TO FM
  DATA FP/ 50H(19X,      (15X,      ,A10,      * X(*,I2,*, REPLACE
1  40H* = *,A10,10X,      17X,      1PE12.3,      , REPLACE
2  40H1PE10.3,* PEAK TO PK*,5X,E3.1,2X,E3.1) / REPLACE

```

```

C
  GO TO (1,50), IC
1 DO 2 I=1,M
  LUPR(I) = LLOP(I) = LUPD(I) = LLOD(I) = LPPR(I) = .FALSE.
2 CONTINUE
  GO TO (5,10,15), IP

```

```

C
C SET CONSTRAINTS FOR CONTROLS UC
5 LUPR(1) = LLOP(1) = .TRUE.
  LUPR(2) = LLOP(2) = LUPD(2) = LLOD(2) = .TRUE.
  UPRND(1) = 1.      & LORND(1) = - 1.
  UPRND(2) = 3.25    & LORND(2) = 0.
  UPPER(2) = .0033   & LOWER(2) = -.0033
  RETURN

```

```

C
C SET CONSTRAINTS FOR STATE VARIABLES X
10 LPPR(1) = .TRUE.
  LUPR(10) = LLOP(10) = LLOD(10) = .TRUE.
  LLOP(11) = LLOP(12) = LLOP(15) = LLOP(16) = .TRUE.
  LUPR(19) = LLOP(19) = LUPD(19) = LLOD(19) = .TRUE.
  LUPR(21) = LLOP(21) = LUPD(21) = LLOD(21) = .TRUE.
  LUPR(22) = .TRUE.
  LUPR(23) = LLOP(23) = .TRUE.
  LUPR(28) = LUPR(29) = .TRUE.
  LUPR(30) = LLOP(30) = .TRUE.
  PPRND(1) = .2
  UPRND(10) = SS(10) + 40. & LORND(10) = SS(10) - 40.
  LOWER(17) = - 1.
  LORND(11) = LORND(12) = LORND(15) = LORND(16) = 0.
  UPRND(19) = 1.      & LORND(19) = 0.
  UPPER(19) = .033    & LOWER(19) = -.033
  UPRND(21) = 1.      & LORND(21) = - 1.
  UPPER(21) = .2      & LOWER(21) = -.2
  UPRND(22) = .97
  UPRND(23) = 1.      & LORND(23) = 0.
  UPRND(29) = UPRND(29) = 1.
  UPRND(30) = 1.      & LORND(30) = - 1.
  RETURN

```

C SET CONSTRAINTS FOR ADDITIONAL PARAMETERS XTR

```

15 LUPR( 6) = LLOP( 6) = .TRUE.
  LUPR( 7) = LLOP( 7) = .TRUE.
  LUPD( 8) = LLOP( 8) = .TRUE.
  LPPR(9) = .TRUE.
  LLOP(10) = LLOP(11) = LLOP(12) = LLOP(13) = .TRUE.
  UPRND( 6) = 8.      & LORND( 6) = - 8.
  UPRND( 7) = 1.      & LORND( 7) = 0.
  UPPER( 8) = 3.E-5    & LOWER( 8) = - 3.E-5

```

PPBND(9) = .24

LOBND(10) = LOBND(11) = LCRND(12) = LOBND(13) = 0.
RETURN

```

C
C PRINT CONSTRAINTS, PEAK VALUES AND VIOLATIONS
C PRINT TITLES AND SET FORMAT DATA
C
50 DO 81 I=1,12
81 FM(I) = FO(I)
   GO TO (51,52,53), IP
51 PRINT 101
101 FORMAT (15X,*CONTROL INPUTS (UC)*/)
   GO TO 54
52 PRINT 102
102 FORMAT (/15X,*STATE VARIABLES*/ )
   FM(4) = FR(5)
   GO TO 54
53 PRINT 103
103 FORMAT (/15X,*ADDITIONAL PARAMETERS*/ )
   FM(3) = FR(4)   &   FM(4) = FR(3)   &   FM(5) = FR(6)
C
C SCAN PARAMETERS
54 DO 80 I=1,M
   IF (.NOT.(LUPB(I).OR.LLOB(I).OR.LUPD(I).OR.LLOD(I).OR.LPPB(I)))
1   GO TO 80
   IF (.NOT.(LUPB(I).OR.LLOB(I).OR.LPPB(I))) GO TO 69
C
C PARAMETER BOUNDS, SET FORMATS AND PRINT
   FM(7) = FR(9)
   GO TO (55,55,56), IP
55 FM(1) = FR(1)   &   FM(2) = FM(3) = FR(3)
   GO TO 57
56 FM(1) = FR(2)   &   FM(2) = FR(3)
57 IF(.NOT.((LUPB(I).AND.LLOB(I)).OR.LPPB(I))) GO TO 60
   IF (.NOT.LPPB(I)) GO TO 40
   FM(9) = FR(10)   &   FM(10) = FR(11)   &   FM(11) = FR(12)
   FM(12) = FR(13)
   GO TO (31,31,32), IP
31 PRINT FM,      I ,NAM(I),UNT(I),SS(I),PPBND(I),PPK(I),NPK(I)
   GO TO 66
32 PRINT FM, NAMV(I),NAM(I),UNT(I),SS(I),PPBND(I),PPK(I),NPK(I)
   GO TO 66
40 GO TO (58,58,59), IP
58 PRINT FM,      I ,NAM(I),UNT(I),SS(I),UPBND(I),LOBND(I),PPK(I),
1   NPK(I)
   GO TO 66
59 PRINT FM, NAMV(I),NAM(I),UNT(I),SS(I),UPBND(I),LOBND(I),PPK(I),
1   NPK(I)
   GO TO 66
60 IF (LUPB(I)) GO TO 63
   FM(9) = FR(7)
   GO TO (61,61,62), IP
61 PRINT FM,      I ,NAM(I),UNT(I),SS(I),LCRND(I),PPK(I),NPK(I)
   GO TO 66
62 PRINT FM, NAMV(I),NAM(I),UNT(I),SS(I),LCRND(I),PPK(I),NPK(I)
   GO TO 66
63 FM(10) = FR(8)
   GO TO (64,64,65), IP
64 PRINT FM,      I ,NAM(I),UNT(I),SS(I),UPBND(I),PPK(I),NPK(I)
   GO TO 66
65 PRINT FM, NAMV(I),NAM(I),UNT(I),SS(I),UPBND(I),PPK(I),NPK(I)
C
C CHECK VIOLATION
66 IF (.NOT.VIOLB(I)) GO TO 67

```


PRINT 104

104 FORMAT (1H+,111X,*CONSTRAINT VIOLATED*)

```

C
C RESTORE FORMAT AS CHANGED FROM STATEMENT 55 ONWARDS
67 FM(1)=FO(1) &FM(2)=FO(2) &FM(7)=FO(7) &FM(9)=FO(9) &FM(10)=FO(10)
   FM(11) = FO(11) & FM(12) = FO(12)
   GO TO (68,63,69), IP.
68 FM(3) = FO(3)
C
C PARAMETER DERIVATIVES ROUNDS
69 IF (.NOT.(LUPD(I).OR.LLOD(I))) GO TO 80
   IF (.NOT.(LUPD(I).AND.LLOD(I))) GO TO 72
   GO TO (70,71,71), IP
70 PRINT FM, I, NAM(I), UNT(I), UPDER(I), LODER(I), PPDER(I), NPDER(I)
   GO TO 78
71 PRINT FM, NAMV(I), NAM(I), UNT(I), UPDER(I), LODER(I), PPDER(I), NPDER(I)
   GO TO 78
72 IF (LUPD(I)) GO TO 75
   FM(9) = FP(7)
   GO TO (73,73,74), IP
73 PRINT FM, I, NAM(I), UNT(I), LODER(I), PPDER(I), NPDER(I)
   GO TO 78
74 PRINT FM, NAMV(I), NAM(I), UNT(I), LODER(I), PPDER(I), NPDER(I)
   GO TO 78
75 FM(10) = FR(8)
   GO TO (76,76,77), IP
76 PRINT FM, I, NAM(I), UNT(I), UPDER(I), PPDER(I), NPDER(I)
   GO TO 78
77 PRINT FM, NAMV(I), NAM(I), UNT(I), UPDER(I), PPDER(I), NPDER(I)
C
C CHECK VIOLATION
78 IF (.NOT.VIOLD(I)) GO TO 79
   PRINT 104
C
C RESTORE FORMAT AS CHANGED FROM STATEMENT 72 ONWARDS
79 FM(9) = FO(9) & FM(10) = FO(10)
80 CONTINUE
   RETURN
   END

```

```

C
C
C
C SUBROUTINE PLOTS ( N,NC,NXTR,NL,XP,UCP,XTRP,TP,TL,ND,NU,XS,UCS,
1  XTRS,NAMX,NAMUC,NAMXTR,NAMXTRV,UNTX,UNTUC,UNTXTR,IEXC,ISY,ICASE
2  ,NW,IA,ITYPE,IQUICK,KR,URUC,LUC,KUC)

```

5.3.72

PLOTS SELECTED PARAMETERS FROM X, XTR, AND UC ON A SCALE SUCH AS TO FIT INTO THE RANGE -0.2 TO + 1.0

THE SELECTING OF PARAMETERS IS DONE MANUALLY

THE SCALING OF PARAMETERS IS DONE AUTOMATICALLY

PRINTS THE PEAK VALUES OF PARAMETERS BOUND BY CONSTRAINTS (AND/OR ITS DERIVATIVES) AND FLAGS A WARNING WHEN A CONSTRAINT IS VIOLATED.

PRINTS ALL PEAK VALUES X(J), COMPUTES AND PRINTS ..WEIGHTED GAIN.. MATRIX [KR(I,J)*X(J)PEAK] IN ORDER TO DETERMINE ..REDUCED SET..

```

REAL XP(N,NL),UCP(NC,NL),XTRP(NXTR,NL),TP(NL),XS(N),UCS(NC),
1  XTRS(NXTR),KR(NC,N)
INTEGER NK(5),BLANK(6),ICASE(NW),NAMX(N),NAMUC(NC),NAMXTR(NXTR),
1  NAMXTRV(NXTR),UNTX(N),UNTUC(NC),UNTXTR(NXTR)
REAL SCALX(31),SCALUC(2),SCALXTR(14)

```

INTEGER WARNX(31),WARNUC(2),WARNXTR(14)

C DEFINITIONS FOR THE CONSTRAINTS PART

```

REAL UBX(31),LBX(31),UBX(31),LDBX(31),PPX(31),NPX(31),PPPX(31),
1  NPDUX(31),UBUC(2),LBUC(2),UDUC(2),LDUC(2),PPUC(2),NPUC(2),
2  PPDUUC(2),NPDUC(2),UBXR(14),LBXR(14),UDXR(14),LDXR(14),PPXR(14),
3  ,NPXR(14),PPDXR(14),NPDXR(14),PPDX(31),PPDXR(14),PPBUC(2),
4  PMDX(31)
LOGICAL LUBX(31),LLBX(31),LUDX(31),LLDX(31),VBX(31),VDBX(31),
1  LUBUC(2),LLBUC(2),LUDUC(2),LLDUC(2),VBUC(2),VDBUC(2),
2  LUBXR(14),LLBXR(14),LUDXR(14),LLDXR(14),VBXR(14),VDBXR(14)
LOGICAL LPPBX(31),LPPDXR(14),LPPBUC(2)
DATA MK,9LANK/ 1H1,1H2,1H3,1H4,1H5,60H
1
GMAX = 1.
GMIN = -.2
GPEF = 0.

```

C SET CONSTRAINTS

```

CALL CONST1(1,1,NC,UBUC,LBUC,UDUC,LDOC,PPUC,NPUC,PPDUUC,NPDUC,
1  LUBUC,LLBUC,LUDUC,LLDUC,VBUC,VDBUC,UCS,WARNX,NAMUC,UNUC,
2  PPBUC,LPPBUC)
IF (KUC.EQ.1) RETURN
CALL CONST1(1,2,H,UBX,LBX,UDX,LDBX,PPX,NPX,PPPX,NPDUX,
1  LUBX,LLBX,LUDX,LLDX,VBX,VDBX,XS,WARNX,NAMX,UNTX,
2  PPBX,LPPBX)
CALL CONST1(1,3,MXTR,UDXR,LBXR,UDXR,LDBR,PPXR,NPXR,PPDXR,NPDXR,
1  LUBXR,LLBXR,LUDXR,LLDXR,VBXR,VDBXR,XTR,NAMXTRV,NAMXTR,UNTXTR,
2  PPBXR,LPPBXR)

```

C COMPUTE SCALES, PEAK VALUES, ETC., PRESERVE MAX. MOD FOR X ONLY

```

CALL SCALPL1(NC,NL,XP,SCALX,WARNX,GMAX,GMIN,TP,UBX,LBX,
1  UDX,LDBX,PPX,NPX,PPPX,NPDUX,LUBX,LLBX,LUDX,LLDX,
2  VBX,VDBX,XS,PPBX,LPPB,PMDX,1)
CALL SCALPL1(NC,NL,UCP,SCALUC,WARNUC,GMAX,GMIN,TP,UBUC,LBUC,
1  UDUC,LDOC,PPUC,NPUC,PPDUUC,NPDUC,LUBUC,LLBUC,LUDUC,LLDUC,
2  VBUC,VDBUC,UCS,PPBUC,LPPBUC,PMDC,0)
CALL SCALPL1(MXTR,NL,XTR,SCALXTR,WARNXTR,GMAX,GMIN,TP,UBXR,LBXR,
1  UDXR,LDBR,PPXR,NPXR,PPDXR,NPDXR,LUBXR,LLBXR,LUDXR,LLDXR,
2  VBXR,VDBR,XTR,PPBXR,LPPBXR,PMDX,0)

```

C CHECK WHETHER IQUICK = 1, IF SO JUMP PLOTS FOR EXC(5)

```

IF (IQUICK.NE.1) GO TO 11
IF (ITYPE - 5) 11,8,11

```

C SCALE AND PLOT X(1), X(4), X(9), X(9), X(10)

```

11 PRINT 100, IEXC,ISY,ICASE,ITYPE
PRINT 101, MK(1),NAMX(1),SCALX(1),UNTX(1),XS(1),WARNX(1),
1  MK(2),NAMX(4),SCALX(4),UNTX(4),XS(4),WARNX(4),
2  MK(3),NAMX(8),SCALX(8),UNTX(8),XS(8),WARNX(8),
3  MK(4),NAMX(9),SCALX(9),UNTX(9),XS(9),WARNX(9),
4  MK(5),NAMX(10),SCALX(10),UNTX(10),XS(10),WARNX(10)
100 FORMAT (1H1,9X,A10,*RESPONSE OF SYSTEM *,A10,*,*,3A10,*,EXC TY
1PF (*,I2,*)*///34X,*SCALE*,10X,*UNIT S. STATE VALUE*/)
101 FORMAT(
1  (*,A1,*) * X( 1) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
2  (*,A1,*) * X( 4) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
3  (*,A1,*) * X( 8) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
4  (*,A1,*) * X( 9) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
5  (*,A1,*) * X(10) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
NLIN = - 1
DO 1 T=1,NL
A = XP(1,I)/SCALX(1)

```

```
B = XP(4,I)/SCALX(4)
```

```
G = XP(8,I)/SCALX(8)
```

```
D = XP(9,I)/SCALX(9)
```

```
E = XP(10,I)/SCALX(10)
```

```
TIME = TP(I)
```

```
1 CALL GPH (BLANK,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,MK)
```

C

C SCALE AND PLOT X(11), X(12), X(13), X(16), X(17)

```
PRINT 100, IEXC,ISY,ICASE,ITYPE
```

```
PRINT 102, MK(1),NAMX(11),SCALX(11),UNTX(11),XS(11),WARNX(11),
```

```
1 MK(2),NAMX(12),SCALX(12),UNTX(12),XS(12),WARNX(12),
```

```
2 MK(3),NAMX(13),SCALX(13),UNTX(13),XS(13),WARNX(13),
```

```
3 MK(4),NAMX(16),SCALX(16),UNTX(16),XS(16),WARNX(16),
```

```
4 MK(5),NAMX(17),SCALX(17),UNTX(17),XS(17),WARNX(17)
```

```
102 FORMAT (
```

```
1* (*,A1,*) + X(11) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
2* (*,A1,*) + X(12) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
3* (*,A1,*) + X(13) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
4* (*,A1,*) + X(16) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
5* (*,A1,*) + X(17) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
```

```
NLIN = - 1
```

```
DO 2 I=1,NL
```

```
A = XP(11,I)/SCALX(11)
```

```
B = XP(12,I)/SCALX(12)
```

```
C = XP(13,I)/SCALX(13)
```

```
D = XP(16,I)/SCALX(16)
```

```
E = XP(17,I)/SCALX(17)
```

```
TIME = TP(I)
```

```
2 CALL GPH (BLANK,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,MK)
```

C

C SCALE AND PLOT X(18), X(19), X(20), X(21), X(22)

```
PRINT 100, IEXC,ISY,ICASE,ITYPE
```

```
PRINT 103, ((MK(I),NAMX(I+17),SCALX(I+17),UNTX(I+17),XS(I+17),
```

```
1 WARNX(I+17)),I=1,5)
```

```
103 FORMAT (
```

```
1* (*,A1,*) + X(18) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
2* (*,A1,*) + X(19) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
3* (*,A1,*) + X(20) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
4* (*,A1,*) + X(21) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
5* (*,A1,*) + X(22) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
```

```
NLIN = - 1
```

```
DO 3 I=1,NL
```

```
A = XP(18,I)/SCALX(18)
```

```
B = XP(19,I)/SCALX(19)
```

```
C = XP(20,I)/SCALX(20)
```

```
D = XP(21,I)/SCALX(21)
```

```
E = XP(22,I)/SCALX(22)
```

```
TIME = TP(I)
```

```
3 CALL GPH (BLANK,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,MK)
```

C

C SCALE AND PLOT X(23), X(30), XA(2), XA(3), XE(2)

```
PRINT 100, IEXC,ISY,ICASE,ITYPE
```

```
PRINT 104, MK(1),NAMX(23),SCALX(23),UNTX(23),XS(23),WARNX(23),
```

```
1 MK(2),NAMX(30),SCALX(30),UNTX(30),XS(30),WARNX(30),
```

```
2 ((MK(I+2),NAMXTPV(I),NAMXTR(I),SCALXTR(I),UNTXTR(I),XTRS(I),
```

```
3 WARNXTR(I)),I=1,3)
```

```
104 FORMAT (
```

```
1* (*,A1,*) + X(23) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
2* (*,A1,*) + X(30) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
3* (*,A1,*) + *,A10 ,* = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
4* (*,A1,*) + *,A10 ,* = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
```

```
5* (*,A1,*) + *,A10 ,* = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
```

```
NLIN = - 1
```

```
DO 4 I=1,NL
```

```

      A = XP(23,I)/SCALX(23)

      B = XP(30,I)/SCALX(30)
      C = XTRP(1,I)/SCALXTR(1)
      D = XTRP(2,I)/SCALXTR(2)
      E = XTRP(3,I)/SCALXTP(3)
      TIME = TP(I)
4 CALL GPH (BLANK,NLIN,TIME,GMAX,GPEF,GMIN,A,B,C,D,E,MK)
C
C SCALE AND PLOT XC(7),YC(12),Y(3)-UD(6),U2(4)
PPINT 100, IEXC,ISY,ICASE,ITYPE
PRINT 105, ((MK(I),NAMXTRV(I+3),NAMXTR(I+3),SCALXTR(I+3),
1          UNTXTR(I+3),XTRS(I+3),WARNXTR(I+3)),I=1,4)
105 FORMAT (
1          1* (*,A1,*) ,*,A10 ,* = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10))
      NLIN = - 1
      DO 5 I=1,NL
      A = XTRP(4,I)/SCALXTP(4)
      B = XTRP(5,I)/SCALXTP(5)
      C = XTRP(6,I)/SCALXTP(6)
      D = XTRP(7,I)/SCALXTR(7)
      E = 2.
      TIME = TP(I)
5 CALL GPH (BLANK,NLIN,TIME,GMAX,GPEF,GMIN,A,B,C,D,E,MK)
C
C SCALE AND PLOT XC(1),YA(5) (XTP((8,9)), X(2),X(3)
PPINT 100, IEXC,ISY,ICASE,ITYPE
PPINT 107, ((MK(I),NAMXTPV(I+7),NAMXTR(I+7),SCALXTR(I+7),
1          UNTXTR(I+7),XTRS(I+7),WARNXTR(I+7)),I=1,2),
2          ((MK(I+1),NAMX(I),SCALX(I),UNTX(I),XS(I),WARNX(I)),
3          I=2,3)
107 FORMAT (
1          1* (*,A1,*) ,*,A10 ,* = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10)/
2          2* (*,A1,*) ,* X( 2) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
3          3* (*,A1,*) ,* X( 3) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
      NLIN = - 1
      DO 7 I=1,NL
      A = XTRP(8,I)/SCALXTR(8)
      B = XTRP(9,I)/SCALXTP(9)
      C = XP(2,I)/SCALX(2)
      D = XP(3,I)/SCALX(3)
      E = 2.
      TIME = TP(I)
7 CALL GPH (BLANK,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,MK)
C
C SCALE AND PLOT CONTROL INPUTS UC(1),UC(2)
IF (IA.EQ.1) GO TO 8
PRINT 100, IEXC,ISY,ICASE,ITYPE
PRINT 106, ((MK(I),NAMUC(I), SCALUC(I),UNTXUC(I), UCS(I),
1          WARNUC(I)),I=1,NC)
106 FORMAT (
1          1* (*,A1,*) ,* UC(1) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 /
2          2* (*,A1,*) ,* UC(2) = *,A10,* 1. = *,1PE7.0,2X,A10,E13.4,A10 )
      NLIN = - 1
      DO 6 I=1,NL
      A = UCP(1,I)/SCALUC(1)
      B = UCP(2,I)/SCALUC(2)
      C = D = E = 2.
      TIME = TP(I)
6 CALL GPH (BLANK,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,MK)
PRINT 109
109 FORMAT (1HP)
C
C PRINT PEAK VALUES AND CONSTRAINTS OF PARAMETERS AND/OR DERIVATIVES
C

```

8 PRINT 108, IEXC, ISY, ICASE, ITYPE

108 FORMAT (1H1, 9X, A10, *RESPONSE OF SYSTEM *, A10, *, *, 3A10, *, EXC TY
1PE (*, I2, *) *///10X, *LIST OF CONSTRAINED PARAMETERS AND/OR DERIVATI
2VES (A VIOLATED CONSTRAINT YIELDS A WARNING) *///
3 28X, *NAME*, 8X, *UNIT*, 10X, *S. STATE*, 10X, *CONSTRAINTS*, 12X, *PE
4AK VALUES*/10X, *(LINEAR DEVIATIONS)*/67X, *UPPER*, 7X, *LOWER*, 7X, *PO
5SITIVE NEGATIVE*//)

CALL CONSTR1 (2, 1, NC, URUC, LBUC, UDUC, LOUC, PPUC, NPUC, PPOUC, NPOUC,
1 LURUC, LLUC, LUUC, LLOUC, VBUC, VUC, UCS, WARNX, NAMUC, UNTUC,
2 PPOUC, LPPUC)

CALL CONSTR1 (2, 2, N, URX, LBX, UTX, LTX, PDX, NPX, PPDX, NPD
1 LURX, LLBX, LUUX, LLOX, VBX, VOX, XS, WARNX, NAMX, UNTX,
2 PPDX, LPPDX)

CALL CONSTR1 (2, 3, NXTR, URXR, LXR, UXR, LXR, PPR, NPPR, PPRR, NPPR,
1 LURR, LLRR, LUUR, LLOUR, VBR, VOR, XTR, NAMXTPV, NAMXTR, UNTXTR,
2 PPRR, LPPRR)

C COMPUTE ..WEIGHTED GAIN.. MATRIX [KR(I, J)*X(J)PEAK]

C PRINT PEAK VALUES AND ..WEIGHTED GAIN.. MATRIX

C (URX, LBX ARE NOT NEEDED ANY MORE, HENCE THEY WILL BE USED FOR THE
C TWO VECTORS OF THE W.G. MATRIX)

C DO 21 J=1, N
C URX(J) = KR(1, J)*PMDX(J)
21 LBX(J) = KR(2, J)*PMDX(J)

C FIND THE MAXIMUM VALUE OF EACH VECTOR IN THE W.G. MATRIX
C MAX1 = 0 & MAX2 = 0 & WM1MAX = 0. & WM2MAX = 0.

DO 25 J=1, N
IF (ABS(URX(J)) - WM1MAX) 23, 23, 22
22 WM1MAX = ABS(URX(J))
MAX1 = J
23 IF (ABS(LBX(J)) - WM2MAX) 25, 25, 24
24 WM2MAX = ABS(LBX(J))
MAX2 = J
25 CONTINUE
WM1DOM = WM1MAX/10.
WM2DOM = WM2MAX/10.

C PRINT TITLES

PRINT 110, IEXC, ISY, ICASE, ITYPE

C PRINT VALUES AND POINT AT THE DOMINANT ONES, I.E. WHEN THEY ARE
C .GE. (MAXIMUM VALUE/10)

C DO 31 J=1, N
C PRINT 111, (J, NAMX(J), PDX(J), NPX(J), URX(J), LBX(J))
C IF (J - MAX1) 27, 26, 27
26 PRINT 112
C GO TO 28
27 IF (ABS(URX(J)).GE.WM1DOM) PRINT 114
28 IF (J - MAX2) 30, 29, 30
29 PRINT 113
C GO TO 31
30 IF (ABS(LBX(J)).GE.WM2DOM) PRINT 115
31 CONTINUE
PRINT 116
RETURN

110 FORMAT (1H1, 9X, A10, *RESPONSE OF SYSTEM *, A10, *, *, 3A10, *, EXC TY
1PE (*, I2, *) *///10X, 44H..WEIGHTED GAIN.. MATRIX [KR(I, J)*X(J)PEAK]
2 //10X, *(BY X(J)PEAK WE MEAN THE POSITIVE OR NEGATIVE PEAK WITH TH
3E LARGEST ABSOLUTE VALUE) *///28X, *NAME*, 15X, *PEAK VALUES*, 14X, 16HK
4P(I, J)*X(J) PEAK, 12X, *CATEGORY*/45X, *POS.*, 7X, *NEG.*, 13X, *I=1*, 8X,
5*I=2*, 6X, *FOR UC(1)*, 4X, *FOR UC(2)*//)

```

111 FORMAT(20X,*X(*,I2,*) E *,A10,10E13.1,E11.1,E16.1,E11.1)

112 FORMAT (1H+, 9X,*MAX. VALUE*)
113 FORMAT (1H+,106X,*MAX. VALUE*)
114 FORMAT (1H+, 93X,*OOM. VALUE*)
115 FORMAT (1H+,156X,*OOM. VALUE*)
116 FORMAT (/ ** / 10X,*(THE DOMINANT VALUES ARE DEFINED AS THOSE WITHIN
    ONE DECADE OF THE MAXIMUM VALUE)*)
    END

```

C
C
C

```

SUBROUTINE GPH (ALF,NLIN,TIME,GMAX,GREF,GMIN,A,B,C,D,E,IST)
REAL V(5)
INTEGER ALF(6),IST(5),LA(121),KSTAR(7)
IF(NLIN.GE.0) GO TO 1
DO 400 I=1,5
DATA(KSTAR(I),I=1,7) / 1H.,1H.,1H*,1H0,1H-,1H+,1HX/
DATA IBLANK /10H /,IREF/1H+/
400 KSTAR(I)=IST(I)
NLIN=0
1 IF(NLIN.EQ.0) WRITE(6,3) ALF
3 FORMAT( / 6A10 /* ++++++
1+++++
2+++++ TIME (SEC)*/ -0.2 -0.1 0. .1
3.2 .9 .3 .4 .5 .6 .7 .8
4 .9 1. +++++* + + + + +
5 + + + + +
6 + + + + +
NLIN=NLIN+1
T=TIME
GALL=GMAX-GMIN

```

C QUANTISE VALUES

```

QUANT=GALL/120.
QLOW =GMIN -QUANT/2.0
QGAP =GALL +QUANT
DO 4 J=1,121
4 LA(J)=IBLANK
V(1)=A
V(2)=B
V(3)=C
V(4)=D
V(5)=E
V(6) = GREF
DO 5 M=1,6
L= IFIX(121.*(V(M)-QLOW)/QGAP) +1
IF(L.LT.1.OR.L.GT.121) GO TO 5
IF(LA(L).NE.IBLANK) GO TO 6
LA(L)= KSTAR(M)
GO TO 5
6 LA(L)=KSTAR(7)
5 CONTINUE

```

C LINE COUNT TO REPEAT REFERENCE POINTS

```

IREM= MOD(NLIN,20)
IF(NLIN.EQ.1) GO TO 110
IF (IREM.NE.1) GO TO 120
DO 8 I=1,121,10
8 IF (I.EQ.21) GO TO 8
IF (LA(I).NE.IBLANK) GO TO 8
LA(I) = IREF

```

A CONTINUE

```

100 WRITE(6,101) LA,T
101 FORMAT(1H0,121A1,F12.2)
RETURN

```

110 WRITE(6,111) LA

111 FORMAT(1X,121A1,12H ZERO)
 RETURN
 END

```

C
C
C
  SUBROUTINE SCALPL1 (M,N,A,SCAL,WARN,GP,GN,TP,UPBND,LOBND,UPDER,
1  LODER,PPK,NPK,PPER,NPDER,LUPB,LLPB,LUPD,LLPD,VIOLB,VIOLD,SS,
2  PPRND,LPRD,PMOX,IX)
C
C  31.1.72
C
C  DETERMINES THE SCALE OF THE PARAMETER CONTAINED IN EACH ROW OF MATRIX A(M,N)
C  SUCH THAT IT FITS INTO A PLOT WHERE THE MAXIMUM DIMENSIONS ARE GP=+1.,GN=-.2
C  THE VALUES GP, GN ARE CHOSEN HERE FOR CONVENIENCE. BUT ABS(GN) MUST IN NO
C  CIRCUMSTANCES BE SMALLER THAN GP/10.
C
C  TESTS ARE MADE TO SEE IF CERTAIN PARAMETERS AND OR THEIR DERIVATIVES VIOLATE
C  CONSTRAINTS.
C
  REAL A(M,N),SCAL(M),TP(N),SS(M),UPBND(M),LOBND(M),UPDER(M),
1  LODER(M),PPK(M),NPK(M),PPER(M),NPDER(M),PPRND(M),PMOX(M)
  LOGICAL LUPB(M),LLPB(M),LUPD(M),LLPD(M),VIOLB(M),VIOLD(M),LPPB(M)
  INTEGER WARN(M)
  DATA IBLANK,INEGL/10H                   ,10H   NO PLOT/
  Z = 1.E-10
  IF (ABS(GN).LT.ABS(.1*GP)) GO TO 21
  GPM = .5*GP
  GPN = .2*GP
  GP2 = 2.*GP
  DO 1 I=1,M
1  WARN(I)=IBLANK
C
C  START SCANNING PARAMETERS
C
  DO 60 I=1,M
C
C  COMPUTE MAX POS VALUE, CALL VMAXP
C  COMPUTE MAX NEG VALUE, CALL VMAXN
C
  VMAXP = 0.
  VMAXN = 0.
  DO 4 J=1,N
  IF (A(I,J)) 2,3,3
2  IF (A(I,J).GE.VMAXN) GO TO 4
  VMAXN = A(I,J)
  GO TO 4
3  IF (A(I,J).LE.VMAXP) GO TO 4
  VMAXP = A(I,J)
4  CONTINUE
C
C  DETERMINE WHICH OF ABS(VMAXP), ABS(VMAXN) IS LARGER, CALL VMAXPN
C  CALL THE SMALLER ONE VMAXOP
C
  IF (ABS(VMAXP) - ABS(VMAXN)) 5,6,6
5  VMAXPN = VMAXN
  VMAXOP = VMAXP
  GO TO 7
6  VMAXPN = VMAXP
  VMAXOP = VMAXN
C
C  CHECK IF VMAXPN IS NOT NEGLIGIBLE. IF IT IS, DO NOT PLOT THIS PARAMETER, I.E
C  1. MAKE ALL VALUES OF THIS PARAMETER LARGER THAN GP, SAY 2*GP

```

```

C      2. MAKE THE SCALE OF THIS PARAMETER 1.

C      3. GIVE WARNING THAT THE PARAMETER IS NOT PLOTTED
C      4. GO TO CONSTRAINTS PART OF ROUTINE (IF DERIVATIVES ARE COMPUTED, THEY
C      WILL BE FALSELY ZERO. THIS IS NO ESSENTIAL MISTAKE, THE ABSOLUTE
C      VALUES BEING NEGLIGIBLE ANYWAY)

      7 IF (ABS(VMAXPN) - Z), 8,8,10
      8 DO 9 J=1,N
      9 A(I,J) = GP2
      SCAL(I) = 1.
      WAPN(I) = INEGL
      GO TO 20

C
C      DETERMINE ORDER OF VMAXPN, CALL IT IORDER
C      DETERMINE PROVISIONAL VALUE OF SCAL ACCORDING TO IORDER, CALL IT SCALP
C      (RECALL THAT LOG OF A NUMBER SMALLER THAN 1 IS NEGATIVE, THEREFORE THE
C      INTEGER OF THE LOG IS 1 UP THE SCALE, I.E. JUST THE ORDER WE ARE LOOKING FOR
C      ON THE OTHER HAND, FOR NUMBERS LARGER THAN 1, INCREASE THE INTEGER BY 1)
C
      10 IORDER = IFIX(ALOG10(ABS(VMAXPN)))
      IF (ABS(VMAXPN).GT.1.) IORDER = IORDER + 1
      SCALP = 10.**IORDER
      SCALP = SIGN(SCALP,VMAXPN)

C
C      DETERMINE PROVISIONAL SCALED VALUE OF VMAXPN, CALL IT VMAXSC
C      DETERMINE PROVISIONAL SCALED VALUE OF VMAXOP, CALL IT VMOPSC
C      TEST IF VMOPSC FITS INTO GN
C
      VMOPSC = VMAXOP/SCALP
      IF (ABS(VMOPSC) + GN) 12,12,11
      11 SCALP = 10.*SCALP
      12 VMAXSC = VMAXPN/SCALP
      SCAL(I) = SCALP

C
C      TEST IF VMAXSC IS SMALLER THAN GPM OR GPS, CHANGE SCAL(I) ACCORDINGLY
C      TEST AT EACH STEP IF VMOPSC FITS INTO GN
C
      IF (ABS(VMAXSC) - GPM) 13,13,20
      13 SCAL(I) = SCALP*GPM
      IF (ABS(VMAXOP/SCAL(I)) + GN) 14,14,17
      14 IF (ABS(VMAXSC) - GPS) 15,15,20
      15 SCAL(I) = SCALP*GPS
      IF (ABS(VMAXOP/SCAL(I)) + GN) 20,20,16
      16 SCAL(I) = SCALP*GPM
      GO TO 20
      17 SCAL(I) = SCALP

C
C      CONSTRAINT PART OF ROUTINE
C
      20 PPK(I) = VMAXP
      NPK(I) = VMAXN

C
C      IF THE SCALING IS DONE FOR VECTOR X , PRESERVE VMAXPN
C
      IF (IX.NE.1) GO TO 18
      PMDX(I) = VMAXPN

C
C      IF NECESSARY, COMPUTE PEAK POSITIVE AND NEGATIVE DERIVATIVES
C
      18 IF (.NOT.(LUPD(I).OR.LLOD(I))) GO TO 54
      DERM = 0.
      DERMN = 0.
      DO 53 J=2,N
      DER = (A(I,J) - A(I,J-1))/(TP(J) - TP(J-1))

```


IF (DER) 51,52,53

51 IF (DER.GE.DERMN) GO TO 53

DERMN = DER

GO TO 53

52 IF (DER.LE.DERP) GO TO 53

DERP = DER

53 CONTINUE

PPDER(I) = DERP

NPDER(I) = DERMN

C DETERMINE IF PEAK VALUES OF PARAMETERS AND/OR DERIVATIVES VIOLATE
C CONSTRAINTS.

C RECALL THAT THE VALUES HERE ARE DEVIATIONS FROM THE STEADY STATE, HENCE TEST

C UPPER BOUND . GE . (PEAK + STEADY STATE)

C AND LOWER BOUND . LE . (PEAK + STEADY STATE)

54 VIOLB(I) = .FALSE.

VIOLQ(I) = .FALSE.

IF (.NOT.LUPB(I)) GO TO 55

IF ((PPK(I) + SS(I)).GT.UPBNQ(I)) VIOLB(I) = .TRUE.

55 IF (.NOT.LLOB(I)) GO TO 56

IF ((NPK(I) + SS(I)).LT.LCBND(I)) VIOLB(I) = .TRUE.

56 IF (.NOT.LUPD(I)) GO TO 57

IF (PPDER(I).GT.UPDER(I)) VIOLQ(I) = .TRUE.

57 IF (.NOT.LLOD(I)) GO TO 58

IF (NPDER(I).LT.LODER(I)) VIOLQ(I) = .TRUE.

58 IF (.NOT.LPPB(I)) GO TO 60

IF ((PPK(I) - NPK(I)).GT.PPBNQ(I)) VIOLB(I) = .TRUE.

60 CONTINUE

RETURN

21 PRINT 100

STOP

100 FORMAT (1H1///* ABS(GMIN).LT.ABS(GMAX/10)*///* REARRANGE THESE VALU
1ES*)

END

C SUBROUTINE STORE (IS,N,NC,NXTR,NLIM,X,UC,XTR,XP,UCP,XTRP,T,TP,
1 TLAST,NT,NTLAST)

C 22.9.70

C STORES THE VALUES OF X, UC, XTR AND T IN ORDER TO PLOT THEM AFTER HAVING
C COMPUTED THE SIMULATION.

C REAL X(N), UC(NC),XTR(NXTR),XP(N,NLIM),UCP(NC,NLIM),

1 XTRP(NXTR,NLIM),TP(NLIM)

GO TO (10,20), IS

C CLEAR ALL MATRICES BEFORE BEGINNING SIMULATION AND SET INITIAL VALUES

10 CALL NULL (XP,N,NLIM)

CALL NULL (UCP,NC,NLIM)

CALL NULL (XTRP,NXTR,NLIM)

CALL NULL2V (TP,NLIM)

T = TLAST = 0.

NT = NTLAST = 0

RETURN

C STORE VALUES

20 NT = NT + 1

NTLAST = NT

TLAST = T

```

      TP(NT) = T
      DO 21 I=1,N
21  XP(I,NT) = X(I)
      DO 22 I=1,NC
22  UCP(I,NT) = UC(I)
      DO 23 I=1,NXTR
23  XTRP(I,NT) = XTR(I)
      RETURN
      END

```

```

      SUBROUTINE LISTVS (N,NU,NC,ND,X,UC,UD,XS,US,NAMX,NAMU,UNTX,UNTU,
1      IX,NXTR,XTR,XTRS,NAMXTR,NAMXTRV,UNTXTR,UCBAR,KOU,UBUC,LBUC,
2      UCS,NAMUC,UNTUC)

```

5.3.72

PRINTS VALUES OF X, U, XTR, AS WELL AS NAMES, UNITS, S.S. VALUES.
 PRINTS UCBAR AND KOU (SEE EXPLANATION IN MAIN PROGRAM)
 PRINTS CONSTRAINTS ON UC

```

      REAL X(N),UC(NC),UD(ND),XS(N),US(NU),XTR(NXTR),XTRS(NXTR),
1      UCBAR(NC),KOU(NC),UCS(NC),UBUC(NC),LBUC(NC)
      INTEGER NAMX(N),NAMU(NU),UNTX(N),UNTU(NU),NAMUC(NC),UNTUC(NC)
      INTEGER NAMXTR(NXTR),NAMXTRV(NXTR),UNTXTR(NXTR)

```

```

      PPINT 100
      PRINT 101
      DO 1 I=1,ND
1  PPINT 102, I,UD(I),NAMU(I),UNTU(I),US(I)
      PPINT 103
      DO 2 I=1,NC
      CONP = UBUC(I) - UCS(I)
      CONN = LBUC(I) - UCS(I)
2  PRINT 104, I,UC(I),NAMUC(I),UNTUC(I),UCS(I),UCBAR(I),KOU(I),CONP,
1      CONN
      PPINT 105
      DO 3 I=1,N
3  PRINT 106, IX,I,X(I),NAMX(I),UNTX(I),XS(I)
17.9.70
      PRINT 107
      DO 4 I=1,NXTR
4  PRINT 108, NAMXTRV(I),XTR(I),NAMXTR(I),UNTXTR(I),XTRS(I)

```

```

      RETURN
100 FORMAT (10X,'DISTURBANCE INPUTS (UD)*')
101 FORMAT (35X,'NAME',9X,'UNIT',3X,'STEADY STATE VALUE*')
102 FORMAT (10X,'UD(*,I1,*) = ',E11.4,2(3X,A10),3X,E11.4)
103 FORMAT (/10X,'CONTROL INPUTS (UC)*',49X,'UCBAR',11X,'KOU',13X,
1      '*CONSTRAINTS ON UC*/107X,'POSITIVE',6X,'NEGATIVE*')
104 FORMAT (10X,'UC(*,I1,*) = ',E11.4,2(3X,A10),3X,E11.4,2(5X,E11.4,
1      3X,E11.4))
105 FORMAT (/10X,'STATE VARIABLES*')
106 FORMAT (9X,A2,'(*,I2,*) = ',E11.4,2(3X,A10),3X,E11.4)
107 FORMAT (/10X,'ADDITIONAL PARAMETERS*')
108 FORMAT (5X,A10,'* = ',E11.4,2(3X,A10),3X,E11.4)
      END

```

SUBROUTINE COMPOR (C,P,M,N,INC,INR)

C 13.7.70

C
C COMPARES COMPLEX MATRIX C WITH REAL R
C

```

      REAL R(M,N)
      COMPLEX C(I,N)
      INTEGER FRMT(15),RT1CN,RT1OFF,RT2CN,RT2OFF,W1OFF,W1RLCO,W1LRGE,
1  W2OFF,W2RLCO,W2LRGE,W2PO,RFONA,RFONB,RFONC,RFOFFA,RFOFFB,RFOFFC
      LOGICAL PRT1,PRT2,PRT3,PRT4,PRT5
      DATA FRMT /
1  10H(* C(*,I2,, 10H*,*,I2,*), 10H= *,E15.8,,
2  10H*,*,E15.8,, 10H      , 10H4X,      *, 10H      ,
3  10H*,6X,      , 10H      , 10H      , 10H      , 10H      ,
4  10H      , 10H,4X,      *, 10H      , 10H*) /
      DATA RT1CN,RT1OFF,RT2CN,RT2OFF,W1OFF,W1RLCO,W1LRGE,W2OFF,
1  W2RLCO,W2LRGE,W2PO,RFONA,RFONB,RFONC,RFOFFA,RFOFFB,RFOFFC/
2  10H5X,1PE6.0,, 10H11X,      , 10H 5X,1PE6.0, 10H11X      ,
3  10H      , 10HREAL(C)=0., 10HGT. 1.E-4, 10H      ,
4  10HREAL(C)=0., 10HGT. 1.E-4, 10HP = 0.      , 10H*R(*,I2,*),
5  10H*,I2,*),=*, 10H,E16.8,      , 10H26X,      , 10H      ,
6  10H      /

```

C
C THE FORMAT FOR A TYPICAL OUTPUT WITH HIGH RATIO OF IMAG(C)/REAL(C)
C AND HIGH FRACTIONAL DIFFERENCE OF REALS WOULD BE AS FOLLOWS
C

```

(* C(*,I2,*),*,I2,*) = *,E15.8,*,*,E15.8,5X,1PE6.0,4X,*GT. 1.E-4 *,
6X,*R(*,I2,*),*,I2,*) = *,E15.8,10X,1PE6.0,4X,*GT. 1.E-4 *)

```

C THE COMPUTABLE ELEMENTS OF THE VARIABLE FORMAT CORRESPOND AS FOLLOWS
C

```

C RATIO 1   = FRMT(5)   = RT1CN OR RT1OFF
C RATIO 2   = FRMT(12)  = RT2CN OR RT2OFF
C WARNING 1 = FRMT(7)   = W1OFF OR W1RLCO OR W1LRGE
C WARNING 2 = FRMT(14)  = W2OFF OR W2RLCO OR W2LRGE OR W2PO
C REAL FLD  = FRMT(9,10,11) = (RFONA,RFONB,RFONC) OR (RFOFFA,RFOFFB,RFOFFC)
C

```

```

      PRINT 100, INC, IWR

```

```

      PRINT 101

```

```

      Z7 = 1.E-7

```

```

      7 = 1.E-7

```

```

      RZ = 1.E-4

```

```

      DO 1 I=1,M

```

```

      DO 1 J=1,N

```

```

      K1 = K2 = K3 = K4 = 0

```

```

      FRMT(5) = RT1OFF

```

```

      FRMT(12) = RT2OFF

```

```

      FRMT(7) = W1OFF

```

```

      FRMT(14) = W2OFF

```

```

      FRMT(9) = RFOFFA

```

```

      FRMT(10) = RFOFFB

```

```

      FRMT(11) = RFOFFC

```

```

      CRA = ABS(REAL(C(I,J)))

```

```

      CIA = ABS(AIMAG(C(I,J)))

```

```

      RA = ABS(R(I,J))

```

```

      IF (CRA) 12,11,12

```

```

11 IF (RA - ZZ) 25, 25,13

```

```

13 K1 = K3 = 1

```

```

      FRMT(9) = RFONA & FRMT(10) = RFONB & FRMT(11) = RFONC

```

```

      FRMT(14) = W2RLCO

```

```

      GO TO 200

```

```

12 IF (CRA - ZZ) 14,14,16

```

```

C 14 IF (RA - ZZ) 200,200,15

```

```

14 IF (RA - ZZ) 25, 25,15

```

```

25 IF (CIA - Z) 200,200,26

```

```

26 K1 = K3 = 1

```

FRMT(9) = RFON A & FRMT(10) = RFON B & FRMT(11) = RFON C

IF (RA.EQ.0.) FRMT(14) = W2R0

NEW

GO TO 18

15 K1 = K3 = 1

FRMT(9) = RFON A & FRMT(10) = RFON B & FRMT(11) = RFON C

GO TO 200

16 IF (RA) 20,17,20

17 FRMT(14) = W2R0

K1 = K3 = 1

FRMT(9) = RFON A & FRMT(10) = RFON B & FRMT(11) = RFON C

23 IF (CPA - 37) 18,18,19

18 IF (CIA - 32) 200,200,19

19 FRMT(7) = W1LRGE

GO TO 200

20 RTPS = ABS((REAL(C(I,J)) - R(I,J))/REAL(C(I,J)))

IF (RTPS - 7) 200,200,21

21 K1 = K3 = K4 = 1

FRMT(9) = RFON A & FRMT(10) = RFON B

FRMT(11) = RFON C

FRMT(12) = RT20N

IF (RTPS - 32) 200,200,22

22 IF (PA - ZZ) 23,23,24

24 FRMT(14) = W2LRGE

200 PPT1 = K1.EQ.1.AND.K2.EQ.0.AND.K3.EQ.0.AND.K4.EQ.0

PPT2 = K1.EQ.1.AND.K2.EQ.1.AND.K3.EQ.0.AND.K4.EQ.0

PPT3 = K1.EQ.1.AND.K2.EQ.1.AND.K3.EQ.1.AND.K4.EQ.0

PPT4 = K1.EQ.1.AND.K2.EQ.1.AND.K3.EQ.1.AND.K4.EQ.1

PPT5 = K1.EQ.1.AND.K2.EQ.0.AND.K3.EQ.1.AND.K4.EQ.0

PPT6 = K1.EQ.1.AND.K2.EQ.0.AND.K3.EQ.1.AND.K4.EQ.1

IF (PPT1) GO TO 201

IF (PPT2) GO TO 202

IF (PPT3) GO TO 203

IF (PPT4) GO TO 204

IF (PPT5) GO TO 205

IF (PPT6) GO TO 206

GO TO 1

201 PRINT FRMT, I,J,C(I,J)

GO TO 1

202 PRINT FRMT, I,J,C(I,J),RTIM

GO TO 1

203 PRINT FRMT, I,J,C(I,J),RTIM,I,J,R(I,J)

GO TO 1

204 PRINT FRMT, I,J,C(I,J),RTIM,I,J,R(I,J),RTRS

GO TO 1

205 PRINT FRMT, I,J,C(I,J),I,J,R(I,J)

GO TO 1

206 PRINT FRMT, I,J,C(I,J),I,J,R(I,J),RTRS

1 CONTINUE

RETURN

100 FORMAT (1H1, 9X,*MATRICES *,A10,* (COMPLEX) AND *,A10,* (REAL
1) SHOULD BE EQUAL*///10X,*FOLLOWING IS A LIST OF NOTABLE INCONGR
2UENCIES (LET C AND R DENOTE THE COMPLEX AND REAL MATRICES RESP
3ECTIVELY)*///10X,*ELEMENTS OF C AND R WILL BE PRINTED IF ONE O
4R MORE OF THE FOLLOWING CONDITIONS HOLD*///

510X,*1. REAL(C) GT. 1.E-7 AND ((REAL(C)-R)/REAL(C)) GT. 1.E-7*//

610X,*2. (REAL(C) OR IMAG(C) GT.1.E-7) AND (R = 0. OR LT.1.E-7)*//

710X,*3. R GT. 1.E-7 AND (REAL(C) = 0. OR LT. 1.E-7)*///

810X,*NOTES*//15X,*A. COMPARISONS ARE MADE BETWEEN ABSOLUTE VALUES*//

915X,*B. FOR CONDITION (1) THE RATIO IS PRINTED IN THE COLUMN UNDER
A THE TITLE ..FRACT.DIFF(REALS)..*//

918X,*A WARNING IS PRINTED NEXT TO THE RATIO IF IT IS GT. 1.E-4*//

C15X,*C. FOR CONDITION(2), IF REAL(C) OR IMAG(C) IS GT. 1.E-4 A WAR
DNING IS PRINTED NEXT TO THE COMPLEX ELEMENT*////////

101 FORMAT (15X,*REAL PART*,7X,*IMAG PART*,65X,*FRACT.DIFF(REALS)*///)

END

C
C
C

SUBROUTINE WRPBX (A,M,N,IWR)

C
C
C
C
C

31.7.70

WRITES MATRIX OF REAL VECTORS

```

      REAL A(M,N)
      INTEGER JJ(8),IWR(3)
      K = 0
      NN = N
      KK = 1
      IF (NN - 8) 2,2,1
1     K = NN - 8
      NN = 8
2     PRINT 100
      PRINT 101, IWR,M,NN,KK
      L = KK + NN - 1
      KI = 0
      DO 3 I=KK,L
      KI = KI + 1
3     JJ(KI) = I
      PRINT 102, (JJ(I),I=1,NN)
      PRINT 103
      DO 4 I=1,M
4     PRINT 104, I, (A(I,J),J=KK,L)
      IF (K.LE.0) RETURN
      IF (K - 8) 5,5,6
5     NN = K
      KK = KK + 8
      K = 0
      GO TO 2
6     K = K - 8
      KK = KK + 8
      GO TO 2
100  FORMAT (1H1)
101  FORMAT (31X,*MATRIX *,3A10,* IS PARTITIONED INTO DIMENSIONS *,
1     I2,* BY *,I2//31X,*THE ELEMENTS OF THIS MATRIX STARTING AT ROW
2     1 AND COLUMN *,I2,* ARE AS FOLLOWS*///)
102  FORMAT (* COL NO *,I3,7(13X,I3))
103  FORMAT (1H1,* ROW NO*/* ,*/* ,*//)
104  FORMAT (1X,I3,1X,8(E15.8,1X))
      END

```

C
C
C

SUBROUTINE INVCOM (B,A,M,IRANK)

COMPLEX A(M,M),B(M,M)

COMPLEX APIVOT,TEMP

DIMENSION ISWCH(100),JSWCH(100)

IRANK = M

DO 10 I1=1,M

DO 10 I2=1,M

10 A(I1,I2)=B(I1,I2)

C
C
C

INITIALISE PIVOT SWITCHES

DO 1 I=1,M

ISWCH(I)=0

JSWCH(I)=0

1 CONTINUE

```

C
C MAIN LOOP IS EXECUTED M TIMES. FIND PIVOT AND INTERCHANGE CORRESPONDING X AND
C Y, IF NO PIVOT MATRIX IS SINGULAR
C
  DO 2 I=1,M
C
C SEARCH FOR PIVOT
C
  AMAX=0.0
  DO 3 I1=1,M
  DO 4 I2=1,M
    IF((ISWCH(I1).NE.3.OR.JSWCH(I2).NE.0) GO TO 4
    IF (ABS(A(I1,I2)).LE.AMAX) GO TO 4
    IF (CA2S(A(I1,I2)).LE.AMAX) GO TO 4
    IPIVOT=I1
    JPIVOT=I2
C
C AMAX=ABS(A(I1,I2))
C AMAX = CA2S(A(I1,I2))
  4 CONTINUE
  3 CONTINUE
  IF(AMAX.LE.5.000001) GO TO 999
  APIVOT=1./A(IPIVOT,JPIVOT)
  A(IPIVOT,JPIVOT)=APIVOT
  ISWCH(IPIVOT)=JPIVOT
  JSWCH(JPIVOT)=IPIVOT
C
C PIVOT FOUND REDUCE ELEMENTS
C
C
C
C REDUCE ELEMENTS OFF PIVOT ROW AND PIVOT COLUMN
C
  DO 5 I1=1,M
  DO 6 I2=1,M
    IF(I1.EQ.IPIVOT.OR.I2.EQ.JPIVOT) GO TO 6
    DO 6 I2=1,M
      A(I1,I2)=A(I1,I2)-A(IPIVOT,I2)*A(I1,JPIVOT)*APIVOT
  6 CONTINUE
  5 CONTINUE
C
C REDUCE ELEMENTS ON PIVOT ROW AND COLUMN
C
  DO 99 I1=1,M
    IF(I1.EQ.IPIVOT) GO TO 66
    A(I1,JPIVOT)=A(I1,JPIVOT)*APIVOT
  66 IF(I1.EQ.JPIVOT) GO TO 99
    A(IPIVOT,I1)=-A(IPIVOT,I1)*APIVOT
  99 CONTINUE
  2 CONTINUE
C
C INTERCHANGE ROWS AND COLUMNS IF NECESSARY
C
  DO 77 I1=1,M
  11 IF((ISWCH(I1).EQ.I1) GO TO 77
    K=ISWCH(I1)
    DO 88 J=1,M
      TEMP=A(I1,J)
      A(I1,J)=A(K,J)
      A(K,J)=TEMP
  88 CONTINUE
    ITEMP=ISWCH(I1)
    ISWCH(I1)=ISWCH(K)
    ISWCH(K)=ITEMP
    GO TO 11
  77 CONTINUE

```

NEW CARD

NEW CARD

DO 55 I=1,M

12 IF (JSMCH(I).EQ.I) GO TO 55

K=JSMCH(I)

DO 44 J=1,M

TEMP=A(J,I)

A(J,I)=A(J,K)

A(J,K)=TEMP

44 CONTINUE

ITEMP=JSMCH(I)

JSMCH(I)=JSMCH(K)

JSMCH(K)=ITEMP

GO TO 12

55 CONTINUE

RETURN

C

C MATRIX SINGULAR

C

999 CALL SLITE(1)

WRITE(6,200)

200 FORMAT(1H ,15H MATRIX SINGULAR)

IRANK=I-1

RETURN

END

C

C

C

APPENDIX E.

Listing of the more important routines in SIMSAT

(Only routines not common with SIMULA are listed).

SIMSAT

EIGSAT

SATCOD

UPDEIG

UPDVBUL

EXCITSD

PROGRAM SIMSAT (INPUT,OUTPUT,PEPMDTA,TAPE5=INPUT,TAPE6=OUTPUT,
1 TAPE2=PERMDTA)

ORDER OF DATA IS YPO, ICHICK, MI, IKDU, MP, KRED, ISKP5, INC (SET)
TMAX, RTMLIN, DT10, DT100, DT1000, DT10000 (SET)
NCI(I) (SET IF REQUIRED)
KR, KDP (REGULA), XAX1R (MXEV2X1, XAX1U), SSXTR, SSXU
(MXEV2X1, NEWXU), B, A (MXEV2X1), PROACC, COSTS (REG.)
KPFILT (SET IF REQUIRED), KDU (SET IF REQ.).
KR, KDP, PROACC, COSTS, READ FROM TAPE MI IF REQUIRED
XAX1R, SSXTR, SSXU, B, A READ FROM TAPE MP IF REQUIRED

12.6.72 (L. TEPPER, CONTROL SYSTEMS SECTION, DEPARTMENT OF
COMPUTING AND CONTROL, IMPERIAL COLLEGE)

SIMULATION (COMPUTATION AND PLOTTING) OF TRANSIENTS FROM THE EXCITATION OF
STATE VARIABLES OR DISTURBANCE INPUTS.
THE CONTROL INPUTS UC ARE NOT ALLOWED TO VIOLATE CONSTRAINTS.

PART (I) FREE SIMULATION, I.E. NO SATURATION
(IMPLEMENTED IN PROGRAM SIMULA)

IN REGULA WE HAD
 $\dot{X} = A \cdot X + B \cdot U,$ $U = [U(1), \dots, U(10)]^T$ (1)

$Y = C \cdot X$ (2)

BUT ONLY 2 INPUTS WERE USED FOR CONTROL, HENCE CORRECT REPRESENT. IS

$\dot{X} = A \cdot X + B_C \cdot U_C + B_D \cdot U_D,$ $U_C = [U(5), U(6)]^T$ (3)
 $U_D = [U(1), \dots, U(4), U(9), U(10), U(7), U(8)]^T$

AFTER SOLVING THE RICCATI EQUATION FOR THE OPTIMAL REGULATOR..

$\dot{X} = A_{OC} \cdot X + B_{DG} \cdot U_D$ (4)

* $U_C(T) = -K_R \cdot X(T) - K_{DG} \cdot U_D$ * (5)

WHERE

$K_R(NC, N) \equiv$ OPTIMAL FEEDBACK GAIN MATRIX (PRODUCED BY REGULA)
 $A_{OC}(N, N) \equiv$ OPTIMAL STATE MATRIX
 $= A - B_C \cdot K_R$ (6)

$B_{DG}(N, ND) \equiv$ OPTIMAL DRIVING MATRIX
 $= [I - B_C \cdot K_R] \cdot [B_D + B_C \cdot K_{DG}]$ (7)

$K_{DG}(NC, ND) \equiv$ DISTURBANCE GAIN MATRIX (OVERALL)
 $= K_{DR} \cdot B_D + [K_{DR} \cdot B_C - I] \cdot K_{CD}$ (8)

$K_{DR}(NC, N) \equiv$ DISTURBANCE GAIN MATRIX (FROM B) (PRODUCED BY REGULA)
 $K_{CD}(NC, ND) \equiv$ GAIN MATRIX FROM U_D TO U_{CD} (TO BE SET FOR TYPES OF
EXCITATION. MAKING IT GENERAL IS TEDIOUS AND USELESS)

U_{CD} $\equiv$ STEADY STATE DEVIATION (DUE TO U_D) FROM STEADY STATE.
 $= K_{CD} \cdot U_D$

FURTHER ON BELOW, WE SHALL USE FOR SIMPLICITY THE TERM

$K_{DU} \equiv -K_{DG} \cdot U_D$ (8A)

THE SOLUTION OF (3) FOR STEP INPUTS U_C, U_D IS

$X(T) = [EXP(A \cdot T)] \cdot X(0) + [INT(0-T) EXP(A \cdot T_1)] \cdot T_1 \cdot [B_C \cdot U_C + B_D \cdot U_D]$ (9)

G THE SOLUTION OF (4) FOR STEP INPUT UD (AND HENCE UCBAR) IS

G $X(T) = [EXP(AOC*T)]*X(0) + [INT(0-T)[EXP(AOC*T1)]*DT1]*BDG*UD$ (10)

G USING EIGENVALUES/EIGENVECTORS $RTC(A)$, $W(A)$, $V(A)$, AND

G $RTC(AOC)$, $W(AOC)$, $V(AOC)$

G THE SOLUTION (9) IS (OMITTING THE BRACKETED A)

G *****

G * $X(T) = W*[EXP(RTC*T)]*V*X(0) + W*S(T)*V*[PC*UC + PD*UD]$ * (11)

G *****

G WHERE $S(T) = [RTC]INV*[EXP(RTC*T) - I]$

G THE SOLUTION (10) IS (OMITTING THE BRACKETED AOC)

G *****

G * $X(T) = W*[EXP(RTC*T)]*V*X(0) + W*S(T)*V*BDG*UD$ * (12)

G *****

G NOTE... EQ. (5) FOR UC(T) IS USED ONLY IN CONJUNCTION WITH EQ. (12)

G ADDITIONAL PARAMETERS XTR (SEE EXTRA) ARE COMPUTED AS FUNCTIONS OF X

G PART(II) SATURATED SIMULATION. THE CONTROL INPUTS ARE NOT

G ALLOWED TO VIOLATE PHYSICAL CONSTRAINTS.

G (IMPLEMENTED IN THIS PROGRAM, SIMSAT2)

G THE OPTIMALLY CONTROLLED SYSTEM AOC OF EOS. (4,10) IS

G $AOC = A - BC*KR$ (6)

G $= A - BC(1)*[KR(1)]T - PC(2)*[KP(2)]T$ (13)

G FROM EOS. (4,5,8,9A)

G $XDOT = [A - BC(1)*[KR(1)]T - BC(2)*[KP(2)]T]*X + BD*UD +$

G $+ BC(1)*KDU(1) + BC(2)*KDU(2)$ (13A)

G IF UC(2) SATURATES, CALL IT USAT(2), THEN

G $XDOT = [A - BC(1)*[KR(1)]T]*X + BC(1)*KDU(1) + BC(2)*USAT(2) +$

G $+ PD UD$ (14)

G IF UC(1) SATURATES

G $XDOT = [A - BC(2)*[KP(2)]T]*X + BC(2)*KDU(2) + BC(1)*USAT(1) +$

G $+ PD UD$ (15)

G IF UC(1) AND UC(2) SATURATE

G $XDOT = A*X + BC(1)*USAT(1) + BC(2)*USAT(2) + PD*UD$ (16)

G THUS THERE ARE 4 POSSIBLE AOC(L) AND $SUM(L)[BC(I)*KDU(I)]$,

G AS WELL AS 9 POSSIBLE $SUM(0)[BC(J)*USAT(J)]$

G THE GENERAL EXPRESSIONS FOR EOS. (13A,...,16) ARE

G $AOC(L) = A - SUM(L)[BC(I)*[KR(I)]T]$, $L = 1,...,4$ (17)

G $J = 1,...,9$

G J, NE, I

G $XDOT = AOC(L)*X + SUM(0)[BC(J)*USAT(J)] + SUM(L)[BC(I)*KDU(I)] +$

G $+ PD*UD$ (18)

G A SATURATION CODE TABLE FOR AOC(L) AND $SUM(0)$ IS FOUND FURTHER BELOW.

G $SUM(L)[BC(I)*KDU(I)]$ HAS THE SAME CODE NUMBER AS AOC(L).

G AS THE DIGITAL SIMULATION IS DISCRETE, IT IS NECESSARY TO APPLY USAT

G DURING A COMPLETE INTERVAL, REGARDLESS OF THE PRECISE CROSSING POINT

G OF U AND THE SATURATION LINE. AMONG SEVERAL POSSIBLE APPROACHES,

G THE CHOICE HERE IS TO CHECK THE SATURATION STATUS AT THE BEGINNING

G OF EACH TIME INTERVAL. ACCORDINGLY, EQ. (18) IS INTEGRATED OVER AN

C COMPUTE $RTC1(N), W1(N,N), \dots, RTC4(N), W4(N,N)$ FOR $AOC(L)$, EQ.(21)

C COMPUTE $INV(RTC1), INV(W1)$. STORE $RTC1, W1, INV(RTC1), INV(W1)$ IN
C $RTC, W, RTGIN, V$, RESPECTIVELY. CALL EIGSAT FOR THIS STEP.

C 4. SET TYPE OF EXCITATION GIVING A TYPE NUMBER AND CALLING EXCITE.
C IN THIS ROUTINE $X(0)$ AND U (STEP INPUT) ARE SET. ALSO...

C SET $K00$ FOR ANY PARTICULAR KIND OF EXCITATION

C COMPUTE $K0G$ SEE EQ.(4)

C $UCBAR = K00*U0$ (SEE EXPLANATIONS FOLLOWING EQ.(5))

C $K0U = -K0G*U0$ (SEE EQ.(5))

C $B0(1) = A0G*U0$ (SEE EQ.(12))

C $= A0(1)*K0U(1) + A0(2)*K0U(2) + B0*U0$ (SEE EQ.(21))

C $VBU(1) = V(1)*A0G*U0$ (SEE EQ.(21))

C $B0(1), VBU(1)$ ARE FOR THE NON-SAT. CASE, I.E. $SUM(0) = 0$,

C $SUM(L) = A0*K0U$, $KASAT = KSUM9 = 1$

C READ $K0U$ IF $IK0U = 1$, I.E. TEST INFLUENCE OF $K0U$

C COMPUTE $K0U$ IF $IDC = 1$, WITH NEW FORMULA IN ROUTINE DOCCOMP,
C FOR D.C. COMPENSATION

C 5. $UC(0)$ AND $XTR(0)$ CALLING EXTRA3(2,...).

C 6. CHECK IF $UC(0)$ VIOLATES CONSTRAINTS. IF SO, SET NEW CODE NUMBERS
C AND CHANGE UC TO $USAT$. CALL SATCOD FOR THIS STEP.

C 7. PRINT $X(0), U(0), XTR(0), UCBAR, K0U, CONSTRAINTS$, CALLING LISTVS

C 8. SET $TMAX$ (MAX. TIME), $NLIM$ (MAX. NO. OF TIME INTERVALS), T (INITIAL
C TIME), DT (INITIAL TIME INTERVAL). SET ALL ARRAYS IN STORE TO 0
C AND INITIALIZE, CALLING STORE (1,...). PRINT TIME.

C FOR EACH T

C 9. CHECK IF NEW CODE NUMBER($KASNEW$) IS EQUAL TO PREVIOUS($KASAT$). IF
C NOT, UPDATE EIGENVALUES/VECTORS AND INVERSES CALLING UPDEIG.

C 10. CHECK IF NEW CODE NUMBER($KSPNEW$) IS EQUAL TO PREVIOUS($KSUM9$). IF
C NOT, UPDATE VBU CALLING UPDVBU.

C 11. $EXP(PTC*DT)$

C 12. $S = INV(PTC)[EXP(PTC*DT) - I]*VBU$

C 12A IF $RTC(I) = 0$ $S(I) = DT*VBU(I)$

C 13. $X2 = 2ND$ TERM OF EQ. (20). CHECK REAL. PRINT IF $IPR.NE.0$

C 14. $X1 = 1ST$ TERM OF EQ. (20). CHECK REAL. PRINT IF $IPR.NE.0$

C 15. $X = X1 + X2$. PRINT IF $IPR.NE.0$

C 16. UC, XTR , CALLING EXTRA (3,...).

C 17. CHECK IF UC VIOLATES CONSTRAINTS. IF SO, PROCEED AS IN STEP 6.

C 18. PRINT UC, XTR IF $IPR.NE.0$

C 19. STOP THE RESPONSE FOR TIME T.

C 20. SET NEW TIME INTERVAL DT AND GO TO 9.

C 21. PRINT FINAL VALUES OF X, U, XTR , CALLING LISTVAL.

C 22. PRINT THE VALUES OF X, U, XTR FOR ALL T MATRIXWISE, IF $KWRSIM.NE.0$

C 23. PLOT SELECTED PARAMETERS OF X, UC, XTR . PRINT TABLE OF CONSTRAINTS
C AND PEAK VALUES.

C 24. SET NEW TYPE OF EXCITATION AND GO TO 4.

LIST OF SUBROUTINES

EIGSAT	COMPUTES E-VALUES/VECTORS OF AOC FOR VARIOUS MODES OF SATURATION
SATCOD	SETS CODE NUMBER OF SATURATION MODE AND MAKES $UC=USAT$
UPDEIG	UPDATES E-VALUES/VECTORS AND INVERSES WITH CHANGE OF CODE NUMBER
UPDVBU	UPDATES VBU (EQ. 21) WITH CHANGE OF CODE NUMBER
EXCITS	SETS INITIAL EXCITATION $X(0)$ AND/OR STEP INPUT U
EXTRA3	COMPUTES VARIOUS VALUES INVARIANT DURING SIMULATION
ROROUS	COMP., PRINTS PARAMETERS OTHER THAN X , READS REQ. DATA
CONSTR	REORDERS STEADY STATE VALUES, NAMES AND UNITS OF U
PLOTS	SETS CONSTRAINTS AND TESTS VIOLATIONS, PRINTS TABLE
GOH	PLOTS SELECTED PARAMETERS FROM X, XTR AND UC CALLED FROM PLOT FOR EACH LINE OF PRINTING (I.E. EACH TIME T)

```

G   SCALPL1      COMPUTES SCALE OF EACH PARAMETER, TESTS CONSTR. VIOL.

C   STORE        STORES X, UC, XTR AND T FOR ALL T
C   LISTVS       PRINTS INITIAL, FINAL, SS VALUES OF X, U, XTR. ALSO
C               UCRAP,KOU,CONSTRAINTS
C   TREVOR       CHANGES COMPLEX VECTOR INTO REAL IF IMAG PART IS NEGLIGIBLE
C   WRRMX        WRITES REAL MATRIX COLUMNWISE (NO LIMIT ON DIMENSIONS)
C   WRCMX        WRITES COMPLEX MX COLUMNWISE (NO LIMIT ON DIMENSIONS)
C   WRRMXT       EXTENSION OF WRRMX, T HEADING EACH COLUMN, NAME OF
C               ALL PARAMETERS ON EACH SUBMATRIX
C   RCM2013      READS COMPLEX MATRIX (4E20.13)
C   RR42013      READS REAL MATRIX (4E20.13)
C   RCV2013      READS COMPLEX VECTOR (4E20.13)
C   RPV2013      READS REAL VECTOR (4E20.13)
C   RIV8A10      READS INTEGER VECTOR (8A10)
C   CARVUP       TRANSFERS SUBMATRIX FROM PARTITIONED MATRIX
C   MULMXV       MULTIPLIES REAL MATRIX AND REAL VECTOR
C   NULL,MULTI,SUBTR NULL MATRIX, PRODUCT, SUBTRACTION OF MATRICES
C   PORDRM       SWITCHES ROWS OR COLUMNS IN REAL MATRIX
C   NULLRV,NEGVR,ADDV NULL VECTOR, NEGATIVE VR., ADDITION OF VECTORS
C   EQUIVC,EQUIVC EQUIVALENCE OF COMPLEX MATRICES, VECTORS
C   MULCOM       PRODUCT OF TWO COMPLEX MATRICES
C   INVCOM       INVERTS COMPLEX MATRIX
C   MLMXDC       PROD. OF A(SQUARE), J(DIAG.) COMPL. MXS. (A*D OR D*A)
C   MULCMCV      PROD. OF COMPL. MATRIX AND COMPL. COL. VECTOR
C   MULCMPV      PROD. OF COMPL. MATRIX AND REAL COL. VECTOR, RESULT COMPLEX
C   MULDVC       PROD. OF COMPL. DIAGONAL MATRIX AND COMPLEX VECTOR
C   EIGVV1       COMPUTES , PRINTS EIGENVALUES, EIGENVECTORS OF A REAL SQUARE
C               MATRIX, USING FRANCIS. QP ALGORITHM FOR E-VALUES, WILKINSON'S
C               INVERSE ITERATION METHOD FOR E-VECTORS AND THE RAYLEIGH QUOT.
C               SCHEME FOR E-VALUES CORRECTION.
C               THIS IS A MODIFIED VERSION OF EIGSYS (VAN HESS/WEINER, NORTH
C               WESTERN UNIV., EVANSTON, ILL.) AND CALLS THE FOLLOWING
C   HESSEN,OREIG,ORT,ASOR,EIGVEC,DEISP,WRCMX
C   SLOPE1       A NUMERICAL TABLE
C   INTPOL       STRAIGHT INTERPOLATION

C   LIST OF DIMENSIONS AND ARRAYS

C   N = NUMBER OF STATE VARIABLES
C   NU = .. .. INPUTS
C   NC = .. .. CONTROL INPUTS
C   ND = .. .. DISTURBANCE INPUTS
C   NXTP = .. .. ADDITIONAL VARIABLES XTR

C   A(N,N)       STATE MATRIX (BEFORE APPLYING OPTIMAL CONTROL)
C   R(N,NU)       DRIVING MATRIX
C   RTC1(N),...,RTC4(N) COMPLEX E-VALUES OF AOC, IN 4 SATURATION MODES
C   W1(N,N),...,W4(N,N) COMPLEX E-VECTOR MATRICES CORRESP. TO RTC1,...,RTC4
C   RTC(N),RTCI(N) CURRENT RTC(1,...,4) AND ITS INVERSE, COMPLEX
C   W(N,N),V(N,N) CURRENT W (1,...,4) AND ITS INVERSE, COMPLEX
C   RTP(N),RTI(N) AUXILIARY REAL VECTORS FOR EIGSAT
C   A1(N,N),A2(N,N),RW(N,4) AUXILIARY REAL MATRICES FOR EIGVV1
C   R1(N),...,R6(N) AUXILIARY REAL VECTORS FOR EIGVV1
C   RA(N,N)       AUXILIARY COMPLEX MATRIX FOR EIGVV1
C   C1(N),...,C4(N) AUXILIARY COMPLEX VECTORS FOR EIGVV1
C   AD(N,N)       AUX. DOUBLE PRECISION MATRIX FOR EIGVV1
C   D1(N), ,D4(N) AUX. DOUBLE PRECISION VECTRS FOR EIGVV1
C   IROW(N,2)     AUX. INTEGER MATRIX FOR EIGVV1
C   A3(N,N)       AUX. REAL MATRIX FOR EIGSAT
C   SR(N)         AUX. REAL VECTOR FOR UPDVBU
C   BU(N)         REAL, R*U
C   EXPRTC(N)     COMPLEX, EXP(RTC*DT)
C   S(N)          COMPLEX, [RTC]INV*[EXP(RTC*DT)- I]*VBU
C   VRU(N)        COMPLEX, EQ. (25)

```

```

C   RC(N,NC),RD(N,ND),UC(NC),UD(ND)      SEE EQUATION (3)

C   X(N)          INITIAL AND CURRENT VALUES OF X (KEPT FROM ONE STEP
C                   TO THE NEXT)
C   X1(N)         CURRENT VALUES OF 1ST TERM OF EQ.(20) FOR X
C   X2(N)         CURRENT VALUES OF 2ND TERM OF EQ.(20) FOR X
C   XTR(NXTR)     CURRENT VALUES OF XTR, THE ADDITIONAL VARIABLES
C   XS(N),US(NU),UCS(NC),XTPS(NXTR)      STEADY STATE VALUES OF X,U,UC,XTR
C   NAMEX(N),NAMEU(NU),NAMEUC(NC),NAMXTR(NXTR)  NAMES OF X,U,UC,XTR
C   UNITY(N),UNITU(NU),UNITUC(NC),UNITXTR(NXTR)  UNITS OF X,U,UC,XTR
C   NAMXTPV(NXTR) VARIABLE TYPE AND NUMBER AS DEFINED IN PROGRAM MHEV2X1
C   ICASE(NW)     NUMBER OF CASE BEING SIMULATED
C   DATCASE(2)    CASE NUMBER AND POWER LEVEL READ IN ON FORMAT (A2,A3)
C   INAME(8)      NAME OF VECTOR OR MATRIX READ IN ON FORMAT (8A10)
C   XAR(NAR)      XTR VARIABLES DEFINED AS XA IN PROGRAM MHEV2X1
C   XAX1R(NAR,NL) MATRIX IN EQ. XAP=XAX1R*X1, WHERE X1(NL) IS THE VECTOR OF THE
C                   1ST NL COMPONENTS OF X
C   KR(NC,N)      MATRIX IN EQ. UC=KR*X
C   XP(N,NLIM),UCP(NC,NLIM),XTRP(NXTR,NLIM),TP(NLIM)
C                   ARRAYS FOR STORING X, UC, XTR, T FOR ALL T
C   UBUC(NC),LBUC(NC) UP, LOW. BOUNDS OF UC ,REAL
C   UVUC(NC),LVUC(NC) VIOLATION OF UBUC,LBUC ,LOGICAL

C   NAMCOST(65)   NAMES,COSTS OF PENAL. VAR. ON 13 CARDS,FORMAT(5A10)
C   UCBAR(NC)     SETTLING-DOWN VALUE OF UC (DUE TO PERMANENT DIST. UD)
C   KDU(NC)       FIXED(COMP.) ADDED CONTROL(DUE TO PERMANENT DIST. UD)
C   KDUOR(NC)     FIXED(FORG.) ADDED CONTROL(DUE TO PERMANENT DIST. UD)
C   KRFILT(NC,N)  FILTER PATTERN FOR REDUCED SET OF F.B. VAR. (KRED=1)
C   KOR(NC,N)     MATRIX OBTAINED FROM REGULA TO COMPUTE KDU
C   KOD(NC,ND)    MATRIX SET FOR AN EXC. TYPE TO COMPUTE KDU
C   KOG(NC,ND)    SEE EQ.(8)
C   RC(NC,NC),ACD(NC,ND)  AUXILIARY MATRICES FOR ABOVE

C   REAL A(30,30)
C   COMPLEX RTC(30),W(30,30),V(30,30),W1(30,30),W2(30,30),W3(30,30),
1   W4(30,30),RTC1(30),RTC2(30),RTC3(30),RTC4(30)
C   REAL A1(30,30),A2(30,30),RTR(30),RTI(30),RW(30,4),R1(30),R2(30),
1   R3(30),R4(30),R5(30),R6(30)
C   INTEGER IRON(30,2),INAME(8)
C   COMPLEX OA(30,30),C1(30),C2(30),C3(30),C4(30)
C   DOUBLE DA(30,30),D1(30),D2(30),D3(30),D4(30)
C   EQUIVALENCE (W,DA,AD),(C1,D1),(C2,D2),(C3,D3),(C4,D4)
C   COMPLEX RTCIN(30),EXPTC(30),S(30),VRU(30)
C   REAL B(30,10),X2(30),UC(2),UD(3),BC(30,2),BD(30,8),BU(30),X(30),
1   X1(30),US(10),XS(30),UCS(2),A3(30,30),SB(30)
C   REAL UCBAR(2),KDU(2),KDUOR(2)
C   REAL C(10,30),Y(10),Y1(10)      ***FOR THE TIME BEING Y IS NOT USED***
C   INTEGER ICASE(3),NAMEX(30),NAMEU(10),UNITX(30),UNITU(10),NAMEUC(2)
C   INTEGER UNITUC(2),NAMCOST(65),DATCASE(2),KRFILT(2,30)

C   DEFINITIONS FOR EXTRA3 AND EXCITS
C   REAL XTR(13),XTRS(13),KR(2,30),XAR(7),XAX1R(7,27)
C   REAL KOR(2,30),KOD(2,8),KOG(2,8),RC(2,2),ACD(2,8)
C   REAL RC1(2,2),AKDU(2)
C   KOD,KOG,ACD,RC,RC1,AKDU, ARE AUXILIARY FOR ROUTINE EXCITS3D
C   EQUIVALENCE (A1,KOD),(A1(1,2),KOG),(A1(1,3),ACD)
C   INTEGER NAMXTR(14),NAMXTRV(14),UNITXTR(14),NCI(2)

C   DEFINITIONS FOR STORE
C   REAL XP(30,92),UCP(2,92),XTRP(13,92),TP(92)
C   REAL UBUC(2),LBUC(2)
C   LOGICAL UVUC(2),LVUC(2)

C   DATA ICASE/30H      OF REGULA FOR CASE 3/
C   DATA ICASE/30HCASE 7 OF FORCK1 (MHEV2X1) /      FOR A
C   IA = 1 * OPEN LOOP CASE, EQ.(11) / THIS CASE WILL NOT BE RUN ON
C   IA = 2 * CLOS LOOP CASE, EQ.(12) / SIMSAT. IT DOES NOT MAKE SENSE

```

```

C          SINCE FOR UC INACTIVE THERE IS NO POINT IN CONSTRAINING
C
C          IT. IT IS LEFT IN HERE FOR THE CONVENIENCE OF ROUTINES
C          COMPATIBILITY WITH SIMULA
C          IA = 2
C          IPR = 0 * INDIVIDUAL VECTORS FOR EACH TIME      N O T      PRINTED
C          IPPA = 0 * MATRICES  AOC(1),...,AOC(4) NOT PRINTED.
C          IPD = 0 * XAX1R,8,A,20,20 NOT PRINTED
C          IQUICK = 1 * EXC(5) NOT PLOTTED, KR,KOR NOT PRINTED(TO FIT J6 ON CDC)
C          MI = 5 * KR,KOR,PROAOC,COSTS, READ FROM CARDS
C          MI = 3 * KP,KOR,PROAOC,COSTS, READ FROM TAPE 3
C          IKOU = 1 * KOU TO BE READ IN, TO TEST INFLUENCE OF ADDED CONTROL
C          MP = 5 * XAX1R,SSXTR,SSXU,3,A, READ FROM CARDS
C          MP = 2 * XAX1R,SSXTR,SSXU,3,A, READ FROM TAPE 2
C          KRED = 1 * REDUCED SET OF F.3. VAP., HENCE READ FILTER PATTN KRFILT
C          ISKP5 = 1 * EXC(5) NOT RUN AT ALL
C          RTMLIM = REAL TIME LIMIT
C          TMAX = SIMULATION TIME LIMIT
C          DT10 = INTEGRATION INTERVAL AFTER 10 SEC. SIMULATION
C          DT100 = INTEGRATION INTERVAL AFTER 100 SEC. SIMULATION
C          DT1000 = INTEGRATION INTERVAL AFTER 1000 SEC. SIMULATION
C          DT10000 = INTEGRATION INTERVAL AFTER 10000 SEC. SIMULATION
C          ISAT = 0 FREE SIMULATION, I.E. NO SATURATION
C          ISAT = 1
C          READ 121, IPD,IQUICK,MI,IKOU,MP,KRED,ISKP5,IDC
C          READ 125, TMAX,RTMLIM
C          READ 126, DT10,DT100,DT1000,DT10000
C          IF (IQUICK.EQ.1) IPR = 0
C          IPPA = 0
C          IPR = 0
C          N = 30
C          NU = 10
C          NC = 2
C          ND = 8
C          NW = 3
C          NXTP = 13
C          NAR = 7
C          NL = 27
C          NOYD = 8
C          NNAMCOST = 5 * (3+NC+NOYD)
C          IF (IDC.NE.1) GO TO 48
C          READ 127, (NCI(I),I=1,NC)
C
C          IF MP = 5 A CARD MUST BE INSERTED AFTER THE .KEYS. CARD, IN FORMAT
C          122, INCLUDING CASE NUMBER (FOCK1,MXEV2X1) AND PC. LEVEL OF POWER
C
C          48 READ (MP,122) DATCASE
C          PRINT 123, DATCASE
C          PRINT 124, IPD,IQUICK,MI,IKOU,MP,KRED,ISKP5,TMAX,RTMLIM,
C          1 DT10,DT100,DT1000,DT10000,IDC
C          IF (IDC.NE.1) GO TO 49
C          PRINT 128, ((I,NCI(I)),I=1,NC)
C          49 PRINT 129
C
C          READ IN KR(NC,N),KOR(UC,N),XAX1R(NAR,NL),XTRS(NXTR),NAMXTRV(NXTR),
C          NAMXTR(NXTR),UNTXTR(NXTR)
C
C          CALL EXTRA3 (1,INAME,8,N,NU,X,XS,US,NC,ND,UC,U7,NXTR,XTR,XTRS,
C          1 NAMXTR,NAMXTRV,UNTXTR,NAMEU,KR,IA,IPR,NAR,NL,NAR,XAX1R,ICASE,
C          2 NW,IPD,KOR,KOU,MI,MP,IQUICK)
C
C          READ IN XS(N), NAMEX(N), UNITX(N), US(NU), NAMEU(NU), UNITU(NU)
C
C          READ (MP,101) INAME
C          CALL IRV2013 (XS,N,MP)

```

CALL IIV8A10 (NAMEX,N,MP)

CALL IIV8A10 (UNITX,N,MP)
 PRINT 102, INAME
 READ (MP,101) INAME
 CALL IPV2013 (US,NU,MP)
 CALL IIV8A10 (NAMEU,NU,MP)
 CALL IIV8A10 (UNITU,NU,MP)
 PRINT 102, INAME

READ IN B(N,NU), A(N,N).

READ (MP,101) INAME
 CALL IPM2013 (R,N,NU,MP)
 PRINT 102, INAME
 IF (IPD) 3,5,3
 3 CALL WRRMX (R,N,NU,30H3(BEFORE REORDERING))
 5 READ (MP,101) INAME
 CALL IPM2013 (A,N,N,MP)
 PRINT 102, INAME
 IF (IPD) 10,26,10
 10 CALL WRRMX (A,N,N,30HSTATE (INPUT))

READ IN PROAOC (1 CARD), COSTS (13 CARDS).

26 IF (IA - 1) 27,28,27
 27 READ (MI,115) PROAOC
 READ (MI,114) (NAMCOST(I),I=1,NNAMCST)
 28 IF (EOF,MI) 1,2
 1 PRINT 103
 STOP
 2 PRINT 104

CHECK IF A REDUCED SET OF FEEDBACK VARIABLES IS REQUIRED INSTEAD
 OF THE FULL SET. IF SO, CHANGE MATRIX KP READING FILTER PATTN KRFILT

IF (KPED - 1) 45,29,45
 29 CALL FILTER (NC,N,KP,KRFILT,NAMEX,1)
 45 IS = 10HACC(30,30)
 IS = 10HA (30,30)

REORDER MATRIX B (CHANGE COLUMNS 5,6 FOR 9,10). BREAK UP B INTO BD AND BC
 REORDER US (CHANGE 5,6 FOR 9,10)

CALL RORDPM (1,3,N,NU,5,9)
 CALL RORDPM (1,3,N,NU,6,10)
 CALL CARVUP (B,BD,N,NU,N,NQ,1,1)
 CALL CARVUP (B,BC,N,NU,N,NQ,1,9)
 IF (IPD) 46,47,46
 46 CALL WRRMX (BD,N,NQ,30HBD (DISTURBANCE INPUT MATRIX))
 CALL WRRMX (BC,N,NQ,30HBC (CONTROL INPUTS MATRIX))
 47 CALL RORDUS (NU,US,NAMEU,UNITU,NAMEUC,NQ,NQ,UCS,UNITUC)

CHECK UC

THE FOLLOWING STATEMENT CHECKS DATA ELEMENTS.
 SUBROUTINE CHDATA IS PARTICULAR TO THIS SYSTEM FOR CASE (3) OF
 FORCK1(MXEY2X1) AND 100 PC. POWER LEVEL. IT MUST BE CHANGED FOR ANY
 CHANGE IN INPUT. THE OBJECT IS TO FLAG A MISTAKE IN DATA RATHER THAN
 HAVING LONG PRINTOUTS, FOR PRINTOUT ECONOMY'S SAKE.

CALL CHDATA (BC,N,NC,NAMCOST,NNAMCST)

KASAT = KASNEW = 1
 KSUNR = KSNEW = 1

COMPUTE EIGENVALUES/VECTORS FOR AOC (OPTIMALLY CONTROLLED) FOR THE

C 4 FOLLOWING POSSIBILITIES, IN 3 OF WHICH ONE OR BOTH CONTROL INPUTS
C ARE SATURATED.

C NO SATURATION, AOC(1) = A - RC*KR COMPUTE W1,RTC1,V,RTICIN
C UC(2) SATURATED, AOC(2) = A - RC(1)*(KR(1))T COMPUTE W2,RTC2
C UC(1) SATURATED, AOC(3) = A - RC(2)*(KR(2))T COMPUTE W3,RTC3
C UC SATURATED, AOC(4) = A COMPUTE W4,RTC4

C CALL EIGSAT (N,A,A1,A2,A3 ,RTR,PTI,PTC,RTC1,RTC2,RTC3,RTC4,
1 W,W1,W2,W3,W4,V,PTICIN,R1,P2,R3,P4,P5,P6,RW, 4,IROW, 2,DA,C1,
2 C2,C3,C4,AD,D1,D2,D3,D4,RC,KR,NC,IPRA)

C A3 IS NOW AOC(1) = A - RC*KR
C W,V,RTC,RTICIN ARE NOW W1,INV(W1),RTC1,INV(RTC1), RESPECTIVELY.

C COMPAE PRAOC (PRODUCT OF COMPUTED E-VALUES OF AOC) WITH
C PROAOC (READ-IN PRODUCT OF E-VALUES OF AOC) AND PRINT

C PRAOC = 1.
C DO 51 I=1,N
51 PRAOC = PRAOC*RTC1(I)
C RELPP = (PRAOC - PROAOC)/PROAOC
C PRINT 119, PRAOC, PROAOC, RELPP
C IF (KRED.EQ.1) GO TO 38
C IF (ABS(RELPP) - 1.E-5) 53,53,52
52 PRINT 105
C STOP
53 PRINT 104

C SET TYPE OF EXCITATION AND TEST IF KDU IS REQUIRED TO BE CHANGED
C FOR TESTING PURPOSES. IF SO, READ IN KDU.

C 38 ITYPE = 6
C 4 CALL EXCITSN (X,UC,UD,N,NC,ND,IXN,IUC,IUD,IS,ICASE,NW,RC,DD,BU,R2,
1 V,VBU, IEXC,XS,US,NU, ITYPE,KDR,UCBAR,KDU,KOD,KOG,
2 AGD,PC,KDUOC,IKDU,A3,A1,A2,PC1,NCI,AKDU,IDC)

C VBU IS NOW INV(W1)*BGG*UD, SEE EQ.(12)

C BU = BGG*UD = [BO*UD + BO*KDU] / VALID ONLY FOR L = KASAT = 1
C VBU = V*BU / Q = KSUMB = 1

C I.E. WHEN THE CONTROL VECTOR IS WITHIN THE CONSTRAINTS

C PRINT 111, IEXC,IS,ICASE,ITYPE
C PRINT 112
C IF (IUC.EQ.0) GO TO 25
C PRINT 118
25 CALL EXTRAT(2,INAME,9,N,NU,X, XS,US,NC,ND,UC,UD,NXTR,XTR,XTRS,
1 NAMXTR,NAMXTRV,UNTXTR,NAMEN,KR,IA,IPR,NAR,NL,XAR,XAX1R,ICASE,
2 NW,IPD,KDR,KDU,MI,MP,IQUICK)

C SET CONSTRAINTS ON CONTROL INPUTS UC

C CALL PLOTS (N,NC,NXTR,2, XP,UCP,XTRP,TP,TLAST,ND,NU,XS,UCS, *ADJUST*
1 XTRS,NAMEX,NAMXUC,NAMXTR,NAMXTRV,UNITX,UNITUC,UNTXTR,IEXC,IS, * WITH *
2 ICASE,NW,IA,ITYPE,IQUICK,KP,UBUC,LBUC,1) * EXTRA*

C CHECK IF UC VIOLATES CONSTRAINTS AND IF SO, CHANGE UC TO THE
C CONSTRAINED(SATURATED) VALUE. SET CORRESPONDING CODE NUMBERS
C KASNEW AND KSRNEW.

C IF (ISAT) 41,40,41
41 CALL SATCOD (NC,UC,UBUC,LBUC,UCS,UVUC,LVUC,KASNEW,KSRNEW)

GO TO 42

```

40 KASAT = KASNEW = KSUMB = KSBNEW = 1
42 CALL LISTVS (N,NU,NC,N7,X,UC,UD,XS,US,NAMEX,NAMEU,UNITX,UNITU,
1      2HYD,NXTR,XTR,XTRS,NAMXTR,NAMXTRV,UNTXTR,UCBAR,KDU,
2      UBUC,LBUC,UCS,NAMEUC,UNITUC)

```

SET TIME INTERVALS (LONGEST TIME CONSTANT OF A IS OF ORDER 2000 SECONDS)

```

      T = 0.
      DT = 0.
      IF (ITYPE.EQ.5) TMAX = 1000.
      NLIM = 92
      CALL STORE (1,N,NC,NXTR,NLIM,X,UC,XTR,XP,UCP,XTPD,T,TP,TLAST,NT,
1      NTLAST)

```

PROCEED COMPUTING FOR EACH T (STEPS 9 TO 20)

```

6 CALL SECOND (RTIME)
  IF (RTIME - RTMLIM) 37,14,14

```

CDC 6630

```

37 IF (IPR) 16,12,16
12 IF (T) 17,13,17
13 PRINT 116, T, RTIME, KASNEW, KSBNEW
  GO TO 17
16 PRINT 106, T
17 IF (ISAT) 39,33,39
39 IF (KASAT - KASNEW) 30,31,30
30 KASAT = KASNEW

```

UPDATE EIGENVALUES/VECTORS AND INVERSES, I.E.
 REPLACE W,RTC FROM W1,...,W4,RTC1,...,RTC4, ACCORDING TO KASAT
 COMPUTE V,RTG FROM W,RTC

```

      CALL UPDEIG (N,V,RTGIN,W1,W2,W3,W4,RTC1,RTC2,RTC3,RTC4,
1      W,RTC,KASAT)

```

```

31 IF (KSUMB - KSBNEW) 32,33,32
32 KSUMB = KSBNEW
  PRINT 117, T, RTIME, KASAT, KSUMB
  PRINT 120, ((I,RTC(I)),I=1,N)

```

UPDATE, I.E. COMPUTE VBU = V*(SB + BU)
 FOR ONE OF THE 9 POSSIBLE VALUES OF SB, ACCORDING TO KSUMB
 CALL UPDVBU1 (N,NC,V,BU,VBU,KASAT,UC,BC,SB,A1,A2,NB,ND,UD,KDU,R2)

```

33 DO 7 I=1,N
  EXPRTC(I) = CEXP(RTC(I)*DT)
  IF (CABS (RTC(I)) - 1.E-10) 8,9,9
  8 S(I) = DT*VBU(I)
  GO TO 7
  9 S(I) = RTCIN(I)*(EXPRTC(I) - 1.)*VBU(I)
7 CONTINUE
  CALL MULCMGV (W,S,C4,N,N)

```

C4 IS NOW X2(T), 2ND TERM OF EQ. (24), AND SHOULD BE REAL. CHECK
 IF IT IS AND CONSTRUCT REAL.

```

      IF (IPR) 19,20,19
19 PRINT 107
20 CALL TREVOR (C4,X2,N,10HX2(T),NAMEX,IPR)
  CALL MULCMRV (V,X,C4,N,N)
  CALL MULDVC (EXPRTC,C4,C2,N)
  CALL MULCMGV (W,C2,C4,N,N)

```

```

C  C4 IS NOW X1(T), 1ST TERM OF EQ. (24), AND SHOULD BE REAL. CHECK IF
C  IT IS AND CONSTRUCT REAL.
C
  IF (IPP) 21,22,21
21 PRINT 108
22 CALL TREVCR (C4,X1,N,10HX1(T) ,NAMEX,IPR)
  CALL ADDV (X2,X1,X,N)
  IF (IPP) 18,11,18
18 PRINT 109, T
  PRINT 110, ((I,X(I),NAMEX(I)),I=1,N)
11 CALL EXTPA3 (3,NAME,N,N,X,XS,US,NO,NO,UC,UD,NXTR,XTR,XTRs,
  1 NAMXTR,NAMXTRV,UNTSTR,NAMEU,KR,IA,IPP,HAR,NL,XAR,XAX1R,ICASE,
  2 NW,IPD,KOR,KOU,MT,MP,IQUICK)
C
C CHECK IF UC VIOLATES CONSTRAINTS AND IF SO, CHANGE UC TO THE
C CONSTRAINED(SATURATED) VALUE. SET CORRESPONDING CODE NUMBERS
C KASNEW AND KS3NEW.
C
  IF (ISAT) 43,44,43
43 CALL SATCOD(NC,UC,UBUC,LBUC,UCS,UBUC,LVUC,KASNEW,KS3NEW)
44 TE = T + 1.E-4
C
  IF (T - 10.) 35,35,34
  IF (T - 1. ) 35,35,34
34 TLAST = TLAST + 1.E-4
  IT = TE
  ITL = TLAST
  IF (IT - ITL - IDS) 36,35,36
35 CALL STOPE (2,N,NC,NXTR,NLIT,X,UC,XTR,XP,UCP,XTPP,T,TP,TLAST,NT,
  1 NTLAST)
36 IF (NTLAST.GE.NLIT) GO TO 14
  IF(TE.GE.TMAX) GO TO 14
  IF(TE.GE. 0.) DT = .01
  IF(TE.GE. .3) DT = .05
  IF(TE.GE. 1.) DT = .1
  IF(TE.GE. 1.) IOS = 1
  IF(TE.GE. 10.) DT = DT10
  IF(TE.GE. 10.) IOS = 10
  IF(TE.GE. 100.) DT = DT100
  IF(TE.GE. 100.) IOS = 100
  IF(TE.GE.1000.) DT = DT1000
  IF(TE.GE.2000.) IOS = 1000
  IF(TE.GE.10000.) DT = DT10000
  IF(TE.GE.10000.) IOS =10000
  T = T + DT
  GO TO 6
C
C PRINT FINAL VALUES OF X(T)
C
14 PRINT 117, T,RTIME,KASNEW,KS3NEW
  PRINT 111, IEXC,IS,ICASE,ITYPE
  PRINT 113, T
  CALL LISTVS (N,NU,NC,ND,X,UC,UD,XS,US,NAMEX,NAMEU,UNITX,UNITU,
  1 2H X,NXTR,XTR,XTRs,NAMXTR,NAMXTRV,UNTSTR,UCBAP,KOU,
  2 UBUC,LBUC,UCS,NAMEUC,UNITUC)
C
C PRINT ALL VALUES IN MATRIX FORM
C KWSIM = 0 - RESULTS OF SIMULATION NOT PRINTED (PLOTED ONLY)
C
  KWSIM = 0
  IF (KWSIM.EQ.0) GO TO 24
  CALL WRPHXT (XP, N, NTLAST,30HX(T) FROM T = 0. TO T = ,
  1 TLAST,TP,10H SECONDS ,NAMEX,0)
  CALL WRPHXT (UCP, NC, NTLAST,30HUC(T) FROM T = 0. TO T = ,
  1 TLAST,TP,10H SECONDS ,NAMEUC,0)

```

CALL WRRMXT (XTRP,NXTP,NTLAST,30HXTR(T) FROM T = 0. TO T = ,

```

1      TLAST,TP,10H SECONDS ,NAMXTR,0)
24 CALL PLOTS (N,NC,NXTR,NTLAST,XP,UCP,XTRP,TP,TLAST,ND,NU,XS,UCS, *ADJUST*
1      XTRS,NAMEX,NAMENC,NAMXTP,NAMXTPV,UNITX,UNITUC,UNTXTR,IEXC,IS, * WITH *
2      ICASE,NW,IA,ITYPE,IQUICK,KP,UBUC,LBUC,0) * EXTRA*
PRINT 114, (NAMCOST(I), I=1,VNAMCOST)
IF (ISKP5.E0,1) GO TO 1000
IF (ITYPE - 5) 23,1000,23
23 ITYPE = 5

```

C TEST IF THIS IS A RUN WITH IMPOSED KOU. IF SO, MISS EXC(5)

C IF (IKOU - 1) 4,1000,4
1000 STOP

C

```

100 FORMAT (//////10X,*REAL TIME IS*,F11.3,*SEC*)
101 FORMAT (8A10)
102 FORMAT (10X,8A10)
103 FORMAT (10X,*END OF FILE CARD READ AS DATA*//10X,*ONE OR MORE DATA
1 CAPDS MISSING*)
104 FORMAT (10X,*DATA O.K.*////////)
105 FORMAT (////10X,*DATA WRONG. RUN AGAIN WITH IP0 = 1*)
106 FORMAT (1H1, 9X,*T = *,F11.3,* SECONDS*)
107 FORMAT (/10X,*THE FOLLOWING VECTOR X2(T) IS THE 2ND TERM OF EQ.
1 (24) IN MAIN PROGRAM, I.E. THE RESPONSE*//10X,*OVER THE LAST INTE
2RVAL ONLY) TO THE EXCITATION OF THE DISTURBANCE INPUTS AND THE SAT
3URATED CONTROL INPUTS (IF ANY)*)
108 FORMAT (////10X,*THE FOLLOWING VECTOR X1(T) IS THE 1ST TERM OF EQ
1. (24) IN MAIN PROGRAM, I.E. THE RESPONSE*//10X,*TO THE STATE EXCIT
2ATION, TAKEN AS IT WAS IN THE LAST CALCULATED INSTANT*)
109 FORMAT (////10X,*FOLLOWING IS THE RESPONSE X(T) FOR T =*,F11.3,
1* SECONDS.*///)
110 FORMAT (15X,*X(T) (*,I2,*) = *,E11.4,5X,A10)
111 FORMAT (1H1, 9X,A10,*RESPONSE OF SYSTEM *,A10,*, *,3A10,*, EXC T
1YPE (*,I2,*)*///)
112 FORMAT (10X,*FOLLOWING IS THE LIST OF VALUES OF THE INITIAL STATE
1EXCITATION (X0) AND THE DISTURBANCE INPUTS (U0)*///10X,*THE VALU
2ES OF THE CONTROL INPUTS (UC) AND ADDITIONAL PARAMETERS ARE THE
3RESULTS OF THE CONTROL LAW FOR THIS*//10X,*CASE AND OF THE ALGEBRAI
4C RELATIONS IN THE BASIC MODEL, RESPECTIVELY. THEY DO NOT RE
5PRESENT EXCITATION*//10X,*OF THE SYSTEM (UNLESS IT IS SPECIFICALLY
6STATED TO BE SO)*///10X,*THE VALUES OF UC ARE LIMITED BY CONSTRA
7INTS (PRINTED HERE) IN THIS INITIAL LIST AND ALL ALONG THE SIMULAT
8ION*/// 10X,*THE STEADY STATE VALUES ARE ALSO PRINTED,
9 AS ARE UCRAR = STEADY STATE DEVIATION OF UC FROM S.S. VALUE,*//
A 69X,40HIN THE PRESENCE OF UD (UCRAR = KOU*UD)//
B 63X,*KOU = FIXED AMOUNT OF CONTROL TO BE ADDED TO THE FEEDBAC
CK*// 69X,46H(UC(T) = - KR*X(T) + KOU = - KR*X(T) - KDG*UD)//
D61X,*CONSTRAINTS ON UC*///)
113 FORMAT (10X,*F I N A L V A L U E S (T = *,F11.3,* SECONDS)*///)
114 FORMAT (5A10)
115 FORMAT (50X,E20.13)
116 FORMAT (1H1,9X,*SIMULATION COMPUTING STARTS. TIMES T ARE PRINTED
1.*//10X,*ALL ELEMENTS SHOULD BE REAL*//10X,*THOSE ELEMENTS (IF ANY)
2 WHICH HAVE IMAG PART GT.1.E-7 AND (IMAG/REAL RATIO GT.1.E-4 OR RE
3AL ZERO) ARE PRINTED*//10X,*X2(T) ARE ELEMENTS REPRESENTING THE 2ND
4 TERM OF EQ. (24) IN MAIN PROGRAM*//10X,*X1(T) ARE ELEMENTS REPRESE
5NTING THE 1ST TERM OF EQ. (24) IN MAIN PROGRAM*//10X,*WHERE*//10X,
6121HX(K+1) = W(L)*(EXP(PTC(L)*DT(K)))1*V(L)*X(K) + W(L)*PTCIN(L)*[E
7XP(PTC(L)*DT(K)) - 1]*V(L)*[SUM(O)(PC(J)*USAT(J)) + 90*UD] //10X,
8*WHERE*//15X,*USAT(J) = VALUE OF UC(J) CORRESPONDING TO UPPER OR LO
9WER CONSTRAINT*//15X,* UD = VECTOR OF DISTURBANCE INPUTS*///29X
A,*REAL PART*,32X,*IMAG PART*///10X,*T = *,F11.3,* SEC*,
B 20X,*REAL TIME IS*,F11.3,* SEC*,

```

C 15X,*KASAT = *,I1,5X,*KSUMB = *,I1)

```

117 FORMAT (10X,*T = *,F11.3,* SEC*,20X,*REAL TIME IS*,F11.3,* SEC*,
1 15X,*KASAT = *,I1,5X,*KSUMB = *,I1)
118 FORMAT (10X,*WARNING ... . THE CONTROL INPUTS (UC) HAVE BEEN E
1XCITED*///)
119 FORMAT (///10X,*PRODUCT OF COMPUTED EIGENVALUES OF AOC =*,E20.13/
1 10X,*READ-IN PRODUCT OF EIGENVALUES OF AOC =*,E20.13/
2 10X,*FRACTIONAL DIFFERENCE =*,E20.13/)
120 FORMAT (///31X,*NEW EIGENVALUES*///23X,*REAL PART*,7X,*IMAG PART*
1 //(10X,*ROOT (*,I2,*)*,2E16.9))
121 FORMAT (8I1)
122 FORMAT (A2,A3)
123 FORMAT (1H1///10X,*DATA INPUT TO SIMSAT , OUTPUT FROM MXEV2X1*/
1/10X,*CASE (*,A2,*) OF F00CK1 , POWER LEVEL = *,A3,* PG,*)
124 FORMAT (///10X,*IPD = *,I1,* + (0 IMPLIES XAX1R,R,A,AOC,RD,
1BC N O T PRINT.)*//10X,*IQUICK = *,I1,* + (1 IMPLIES EXC(5) N O
2T PLOTTED)*//10X,*MT = *,I1,* + (5 IMPLIES KR,KDR,PROAOC,COSTS
3, READ IN FROM CARDS, 3 FROM TAPE 3)*//10X,*IKDU = *,I1,* + (1 IM
4PLIES KDU TO BE READ IN, TO TEST INFLUENCE OF FIXED ADDED CONTROL)
5*//10X,*MP = *,I1,* + (5 IMPLIES XAX1R,SSXTR,SSYU,R,A, READ IN
6FROM CARDS, 2 FROM TAPE 2)*//10X,*KRED = *,I1,* + (1 IMPLIES N O
7 T ALL STATE VARIABLES FED BACK, HENCE FILTER PATTERN TO BE READ
8IN)*//10X,*ISKPS = *,I1,* + (1 IMPLIES EXC(5) NOT RUN AT ALL)*//10X
9,*TMAX = *,F7.0,* (SIMULATION TIME LIMIT, SECONDS)*//10X,*RTMLIM =
A *,F6.0,* (REAL TIME LIMIT, SECONDS)*//
10X,*DT10 = *,F8.1,* (INTEGR. INTERVAL AFTER 10 SEC. SIMUL.)*//
10X,*DT100 = *,F8.1,* (INTEGR. INTERVAL AFTER 100 SEC. SIMUL.)*//
10X,*DT1000 = *,F8.1,* (INTEGR. INTERVAL AFTER 1000 SEC. SIMUL.)*//
10X,*IDC = *,I1,* + (1 IMPLIES DC COMPENSATION USED, RECOMPUTIN
GG KDU, SUCH THAT X(NCI(I)) = 0)*///)
125 FORMAT (6X,F7.0,20X,F6.0)
126 FORMAT (4(10X,F10.1))
127 FORMAT (8(8X,J2))
128 FORMAT (10X,*NCI(*,I1,*) = *,I2,* (SEE EXPLANATION FOR IDC = 1
1)*)
129 FORMAT (///)
END

```

C
C
C

```

SUBROUTINE SIGSAT (N,A,A1,A2,AOC,RTR,RTI,RTC,RTC1,RTC2,RTC3,RTC4,
1 W,W1,W2,W3,W4,V,PTCIN,R1,R2,R3,R4,R5,R6,RW,N4,IROW,N2,DA,C1,
2 C2,C3,C4,A0,D1,D2,D3,D4,DC,KR,NC,K)

```

C
C
C

21.1.71

C
C
C
C
C
C

COMPUTE EIGENVALUES/VECTORS FOR AOC (OPTIMALLY CONTROLLED) FOR THE
4 FOLLOWING POSSIBILITIES, IN 3 OF WHICH ONE OR BOTH CONTROL INPUTS
ARE SATURATED.

C
C
C
C
C
C
C

```

NO SATURATION, AOC(1) = A - DC*KR COMPUTE W1,RTC1,V,RTCIN
UC(2) SATURATED, AOC(2) = A - DC(1)*(KR(1))T COMPUTE W2,RTC2
UC(1) SATURATED, AOC(3) = A - DC(2)*(KR(2))T COMPUTE W3,RTC3
UC SATURATED, AOC(4) = A COMPUTE W4,RTC4

```

C

```

REAL A(N,N),A1(N,N),A2(N,N),AOC(N,N),RTR(N),RTI(N),R1(N),R2(N),
1 P3(N),P4(N),P5(N),P6(N),PW(N,N4),BC(N,NC),KP(NC,N)
COMPLEX RTC(N),RTC1(N),RTC2(N),RTC3(N),RTC4(N),W(N,N),W1(N,N),
1 W2(N,N),W3(N,N),W4(N,N),V(N,N),PTCIN(N),DA(N,N),C1(N),C2(N),
2 C3(N),C4(N)
DOUBLE AD(N,N),D1(N),D2(N),D3(N),D4(N)
INTEGER IROW(N,N2)

```

C

C AOC(4)

C

```

PRINT 104
CALL EIGVV1 (3,0,0,0,K,N,A, A1,A2,RTR,RTI,RTC4,W4,R1,R2,R3,R4,RW,
1 IROW,R5,R6,DA,C1,C2,C3,C4,AD,D1,D2,D3,D4)
CALL INVCOM (W4,V,N,IRANK)
IWA = 10HAOC4(SA)=A
PRINT 100, IWA,N,N,IRANK
IF (IRANK.NE.N) STOP

```

C

C

C

AOC(3)

```

PRINT 103
CALL CAPVUP (BC,A1,N,NC,N,1,1,2)
CALL CAPVUP (K3,A2,NC,N,1,N,2,1)
CALL MULTI (A1,A2,AOC,N,1,N)
CALL SUBTR (A,AOC,AOC,N,N)
CALL EIGVV1 (3,0,0,0,K,N,AOC,A1,A2,RTR,RTI,RTC3,W3,P1,P2,R3,R4,RW,
1 IROW,R5,R6,DA,C1,C2,C3,C4,AD,D1,D2,D3,D4)
CALL INVCOM (W3,V,N,IRANK)
IWA = 10HAOC3(SAT)
PRINT 100, IWA,N,N,IRANK
IF (IRANK.NE.N) STOP

```

C

C

C

AOC(2)

```

PRINT 102
CALL CAPVUP (BC,A1,N,NC,N,1,1,1)
CALL CAPVUP (K3,A2,NC,N,1,N,1,1)
CALL MULTI (A1,A2,AOC,N,1,N)
CALL SUBTR (A,AOC,AOC,N,N)
CALL EIGVV1 (3,0,0,0,K,N,AOC,A1,A2,RTR,RTI,RTC2,W2,R1,R2,R3,R4,RW,
1 IROW,R5,R6,DA,C1,C2,C3,C4,AD,D1,D2,D3,D4)
CALL INVCOM (W2,V,N,IRANK)
IWA = 10HAOC2(SAT)
PRINT 100, IWA,N,N,IRANK
IF (IRANK.NE.N) STOP.

```

C

C

C

AOC(1)

```

PRINT 101
CALL MULTI (BC,KR,AOC,N,NC,N)
CALL SUBTR (A,AOC,AOC,N,N)
CALL EIGVV1 (3,0,0,0,K,N,AOC,A1,A2,RTR,RTI,RTC1,W1,R1,R2,R3,R4,RW,
1 IROW,R5,R6,DA,C1,C2,C3,C4,AD,D1,D2,D3,D4)
CALL INVCOM (W1,V,N,IRANK)
IWA = 10HAOC1 = AOC
PRINT 100, IWA,N,N,IRANK
IF (IRANK.NE.N) STOP

```

C

```

DO 1 I=1,N
1 RTG1(I) = 1./RTC1(I)
CALL EQUVC (RTC1,RTG,N)
CALL EQUVC (W1,W,N,N)
RETURN
100 FORMAT (///10X,*RANK OF EIGENVECTOR MATRIX OF *,A10,*(*,I2,*,*,
1 I2,*) IS*,I6///)
101 FORMAT (1H1,9X,*EIGENVALUES OF AOC(1) = AOC*////////)
102 FORMAT (1H1,9X,*EIGENVALUES OF AOC(2) = AOC (UC(2) SATUR.)*////////)
103 FORMAT (1H1,9X,*EIGENVALUES OF AOC(3) = AOC (UC(1) SATUR.)*////////)
104 FORMAT (1H1,9X,*EIGENVALUES OF AOC(4) = A (UC(1,2) SATUR.)*////////)
END

```

C

C

SUBROUTINE SATCOD (NC,UC,UBUC,LBUC,UCS,UBUC,LVUC,KASNEW,KSRNEW)

21.1.71

CHECK IF UC VIOLATES CONSTRAINTS AND IF SO, CHANGE UC TO THE
CONSTRAINED (SATURATED) VALUE. SET CORRESPONDING CODE NUMBERS
KASNEW AND KSRNEW.

SATURATION CODE TABLE

UBUC(1)	LBUC(1)	UBUC(2)	LBUC(2)	KASAT	KSUMB
---	---	---	---	1	1
---	---	VIOL.	---	2	2
---	---	---	VIOL.	2	3
VIOL.	---	---	---	3	4
---	VIOL.	---	---	3	5
VIOL.	---	VIOL.	---	4	6
VIOL.	---	---	VIOL.	4	7
---	VIOL.	VIOL.	---	4	8
---	VIOL.	---	VIOL.	4	9

REAL UC(NC),UBUC(NC),LBUC(NC),UCS(NC)

LOGICAL UVUC(NC),LVUC(NC)

DO 10 I=1,NC

UVUC(I) = (UC(I) + UCS(I)).GE.UBUC(I)

10 LVUC(I) = (UC(I) + UCS(I)).LE.LBUC(I)

IF (.NOT.(UVUC(1).OR.LVUC(1).OR.UVUC(2).OR.LVUC(2))) GO TO 1
IF (.NOT.(UVUC(1).OR.LVUC(1)).AND.UVUC(2)) GO TO 2
IF (.NOT.(UVUC(1).OR.LVUC(1)).AND.LVUC(2)) GO TO 3
IF (.NOT.(UVUC(2).OR.LVUC(2)).AND.UVUC(1)) GO TO 4
IF (.NOT.(UVUC(2).OR.LVUC(2)).AND.LVUC(1)) GO TO 5
IF (UVUC(1).AND.UVUC(2)) GO TO 6
IF (UVUC(1).AND.LVUC(2)) GO TO 7
IF (LVUC(1).AND.UVUC(2)) GO TO 8
IF (LVUC(1).AND.LVUC(2)) GO TO 9

1 KASNEW=1 & KSRNEW=1 & RETURN
2 KASNEW=2 & KSRNEW=2 & UC(2)=UBUC(2)-UCS(2) & RETURN
3 KASNEW=2 & KSRNEW=3 & UC(2)=LBUC(2)-UCS(2) & RETURN
4 KASNEW=3 & KSRNEW=4 & UC(1)=UBUC(1)-UCS(1) & RETURN
5 KASNEW=3 & KSRNEW=5 & UC(1)=LBUC(1)-UCS(1) & RETURN
6 KASNEW=4 & KSRNEW=6 & UC(1)=UBUC(1)-UCS(1)
UC(2)=UBUC(2)-UCS(2) & RETURN
7 KASNEW=4 & KSRNEW=7 & UC(1)=UBUC(1)-UCS(1)
UC(2)=LBUC(2)-UCS(2) & RETURN
8 KASNEW=4 & KSRNEW=8 & UC(1)=LBUC(1)-UCS(1)
UC(2)=UBUC(2)-UCS(2) & RETURN
9 KASNEW=4 & KSRNEW=9 & UC(1)=LBUC(1)-UCS(1)
UC(2)=LBUC(2)-UCS(2) & RETURN

END

SUBROUTINE UPDEIG (N,V,RTGIN,W1,W2,W3,W4,RTC1,RTC2,RTC3,RTC4,
1 W,RTC,KASAT)

21.1.71,

UPDATE EIGENVALUES/VECTORS AND INVERSES, I.E.

REPLACE W,RTC FROM W1,...,W4,RTC1,...,RTC4, ACCORDING TO KASAT

COMPUTE V,RTGIN FROM W,RTC

```

      COMPLEX W(N,N),RTC(N),V(N,N),PTCIN(N),W1(N,N),W2(N,N),W3(N,N),
1      W4(N,N),RTC1(N),RTC2(N),RTC3(N),RTC4(N)
      GO TO (1,2,3,4), KASAT
1     CALL EQUVC (RTC1,RTC,N)
      CALL EQUVC (W1,W,N,N)
      GO TO 5
2     CALL EQUVC (RTC2,RTC,N)
      CALL EQUVC (W2,W,N,N)
      GO TO 5
3     CALL EQUVC (RTC3,RTC,N)
      CALL EQUVC (W3,W,N,N)
      GO TO 5
4     CALL EQUVC (RTC4,RTC,N)
      CALL EQUVC (W4,W,N,N)
5     CALL INVCOM (W,V,N,IPANK)
      DO 6 I=1,N
6     RTCIN(I) = 1./RTC(I)
      RETURN
      END

```

```

      SUBROUTINE UPDVRU1 (N,NC,V,PU,VPU,KASAT,UC,RC,SQ,A1,A2,ND,BD,UD,
1      KDU,SL)

```

```

      1.5.72

```

```

      UPDATE VRU, I.E. COMPUTE

```

```

      VRU(I) = V(L)*[SUM(I)[RC(J)*USAT(J)] + SUM(L)[RC(I)*KDU(I)] + PD*UD]
      J.NE.I, SEE EQ.(21) IN MAIN PROGRAM

```

```

      FOR ONE OF THE 9 POSSIBLE VALUES OF SUM(I), ACCORDING TO KSUMB

```

```

      FOR ONE OF THE 4 POSSIBLE VALUES OF V(L), ACCORDING TO KASAT

```

```

      FOR ONE OF THE 4 POSSIBLE VALUES OF SUM(L), ACCORDING TO KASAT

```

```

      RC(I) = COL. VECTOR OF RC CORRESP. TO NON-SAT. CONTROL INPUTS

```

```

      RC(J) = COL. VECTOR OF RC CORRESP. TO SATURAT. CONTROL INPUTS

```

```

      KDU(I) = VALUE OF FIXED ADDED CONTROL TO EACH ACTIVE INPUT

```

```

      USAT(J) = VALUE OF UC(J) CORRESP. TO UPPER OR LOWER CONSTRAINT.

```

```

      THIS ROUTINE IS CALLED AFTER SATCON, WHERE UC(J) TAKES

```

```

      THE SATURATED VALUE WHEN A CONSTRAINT VIOLATION OCCURS

```

```

      REAL RU(N),RC(N,NC),UC(NC),SQ(N),A1(N,NC),A2(N,NC),BD(N,N),

```

```

1      UD(ND),KDU(NC),SL(N)

```

```

      COMPLEX V(N,N),VRU(N)

```

```

      CALL MULMXV (RU,UD,BU,N,ND)

```

```

      CALL CAPVUP (RC,A1,N,NC,N,NC,1,1)

```

```

      CALL CAPVUP (RC,A2,N,NC,N,NC,1,1)

```

```

      A1 AND A2 ARE NOW BOTH EQUAL TO MATRIX RC

```

```

      GO TO (1,2,3,4), KASAT

```

```

1     CALL NULLRV (SQ,N)

```

```

      GO TO 7

```

```

2     DO 12 I=1,N

```

```

      A1(I,1) = 0

```

```

12    A2(I,2) = 0.

```

```

      GO TO 4

```

```

3     DO 13 I=1,N

```

```

      A1(I,2) = 0.

```

```

13    A2(I,1) = 0.

```

```

4     CALL MULMXV (A1,UD,SQ,N,NC)

```

```

      IF (KASAT - 4) 7,6,7

```

```

6     CALL NULLRV (SL,N)

```

```

      GO TO 5

```

```

7     CALL MULMXV (A2,KDU,SL,N,NC)

```


5 CALL ADDV (SQ,BU,SQ,N)

CALL ADDV (SQ,SL,SQ,N)
CALL MULCMRV(V,SQ,VBU,H,N)
RETURN
END

SUBROUTINE EXCITSQ(X0,UC,U0,N,NC,N0,IX0,IUC,IUD,IS,ICASE,NW,RC,R0,
1 RU,R2,V,VBU, IEXC,XS,US,NU,ITYPE,KR2,UCBAR,
2 KDG,KGD,KGG,ACD,RC,KOUOR,IKOU,ACC,A1,A2,RC1,
3 NCI,AKOU,IDG)

12.6.72

SET INITIAL EXCITATION X0 FOR STATE VARIABLES
SET STEP INPUTS UC (CONTROL) AND U0 (DISTURBANCE)

COMP. UCBAR = SETTLING-DOWN VALUE OF UC

= KGD*U0

KDG = DISTURBANCE GAIN MATRIX (OVERALL), EQ. (8) MAIN PROG.

= KR*P0 + [KR*P0 - I]*KGD

VBU(1) = V(1)*KDG*U0

EQ. (21) MAIN PROG.

KOU = - KGG*U0

EQ. (8) MAIN PROG.

FOR IKOU = 1, READ IN FORCED KOU TO TEST ITS INFLUENCE

FOR IDG = 1, COMPUTE AGAIN KOU ACCORDING TO EQUATIONS IN ROUTINE
DCCOMP, FOR DC COMPENSATION IN THE CASE OF FILTER USED

INTEGER ICASE(NW),NCI(NC)

COMPLEX V(1,N),VBU(N)

REAL X0(N),UC(NC),U0(N0),RC(N,NC),BD(N,N0),RU(N),R2(N),

1 XS(N),US(NU),KR(NC,N),UCBAR(NC),KOU(NC),KGD(NC,N0),

2 KGG(NC,N0),ACD(NC,N0),RC(NC,NC),KOUOR(NC)

REAL ACC(N,N),A1(N,N),A2(NC,N),RC1(NC,NC),AKOU(NC)

IX0 = 0 & IUC = 0 & IUD = 0

=0

CALL NULLPV (X0,N)

CALL NULLRV (UC,NC)

CALL NULLPV (U0,N0)

CALL NULLPV (RU,N)

CALL NULLRV (UCBAR,NC)

CALL NULLRV (KOU, NC)

CALL NULL (KGD,NC,N0)

(RECALL TO RESET IX0,ETC TO 1 WHERE NECESSARY)

EXCITATION TYPE (1) X(9) = UC = U0 = 0.

EXCITATION TYPE (2) X0(10) = 20. PSTA, U0(8) = .1*US(8)

EXCITATION TYPE (3) U0(1) = 10 MILINILES = 10E-5

EXCITATION TYPE (4) U0(8) = .1*US(8)

EXCITATION TYPE (5) X0(10) = 20. PSTA

EXCITATION TYPE (6) U0(8) = -.1*US(8)

FROM THE STEADY STATE SET OF EQUATIONS (83,175,177 OF WINFRITH
SIMULATOR) AND DUE TO THE FACT THAT STEP INPUTS ARE USED, THE
FOLLOWING STATEMENT IS INTRODUCED FOR EXC. TYPES (1) TO (6). (CHCK 3)

KGD(2,8) = 8.1195/XS(10)

GO TO (9,2,3,4,5,6), ITYPE

2 IX0 = IUD = 1

X0(10) = 20. & U0(8) = .1*US(8)

GO TO 9

3 IUD = 1

U0(1) = 10.E-5

GO TO 9

OK9AL

4 IUD = 1

UD(8) = .1*US(8)

PE(MW)

GO TO 9

5 IXC = 1

XO(10) = 20.

PD(PSIA)

GO TO 9

6 IUD = 1

UD(8) = - .1*US(8)

PE(MW)

9 CALL MULMXV (KOD,UD,UCRAB,UC,ND)

IF (IUC.EQ.0.AND.IUD.EQ.0) GO TO 10

KDU = - KDG*UD

KDG = KOR*BO + (KOR*BC - I1*KCO

CALL MULTI (KOR,BO,BC,NC,N,ND)

DO 15 I=1,NC

15 RC(I,I) = RC(I,I) - 1.

CALL MULTI (RC,KCO,ACD,NC,NC,ND)

CALL MULTI (KOR,BO,KDG,NC,N,ND)

CALL ADD (KDG,ACD,KDG,NC,ND)

CALL WPMX (KDG,NC,ND,30HKDG (UC(I) = -KR*X(I) -KDG*UD))

CALL MULMXV(KDG,UD,KDU,NC,ND)

CALL NEGVR (KDU,NC)

TEST IF KDU IS REQUIRED TO BE CHANGED TO EVALUATE ITS INFLUENCE

IF (IKDU.EQ.1) CALL KDUCHG (NC,KDU,KDUOR)

TEST IF KDU HAS TO BE RECOMPUTED FOR DC COMPENSATION

IF (IDC.EQ.1) CALL DCCOMP (N,NC,ND,ACD,A1,A2,RC,RC1,NCI,KDUOR,KDU,
1 BC,BO,UD,ACD,AKDU)

VBU = V*BOG*UD

FOR NON-SATURATING CASE

BOG = [I - BO*KDR1*(BO + BO*KCO) = BO - BO*KDG

HENCE VBU = V*(BO*UD + BO*KDU)

CALL MULMXV (BO,UD,BU,N,ND)

CALL MULMXV (BO,KDU,R2,N,NC)

CALL ADDV (BU,R2,BU,N)

10 CALL MULCHRV (V,BU,VBU,N,N)

VBU IS NOW INV(W1)*BOG*UD,

EQ.(21) MAIN PROG.

BU = BOG*UD = (BO*UD + BO*KDU) / VALID ONLY FOR L = KASAT = 1
VBU = V*BU / Q = KSUM = 1

I.E. WHEN THE CONTROL VECTOR IS WITHIN THE CONSTRAINTS

IFXC = 10HSTEP

RETURN

END

Appendix FList of results of SIMULA for cases of REGULA

Below tables F.1 (a) to F.1 (j) are listed with the results of all the cases run. Only the most important peak values are listed, and underlined when a constraint is violated. The constraint values are written on table f.1 (a).

All results are for excitation type (6), i.e. a change of -10% in demand of electric power.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c1}	u_{c2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	y'_M	y'_{TCV}	x_{18} P_{MEC}	u_{c1}	u_{c2}	x_{30}	x_{40}
316	.1	1	1	1	6.	+0.73 <u>-7.5</u>	+1.1 -.15	<u>2.4</u>	3.9
317	.3	1	1	1	6.5	+0.79 <u>-7.8</u>	+1.2 -.16	<u>2.6</u>	1.5
300	1	1	1	1	6.5	+0.81 <u>-7.9</u>	+1.3 -.16	<u>2.8</u>	.46
306	1	1	1	10	5.5	+1.1 <u>-11.</u>	+1. -.14	<u>2.2</u>	1.1
308	1	1	1	E2	3.5	+2. <u>-18.</u>	+0.51 -.13	<u>1.2</u>	4.8
309	1	1	1	E3	.5	+3. <u>-23.</u>	+0.04 -.15	.29	9.4
310	1	1	1	E4	.5	+3.1 <u>-25.</u>	0. -.11	.22	11.
302	1	1	10	1	10.	+0.41 <u>-4.1</u>	+1.9 -.2	<u>4.</u>	.64
327	1	1	10	E3	.6	+1.3 <u>-12.</u>	+0.1 -.13	.4	13.
325	1	1	10	E4	.5	+1.3 <u>-12.</u>	0. -.1	.22	15.
303	1	1	E2	10	13.	+0.27 <u>-2.8</u>	+2.6 -.23	<u>5.3</u>	2.3
328	1	1	E2	E3	.6	+0.68 <u>-6.</u>	+0.2 -.11	.62	19.
326	1	1	E2	E4	.5	+0.72 <u>-6.3</u>	0. -.11	.23	22.
305	1	1	E3	1	25.	+0.09 <u>-1.2</u>	+5.5 -.3	<u>11.</u>	1.6

Table F.1 (a). Results of SIMULA (free simulation) for cases of REGULA

Case (3) in FDBCK1 at 100% of full power

Excitation (6): $u_{d8} = -.1u_{s8}$ ($u_{d8} = P_E$, step disturbance,
 $u_{s8} = P_E$, steady state).

Constraints on deviations:

$$\begin{aligned}
 -40. &\leq x_{10} \leq +40. \quad \text{psia} \\
 -1. &\leq x_{30} \leq +1. \quad \text{c/s} \\
 -1. &\leq u_{c1} \leq +1. \\
 -1.45 &\leq u_{c2} \leq +1.8
 \end{aligned}$$

Notes: (i) Values underlines violate constraints.

(ii) E3 = 10^3

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	y'_M	y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
329	1	1	E3	E3	.8	+ .34 <u>-3.2</u>	+ .41 - .12	<u>1.1</u>	31.
330	1	1	E3	E4	.5	+ .36 <u>-3.4</u>	0. - .09	.23	36.
304	1	1	E3	E5	.5	+ .36 <u>-3.4</u>	0. - .09	- .79	37.
307	1	1	E4	1	45.	+ .06 - .63	+ <u>9.3</u> - .55	<u>19.</u>	2.8
318	1	1	E4	10	35.	+ .08 - .8	+ <u>7.2</u> - .4	<u>15.</u>	6.1
319	1	1	E4	E2	20.	+ .12 <u>-1.3</u>	+ <u>3.7</u> - .3	<u>7.6</u>	23.
324	1	1	E4	E3	.9	+ .18 <u>-1.7</u>	+ .8 - .1	<u>1.8</u>	<u>52.</u>
321	1	1	E4	E4	.5	+ .19 <u>-1.8</u>	+ .01 - .1	.33	<u>61.</u>
322	1	1	E4	E4	.5	+ .06 - 1.	0. - .1	.25	<u>110.</u>
344	1	1	E4	E5	.5	+ .19 - 1.8	0 - .1	.23	<u>62.</u>
0080	1	1	E5	E3	.6	+ .07 - .93	+ 1.4 <u>-1.6</u>	<u>3.2</u>	<u>90.</u>
331	1	1	E5	E4	.5	+ .06 - .99	+ .09 - .1	.57	<u>110.</u>
336	1	1	E5	E5	.5	+ .06 <u>-1.</u>	0. - .1	.25	<u>110.</u>
0084	1	1	E5	E6	.5	+ .06 <u>-1.</u>	0. - .1	.23	<u>110.</u>
0081	1	1	E6	E3	.4	+ .05 - .5	+ <u>2.5</u> - .56	<u>5.4</u>	<u>160.</u>
334	1	1	E6	E4	.5	+ .05 - .53	+ .21 - .15	<u>1.</u>	<u>190.</u>
335	1	1	E6	E5	.6	+ .05 - .53	+ .21 - .15	.45	<u>190.</u>
333	1	1	E6	E6	.6	+ .05 - .53	0. - .1	.39	<u>190.</u>
332	1	1	E7	E5	.5	+ .03 - .27	0. - .12	.77	<u>330.</u>
340	1	1	E7	E6	.5	+ .03 - .27	0. - .1	.68	<u>330.</u>
341	1	1	E7	E7	.5	+ .03 - .27	0. - .1	.67	<u>330.</u>

Table F.1 (b). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec) x_{18} P_{MEC}	Peak deviations from s.s.			
	p_d	δf	Y_M	Y_{TCV}		u_{c_1}	u_{c_2}	x_{30}	x_{10}
311	1	10	1	1	4.5	+0.99 <u>-14.</u>	+0.76 <u>-.5</u>	<u>1.7</u>	5.7
0001	1	10	E4	1	25.	+0.62 <u>-1.1</u>	+5.3 <u>-.22</u>	<u>11.</u>	13.
352	1	10	E4	E3	.9	+0.18 <u>-1.7</u>	+0.75 <u>-.11</u>	<u>1.8</u>	<u>52.</u>
312	1	E2	1	1	.6	+2.8 <u>-22.</u>	+0.31 <u>-2.9</u>	.55	7.9
0002	1	E2	E4	1	2.	+0.16 <u>-1.6</u>	+1.6 <u>-.17</u>	<u>3.6</u>	<u>42.</u>
351	1	5E2	E4	1	.4	+0.18 <u>-1.8</u>	+0.34 <u>-2.</u>	.97	<u>57.</u>
314	1	E3	1	1	.09	+3.1 <u>-25.</u>	+0.65 <u>-5.7</u>	.07	11.
313	1	E3	1	E3	.3	+3.1 <u>-25.</u>	0. <u>-.32</u>	.14	11.
0003	1	E3	E4	1	1.7	+0.31 <u>-3.</u>	+0.93 <u>-.22</u>	.51	<u>60.</u>
315	1	E4	1	1	.06	+3.7 <u>-25.</u>	+1.3 <u>-8.9</u>	.03	11.
323	1	E5	E4	1	.05	+0.19 <u>-1.9</u>	+3. <u>-15.</u>	.02	<u>63.</u>
0090	2.1	1	E5	E5	.5	+0.11 <u>-1.2</u>	0. <u>-.09</u>	.25	<u>90.</u>
0091	2.1	1	E5	E6	.5	+0.11 <u>-1.2</u>	0. <u>-.09</u>	.23	<u>91.</u>
0092	2.1	1	E6	E5	.5	+0.07 <u>-.65</u>	0. <u>-.1</u>	.43	<u>160.</u>
0093	2.1	1	E6	E6	.5	+0.07 <u>-.65</u>	0. <u>-.09</u>	.33	<u>160.</u>
0086	3.1	1	E5	E5	.5	+0.14 <u>-1.3</u>	0. <u>-.09</u>	.25	<u>82.</u>
0088	3.1	1	E5	E6	.5	+0.14 <u>-1.3</u>	0. <u>-.09</u>	.23	<u>82.</u>
0087	3.1	1	E6	E5	.5	+0.08 <u>-.7</u>	0. <u>-.1</u>	.44	<u>140.</u>
0089	3.1	1	E6	E6	.5	+0.08 <u>-.7</u>	0. <u>-.09</u>	.3	<u>140.</u>
350	5	1	E4	E4	.6	+0.28 <u>-2.7</u>	+0.23 <u>-.11</u>	.72	39.
353	5	10	E4	E3	.8	+0.23 <u>-2.3</u>	+1.7 <u>-.12</u>	<u>3.7</u>	23.

Table F.1 (c). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
301	10	1	1	1	6.5	+0.82 <u>-8.</u>	+1.3 -0.16	-0.11	0.05
343	10	1	E4	E4	.8	+0.34 <u>-3.2</u>	+0.41 -0.12	1.1	31.
345	10	1	E4	E5	.5	+0.36 <u>-3.4</u>	0. -0.95	.23	36.
346	10	1	E5	E4	.9	+0.18 <u>-1.7</u>	+0.78 -0.11	1.8	52.
348	10	1	E5	E5	.5	+0.19 <u>-1.8</u>	+0.01 -0.1	.33	61.
0085	10	1	E5	E6	.5	+0.19 <u>-1.8</u>	0. -0.1	.23	62.
339	10	1	E6	E5	.5	+0.06 -0.99	+0.1 -0.1	.57	110.
338	10	1	E6	E6	.5	+0.06 <u>-1.</u>	0. -0.1	.25	110.
337	10	1	E7	E5	.5	+0.05 -0.53	+0.21 -0.15	.99	190.
342	10	1	E7	E7	.5	+0.05 -0.53	0. -0.1	.39	190.
0006	10	E2	E4	1	16.	+0.19 <u>-2.</u>	+3.1 -0.3	6.5	7.7
0007	10	E2	E4	10	15.	+0.19 <u>-2.</u>	+3.1 -0.11	6.4	7.8
0008	10	E2	E4	E2	25.	+0.2 <u>-2.1</u>	+2.9 -0.1	6.	9.
0118	10	E2	E4	E3	8.	+0.26 <u>-2.6</u>	+1.8 -0.19	3.9	16.
0119	10	E2	E4	E4	.7	+0.34 <u>-3.2</u>	+0.4 -0.12	1.	31.
0120	10	E2	E4	E5	.5	+0.36 <u>-3.4</u>	0. -0.1	.23	36.
0134	10	E2	E5	E5	.5	+0.19 <u>-1.8</u>	+0.01 -0.09	.33	110.
0135	10	E2	E6	E5	.5	+0.06 -0.99	+0.09 -0.1	.57	110.
0136	10	E2	E7	E5	.4	+0.05 -0.53	+0.21 -0.15	.98	190.
0137	10	E2	E8	E5	Imprecise in REGULA				
0004	10	E3	E4	1	1.7	+0.31 <u>-3.</u>	+0.9 -0.2	2.1	25.

Table F.1 (d). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
0005	10	E3	E4	10	1.8	+. <u>31</u> <u>-3.</u>	+. <u>9</u> <u>-.15</u>	<u>2.1</u>	25.
320	10	E4	1	1	.05	+ <u>7.8</u> <u>-53.</u>	+ <u>2.6</u> <u>-17.</u>	.06	8.1
0009	10	E4	E4	10	.11	+. <u>36</u> <u>-3.4</u>	+. <u>4</u> <u>-4.1</u>	.29	36.
0012	10	E4	E4	E2	.16	+. <u>36</u> <u>-3.4</u>	+. <u>06</u> <u>-1.5</u>	.29	36.
0015	10	E4	E4	E3	.2	+. <u>36</u> <u>-3.4</u>	+. <u>02</u> <u>-.6</u>	.29	36.
0010	10	E4	E5	10	.13	+. <u>19</u> <u>-1.8</u>	+. <u>3</u> <u>-3.5</u>	.51	<u>60.</u>
0013	10	E4	E5	E2	.2	+. <u>19</u> <u>-1.8</u>	+. <u>1</u> <u>-1.3</u>	.51	<u>60.</u>
0016	10	E4	E5	E3	.2	+. <u>19</u> <u>-1.8</u>	+. <u>1</u> <u>-.6</u>	.5	<u>60.</u>
0011	10	E4	E6	10	.1	+. <u>06</u> <u>-.99</u>	+. <u>45</u> <u>-4.4</u>	.89	<u>100.</u>
0014	10	E4	E6	E2	.15	+. <u>06</u> <u>-.99</u>	+. <u>25</u> <u>-1.6</u>	.89	<u>100.</u>
0017	10	E4	E6	E3	.2	+. <u>06</u> <u>-.99</u>	+. <u>24</u> <u>-.65</u>	.87	<u>100.</u>
0029	10	E5	E4	E2	.1	+. <u>36</u> <u>-3.4</u>	+. <u>39</u> <u>-3.4</u>	.04	37.
0030	10	E5	E5	E2	.1	+. <u>19</u> <u>-1.9</u>	+. <u>38</u> <u>-3.3</u>	.05	<u>63.</u>
0098	20	1	E5	E5	.5	+. <u>23</u> <u>-2.2</u>	+. <u>08</u> <u>-.09</u>	.44	<u>52.</u>
0100	20	1	E5	E6	.5	+. <u>23</u> <u>-2.2</u>	0. <u>-.09</u>	.23	<u>53.</u>
0099	20	1	E6	E5	.5	+. <u>1</u> <u>-1.2</u>	+. <u>21</u> <u>-.11</u>	.77	<u>88.</u>
0101	20	1	E6	E6	.5	+. <u>11</u> <u>-1.2</u>	0. <u>-.09</u>	.25	<u>91.</u>
0094	30	1	E5	E5	.6	+. <u>25</u> <u>-2.4</u>	+. <u>1</u> <u>-.1</u>	.54	<u>46.</u>
0096	30	1	E5	E6	.5	+. <u>26</u> <u>-2.5</u>	0. <u>-.09</u>	.23	<u>48.</u>
0095	30	1	E6	E5	.6	+. <u>13</u> <u>-1.3</u>	+. <u>3</u> <u>-.11</u>	.95	<u>78.</u>
0097	30	1	E6	E6	.5	+. <u>14</u> <u>-1.3</u>	0. <u>-.09</u>	.25	<u>82.</u>

Table F.1 (e). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
0103	30	1	5E6	E6	.5	+0.07 -0.87	0. -0.09	.37	<u>120.</u>
347	E2	1	E5	E4	13.	+0.23 <u>-2.4</u>	+2.1 -0.17	<u>4.5</u>	14.
349	E2	1	E5	E5	.8	+0.34 <u>-3.2</u>	+0.41 -0.12	<u>1.1</u>	31.
0142	E2	1	E5	E6	.5	+0.36 <u>-3.4</u>	0. -0.09	.23	36.
0104	E2	1	5E6	E6	.5	+0.11 <u>-1.2</u>	+0.06 -0.1	.49	<u>90.</u>
0105	E2	1	E7	E6	.5	+0.06 -0.99	+0.09 -0.1	.57	<u>110.</u>
0122	E2	E2	E4	E4	6.	+0.46 <u>-4.6</u>	+1.2 -0.17	<u>2.6</u>	8.5
0121	E2	E2	E4	E5	.6	+0.68 <u>-6.</u>	+0.2 -0.11	.62	19.
0123	E2	E2	E4	E6	.5	+0.72 <u>-6.3</u>	0. -0.1	.23	22.
0138	E2	E2	E5	E5	.7	+0.34 <u>-3.2</u>	+0.41 -0.12	<u>1.1</u>	31.
0139	E2	E2	E6	E5	.9	+0.18 <u>-1.7</u>	+0.78 -0.11	<u>1.8</u>	<u>52.</u>
0140	E2	E2	E7	E5	.6	+0.07 -0.93	+1.4 -0.16	<u>3.2</u>	<u>90.</u>
0141	E2	E2	E8	E5	.4	+0.05 -0.5	+2.5 -0.56	<u>5.4</u>	<u>160.</u>
0037	E2	E4	E5	E3	1.8	+0.31 <u>-3.</u>	+0.92 -0.15	<u>2.1</u>	26.
0023	E2	E4	E6	10	2.2	+0.16 <u>-1.6</u>	+3.1 -0.61	<u>3.6</u>	<u>42.</u>
0019	E2	E4	E6	E2	2.2	+0.16 <u>-1.6</u>	+1.6 -0.17	<u>3.6</u>	<u>42.</u>
0018	E2	E4	E6	E3	1.9	+0.16 <u>-1.6</u>	+1.6 -0.15	<u>3.5</u>	<u>42.</u>
0020	E2	E4	E7	E2	1.5	+0.07 -0.87	+2.9 <u>-1.8</u>	<u>6.1</u>	<u>72.</u>
0031	E2	E5	E5	E2	.11	+0.36 <u>-3.4</u>	+0.4 <u>-4.1</u>	.29	36.
0038	E2	E5	E5	E3	.15	+0.36 <u>-3.4</u>	+0.06 <u>-1.5</u>	.29	36.
0021	E2	E5	E6	E2	.13	+0.19 <u>-1.8</u>	+0.3 <u>-3.5</u>	.51	<u>60.</u>

Table F.1 (f). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
0022	E2	E5	E6	E3	.2	+0.19 <u>-1.8</u>	+0.1 -1.3	.51	<u>60.</u>
0106	3E2	1	5E6	E6	.6	+0.16 <u>-1.6</u>	+0.24 -.1	.81	<u>67.</u>
0107	3E2	1	E7	E6	.6	+0.13 <u>-1.3</u>	+0.31 -.11	.95	<u>78.</u>
0111	5E2	1	5E6	E6	.6	+0.18 <u>-1.8</u>	+0.39 -.1	<u>1.1</u>	<u>56.</u>
0113	5E2	1	E7	E6	.6	+0.15 <u>-1.5</u>	+0.47 -.12	<u>1.3</u>	<u>67.</u>
0114	5E2	1	E7	E7	.5	+0.16 <u>-1.5</u>	0. -.09	.27	<u>73.</u>
0041	5E2	E5	E6	E3	.8	+0.27 <u>-2.6</u>	+0.59 -.54	<u>1.4</u>	34.
0042	5E2	E5	E7	E3	.9	+0.15 <u>-1.4</u>	+1.1 -.18	<u>2.5</u>	<u>59.</u>
0046	5E2	E6	E6	E3	.1	+0.29 <u>-2.8</u>	+0.36 <u>-3.8</u>	.18	<u>42.</u>
0143	E3	1	E5	E6	.6	+0.68 <u>-6.</u>	+0.2 -.11	.63	19.
0112	E3	1	5E6	E6	.8	+0.21 <u>-2.1</u>	+0.65 -.1	<u>1.6</u>	<u>45.</u>
0108	E3	1	E7	E6	.9	+0.18 <u>-1.7</u>	+0.78 -.11	<u>1.8</u>	<u>52.</u>
0109	E3	1	E7	E7	.5	+0.19 <u>-1.7</u>	+0.01 -.1	.33	<u>61.</u>
0115	E3	1	E8	E7	.5	+0.6 -.99	+0.09 -.1	.57	<u>110.</u>
0124	E3	E2	E4	E5	4.	+0.99 <u>-8.8</u>	+0.8 -.14	<u>1.8</u>	6.1
0125	E3	E2	E4	E6	.5	+0.13 <u>-12.</u>	+0.1 -.13	.4	13.
0024	E3	E4	E6	E3	16.	+0.19 <u>-2.</u>	+0.1 <u>-1.15</u>	<u>6.4</u>	7.8
0025	E3	E5	E6	E3	1.8	+0.31 <u>-3.</u>	+0.93 -.15	<u>2.1</u>	25.
0026	E3	E5	E7	E3	2.2	+0.16 <u>-1.6</u>	+1.6 -.17	<u>3.6</u>	<u>42.</u>
0060	E3	E5	E7	E4	1.9	+0.16 <u>-1.6</u>	+1.6 -.15	<u>3.5</u>	<u>42.</u>
0062	E3	E5	E7	E5	1.7	+0.17 <u>-1.6</u>	+1.4 -.13	<u>3.</u>	<u>45.</u>

Table F.1 (g). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30}	x_{10}
0061	E3	E5	E8	E4	1.2	+0.07 -0.87	+2.9 -0.64	6.	72.
0063	E3	E5	E8	E5	.7	+0.07 -0.9	+2.5 -0.24	5.3	77.
0051	E3	5E5	E6	E3	.2	+0.35 -3.3	+0.23 -2.8	.56	34.
0050	E3	5E5	E7	E3	.3	+0.18 -1.8	+0.34 -2.	.97	57.
0027	E3	E6	E6	E3	.12	+0.36 -3.4	+0.4 -4.1	.29	36.
0032	E3	E6	E6	E4	.16	+0.36 -3.4	+0.06 -1.5	.29	36.
0033	E3	E6	E6	E5	.2	+0.36 -3.4	+0.02 -0.62	.29	36.
0028	E3	E6	E7	E3	.14	+0.19 -1.8	+0.3 -3.5	.51	60.
0034	E3	E6	E7	E4	.13	+0.19 -1.8	+0.1 -1.3	.51	60.
0044	E3	E6	E7	E5	.3	+0.27 -2.6	+0.58 -0.26	1.4	34.
0110	3E3	1	E7	E7	.6	+0.25 -2.4	+0.13 -0.1	.54	46.
0116	3E3	1	E8	E7	.6	+0.13 -1.3	+0.31 -0.11	.95	78.
0052	3E3	E6	5E6	E4	.4	+0.29 -2.8	+0.32 -1.1	.9	36.
0053	3E3	E6	E7	E4	.4	+0.24 -2.4	+0.39 -0.91	1.1	42.
0057	3E3	E6	E7	E5	.4	+0.24 -2.4	+0.38 -0.39	1.	43.
0065	3E3	2E6	E7	E4	Imprecise in REGULA				
0075	3E3	2E6	E8	E5	" " "				
0054	3E3	3E6	E7	E4	" " "				
0055	3E3	3E6	5E7	E4	" " "				
0056	3E3	3E6	5E7	E5	.2	+0.16 -1.6	+0.13 -0.84	.57	68.
0068	3E3	3E6	E8	E5	.16	+0.13 -1.3	+0.17 -0.85	.68	79.

Table F.1 (h). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results				
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec) x_{18} P_{MEC}	Peak deviations from s.s.			
	p_d	δf	Y'_M	Y'_{TCV}		u_{c_1}	u_{c_2}	x_{30}	x_{10}
0117	5E3	1	E8	E7	.6	+0.15 <u>-1.5</u>	+0.47 -0.12	<u>1.3</u>	<u>67.</u>
0126	5E3	1	E8	E8	.5	+0.16 <u>-1.5</u>	0. -0.09	.27	<u>73.</u>
0127	5E3	1	E9	E9	.5	+0.08 -0.82	+0.02 -0.1	.47	<u>130.</u>
0064	5E3	E5	E8	E5	Imprecise in REGULA				
0049	5E3	E6	E6	E3	.5	+0.51 <u>-4.9</u>	+0.47 <u>-3.9</u>	.81	20.
0043	5E3	E6	E7	E4	.8	+0.27 <u>-2.6</u>	+0.59 -0.51	<u>1.4</u>	34.
0045	5E3	E6	E7	E5	.8	+0.27 <u>-2.6</u>	+0.58 -0.26	<u>1.4</u>	34.
0058	5E3	E6	E8	E5	Imprecise in REGULA				
0066	5E3	2E6	E7	E4	" " "				
0067	5E3	2E6	E8	E4	" " "				
0070	5E3	3E6	5E7	E5	.3	+0.19 <u>-1.8</u>	+0.26 -0.72	.82	<u>58.</u>
0071	5E3	3E6	E8	E5	.3	+0.15 <u>-1.5</u>	+0.32 -0.72	.97	<u>69.</u>
0059	5E3	5E6	E8	E5	Imprecise in REGULA				
0048	5E3	E7	E7	E4	" " "				
0144	E4	1	E7	E8	.5	+0.36 <u>-3.4</u>	0. -0.09	.23	36.
0145	E4	1	E8	E8	.5	+0.19 <u>-1.8</u>	+0.01 -0.09	.33	<u>61.</u>
0128	E4	1	E9	E8	.5	+0.06 -0.99	+0.09 -0.1	.57	<u>110.</u>
0129	E4	1	E9	E9	.5	+0.01 <u>-1.</u>	0. -0.09	.25	<u>110.</u>
0130	E4	1	E10	E8	.4	+0.05 -0.53	+0.21 -0.15	.99	<u>190.</u>
0131	E4	1	E10	E9	.5	+0.05 -0.53	0. -0.1	.45	<u>190.</u>
0132	E4	1	E10	E10	.5	+0.05 -0.53	0. -0.1	.39	<u>190.</u>

Table F.1 (i). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

Case	Costs				Results			
	x_{10}	x_{30}	u_{c_1}	u_{c_2}	τ (sec)	Peak deviations from s.s.		
	p_d	δf	y'_M	y'_{TCV}	x_{18} P_{MEC}	u_{c_1}	u_{c_2}	x_{30} x_{10}
0146	E4	10	E6	E7	Imprecise in REGULA			
0147	E4	10	E7	E7	.7	$\begin{array}{ l} +.34 \\ -3.2 \end{array}$	$\begin{array}{ l} +.41 \\ -.12 \end{array}$	$\begin{array}{ l} 1.1 \\ 31. \end{array}$
0148	E4	E2	E5	E6	4.2	$\begin{array}{ l} +.99 \\ -8.8 \end{array}$	$\begin{array}{ l} +.8 \\ -.14 \end{array}$	$\begin{array}{ l} 1.8 \\ 6. \end{array}$
0149	E4	E2	E6	E6	Imprecise in REGULA			
0035	E4	E6	E7	E4	"	"	"	"
0072	E4	3E6	E8	E5	"	"	"	"
0073	E4	3E6	E8	E7	.5	$\begin{array}{ l} +.18 \\ -1.8 \end{array}$	$\begin{array}{ l} +.32 \\ -.14 \end{array}$	$\begin{array}{ l} .93 \\ 57. \end{array}$
0074	E4	3E6	3E8	E7	.5	$\begin{array}{ l} +.13 \\ -1.3 \end{array}$	$\begin{array}{ l} +.45 \\ -.14 \end{array}$	$\begin{array}{ l} 1.2 \\ 74. \end{array}$
0069	E4	5E6	E8	E7	.3	$\begin{array}{ l} +.19 \\ -1.8 \end{array}$	$\begin{array}{ l} +.2 \\ -.18 \end{array}$	$\begin{array}{ l} .7 \\ 59. \end{array}$
0036	E4	E7	E7	E4	Imprecise in REGULA			
0076	3E4	5E6	E8	E7	"	"	"	"
0077	5E4	5E6	E8	E7	"	"	"	"
0078	5E4	E7	E8	E7	"	"	"	"
0079	E5	E7	E9	E7	1.7	$\begin{array}{ l} +.17 \\ -1.6 \end{array}$	$\begin{array}{ l} +1.4 \\ -.13 \end{array}$	$\begin{array}{ l} 3. \\ 45. \end{array}$

Table F.1 (j). Results of SIMULA for cases of REGULA

Continuation of Table F.1 (a) where the comments are written.

APPENDIX G.

SIMULATION PLOTS AS PRODUCED BY SIMULA AND SIMSAT

Following are the simulation plots produced for case 0126 of REGULA, which is one of the optimal controllers recommended, (see section 6.3.5), as well as plots for the uncontrolled plant according to the model of case (7) in FDBCK1 of MXEV2X1 (see section 6.4).

Specifically:

Section (G.1) contains plots of case 0126 from SIMULA (free simulation).

Section (G.2) contains plots of case 0126 from SIMSAT (saturated simulation) with all state variables fed back.

Section (G.3) contains plots of case 0126 from SIMSAT with a reduced set of feedback variables and D.C. compensation.

Section (G.4) contains plots of the uncontrolled plant, corresponding to case (7) in FDBCK1 of MXEV2X1, the purpose being to compare the response of the linear model with the analog record of the non-linear model.

Several general comments should be made on these plots:

- (i) The time scale is piece-wise linear. This allows to examine the short transients as well as the long-run behaviour of the response. On the other hand the response thus acquires an unfamiliar shape, presenting several sharp turns due to scale changes. These changes take place at 0.3, 1, 10, 100 and 1000 second respectively.
- (ii) There are several parameters on each plot (up to 5) which are symbolized by the numbers (1) to (5). Although it requires some effort to follow the curves, it was decided not to join the successive points on each curve as it is felt that the picture would be more confused.

The symbol (X) denotes that two or more curves fall on the same point. The arrows are reference points for the main time axis and for the (0.1) intervals on the amplitude scale. A symbol (X) in the place of an arrow indicates that one or more curves fall on that point.

- (iii) The amplitude is normalized in all plots, from -0.2 to +1.0. At the top of each plot there is a table indicating the amplitude scale for every curve plotted.
- (iv) The steady state values are written at the top of each plot.
- (v) Not all the variables are plotted.
- (vi) There is a table of constrained parameters at the end of each section (after the plots) with a warning printed wherever a constraint is violated.

G.1. Case 0126 as plotted by SIMULA

Following are the plots of the free simulation of the optimally controlled plant for case (0126) of REGULA.

The excitation is type (6), i.e. :

$$u_{d8} = -0.1 u_{s8}$$

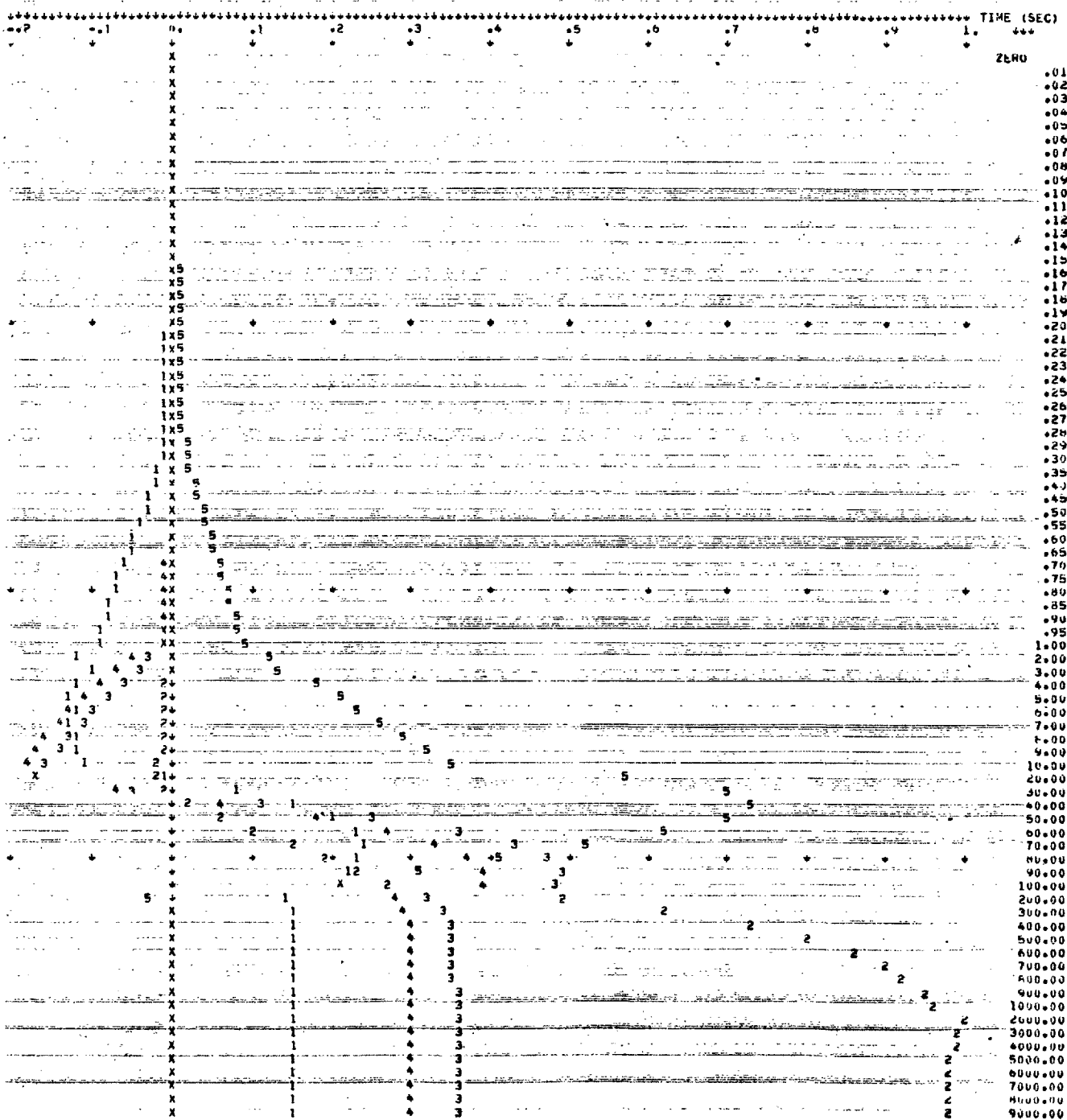
where $u_{d8} \equiv$ Electrical power disturbance, step input.

$u_{s8} \equiv$ Electrical power, steady state value.

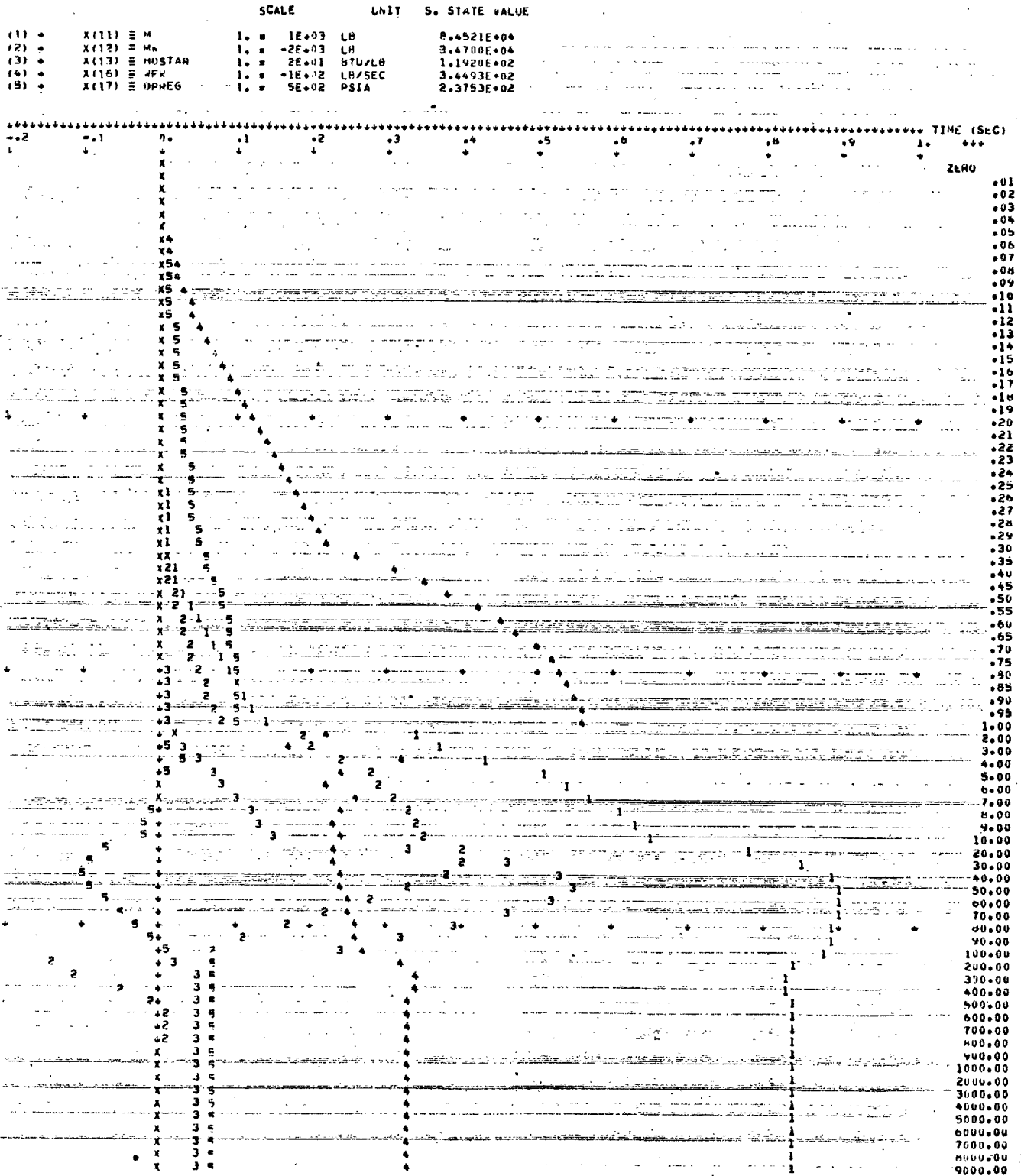
See comments in introduction to appendix (G).

STEP RESPONSE OF SYSTEM ACC(30,30) CASE 126 OF REGULA FOR CASE 3: FAC TYPE (0)

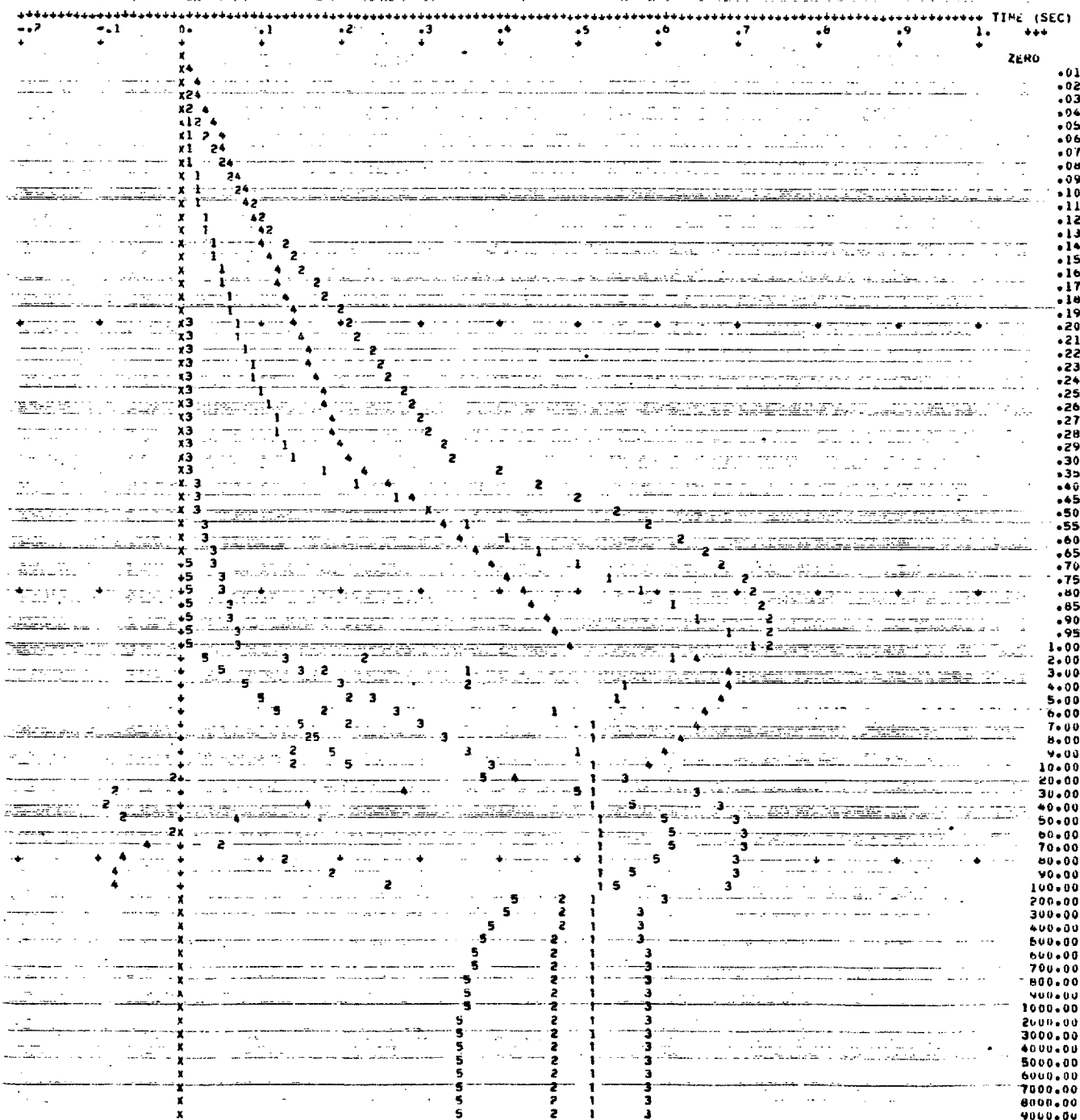
		SCALE	UNIT	S. STATE VALUE
(1) +	X(1) = PWDASH	1. = -5E+01	NORMALIZED	1.0576E+00
(2) +	X(4) = TMAY	1. = -2E+00	DEG C	4.9950E+01
(3) +	X(9) = TF1STAR	1. = -2E+02	DEG F	1.1703E+03
(4) +	X(9) = TF2STAR	1. = -1E+02	DEG F	5.7385E+02
(5) +	X(10) = PD	1. = 1E+02	PSIA	9.4000E+02



STEP RESPONSE OF SYSTEM ACC(31+30). CASE 126 OF REGULA FOR CASE 3: EXC TYPE (A)



		SCALE	UNIT	% STATE VALUE
(1)	X(19) = PNECH	1. = -2E+01	MM(G)	1.1000E+02
(2)	X(19) = YZ	1. = -1E-01	NORMALIZED	4.6410E+01
(3)	X(20) = MHFSTAR	1. = -1E+01	BTU/LH	-3.3557E+01
(4)	X(21) = YH	1. = -2E+00	NORMALIZED	0.E+00
(5)	X(22) = MDASH	1. = -5E-02	NORMALIZED	9.2000E+01

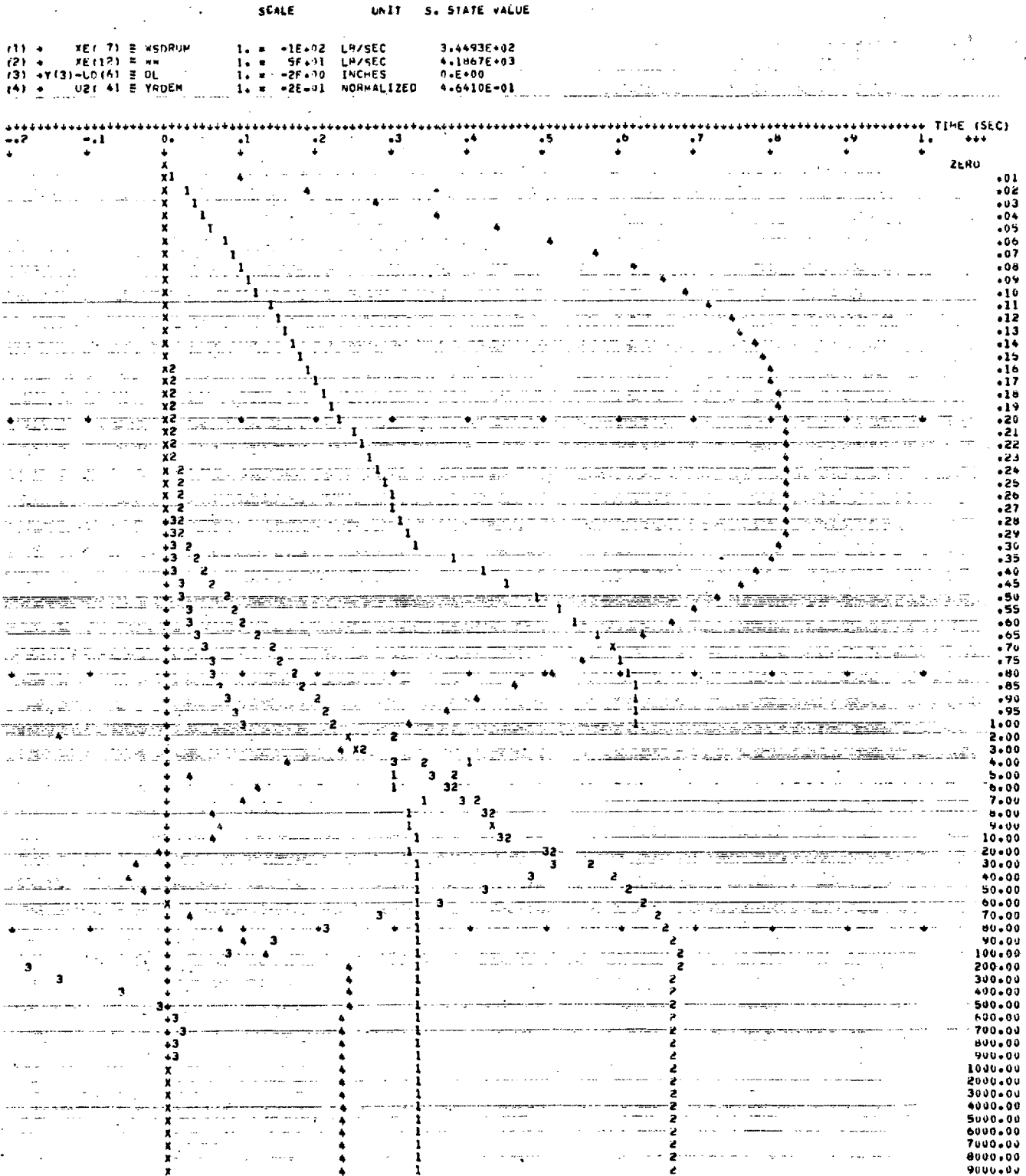


STEP RESPONSE OF SYSTEM ACC(30,30) CASE 126 OF REGULA FOR CASE 3: LAC TYPE (6)

		SCALE	UNIT	S. STATE VALUE	
(1) *	X(23) B YFCV	1. = -2E-01	NORMALIZED	9.5014E-01	
(2) *	X(37) B DF	1. = 5E-01	C/S	0.E+00	
(3) *	XA(2) FHAM	1. = -5E-02		2.6852E-01	
(4) *	XA(7) TCOOLSTR	1. = 2E-01	DEG F	1.3524E+02	
(5) *	XE(2) TFAVSTAH	1. = -2E+02	DEG F	8.1209E+02	

				TIME (SEC)	
				0.0	
				0.1	
				0.2	
				0.3	
				0.4	
				0.5	
				0.6	
				0.7	
				0.8	
				0.9	
				1.0	
				1.1	
				1.2	
				1.3	
				1.4	
				1.5	
				1.6	
				1.7	
				1.8	
				1.9	
				2.0	
				2.1	
				2.2	
				2.3	
				2.4	
				2.5	
				2.6	
				2.7	
				2.8	
				2.9	
				3.0	
				3.5	
				4.0	
				4.5	
				5.0	
				5.5	
				6.0	
				6.5	
				7.0	
				7.5	
				8.0	
				8.5	
				9.0	
				9.5	
				10.0	
				20.00	
				30.00	
				40.00	
				50.00	
				60.00	
				70.00	
				80.00	
				90.00	
				100.00	
				200.00	
				300.00	
				400.00	
				500.00	
				600.00	
				700.00	
				800.00	
				900.00	
				1000.00	
				2000.00	
				3000.00	
				4000.00	
				5000.00	
				6000.00	
				7000.00	
				8000.00	
				9000.00	

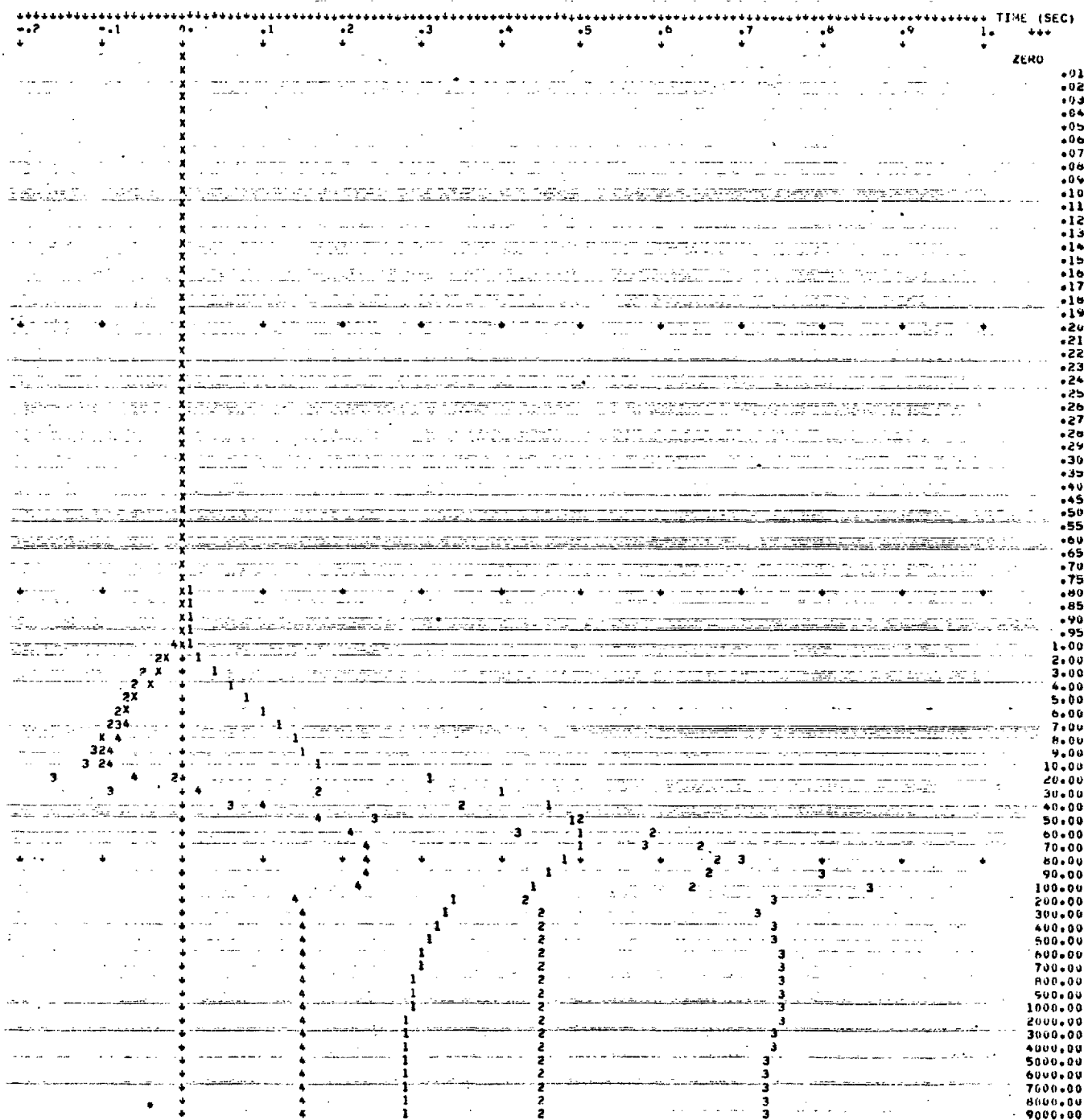
STEP RESPONSE OF SYSTEM ARC(30,31), CASE 126 OF REGULA FOR CASE 3, EIC TYPE (6)



```

(1) * XE (1) = DASH 1. = -2E+03 2.3775E+02
(2) * XA (5) = DASH 1. = -2E+01 NORMALIZED 1.0136E+00
(3) * X1 (2) = X1RASHDASH 1. = -1E+01 NORMALIZED 1.0576E+00
(4) * X4 (3) = X4RASHDASH 1. = -5E+01 NORMALIZED 1.0576E+00

```



STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CASE 3, EXC TYPE (6)

SCALE				UNIT		S. STATE VALUE												TIME (SEC)
(1)	•	UC(1) = YMDEN	1. = +2E+00	NORMALIZED	0.E+00													
(2)	•	UC(2) = YTCVDEM	1. = -1E+01	NORMALIZED	1.4501E+00													
.....																		
-0.2	-0.1	0.	.1	.2	.3	.4	.5	.6	.7	.8	.9	1.	...					
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		
.....																		

STEP RESPONSE OF SYSTEM AOC(30,30), CASE 126 OF REGULA FOR CASE 3, EAC TYPE (6)

LIST OF CONSTRAINED PARAMETERS AND/OR DERIVATIVES (A VIOLATED CONSTRAINT YIELDS A WARNING)

		NAME	UNIT	S.STATE	CONSTRAINTS		PEAK VALUES (LINEAR DEVIATIONS)		
					UPPER	LOWER	POSITIVE	NEGATIVE	
CONTROL INPUTS (UC)									
		UC(1) == YDEM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.6E-01	-1.5E+00	CONSTRAINT VIOLATED
		UC(2) == YTCVDEM	NORMALIZED	1.450E+00	3.250E+00	0.E+00	0.E+00	-9.3E-02	
DERIVATIVE OF		UC(2) == YTCVDEM	NORMALIZED/SEC		3.300E-03	-3.300E-03	2.2E-02	-2.4E-03	CONSTRAINT VIOLATED
STATE VARIABLES									
		X(1) == PHDASH	NORMALIZED	1.058E+00	2.000E-01	PEAK TO PK	6.6E-02	-1.2E-01	
		X(10) == PD	PSIA	9.400E+02	9.800E+02	9.000E+02	7.3E+01	-3.3E+00	CONSTRAINT VIOLATED
DERIVATIVE OF		X(10) == PD	PSIA /SEC			-1.000E+00	1.1E+01	-1.1E+00	CONSTRAINT VIOLATED
		X(11) == M	LB	8.452E+04		0.E+00	9.0E+02	0.E+00	
		X(12) == MW	LB	3.470E+04		0.E+00	2.8E+02	-8.1E+02	
		X(15) == OMEGA	RAD/SEC	1.516E+02		0.E+00	7.0E+02	0.E+00	
		X(16) == FV	LB/SEC	3.449E+02		0.E+00	0.E+00	-5.0E+01	
		X(19) == YR	NORMALIZED	4.641E-01	1.000E+00	0.E+00	9.3E-03	-7.4E-02	
DERIVATIVE OF		X(19) == YR	NORMALIZED/SEC		3.300E-02	-3.300E-02	5.1E-02	-1.4E-01	CONSTRAINT VIOLATED
		X(21) == YM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.6E-01	-1.4E+00	CONSTRAINT VIOLATED
DERIVATIVE OF		X(21) == YM	NORMALIZED/SEC		2.000E-01	-2.000E-01	3.7E-02	-1.0E+00	CONSTRAINT VIOLATED
		X(22) == PDASH	NORMALIZED	9.200E-01	9.700E-01		0.E+00	-3.1E-02	
		X(23) == YTCV	NORMALIZED	9.501E-01	1.000E+00	0.E+00	0.E+00	-1.4E-01	
		X(28) == IF1	INCHES	0.E+00	1.000E+00		4.5E-02	-3.5E-01	
		X(29) == IF2		0.E+00	1.000E+00		6.6E-05	-9.4E-03	
		X(30) == CF	C/S	0.E+00	1.000E+00	-1.000E+00	2.7E-01	-3.4E-02	
ADDITIONAL PARAMETERS									
		Y(3)-UP(6) == PL	INCHES	0.E+00	8.000E+00	-8.000E+00	3.5E-01	-1.0E+00	
		UP(4) == YDEM	NORMALIZED	4.641E-01	1.000E+00	0.E+00	2.8E-02	-1.0E-01	
DERIVATIVE OF		XE(1) == PKM	/SEC		3.000E-05	-3.000E-05	4.4E-06	-3.4E-05	CONSTRAINT VIOLATED
		XA(5) == CDASH	NORMALIZED	1.014E+00	2.400E-01	PEAK TO PK	2.0E-02	-1.3E-01	
		XA(6) == CHSC	RTU/LB	1.857E+01		0.E+00	4.4E+00	-2.4E+00	
		XA(8) == CHR	RTU/LB	4.945E+01		0.E+00	0.E+00	-7.1E+00	
		XA(12) == KXCOUT		1.402E+00		0.E+00	0.E+00	-3.0E-01	
		XA(14) == KXPHAR		9.535E-01		0.E+00	0.E+00	-1.3E-01	

G.2. Case 0126 as plotted by SIMSAT (full set of feedback variables).

Following are the plots of the saturated simulation (control inputs are not allowed to violate constraints) of the controlled plant for case (0126) of REGULA. All the state variables are fed back.

The excitation is type (6), i.e. :

$$u_{d8} = -0.1 u_{s8}$$

where $u_{d8} \equiv$ Electrical power disturbance, step input.

$u_{s8} \equiv$ Electrical power, steady state value.

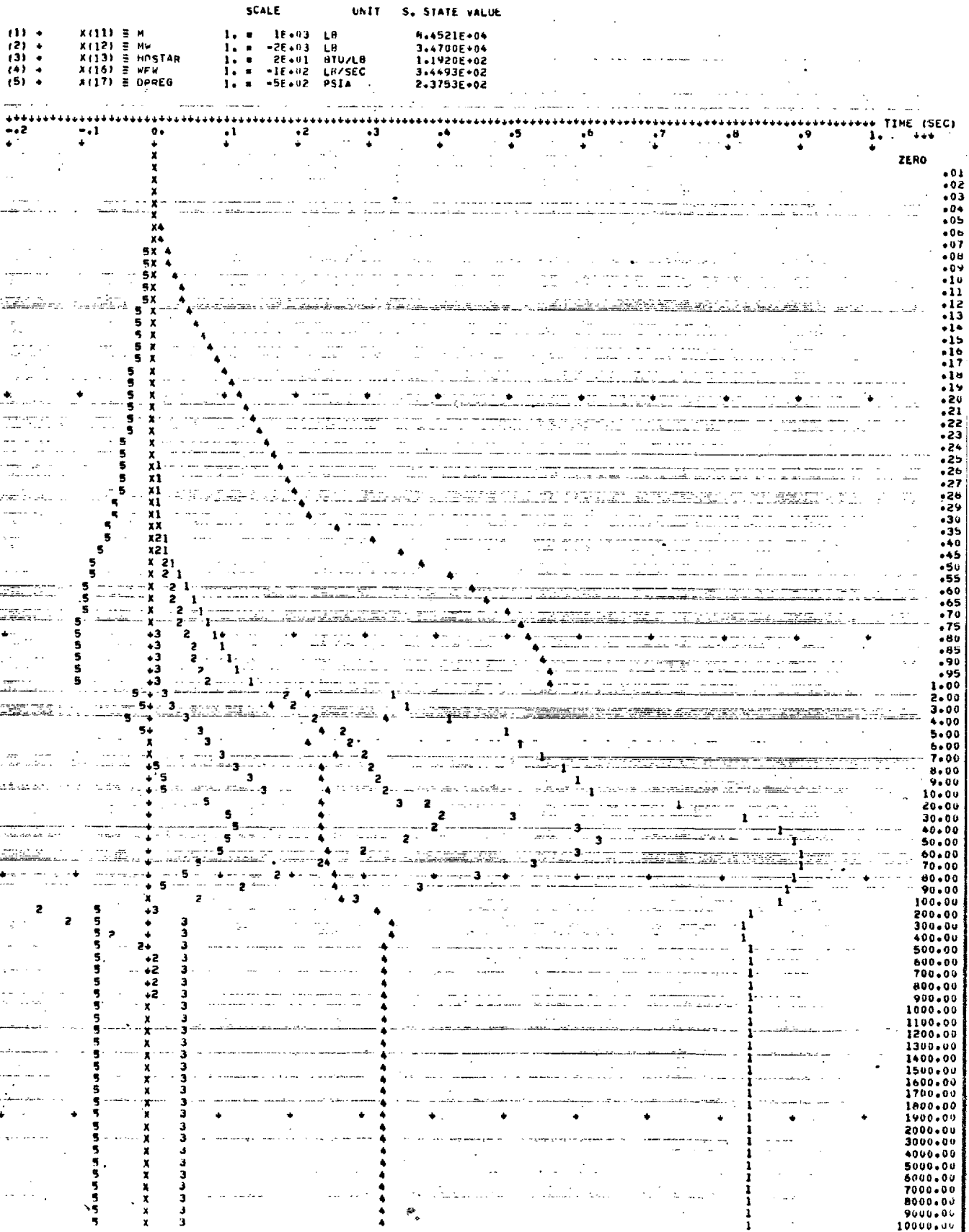
See comments in introduction to appendix (G).

STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CASE 3, EXC TYPE (6)

		SCALE	UNIT	S. STATE VALUE
(1) *	X(1) = PHIDASH	1. = -5E+01	NORMALIZED	1.0576E+00
(2) *	X(4) = TMAY	1. = -2E+00	DEG C	4.9950E+01
(3) *	X(8) = YF1STAR	1. = -5E+02	DEG F	1.1703E+03
(4) *	X(9) = YF2STAR	1. = -2E+02	DEG F	5.7385E+02
(5) *	X(10) = PD	1. = 1E+02	PSIA	9.4000E+02

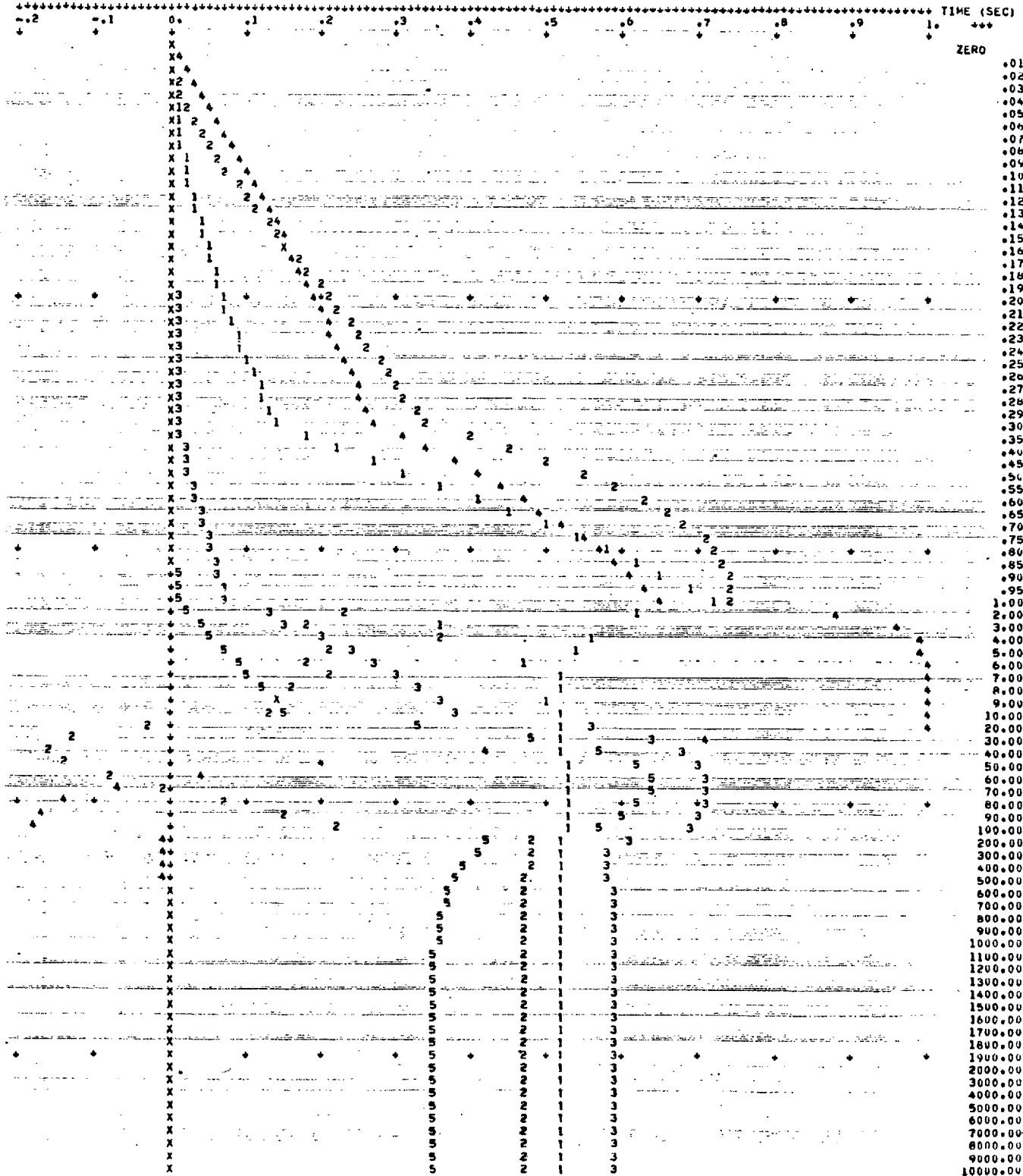
			TIME (SEC)
			0.01
			0.02
			0.03
			0.04
			0.05
			0.06
			0.07
			0.08
			0.09
			0.10
			0.11
			0.12
			0.13
			0.14
			0.15
			0.16
			0.17
			0.18
			0.19
			0.20
			0.21
			0.22
			0.23
			0.24
			0.25
			0.26
			0.27
			0.28
			0.29
			0.30
			0.35
			0.40
			0.45
			0.50
			0.55
			0.60
			0.65
			0.70
			0.75
			0.80
			0.85
			0.90
			0.95
			1.00
			2.00
			3.00
			4.00
			5.00
			6.00
			7.00
			8.00
			9.00
			10.00
			20.00
			30.00
			40.00
			50.00
			60.00
			70.00
			80.00
			90.00
			100.00
			200.00
			300.00
			400.00
			500.00
			600.00
			700.00
			800.00
			900.00
			1000.00
			1100.00
			1200.00
			1300.00
			1400.00
			1500.00
			1600.00
			1700.00
			1800.00
			1900.00
			2000.00
			3000.00
			4000.00
			5000.00
			6000.00
			7000.00
			8000.00
			9000.00
			10000.00

STEP RESPONSE OF SYSTEM AGC(30.30) CASE 126 OF REGULA FOR CASE 3: EAC TYPE (6)

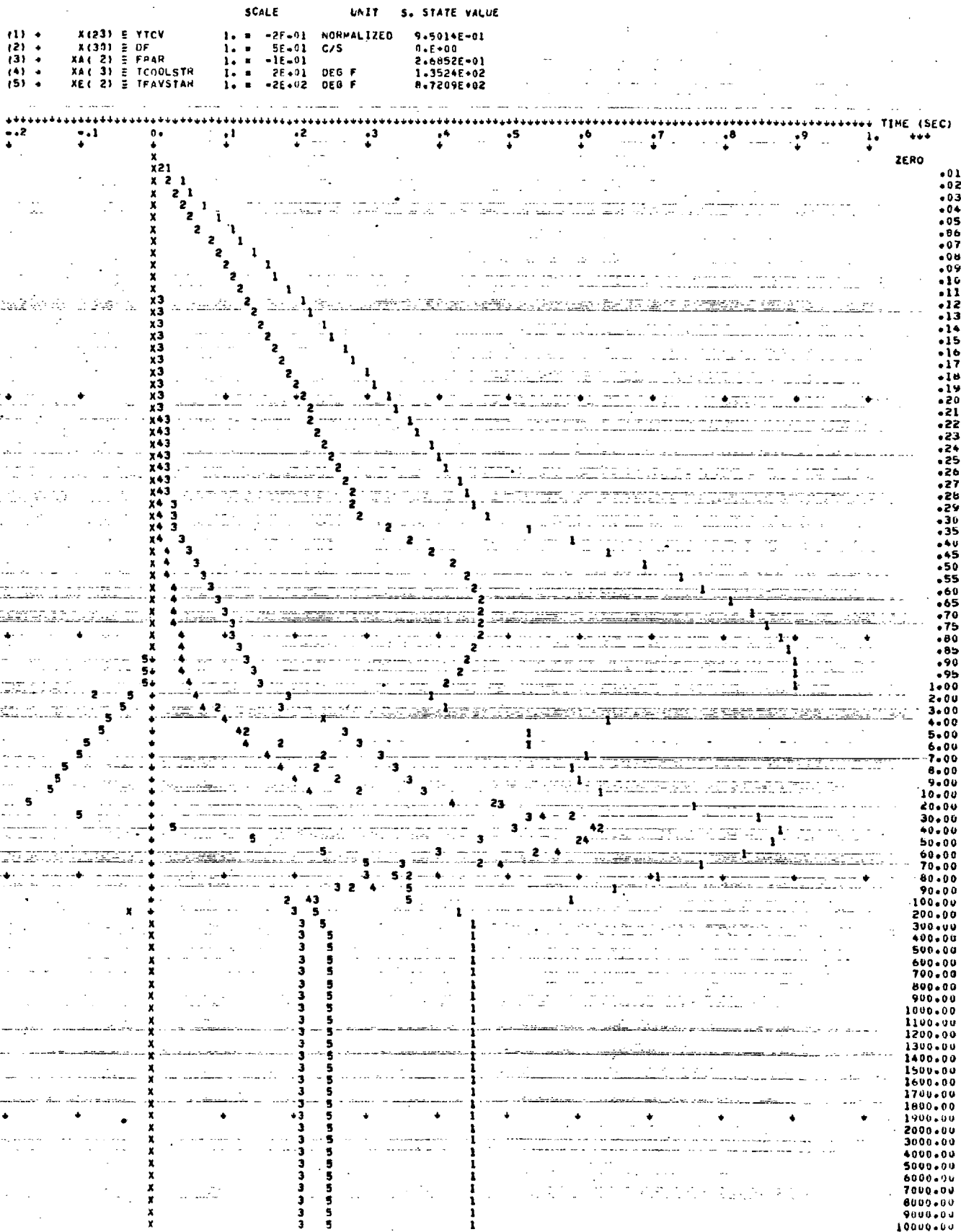


STFP RESPONSE OF SYSTEM AUC(30*30): CASE 126 OF REGULA FOR CASE 3: EAC TYPE (0)

		SCALE	UNIT	S. STATE VALUE
(1) +	X(18) = PWECH	1. = -2E+01	MW(E)	1.1000E+02
(2) +	X(19) = YR	1. = -1E+01	NORMALIZED	4.6410E-01
(3) +	X(20) = HFMSTAR	1. = -1E+01	BTU/LH	-3.3557E+01
(4) +	X(21) = YM	1. = -1E+00	NORMALIZED	0.E+00
(5) +	X(22) = HDASH	1. = -5E+02	NORMALIZED	9.2000E-01



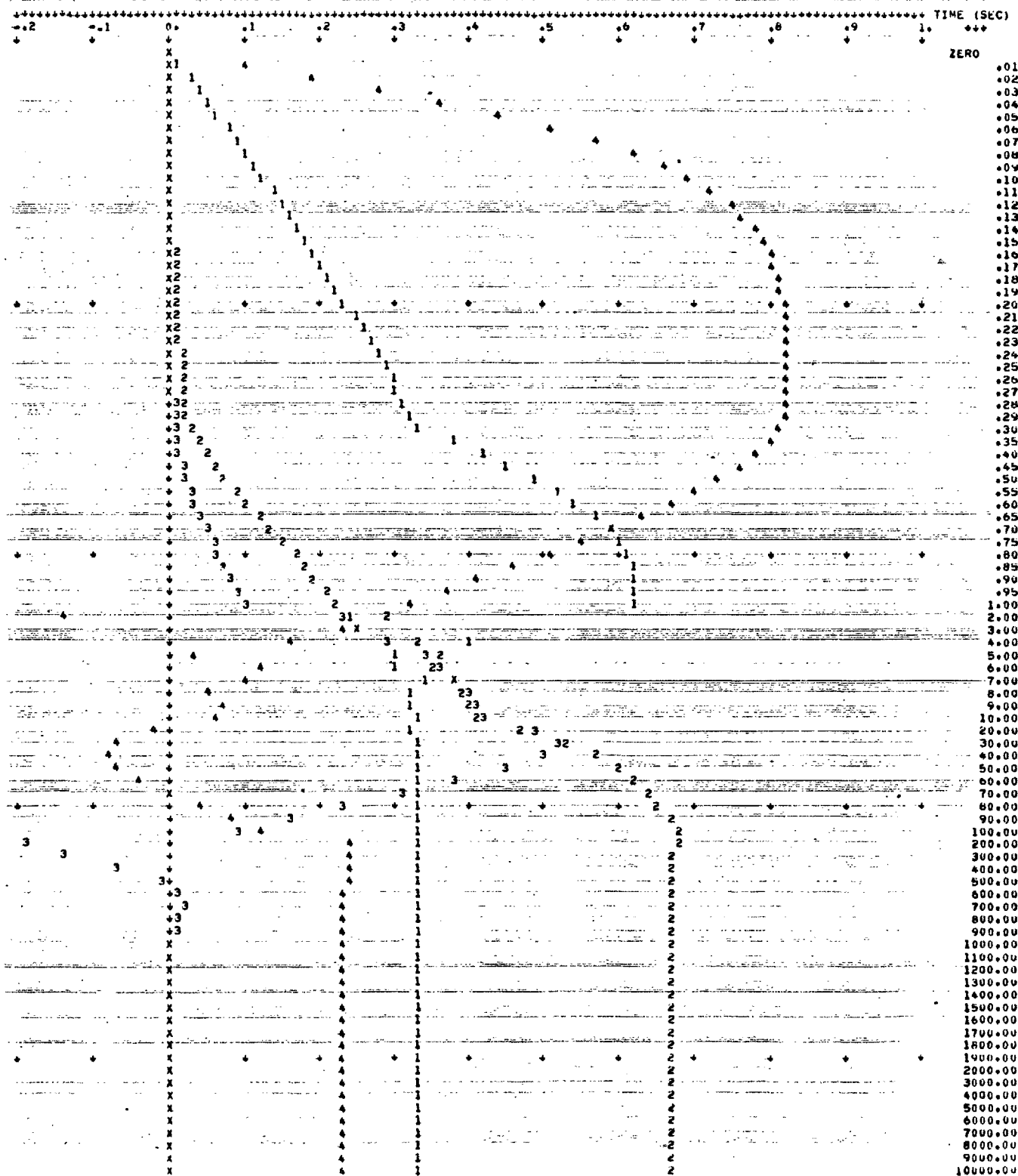
STEP RESPONSE OF SYSTEM AUG(30,30): CASE 126 OF REGULA FOR CASE 3: EXC TYPE (6)



STEP RESPONSE OF SYSTEM ACC(30*30): CASE 126 OF REGULA FOR CASE 3: EAC TYPE (6)

SCALE UNIT S. STATE VALUE

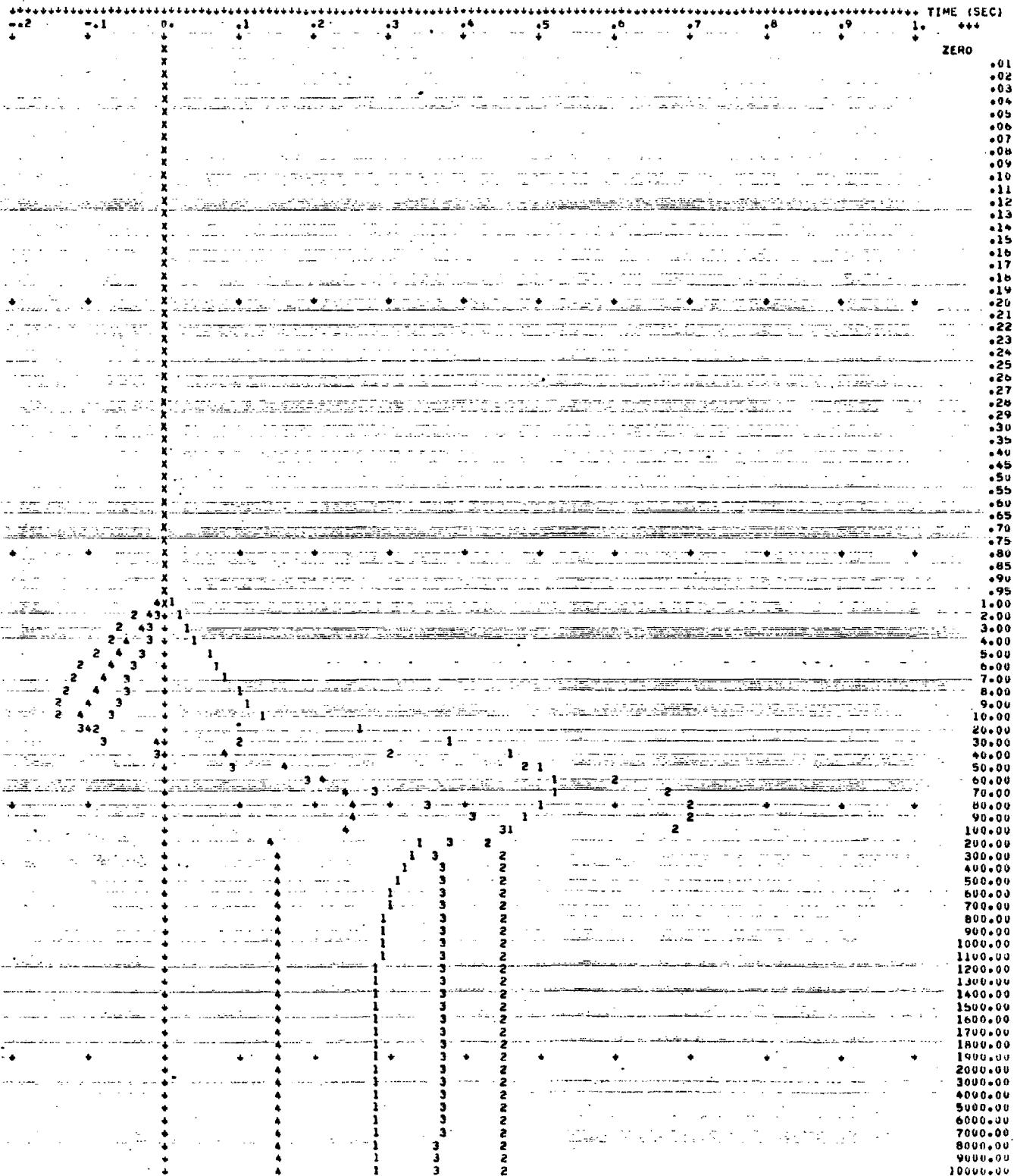
(1) * XE(7) E WSDRUM 1. * -1E+02 LB/SEC 3.4493E+02
 (2) * XE(12) E WW 1. * 5E+01 LB/SEC 4.1867E+03
 (3) * Y(3) - UN(6) E UL 1. * -2E+00 INCHES 0.E+00
 (4) * U2(4) E YNDEM 1. * -2E+01 NORMALIZED 4.6410E+01



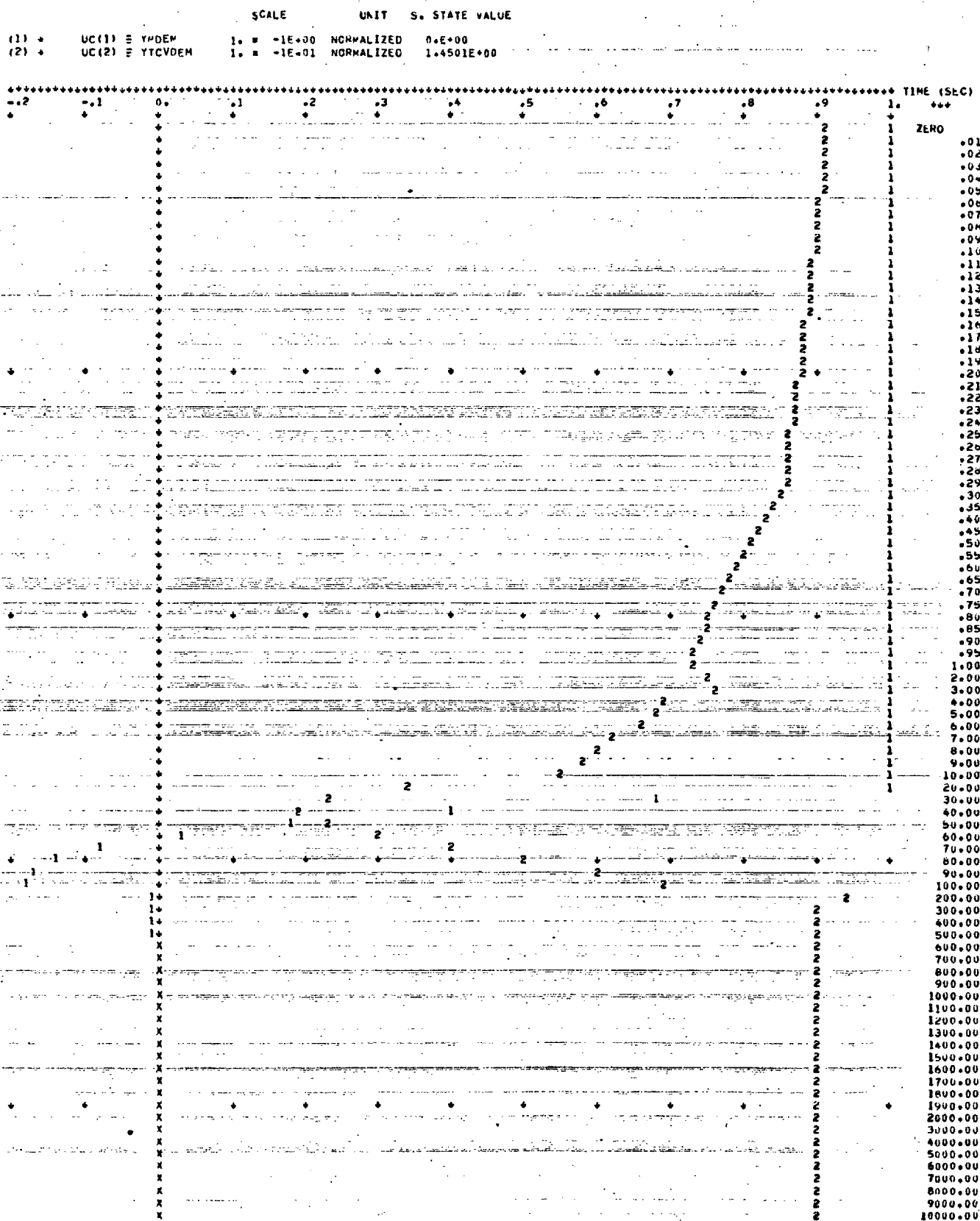
STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CNSE 3, EAC TYPE (0)

SCALE UNIT S. STATE VALUE

(1) + XE(1) = DKM 1. -2E-03 2.3775E-02
 (2) + XA(5) = QDASH 1. -2E-01 NORMALIZED 1.0136E+00
 (3) + X(2) = X1RARDASH 1. -2E-01 NORMALIZED 1.0576E+00
 (4) + X(3) = X2RARDASH 1. -5E-01 NORMALIZED 1.0576E+00



STPP RESPONSE OF SYSTEM AOC(30,30): CASE 126 OF REGULA FOR CASE 3: EAC TYPE (6)



STEP RESPONSE OF SYSTEM AOC(30,30), CASE 126 OF REGULA FOR CASE 3, EXC TYPE (6)

LIST OF CONSTRAINED PARAMETERS AND/OR DERIVATIVES (A VIOLATED CONSTRAINT YIELDS A WARNING)

	NAME	UNIT	S.STATE	CONSTRAINTS		PEAK VALUES (LINEAR DEVIATIONS)		
				UPPER	LOWER	POSITIVE	NEGATIVE	
CONTROL INPUTS (UC)								
	UC(1) = YDEM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.8E-01	-1.0E+00	
	UC(2) = YTCVDEM	NORMALIZED	1.450E+00	3.250E+00	0.E+00	0.E+00	-9.4E-02	
DERIVATIVE OF	UC(2) = YTCVDEM	NORMALIZED/SEC		3.300E-03	-3.300E-03	2.3E-02	-2.3E-03	CONSTRAINT VIOLATED
STATE VARIABLES								
	X(1) = PHDASH	NORMALIZED	1.058E+00	2.000E-11	PEAK TO PK	7.5E-02	-1.3E-01	
	X(10) = PD	PSIA	9.400E+02	9.800E+02	9.000E+02	8.5E-01	-3.9E+00	CONSTRAINT VIOLATED
DERIVATIVE OF	X(10) = PD	PSIA /SEC			-1.000E+00	1.1E-01	-1.2E+00	CONSTRAINT VIOLATED
	X(11) = V	LB	8.452E+04		0.E+00	9.1E+02	0.E+00	
	X(12) = VW	LB	3.470E+04		0.E+00	2.9E+02	-8.2E+02	
	X(15) = OMEGA	RAD/SEC	1.516E+02		0.E+00	7.5E-02	0.E+00	
	X(16) = VFW	LB/SEC	3.449E+02		0.E+00	0.E+00	-5.6E+01	
DERIVATIVE OF	X(19) = YR	NORMALIZED	4.641E-01	1.000E+00	0.E+00	1.6E-02	-7.4E-02	
	X(19) = YR	NORMALIZED/SEC		3.300E-12	-3.300E-02	5.1E-02	-1.4E-01	CONSTRAINT VIOLATED
DERIVATIVE OF	X(21) = YM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.8E-01	-1.0E+00	
	X(21) = YM	NORMALIZED/SEC		2.000E-11	-2.000E-01	2.9E-02	-1.0E+00	CONSTRAINT VIOLATED
	X(22) = HDASH	NORMALIZED	9.200E-01	9.700E-01		0.E+00	-3.2E-02	
	X(23) = YTCV	NORMALIZED	9.501E-01	1.000E+00	0.E+00	0.E+00	-1.8E-01	
	X(28) = IF1	INCHES	0.E+00	1.000E+00		5.0E-02	-3.6E-01	
	X(29) = IF2		0.E+00	1.000E+00		5.4E-04	-9.4E-03	
	X(30) = CF	C/S	0.E+00	1.000E+00	-1.000E+00	3.1E-01	-3.8E-02	
ADDITIONAL PARAMETERS								
	Y(3)-UD(6) = CL	INCHES	0.E+00	8.000E+00	-8.000E+00	3.7E-01	-1.0E+00	
DERIVATIVE OF	U2(4) = YPDEM	NORMALIZED	4.641E-01	1.000E+00	0.E+00	2.8E-02	-1.6E-01	
	XE(1) = PKM	/SEC		3.000E-05	-3.000E-05	4.9E-06	-2.8E-05	
	XA(5) = CDASH	NORMALIZED	1.014E+00	2.400E-01	PEAK TO PK	2.8E-02	-1.4E-01	
	XA(6) = CHSC	RTU/LB	1.857E+01		0.E+00	5.1E+00	-2.9E+00	
	XA(8) = CHB	RTU/LB	4.945E+01		0.E+00	0.E+00	-7.4E+00	
	XA(12) = KXCOUT		1.402E+00		0.E+00	0.E+00	-3.2E-01	
	XA(14) = KXRRAR		9.535E-01		0.E+00	0.E+00	-1.4E-01	

G.3. Case 0126 as plotted by SIMSAT (reduced set of feedback variables and D.C. compensation).

Following are the plots of the saturated simulation of the controlled plant for case (0126) of REGULA. Only the "dominant" variables are fed back (see section 6.3.4). D.C. compensation was used to force $x_{10}(p_d)$ and $x_{30}(\delta f)$ to settle at value 0 for step input.

The excitation is type (6), i.e. :

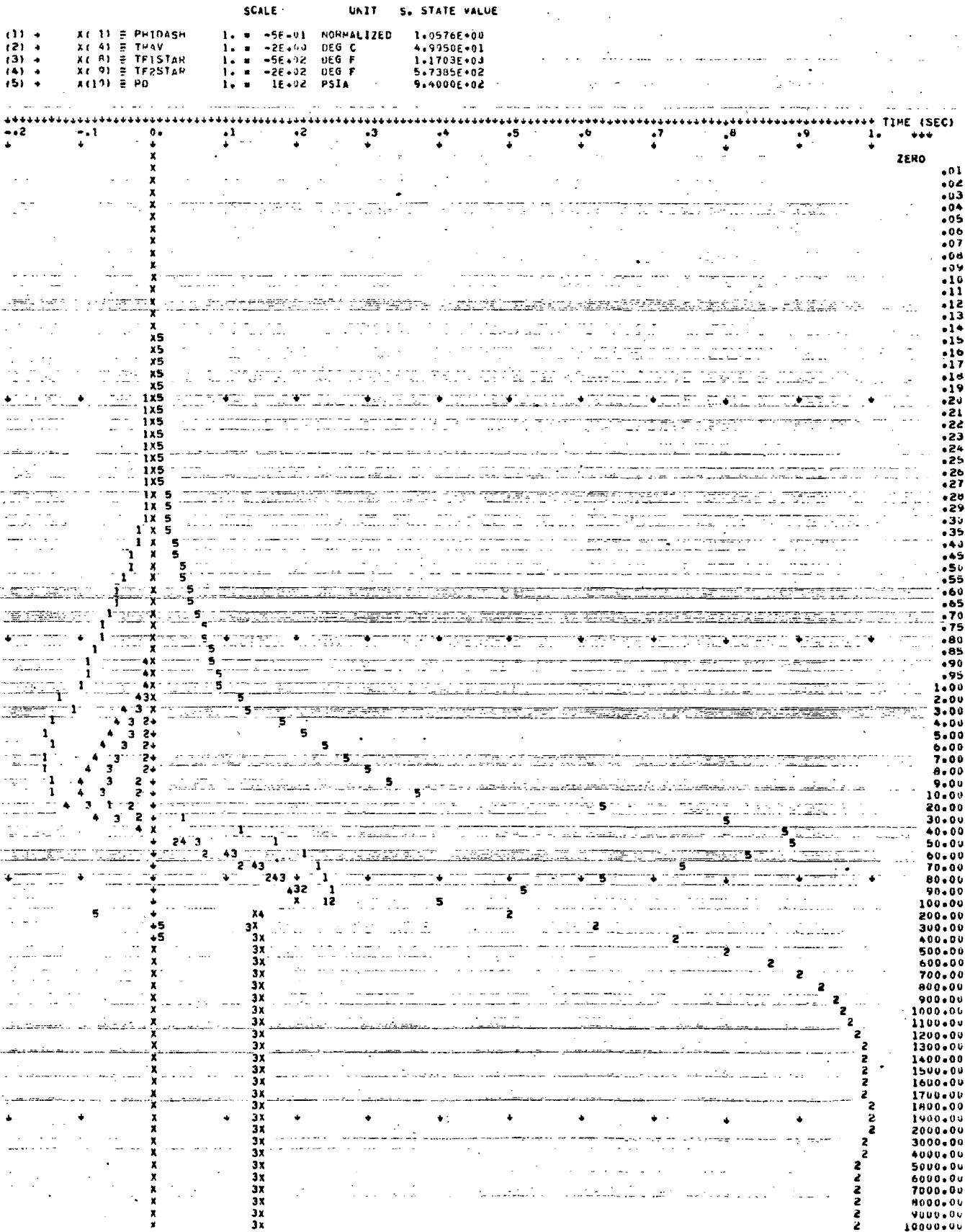
$$u_{d8} = -0.1 u_{s8}$$

where $u_{d8} \equiv$ Electrical power disturbance, step input.

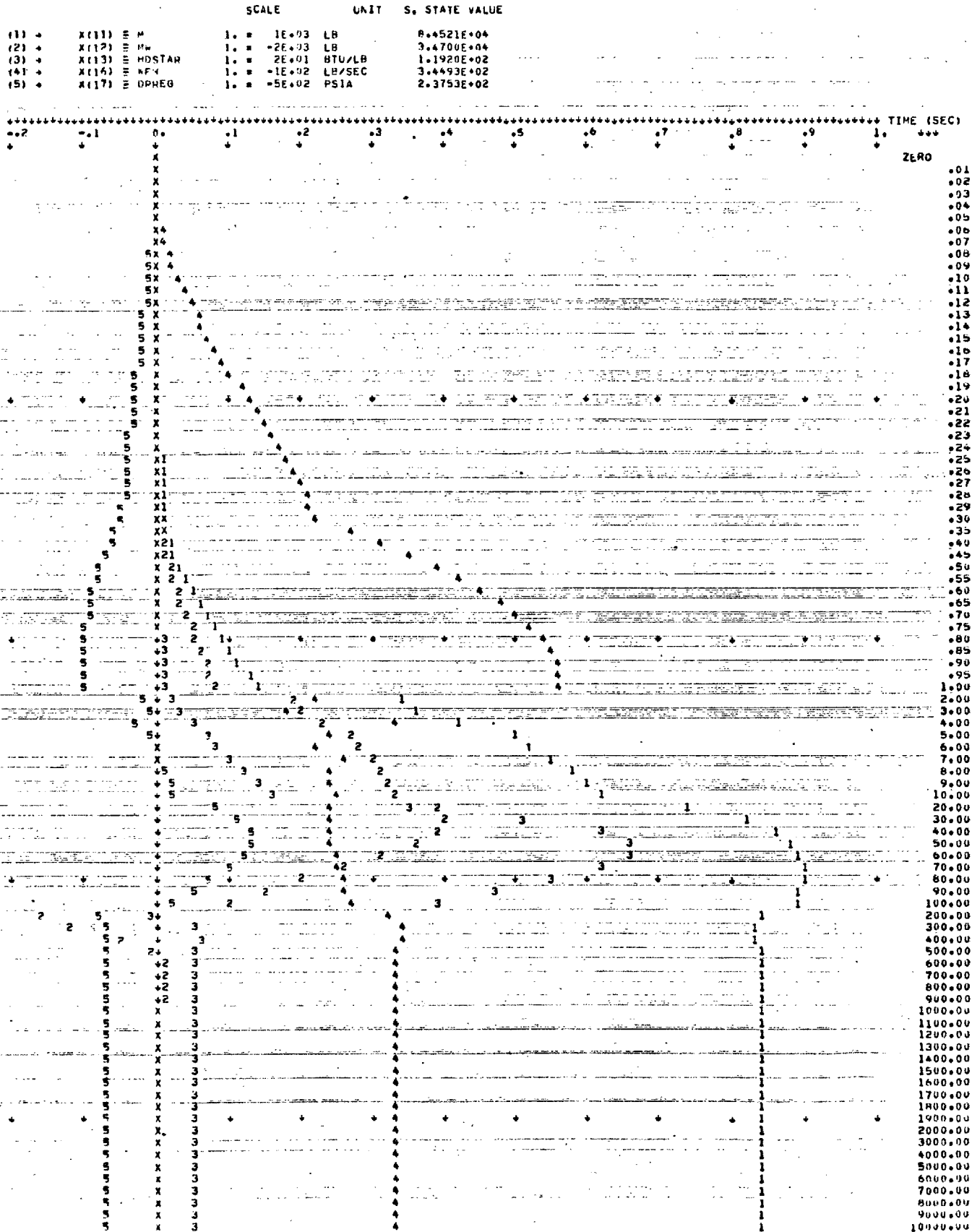
$u_{s8} \equiv$ Electrical power, steady state value.

See comments in introduction to appendix (G).

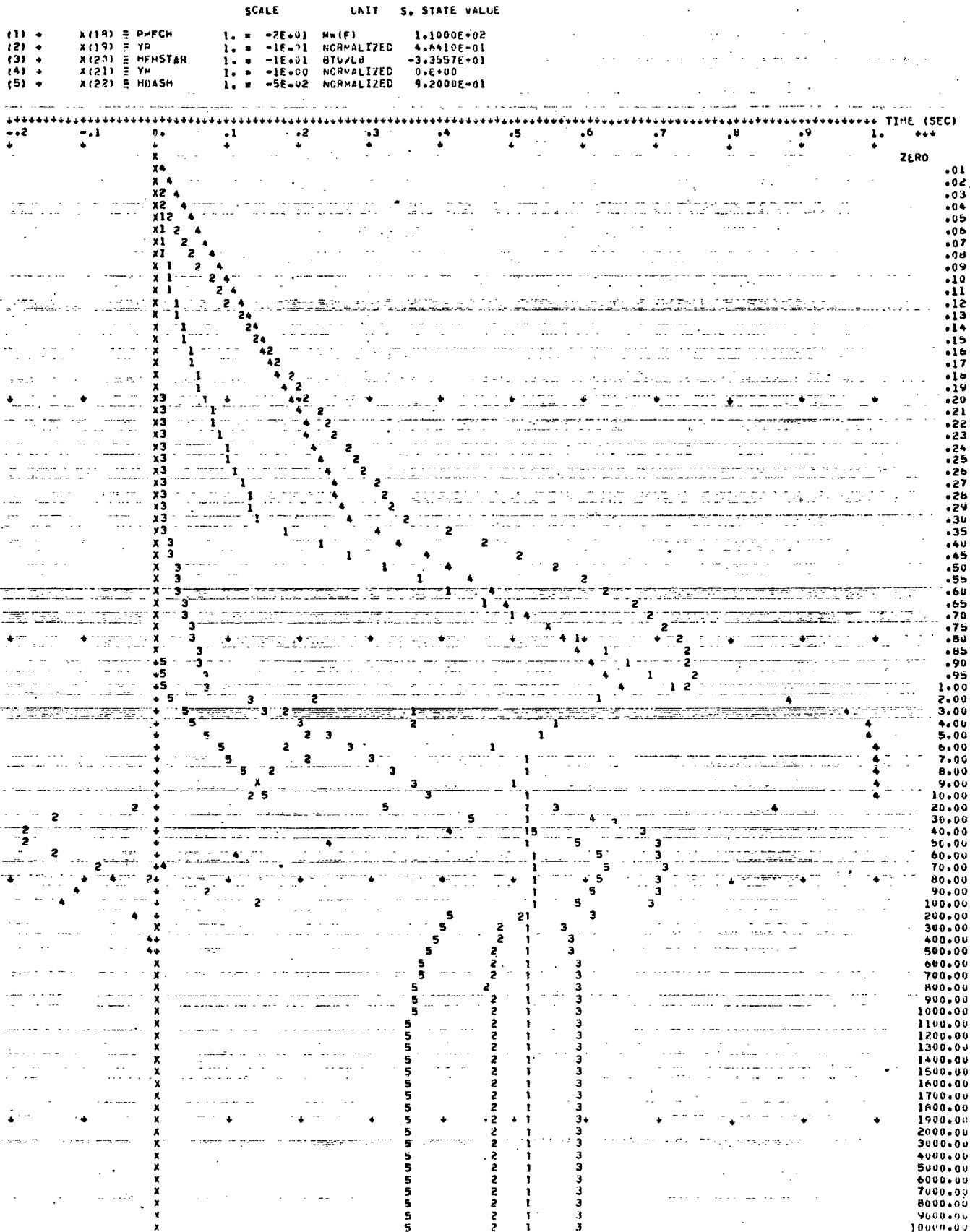
STEP RESPONSE OF SYSTEM ACC(30.30): CASE 126 OF REGULA FOR CASE 3: PRC TYPE (6)



STPP RESPONSE OF SYSTEM ALC(30,30) CASE 126 OF REGULA FOR CASE 3; PAC TYPE (6)

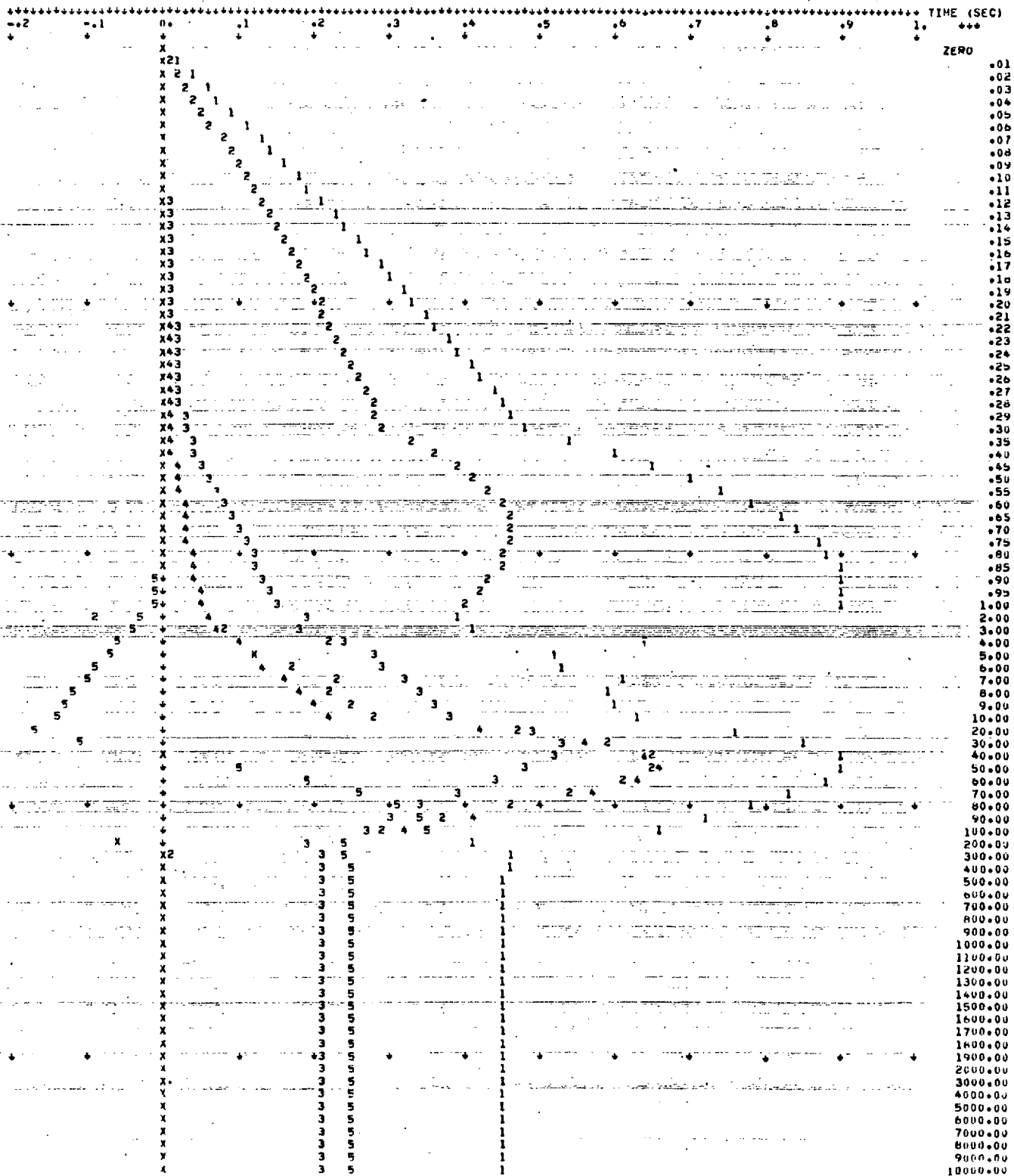


ST+P RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CASE 3, FAC TYPE (6)



STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CASE 3, PAC TYPE (6)

		SCALE	UNIT	S. STATE VALUE
(1) +	X(23) = YTCV	1. = -2E-01	NORMALIZED	9.5014E-01
(2) +	X(30) = DF	1. = 5E-01	C/S	0.E+00
(3) +	XA(2) = FRAR	1. = -1E-01		2.6452E-01
(4) +	XA(1) = TCOOLSTR	1. = 2E+01	DEG F	1.3524E+02
(5) +	XE(2) = TFAVSTAN	1. = -2E+02	DEG F	8.7209E+02



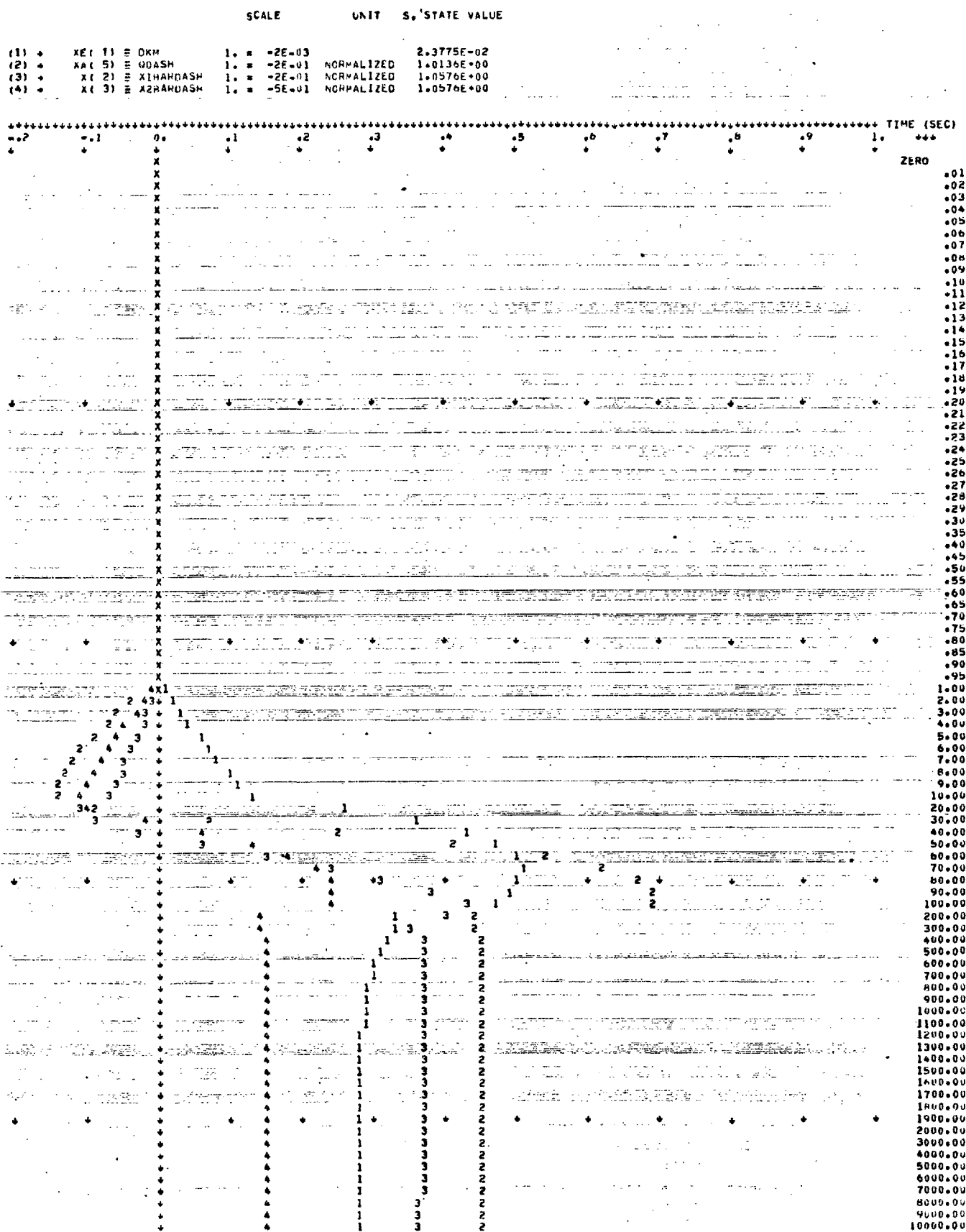
STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF HELIUM FOR CASE 3; FAC TYPE (6)

SCALE UNIT % STATE VALUE

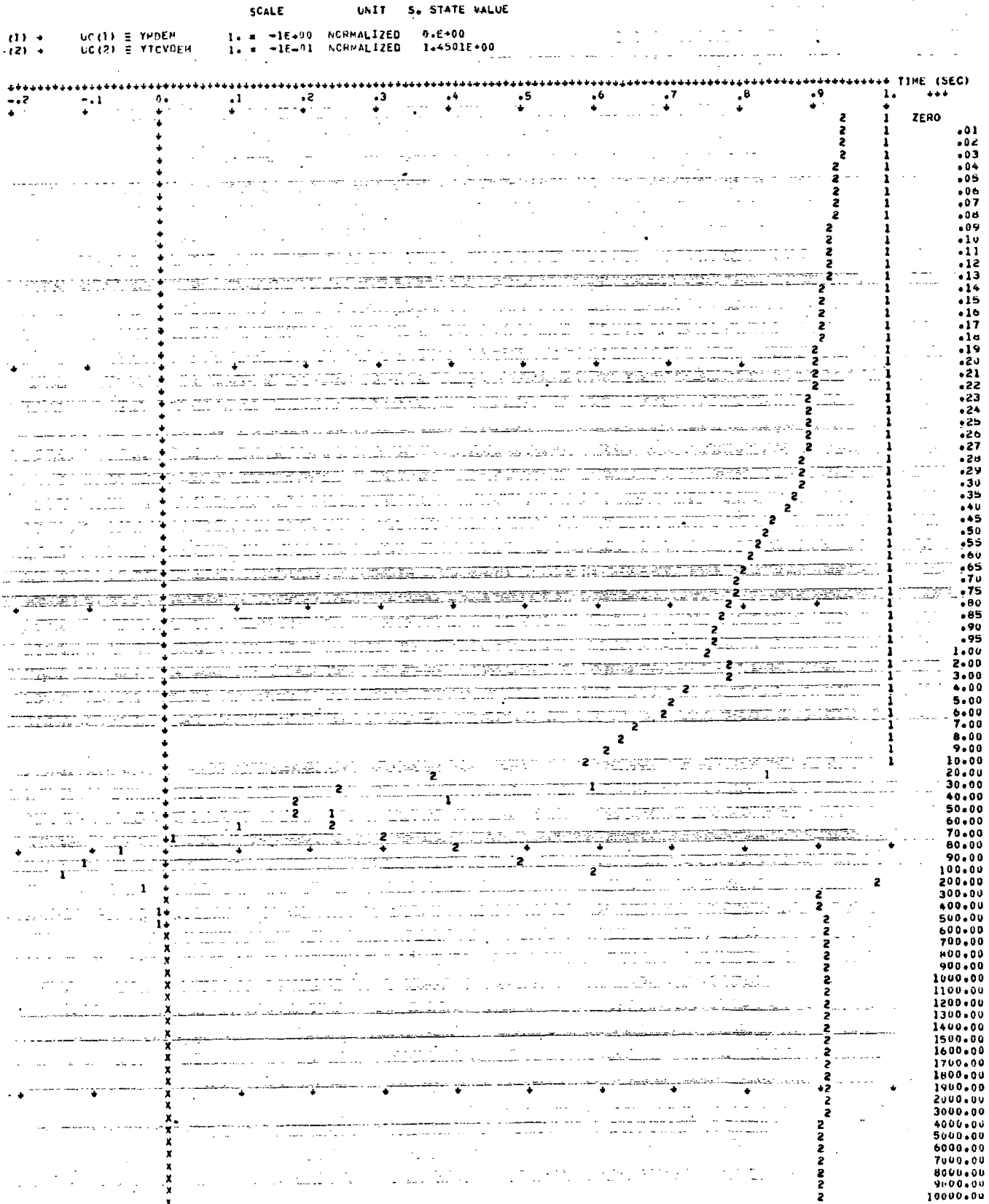
(1) * XE(7) = WSDRCH 1. = -1E+02 LH/SEC 3.4493E+02
 (2) * XE(12) = WW 1. = 5E+01 LH/SEC 4.1867E+03
 (3) * Y(3) = UN(6) = DL 1. = -5E+00 INCHES 0.E+00
 (4) * U2(4) = YRDEM 1. = -2E+01 NORMALIZED 4.6410E-01



STEP RESPONSE OF SYSTEM ACC(30,30), CASE 126 OF REGULA FOR CASE 3, PAC TYPE (6)



STEP RESPONSE OF SYSTEM ACC(33.30), CASE 126 OF REGULA FOR CASE 3: PAC TYPE (6)



STEP RESPONSE OF SYSTEM AOC(30,30), CASE 126 OF REGULA FOR CASE 3, EXC TYPE (6)

LIST OF CONSTRAINED PARAMETERS AND/OR DERIVATIVES (A VIOLATED CONSTRAINT YIELDS A WARNING)

NAME		UNIT	S.STATE	CONSTRAINTS		PEAK VALUES (LINEAR DEVIATIONS)		
				UPPER	LOWER	POSITIVE	NEGATIVE	
CONTROL INPUTS (UC)								
	UC(1) =	YMDEN	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.4E-01	-1.0E+00
	UC(2) =	YTCVDEM	NORMALIZED	1.450E+00	3.250E+00	0.E+00	0.E+00	-9.8E-02
DERIVATIVE OF	UC(2) =	YTCVDEM	NORMALIZED/SEC		3.300E-03	-3.300E-03	2.3E-02	-2.6E-03
STATE VARIABLES								
	X(1) =	RHIDASH	NORMALIZED	1.058E+00	2.000E-01	PEAK TO PK	7.5E-02	-1.2E-01
	X(10) =	PD	PSIA	9.400E+02	9.800E+02	9.000E+02	8.9E+01	-8.0E+00
DERIVATIVE OF	X(10) =	PD	PSIA /SEC			-1.000E+00	1.1E+01	-1.2E+00
	X(11) =	N	LB	8.452E+04		0.E+00	9.0E+02	0.E+00
	X(12) =	MW	LB	3.470E+04		0.E+00	3.2E+02	-8.1E+02
	X(15) =	OMEGA	RAD/SEC	1.516E+02		0.E+00	7.6E-02	0.E+00
	X(16) =	WFW	LB/SEC	3.449E+02		0.E+00	0.E+00	-5.6E+01
	X(19) =	YR	NORMALIZED	4.641E-01	1.000E+00	0.E+00	1.8E-02	-7.5E-02
DERIVATIVE OF	X(19) =	YR	NORMALIZED/SEC		3.300E-02	-3.300E-02	5.2E-02	-1.4E-01
	X(21) =	YM	NORMALIZED	0.E+00	1.000E+00	-1.000E+00	1.3E-01	-1.0E+00
DERIVATIVE OF	X(21) =	YM	NORMALIZED/SEC		2.000E-01	-2.000E-01	2.5E-02	-1.0E+00
	X(22) =	PDASH	NORMALIZED	9.200E-01	9.700E-01		0.E+00	-3.1E-02
	X(23) =	YTCV	NORMALIZED	9.501E-01	1.000E+00	0.E+00	0.E+00	-1.8E-01
	X(28) =	JF1	INCHES	0.E+00	1.000E+00		5.4E-02	-3.7E-01
	X(29) =	JF2		0.E+00	1.000E+00		8.1E-04	-9.4E-03
	X(30) =	CF	C/S	0.E+00	1.000E+00	-1.000E+00	3.2E-01	-4.5E-02
ADDITIONAL PARAMETERS								
	Y(3)=UD(6) =	DL	INCHES	0.E+00	8.000E+00	-8.000E+00	4.1E-01	-1.0E+00
	U2(4) =	YRDEM	NORMALIZED	4.641E-01	1.000E+00	0.E+00	2.8E-02	-1.7E-01
DERIVATIVE OF	XE(1) =	PKM	/SEC		3.000E-05	-3.000E-05	3.3E-06	-2.8E-05
	XA(5) =	GDASH	NORMALIZED	1.014E+00	2.400E-01	PEAK TO PK	2.8E-02	-1.4E-01
	XA(6) =	CHSC	RTU/LB	1.857E+01		0.E+00	5.1E+00	-2.8E+00
	XA(8) =	CMR	RTU/LB	4.945E+01		0.E+00	0.E+00	-7.3E+00
	XA(12) =	KXCOUT		1.402E+00		0.E+00	0.E+00	-3.2E-01
	XA(14) =	KXRBAR		9.535E-01		0.E+00	0.E+00	-1.4E-01

G.4. Checking of the linear model against the analog record
of the non-linear model.

Following are the plots of the free simulation of the uncontrolled plant. As mentioned in section (6.4) and elsewhere a special case was selected in the routine FDBCK1 of MXEV2X1 (see section 5.2.3): this was case (7) which differs from case (3) - used for the optimal control studies - in that it does not include load following. The reason for this difference was mentioned in the sections referred to above.

The model of case (7) in FDBCK1 was constructed with the sole purpose of comparing the responses of the linear model on the digital computer and the non-linear model on the analog computer.

The results were discussed in section (6.4).

The excitation is type (3), i.e.:

$$u_{d_1} = 10 \text{ miliniles} = 10^{-5}$$

$$\text{where } u_{d_1} \equiv \delta k_{bal}$$

See general comments in introduction to section (G).

Special comments:

(i) The parameters ticked (✓) at the tops of the plots indicate responses recorded on the analog computer for the non-linear model.

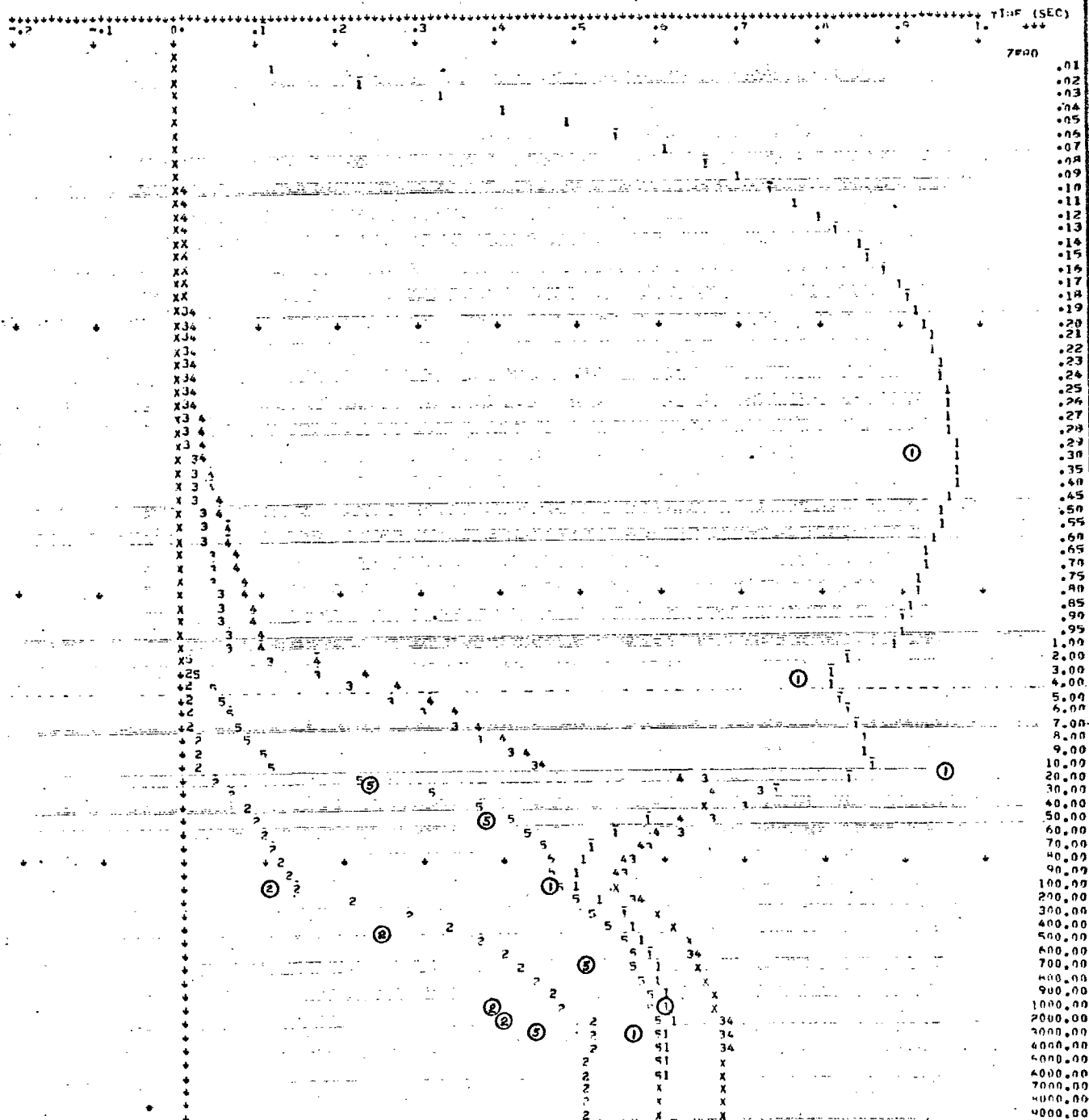
(ii) The numbers encircled indicate by their positions the values of the analog responses for the particular times involved, to be compared with the digital plot of the same number.

(iii) In the final table for constraints there are some differences with the tables for the other plots seen before:

the values of constraints for $u_{c_2}(Y'_{TCV})$, $x_1(\bar{\Phi}')$ and $x_{A_5}(Q')$ are different. This simulation was carried out at an earlier stage in the project. The difference does not affect the comparison of linear and non-linear models.

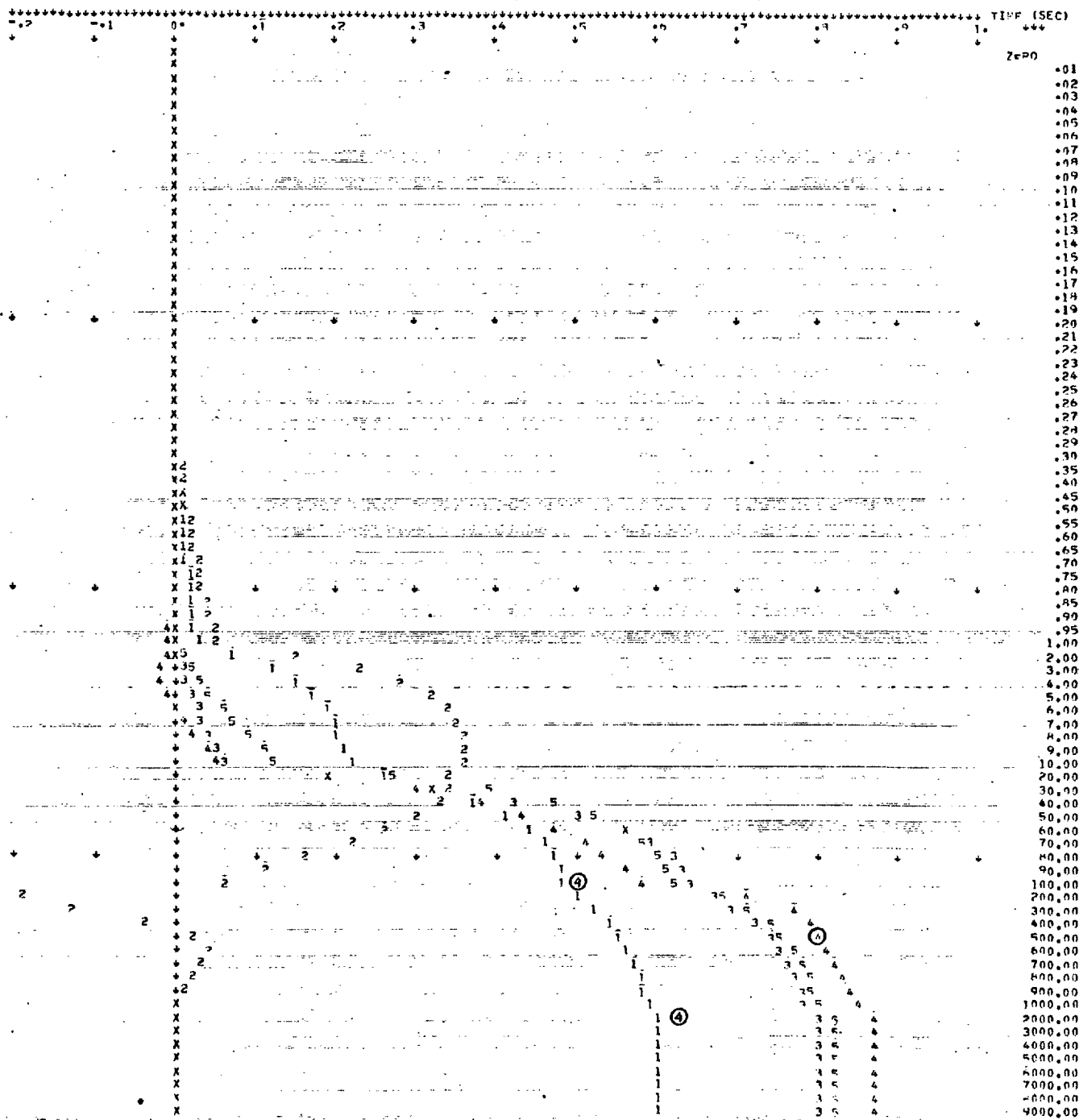
STEP DEPENDS OF SYSTEM (1, 12, 13) CASE 7 OF MODEL (EXC 7) END TYPE (2)

		SCALE	UNIT	S. STATE VALUE
✓(1) *	X(1) PHIDASH	1. = 2E+02	NORMALIZED	1.057AE+00
✓(2) *	X(4) THAV	1. = 5E+01	DEG C	4.975E+01
(3) *	X(8) TF1STAR	1. = 2E+01	DEG F	1.170E+03
✓(4) *	X(9) TF2STAR	1. = 1E+01	DEG F	3.733E+02
✓(5) *	X(10) PD	1. = 2E+01	PSIA	9.400E+02



STEP RESPONSE OF SYSTEM A (2,0,0), CARRY OF BLOCK (10,000), ONE TYPE (1)

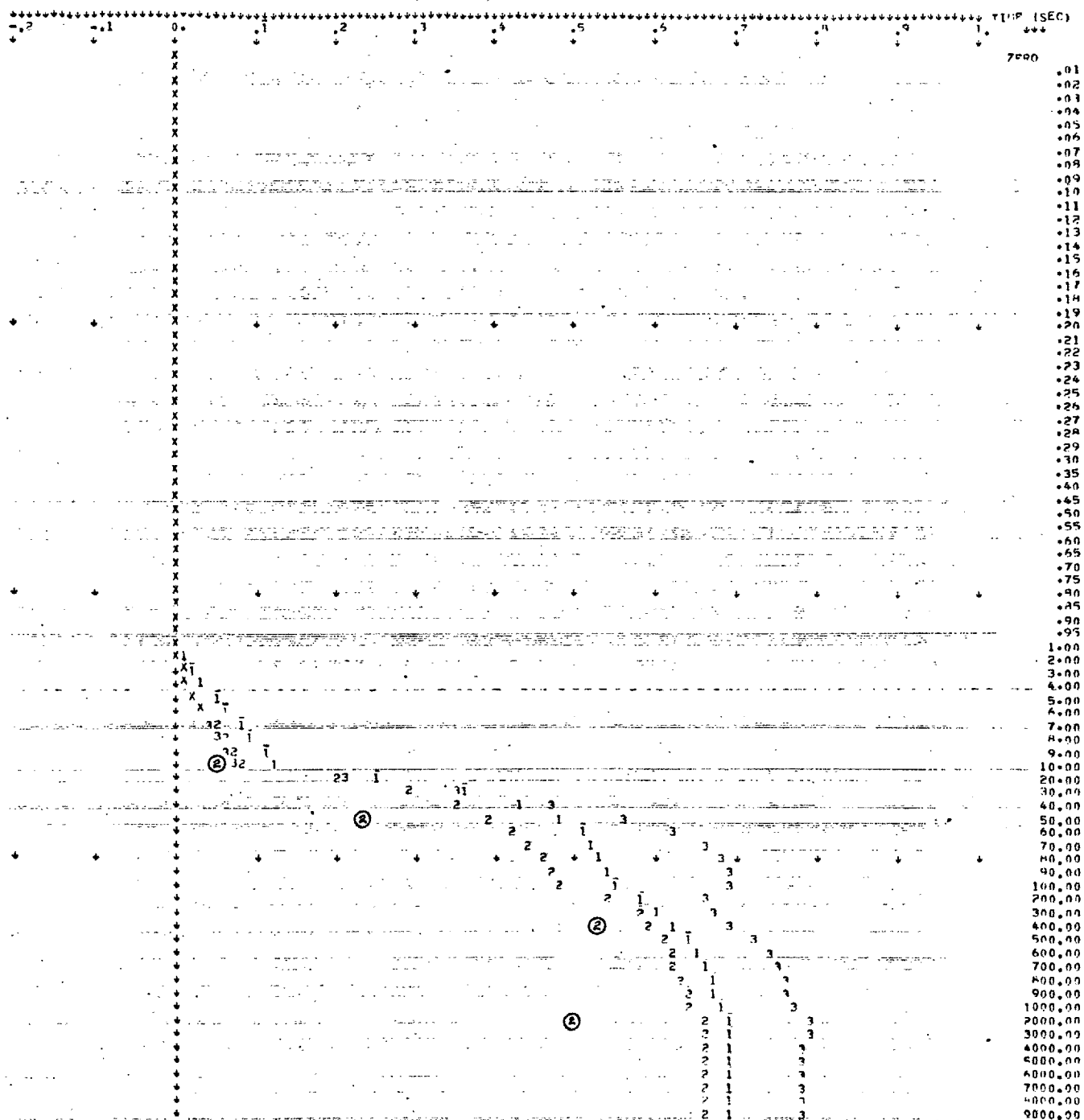
	SCALE	UNIT	S. STATE VALUE
(1) + X(11) = M	1. = -2E+02	LN	8.4521E+04
(2) + X(12) = M ₁	1. = 1E+02	LN	3.4730E+04
(3) + X(13) = HDSTAN	1. = 2E+00	BTU/LB	1.1420E+02
(4) + X(16) = WFA	1. = 5E+00	LN/SEC	3.4423E+02
(5) + X(17) = DPREG	1. = -2E+01	PSIA	2.3733E+02



STEP HEADLINE OF SYSTEM A (22,23) GAGE Y OF FINISH (100,00) , GAGE TYPE (1)

		SCALE	UNIT	% STATE VALUE	
(1) *	X(101) = PWECH	1. = 2E+00	MM(E)	1.1000E+02	
(2) *	X(109) = YP	1. = 2E+02	NORMALIZED	4.0413E-01	
(3) *	X(20) = HPHSTAR	1. = 1E+00	ATU(LB)	-1.3507E+01	
(4) *	X(21) = Y4	1. = 1E+00	NORMALIZED	0.	IN PLOT
(5) *	X(22) = HDASH	1. = 1E+00	NORMALIZED	9.2000E-01	NO PLOT

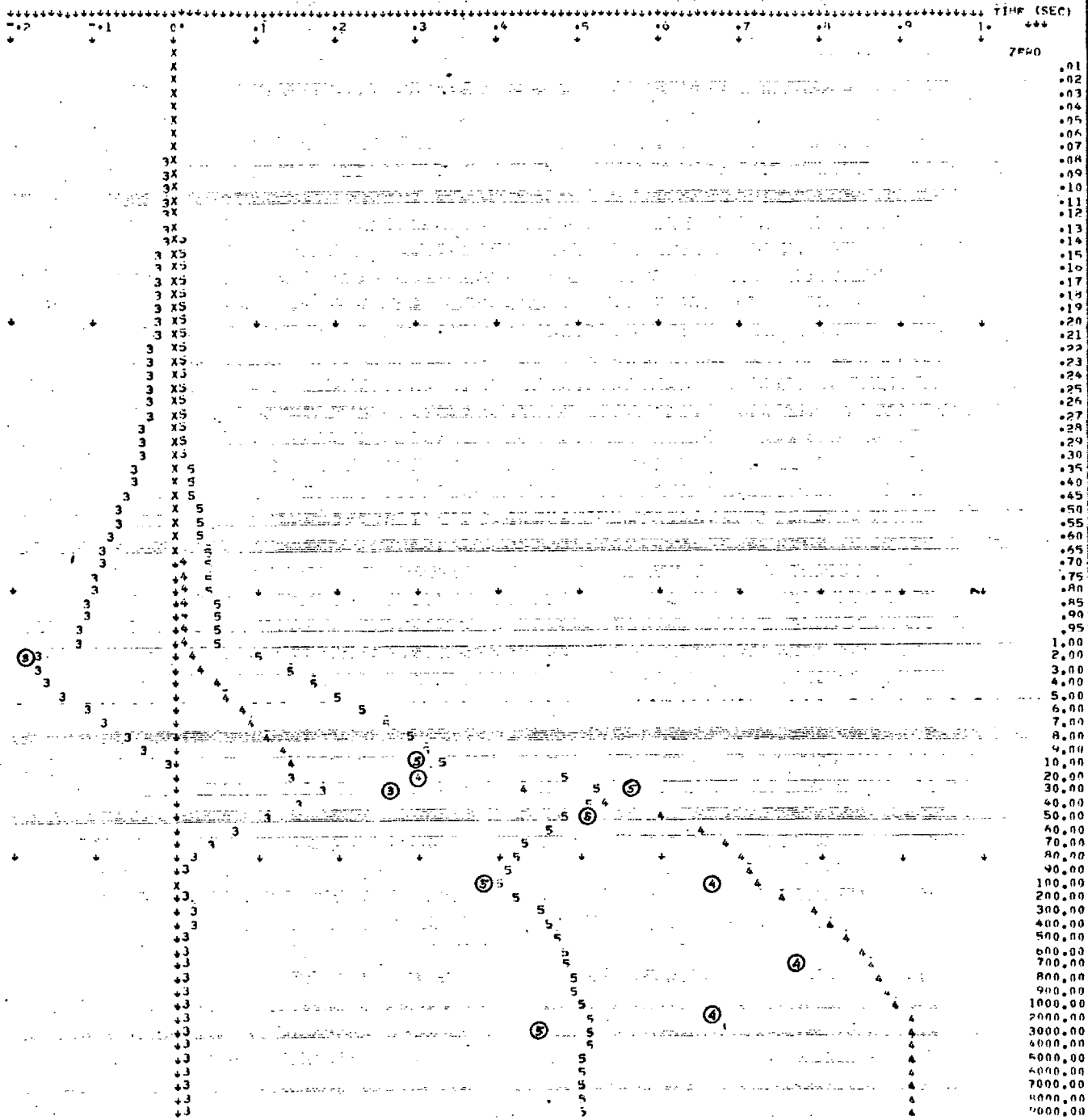
Negligible



STEP RESPONSE OF SYSTEM 4 (21,00), CASE 7 OF FORM 1 (XLEV21), EAC TYPE (3)

	SCALE	UNIT	S. STATE VALUE
✓(1) * X(23) = YTCV	1. = 1E+00	NORMALIZED	9.5014E-01 NO PLOT
✓(3) * X(1) = FVAR	1. = 5E+33		2.6232E-01
✓(4) * X(1) = TCOPLSTR	1. = 2E+00	DEG F	1.3534E+02
✓(5) * X(2) = TFAVSTAR	1. = 2E+01	DEG F	3.7209E+02

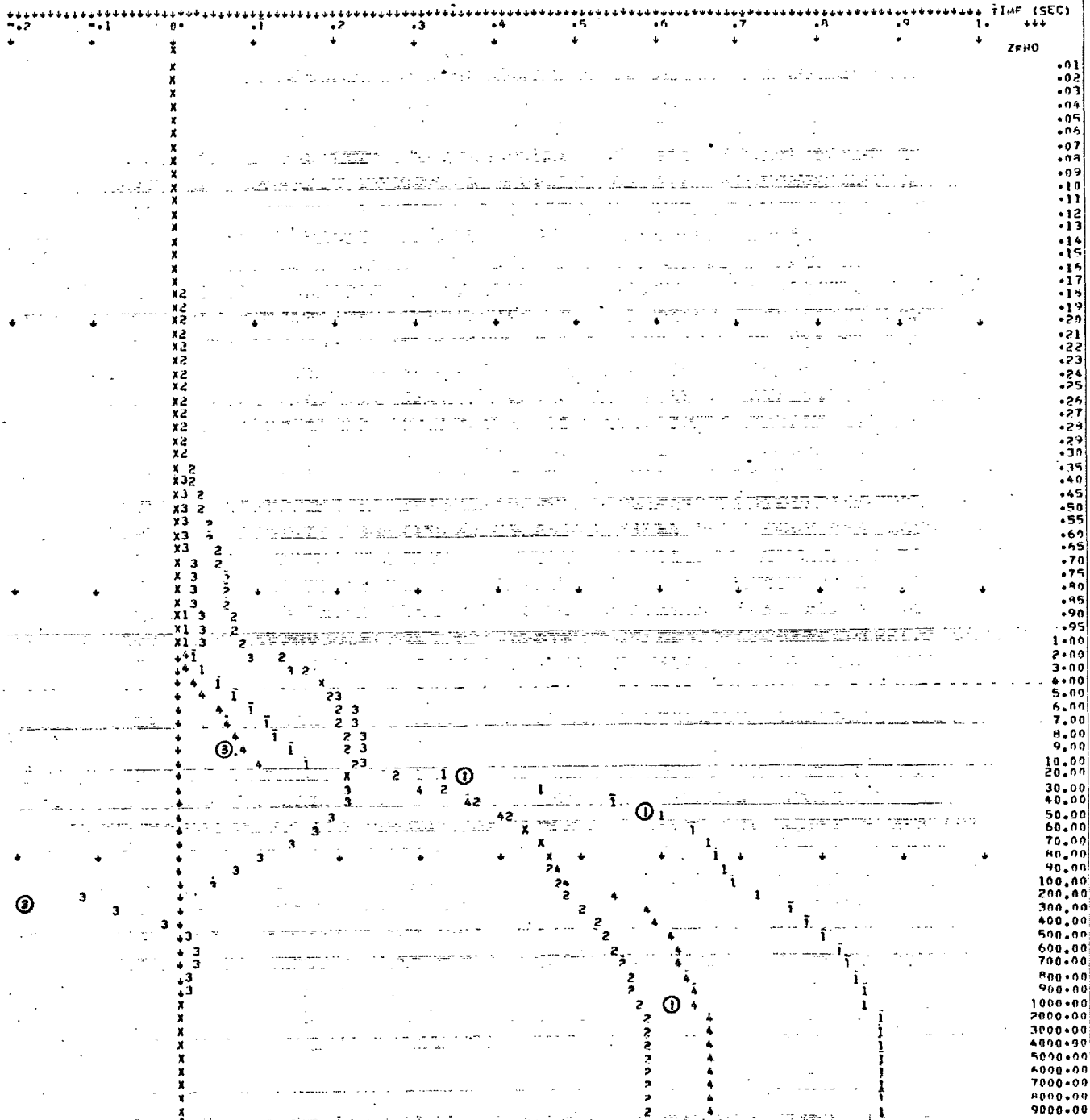
Negligible



STEP REINITIALISE OF SYSTEM A (21,00), CASE 7 OF BLOCK (10,000), ESC TYPE (3)

SCALE UNIT % STATE VALUE

✓ (1) * XE(7) = WSDRUM 1. = 5E+00 LB/SEC 3.4473E+02
 (2) * XE(12) = WW 1. = 1E+01 LB/SEC 4.1867E+03
 (3) * Y(3) = J(6) DL 1. = 2E+01 INCHES 0.
 (4) * U2(4) = YRDEM 1. = 2E+02 NORMALIZED 4.6410E-01



STEP RESPONSE OF SYSTEM A (29,29) CASE 7 OF FDBCK1 (MXEV2X1) , EXC TYPE (3)

LIST OF CONSTRAINED PARAMETERS AND/OR DERIVATIVES (A VIOLATED CONSTRAINT YIELDS A WARNING)

	NAME	UNIT	S.STATE	CONSTRAINTS		PEAK VALUES (LINEAR DEVIATIONS)	
				UPPER	LOWER	POSITIVE	NEGATIVE
CONTROL INPUTS (UC)							
	UC(1) = YMDEN	NORMALIZED	0.	1.000E+00	-1.000E+00	0.	0.
	UC(2) = YTCVDEM	NORMALIZED	1.450E+00	1.500E+00	-5.000E-01	2.	0.
DERIVATIVE OF	UC(2) = YTCVDEM	NORMALIZED/SEC		3.300E-03	-3.300E-03	0.	0.
STATE VARIABLES							
	X(1) = PHIDASH	NORMALIZED	1.050E+00	1.150E+00	9.576E-01	1.9E-02	0.
	X(10) = PD	PSIA	9.400E+02	9.800E+02	9.500E+02	1.2E+01	0.
DERIVATIVE OF	X(10) = PD	PSIA /SEC			-1.000E+00	2.5E-01	-1.8E-05
	X(11) = M	LB	8.452E+04		0.	0.	-1.2E+02
	X(12) = W	LB	3.470E+04		0.	3.6E+01	-1.9E+01
	X(15) = OMEGA	RAD/SEC	1.516E+02		0.	0.	-1.8E-03
	X(16) = WFW	LB/SEC	3.449E+02		0.	4.4E+00	-9.5E-02
	X(19) = YR	NORMALIZED	4.641E-01	1.000E+00	0.	1.3E-02	-2.1E-11
DERIVATIVE OF	X(19) = YR	NORMALIZED/SEC		3.300E-02	-3.300E-02	2.5E-04	-2.0E-08
	X(21) = YM	NORMALIZED	0.	1.000E+00	-1.000E+00	0.	-1.5E-26
DERIVATIVE OF	X(21) = YM	NORMALIZED/SEC		2.000E-01	-2.000E-01	0.	0.
	X(22) = HDASH	NORMALIZED	9.200E-01	9.700E-01		7.4E-18	-2.0E-15
	X(23) = YTCV	NORMALIZED	9.501E-01	1.000E+00	0.	0.	-5.3E-15
	X(28) = IF1	INCHES	0.	1.000E+00		1.6E-02	-7.2E-03
	X(29) = IF2		0.	1.000E+00		2.6E-03	-4.5E-14
ADDITIONAL PARAMETERS							
	Y(3)-UD(6) = DL	INCHES	0.	8.000E+00	-8.000E+00	4.5E-02	-2.4E-02
	U(4) = YRDEM	NORMALIZED	4.641E-01	1.000E+00	0.	1.3E-02	-2.1E-09
DERIVATIVE OF	X(1) = DKM	/SEC		3.000E-05	-3.000E-05	0.	0.
	XA(5) = QDASH	NORMALIZED	1.014E+00	1.144E+00	8.836E-01	1.3E-02	0.
	XA(6) = OHSC	BTU/LB	1.857E+01		0.	5.0E-01	0.
	XA(8) = DHA	BTU/LB	4.945E+01		0.	6.1E-01	0.
	XA(12) = KXCOUT		1.402E+00		0.	7.9E-03	0.
	XA(14) = KXRBAR		9.535E-01		0.	2.8E-03	0.