

DOE/OR/00033--T201

Received by CSTI

JUL 18 1986

3.0
5/10

A REFLECTED KINETICS MODEL FOR NUCLEAR
SPACE REACTOR KINETICS AND CONTROL SCOPING CALCULATIONS

DOE/OR/00033--T201

DE86 013123

A Dissertation

by

KENNETH EDWARD WASHINGTON

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the requirement for the degree of
DOCTOR OF PHILOSOPHY

May 1986

APPROVED FOR RELEASE OR
PUBLICATION - O.R. PATENT GROUP
BY EA DATE 7-10-86

MASTER

Major Subject: Nuclear Engineering

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.


A REFLECTED KINETICS MODEL FOR NUCLEAR
SPACE REACTOR KINETICS AND CONTROL SCOPING CALCULATIONS

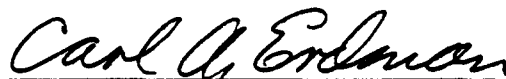
A Dissertation

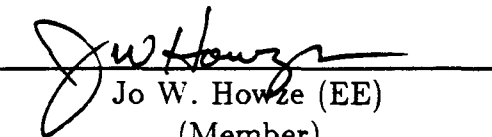
by


KENNETH EDWARD WASHINGTON


Approved as to style and content by:


Gerald A. Schlapper (NE)
(Chairman of Committee)



Carl A. Erdman (NE)
(Member)


Jo W. Howze (EE)
(Member)


K. Lee Peddicord (NE)
(Member)


K. Lee Peddicord
(Head of Department)

May 1986

APPROVED FOR PUBLICATION
PUBLICATION - C.R. FAULT GROUP
BY  DATE 7-10-86

ABSTRACT

A Reflected Kinetics Model for Nuclear Space

Reactor Kinetics and Control Scoping Studies. (May 1986)

Kenneth Edward Washington, B.S., Texas A&M University;

M.S., Texas A&M University

Chairman of Advisory Committee: Dr. Gerald A. Schlapper

Renewed interest in space nuclear applications over the past several years has motivated the study of a specialized reactor kinetics model. Long term civilian and dedicated military missions warrant the consideration of a kinetics model favorable for study of the feasibility of automatic control of these devices. The need to bridge this gap between reactor kinetics and automatic control in conjunction with the control drum design characteristic of next generation "paper" space reactors inspired the development of a new Reflected Kinetics (RK) model. An extension of the conventional point kinetics (PK) model was done in order to explicitly correlate reactivity and the reflector/absorber control drums characteristic of space nuclear reactor designs. Open loop computations and numerical comparisons to analytic PK equations indicated that the RK model is a functional alternative to "equivalent bare point kinetics" in the analysis of moderate transients. It was shown that variations in the RK reflector-to-core transfer probabilities and coolant flow rate do indeed drive the transient differently than the lumped insertion of equivalent reactivity amounts in the core. These computations illustrated the potential importance of the utilization of variable coolant flow rate to aid control in space reactor

systems limited by minimal drum reactivity worth. Additionally the Doppler reactivity shutdown mechanism was concluded to be the primary reliable means of safety shutdown in such systems. The structure of the RK equations proved to be advantageous for integration of automatic control. Unity feedback and two existing control techniques, state feedback and modal, were applied in the solution of step and ramp servo mechanism problems. Advantages and disadvantages of state feedback control and the recently developed modal control as applied to the RK model were identified. Numerical computations of the resulting closed loop RK equations lead to the conclusion that the drawbacks of a large closed loop system and blind eigenvalue assignment in the state feedback case outweigh the advantage of definite system stability. Even though stability cannot be guaranteed in the modal case, it was found that the structure of the RK equations was well suited for the success of this output control technique. Overall it was found that the application of automatic control in space nuclear systems is readily done under the RK model due to the structure of the reactivity inputs in the model. The needs for further study in the analysis of existing designs and the solution of the nonlinear equations were identified.

ACKNOWLEDGMENTS

The author wishes to thank Dr. Gerald Schlapper for his inspiration of this project and his motivational and technical support as committee chairman throughout the entirety of the endeavor. Thanks is also given to Dr. Jo Howze for the sharing of his expertise in the control theory field. Acknowledgement is given to the Air Force Weapons Laboratory for their financial support of this research project. Acknowledgement is also given to the Computer Graphics Facility at Texas A&M University for the use of their computational facilities, plotting equipment, and laser printer. The reasearch was performed under appointment to the Nuclear Engineering, Health Physics, and Radioactive Waste Management Fellowship program administered by Oak Ridge Associated Universities for the U.S. Department of Energy. Finally, I would like to thank my wife, Connie, for her unselfish giving, support, and motivation throughout the entirety of my post graduate education.

TABLE OF CONTENTS

CHAPTER	Page
I	INTRODUCTION AND LITERATURE REVIEW 1
	A. Introduction 1
	B. Objectives 1
	C. Literature Review : Reactor Kinetics Models 2
	D. Literature Review : Nuclear Applications of Automatic Control 4
	E. Proposed Kinetics Model 4
	F. Numerical Methods 6
II	DERIVATION OF THE REFLECTED KINETICS MODEL 7
	A. Introduction 7
	B. Point Kinetics Equations 8
	C. Reflected Kinetics Model 10
	D. Definition of Transfer Probabilities 14
	E. Linearization of the RK Equations 17
	F. Criticality Conditions 20
	G. Summary 22
III	PRELIMINARY VALIDATION OF THE RK MODEL 23
	A. Introduction 23
	B. Analytic Solutions to the RK and PK Equations 24
	C. ASH Exponential Operator Methods 26
	D. Numerical RK, Analytic RK, and Analytic PK Comparisons 29
	E. Comparison of RK and PK Response Functions 36

TABLE OF CONTENTS (Continued)

CHAPTER		Page
IV	THERMAL MODELLING AND TEMPERATURE FEEDBACK EFFECTS	40
	A. Introduction	40
	B. Derivation of the Lumped Parameter Equations	40
	C. Temperature Feedback Effects	45
V	INTEGRATION OF AUTOMATIC CONTROL AND THE RK MODEL	48
	A. Introduction	48
	B. Controllability and Observability	49
	C. Tracking Under the Internal Model Principle	52
	D. Pole Placement Algorithm	54
	E. Observer Design	55
	F. Modal Control Theory	60
VI	NUMERICAL COMPUTATIONS AND RESULTS	65
	A. Introduction	65
	B. Open Loop Numerical Computations	66
	C. State Feedback Numerical Computations	80
	D. Modal Control Numerical Computations	89
VII	SUMMARY AND CONCLUSIONS	97
	A. Summary	97
	B. Conclusions	98
	C. Suggestions for Further Work	101

TABLE OF CONTENTS (Continued)

CHAPTER	Page
REFERENCES	104
APPENDIX A : RK PLANT MATRIX STRUCTURE	107
APPENDIX B : ASH METHODOLOGY	110
APPENDIX C : CODE LISTINGS	114
VITA	154

LIST OF TABLES

Table	Page
I. Definition of Commonly Used Reactor Kinetics Parameters and the RK Transfer Probabilities	9
II. Available Neutronic Data for the Doubly Reflected MURR Facility	30
III. Equivalent Singly Reflected and Bare System Parameters for the MURR Data Set	32
IV. Comparison of the RK Open Loop Eigenvalues and the Poles in the RK Analytic Solution	35
V. Material Properties and Other Pertinent Data for Open and Closed Loop RK Calculations	68
VI. Eigenvalue Comparison for the Zero Temperature Feedback and Reduced Neutron Lifetime Case	68
VII. Eigenvalue and G Matrix Selection for State Feedback Computations	82

LIST OF FIGURES

Figure	Page
1. Bare Core Neutron Flow Diagram	11
2. Reflected Reactor Neutron Flow Diagram	13
3. Comparison of Analytic and Numerical Solutions to the RK Equations	33
4. Comparison of Analytic PK and the Numerical Solution to the RK Equations	34
5. Core Neutron Response for Three Different Reactivity Input Cases	69
6. Core Temperature Change for Three Different Reactivity Input Cases	69
7. Comparison of Neutron Response for Two No Temperature Feedback Cases	73
8. Early Time Response for Varying Core Neutron Lifetimes	73
9. Core Neutron Response for h_f and Σ_f Parameter Variations	74
10. Core Temperature Change for h_f and Σ_f Parameter Variations	74
11. Relative Precursor Concentrations for the Six Delayed Group Case	75
12. Relative Neutron Response for Six Delayed Group Case	75
13. Neutron Response for a 2 Dollar Pulse	76
14. Core Temperature for a 2 Dollar Pulse	76
15. Delayed Precursor Response for a 2 Dollar Pulse	77

LIST OF FIGURES (Continued)

Figure	Page
16a. Case G1 State Feedback Solution to the Ramp Servo Problem	83
16b. Case G1 State Feedback Solution to the Step Servo Problem	83
17. Relative Neutron Response for Case G1 Through G6 for the Ramp Reference Signal	85
18a. Transfer Probability Inputs for the Ramp Cases G1, G2, G4, and the Step R Case	86
18b. Transfer Probability Inputs for the Ramp Cases G3, G5, and G6	86
19a. Coolant Flow Rate Input for the Ramp Cases G1, G2, G4, and the Step R Case	87
19b. Coolant Flow Rate Input for Ramp Cases G3, G5, and G6	87
20a. Core Temperature for Ramp Cases G1, G2, and G4	88
20b. Core Temperature for Ramp Cases G3, G5, and G6	88
21. State Feedback Soltution for an Arbitrarily Chosen Eigenvalue Set	90
22a. Modal Solution for 1 Delayed Precursor Group, Ramp R Problem	93
22b. Modal Solution for 6 Delayed Precursor Groups, Ramp R Problem	93

LIST OF FIGURES (Continued)

Figure	Page
23a. Transfer Probability Inputs for the 1 and 6 Group Modal Solutions, Ramp and Step R Problems	95
23b. Coolant Flow Rate Inputs for the 1 and 6 Group Modal Solutions, Ramp and Step R Problems	95
24. Modal Solution for the 1 Delayed Precursor Group, Step R Problem	96
25. Open Loop Rk Solution Using State Feedback Determined Reactivity Inputs	96

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

A. Introduction

Over the past several years, there has been renewed interest in nuclear space reactor applications. The concept of using nuclear reactors in space, however, is not a new one. Man has been designing nuclear devices for space since the early 1960's in the forms of radioisotope thermoelectric generators (RTGs), small power level reactors, and rocket propulsion systems¹. This early nuclear space effort came to a halt in 1973. Much of the recent interest in nuclear energy in space has been caused by the success of the Space Shuttle program. The Space Shuttle opens the door for long duration civilian and military space missions. One major obstacle in the development of such technology is a feasible source of energy for the space environment. This need has promoted the consideration of nuclear power for space applications.

B. Objectives

One of the most important considerations in using nuclear reactors in space is that of reactor kinetics and control. Certain aspects of space reactors that differ from conventional light water reactors (LWRs) include: the replacement of control rods with reflecting control drums, a higher energy neutron spectrum, and more demanding power responses. These differences warrant special considerations in the development of a reactor kinetics model. Furthermore, potential military space needs also warrant the consideration of intelligent ways to manipulate the control

¹ This Dissertation follows the style of *Nuclear Science and Engineering*

inputs of such complicated systems. Unfortunately the regulatory structure that exists in the nuclear industry has limited the utilization of modern control methods in home based nuclear systems. This has stunted the development and use of control techniques as applied to the nuclear reactor kinetics and has therefore also had a detrimental effect on this phase in the development of the space nuclear program. Other factors that have prevented the development of automatic controllers for space nuclear applications are the difficulties posed by the integration of the point kinetics equations and existing modern multivariable control models. The objective of this research is to develop a model that offers an alternative to the point kinetics (PK) modelling approach in the analysis of space reactor kinetics and control studies. Modelling effort will focus on the explicit treatment of control drums as reactivity input devices so that the transition to automatic control can be smoothly done. The proposed model is developed for the specific integration of automatic control and the solution of the servo mechanism problem. The integration of the kinetics model with an automatic controller will provide a useful tool for performing space reactor scoping studies for different designs and configurations. Such a tool should prove to be invaluable in the design phase of a space nuclear system from the point of view of kinetics and control limitations.

Before discussing the proposed system model and control methods, the literature will be reviewed in two areas. The first area is that of nuclear reactor kinetics models. Existing control techniques applied to nuclear systems and other multivariable methods that appear to be well suited for nuclear systems are then reviewed.

C. Literature Review : Reactor Kinetics Models

A multitude of models and solution methods exist in the literature for the

study of conventional light water reactor kinetics. Probably the most widely known and used of these approaches is the point kinetics (PK) model. The simplicity yet reasonable accuracy of this model when applied to thermal reactors has been the source of its extreme popularity. *Nuclear Reactor Analysis*, by Duderstadt and Hamilton², and *Dynamics of Nuclear Reactors* by Hetrick³ give good reviews of the theory, application, and limitations of the PK model. The main limitation of the standard PK approach from the space reactor perspective is the bare reactor assumption. Another limitation is that the reactivity expression in the PK equations is lumped into the "reactivity" parameter which must be known in order to numerically compute the transient. These limitations make it difficult to treat the effect of controlling reflectors.

The success of the PK approach has dictated its modification and extension into many different forms. Among the most common of these are the adiabatic and quasistatic methods.⁴⁻⁷ While these methods relax the fundamental mode limitation, they still do not treat external reflectors as a direct means of reactivity control. Wasserman⁸ developed a simple model to account for reflected neutrons by treating them as additional delayed neutron groups. The disadvantage of this approach is that information regarding the reaction rates in the reflectors are lost. This information is needed in order to simplify the use of automatic control in the positioning of the reflectors. Various models with special application to coupled reactor cores are presented in *Coupled Reactor Kinetics*⁹. While many of the models are both simple and applicable to reflected systems, the consideration of automatic control was never made.

Other reactor kinetics models involve spatial and nonlinear effects¹⁰⁻¹², modal expansions^{13,14}, and finite difference techniques^{15,16}. A comprehensive overview of

space-time kinetics is found in *Space-Time Nuclear Reactor Kinetics* by Stacey¹⁷.

D. Literature Review : Nuclear Applications of Automatic Control

General concepts in multivariable control theory can be found in *Linear System Theory and Design* by Chen¹⁸. Most of the work done in the nuclear field in automatic control has been limited to conventional light water reactors (LWRs). Owens¹⁹ addresses the subjects of controllability and observability in cylindrical reactors. Weaver and Vanasse²⁰ applied modern control theory techniques to multiregion reactors using full state feedback. Oohori²¹ studied time optimal control for coupled core reactors which could be modified to work with space reactor designs. Numerous authors have presented control schemes for the solution of the Xenon oscillation problem²²⁻²⁴; however, these control schemes are not applicable to the proposed problem since Xenon oscillations are not an anticipated concern in space reactors. Tsuji²⁵ and Cherchas²⁶ present control schemes with application to load following of LWR reactors. Miller²⁷ developed a controller for the study of the effects of sensor failure in LWR systems. These methods are limited to LWR analysis. In order to consider the treatment of control drums modifications of these models would have to be made. Furthermore, often the rigor presented in these models is unwarranted since in a compact, medium spectrum space reactor, the point reactor model is adequate. On the other hand, the bare core limitation of the PK equations and other known variations are too restrictive for our purposes.

E. Proposed Kinetics Model

Perhaps the most important thing to remember in the development of a space reactor kinetics model is that the reactivity inputs that provide power level changes

are primarily due to the reflecting control drums. Furthermore, the effect of the control drums on the core neutronics may change as the power sequence progresses. The model proposed in this research is an extension of the traditional point kinetics approach to account for the influence of these reflecting control drums on core reactivity directly. If the reflectors are taken to be entities that either add or take away neutrons from each generation, then the point kinetics equations can be recast with a few additional terms. These additional terms correspond to the injection of neutrons into the core from the reflectors and the leakage of neutrons from the core to the reflectors. Additional equations also will appear which describe the balance of neutrons in each of the controlling reflectors. The resulting set of equations can be written without reference to a reactivity input since the only external sources of neutrons to the core multiplying medium are the reflecting control drums. Using the above model, we will be able to compute responses of various open loop systems suited for space reactor applications with a minimum of computational effort. This model will provide the basis of the development of more comprehensive system models that include Doppler feedback, heat removal, thermophysical effects, and automatic control applications. Thermal modelling will be handled with the lumped parameter approach. This will involve the homogenization of the core and coolant regions. The structure of the model described above will permit the explicit treatment of temperature feedback effects on material properties and the leakage of neutrons from the core. The transition to automatic control is demonstrated by the application of two known controller methodologies. The first is the well studied state feedback approach. As an alternative to the brute force state feedback approach, modal control developed by Andry²⁸ and successfully applied to the control of aircraft will be considered. This form of output feedback control will prove to be

particularly well suited for the solution of the RK servo mechanism problem.

F. Numerical Methods

The linear dynamical equations in open loop and closed loop form are usually written in matrix notation as,

$$\dot{X} = AX + S, \quad S = BU. \quad (1.1)$$

Conventional methods of solving this matrix equation include various finite difference (FD) routines of which Runge Kutta is probably the most common. However, if the equations are stiff (both large and small eigenvalues in the A matrix), FD methods require extremely small time steps for the convergence to accurate solutions. This can lead to the excessive use of computer time in obtaining accurate late time solutions. Considerable success has been demonstrated in solving linear time invariant equations with an exponential operator technique referred to as ASH²⁹. More recent applications of ASH have also included the solution of stiff, highly coupled dynamical equations resulting from high order spatial modelling of partial differential equations^{30,31}. The dynamical equations resulting from the development of the kinetic model here is expected to present the same difficulties as these previously studied problems. Therefore, a computational advantage may exist in the use of the ASH exponential operator method for the solution of the developed kinetics equations. The ASH method applied to the linear RK dynamical equations will be compared to Runge Kutta methods with regard to computational efficiency and solvability. A straight-forward extension of the ASH methodology is also developed that allows for the ready treatment of elementary time variant source terms.

CHAPTER II

DERIVATION OF THE REFLECTED KINETICS MODEL

A. Introduction

Before one can adequately model the behavior of any system, the aspects which are to be the focus of analysis must be identified. In particular the features of the model that set it aside from other systems should be considered. Since the rigorous treatment of all the phenomena that occur during a reactor transient is not very practical, the present research will focus on the development of a model that will account for the differences between the control of proposed space reactors and conventional LWR reactors. Such a model will provide an alternative to the point kinetics equations in the analysis of space reactors with exterior control drums. Furthermore, the structure of the derived model will prove to be well suited for the use of automatic controllers. This chapter is limited to the modelling of the neutronic behavior of the core and reflector regions. This will provide a foundation upon which the thermal transport model and automatic controllers can be built. Discussion of this theory is reserved for later chapters. Three aspects common to most space reactor designs that warrant special consideration include:

- A space reactor typically obtains positive and negative reactivity input by means of reflecting/absorbing control drum movement.
- The application of such reactors in the space environment usually will dictate the need to change operating power levels more suddenly and rapidly than typically encountered in a power reactor.
- The average neutron energy in reactors of the type considered here is typically

10 KeV as opposed to the thermal energy associated with large power reactors.

From the point of view of kinetics and control, the first point is the most significant and therefore will be the driving motivation behind the model development. The second point provides the primary motivation behind the later consideration of the feasibility of automatically controlled space reactors. The final point renders the point core simplification in the developed model valid.

The reflected kinetics (RK) model is derived by extending the standard PK equations to account for the effect of movable control reflectors. This will allow certain desirable features of the PK approach to be maintained while those that conflict with the needs of our model are modified accordingly. In order to do this, it will be necessary to examine the point kinetics equations from a different point of view than that typically found in the literature.

B. Point Kinetics Equations

The derivation of the PK equations usually involves the following assumptions:

1. Neutron energies belong to a single energy group.
2. Fundamental mode in space has been achieved thereby allowing separation of space and time.
3. The reactor core is bare (ie, no reflectors).
4. Delayed neutron precursors are grouped into predefined groups.
5. Delayed neutron precursors do not diffuse before they decay.

Under these assumptions the point kinetics equations are easily derived from the neutron diffusion equation balance statement.

$$\frac{dn}{dt} = \left[\frac{\rho - \beta}{\Lambda} \right] n(t) - \sum_{i=1}^I \lambda_i C_i(t) \quad (2.1a)$$

Table I

Definition of Commonly Used Reactor Kinetics
Parameters and the RK Transfer Probabilities

n	Neutron density	I	Number of delayed neutron precursors	N	Number of reflectors
C _i	Precursor density	β _i	Delayed neutron fraction	λ _i	Precursor decay constant

K	Infinite multiplication constant
K _{eff}	Effective multiplication constant
ρ	Reactivity = (K _{eff} - 1.0) / K _{eff}
l	Lifetime of Neutrons (sec)
Λ	Mean generation time (sec) = l/K _{eff}
P _{ML}	System non leakage Probability
P _c	Core loss transfer probability
P _{rr}	Transfer probability from core to reflector r.
P _r	Transfer probability from reflector r to core.

$$\frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} n(t) - \lambda_i C_i(t) \quad (2.1b)$$

The parameters in the above equation and other symbols and notation used later are given in Table I.

Hetrick³ derives the PK equations starting from the general neutron transport equation and expresses them in more general terms; however, such rigor is usually unnecessary.

For LWR applications, the PK equations are both convenient to work with and sufficiently accurate. While the accuracy of the PK method may be sufficient for the treatment of space nuclear cores, the differences between LWR's and space reactor cores warrant some special considerations. First, the presence of the reflecting control drums removes the validity of the bare reactor assumption. Secondly, the transients that are expected introduce new difficulties in determining the K_{eff} (or

ρ) resulting from certain control input. Additionally, if the control drum reactivity worth is lumped into an overall reactivity term as in the PK equations, study of the feasibility of the integration of automatic control techniques would be difficult. On the other hand, the fundamental mode assumption in the PK equations is also applicable to space reactors since such compact reactors with intermediate or fast neutron energy spectra exhibit the more rapid transmission of information through the system. This results in a relaxation of the fundamental mode assumption as can be seen upon examination of the magnitude of the second harmonic term to the first in the derivation of the PK equations from the time dependent neutron diffusion equation.

C. Reflected Kinetics Model

The RK model is developed as from the main structure of the point kinetics equations. The primary modification is done in order to accommodate the independent reactivity worths of the control drums that are neutronically coupled to the core. First it is necessary to look at the PK equations from the perspective of neutron flow as shown in Fig. 1. From Fig. 1 an alternate form of the PK equations can be written down immediately by balancing neutrons within a particular generation. First we define the neutron production term, Q , as

$$Q \equiv \frac{(1 - \beta)K_{\infty}n_c(t)}{\ell_{\infty}} - \sum_{i=1}^I \lambda_i C_i. \quad (2.2)$$

The PK equations are then written as,

$$\frac{dn_c}{dt} = P_c Q - \frac{n_c(t)}{\ell_c} \quad (2.3a)$$

$$\frac{dC_i}{dt} = \frac{\beta_i K_{eff}}{\ell} n(t) - \lambda_i C_i(t). \quad (2.3b)$$

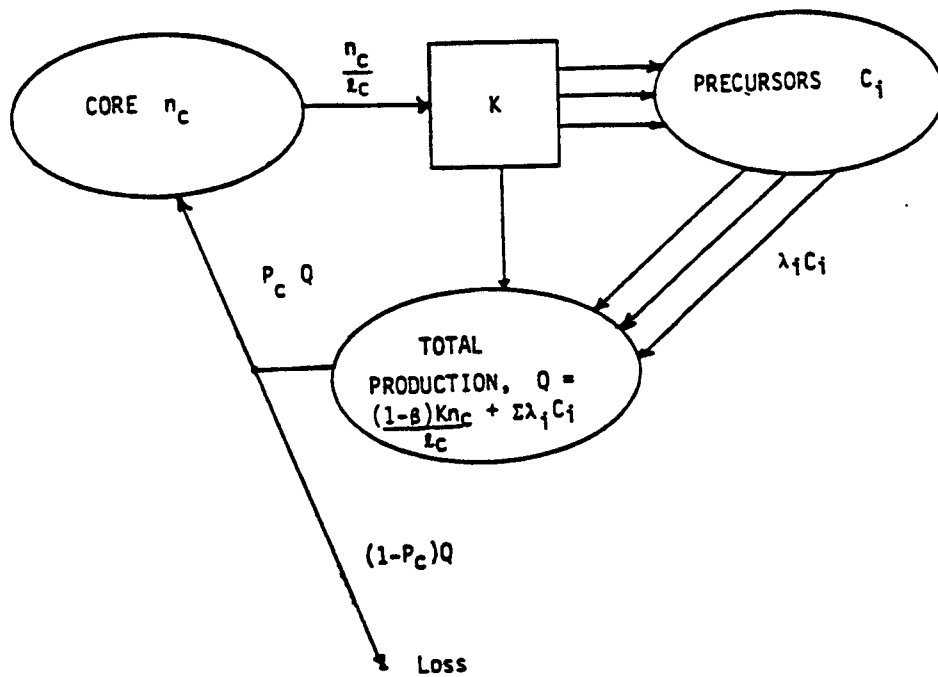


Fig. 1 Bare Core Neutron Flow Diagram

This form of the kinetics equation is particular well suited for extension to the reflected case. Also note that given the proper definition of P_c , equation (2.3) is identical to equation (2.1). The P_c definition is discussed following the development of the RK equations.

The flow diagram shown in Fig. 1 can easily be modified to include the effect of multiple reflectors as shown in Fig. 2. Given that the transfer probability terms (P_c , P_{Fr} , and P_r) are defined as shown in Table 1. the neutron balance statements of the reflected system are given by,

$$\frac{dn_c}{dt} = P_c Q - \frac{n_c(t)}{\ell_c} - \sum_{r=1}^N \frac{P_r n_r(t)}{\ell_r}, \quad (2.4a)$$

$$\frac{dn_r}{dt} = P_{Fr}(1 - P_c)Q - \frac{n_r(t)}{\ell_r}. \quad (2.4b)$$

$$\frac{dC_i}{dt} = \frac{\beta_i K_{\infty}}{\ell_c} n_c(t) - \lambda_i C_i(t). \quad (2.4c)$$

where Q is defined by equation 2.2. These expressions represent the balance of neutrons in the core, the balance of neutrons in the reflectors, and the balance of delayed neutron precursors in the core respectively. Equation 2.4 is the foundation for the RK equations. The similarity between these equations and the point kinetics equations are obvious. This is mainly due to the preservation of the fundamental mode and separation of space and time assumption in both models. Several important differing features are now noted. First, these equations are not driven by a K_{eff} or ρ term representing some lumped driving reactivity input. Second, the RK equations are dependent on the definition of the transfer probability terms. Third, given that P_c , P_{Fr} , P_r , K_{∞} , n_c , and n_r are not constant, the RK basic equations are nonlinear time-variant in nature. Finally, control drum reactivity worth can be treated more explicitly than in the PK equations by direct correlation

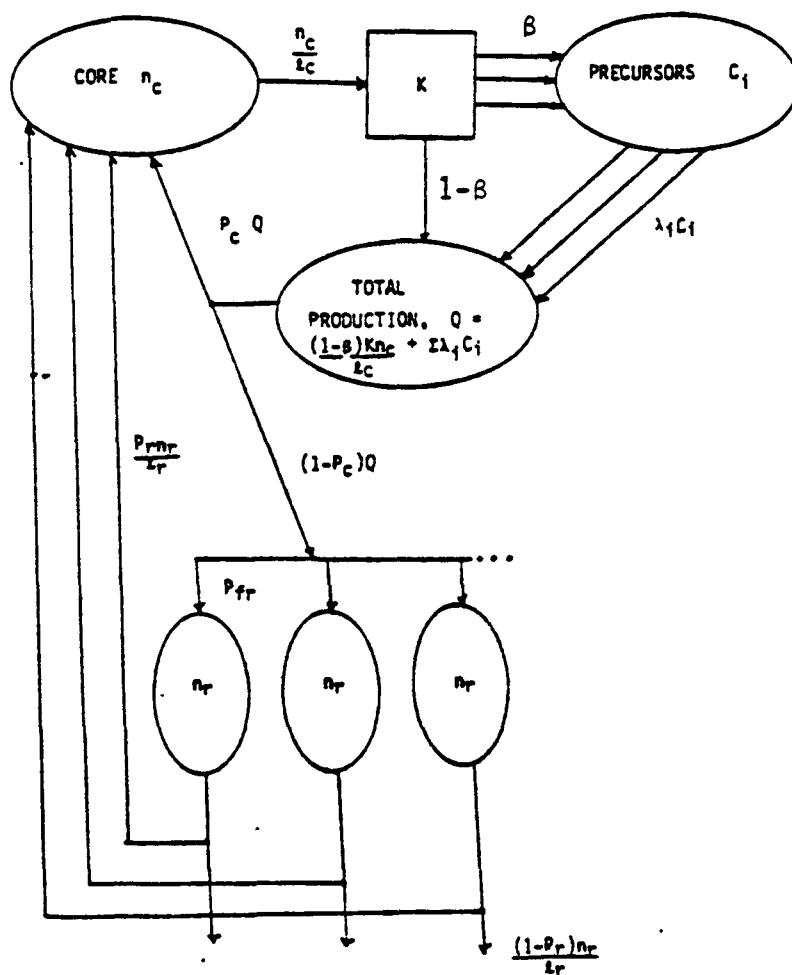


Fig. 2 Reflected Reactor Neutron Flow Diagram

between physical system configuration and the transfer probabilities. The additional equations and non-constant transfer probabilities introduced by the modelling of the reflectors in the RK equations result in analytic solutions that are more difficult or impossible to obtain even for step P_r , K_∞ , or P_c inputs.

In order for the RK equations to assume any useful meaning, the transfer probabilities must be defined. This will be accomplished in the following section. Some simple ways to calculate the numerical values are also suggested.

D. Definition of Transfer Probabilities

The P_c transfer probability is defined as the probability that a fission or precursor neutron will remain in the core until the next generation. There are two subtle differences between P_c and the non-leakage probability (P_{NL}). The first pertains to a bare system in that neutrons that leak from the core in future generations do not contribute to P_c ; however, they are included in P_{NL} . The second subtle difference is that in a reflected system, P_c only refers to loss of neutrons from the core as opposed to the system as a whole. The advantage of using P_c as opposed to a P_{NL} term is the ease at which it allows us to extend the PK equations to the RK equations. In order to clarify the relationship between these two probabilities, consider the following definitions:

$$\begin{aligned} K_{eff} &= K_\infty P_{NL} \\ \ell &= \ell_\infty P_{NL} \end{aligned} \tag{2.5}$$

It can be shown that equation 2.1a becomes,

$$\frac{dn}{dt} = Q - \frac{n(t)}{\ell_\infty P_{NL}} \tag{2.6}$$

If this is equated to the dn/dt expression in equation 2.3, then the following simple relationship is obtained after a little algebra.

$$P_c = P_{NL} + (1 - P_{NL}) \frac{1}{Q} \frac{dn}{dt} \quad (2.7)$$

While this expression has little if any computational value, it does provide valuable insight as to the physical meaning and magnitude of P_c relative to P_{NL} . Since Q is always positive, the relative magnitude of P_c with respect to P_{NL} is determined by the sign of dn/dt . More importantly, at or near steady state, the difference between the two are negligible and for all practical purposes can be taken to be the same. Naturally, this is only true for a bare system. When the transition is made to reflected systems, more elaborate methods of finding P_c will have to be used such as experimental means or Monte Carlo calculations. In section F, a relationship will be derived between reactivity (and K_{eff}) and the transfer probabilities, thereby defining the non-leakage probability for a reflected system in terms of the transfer probabilities. Such a relationship would be needed to find P_c by experimental or calculational means.

The P_{Fr} transfer probability is defined as the relative probability that a neutron lost from the core region will appear in any given reflector initially. This term can be viewed as an importance factor based upon the geometric configuration of the reflectors surrounding the core. For example, a symmetric core with identical reflectors totally surrounding the core at equal intervals results in P_{Fr} values given by,

$$P_{Fr} = \frac{1.0}{N_{ref}} \quad (2.8)$$

Of course such an ideal situation will rarely if ever exist. Furthermore, non-isotropic flux conditions and leakage losses to free space will have an effect on the P_{Fr}

distribution.

Perhaps a choice of P_{F_r} that has more physical justification would be the relative ratio of the total neutron current into each of the reflectors during typical steady state operating conditions. The surface area used in this calculation would be the projected area of the surrounding reflector onto the surface of the core. P_{F_r} would then be calculated by the following equation assuming the steady state flux (or current) was known.

$$P_{F_r} = \frac{\int_{A_r} j \cdot ndA}{\int_{A_{core}} j \cdot ndA} \quad (2.9)$$

The equation above by no means provides a rigorous method of calculating exact P_{F_r} values; however, it does provide a good starting point if one needs a value for doing scoping calculations of the nature desired here. If more rigor is desired, Monte Carlo techniques are suggested. All of the computational results presented in this work use the data from the University of Missouri Research Reactor (MURR) that was experimentally determined.³² This system is a doubly reflected system and is discussed in greater detail in chapter III.

Perhaps the most significant of the transfer probabilities from the point of view of control are the P_r transfer probabilities. P_r is defined as the probability that a neutron lost from a given reflector region will reflect back to the core. Clearly this is highly dependent upon the reflecting and absorbing properties of the portion of the reflector that faces the core. This can also account for (n,2n) reactions that are common in Beryllium (Be) reflectors. A typical design configuration is a reflector consisting of Be on one side of the drum and Boron Carbide (B_4C) on the other side. This offers a means of changing the power level since Be is a good neutron reflector and B_4C is a good neutron absorber. In such a mode of control, the driving reactivity input as modelled by the RK equations is through changes in P_r . For

most cases of interest, it is assumed that P_{F_r} remains unaltered by control drum motion. It is also assumed that ℓ_r is constant. All changes in reflector neutron loss resulting from changes in ℓ_r are assumed to be accounted for in the P_r probability term.

In light of the above discussion, the RK model can be used in analyzing transients by direct change of the transfer probability, P_r . A potential advantage exists in this approach since such parameter changes correspond more directly to control drum motion than changes in K_{eff} or ρ . Reactivity is commonly related to control rod position or material property changes by first order perturbation techniques. Such perturbation calculations with regard to individual control drum position introduce additional room for error particularly for very rapid transients. The RK approach removes this indirect link in the modelling process. Furthermore, the P_r approach provides a more appropriate environment for the use of an automatic controller that will dictate the required reflector control motion. Nominal values of P_r can be found by experimentally measuring K_{eff} for the initial control drum position. This requires the development of a relationship relating the transfer probabilities to K_{eff} . This relationship turns out to be closely related to the critical condition that is developed later in this chapter. Now that the transfer probabilities have taken on more physical meaning, the RK equations can be put into usable form by linearizing them about a steady state operating point.

E. Linearization of the RK Equations

There are several practical reasons for linearizing the RK equation set developed in the earlier part of this chapter. The most fundamental reason for linearization is our desire to design a stabilizing feedback controller that will allow robust

tracking of step and ramp reference signals. Unfortunately, design methods for nonlinear systems are either non-existent or are limited to a small range of special problems that fall into certain well known categories. Also, linearization transforms the model equations into a more advantageous form from the point of view of calculating solution trajectories. In particular, linearization allows use of the exponential operator method that has proven to be more time efficient than Runge Kutta and other finite difference integration techniques. The penalty for linearization is that it imposes a limit on the types of transients that can be analyzed without violating the linearization assumptions. In chapter III, the effects of linearization are demonstrated for a sample problem.

The following definitions are utilized in the linearization process:

$$\begin{aligned} n_c &= n_{c0} + \delta n_c, & n_r &= n_{r0} + \delta n_r, & C_i &= C_{i0} + \delta C_i, \\ K &= K_0 + \delta K, & P_c &= P_{c0} + \delta P_c, & P_r &= P_{r0} + \delta P_r. \end{aligned} \quad (2.10)$$

For notational simplicity, K_∞ is also denoted simply by K . Note that the parameters P_{Fr} , ℓ_r , ℓ_c , and β_i will be taken to be constant. While this is not strictly correct over long periods of time, it is an adequate assumption within the time frame of a typical transient. Linearization is done by first inserting equation 2.10 into the nonlinear RK equations given by equation 2.4. All terms involving the product of two differentials or more are then considered to be negligible with respect to the other terms. The details of the linearization are included since the intermediate steps are used in the derivation of the critical condition in the next section.

First consider linearization of the Q source term. Equation 2.2 is rewritten in terms of the differentials as,

$$Q = (1 - \beta)(K_0 + \delta K) \frac{n_{c0} + \delta n_c}{\ell_c} + \sum_{i=1}^I \lambda_i (C_{i0} + \delta C_i) \quad (2.11)$$

As stated, if we neglect the $\delta K \delta n_c$ term, then Q becomes $Q = Q_0 + \delta Q$, where.

$$Q_0 = \frac{K_0 n_{c0}}{\ell_c}. \quad (2.12a)$$

$$\delta Q = (1 - \beta) \frac{(K_0 \delta n_c - n_{c0} \delta K)}{\ell_c} - \sum_{i=1}^I \lambda_i \delta C_i. \quad (2.12b)$$

In a similar fashion the following relationships are easily derived:

$$P_c Q \simeq P_{c0} Q_0 + (P_{c0} \delta Q + \delta P_c Q_0), \quad (2.13a)$$

$$P_r Q \simeq P_{r0} n_{r0} + (P_{r0} \delta n_r + \delta P_r n_{r0}). \quad (2.13b)$$

$$K_\infty n_c \simeq K_0 n_{c0} - (n_{c0} \delta K + \delta n_c K_0). \quad (2.13c)$$

Before the transient begins, the condition of the system neutronics is given by equation 2.4 with the time derivatives set to zero. This information stipulates the relationship that n_c , n_r , and C_i must have during a steady state (or critical) plant condition. It is easily shown that,

$$n_{r0} = \ell_r P_{Fr} (1 - P_{c0}) Q_0, \quad (2.14a)$$

$$C_{i0} = \frac{\beta_i K_0 n_{c0}}{\ell_c \lambda_i}, \quad (2.14b)$$

The linearization is completed by inserting the equation 2.13 and 2.14 into equation 2.4. After a little algebraic manipulation, the linearized RK equations are found to be,

$$\dot{\delta n}_c = P_{c0} \delta Q + K_0 \frac{n_{c0}}{\ell_c} \delta P_c - \frac{\delta n_c}{\ell_c} - \sum_r \frac{P_{r0} \delta n_r + n_{r0} \delta P_r}{\ell_r}, \quad (2.15a)$$

$$\dot{\delta n}_r = P_{Fr} (1 - P_{c0}) \delta Q - P_{Fr} K_0 \frac{n_{c0}}{\ell_c} \delta P_c - \frac{\delta n_r}{\ell_r}, \quad (2.15b)$$

$$\dot{\delta C}_i = \beta_i \frac{n_{c0}}{\ell_c} \delta K + \beta_i K_0 \frac{\delta n_c}{\ell_c} - \lambda_i \delta C_i. \quad (2.15c)$$

The above kinetics equations are linear, time-invariant descriptions of the space reactor power plant. These equations can be written in matrix format as,

$$\begin{aligned}\dot{x} &= Ax - Bu, \\ x &= [\delta n_c \quad \delta n_r \quad \delta C_i]^T, \\ u &= [\delta K \quad \delta P_c \quad \delta P_r]^T.\end{aligned}\tag{2.16}$$

Detailed expressions of the elements of the A and B matrices are given in Appendix A. These matrices provide the core of the computer codes developed throughout this work for the analysis reflected systems with the RK model.

The conditions for criticality are developed in the following section. This will provide a useful relationship between the transfer probabilities and the core reactivity, ρ . As mentioned before, this is required in order to explain the experimental measurement of the P_r transfer probability. It is also required for the comparison of the RK model to the PK solution as done in chapter III.

F. Criticality Conditions

When the core is in a critical state, the neutron and delayed neutron precursor densities are constant in time. This condition is mathematically represented by zero time derivatives in the RK equations. Equation 2.4a is set to zero and the relationships given by equation 2.14 are inserted so that only the n_{c0}/ℓ_c terms appear. It can be shown that this simplifies to,

$$K_\infty \left[P_c + (1 - P_c) \sum_r P_{Fr} P_r \right] = 1.\tag{2.17}$$

This expression is called the "critical condition" since when the above condition exists the reactor is said to be critical. This critical condition stipulates what

the relationship between the transfer probabilities and the infinite multiplication constant (K_∞) must be in such a critical system. The importance of this statement is that it defines the value of the infinite multiplication constant, given the initial values of the transfer probabilities in an initially critical system. In all of the benchmark and actual computations, this relationship is used in order to determine the nominal value of K_∞ . Since criticality is equivalently defined by $K_{eff} = 1$, then by definition the steady state non-leakage probability is given by.

$$P_{NL} = P_c - (1 - P_c) \sum_r P_{Fr} P_r. \quad (2.18)$$

Note that the non-leakage probability is greater for a reflected system than a bare one as expected due to the neutron "reflection". It now is assumed that this equation is also valid for non steady state conditions. Further insight can be found upon linearization of equation 2.18. Linearization procedures are applied as usual to give,

$$P_{NL0} = P_{c0} + (1 - P_{c0}) \sum_r P_{Fr} P_{r0}. \quad (2.19a)$$

$$\delta P_{NL} = \delta P_c \left[1 - \sum_r P_{Fr} P_{r0} \right] - (1 - P_{c0}) \sum_r P_{Fr} \delta P_r. \quad (2.19b)$$

The effective multiplication constant, K_{eff} , can therefore be expressed in terms of the linearized non leakage probability as follows,

$$K_{eff} \simeq K_0 P_{NL0} - K_0 \delta P_{NL} + P_{NL0} \delta K, \quad (2.20)$$

If we now assume that we are near critical, the reactivity change is given by the change in K_{eff} . Equation 2.19 is substituted into equation 2.20 above and the

reactivity change in terms of the transfer probabilities becomes.

$$\delta\rho = K_0\delta P_c \left[1 - \sum_r P_{Fr}P_{r0} \right] - K_0(1 - P_{c0}) \sum_r P_{Fr}\delta P_r - P_{c0}\delta K - (1 - P_{c0})\delta K \sum_r P_{Fr}P_{r0}. \quad (2.21)$$

This expression is essential in order to test the validity of the RK equations by calculating the equivalent reactivity for a given transfer probability input set. Comparisons with an analytic PK solution which requires a numerical value for reactivity can therefore be done in order to validate the equations for simple cases.

G. Summary

In this chapter we have developed the basic foundation of the Reflected Kinetics Model. This was accomplished by first observing the paths of neutron flow under the point kinetics philosophy. Qualitative discussions of the physical meaning of the neutron transfer probabilities and potential ways for determining their values were discussed. The nonlinear RK equations were written down from first principles of neutron balance using a neutron flow diagram of a reflected system as a guide. Since linear equations are much more attractive from a design and analysis standpoint, the RK equations were then linearized. Finally, some useful relationships involving criticality conditions were then developed for use in the next chapter which will deal with validation of the preliminary RK equations.

CHAPTER III

PRELIMINARY VALIDATION OF THE RK MODEL

A. Introduction

The main objective of this chapter is to benchmark the RK equations by comparing their numerical solution obtained to both an analytic solution and the more conventional point kinetics (PK) method. The first benchmark is a comparison of the linearized RK solution trajectory to the analytic solution for a simplified case. This analytic solution is developed by method of Laplace transforms. The poles of the analytic solution are also compared to the eigenvalues of the numerical RK matrix. Secondly, the equivalent bare system is identified for the same problem so that the point kinetics equations can be solved for comparison. These comparisons are not limited to one precursor group since the analytic PK solution is readily found for a general number of delayed neutron precursor groups. The equivalent bare core reduction involves the use of the equivalent reactivity expressions developed in chapter II. Finally, an analogy is made between the RK and PK transfer functions. This analogy will illustrate why reflectors are often treated as additional fictitious delayed neutron precursor groups in the point reactor kinetics model.

In order to benchmark the RK equations, reliable numerical tools are needed for the computation of the transient responses. The magnitude of the eigenvalues of the open loop system make it computationally unfeasible to use discrete time integration schemes such as Runge Kutta in solving the RK equations. Therefore, all RK transient calculations are performed using the ASH exponential operator method. The ASH method is discussed as well as an extension that allows problems with more general source terms to be solved. The analytic RK solution and the

analytic PK solution are first discussed.

B. Analytic Solutions to the RK and PK Equations

An analytic solution is developed for a simplified version of the nonlinearized RK equations. The method used is that of Laplace transforms. Even though we are dealing with the nonlinear RK equations, the following assumptions make these equations linear without introducing the typical linearization assumptions. The simplifications in addition to those made in chapter II are.

- One delayed neutron group
- One reflector with constant P_{F_r} and P_r .
- The initially critical core is perturbed only by a change in K

Under these assumptions equation 2.4 reduces to three coupled linear expressions. It is important to note that the linearization assumptions used in chapter II are not applied in the analytic solution. These three coupled equations are solved for the neutron concentration only. In the Laplace transform space the coupled transient equations are reduced to coupled algebraic ones. If s is defined as the Laplace transform variable then a little algebraic manipulation reveals that the laplace transform of the core density, $N_c(s)$, is given by.

$$\frac{n_c(s)}{n_c(0)} = \frac{N(s)}{M(s)}, \quad (3.1)$$

where,

$$N(s) = \lambda_c K_0 [(s + \lambda_r) P_{F_r} P_r (1 - P_c) - \beta (s + \lambda_r) P_c + \beta \lambda_r P_{F_r} P_r (1 - P_c)] + (s + \lambda)(s - \lambda_r), \quad (3.2a)$$

$$M(s) = (s + \lambda)(s + \lambda_c)(s + \lambda_r) - \lambda_c K (s + \lambda - \beta s) [(s + \lambda_r) P_c + \lambda_r P_{F_r} P_r (1 - P_c)]. \quad (3.2b)$$

The inversion of this transform is done by applying the complex inversion formula. This merely says that the inverse transform is equal to the sum of the residues of the function $n_c(s)\exp(st)$ at its singularities. For the three simple poles in this case the residues are found by differentiating the denominator term with respect to s . The solution is therefore given by.

$$n_c(t) = n_c(0) \sum_{s=p} \frac{N(s)}{M'(s)} e^{st}. \quad (3.3)$$

where p are the roots of $M(s) = 0$. The derivative of the denominator is readily found to be,

$$\begin{aligned} M'(s) = & (s + \lambda)(s - \lambda_c) - (s + \lambda_c)(s - \lambda_r) - (s + \lambda)(s - \lambda_r) \\ & - (1 - \beta)\lambda_c K'(s + \lambda_r)P_c + \lambda_r P_{Fr}P_r(1 - P_c) - \lambda_c K(s - \lambda - s\beta)P_c. \end{aligned} \quad (3.4)$$

The poles are numerically calculated by the commonly used modified Regula Falsa root finding technique and the solution is constructed by summing the contributions at each of the three poles given by equation 3.3.

The step reactivity point kinetics problem can be solved analytically in a similar fashion. The easiest approach is again by method of Laplace transform. The solution is constructed for the step reactivity case and for a general number of precursor groups, I , given by equation 2.1. The core neutron concentration is found to be given by.

$$\frac{n_c(t)}{n_c(0)} = \frac{\rho}{\Lambda} \sum_{n=1}^{I+1} \frac{e^{s_n t}}{s_n \left[1 + \sum_{i=1}^I \frac{\lambda_i \beta_i / \Lambda}{(s_n + \lambda_i)^2} \right]} \quad (3.5)$$

where s_n are the $(I + 1)$ roots of the equation,

$$\frac{\rho}{\Lambda} = s_n \left[1 - \sum_{i=1}^I \frac{\beta_i / \Lambda}{(s_n - \lambda_i)} \right]. \quad (3.6)$$

Again, the roots are found numerically and the solution is constructed by summing the contribution of each of the residues at the poles. For the 6 group case, there are seven simple poles that satisfy equation 3.6.

C. ASH Exponential Operator Methods

Recall that the RK equations developed in chapter II as well as most problems of interest in the control field are written in the general matrix form.

$$\dot{X} = AX + S, \quad S = BU. \quad (3.7)$$

The numerical solution of the RK open loop equations is therefore equivalent to the solution to this matrix equation. Conventional solution techniques are based on discrete integration techniques which fail if the equations are stiff. Also, if late time solutions are desired, then excessive computation time due to small time steps may result. The ASH approach is an operator method that solves equations of this type essentially analytic in time. The computational advantages that ASH offer over discrete integration techniques in solving hyperbolic and parabolic partial differential equations reduced to the form of equation 3.7 are discussed by Lee and Washington.³³

It is assumed that A and S are constant over the time interval $[0, t]$. This limitation will be avoided for the source; however, the restriction on the A matrix to be time invariant must remain in order for the ASH method to work. Assuming that matrix exponentiation is well defined and obtainable, then the solution to equation 3.7 is given by the analytic expression,

$$X(t) = e^{At}X(0) - A^{-1}(e^{At} - I)BU. \quad (3.8)$$

The ASH method consists primarily of a computationally efficient and accurate way of expressing the matrix exponential terms in the above expression. This is basically done by scaling the At matrix down by powers of two until a Taylor series for e^{At} , truncated to a specified number of terms for a desired accuracy, converges with minimum roundoff error. A recurrence relation is then used to rescale the solution. Details of the scaling laws and rescaling recurrence relationships are given in Appendix B.

In the past, one of the primary limitations of the ASH solution technique has been that the scaling laws and recursion relations are valid only for constant source terms. Even though Lee²⁹ developed recursion relations for linear source terms, the complexity of the algorithms indicate that extensions to more elaborate sources such as higher order polynomials, exponentials, or trigonometric forms would be nearly impossible to obtain. The simplicity of the Extended Source ASH (ESASH) method presented below is that it removes the constant source limitation of ASH while still avoiding the use of complex recursion relations.

The idea behind the ESASH approach is to include time states which correspond to the functional forms desired in the source vector. These states are then coupled to the other states in the system thereby leaving the source input vector a constant. For example, consider the equation,

$$\dot{t}_1 = 1, \quad t_1(0) = 0 \quad (3.9)$$

The solution to this simple equation states that $t_1 = t$. Therefore, if this is added to the state equations and the initial condition is set to zero, then linear source terms can be treated by coupling this state to the equations that have linear source input terms. This idea can be extended to higher order polynomials such as t^2 where we

merely need to include the transient equation.

$$\dot{t}_2 = 2t_1. \quad t_2(0) = 0 \quad (3.10)$$

In this case, since t_1 is equal to time, then t_2 will become equal to the square of time. The extension to higher order polynomials is obvious. This extension is not limited to polynomial sources. Suppose one wishes to use ASH to solve a problem with an exponential source term. Using the same idea, we write a time derivative which when solved will give us an exponential in time, which can therefore be included as an input. The pertinent equations that produce this exponential source are given by.

$$\dot{t}_e = \alpha t_e. \quad t_e(0) = 1.0, \quad t_e = e^{\alpha t} \quad (3.11)$$

Trigonometric functions can be represented by the two simple equations:

$$\begin{aligned} \dot{t}_c &= \omega t_s, \quad t_c(0) = 1. \quad t_c = \cos(\omega t) \\ \dot{t}_s &= -\omega t_c, \quad t_s(0) = 0. \quad t_s = \sin(\omega t) \end{aligned} \quad (3.12)$$

The success of the ESASH method is dependent upon the accurate calculation of the time states. Since ASH automatically ensures that the cumulative error is less than some user input tolerance (see Appendix B), then this should not be a problem. The additional computational time required to solve the resulting larger system is little sacrifice compared to the development and application of more complex recursion relations if they could even be found. The penalty for this approach is the additional memory requirements needed to accommodate the enlarged equation set. While the idea of expressing high order equations as a collection of many low order ones is certainly not a new one, the application of this simple idea to the ASH methodology will lead to the more accurate solution of a wider range of transient

problems. In particular, this modification has enabled the use of ASH in solving the ramp reactivity input and ramp closed loop RK servo problem. The improved accuracy that ASH offers over finite difference methods, and the ability of ASH to solve systems with large negative eigenvalues with minimal computational effort make it the method of choice in solving the open loop RK equations.

Three computer codes were written which evaluate the open loop RK equations, the analytic simplified RK solution, and the analytic bare reactor PK solution respectively. The first program is a direct application of the ASH methodology to the RK equation set in the FORTRAN language. This code basically develops the A and S matrices for a selected set of system parameters and input conditions. The second program interactively solves for each of the three poles in the analytic RK solution. The contribution of each of the three residues in the inverse Laplace transform are then numerically added to construct the solution. The third program is an implementation of the step reactivity PK solution. It was written so that over many time steps the equations are solved analytically for different user chosen step reactivities. This allows for the approximation of reactivity inputs that are more complex than steps over the duration of a transient. These programs are used in the comparisons discussed in the next section.

D. Numerical RK, Analytic RK, and Analytic PK Comparisons

The benchmark runs that are made involve the data given in Table II. This data is the result of experiments performed at the University of Missouri Research Reactor (MURR) facility. The reasoning behind the use of this reactor's data is discussed in chapter VI. The value of K listed in Table II was determined from the critical condition as discussed in chapter II. Note that the data in Table II may not

Table II
Available Neutronic Data for the
Doubly Reflected MURR Facility

<u>Core</u>	<u>Reflector #1</u>	<u>Reflector #2</u>
ℓ_c 5.7×10^{-4} sec	ℓ_r 3.22×10^{-4} sec	2×10^{-3} sec
P_c 0.34695	P_r 0.2550	0.41981
β 0.00738	P_{Fr} 0.7349	0.26506
λ 0.7738 sec^{-1}		
K 1.8450		

be typical for a space reactor, it merely provides a set of input parameters that describe the conditions of a real reactor. In each of the benchmarks, the input perturbation is equivalent to a 0.001 reactivity insertion.

The objective is to construct two equivalent systems for the doubly reflected system described by the data in Table II. The first simplification that is considered is a singly reflected case so that the analytic RK solution can be applied. This single reflector reduction will also be important in chapter V since one of the controllers discussed will only be applicable to singly reflected systems as a result of a mathematical restriction. The equivalent reflector lifetime is given by a weighted sum of the lifetimes of the individual reflectors. The destination transfer probability,

P_{Fr} is also given by the sum of the individual ones. The core return probability, P_r , is a weighted average on the destination transfer probability of the individual return probabilities, and is expressed mathematically as.

$$P_{r(eff)} = \frac{\sum_r P_{Fr} P_r}{\sum_r P_{Fr}} \quad (3.13)$$

With these definitions, the critical reflected system and the equivalent critical singly reflected system have the same infinite multiplication constant. Next, the equivalent bare core configuration is found so that the classical point kinetics equations are applicable. The mean generation time (Λ) in equation 3.5 is calculated as usual as the bare core lifetime divided by K . The equivalent bare core lifetime is given by the sum of the lifetime of the neutrons that remain in the core and the lifetimes of the reflector neutrons that return to the core. Mathematically this can be written as,

$$\ell_c(ef) = P_c \ell_c - (1 - P_c) \sum_r P_{Fr} P_r \ell_r. \quad (3.14)$$

Recall that equation 2.21 was derived for the specific purpose of the calculation of the equivalent reactivity. The PK comparisons are done with six delayed neutron precursor groups. The delayed neutron fractions and decay constants are those of the fast fission of Uranium 235 (U235) taken from Keepin.³⁴ The delayed neutron fractions and decay constants have been weighted so that they correspond to the same total delayed neutron fraction and effective decay constant respectively. The singly reflected and bare equivalent system parameters are given in Table III.

A comparison of the linearized and analytic RK core densities is shown in Fig. 3. The input in this case was in the form of a step as evidenced by the predictable prompt jump shown in Fig. 3. The linearized RK calculation with 6 delayed neutron groups is compared to the PK analytic solution in Fig. 4. The

Table III

Equivalent Singly Reflected and Bare
System Parameters for the MURR Data Set

<u>Singly Reflected</u>	<u>Bare</u>
λ_c 5.7×10^{-4} sec	Λ 7.5474×10^{-4} sec
P_c 0.34695	δp 0.000542
λ_r 2.83×10^{-4} sec	
P_{Fr} 1.0000	
P_r 0.29868	
δP_r 0.00083	

0.001 reactivity insertion in this case was done over a period of 32 seconds in the form of a steady ramp.

At early times the validity of the linearization assumptions are evident in both Fig. 3 and Fig. 4. However, violation of the linearization assumptions eventually gives rise to unacceptable errors in the results. This is evident at late times in both Fig. 3 and Fig. 4. where the true response diverges from the linear approximation. When the change in the densities have increased more than about 10 percent, the neglect of the second order differential terms is a poor assumption.

Another observation made from Fig. 4 is that the linear model gives responds symmetrically to equal magnitude, opposite sign reactivity input; however, the

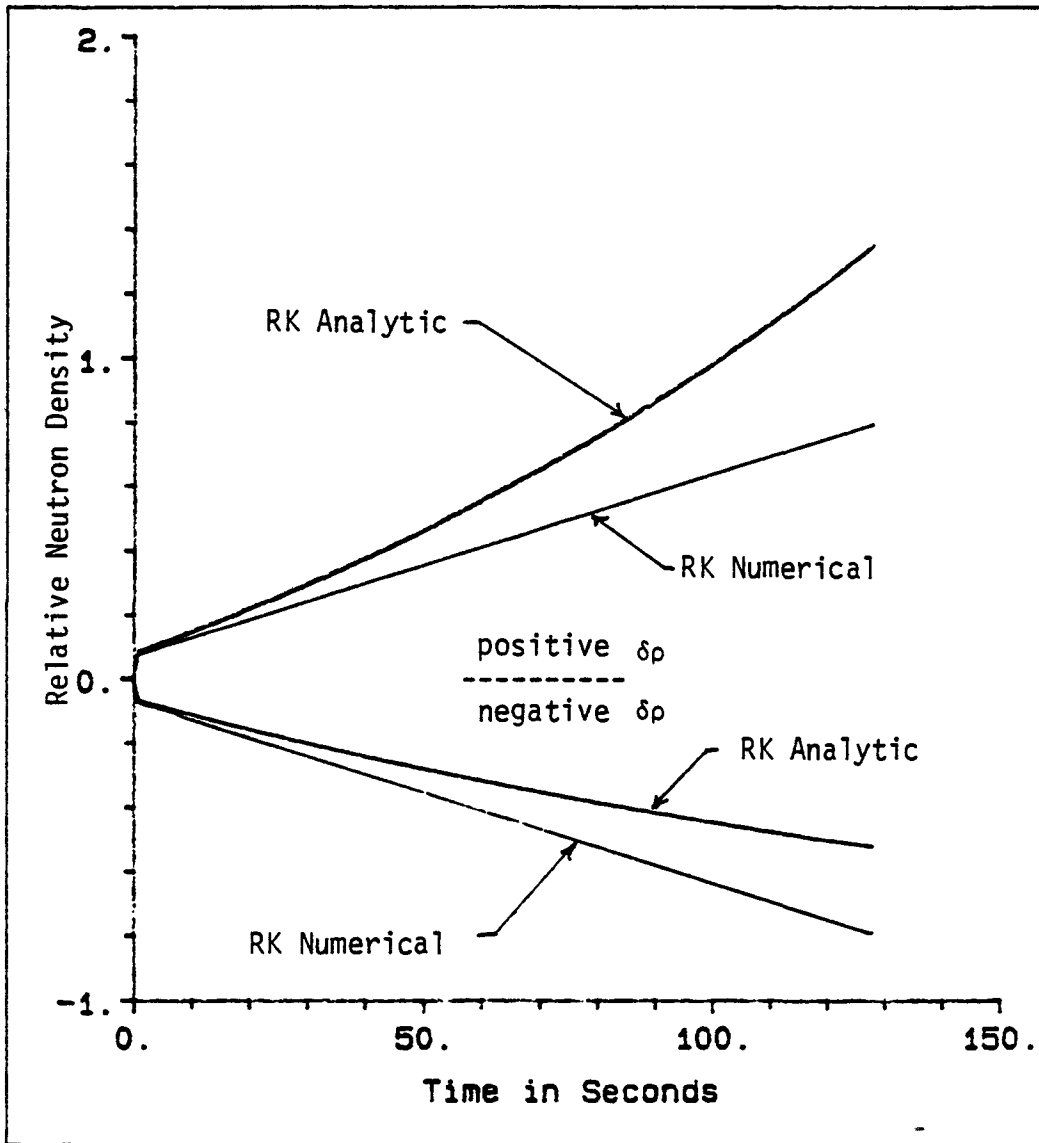


Fig. 3 Comparison of Analytic and Numerical Solutions to the RK Equations

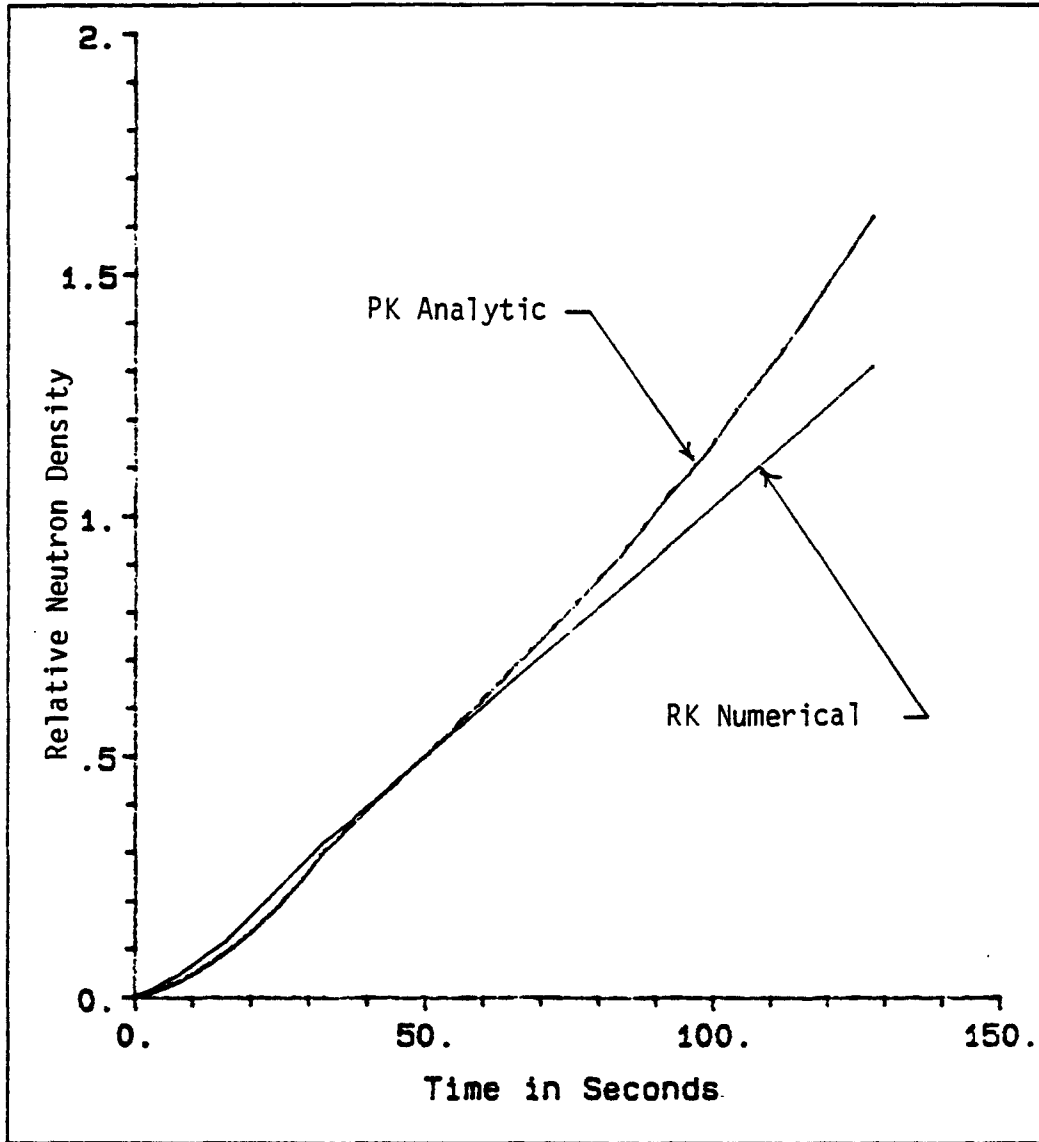


Fig. 4 Comparison of Analytic PK and the Numerical Solution to the RK Equations

Table IV
 Comparison of the RK Open Loop Eigenvalues
 and the Poles in the RK Analytic Solution

RK (A) Eigenvalues	RK Poles Positive ρ	RK Poles Negative ρ
1.3677E-15	6.0789E-03	-5.2639E-03
-1.0844E+01	-1.0065E+01	-1.1624E+01
-3.7345E+03	-3.7346E+03	-3.7342E+03

analytic RK response is not symmetric. This behavior is expected since the non symmetric terms were assumed to be negligible in the linearization process. Figure 4 also illustrates the success of the ESASH method in solving problems with linear source input vectors.

The poles of the analytic RK transfer function are next compared to the eigenvalues of the RK open loop state equations. Since both models represent the same phenomenon, certain information should be common between these poles and eigenvalues. As seen in Table IV, two of the poles are nearly identical to two of the eigenvalues.

The third eigenvalue, which is essentially zero for the linearized model, is explained by the fact that the linearized RK equations effectively represent a critical system with an external source of neutrons. This neutron source is provided by the reflectors and modelled as an external forcing function, δP_r . On the other

hand, the analytic RK equations are not driven by an external forcing function. Instead, they behave more like the true system which becomes either supercritical or subcritical once it is altered. This explains the departure of the two solutions at late times from a more physical point of view since a supercritical system with no feedback is unstable and the small positive pole becomes very important at late times. Temperature feedback effects are discussed in chapter IV.

E. Comparison of RK and PK Response Functions

The response function relating core neutron density and reactivity is derived from the RK equations developed in chapter II as usual by Laplace transforming the transient equations. The main difficulty that arises in the development of such a transfer function is the identification of a reactivity term from the direct probability inputs used in the RK model. This is necessary in order to make a meaningful comparison between the RK and the PK response functions.

We begin the response function development with the precursor balance given by equation 2.4c. It is easily shown that the Laplace transform of equation 2.4c reduces to,

$$\delta C_i(s) = \frac{\beta}{s + \lambda_i} \frac{K_0 \delta n_c(s) + n_{c0} \delta K(s)}{\ell_c} \quad (3.15)$$

Next the delayed neutron kernel and its Laplace transform are defined as,

$$D(t) \equiv \sum_i \frac{\beta_i}{\beta} e^{-\lambda_i t}, \quad (3.16a)$$

$$D(s) = \sum_i \frac{\beta_i / \beta}{s + \lambda_i}. \quad (3.16b)$$

The delayed neutron kernel is commonly used in point kinetics analysis and it appears in the PK response function. A little algebraic manipulation reveals the

useful relationship,

$$\sum_i \frac{\lambda_i \beta_i}{s - \lambda_i} = \beta [1 - sD(s)]. \quad (3.17)$$

The Laplace transform of the neutron production, δQ , is found to be,

$$\delta Q = \frac{K_0 \delta n_c - n_{c0} \delta K}{\ell_c} [1 - \beta sD(s)]. \quad (3.18)$$

The reflector and core neutron density transfer functions can now be written down directly in terms of δQ as,

$$\delta n_r(s - 1/\ell_r) = P_{F_r}(1 - P_{c0})\delta Q - P_{F_r}K_0 \frac{n_{c0}}{\ell_c} \delta P_r, \quad (3.19a)$$

$$\delta n_c(s + 1/\ell_c) = P_{c0}\delta Q + K_0 \frac{n_{c0}}{\ell_c} \delta P_c - \sum_r \frac{P_{r0}\delta n_r + n_{r0}\delta P_r}{\ell_r}. \quad (3.19b)$$

Finally, the δQ , n_{r0} , and δn_r terms are eliminated from equation 3.19b and the response function is written as,

$$\begin{aligned} N(s) = & P_{c0}\delta K [1 - \beta sD(s)] + K_0\delta P_c + \sum_r P_{F_r}\delta P_r(1 - P_{c0})K_0 \\ & + \frac{P_{r0}P_{F_r}}{(s\ell_r + 1)} [(1 - P_{c0})\delta K [1 - \beta sD(s)] - K_0\delta P_c] \end{aligned} \quad (3.20a)$$

$$\begin{aligned} M(s) = & \frac{s}{\beta} [\ell_c + P_{c0}K_0\beta D(s)] + \frac{1}{\beta} [(1 - P_{c0}K_0) \\ & - (1 - P_{c0})[1 - \beta sD(s)]K_0 \sum_r \frac{P_{r0}P_{F_r}}{(s\ell_r + 1)}] \end{aligned} \quad (3.20b)$$

$$\frac{\delta n_c(s)/n_{c0}}{1/\beta} = \frac{N(s)}{M(s)} \quad (3.20c)$$

In order to write the RK transfer function in the desired form the equivalent reactivity in equation 3.20 must be identified. A few assumptions regarding the magnitude of the poles are made so that the numerator, $N(s)$, term can be simplified.

$$s\ell_r + 1 \simeq 1, \quad (3.21)$$

$$1 - \beta sD(s) \simeq 1.$$

These assumptions are valid since $s\ell_r$ and $\beta sD(s)$ are typically smaller than unity for the system poles, s , that solve the expression $M(s) = 0$. This is easily verified for the benchmark case in the previous section as seen by the values in Table III and Table IV. Extremely large poles that do not satisfy these assumptions are accounted for by the fact that they make little or no contribution at all to the transient response due to the nature of the quickly diminishing $\exp(st)$ term in the inverse Laplace transform. Under these two assumptions the change in reactivity is equal to the numerator term, $\delta\rho = N(s)$. The RK response function is therefore reduced to the following form,

$$\frac{\delta n_c / n_{cl.}}{\delta\rho / \beta} = \frac{1}{M(s)}. \quad (3.22)$$

When there are no reflectors, this expression reduces to,

$$M(s) = \frac{s}{\beta} (\ell_c - \beta D(s)). \quad (3.23)$$

Equation 3.23 is also the linearized point kinetics core response function. This limiting behavior is importance for the following reason. In the modelling of a reflected system where the reflectors have little impact on the neutronics of the system, it is desirable for the solution to limit to that of a bare reactor. In other words, the reflector modelling should hold for weak and strong reflection configurations. The last two terms in the RK response function are present as a result of the modelling of the reflectors. The first extra term represents the fact that the core criticality is more than a function of the leakage from the core. This is a result of the fact that when neutrons are lost from the core in a reflected system, they may return or "reflect" thereby remaining in the neutron generation. The second extra term represents the delayed return of the neutrons

and possible creation of $(n,2n)$ neutrons from the reflectors back into the core. These last two terms could also lead to an alternative treatment of a reflected system in a point kinetics environment. If appropriate definitions are made for $\lambda(\text{reflector})$ and $\beta(\text{reflector})$ then additional fictitious delayed neutron precursor groups which represent the reflectors can be identified. These additional terms can then be absorbed into the delayed neutron kernel, $D(s)$. A realization of this transfer function would merely lead to the standard point kinetics equations with additional delayed neutron groups. However in doing so, the ability to directly model the control function of the reflectors would be lost.

CHAPTER IV

THERMAL MODELLING AND TEMPERATURE FEEDBACK EFFECTS

A. Introduction

In chapter II, the RK model was developed basically to assess the transient behavior of the core and the reflectors with no regard to thermally induced effects. The intent of this chapter is to introduce into the RK equations such phenomena by modelling the temperature change in both the core and coolant regions. The lumped parameter procedure will be applied in order to express the relationship between the thermal energy production and the core and coolant temperatures. These temperatures are needed in order to model the temperature feedback effects. The three effects to be considered are Doppler feedback, coolant temperature feedback, and thermal expansion feedback effects. As an additional result of the application of the lumped parameter method in the coolant region, the coolant velocity is extracted as an additional control variable. This plays an important role in obtaining favorable closed loop transient responses. The open loop RK matrices are then identified in order to proceed with the application of control techniques.

B. Derivation of the Lumped Parameter Equations

In order to model the transport of heat from the core, the heat conduction equation is integrated over the homogenized fuel region and the coolant channel, and appropriate volume averaged properties are identified. This approach is frequently referred to as the lumped parameter method. The heat conduction equation within

an energy producing region is written as.

$$\rho c_p \left[\frac{\partial T}{\partial t} - v \cdot \nabla T \right] = q''' - \nabla \cdot K \nabla T \quad (4.1)$$

where T is temperature, q''' is volumetric heat generation rate, ρ is material density, c_p is heat capacity, K denotes thermal conductivity of the material, and v is an advection velocity. The two regions of interest are the averaged core region and the coolant channel.

One major simplification made in the modelling of the energy transport from the fuel into the coolant is that the entire core region is lumped into a single effective volume. This volume includes fuel, structural materials, cladding, and any other non-coolant materials within the boundary of the primary cycle. This region will be referred to simply as the "core". Within the core region, the convection term ($v \cdot \nabla T$) is zero and the energy production term is calculated from the neutron flux, ϕ , directly as,

$$q''' = \gamma \Sigma_f \phi = \gamma \Sigma_f v_n n, \quad (4.2)$$

where Σ_f denotes the core effective macroscopic fission crosssection is denoted by Σ_f and γ denotes the quantity of energy produced per fission. Typical values for γ and Σ_f are 200 MeV per fission and 0.1 cm^{-1} respectively. Also note that the neutron flux is directly proportional to the neutron density with the neutron average speed, v_n , as the constant of proportionality. Finally, it is also assumed that K , c_p , and ρ are constant throughout the extent of any transient of concern. The conduction equation to be integrated term by term over the core is therefore,

$$\rho c_p \frac{\partial T}{\partial t} = \gamma \Sigma_f v_n n - K \nabla^2 T \quad (4.3)$$

The integral of the first term over the core volume is given by,

$$\rho c_p \frac{dT_f}{dt} V_{core} \quad (4.4)$$

where T_f is the core averaged temperature defined mathematically as,

$$T_f \equiv \frac{1}{V_{core}} \int_{V_{core}} T d^3\tau. \quad (4.5)$$

The coolant average temperature, T_c , and the core averaged neutron density, n_c , are defined in a similar manner. Since the fission crosssection is assumed to be independent of position, the integral of the second term over the core volume is,

$$\gamma \Sigma_f v_n n_c V_{core}. \quad (4.6)$$

Finally, the conduction term is integrated and the Gauss divergence theorem is applied resulting in the following,

$$K \int_V \nabla^2 T d^3\tau = K A_s \nabla T|_{surface}, \quad (4.7)$$

where A_s is the heat transfer surface area between the lumped core region and the coolant channel. Newton's law of cooling provides the necessary boundary condition at the surface so that the gradient term can be represented by a linear combination of the volume averaged temperatures T_f and T_c .

$$K \nabla T|_{surface} = -h_f (T_f - T_c). \quad (4.8)$$

This approach is valid assuming that the heat transfer coefficient, h_f , can be experimentally determined with respect to the difference of the two average temperatures T_f and T_c . This approach is reasonable since the average temperatures are readily measurable.

If we collect the three integrated terms in the conduction equation and divide through by the core volume, the total differential equation that we need to model the heat removal from the core becomes,

$$\rho_f c_{pf} \frac{dT_f}{dt} = \gamma \Sigma_f v_n n_c - \frac{A_s h_f}{V_f} (T_f - T_c). \quad (4.9)$$

In order to include this equation in the state space equations derived in chapter II, equation 4.9 must be written in linear terms. This is necessary so that the neutron density is consistent with the form of the linearized RK equations. The linear perturbation form of the core temperature equation is written as,

$$\delta \dot{T}_f = -\frac{1}{\tau_f} \delta T_f - \frac{1}{\tau_c} \delta T_c - \frac{\gamma \Sigma_f v_n}{\rho_f c_{pf}} n_c, \quad (4.10)$$

where the thermal constant τ_x for material "x" is defined as.

$$\tau_x \equiv \frac{\rho_x c_{px}}{A_s h_f}. \quad (4.11)$$

Note that in expressing the lumped parameter equation in this form that no linearization assumptions are made since equation 4.9 is already linear under the lumped parameter assumptions.

The process performed above is now repeated by integrating equation 4.1 over the coolant channel. In integrating the conduction equation over the coolant channel one must first realize that the convection term can no longer be ignored since the coolant consists primarily of an axial velocity component. Secondly, within the coolant volume there is practically no energy production thereby making q''' zero. Integration of the heat conduction term over the coolant volume yields analagous results as before with the exception of a sign change that signifies the direction of the flow of heat. The integrals of the other terms are also similar to the fuel case where the only difference is that the density and the heat capacity are now that of the coolant material. Finally, if we assume that the coolant velocity and the temperature gradient in the coolant region are independent of spatial position, the lumped parameter equation for the coolant region resulting from the integration of the terms in equation 4.1 is,

$$\rho_c c_{pc} \left[\frac{dT_c}{dt} + v_c \cdot \nabla T_c \right] = \frac{A_s h_f}{V_c} (T_f - T_c). \quad (4.12)$$

The c subscripts in equation 4.12 denote coolant properties. The convection term is simplified by assuming a uniform, linear temperature rise in the coolant channel. Furthermore, it is assumed that the temperature gradient is primarily in the same direction as the coolant flow. Therefore the convection term reduces to the form,

$$v_c \cdot \nabla T_c \simeq \frac{2v_c}{L}(T_c - T_i), \quad (4.13)$$

where L is the coolant channel length and T_i is the inlet coolant temperature. If we substitute equation 4.13 into equation 4.12 and again use the definition of the thermal time constant given by equation 4.11, the coolant channel lumped parameter equation becomes,

$$\dot{\delta T}_c = \frac{1}{\tau_c}(T_f - T_c) - \frac{v_c}{\tau_L v_{c0}}(T_i - T_c). \quad (4.14)$$

The axial time constant, τ_L , is defined as the average time it takes the coolant to pass through one half the channel length, and is expressed mathematically as,

$$\tau_L \equiv \frac{L}{2v_{c0}}. \quad (4.15)$$

Unlike the core lumped parameter equation, the coolant region lumped parameter expression is nonlinear. The nonlinearity is caused by the fact that the coolant velocity, v_c , is taken to be time variant and will prove to play a major role in the control process. The result of the linearization of equation 4.14 is.

$$\dot{\delta T}_c = \left(-\frac{1}{\tau_c} - \frac{1}{\tau_L} \right) \delta T_c + \frac{1}{\tau_c} \delta T_f + \frac{(T_i - T_{c0})}{\tau_L} \frac{\delta v_c}{v_{c0}}. \quad (4.16)$$

The development of equation 4.10 and equation 4.16 above provides two more states that are added to the RK state equations. This is an important step in the development of the model since these states allow us to model the temperature induced effects in a simple and direct manner that is well suited for the desired scoping studies.

C. Temperature Feedback Effects

In a thermal power reactor the temperature changes in the core and coolant regions are typically modelled with a constant temperature feedback coefficient. These coefficients are defined by the general equation.

$$\alpha_T \equiv \frac{\partial \rho}{\partial T} \quad (4.17)$$

This equation translates into the discrete form shown below which is of most use when analyzing transients within the point reactor framework,

$$\Delta \rho \simeq \alpha_T \Delta T \quad (4.18)$$

In the RK model, reactivity is represented explicitly as the infinite multiplication constant, the core loss probability, and the reflector return probabilities. This modelling approach requires a more specific means of treating the temperature changes in the core and coolant regions. In particular, since we have direct control over the leakage properties of the core through reflector position, only the infinite multiplication factor, K , and the core loss probability, P_c , will be sensitive to these temperatures. The advantage of the reactivity separation principle in the RK model is clearly evident in the treatment of temperature feedback effects. The mechanism for temperature feedback within the core region is primarily the Doppler effect. The Doppler effect is basically a broadening of the resonance region of absorption cross section of Uranium fuel material in the core region, which increases the fraction of absorptions that do not cause fission. This change in the material properties results in a negative reactivity contribution with increasing core temperature (and power). This is a common safety shutdown mechanism in commercial power reactors, small experimental reactors, and space reactor concepts. Since within the RK model

framework the reactivity is handled in a direct manner. the reactivity changes denoted by equation 4.18 can be modelled more directly as a change in only the infinite multiplication constant since that term is an explicit statement of material properties. In the RK model the δK term will be removed from the input vector and modelled internally as.

$$\delta K = -\alpha_f \delta T_f - \alpha_c \delta T_c. \quad (4.19)$$

where α_f and α_c are the Doppler and coolant temperature feedback coefficients respectively.

Since temperature variations in the core will cause physical changes to occur. there must exist a relationship between the temperature rise and the core loss probability. This relationship is independent of the neutronic effects described by the variation of δK in equation 4.19. The structure of the RK equations makes it a trivial matter to remove the δP_c term from the input vector and treat it internally as,

$$\delta P_c = -\alpha_e \delta T_f, \quad (4.20)$$

where α_e is the core loss feedback coefficient.

The RK state equations are restructured as a result of the above temperature feedback modelling. As usual the system is described by the matrix equation $\dot{X} = AX + BU$, where the state vector X and input vector U are now given by,

$$X = [\delta n_c \quad \delta n_r \quad \delta C_1 \quad \delta T_f \quad \delta T_c]^T, \quad (4.21a)$$

$$U = \begin{bmatrix} \delta P_r \\ \delta v_c/v_{c0} \end{bmatrix}. \quad (4.21b)$$

Internal to the structure of the newly formed A and B matrices are the previous ones given in Appendix A. The new matrices are formed directly from these matrices

and the structure of the equations developed in this chapter. These matrices for a 1 reflector, 1 precursor group simplification are also given in Appendix A. The subroutine MAKEAB is the FORTRAN translation of the modelling developed in this chapter. This module is the heart of all of the RK numerical codes listed in Appendix C.

CHAPTER V
INTEGRATION OF AUTOMATIC CONTROL
AND THE RK MODEL

A. Introduction

In the last few years, automatic control has become an integral part of almost every branch of science and engineering. This interest has been the result of the need for more intelligent ways to control large complicated systems with multiple inputs. A nuclear power plant is such a system. Unfortunately the regulatory structure that exists in the nuclear industry has limited the utilization of modern control methods in nuclear systems. Military applications of nuclear space reactors make the use of automatic control a requirement. However, the situation that exists in the LWR nuclear power industry has slowed this development phase of the space nuclear effort. The other factors that have prevented the development of automatic controllers for space are the difficulties posed by the integration of the point kinetics equations and potential control models. The objective of this chapter is to propose two multivariable controllers and integrate them into the RK model. This will illustrate the adaptability of the RK equations to the automatic form of input. Furthermore, the closed loop numerical computations presented in chapter VI will verify the need for automatic control in space nuclear applications.

Thus far, the RK model developed in the preceding chapters is only useful as an open loop analysis tool. The term "open loop" is used throughout this chapter to signify the responses resulting from operator determined inputs. This type of analysis is limiting from the point of view of scoping studies in that the selection of inputs can only be useful as a parameter sensitivity study. An automatic controller

(or servo compensator) provides the connection between the system outputs and the inputs that are needed for doing closed loop response studies. For example, in the tracking problem, this connection is done such that the output of concern follows a certain reference signal in time. This problem is also called the servo problem, hence the term "servo compensator". In this chapter two control methods are applied to the solution of the RK tracking problem. In both cases, the reference signals of concern are arbitrary steps and ramps.

The first controller is based on the well studied conventional state feedback approach. Since the precursor states in the RK model are assumed to be unmeasurable, an observer is necessary to estimate these states for feedback. Tracking requirements and plant stability can be satisfied through the use of this type of controller if the RK plant is controllable and observable. Controllability and observability are discussed in the following section. The second controller is based on output feedback and the spectral assignment of eigenmodes. This approach, referred to as modal control, was first developed in order to stabilize and mold the response of aircraft.²⁸ These investigations however did not consider the solution of the servo problem. The capability to solve the RK servo problem is demonstrated in the numerical results presented in chapter VI. The advantages and disadvantages of the state feedback and modal control approaches for to the solution of the RK servo problem are identified.

B. Controllability and Observability

The properties of controllability and observability are sufficient conditions to guarantee the existence of an automatic controller that will stabilize the closed loop system. In the state feedback case, controllability and observability are also

necessary since some of the states usually must be estimated with an observer. In the RK system, this is true since the precursor states are not measurable. Furthermore, a state feedback controller will have the ability to arbitrarily assign all of the closed loop eigenvalues. A controllable system is formally defined as one in which all of the states can be driven from an initial condition to any arbitrary final state with an appropriately chosen piecewise continuous input function. An observable system is defined as one in which knowledge of the input, output, and the system matrices are sufficient to determine the initial condition of all the states in the system. Mathematically, the property of controllability for linear time invariant systems can be determined by checking the dimension of the controllability array, (A, B) . The controllability array (A, B) is defined as,

$$(A, B) \equiv [B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B] \quad (5.1)$$

If the controllability array has rank equal to the rank of A (n), then the system is said to be controllable. Dually, the property of observability for a linear, time invariant system can be determined by checking the rank of the observability array, (C, A) . Again, if the rank of the observability array is equal to n , the system is observable.

Since the reflector return probabilities, P_r , only enter into the core neutron density equation, then an increase in the number of reflectors has no effect on the controllability properties of the RK system. Mathematically, this happens because the rank of B is at most two, regardless of the number of reflectors. The point treatment of the core in the RK model makes the distinction of the geometric location of the reflectors with respect to the core impossible. Therefore controllability only needs to be tested for the one reflector case. One should also

note the structure of the delayed neutron precursor states given by the third block row of the A matrix in Appendix A. Since each of the states are connected to the core density through the A_{21} block, and are linearly independent due to the diagonal structure of the A_{33} block, then we find again that controllability is guaranteed for multiple group problems if it exists for the one group problem.

Controllability and observability are tested directly by constructing the (A, B) and (C, A) arrays. Gaussian elimination with full pivoting is then used to find the rank of these matrices. While in general the numerical computation of rank cannot reliably be done, for certain systems that are well conditioned, the roundoff error that usually creates the computational difficulties is avoided. The validity of the rank computed using the full pivoting elimination scheme is tested by comparing the magnitude of the largest and the smallest element in the array before and after reduction. If many orders of magnitude are spanned between these numbers, then the rank obtained by this direct approach would be suspect. Using the MURR input data given in Table III, it was first found that the RK system is completely controllable and observable as long as at least one of the temperature feedback coefficients, α_f , α_c , and α_e is negative. Physically this indicates the fact that reactor instabilities caused by positive temperature (or power) feedback effects cannot be overridden by careful input manipulation. Fortunately, all conventional LWR designs and the majority of space reactor systems of interest exhibit an overall negative temperature feedback coefficient. The system was also found to be uncontrollable if P_{F_r} was zero. This is interpreted physically by the fact that reflector variations will certainly have no impact on the system response if neutrons never interact with the control drum. Again, this condition is unrealistic and will not exist in the systems under analysis. Controllability and observability of the RK system has therefore

been established for this test case.

C. Tracking Under the Internal Model Principle

In order to achieve robust asymptotic tracking of step and ramp reference signals, the application of the internal model principle is essential. Robustness refers to the ability of the closed loop system to maintain tracking when the plant parameters are slightly perturbed. The internal model principle basically states that in order to track a certain response with zero steady state error, an identical match of the reference modes must be included in the system internally. This will cause the unstable reference mode to be cancelled by the internal model. This is best illustrated in the Laplace transform space. Suppose we want to track a step signal; in the Laplace transform space a step is represented by $1/s$. The internal model principle dictates the addition of an integrator to the system so that the transfer function between the error and the reference signal has an s in the numerator. This internal “ s ” will cancel the “ s ” posed by the step reference signal, thereby resulting in asymptotic tracking with zero error. Furthermore, this tracking will persist even if the plant parameters are modified, as long as closed loop stability is not corrupted.

There are two basic limitations of the application of this principle. First, tracking is only obtained when unity feedback is employed, i.e., the input to the controller must be the error signal, $e = R(t) - y(t)$. In practice, unity feedback situations unfortunately rarely exist because the output, $y(t)$, is usually not the same as the measurable output, $y_m(t)$. This is because measured signals are typically voltages or currents, while the control variable is more physical (such as power or temperature). This also occurs on the input side where the controller

determines voltage inputs which most likely are incompatible with the inputs in the model equations. Often the transducer that provides this link has unity gain and also responds much faster than the system. In such a case, it is standard engineering practice to ignore the transducer as will be done here. Control methods that solve the more difficult general problem where the transducers are not neglected are beyond the scope of this study. Interested readers are referred to Schumacher³⁵ and Ohm³⁶ who take a geometric approach to develop constructive proofs of the solvability of the general problem. Since the main purpose here is to determine the potential advantages of automatic versus manual control in space nuclear applications, unity feedback will be assumed. The second limitation of the internal model principle is that unstable zeroes of the open loop plant cannot be tracked. This limitation exists because any unstable zeros will offset the internal modes leaving the reference signal to be tracked unaccounted for.

The desire to track steps and ramps leads to the addition of two integrators to the RK system. These two integrators provide two error transfer function zeros (at zero) which are needed in order to track the first order polynomial reference signals. Two more states are therefore added to the RK system equations resulting in an augmented system. These states, denoted by q_1 and q_2 , correspond to the following dynamical equations,

$$\dot{q}_1 = R(t) - y, \quad \dot{q}_2 = q_1. \quad (5.2)$$

and the augmented plant A matrix becomes,

$$A = \begin{pmatrix} A_p & 0 & 0 \\ -C_p & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (5.3)$$

Once stability is achieved, the tracking of steps and ramps will result. The primary work involved in the development of a state feedback controller is the development

of a feasible algorithm to compute the feedback gains. The algorithm chosen for the state feedback controller is based on the solution of Sylvester's equation. This methodology was first published by Bhattacharyya and deSouza³⁷ and studied further by Keel in 1982³⁸. The theory behind this favorable pole placement algorithm is briefly reviewed below for completeness and continuity.

D. Pole Placement Algorithm

The pole placement algorithm addresses the problem of finding a feedback matrix, F , such that the control law, $u = Fx$, results in a stable closed loop matrix, $A - BF$. Furthermore, arbitrary assignment of all the eigenvalues of the closed loop system is desired. Suppose we construct a matrix, M , such that the spectrum of M is equal to the desired spectrum of the closed loop. This is written as,

$$\sigma(A + BF) = \sigma(M), \quad T^{-1}(A - BF)T = M, \quad (5.4)$$

where the above matrix T is invertible and must exist for the problem to be solvable. If we let the matrix, G , be defined as the product of F and T then we have,

$$AT - TM = -BG, \quad FT = G. \quad (5.5)$$

The pole placement problem is solved if an arbitrarily chosen G matrix is found such that equation 5.5 has a solution T that is invertible. Keel³⁸ showed that if (A, B) is controllable and (G, M) is observable then such a solution exists. Furthermore, G parameterizes the family of all F matrices that will assign the eigenvalues of $A - BF$ to that of the chosen matrix, M . For the single input case, the F matrix is unique and therefore independent of the selection of G .

Equation 5.5 is a well studied equation known as Sylvester's equation. Established stable computational solutions to Sylvester's equation exist for T ³⁹, most of

which involve reduction of the A , M , and BG matrices to Hessenberg-Schur form. This reduction algorithm is the same one that is used in most eigenvalue finding routines. Since a computer program is being constructed to perform closed loop RK analysis that uses such an eigenvalue routine, the implementation of a Sylvester equation solution into the program was considerably simplified.

Using the pole placement algorithm discussed above, the loop can be closed using the state feedback law,

$$U = [F_{p1} \quad F_{p2} \quad F_{p3} \quad F_{q1} \quad F_{q2}] \begin{pmatrix} x_{p1} \\ x_{p2} \\ x_{p3} \\ q_1 \\ q_2 \end{pmatrix}. \quad (5.6)$$

Note that the plant states have been divided into three parts: core neutron concentration, reflector concentrations and temperatures, and the unmeasurable precursor concentrations. The purpose for this division is made clear in the following section concerning precursor state estimation. The closed loop RK model using a state feedback controller would be complete if the precursor states, represented by element x_{p3} , were available from measurement. However, these precursor groups are fictitious in the sense that they exist only in order to describe observed groups of delayed neutrons having certain decay characteristics. Furthermore, even if each actual delayed neutron precursor were taken into account, it would be a formidable reactor instrumentation problem to measure such intricate particle densities. For these reasons a state estimator (also often called an observer) must become an integral part of the closed loop RK equations.

E. Observer Design

The basic purpose of an observer is to develop equations that model measure-

able auxiliary states, say z , that asymptotically approach the states that are not measurable. These auxiliary states are then substituted for the unmeasurable ones in the state feedback control law. Consider the following dynamical equation.

$$\dot{z} = Nz - Ly_m - Gu. \quad (5.7)$$

where N , L , and G are unknown matrices. If the matrix V is chosen such that Vx represents the unmeasurable states, then the goal of the observer is to have,

$$\lim_{t \rightarrow \infty} z - Vx = 0. \quad (5.8)$$

In order for this to happen, the error defined as $e \equiv z - Vx$, must be represented by a stable unforced dynamical equation. It follows directly from equation 5.7 that the error is given by,

$$\dot{e} = \dot{z} - V\dot{x} = Ne + (NV - VA + LC)x + (G - VB)u. \quad (5.9)$$

Since the V matrix is user specified, G can always be found directly as $G = VB$. The difficulty lies in finding an L matrix such that $NV - VA + LC = 0$ and $\sigma(N)$ is stable. In general this is a difficult task and has been approached in many different and complicated ways. When V is chosen to be the identity matrix, the observer is called an identity observer and all of the states are being estimated. The solution of the identity observer problem is significantly easier than the general one, since equation 5.9 reduces to,

$$V = I, \quad G = B, \quad N = A - LC. \quad (5.10)$$

Equation 5.10 can be solved using the state feedback pole placement algorithm discussed in the previous section by finding L such that $\sigma(A - LC)$ is stable. Perhaps

the most straight forward solution of the general observer problem is to somehow reduce it to an equivalent identity observer problem. This is the approach taken here.

The first step is to divide the unaugmented system into two parts: measureable and unmeasureable. Since the observer is driven by the input and the output only, the partition is made such that the first m states are the outputs of the system. Normally this would involve a coordinate transformation; however, the output of most interest in the RK system is merely the core neutron concentration which is the first state. The partitioned system is therefore given by,

$$\dot{y}_m = A_{11}y_m - A_{12}x_u + B_1u. \quad (5.11a)$$

$$\dot{x}_u = A_{21}y_m + A_{22}x_u - B_2u. \quad (5.11b)$$

The identity observer reduction is done by taking $A_{12}x_u$ as the output of the unmeasureable equation, and $A_{21}y_m - B_2u$ as the input. The motivation for making these definitions is as follows. The chosen input consists of the only terms not involving x_u in the unmeasureable equation, and the chosen output consists of the only term involving x_u in the measureable equation. The observer dynamical equation is given by,

$$\dot{z} = (A_{22} - LA_{12})z - LA_{12}x_u + (A_{21}y_m - B_2u). \quad (5.12)$$

Equation 5.11b is used to eliminate the $A_{12}x_u$ above to give,

$$\dot{z} = (A_{22} - LA_{12})z - L(y_m - A_{11}\dot{y}_m - B_1u) - (A_{21}y_m + B_2u), \quad (5.13)$$

It is easily shown that the error between z and x_u is given by,

$$(\dot{z} - \dot{x}_u) = (A_{22} - LA_{12})(z - x_u). \quad (5.14)$$

In order for z to estimate x_u then we choose L such that $A_{22} - LA_{12}$ is stable. As stated before, the spectrum of this matrix can be assigned arbitrarily using the pole placement algorithm. One more step must be done in order to integrate the observer dynamical equations and the RK plant equations. The variable $w \equiv z - Ly_m$ is introduced into equation 5.13 in order to eliminate the \dot{y}_m term. The dynamical equations that are added to the system therefore are given by,

$$\dot{w} = (A_{22} - LA_{12})w + [(A_{22} - LA_{12})L - LA_{11} - A_{21}]x_m - (B_2 - LB_1)u. \quad (5.15)$$

At this point, two options are available. The first option is to use all of the estimated observer states in the state feedback law. This would be somewhat wasteful since only the precursor states are unavailable and the reflector neutron concentrations and the temperatures can be measured. This leaves the second choice of replacing only the precursor states with the observed ones the better option. This can be done because these are the only states that are truly not available for measurement. This brings forth the question of why the unused auxiliary states were included in the observer dynamical equations. The answer to this question is that the information provided by these states is necessary in order to observe the unmeasurable precursor states. In other words, the identity observer reduction method can only be done if we observe all of the states not included in the output, thereby making the minimum order of the observer equal to the order of the open loop system less the number of outputs.

Now we have enough information to write the final closed loop equation. For simplicity the following definitions are made,

$$\begin{aligned} A_c &\equiv A_{22} - LA_{12} \\ B_c &\equiv B_2 - LB_1 \\ D_c &\equiv (A_{22} - LA_{12})L - LA_{11} - A_{21} \end{aligned} \quad (5.16)$$

and the observer is partitioned into the measureable (unused) auxiliary states and the unmeasureable (used) states as follows.

$$w_m = A_{c1}w_m - A_{c2}w_u - B_{c1}u - D_{c1} \quad (5.17a)$$

$$w_u = A_{c3}w_m + A_{c4}w_u - B_{c2}u - D_{c2}. \quad (5.17b)$$

The motivation behind the division of the plant into the three parts given by equation 5.3 are now evident since it is trivial to replace $x_{1,3}$ with the measureable variable $z_u = w_u + L_u x_m$ in the state feedback control law. Once this is done, all of the states in the control law are measureable and the controller becomes legitimate. The feedback gains F are unchanged: however, now the closed loop system is much larger because it includes the observer dynamical equations. The final state feedback controlled system with arbitrary pole placement is written in matrix form as,

$$\dot{X}_{cl} = A_{cl}X_{cl} + B_{cl}R(t). \quad (5.18)$$

where details of the above matrices are given in Appendix A in block matrix form.

The application of the state feedback controller developed above is simple and straight forward. However, the selection of the closed loop eigenvalues and the parameterizing G matrix is at best a trial and error process. Given the open loop plant and the $R(t)$ tracking signal, one merely has to choose a desirable set of closed loop eigenvalues. This selection is made by the choice of the M matrix in the pole placement algorithm. The pole placement algorithm is utilized once to find the F state feedback gains and again to find the L observer gains. Once these two matrices are found, the A_{cl} matrix can be constructed. This methodology was incorporated into the computer code RKSF (Reflected Kinetics State Feedback). The pole placement algorithm was adopted from a program originally written by

Keel³⁵ The difficulty that arises in the use of RKSF is in choosing a suitable set of closed loop eigenvalues and the G parameterizing matrix. Some numerical results obtained from the RKSF code are presented in the next chapter. As the results will indicate, complete stability results in asymptotic tracking of step and ramp signals as desired. This demonstrates the application of the modern multivariable control techniques to the reflected kinetics approach of space nuclear reactor kinetics studies. The modal controller presented below avoids some of the limitations posed by the state feedback controller and still obtains asymptotic tracking of the desired step and ramp reference signals.

F. Modal Control Theory

There are three primary motivations for the consideration of modal control (eigenstructure assignment). First, the open loop results given in chapter 6 will indicate the stability and rigidity of two of the eigenmodes of the RK system. Therefore, the decoupled assignment of chosen eigenmodes could provide a potential control advantage. Secondly, the modal method is based on output feedback which significantly reduces the size of the closed loop system from that of the state feedback case. Finally, the modal method is desired in order to shape the transient response of the system by coupling or decoupling certain key eigenmodes. The transient response is dependent upon the eigenvectors as well as the eigenvalues of the system, and the state feedback G matrix parameterization is limited in the assignability of such eigenvectors. Even though we sacrifice the assignability of all of the eigenvalues in the modal method, we gain the ability to assign as many eigenvalues and eigenvectors as there are measurable system outputs.

Andry²⁸ utilized modal control to solve the aircraft stability problem. In order

to solve the servo problem with modal control, the integrators are taken to be measureable outputs. These additional outputs are needed in order to stabilize the closed loop system. The achievable eigenvector and the feedback gain computation algorithms are repeated here for the purpose of continuity.

The derivation of the feedback matrix F needed to achieve the eigenvalue/eigenvector pairs chosen must begin by identifying the eigenvectors that can be exactly achieved. As we will see, it is rarely possible to assign the exact desired eigenvectors if we require the exact assignment of the desired eigenvalues. We proceed by assuming that the input matrix B has full rank. It is then assumed that none of the open loop eigenvalues match the eigenvalues that are to be assigned by the modal controller. Given the output control law with r outputs,

$$u = Fy = FCx. \quad (5.19)$$

the task is to find F such that the closed loop matrix, $A - BFC$, assigns the r eigenvalue/eigenvector pairs as desired.

Suppose we have chosen a set of r desired eigenvalues and eigenvectors to assign using the modal method. These modes are denoted by λ_i and V_i^d , where the d superscript on the eigenvector denotes "desired". By definition, the closed loop eigenvectors of the system must satisfy the expression,

$$(A - BFC)V_i^a = \lambda_i V_i^a, \quad (5.20)$$

where V_i^a is an exactly achievable eigenvector. It is important to realize that V_i^a will not be equal to V_i^d in most cases. What this means is that the desired eigenvector most likely will not lie in the correct subspace thereby making it impossible to achieve under the chosen eigenvalue. However if we find the eigenvector in the

achievable subspace that is closest to the desired one. then this is the best we can do.

The achievable eigenvectors are found by projecting the desired eigenvectors onto the achievable subspace. When equation 5.20 is rewritten as.

$$V_i^a = L_i F C V_i^d, \quad L_i \equiv (\lambda_i I - A)^{-1} B. \quad (5.21)$$

we see that the columns of the L_i matrix span this subspace. Therefore, some vector, say z_i , exists such that $V_i^a = L_i z_i$. This vector is found by minimizing the geometric norm between the desired and achievable eigenvectors with respect to z_i :

$$J \equiv \|V_i^d - L_i z_i\|^2. \quad (5.22)$$

$$\frac{\partial J}{\partial z_i} = -2L_i^T (V_i^d - L_i z_i) = 0. \quad (5.23)$$

The above equation is solved for z_i to give,

$$z_i = (L_i^T L_i)^{-1} L_i^T V_i^d. \quad (5.24)$$

This projection procedure transforms the desired eigenvalue/eigenvector pairs into achievable ones. A slight modification of the projection equations is needed if one wishes to assign an eigenvector where some of the elements are unspecified. In this case we define a matrix P such that PV_i^d strips off only the specified elements. Equation 5.24 is then modified by replacing L_i with PL_i and V_i with PV_i .

The achievable eigenvector calculation must be done for each of the modes that one wants to assign. This is necessary in order to exactly assign the eigenvalues. The F gains are then computed using the algorithm reviewed below.

Since B was assumed to have full rank, then a transformation matrix, T_b , exists such that,

$$T_b^{-1} B = \begin{pmatrix} I_m \\ 0 \end{pmatrix}. \quad (5.25)$$

Any matrix G that fills out $T_t = (B \ G)$ in non singular fashion will give us the desired transformation. Now we perform a similarity transformation on the open loop equation by letting $X = T_b X_t$. The achievable eigenvectors are also transformed as $V_t^a = T_b V_{it}^a$. These eigenvectors are broken into the first m rows and the bottom $n - m$ rows as.

$$V_{it}^a = (q_t, w_t)^T. \quad (5.26)$$

Recall that the eigenvalue, eigenvector pairs satisfy the relationship.

$$(\lambda_t I - A_t) V_{it}^a = B_t F C_t V_{it}^a \quad (5.27)$$

regardless of the basis. Since the first m terms of the B_t matrix constitute the identity matrix, then the first m rows of equation 5.27 in the "t" basis gives,

$$(A_{1t} + F C_t) V_{it}^a = \lambda_t q_t. \quad (5.28)$$

where A_{1t} denotes the first m rows of A_t . Since there are r assignable modes, we write the above equation once for each mode and collect them into one matrix expression that is easily solved for F as,

$$Q_t \equiv (\lambda_1 q_1 \cdots \lambda_r q_r), \quad (5.29a)$$

$$(A_{1t} + F C_t) V_t^a = Q_t. \quad (5.29b)$$

$$F = (Q_t - A_{1t} V_t^a) (C V^a)^{-1} \quad (5.30)$$

The $C V^a$ matrix will not be invertible or will be ill conditioned if the measurable output has little influence on the achievable eigenvectors. Mathematically this is written as,

$$Im(V) \cap ker(C) \neq 0. \quad (5.31)$$

In the RK system, where the output of interest is usually equal to the first state, this merely stipulates that the achievable eigenvector cannot have a zero component in the first row.

While the modal assignment theory presented above is only for real eigenvalue and eigenvector assignments, Moore⁴⁰ developed a transformation that avoids the need for complex arithmetic, thereby making equation 5.30 a general result. The F matrix computed using this approach will exactly assign r eigenvalues and will also assign the projection of r eigenvectors onto the achievable subspace. The application of this modal theory is simple in that one merely has to choose the r eigenmodes to assign. Perhaps more important than the assignment of the particular values in the eigenvectors is the power this method gives us to assign zeros to certain elements, thereby decoupling various modes from each other. Again, the exact assignment of a zero may not be possible due to the necessity of the vectors inclusion in the achievable subspace.

The application of this modal controller to the RK system was implemented in the FORTRAN computer code RKMOTAL (see Appendix C). This code performs the achievable eigenvector calculation given a requested set of desirable eigenvalue/eigenvector pairs. It then continues with the feedback gain computation and the construction of the closed loop matrix. Numerical examples are presented in the following chapter.

CHAPTER VI

NUMERICAL COMPUTATIONS AND RESULTS

A. Introduction

The numerical results and computations are divided into two primary categories: manual (open loop) and automatic (closed loop). These numerical computations are intended to provide valuable insight concerning the transient capabilities, dependencies, and limitations of the reflector and coolant flow controlled space reactor. The RK open loop analysis computer code (RKOPEN) is based on the modelling given in chapter II and chapter IV, and was developed for the specific purpose of performing the open loop computations. The scope of the RK model as discussed in chapter II is restricted to scoping feasibility studies. Furthermore the linearization assumptions as seen in chapter III limit the range of valid transients that can be considered. RKOPEN is first utilized to demonstrate the importance of the origin of the reactivity input in the transient behavior of such systems. Furthermore, numerical computations are also performed in order to qualitatively verify the correctness of RKOPEN from a physical standpoint. The validation of RKOPEN is extremely important in order for the automated systems to properly serve their function. This is due to the fact that the basic structure of RKOPEN is also an integral part of the two closed loop codes. A parameter sensitivity scoping study intended to determine which parameters should be the focus of detailed core design is also done. The parameters of primary interest are core neutron lifetime, temperature feedback coefficients, and heat transfer coefficient. Finally, the open loop computations will prepare us for the intelligent use of the two controllers in solving the RK tracking problem.

The RK state feedback controlled system is described by the RKSF computer code. The RK modal controlled system is described by the RKMODAL code. These programs are the result of the integration of the control theory reviewed in chapter V and the RK plant model described in chapter II and chapter IV. The closed loop computations are intended to serve three purposes. First they are intended to demonstrate the success of the integration of the RK model and an automatic controller. Secondly, the capability to track arbitrary steps and ramps under both the state feedback and the modal controller will be demonstrated. These solutions of the servo problem permit the identification of the advantages and disadvantages of both forms of control as applied to the RK system. Finally, the control inputs are then used as approximate inputs to the RKOPEN code and the transient responses are found. This will illustrate the usefulness of autonomous control as opposed to unprecise manual methods. This exercise will also identify a potential form of control for dedicated space reactor missions.

B. Open Loop Numerical Computations

The transient behavior of the system is first considered as various system parameters are varied over appropriate ranges. While variations in the fuel, reflectors, and coolant (type and configuration) cannot be accounted for directly in a simplified point model such as RK, variations in the parameters that correspond to their selection can easily be done. In particular, reactor lifetime (fuel and reflector characteristics), reactivity feedback coefficients (fuel and coolant type), macroscopic fission cross section (fuel composition), and heat transfer coefficient (core/coolant interface) will be considered. The information provided by these open loop runs is valuable in determining which parameters should be the focus of more

comprehensive space reactor core designs. The neutronic parameters of the reference system used in these computations was again based on the MURR facility. There are two motivations for using the MURR data. First, the neutron spectrum and the core geometry of the MURR facility are similar to most paper multi-megawatt space reactors under consideration. Secondly, since most of the data for such paper designs are classified, a deliberate attempt was made to use available data that could in no way be associated with such sensitive information. The thermal properties of the fuel and the coolant were taken to be the average values for Uranium-235 (U235) fuel and Helium (He) coolant in the estimated operating temperature range. Helium was considered as the coolant since many "paper" multi-megawatt space reactor designs involve a packed fuel bed design through which He coolant is usually a requirement. The parameters required as input to the RKOPEN code for the MURR reference system are given in Table V.

The reflector reactivity input (δP_r) for the reference case corresponds to a reactivity ramp lasting for 32 seconds from zero to a value of $\delta \rho = 0.001$. Beyond 32 seconds, the reactivity equivalent is held at 0.001. The core neutron concentration and bulk core temperature are plotted in Fig. 5 and Fig. 6 respectively. Space limitations prevent the showing of the reflector and coolant temperatures. Also the precursor concentrations are presented only for the last two cases discussed later in this section. In all of the RKOPEN figures presented, the values represent the change above the initial value of that state. The initial values of the states for the reference problem are also given in Table V. The eigenvalues of the RK system matrix are given in Table VI. These eigenvalues are useful for the qualitative explanation of the transient response shown in Fig. 5 and Fig. 6. First, the fact that all of the eigenvalues are negative indicates that once perturbed by an

Table V

Material Properties and Other Pertinent Data
for Open and Closed RK Calculations

<u>Core/Fuel</u>	<u>Coolant/Thermal</u>	<u>Reflector/Misc.</u>
λ_c 5.7×10^{-4} s	τ_c 5.4059×10^{-4} s	λ_r 2.83×10^{-4} s
τ_f 0.34695 s	τ_L 7.0674×10^{-4} s	P_r 0.29868
P_c 0.34695	h_f 2400 Btu/ft ² -s	P_{Fr} 1.0000
α_f 10^{-4}	α_c 10^{-6}	α_E 10^{-6}
C_{pp} 52.668 Btu/ft ³ F	C_{pp} 0.05566 Btu/ft ³ F	
T_{fo} 1478.9 °C	T_c 436.5 °C	T_{in} 323.0 °C
n_{co} 1.0×10^5 cm ⁻³ s ⁻¹	V_{co} 1548.0 lbm/hr	n_{ro} 6.8065×10^4 cm ⁻³ s ⁻¹
χ 1.0948 KW/n	A_s 7577 ft ²	C_{io} 3.5651×10^7 cm ⁻³ s ⁻¹
V_f 31.98 ft ³	V_c 49.06 ft ³	Σ_f 0.2469 cm ⁻¹

Table VI

Eigenvalue Comparison for the Zero Temperature
Feedback and Reduced Neutron Lifetime Case

Reference Case	Zero Temperature Feedback Case	Reduced Neutron Lifetime Case
-3.7345E+03	-3.7345E+03	-4.3665E+03
-3.2665E+03	-3.2665E+03	-3.2665E+03
-1.0576E+01	-1.0844E+01	-1.8243E+01
-1.5552E+00	-1.2991E+00	-1.5390E+00
-1.1867E-02	1.3677E-15	-1.1893E-02

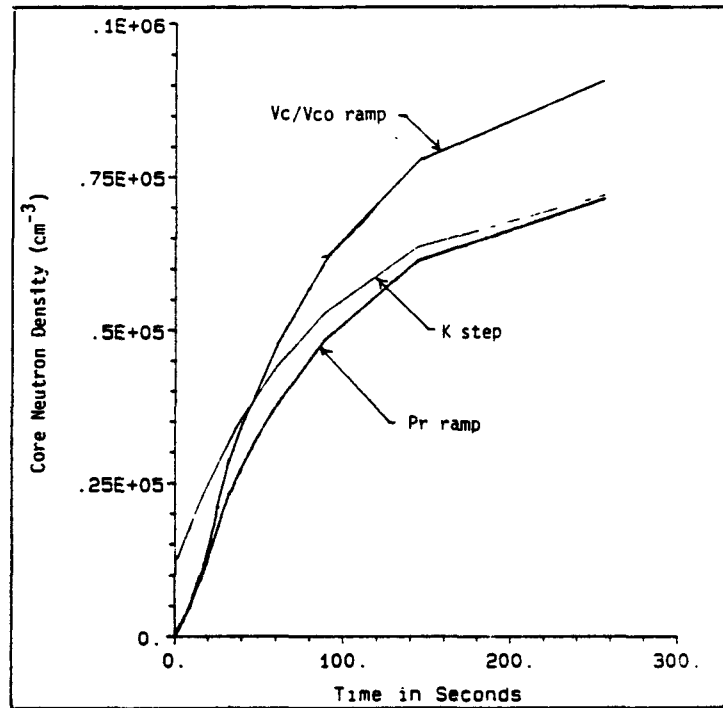


Fig. 5 Core Neutron Response for Three Different Reactivity Input Cases

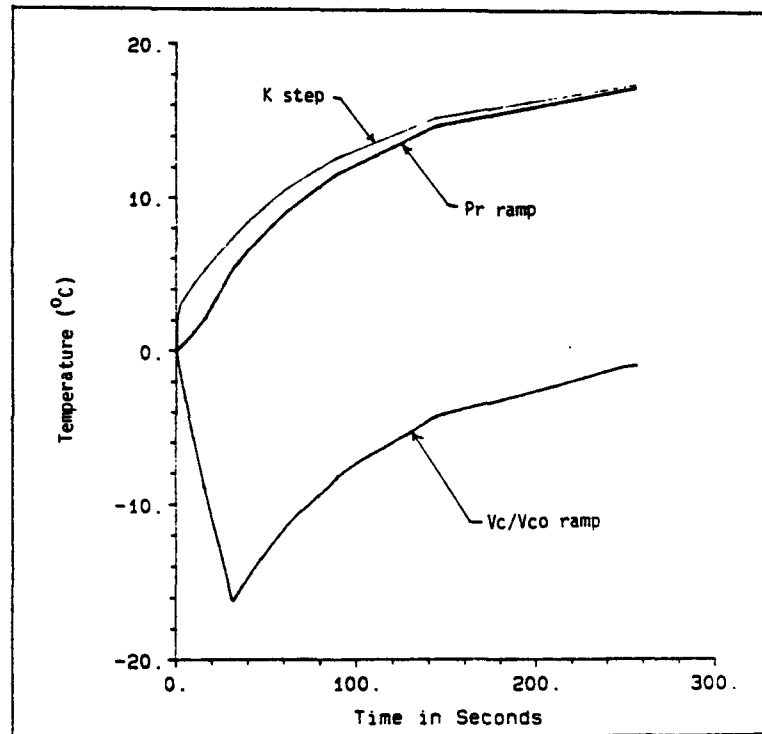


Fig. 6 Core Temperature Change for Three Different Reactivity Input Cases

comprehensive space reactor core designs. The neutronic parameters of the reference system used in these computations was again based on the MURR facility. There are two motivations for using the MURR data. First, the neutron spectrum and the core geometry of the MURR facility are similar to most paper multi-megawatt space reactors under consideration. Secondly, since most of the data for such paper designs are classified, a deliberate attempt was made to use available data that could in no way be associated with such sensitive information. The thermal properties of the fuel and the coolant were taken to be the average values for Uranium-235 (U235) fuel and Helium (He) coolant in the estimated operating temperature range. Helium was considered as the coolant since many "paper" multi-megawatt space reactor designs involve a packed fuel bed design through which He coolant is usually a requirement. The parameters required as input to the RKOPEN code for the MURR reference system are given in Table V.

The reflector reactivity input (δP_r) for the reference case corresponds to a reactivity ramp lasting for 32 seconds from zero to a value of $\delta\rho = 0.001$. Beyond 32 seconds, the reactivity equivalent is held at 0.001. The core neutron concentration and bulk core temperature are plotted in Fig. 5 and Fig. 6 respectively. Space limitations do not permit the showing of the reflector and coolant temperatures. Also the precursor concentrations are presented only for the last two cases discussed later in this section. In all of the RKOPEN figures presented, the values represent the change above the initial value of that state. The initial values of the states for the reference problem are also given in Table V. The eigenvalues of the RK system matrix are given in Table VI. These eigenvalues are useful for the qualitative explanation of the transient response shown in Fig. 5 and Fig. 6. First, the fact that all of the eigenvalues are negative indicates that once perturbed by an

external input, the system will eventually assume a new steady state configuration. This open loop stability is a direct result of the negative temperature feedback mechanisms. This stability is graphically visualized by noting that the temperature increase shown in Fig. 6 corresponds to the neutron density plateau shown in Fig. 5. When α_f , α_c , and α_e are zero or positive, non negative eigenvalues should appear as an indication of the unstable system configuration. This behavior is verified in the first open loop parameter study. The two extremely large negative eigenvalues dictate the shape of the early transient response of the system. The result of extensive numerical experimentation gave rise to the conclusion that these eigenvalues are very weak functions of all plant parameters except the core and reflector lifetimes.

Comparisons are made between the reference case and two other input scenarios as shown in Fig. 5 and Fig. 6. The first alternate input corresponds to an equivalent step reactivity insertion through an instantaneous change in the infinite multiplication constant. The equivalent change in K was found by backsolving equation 2.27 given $\delta\rho = 0.001$. One should note that the step K and the reflector ramp input responses asymptotically approach each other beyond 200 seconds. This convergence occurs for both the neutron concentration and the fuel and coolant temperatures. However, at early times the transient responses are very different. The most evident difference is the prompt jump behavior of the step reactivity insertion responses. This prompt jump, stable period, behavior is characteristic for the response of a reactor to abrupt changes in reactivity. The physical difference between abrupt changes in K and gradual changes in the reflector is preserved in the RK model through the direct reflector reactivity input capability. The importance of this feature of the model is emphasized by the comparisons given

in these figures. The remaining curve on these figures correspond to a transient induced by an increase in the coolant velocity. While it is impossible to determine what coolant velocity increase corresponds exactly to $\delta\rho = 0.001$, an approximation was determined by trial and error for qualitative discussion purposes. The chosen insertion was a ramp lasting 32 seconds resulting in a 20 percent rise in the coolant flow rate. As shown in Fig. 6, the improved heat transfer caused by the increased flow induces a steady drop in the fuel temperature (and coolant temperature not shown). This decrease in fuel temperature provides the positive reactivity by virtue of the negative temperature feedback mechanisms that drives the transient. After 32 seconds, the fuel and coolant temperatures are shown to rise as a result of the increased production in the core and reflector regions and the stabilization of the coolant flow. It is particularly informative to note that the core (and reflector) neutron concentrations do not show a prompt jump at all, rather they exhibit a gradual rise to an eventual higher steady state power level. This merely indicates that a 20 percent rise in coolant flow rate has more reactivity worth than $\delta\rho = 0.001$ for this particular configuration. However, as expected, the increased coolant flow causes the final equilibrium temperatures to be lower than in the δP_r input case. This computation demonstrates the fact that a large potential source of reactivity worth exists in the coolant velocity and therefore can be an effective means of reactivity control. Furthermore, the effect that this form of input has on the system is again different than that of control drum input. The additional reactivity control offered by flow rate variation is important in space reactor designs that are often restricted by control drum reactivity limitations. In the automatic operation mode, coolant flow will therefore be utilized to its full advantage.

Variations of the parameters in the reference case input set were the subject of

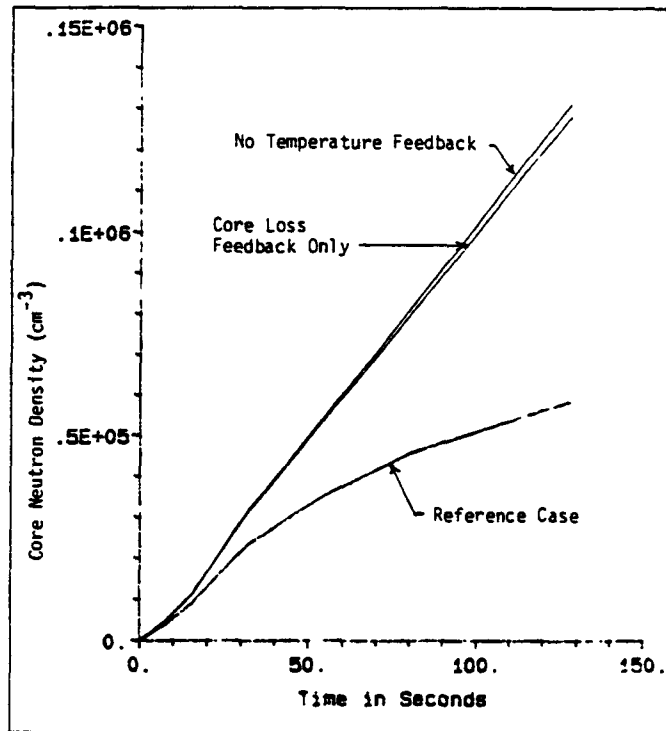


Fig. 7 Comparison of Neutron Response for Two No Temperature Feedback Cases

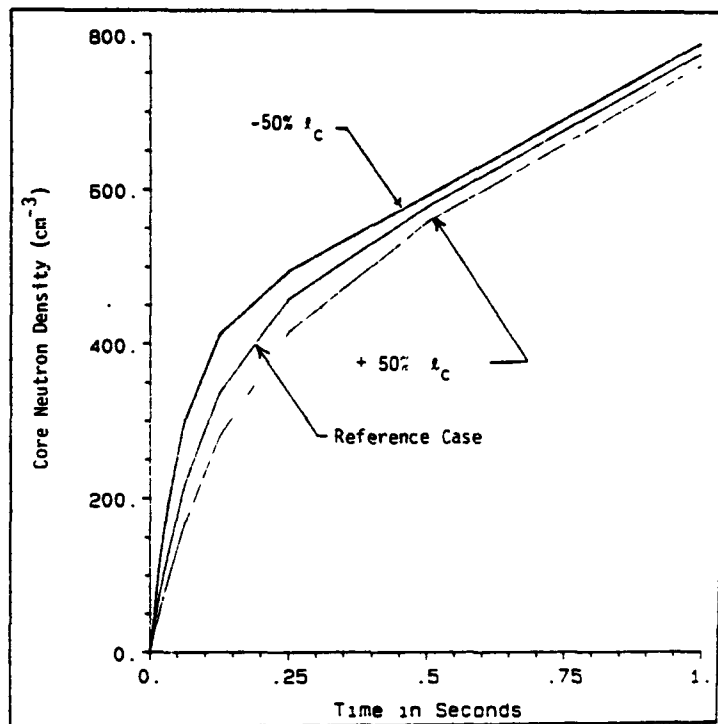


Fig. 8 Early Time Response for Varying Core Neutron Lifetimes

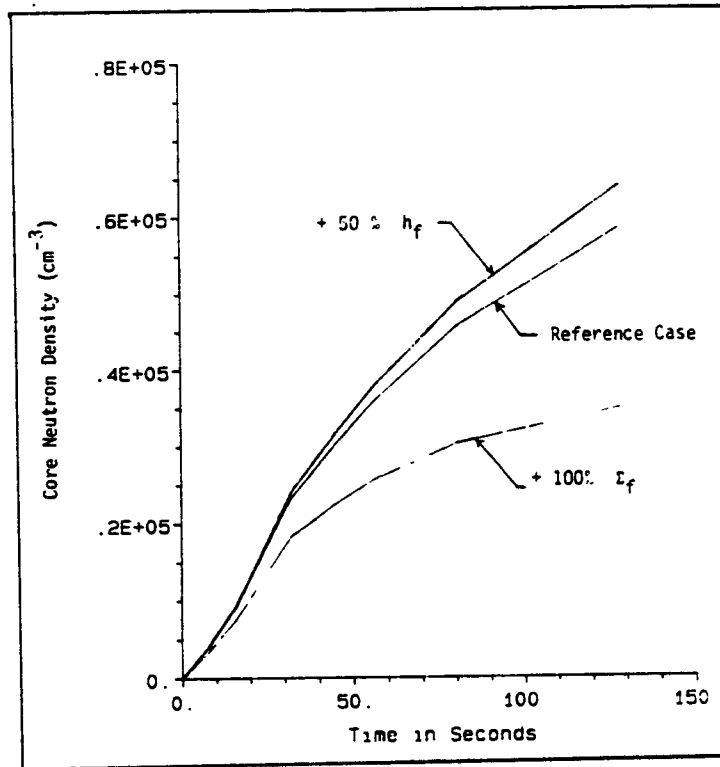


Fig. 9 Core Neutron Response for h_f and Σ_f Parameter Variations

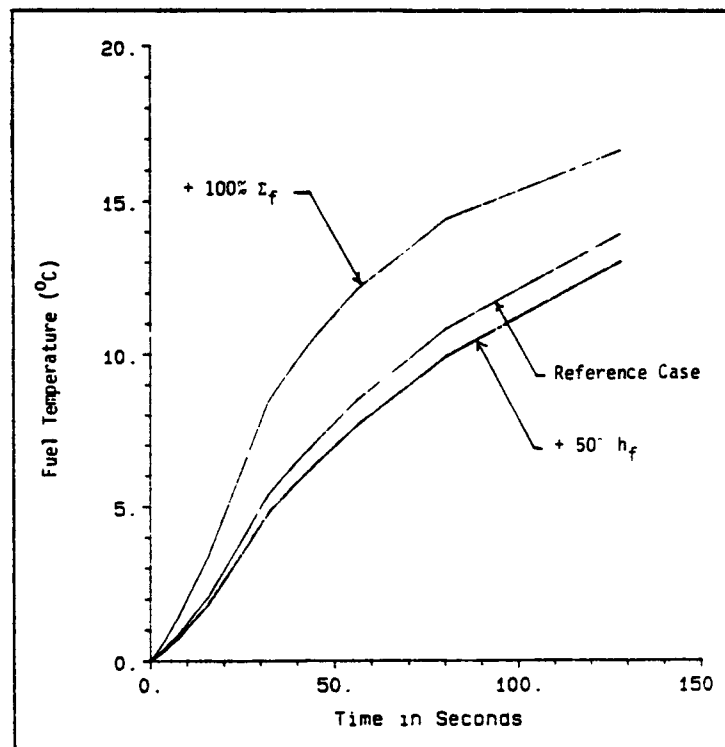


Fig. 10 Core Temperature Change for h_f and Σ_f Parameter Variations

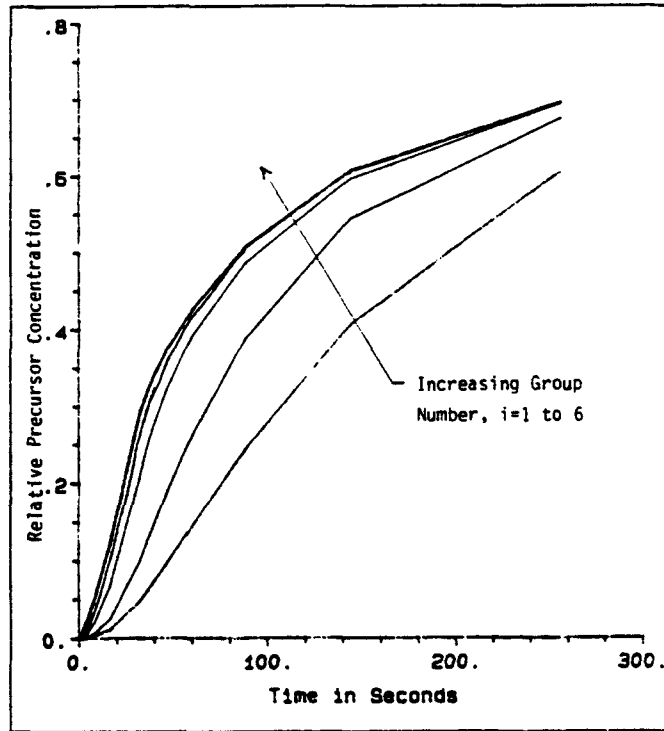


Fig. 11 Relative Precursor Concentrations for the Six Delayed Group Case

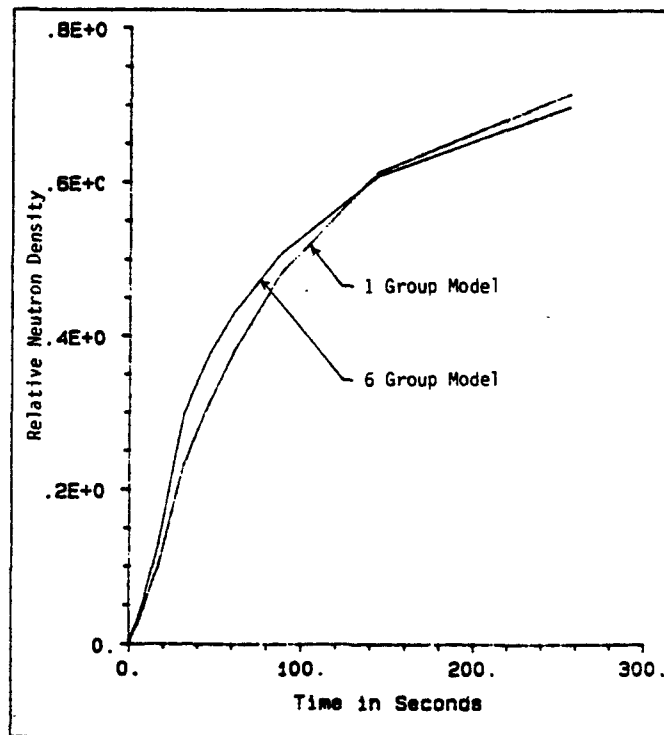


Fig. 12 Relative Neutron Response for the Six Delayed Group Case

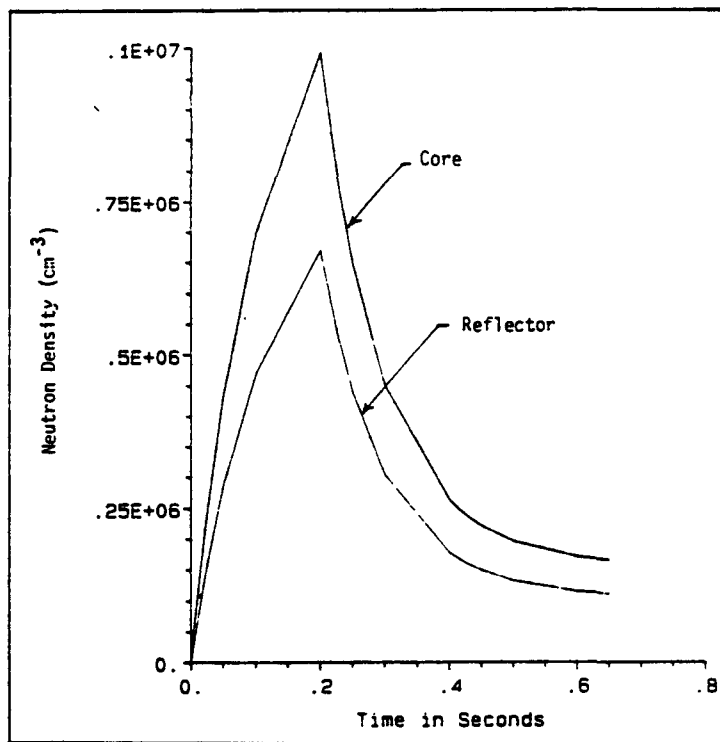


Fig. 13 Neutron Response for a 2 Dollar Pulse

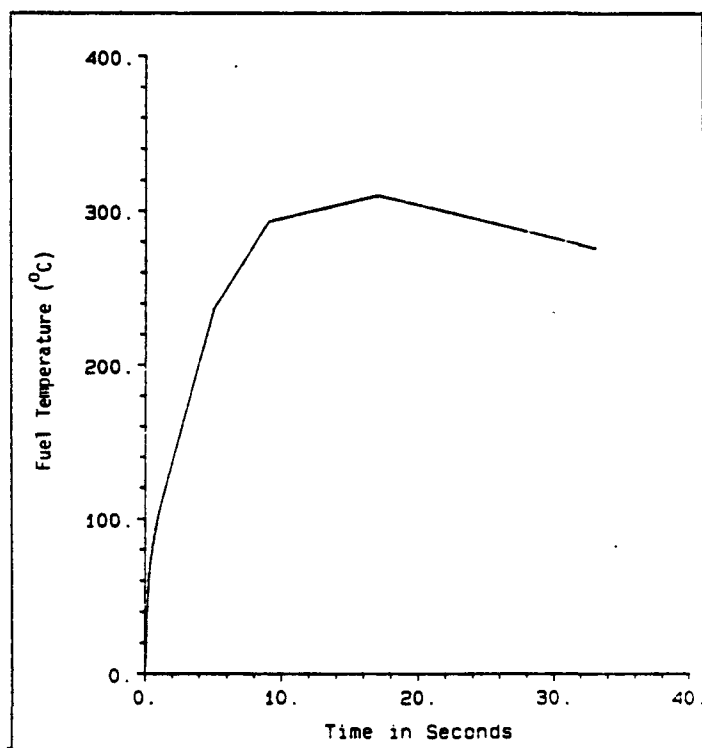


Fig. 14 Core Temperature for a 2 Dollar Pulse

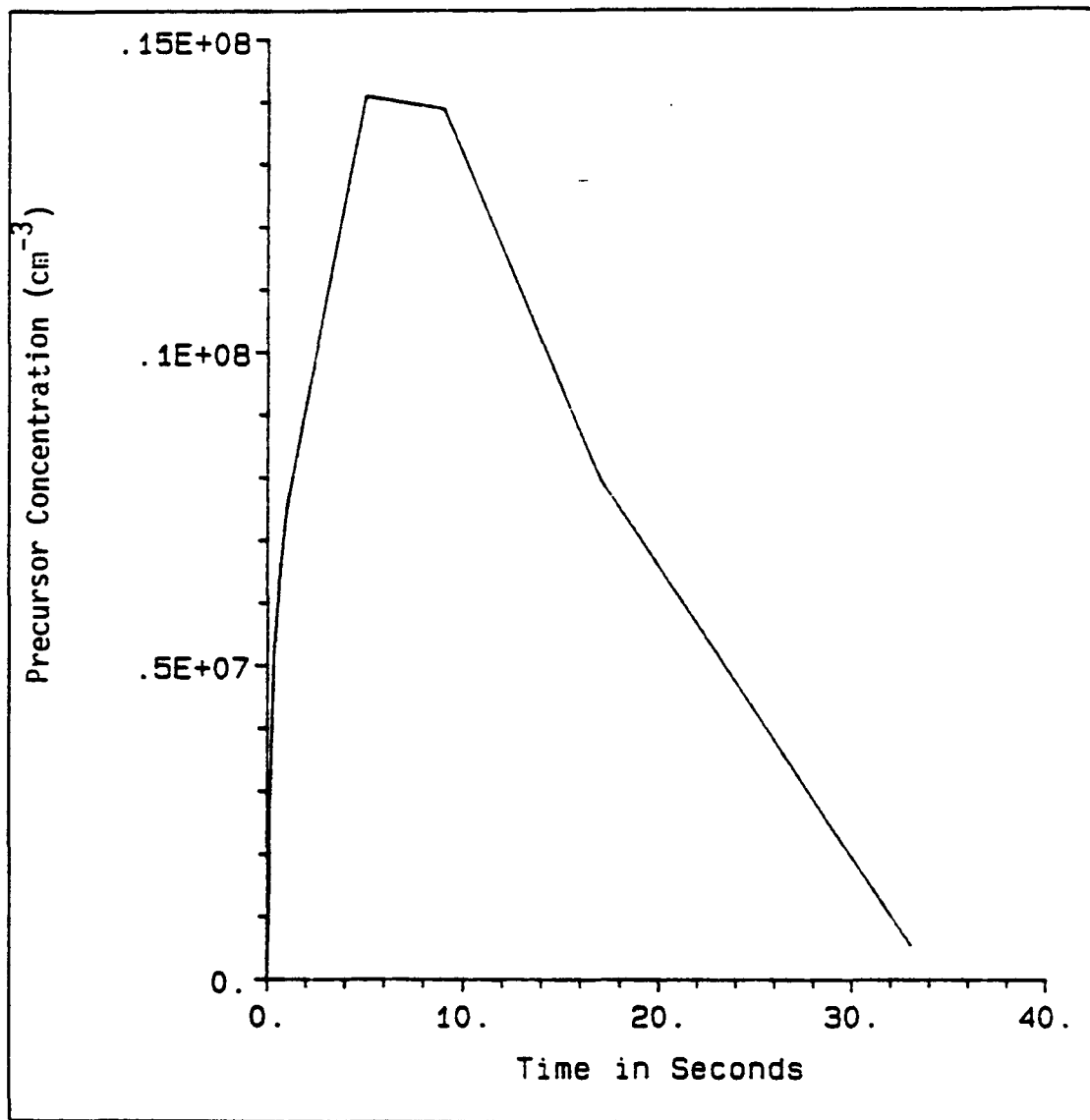


Fig. 15 Delayed Precursor Response for a
2 Dollar Pulse

numerous computations. In addition numerous computations were also performed with multiply reflected systems and the more rigorous six delayed neutron precursor group model. Unfortunately it is not possible to present all of the data gathered from this extensive sensitivity study. The six representative cases summarized below are therefore presented. Each of these cases are singly reflected since in chapter III it was shown that an equivalent singly reflected system can always be found. The generalized structure of the RK model and associated computer codes was retained since future study may involve the modelling of individual core regions that are neutronically coupled to the nearest reflector. In such a model, the equivalent reflector reduction would no longer be valid. In the six representative test cases, only one parameter in the reference case data set will be considered. While the changing of a true design characteristic such as fuel enrichment or coolant type will certainly have an effect on more than one parameter in the RK model equations, the restriction to single parameter variations will avoid the confusion of which parameter is causing what observed effect to occur. Furthermore, this approach will more readily permit the qualitative validation of the RKOPEN code. The transient results for the six cases are given by Fig. 7 through Fig. 15. The reflector densities are not shown since they closely parallel the core response for each of these cases. In the first four cases, the precursor concentrations are also omitted in the interest of brevity. In the last two cases, the precursor concentrations will prove to be of particular interest due to the nature of the problem.

The first case is the no temperature feedback case which is modelled by setting each feedback coefficient, α , to zero. The results are shown in Fig. 7. As previously stated, the absence of the negative temperature feedback safety shutdown mechanisms gives rise to reactor instability. This instability is evident in Fig. 7

where the response does not stabilize to a new value. The second curve shown in Fig. 7 is for a zero core loss feedback coefficient. The similarity between this response and the reference case indicates that the core loss coefficient must be much larger than 1×10^{-6} in order for core loss temperature dependency to constitute an effective shutdown mechanism. It is also interesting to note that the eigenvalues of the RK plant matrix for the no feedback case all remained nearly the same except for the smallest one which went to zero (within the accuracy of the machine) as seen in Table VI. This suggests that the temperature feedback mode is relatively loosely coupled to the other modes in the RK system. The impact that the feedback coefficients have on the stability properties of the system should render its careful consideration in future detailed system designs.

The second case involves two variations of the core lifetime by an order of magnitude. As evident in Fig. 8, the lifetime only has an effect on the early time response of the system. The curve representing the smaller lifetime shows a larger and faster "prompt jump" than the reference case. This is expected since shorter core lifetimes indicate the more rapid transmission of information through the system. Likewise, longer lifetimes represent more sluggish transmissions of information through the system. The core lifetime therefore governs how fast our reactor can respond, which is an important consideration in military space nuclear applications. After one second, the three computations are basically indistinguishable. As stated earlier, this is mathematically explained by the observation that changing the core lifetime only has an appreciable effect on the large negative eigenvalue. The rigidity of this eigenmode will prove to be important from the point of view of modal control.

The third case involves a 100 percent rise in the macroscopic fission cross section. Unlike one would normally expect, this causes the overall core neutron

concentration to be smaller than the reference case as shown in Fig. 9 and 10. This behavior is explained by the fact that even though Σ_f has been modified in magnitude, other reactor parameters (such as K , P_c , and β) that would accompany such a change in fuel characteristics are not accounted for. This illustrates the care that must be taken when developing an input deck for the RK model. Assuming that such a change in Σ_f can be done, more energy will be produced in the core without the sacrifice of anything else. This results in an overall rise in the core temperature as shown in Fig. 10. The negative feedback mechanisms dictate the more rapid stabilization of the core density that is consequently at a lower level. In a thermoelectric energy conversion system, this increase in temperature would result in a rise in electric power output. From this point of view, the effect of the above Σ_f variation is not totally unrealistic. The fourth case is similar in concept to the third except the temperature swings in the opposite direction. As seen in Fig. 10, a 50 percent increase in the heat transfer coefficient causes the reactor to run cooler. This slows the approach to the new equilibrium state and therefore gives rise to higher core densities as shown in Fig. 9. Again, it is important to note that other data such as coolant density and heat capacity may accompany such a change in the heat transfer coefficient. These results indicate that the lumped parameter equations developed in chapter IV and the temperature feedback modelling are functioning as intended.

The fifth case is the same as the reference case with six delayed neutron precursor groups. The different time responses of the precursors are shown in Fig. 11. The effect that the different decay rates of each group have on the transient response of the core neutron density is shown in Fig. 12. The small difference that the six delayed neutron precursor group case makes in the overall response of

the system leads to the conclusion that the additional array storage and processing time required is often not justified. The majority of the scoping calculations were therefore limited to the one delayed neutron precursor approximation. It is however noted that in more rigorous calculations, the use of six (or more) groups may be necessary.

The final open loop computation is done to determine what will occur in the absence of energy removal from the core during the rapid insertion of reactivity. This mode of operation is appropriately called pulsing. In pulsing mode, the transient proceeds so rapidly that the heat does not have time to transport out to through the coolant. This unanswered rise in core temperature gives rise to a large negative reactivity insertion which quickly reverses the transient. The precursor time lag resulting from the delayed neutron effect in this type of transient is particularly evident since the core neutron response is extremely rapid. The pulsing phenomena described above was accurately reproduced by the RKOPEN code as shown in Fig. 13 through Fig. 15. The reactivity input was $\delta P_r = 0.01225$, which corresponds to a 2 "dollar" ($\rho = 2\beta$) reactivity insertion.

C. State Feedback Numerical Computations

For purposes of continuity, all numerical computations involved the use of the same reference system that was used in the open loop computations above (see Table V). As pointed out in chapter IV, the main difficulty that arises in the application of a state feedback controller to large systems is that the eigenvalue selection and G matrix parameterization is primarily a "blind" trial and error process. Numerous sets of eigenvalues were considered before a suitable set was found. It is not possible nor productive to present this multitude of information here. Six carefully chosen G

Table VII

Eigenvalue and G Matrix Selection
for State Feedback Computations

Eigenvalues of Closed Loop for Cases G1 through G6

λ_{r1}	-1.0	-0.5	-2.0	-3.0	-0.8	-0.1	-0.2
λ_{L1}	-5.0	-4.0	-3.0	-2.0			

Eigenvalues of Closed Loop for Arbitrary Case

λ_{r1}	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
λ_{L1}	-5.0	-5.0	-5.0	-5.0			

Parameterizing G Matrices for Case G1 Through G6

Case	G Matrix						
G1	1	0	1	0	1	0	1
	0	1	0	1	0	1	0
G2	0	1	0	1	0	1	0
	1	0	1	0	1	0	1
G3	1	1	1	1	1	1	1
	0	0	0	0	0	1	1
G4	0	0	0	0	0	1	1
	1	1	1	1	1	1	1
G5	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
G6	1	1	0	0	1	1	1
	0	0	1	1	0	1	1

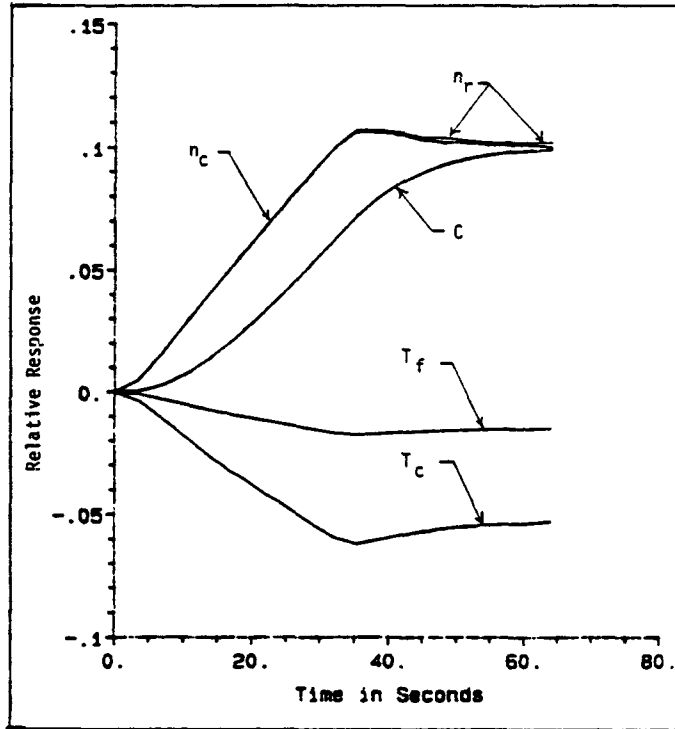


Fig. 16a Case G1 State Feedback Solution to the Ramp Servo Problem

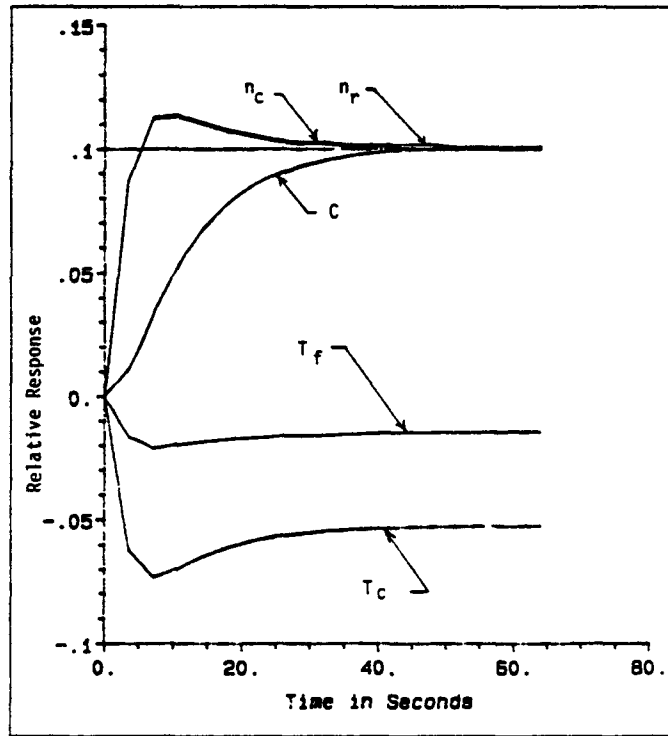


Fig. 16b Case G1 State Feedback Solution to the Step Servo Problem

matrices each producing different control scenarios are considered. The eigenvalues and G matrices are given in Table VII. The different G cases will be referred to as $G1$ through $G6$ for notational simplicity. Finally, an arbitrarily chosen stable eigenvalue set and G matrix are used to demonstrate the potential for disaster despite the stability and tracking behavior of the state feedback closed loop system.

The first problem to be solved is the tracking of a moderate ramp to a 10 percent increase in core neutron concentration in 32 seconds. The second problem is the tracking of a 10 percent step increase in the core density. These two reference signals were chosen because they bracket the range of desirable response times for multi-megawatt space reactor applications. The limitations imposed by the linearization assumptions prevent the consideration of larger reference signals. In all of the figures presented below, the curves are relative with respect to the initial values given in Table V. Case $G1$ is first isolated for general discussion purposes. The ramp case is shown in Fig. 16a and the step case in Fig. 16b. It is stressed that the results shown in Fig. 16 are favorable only as a result of many hours working on the trial and error eigenvalue selection process. Even though care was taken in this selection, some overshoot is shown in Fig. 16a at the 32 second bend in the reference signal. This overshoot is more severe when the controller is requested to track an abrupt step signal as shown in Fig. 16b. It is also noted that the core and reflector responses are nearly identical for both cases. This is primarily due to the linear and point nature in which their coupling is treated in the RK model. On the other hand, the precursor response significantly lags that of the neutron densities due to the decay laws which it must obey. However, the relative value reached upon stabilization are essentially the same. The core and coolant temperature response shown in Fig. 16 is caused by the changes in the coolant flow

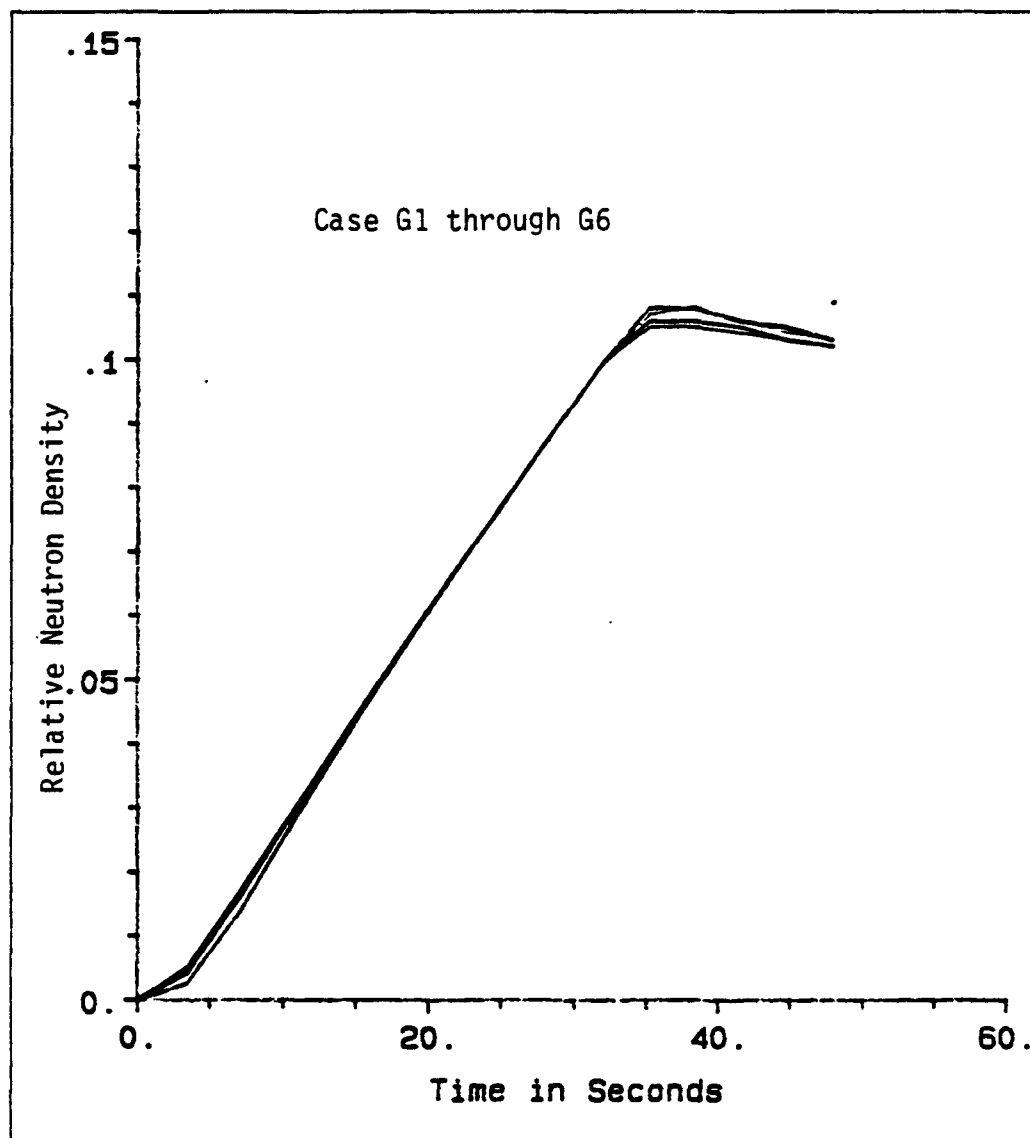


Fig. 17 Relative Neutron Response for Case G1 Through G6 for the Ramp Reference Signal

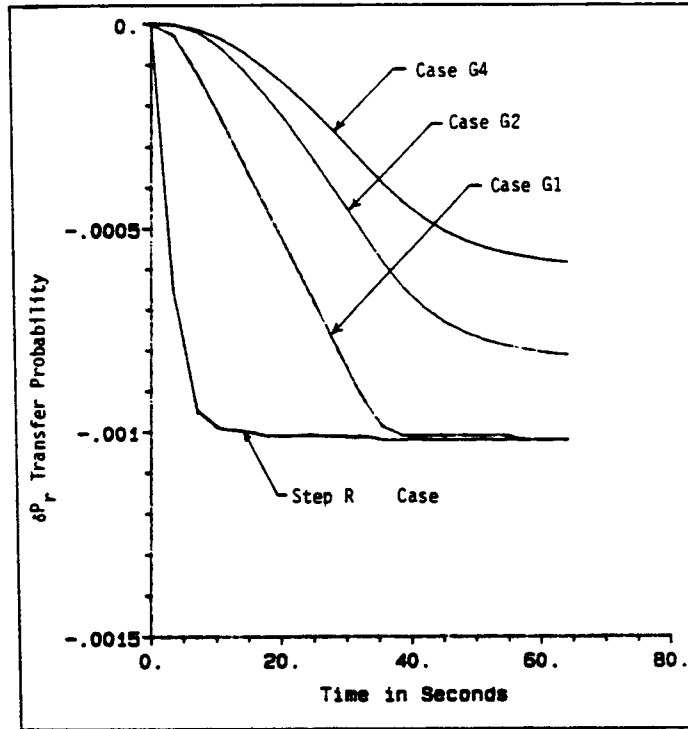


Fig. 18a Transfer Probability Inputs for the Ramp Cases G1, G2, G4, and the Step R Case

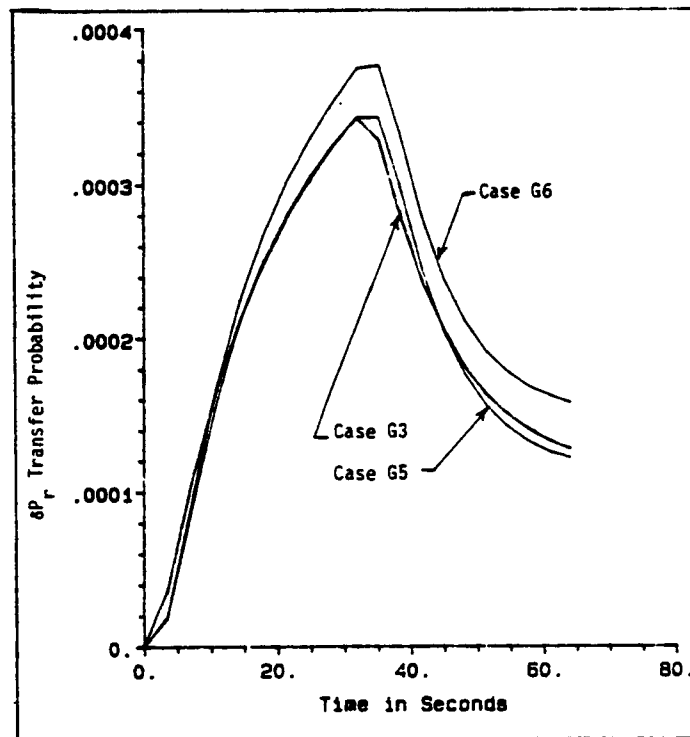


Fig. 18b Transfer Probability Inputs for the Ramp Cases G3, G5, and G6.

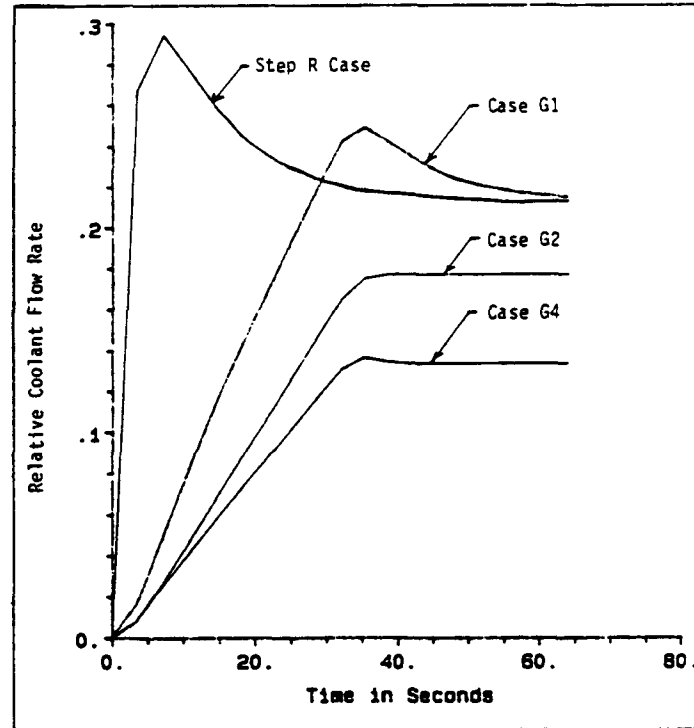


Fig. 19 a Coolant Flow Rate Input for the Ramp Cases G1, G2, G4, and the Step R Case

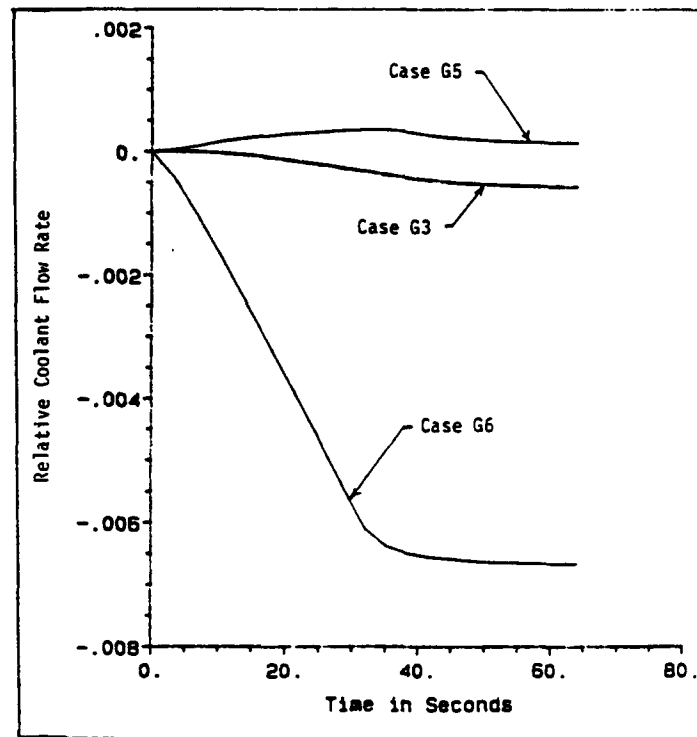


Fig. 19b Coolant Flow Rate Input for Ramp Cases G3, G5, and G6

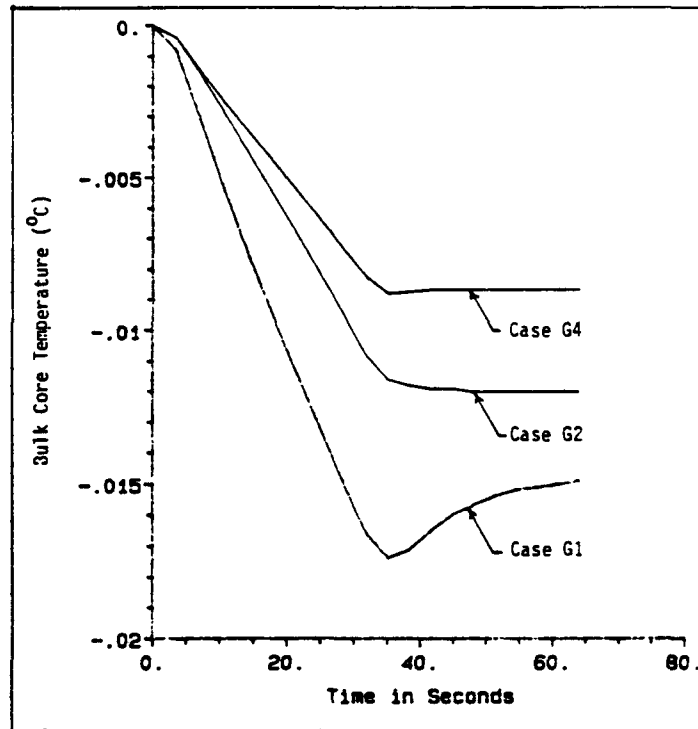


Fig. 20a Core Temperature for Ramp Cases G1, G2, and G4.

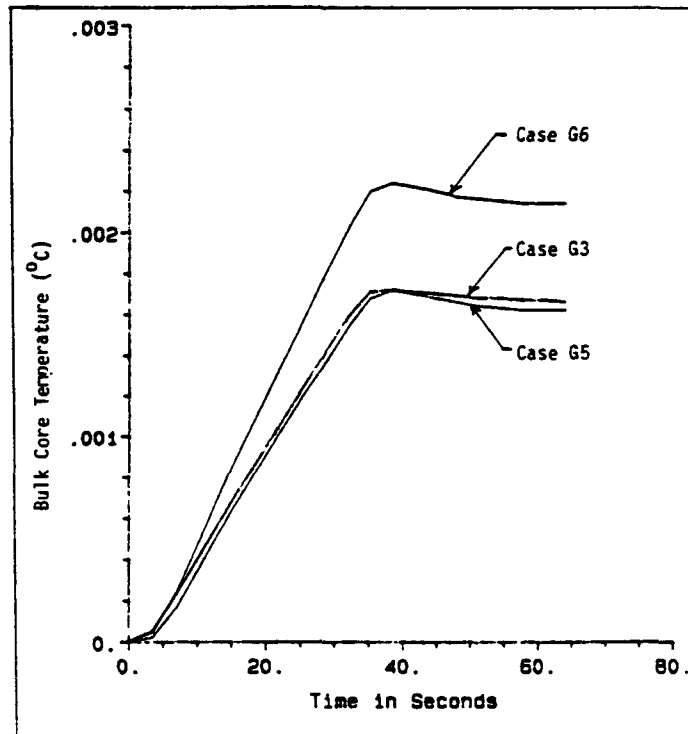


Fig. 20b Core Temperature for Ramp Cases G3, G5, and G6

rate. It is this temperature drop which provides the primary reactivity insertion for the procession of the transient. The fact that the controller feedback gains are not unique for the multivariable case suggests that the primary driving force could just as well be the reflector transfer probabilities. It turns out that three of the six G cases gave rise to P_r dominated transients and the other three were V_c dominated. Before proceeding to discuss the differences between these six cases, it is important to note that the output for each of the cases tracked the reference signal with near identical accuracy for the ramp tracking problem as shown in Fig. 17.

The differences resulting from the "selection" of G matrices primarily appear in the temperature response and the inputs. The δP_r reactivity inputs for each of the six cases are shown in Fig. 18a and Fig. 18b. The $\delta V_c/V_{co}$ inputs are given by Fig. 19a and Fig. 19b. Note that the results were grouped as shown in these three figures because each group exhibits the same basic control inputs with varying magnitudes. The input coolant flow in cases 1,2 and 4 assume the same basic shape of the reference signal. Rapid stability in these cases are achieved by an extremely large coolant flow rate change as seen in Fig. 19a. Also shown on this figure is the coolant flow increase for the step tracking problem. It is noted again that the shape is basically the same except it is squeezed into a shorter time frame. Cases 3, 5, and 6 exhibit much lower coolant flow rates and therefore in one sense solves the tracking problem more efficiently. It is particularly interesting to note that the P_r inputs for these cases, as seen in Fig. 18a, increases initially to instigate the ramp and then attains a negative slope beyond 32 seconds to stabilize the response. The differences in these two groups of input scenarios readily explain the different temperature responses shown in Fig. 20a and Fig. 20b. The results presented above indicate the need for more intelligent and physically based methods of selecting G

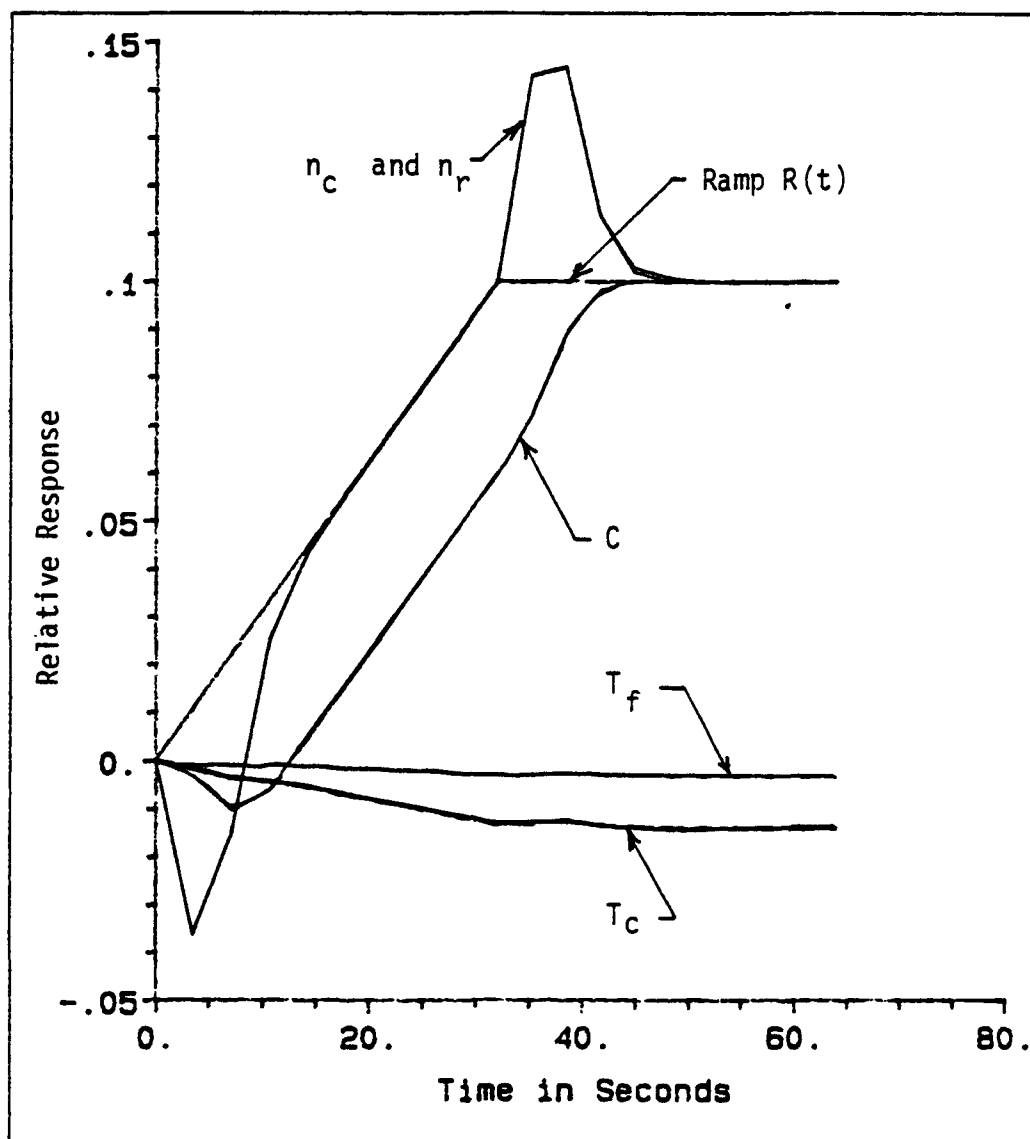


Fig. 21 State Feedback Solution for an Arbitrarily Chosen Eigenvalue Set

matrices if state feedback is to be used in future RK automatic control applications. However, even such methods would not guarantee the favorable transient response nor would it aid in the selection of an adequate eigenvalue set.

The disadvantage of the state feedback control approach discussed above is demonstrated in this final computation. Fig. 21 shows what can happen when an unfavorable eigenvalue set and G matrix happens to be chosen. Even though our system is stable and asymptotically tracks the desired reference signal, clearly the transient response of the system is unacceptable. The potential of this type of response occurring prohibited the use of RKSF with more than one delayed neutron precursor group. This is because when an additional precursor group is added, two more eigenvalues and two more columns of the G matrix must be specified. Further discouragement is given by the fact that the closed loop system grows by two states with the addition of each precursor group. On the other hand, the treatment of multiple precursor groups are readily treated in the modal control case. Additionally, the uncertainties associated with the selection of closed loop eigenvalues and parameterizing matrices are avoided in the modal control case as the results of the following section will indicate.

D. Modal Control Numerical Computations

The two tracking problems solved under state feedback control in the previous section are now solved using modal control as discussed in chapter V. The numerical computations are done with the RKM_MODAL code (see Appendix C). This program was written so that the desired eigenvalues and eigenvectors could be interactively altered until a desirable set of closed loop eigenvalues resulted. This was necessary because the primary disadvantage of the modal approach is that closed loop

column of the desired V_d attempts to force the first assignable eigenvalue to the bulk core temperature mode. Likewise, the second column attempts to capture the second eigenvalue for the coolant temperature mode. The "x"s in the third and fourth row of these vectors gives the modes the freedom to couple the core and coolant temperatures if needed in order to achieve the remaining zeroes in these vectors. The preservation of near zeroes in the first and fifth rows of the first two columns is important since the open loop results told us that the core and precursor eigenvalues were insensitive to other plant information. Finally the third column couples the tracking integrators to the core density mode. This was chosen because the instability is caused by the presence of the integrators. The remaining "x"s that appear were added in order to improve the achievement of the remaining zeroes which were believed to be more important.

The final step in the definition of the eigenmodes is the selection of the three eigenvalues. Three arbitrarily small values were chosen for each of the three modes. The selection was found to not have much effect on the transient behavior of the system within the range of 0 to -4.0 The values chosen were $\lambda_1 = -0.2$, $\lambda_2 = -2.0$, and $\lambda_3 = -4.0$.

The solution of the problem with the six delayed neutron precursor group model is a trivial extension in the modal controlled case. In this case, the fifth row of desirable eigenvector was repeated five times (once for each additional group). Since the feedback law is based on measurable output, the multiple precursor case does not involve the specification of additional columns of V_d nor additional assignable eigenvalues.

The solution of the ramp tracking problem with one and six delayed neutron groups is shown in Fig. 22a and Fig. 22b. The effect of the decoupling and the

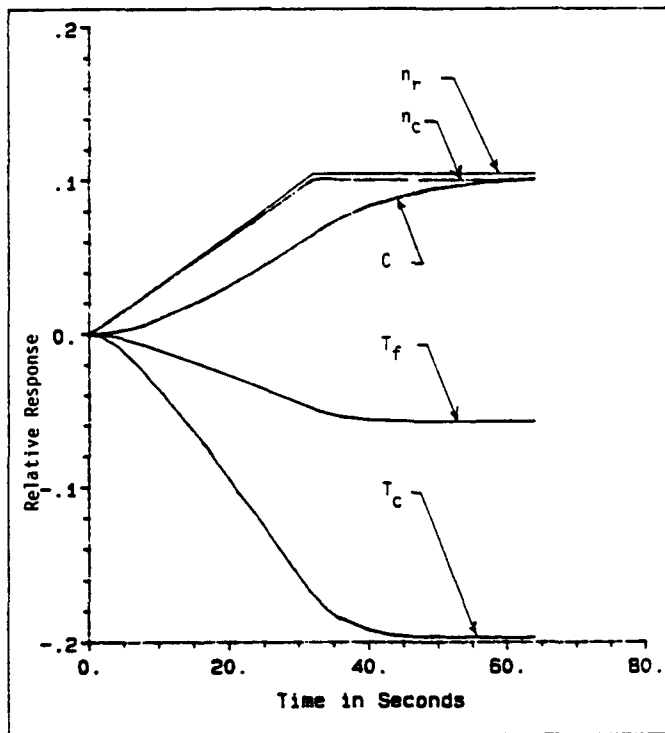


Fig. 22a Modal Solution for 1 Delayed Precursor Group, Ramp R Problem

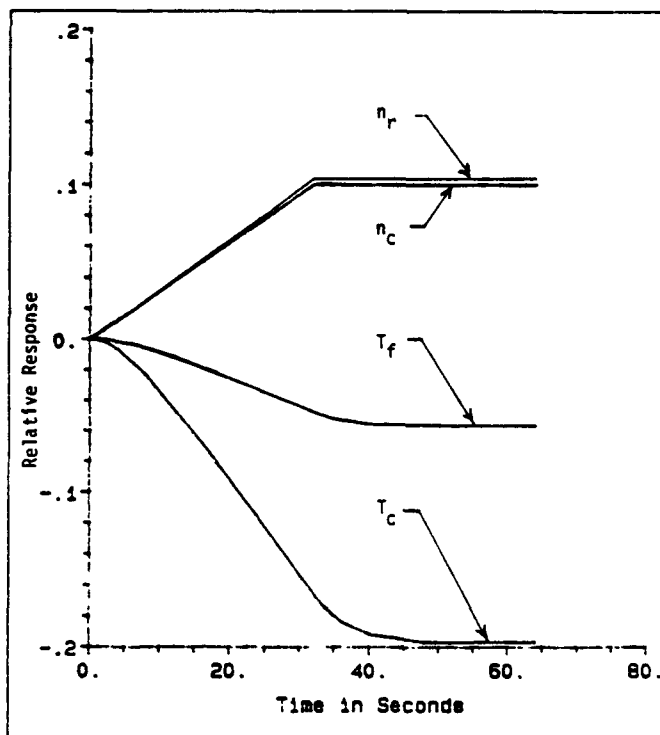


Fig. 22b Modal Solution for 6 Delayed Precursor Groups, Ramp R Problem

retainment of the certain natural eigenmodes in the system are evident in these figures. The overshoot is so small that it is not discernable on these figures. Again we see the time lag that the precursors must exhibit. The precursors are not plotted for the six group case to avoid confusion. The similarity between these two responses is explained by the fact that the controller was intentionally decoupled from the precursor states. The inputs computed from the modal output control law are plotted in Fig. 23a and Fig. 23b. Note that the penalty for the excellent tracking is a more demanding coolant flow rate increase. The solution to the step tracking problem is given in Fig. 24. The overshoot shown is much smaller than in the state feedback case. Again, Fig. 23a and Fig. 23b show the penalties for such accurate and fast signal tracking.

The final numerical computation is referred to as an intelligent manual control. Manually driven control drums or coolant flow rate could certainly not replicate the effect of either of the two control law inputs. The best one could ever hope to do would be linear approximations to these values. The inputs from the state feedback G1 case were broken into three portions and approximated by linear segments with a least square fitting algorithm with forced matching endpoints. The resulting response shown in Fig. 25 illustrates two important points. First, since the internal model principle depends on output feedback principles which physically can not be done, the immediate usefulness of the control schemes presented above are limited. However, as seen in Fig. 25, such computations give valuable insight into more intelligent ways of manually controlling the system. Secondly, assuming that the transducer needed to link the controlled output and the measureable output is negligible, then the intelligent manual input mode of operation is shown to be inferior to the true automatically control cases.

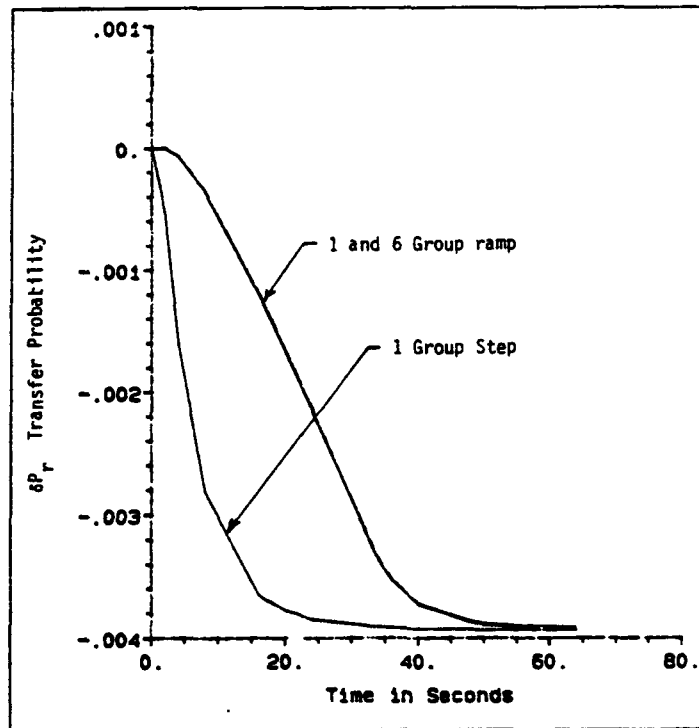


Fig. 23a Transfer Probability Inputs for the 1 and 6 Group Modal Solutions, Ramp and Step R Problems

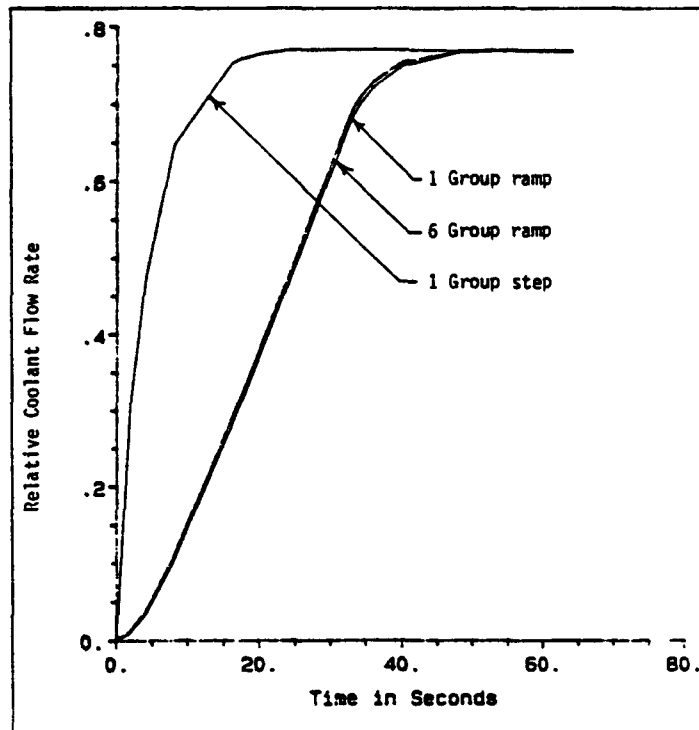


Fig. 23b Coolant Flow Rate Inputs for the 1 and 6 Group Modal Solutions, Ramp and Step R Problems

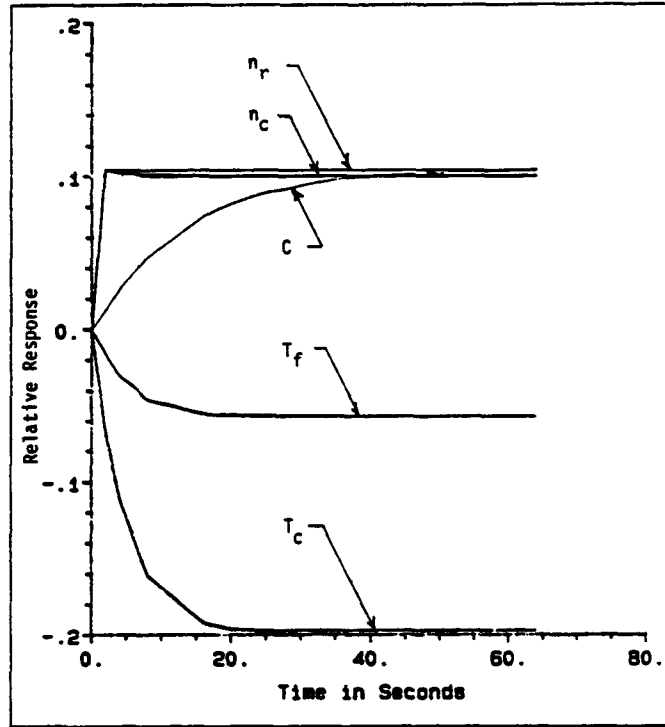


Fig. 24 Modal Solution for the 1 Delayed Precursor Group, Step R Problem

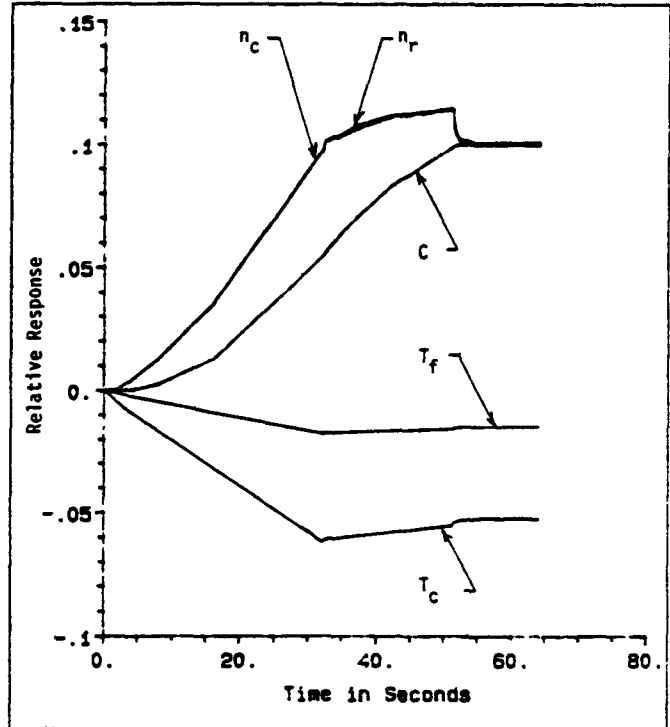


Fig. 25 Open Loop RK Solution Using State Feedback Determined Reactivity Inputs

CHAPTER VII

SUMMARY AND CONCLUSIONS

A. Summary

In chapter two, the RK model was conceived in order to model the effect of movable control drums external to the core in a point reactor environment. The reflected kinetics extension of the point kinetics equations has proven to be an attractive alternative to "equivalent bare point kinetics" in the analysis of space nuclear reactors. The RK structure was developed in order to treat control drums in a direct way. In chapter VI, it was shown that manipulation of the transfer probabilities do indeed drive the transient differently than the lumped insertion of equivalent reactivity amounts in the core. In chapter III, the RK equations were shown to limit properly to the point kinetics equation for a bare system. This limiting process was demonstrated in the time domain and in the comparison of the RK response function to that of the PK equations. Preliminary computations were performed in chapter III which identified the practical limit of the linearization assumptions. An analytic solution of the simplified RK case with constant reactivity inputs and no thermal effects was constructed by method of Laplace transforms. A general form of the step reactivity point kinetics analytic solution was also developed. The development of an extension to the ASH exponential operator method that allows for the solution of non constant sources was also presented in chapter III. This numerical technique was used to solve the linear RK equations as opposed to less efficient and less accurate finite difference methods.

In chapter IV the lumped parameter approach was utilized in order to model the transport of heat from the core region to the coolant fluid. Modelling of the

temperature feedback shutdown mechanisms were added to the RK equation set by using the bulk core and coolant temperatures and constant temperature feedback coefficients. The structure of the reactivity inputs in the RK model, unique from the PK approach, allowed for separation of core loss feedback effects and Doppler temperature feedback. The RKOPEN computer code shown in Appendix C was developed based on the modelling in chapter II and chapter IV. The first portion of chapter VI was dedicated to the testing of the validity of the RKOPEN code. These computations were also performed in order to draw conclusions regarding the importance of several system parameters and control inputs. The RKOPEN code provided the base structure for the two codes that implemented automatic control (RKSF and RKMODAL).

In chapter V, two approaches to automatic control were considered. The theory found in the literature was reviewed for purposes of continuity. Advantages and disadvantages of the state feedback controller and the modal controller as applied to the RK model were identified. The last portion of chapter VI was dedicated to the solution of two servo mechanism problems under both forms of automatic control. Conclusions regarding future use of automated control in space reactor studies are made based on these results.

B. Conclusions

The linearization assumptions were found to be valid for responses deviating less than 10 percent from their nominal values. The linearized RK equations are therefore concluded to be useful in the analysis of nearly all non military space nuclear applications. Furthermore, the use of this linear model in the development of the automatic controllers is viable. To consider some of the transients that may

be of interest in military space nuclear applications. nonlinear analysis must be considered.

The validity of the lumped parameter temperature model was verified in chapter VI where the open loop transient calculations resulting from the variation of physical system inputs qualitatively performed as expected. These computations also pointed out the importance of the ability to treat coolant flow rate as a control variable. The open loop computations also lead to the conclusion that the stability of the manual RK system is highly dependent on the sign and magnitude of the three temperature feedback coefficients. It was also discovered that zero temperature feedback coefficients render the RK system uncontrollable from an automatic control perspective. The implications that the temperature feedback coefficients have on system stability render them of primary concern in space reactor design. The transfer probability structure of reactivity input in the RK model played an important role in the realization of this conclusion.

The primary purpose of any scoping study is to determine which aspects of more detailed designs should be emphasized. The numerical calculations presented in chapter VI indicated that often plant eigenmodes are rigidly fixed in the stable complex plane. This rigidity provided the initial motivation for the consideration of modal control as an alternative to state feedback in the solution of the servo problem. In the RK model, two such rigid eigenmodes were found to be dependent only on the core and reflector lifetimes. This information proved useful in the implementation of the modal control algorithms. Computations involving elementary variations in the heat transfer coefficient and the macroscopic fission cross section reinforced the conclusion that negative temperature feedback has a strong influence on the stability of space nuclear systems. This was further tested by calculating a

pulse with the RKOPEN code. Even though the peak of the pulse indicates a rather extreme violation of the linearization assumptions, the RK model reproduced the temperature shutdown phenomena that in reality truly occurs for such intentionally induced severe transients.

The integration of automatic control principles and the RK model was readily accomplished as a result of the transfer probability reactivity input structure. The robust tracking of steps and ramps was demonstrated under the assumption of unity feedback. Comparisons between the state feedback controller and the modal controller indicated that modal control has the clear advantage in RK applications. The required placement of each system eigenvalue in the state feedback case proved to be wasteful due to the inherent stability exhibited by a nuclear system with negative temperature feedback. Furthermore, the lack of an intelligent eigenvalue selection process and the uncertainty associated with the selection of the G matrix limit the usefulness of state feedback control. One of the problems known to limit the use of modal control is the lack of a guarantee of system stability. This however is a mathematical limitation. In practice, it is trivial to check the stability of the closed loop system and alter the spectral assignment if an unstable configuration is found. This modelling incorporated into to RKMOMDAL code proved to offer a considerable advantages over the brute force state feedback approach.

On the more computational side of the results, the RKOPEN and the RK-MODAL equations were solved with the ASH methodology. The extension discussed in chapter III was implemented into ASH in order to handle linear source terms. The results shown in chapter VI demonstrate the validity of this treatment of non constant sources in the ASH methodology. The ASH method proved to provide a computational advantage over Runge Kutta in solving the dynamical equations in

these codes since the speed of the transient at early times dictates the use of extremely small Runge Kutta time steps. For the same accuracy, ASH was found to require less than a factor of 10 in CPU time for the calculations presented. However, the size of the closed loop matrices and the favorable magnitude of the eigenvalues made Runge Kutta the advantageous numerical method in the state feedback case.

Overall, the RK modelling approach is concluded to have a physical advantage over more implicit modelling of reflector controlled reactors. Additionally, the structure of the RK dynamical equations proved to permit the smooth transition to automatic control. The integration of modal control techniques and the RK dynamical model in conjunction with the computationally efficient ASH solution methodology has provided a useful tool for performing nuclear space reactor kinetic and control scoping studies. Furthermore the methodology and the computer analysis codes developed here have provided a solid foundation upon which more comprehensive kinetics and control models can be built. A few such suggestions of further work are given below.

C. Suggestions for Further Work

The most immediate work that can be done involves the use of actual space nuclear reactor input data. Preliminary criticality experiments will be necessary in order to identify such parameters as core lifetime and the initial transfer probability values. Another obstacle is the fact that much of this information is classified for purposes of national security.

From a numerical standpoint, the most immediate need for future study involves the treatment of the nonlinear RK equations. While valuable insight was found using the linearized system, the range in which the linearization assumptions

apply prevent the study of extremely deviate transients which could be of interest in military space nuclear applications. This study will require a numerical solution method that has the speed and accuracy of ASH, yet the flexibility of Runge Kutta. Further work in the nonlinear analysis should also include the study of linear automatic control applied to the nonlinear model. The study of the stability of the closed loop system will be necessary since stability in the nonlinear system in general cannot be not guaranteed by a controller that is derived from the linearized model.

The consideration of more advanced nuclear modelling would only strengthen the argument against the use of state feedback control. This is the result of the eventual growth of the order of the dynamical equations with added complexity and phenomonology. This fact in addition to the conclusions drawn from chapter VI results indicate that future study should concentrate on lower order control methods. The overall stable structure of the RK equations resulting from the physics of nuclear kinetics are well suited for the modal control method. Further work in the development of a servo compensator that does not depend on unity feedback is also suggested. Alternatives to the internal model principle will therefore need to be considered. Such a study would be needed in order to make the transition to the development of a controller that can actually be constructed for use in automated nuclear space missions. Finally, more sophisticated control laws could be considered which would put constraints on the physical range of the inputs, thereby eliminating the mathematical calculation of unphysical results.

With regard to the nuclear kinetics modelling, the use of Monte Carlo calculations are suggested in order to refine the numerical values of the transfer probabilities. Work has allready begun in this area with the KENO computer code. The

utilization of the information provided by KENO would require multiple energy group and a multiple core region modelling extensions. These considerations would also give rise to more accurate treatment of delayed and reflected neutrons in the system since they are typically at different energies than the average core neutron spectrum. Furthermore, the neutronic coupling of individual core regions to the control drums would bring forth the full advantage of the RK modelling approach.

REFERENCES

1. J. Angelo, Jr. and D. Buden. "Where We Are Today on Space Nuclear Systems," in *1st Symposium on Space Nuclear Power Systems*, Albuquerque, NM, Jan, 1984.
2. J. Duderstadt and L. Hamilton, *Nuclear Reactor Analysis*. John Wiley & Sons: New York, NY, 1976.
3. D. Hetrick. *Dynamics of Nuclear Reactors*. Univ. of Chicago: Chicago, IL, 1971.
4. W. Cadwell, A. Henry, and A. Vigilotti. "WIGGLE - A Program for the Solution of the Two-Group Space-Time Diffusion Equations in Slab Geometry," *Westinghouse Report : WAPD-TM 416*, 1964.
5. K. Ott and D. Meneley, "Accuracy of the Quasistatic Treatment of Spatial Reactor Kinetics," *Nucl. Sci. Eng.*, Vol. 36, pp. 402-411, 1969.
6. J. Devooght and E. Mund, "Generalized Quasistatic Method for Nuclear Reactor Space-Time Kinetics," *Nucl. Sci. Eng.*, Vol. 76, pp. 10-17, 1980.
7. A. Galati, "The Metastatic Method in Nuclear Reactor Core Kinetics Calculations," *Nucl. Sci. Eng.*, Vol. 37, pp. 30-40, 1969.
8. A. Wasserman, "A Simple Model for the Effect of Reflected Neutrons Upon Reactor Kinetics," *Quarterly Tech. Report Special Proj. IDO 16606*, July-Sept, 1959.
9. C. Chezem and W. Kohler, "Coupled Reactor Kinetics," in *National Topical Meeting of the American Nuclear Society*, College Station, TX, Jan, 1967.
10. M. Becker. "A Generalized Formulation of Point Nuclear Reactor Kinetics Equations." *Nucl. Sci. Eng.*, Vol. 31, pp. 458-464, 1958.
11. W. Kastenberg and P. Chambre, "On the Stability of Nonlinear Space-Dependent Reactor Kinetics," *Nucl. Sci. Eng.*, Vol. 31, pp. 67-79, 1968.
12. E. Fuller et al, "Weighted-Residual Methods in Space-Dependent Reactor Dynamics." *Nucl. Sci. Eng.*, Vol. 40, pp. 206-223, 1970.
13. E. Rumble and W. Kastenberg, "On the Application of Eigenfunction Expansions to Problems in Nonlinear Space-Time Reactor Dynamics," *Nucl. Sci. Eng.*, Vol. 49, pp. 172-187, 1972.
14. S. Kaplan, "The Property of Finality and the Analysis of Problems in Reactor Space-Time Kinetics by Various Modal Expansions," *Nucl. Sci. Eng.*, Vol. 9, pp. 357-361, 1961.

15. R. Alcouffe and R. Albrecht, "A Generalization of the Finite Difference Approximation Method with an Application to Space-Time Nuclear Reactor Kinetics," *Nucl. Sci. Eng.*, Vol. 39, pp. 1-13, 1970.
16. A. Wight, K. Hansen, and D. Ferguson. "Application of Alternating-Direction Implicit Methods to the Space-Dependent Kinetics Equations," *Nucl. Sci. Eng.*, Vol. 44, pp. 239-251, 1971.
17. W. Stacey, Jr., *Space-Time Nuclear Reactor Kinetics*. Academic: New York, NY, 1969.
18. C. Chen, *Linear System Theory and Design*. CBS College Publishing: New York, NY. 1984.
19. D. Owens, "Controllability and Observability Considerations in the Design of Sector Control Configurations for Cylindrical Nuclear Power Reactors," *DOE report AEEW-R-829*, 1973.
20. L. Weaver and R. Vanasse, "State Variable Feedback Control of Multiregion Reactors," *Nucl. Sci. Eng.*, Vol. 29, pp. 264-271, 1967.
21. T. Oohuori, "Time-Optimal Power-Change Control for Coupled-Core Reactors," *Nucl. Sci. Tech.*, Vol. 19 n12, pp. 1057-1060, 1982.
22. A. Christie, etal, "Control of Xenon Instabilities in Large Pressurized Water Reactors," *DOE report WCAP-3680-23*, 1969.
23. W. Hanke, "A Method for Solving the Xenon Oscillation Control Problem," *Nucl. Sci. Eng.*, Vol. 72, pp. 265-272. 1979.
24. D. Wiberg, "Optimal Control of Nuclear Reactor Systems," in *Advances in Control Systems*. C. Leondes, ED. Wiley & Sons: New York, NY, pp. 301-388, 1967.
25. M. Tsuji and Y. Ogawa, "Identification of a Pressurized Water Reactor Power Plant by an Auto-Regressive Method," *Hokkaido Daigaku Kogakubu Kenkyu Hokoku*, Vol. 104, pp. 45-56, 1981.
26. D. Cherchas and C. Mewdell, "Control Algorithm for Reactor Spatial Control During Nuclear Station Load Cycling," *Dyn. Syst. Meas. Control Trans. ASME*, Vol. 100 n3. pp. 219-226, 1978.
27. L. Miller. R. Cochran. and J. Howze, "Nuclear Reactor Control System Design With Sensor Failure." *Nucl. Tech.*, Vol. 36, pp. 93-105, 1977.
28. A. Andry, Jr., E. Shaprio, and J. Chung, "Eigenstructure Assignment for Linear Systems," *IEEE Trans. on Aero. Elec. Sys.*, Vol. AES-19 n5, pp. 711-729. 1983.
29. C. Lee, "ASH: An Isotope Transmutation Program Using Matrix Operators," in *Proc. of Int. Conf. on Nucl. Waste Transmutation*, Austin, Tx, 1980.

30. C. Lee and B. Wilson, "Time Dependent Diffusion and Decay Using a Conservation Variational Principle," *Trans. Am. Nucl. Soc.*, Vol. 39, pp. 961-962, 1981.
31. K. Washington, "High Order Numerical Solutions to Time-Dependent Advection Diffusion Problems," M.S. Thesis, Texas A&M University, 1983.
32. G. Schlapper, "A Time Optimal Control Study of a Doubly-Reflected Nuclear Reactor," M.S. Thesis, University of Missouri, 1970.
33. C. Lee and K. Washington, "Comparison of Finite-Difference and Variational Solutions to Advection-Diffusion Problems," *Ann. Nucl. Energy*, Vol. 11, pp. 629-646, 1984.
34. G. Keepin, *Physics of Nuclear Kinetics*. Addison-Wesley: Reading, MA, 1965.
35. J. Schumacher, "Compensator Synthesis Using (C,A,B) Pairs," *IEEE Trans. Aut. Control*, Vol. AC-25, pp. 1133-1138, 1979.
36. D. Ohm, J. Howze, and S. Bhattacharyya, "Structural Synthesis of Multivariable Controllers," *Automatica*, Vol. 21, pp. 35-55, 1985.
37. S. Bhattacharyya and E. de Souza, "Pole Assignment via Sylvester's Equation," *Systems & Control Letters 1*, Vol. 4, pp. 261-263, 1982.
38. L. Keel. "Pole Placement Design for Linear Multivariable Control Systems," M.S. Thesis. Texas A&M University, 1982.
39. G. Golub, S. Nash, and C. Van Loan, "A Hessenberg-Schur Method for the Problem $AX - XB = C$," *IEEE Trans. Aut. Control*, Vol. AC-24, pp. 909-913, 1979.
40. B. Moore, "On the Flexibility Offered by State Feedback in Multivariable Systems Beyond Closed Loop Eigenvalue Assignment," *IEEE Trans. Aut. Control*, Vol. AC-21, pp. 689-692, 1976.

APPENDIX A

RK PLANT MATRIX STRUCTURE

Open Loop No Temperature Feedback Matrices

 A_p

$(P_{CO} (1-B)K_O - 1)\lambda_C$	$P_{1O}\lambda_{r1} \dots P_{NO}\lambda_{rN}$	$P_{CO}\lambda_1 \dots P_{CO}\lambda_I$
$(P_{F1} (1-P_{CO}) (1-B)K_O)\lambda_C$	$-\lambda_{r1}$	$P_{F1} (1-P_{CO})\lambda_1 \dots P_{F1} (1-P_{CO})\lambda_I$
\vdots	\ddots	\vdots
\vdots	$0 \dots$	\vdots
$(P_{FN} (1-P_{CO}) (1-B)K_O)\lambda_C$	$-\lambda_{rN}$	$P_{FN} (1-P_{CO})\lambda_1 \dots P_{FN} (1-P_{CO})\lambda_I$
$B_1 K_O \lambda_C$	0	$-\lambda_1$
\vdots	\vdots	\ddots
\vdots	\vdots	$0 \dots 0$
$B_I K_O \lambda_C$	0	$-\lambda_I$

 B_p

$P_{CO} (1-B)n_{CO}\lambda_C$	$K_O n_{CO}\lambda_C$	$P_{F1} (1-P_{CO})K_O n_{CO}\lambda_C \dots P_{FN} (1-P_{CO})K_O n_{CO}\lambda_C$
$(P_{F1} (1-P_{CO})(1-B)n_{CO})\lambda_C$	$-P_{F1}K_O n_{CO}\lambda_C$	$0 \dots 0$
\vdots	\vdots	\vdots
\vdots	\vdots	\vdots
$(P_{FN} (1-P_{CO})(1-B)n_{CO})\lambda_C$	$-P_{FN}K_O n_{CO}\lambda_C$	$0 \dots 0$
$B_1 n_{CO}\lambda_C$	0	$0 \dots 0$
\vdots	\vdots	\vdots
\vdots	\vdots	\vdots
$B_I n_{CO}\lambda_C$	0	$0 \dots 0$

One Reflector, One Precursor Group Matrices With Temperature Feedback

 A_p

$$\begin{array}{ccccc}
 [P_{co}(1-\beta)K_o-1]\lambda_c & \lambda_r P_{ro} & -[P_{co}(1-\beta)\alpha_r + K_o a_e] n_{co} \lambda_c & -P_{co}(1-\beta) a_c n_{co} \lambda_c & P_{co} \lambda_i \\
 P_{fr}(1-P_{co})(1-\beta)K_o \lambda_c & -\lambda_r & -[(1-P_{co})(1-\beta)\alpha_r + K_o a_e] P_{fr} n_{co} \lambda_c & -(1-P_{co})(1-\beta) a_c P_{fr} n_{co} \lambda_c & 0 \\
 \chi_r / \theta_r c_p r & 0 & -1/\tau_r & 1/\tau_c & 0 \\
 0 & 0 & 1/\tau_c & -1/\tau_c - 1/\tau_L & 0 \\
 \beta_i K_o \lambda_c & 0 & -\beta_i \alpha_r n_{co} \lambda_c & -\beta_i a_c n_{co} \lambda_c & -\lambda_i
 \end{array}$$

 B_p

$$\begin{array}{cc}
 P_{fr}(1-P_{co})K_o n_{co} \lambda_c & 0 \\
 0 & 0 \\
 0 & 0 \\
 0 & (T_w - T_{co})/\tau_L \\
 0 & 0
 \end{array}$$

State Feedback Closed Loop Matrices

 A_{c1}

$$\begin{array}{ccccccc}
 A_{p1} + B_{p1}(F_{p1} + F_{p3}L_u) & A_{p2} + B_{p1}F_{p2} & A_{p3} & 0 & B_{p1}F_{p3} & B_{p1}F_{r1} & B_{p1}F_{r2} \\
 A_{p4} + B_{p2}(F_{p1} + F_{p3}L_u) & A_{p5} + B_{p2}F_{p2} & A_{p6} & 0 & B_{p2}F_{p3} & B_{p2}F_{r1} & B_{p2}F_{r2} \\
 A_{p7} + B_{p3}(F_{p1} + F_{p3}L_u) & A_{p8} + B_{p3}F_{p2} & A_{p9} & 0 & B_{p3}F_{p3} & B_{p3}F_{r1} & B_{p3}F_{r2} \\
 D_{c1} + B_{c1}(F_{p1} + F_{p3}L_u) & B_{c1}F_{p2} & 0 & A_{c1} & A_{c2} + B_{c1}F_{p3} & B_{c1}F_{r1} & B_{c1}F_{r2} \\
 D_{c2} + B_{c2}(F_{p1} + F_{p3}L_u) & B_{c2}F_{p2} & 0 & A_{c3} & A_{c4} + B_{c1}F_{p3} & B_{c2}F_{r1} & B_{c2}F_{r2} \\
 -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0
 \end{array}$$

 B_{c1}

$$\begin{array}{c}
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 1 \\
 0
 \end{array}$$

APPENDIX B

ASH METHODOLOGY

A. INTRODUCTION

Many problems of interest in science and engineering can be expressed as a system of coupled linear time invariant equations. Such a system can be written as:

$$\frac{d\phi}{dt} = A \phi + S \quad (B.1)$$

where A and S are constant over the time interval [0,t]. Conventional techniques for numerically solving equation B.1 are based on finite difference techniques such as explicit and implicit Runge Kutta methods. However, if A is stiff or ill conditioned and the solution ϕ is desired at long times, t, then the necessary small time steps required for accuracy can result in a non-feasible solution time. An alternate approach is presented here based on the simple analytic solution to equation B.1. This method uses an exponential matrix operator and will be referred to as the ASH method. The ASH method has proven to be near analytic accuracy and computationally advantageous to use.

B. THEORY

Under the assumption that A and S are constant over [0,t], the analytic solution is readily verified to be given by

$$\phi(t) = \{e^{At}\} \phi(0) + A^{-1} (\{e^{At}\} - I) S. \quad (B.2)$$

The matrix operator, D(C), is now defined as

$$D(C) = C^{-1} (e^C - I). \quad (B.3)$$

If we let $C = At$ then equation B.2 reduces to:

$$\phi(t) = [I + C D(C)]\phi(0) + t D(C) S \quad (B.4)$$

Several things are worth noting about equation B.4. First, if we are given an autonomous system ($S = 0$) then the operator D(C) is not required; rather only the matrix operator $[I + C D(C)]$. Second, the matrix operator D(C) represents the definition of the $\{e^{At}\}$ matrix since exponentiating non diagonal matrices is not a straightforward calculation.

Finally, if the $D(C)$ matrix can be found then the problem is solved exactly by equation B.4. $D(C)$ is found by using the series representation of e^C .

$$e^C = \sum_0^{\infty} \frac{C^n}{n!} = I + \sum_1^{\infty} \frac{C^n}{n!} = I + C \sum_0^{\infty} \frac{C^n}{(n+1)!} \quad (\text{B.5})$$

Equation B.5 is rearranged (subtract I and multiply by C^{-1}) to give,

$$D(C) = \sum_0^{\infty} \frac{C^n}{(n+1)!} \quad (\text{B.6})$$

The series expression given by equation B.6 would prove computationally difficult to evaluate unless the eigenvalues of the C matrix were bounded by unity. This problem is solved by defining a new matrix, H , which is a scaled down version of C . Consider the following definition

$$H = 2^{-p}C, \quad (\text{B.7})$$

where p is chosen such that the Euclidean norm of H is less than $1/2$ (i.e., $\|H\| < 1/2$). The value of p that satisfies this criterion is given by the following relationship:

$$p > 1 + \frac{\ln(t) + 0.5 \ln[\sum_{ij} A_{ij}^2]}{\ln(2)} \quad (\text{B.8})$$

Now we seek to find a computational expression for the $D(H)$ operator. This is easily done by terminating equation B.6 at M terms such that the next term ($M+1$) gives a negligible contribution to the series. This is a valid convergence criterion because the exponential sequence is monotonically decreasing.

$$D(H) = \sum_0^M \frac{H^n}{(n+1)!} \quad (\text{B.9})$$

where

$$\frac{\|H\|^{M+1}}{(M+2)!} < \frac{1}{2^{M+1} (M+2)!} < \epsilon \quad (\text{B.10})$$

The value M must be determined dynamically for some specified ϵ from equation B.10 prior to the computation of the series. This is required because the series is performed backwards as illustrated by the simple example below.

$$f(x) = 1 + x + x^2 + x^3 + x^4 = 1 + x(1 + x(1 + x(1 + x))) \quad (\text{B.11})$$

Once the $D(H)$ matrix operator is formed, the $D(C)$ and the $[I + C D(C)]$ operators are formed by applying the two recursion relationships below:

$$D(2^n H) = D(2^{n-1} H) [I + 0.5(2^{n-1} H) D(2^{n-1} H)] \quad (\text{B.12})$$

$$[I + (2^n H) D(2^n H)] = [I + (2^{n-1} H) D(2^{n-1} H)]^2 \quad (\text{B.13})$$

Clearly equation B.12 and B.13 must be applied 'p' times until we have scaled back up to $D(C)$ and $[I + C D(C)]$ respectively. These recursion relations are proved in the following section.

C. PROOF OF RECURSION RELATIONS

Equation B.12 is first proved by transfinite induction. When $p=0$, then clearly $C = H$ and $D(C) = D(H)$. When $p=1$, then $C = 2H$ and $D(C) = D(2H) = (2H)^{-1}(e^{2H} - I)$. This factors into:

$$D(2H) = H^{-1}(e^H - I) * 0.5(e^H + I) = D(H) * 0.5(e^H + I) \quad (\text{B.14})$$

Substituting $e^H = H D(H) + I$ gives

$$0.5(e^H + I) = 0.5(H D(H) + I + I) = I + 0.5 H D(H). \quad (\text{B.15})$$

Therefore we have,

$$D(2H) = D(H) [I + 0.5 H D(H)] . \quad (\text{B.16})$$

If this is true for $p=0$ and $p=1$ then by induction it is true for all values of n ($0 < n < p$).

$$D(2^n H) = D(2^{n-1} H) [I + 0.5 (2^{n-1} H) D(2^{n-1} H)] \quad (\text{B.17})$$

The proof of equation B.13 follows in a similar manner. If $p=1$ then, $C = 2H$ and $[I + C D(C)] = [I + 2H D(2H)]$. Substituting in the results from the previous proof at $p=1$ for $D(2H)$ gives

$$\begin{aligned}
 I + C D(C) &= I + 2H D(H) [I + 0.5 H D(H)] \\
 &= I + 2H D(H) + H D(H) H D(H) \\
 &= [I + H D(H)]^2 .
 \end{aligned}
 \tag{B.18}$$

. Again by induction we may deduce in general

$$I + (2^n H) D(2^n H) = [I + (2^{n-1} H) D(2^{n-1} H)]^2 .
 \tag{B.19}$$

APPENDIX C

CODE LISTINGS

```

PROGRAM RKO
C-
C- Open loop analysis program
C-
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R4
      CHARACTER FILEI*8
      DIMENSION Z(2001)
      DATA IZMAX /2000/
10    CONTINUE
      CALL ASK('File name extension',4,R4,R8,I,FILEI)
      OPEN(UNIT=1,FILE=FILEI//'.INP',STATUS='OLD',ERR=10)
      OPEN(UNIT=2,FILE=FILEI//'.OUT',STATUS='NEW',ERR=10)
      OPEN(UNIT=3,FILE=FILEI//'.PRN',STATUS='NEW',ERR=10)
15    CONTINUE
      READ (1,101,ERR=99,END=99) NREF,NGRP,NPD,NSOL
      IF (NREF.LT.0 .OR. NGRP.LT.0 .OR. NPD.LE.0) GO TO 99
C-
C- Calculate the size of the matrix from the number of reflectors, etc
C- the 3 comes from the core equation plus the two temperatures
C- The additional 1 comes from the simulated time variable
C-
      NC = NREF + NGRP + 3 + 1
      NADD = NC*NC
      NPNS = NPD*NSOL
      N1 = 1
      N2 = N1 + NADD
      N3 = N2 + NADD
      N4 = N3 + NADD
      N5 = N4 + NADD
      N6 = N5 + NADD
      N7 = N6 + NADD
      N8 = N7 + NC*NPNS
      N9 = N8 + NC
      N10 = N9 + NC
      N11 = N10 + NPNS
      N12 = N11 + NC
      N13 = N12 + NC
      NLAST = N13 + NC
      NVECT = NLAST - 1
C-
      WRITE (*,102) NVECT
      IF (NVECT.GT.IZMAX .OR. NGRP.GT.6 .OR. NREF.GT.9)
1    GO TO 99
      DO 30 I=1,NVECT
        Z(I) = 0.0D0
30    CONTINUE
C-
C- Call the main routine
C-
      CALL GORES(NREF,NGRP,NC,NPD,NSOL,NPNS,
1      Z(N1),Z(N2),Z(N3),Z(N4),Z(N5),Z(N6),Z(N7),
2      Z(N8),Z(N9),Z(N10),Z(N11),Z(N12),Z(N13))
      GO TO 15
99    CONTINUE
      CLOSE(UNIT=1)

```



```

        CLOSE(UNIT=2)
        CLOSE(UNIT=3)
        STOP
101  FORMAT(5I4)
102  FORMAT(/,' This calculation requires, ',I6,
      1  ' Array elements',/)
      END
      SUBROUTINE GORES(NREF,NGRP,NC,NPD,NSOL,NPNS,
      1  A,B,C,D,E,F,POUT,P,SV,TIMES,
      2  WR,WI,SCALE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R4
      CHARACTER ANS*1,TITLE*70
      COMMON/INFO/ SORCO(20,10),SORC1(20,10)
      DIMENSION A(NC,NC),B(NC,NC),C(NC,NC),D(NC,NC),E(NC,NC),
      1  F(NC,NC),POUT(NC,NPNS),P(NC),SV(NC),TIMES(NPNS),
      2  WR(NC),WI(NC),SCALE(NC),INTG(20)
C-
C- See RESM and RESC code for documentation. Also see SOLVE2
C-
      NDIM = NC
      NCM1 = NC - 1
      NB = NREF + 2
      NSV = 1
C-
C- Read in the title and some preliminary integer flags
C-
      READ (1,103,ERR=99,END=99) TITLE
      IF (TITLE(1:3) .EQ. 'END') GO TO 99
      READ (1,102,ERR=99) NEPS,MMAX,IPRT,IEIG,IPLT
      WRITE (2,214) TITLE
      WRITE (*,214) TITLE
C-
C- Input the source store temporarily in C(J,I)
C- First entry is K ext next are dPr for r=1,NGRP followed by coolant
C-
      WRITE (2,210)
      WRITE (*,210)
      DO 16 INSOL=1,NSOL
        READ (1,101) TIMES(NPD*INSOL)
        WRITE (2,212) TIMES(NPD*INSOL)
        WRITE (*,212) TIMES(NPD*INSOL)
        DO 15 J=1,NB
          READ (1,101) C(J,INSOL),D(J,INSOL)
          WRITE (2,211) J,C(J,INSOL),D(J,INSOL)
          WRITE (*,211) J,C(J,INSOL),D(J,INSOL)
15      CONTINUE
16      CONTINUE
C-
C- Get the A matrix and B matrices
C- B ash used as temp space for B source
C-
      CALL MAKEAB2(NDIM,NREF,NGRP,NC,NB,A,B,IPRT)
C-
C- Initialize Source to zero and fix up t variable initial condition
C-
      DO 11 I=1,NCM1
        DO 10 J=1,NSOL
          SORCO(I,J) = 0.0D0
          SORC1(I,J) = 0.0D0
10      CONTINUE
        P(I) = 0.0D0
11      CONTINUE
        P(NC) = 1.0D0
C-
C- Multiply B * U to get source vector
C-
      DO 31 INSOL=1,NSOL
        DO 30 I=1,NCM1

```

```

                DO 20 J=1,NB
                  SORCO(I,INSOL) = SORCO(I,INSOL) + B(I,J)*C(J,INSOL)
                  SORC1(I,INSOL) = SORC1(I,INSOL) + B(I,J)*D(J,INSOL)
20              CONTINUE
30              CONTINUE
31              CONTINUE
C-
C- Set A and SV for the first loop (not really needed)
C-
                INSOL = 1
                CALL ASV(A,SV,INSOL,NDIM,NC)
C-
C- Print A and B matrices (not including t variable)
C-
                IF (IPRT.GE.3) THEN
                  CALL PRMAT(NDIM,NCM1,NCM1,A,'A Plant matrix')
                  CALL PRMAT(NDIM,NCM1,NB,B,'B Plant matrix')
                END IF
C-
C- Find the eigenvalues of the matrix (not including t variable)
C-
                IF (IEIG.GT.0) THEN
                  CALL EQUAL(A,F,NDIM,NCM1)
                  CALL RG(NDIM,NCM1,F,WR,WI,E,INTG,SCALE,IERR,0)
                  IF (IERR.NE.0) THEN
                    WRITE (2,206)
                    IF (IPRT.GE.1) WRITE (*,206)
                  ELSE
                    IF (IPRT.GE.1) WRITE (*,208)
                    WRITE (2,208)
                    DO 45 I=1,NCM1
                      IF (IPRT.GE.1) WRITE (*,209) WR(I),WI(I)
                      WRITE (2,209) WR(I),WI(I)
45                  CONTINUE
                    END IF
                    IF (IPRT.GE.1) THEN
                      CALL ASK('Proceed with calculation [Y]',4,R4,R8,I,ANS)
                      IF (ANS.EQ.'N' .OR. ANS.EQ.'n') RETURN
                    END IF
                END IF
C-
C- Get solution using ASH SOLVER
C-
                IF (IPRT.GE.1) WRITE (*,*) 'Calling SOLVER for answer'
                CALL SOLVE2(A,B,C,D,E,F,NDIM,NC,NSV,SV,P,TIMES,
1                  POUT,NPD,NSOL,NPNS,NEPS,MMAX,IPRT)
C-
C- Write the plot output file, note that IPLOT dictates
C- how many of the states are included in the PLOT.INP file
C-
                IF (IPLOT .GT. 0) THEN
                  IF (IPLOT .GT. NCM1) IPLOT = NCM1
                  WRITE (3,102) NPNS,IPLOT
                  DO 70 I=1,NPNS
                    WRITE (3,201) TIMES(I),(POUT(J,I),J=1,IPLOT)
70                  CONTINUE
                END IF
C-
C- Output results
C-
                IL = 0
40              CONTINUE
                JB = 5*IL+1
                JE = JB + 4
                IF (JE .GE. NPNS) JE = NPNS
                WRITE (2,202) (TIMES(J),J=JB,JE)
                WRITE (2,213)
                WRITE (2,203) (POUT(1,J),J=JB,JE)
                DO 50 I=1,NREF

```

```

        WRITE (2,204) I,(POUT(I+1,J),J=JB,JE)
50  CONTINUE
    WRITE (2,207) 'F',(POUT(NREF+2,J),J=JB,JE)
    WRITE (2,207) 'C',(POUT(NREF+3,J),J=JB,JE)
    DO 60 I=1,NGRP
        WRITE (2,205) I,(POUT(I+NREF+3,J),J=JB,JE)
60  CONTINUE
    IL = IL + 1
    IF (JE .LT. NPNS) GO TO 40
C-
C- Thats all
C-
    RETURN
99  CONTINUE
    CLOSE(UNIT=1)
    CLOSE(UNIT=2)
    CLOSE(UNIT=3)
    STOP
101 FORMAT(8D10.4)
102 FORMAT(6I4)
103 FORMAT(A)
201 FORMAT(8(1X,E9.3))
202 FORMAT(///,T2,'Time',T10,5(1PD12.4))
203 FORMAT(T2,'Nc',T10,5(1PD12.4))
204 FORMAT(T2,'Nr(',I1,')',T10,5(1PD12.4))
205 FORMAT(T2,'C(',I1,')',T10,5(1PD12.4))
206 FORMAT(//,T2,'*** Error in RG eigenvalue finder ***',//)
207 FORMAT(T2,'T',A1,T10,5(1PD12.4))
208 FORMAT(//,T5,'System eigenvalues ',/,T5,18('-'),//)
209 FORMAT(T5,1PD12.4,' + j(',1PD12.4,')')
210 FORMAT(/,T5,'Inputs to the open loop system',/,T5,35('-'),/)
211 FORMAT(T5,'U(',I2,') = ',1PD12.4,' + ',1PD12.4,' t')
212 FORMAT(T5,'Destination time = ',1PD12.4)
213 FORMAT(T2,75('-'),//)
214 FORMAT(//,1X,A,//)
    END
    SUBROUTINE MAKEAB2(NDIM,NREF,NGRP,NC,NB,A,B,IPRT)
    IMPLICIT REAL*8 (A-H,O-Z)
    DIMENSION A(NDIM,NDIM),B(NDIM,NB),
1     BETA(6),XL(6),PFR(9),XLR(9),PRO(9)
    DATA CONVF1,CONVF2 / 1.4022D-06,6.7050D-05 /
C-
C- The intent of this subroutine is to read in all pertinent
C- information and then calculate the system A and B matrices
C- for an averaged core region and coolant channel for the
C- reflected point kinetics model developed for
C- the air force research contract.
C-
C- This version was modified in that the precursor states were
C- placed at the end in order to simplify the process of designing
C- a minimum order controller for the unreachable concentrations
C-
C- Also the external K eff was reinserted into the B matrix
C-
C- Initialize matrix to zero
C-
    DO 20 I=1,NC
        DO 10 J=1,NC
            A(J,I) = 0.0D0
10     CONTINUE
        DO 11 J=1,NB
            B(J,I) = 0.0D0
11     CONTINUE
20  CONTINUE
C-
C- Read in and write out the problem information
C-
C- Core
C-

```

```

C- PHIO = Initial core flux = XNCO * VEL
C- XNCO = Initial neutron distribution
C- XKINF K - Infinity for the core initially
C- XLC Inverse of core lifetime
C- PCO Initial prob of neutrons staying in the core
C- ENERGY Energy of the neutron spectrum in Kev
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Core input'
      READ (1,200) XNCO,XLC,PCO,ENERGY
      IF (ENERGY .LE. 0.0DO) ENERGY = 10.0DO
C-
C- Reflectors
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Reflector input'
      SUMR = 0.0DO
      IF (NREF.EQ.0) THEN
        XKINF = 1.0DO/PCO
        WRITE (2,201) XNCO,XKINF,XLC,PCO,ENERGY
        WRITE (2,211)
      ELSE
        DO 30 N=1,NREF
          READ (1,200) PFR(N),XLR(N),PRO(N)
          SUMR = SUMR + PFR(N)*PRO(N)
30      CONTINUE
        XKINF = 1.0DO/(PCO + (1.0DO-PCO)*SUMR)
        WRITE (2,201) XNCO,XKINF,XLC,PCO,ENERGY
        WRITE (2,202) NREF
        DO 31 N=1,NREF
          XNRO = PFR(N)*(1.0DO-PCO)*XKINF*XNCO*XLC/XLR(N)
          WRITE (2,203) XNRO,PFR(N),XLR(N),PRO(N)
31      CONTINUE
      END IF
C-
C- Precursors
C-
C- BETA Delayed neutron fraction for group i
C- XL The decay constant for group i
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Precursor input'
      BEFF = 0.0DO
      WRITE (2,204) NGRP
      DO 40 I=1,NGRP
        READ (1,200) BETA(I),XL(I)
        BEFF = BEFF + BETA(I)
        CIO = BETA(I)*XKINF*XNCO*XLC/XL(I)
        WRITE (2,203) CIO,BETA(I),XL(I)
40      CONTINUE
      WRITE (2,205) BEFF
C-
C- Temperature equations in fuel and coolant - temperature feedback
C-
C- CPRF Cp * Rho in the fuel region in BTU/(ft3-F)
C- this is changed to KW-sec/(cm3-C)
C- CPRC Cp * Rho in the coolant region in BTU/(ft3-F)
C-
C- HF newton cooling coefficient at wall
C-
C- SIGF Macroscopic fission cossction in region in 1/cm
C- CHI Power produced per neutron in the core in KW/neutron
C- CHI = SIGFission * sqrt(E) * Constants
C- CONVF1 Conversion factor for CHI to KW/neutron
C- CONVF2 Conversion factor for CP * Rho to KW-sec/Cm3-C
C-
C- ALPHAF Feedback coefficient for fuel temperature (doppler)
C- ALPHAC Feedback coefficient for coolant temperature
C- ALPHAE Feedback coefficient for Pc, could be pos or neg
C-
C- TFO Average initial core temperature (deg C)
C- TCO Average Initial coolant temperature (TCO > TINP)

```

```

C- TINP      Coolant inlet temperature
C-
  IF (IPRT.GE.1) WRITE (*,*) 'Temperature parameters input'
  READ (1,200) ASURF,VOLF,VOLC
  READ (1,200) RHOF,CPF,ALPHAF,SIGF
  READ (1,200) RHOC,CPC,ALPHAC,HF
  READ (1,200) VCO,ALPHAE
  READ (1,200) TFO,TCO,TINP
C-
C- Calculate the needed parameters from the physical ones
C-
  CHI = SIGF * DSQRT(ENERGY) * CONVF1
  CPRF = CPF * RHOF
  CPRC = CPC * RHOC
C-
C- The term in brackets below is the surface area to volume
C- ratio for the fuel and coolant regions respectively
C- TAU values are heat time constants. See derivation for details
C-
  TAUF = CPRF * VOLF / (HF * ASURF) * 3600.0d0
  TAUC = CPRC * VOLC / (HF * ASURF) * 3600.0d0
  TAUL = RHOC * VOLC / (2.0D0 * VCO)
  WRITE (2,212) VCO,TFO,TCO,TINP
  WRITE (2,206)
  WRITE (2,207) TAUF,TAUC
  WRITE (2,209)
  WRITE (2,207) CPRF,CPRC
  WRITE (2,208) TAUL,CHI
  WRITE (2,210) ALPHAF,ALPHAC,ALPHAE
C-
C- Create the array entries
C-
  IF (IPRT.GE.1) WRITE (*,*) 'Creating A and B'
C-
C- Core to itself coupling term and K external to core
C-
  A(1,1) = (PCO*(1.0D0-BEFF)*XKINF - 1.0D0)*XLC
  B(1,1) = PCO*(1.0D0-BEFF)*XNCO*XLC
  SAV1 = (1.0D0-PCO)*(1.0D0-BEFF)*XLC
  SAV2 = XKINF*XNCO*XLC
  SAV3 = (1.0D0-PCO)*SAV2
C-
C- Loop through the number of reflectors
C-
  DO 60 J=1,NREF
  IF (NREF .EQ. 0) GO TO 60
C-
C- Core to reflector
C-
  A(1+J,1) = PFR(J)*XKINF*SAV1
C-
C- Reflector to core
C-
  A(1,1+J) = PRO(J)*XLR(J)
C-
C- Reflector to reflector
C-
  A(1+J,1+J) = -XLR(J)
C-
C- Temperature to reflector terms via k infinity and Pc
C-
  A(1+J,1+NREF+1) = - PFR(J) * (SAV1*XNCO*ALPHAF -
1  SAV2*ALPHAE)
  A(1+J,1+NREF+2) = - PFR(J) * SAV1*XNCO*ALPHAC
C-
C- Input delta Pr term to core
C-
  B(1,J+1) = PFR(J)*SAV3
C-

```

```

C- External K to reflectors
C-      B(1+J,1) = PFR(J)*SAV1*XNCO
C-
C- Precursors to reflectors
C-      DO 50 I=1,NGRP
C-          A(1+J,1+NREF+2+I) = PFR(J)*(1.0DO-PCO)*XL(I)
50      CONTINUE
C-
C- Done looping through number of reflectors
C-
C- 60  CONTINUE
C-
C- Loop through number of precursor groups
C-      DO 70 I=1,NGRP
C-
C- Precursors to the core
C-          A(1,1+NREF+2+I) = PCO*XL(I)
C-
C- Core to precursors
C-          A(1+NREF+2+I,1) = BETA(I)*XKINF*XLC
C-
C- Precursors to precursors
C-          A(1+NREF+2+I,1+NREF+2+I) = -XL(I)
C-
C- Fuel and coolant temperature to precursors
C-          A(1+NREF+2+I,1+NREF+1) = - BETA(I)*XNCO*XLC*ALPHAF
C-          A(1+NREF+2+I,1+NREF+2) = - BETA(I)*XNCO*XLC*ALPHAC
C-          B(1+NREF+2+I,1) = BETA(I)*XNCO*XLC
C-
C- Done with the precursors
C-
C- 70  CONTINUE
C-
C- Do the fuel and coolant temperature terms
C-      NTEMP = 1 + NREF
C-
C- Core to temperature
C-          A(NTEMP+1,1) = CHI/(CPRF * CONV2)
C-
C- Temperature to core
C-          A(1,NTEMP+1) = - PCO*(1.0DO-BEFF)*XNCO*XLC*ALPHAF - SAV2*ALPHAE
C-          A(1,NTEMP+2) = - PCO*(1.0DO-BEFF)*XNCO*XLC*ALPHAC
C-
C- Do temp to fuel temp coupling
C-          A(NTEMP+1,NTEMP+1) = - 1.0DO/TAUF
C-          A(NTEMP+1,NTEMP+2) = 1.0DO/TAUF
C-
C- Do temp to coolant temp coupling
C-          A(NTEMP+2,NTEMP+2) = - (1.0DO/TAUC) - (1.0DO/TAUL)
C-          A(NTEMP+2,NTEMP+1) = 1.0DO/TAUC
C-
C- Coolant velocity input to the coolant temperature
C-          B(NTEMP+2,1+NREF+1) = (TINP - TCO) / TAUL
C-
C- Done
C-

```

```

RETURN
200 FORMAT(8(F10.4))
201 FORMAT(/,T5,'Core input information',/,T5,22('-'),/,T11,'Nc0',
1 T23,'K-inf(0)',T39,'1/Lc',T56,'Pc(0)',T67,'E-n(KeV)',//,
2 T5,1PD12.4,T20,1PD12.4,T35,1PD12.4,T50,OPF12.6,T65,1PD12.4)
202 FORMAT(/,T5,'Reflector input information for ',I1,' reflectors',
1 /,T5,46('-'),/,T11,'Nr0',
2 T24,'Pfr(0)',T39,'1/Lr',T56,'Pr(0)',//)
203 FORMAT(T5,1PD12.4,T20,1PD12.4,T35,1PD12.4,T50,OPF12.6)
204 FORMAT(/,T5,I1,' Group delayed neutron precursor information',/,
1 T5,44('-'),/,T10,'Ci(0)',T23,'Beta(i)',T37,'Lambda(i)',//)
205 FORMAT(/,T5,44('-'),/,T5,' Beta - eff',T20,F10.6)
206 FORMAT(/,T5,'Tau Values',/,T5,10('-'))
207 FORMAT(T5,'Fuel ',1PD12.4,T40,'Coolant ',1PD12.4)
208 FORMAT(/,T5,'TAUL ',1PD12.4,T40,'CHI-Fuel ',1PD12.4,/)
209 FORMAT(/,T5,'Cp*Rho Values',/,T5,13('-'))
210 FORMAT(/,T5,'Temperature Feedback Coefficients',/,T5,32('-'),/,
1 T5,'Alpha - F = ',1PD12.4,/,T5,'Alpha - C = ',1PD12.4,
1 /,T5,'Alpha - E = ',1PD12.4)
211 FORMAT(/,T5,'No reflectors this run. Controlled by Vc only',/)
212 FORMAT(/,T5,'Initial coolant velocity (Vc0) : ',1PD12.4,/,
1 T5,'Initial fuel temperature (Tf0) : ',1PD12.4,/,
2 T5,'Initial coolant temp (Tc0) : ',1PD12.4,/,
3 T5,'Inlet channel temp (Tinp) : ',1PD12.4,/)
END
SUBROUTINE ASV(A,SV,INSOL,NDIM,NC)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/INFO/ SORCO(20,10),SORC1(20,10)
DIMENSION A(NDIM,NC),SV(NC)
C-
C- This sets up A and SV for doing linear source terms
C- within the ASH methodology. We could also update the
C- inner NCM1 block of A and SV if we wanted to here.
C-
NCM1 = NC - 1
DO 10 I=1,NCM1
A(NC,I) = 0.0D0
A(I,NC) = SORC1(I,INSOL)
SV(I) = SORCO(I,INSOL)
10 CONTINUE
SV(NC) = 1.0D0
A(NC,NC) = 0.0D0
RETURN
END

```

Program RKSF

```

C-
C- Research program by Ken Washington
C-
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Z(5001)
      CHARACTER FNAME*12
      DATA NMAX /5000/

C-
C- Driver routine
C-
      GO TO 6
5     CONTINUE
      WRITE (*,*) 'Input file not found or Error in name...try again.'
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      CLOSE(UNIT=4)
6     CONTINUE
      WRITE (*,103)
      READ (*,104,ERR=5,END=5) FNAME
      OPEN (UNIT=1,FILE=FNAME//'.INP',STATUS='OLD',ERR=5)
      OPEN (UNIT=2,FILE=FNAME//'.OUT',STATUS='NEW',ERR=5)
      OPEN (UNIT=3,FILE=FNAME//'.PRN',STATUS='NEW',ERR=5)
      OPEN (UNIT=4,FILE='U'//FNAME//'.PRN',STATUS='NEW',ERR=5)
10    CONTINUE
      READ (1,101,ERR=99,END=99) NREF,NGRP,NPD,NSOL
      IF (NREF.LT.0 .OR. NGRP.LT.0 .OR. NPD.LT.0) GO TO 99

C-
C- Calculate the size of the matrix from the number of reflectors, etc
C-
      NA = NREF + NGRP + 3
      NB = NREF + 1
      NC = NA - 1
      NF = NA + 2
      NCL = 2*NC + 3
      NWORK = NCL + 1
      NA2 = NA*NA
      NW2 = NWORK*NWORK
      NPNS = NPD*NSOL
      N1 = 1
      N2 = N1 + NA2
      N3 = N2 + NA*NB
      N4 = N3 + NB*NF
      N5 = N4 + NC
      N6 = N5 + NA*NF
      N7 = N6 + NCL*NCL
      N8 = N7 + NW2
      N9 = N8 + NW2
      N10 = N9 + NW2
      N11 = N10 + NW2
      N12 = N11 + NW2
      N13 = N12 + NW2
      N14 = N13 + NW2
      N15 = N14 + NWORK*NPNS
      N16 = N15 + NWORK
      N17 = N16 + NWORK
      N18 = N17 + NWORK
      N19 = N18 + NWORK
      N20 = N19 + NA*NPNS
      NLAST = N20 + NPNS
      NVECT = NLAST - 1
      IF (NLAST .GT. NMAX) THEN
          WRITE (*,*) 'System selected is too big, sorry...'
          GO TO 99
      END IF

C-
C- Initialize all vectors to zero
C-

```



```

DO 30 I=1,NVECT
  Z(I) = 0.0D0
30 CONTINUE
  Z(NLAST) = 999.999D0
C-
C- Here we go
C-
  CALL GORKSF(
  1 NREF,NGRP,NA,NB,NC,NF,NCL,NWORK,NPD,NSOL,NPNS,
  2 Z(N1),Z(N2),Z(N3),Z(N4),Z(N5),Z(N6),
  3 Z(N7),Z(N8),Z(N9),Z(N10),Z(N11),Z(N12),Z(N13),
  4 Z(N14),Z(N15),Z(N16),Z(N17),Z(N18),Z(N19),Z(N20))
C-
C- Write the next element of Z
C-
  GO TO 10
99 CONTINUE
  WRITE (2,102) NVECT,Z(NLAST)
  CLOSE(UNIT=1)
  CLOSE(UNIT=2)
  CLOSE(UNIT=3)
  CLOSE(UNIT=4)
  STOP
101 FORMAT(5I4)
102 FORMAT(//,T5,'There were ',I8,' Array elements used',/,
1   T5,'Next available storage register = ',F10.3,/,
2   T5,'Should be 999.999',/)
103 FORMAT('$file name prefix ? ')
104 FORMAT(A)
  END
  SUBROUTINE GORKSF(
  1 NREF,NGRP,NA,NB,NC,NF,NCL,NWORK,NPD,NSOL,NPNS,
  2 AP,BP,F,XL,BPF,ACL,AWORK,BWORK,CWORK,DWORK,
  3 EWORK,FWORK,GWORK,POUT,P,SV,WR,WI,TIMES,OUTVEC)
  IMPLICIT REAL*8 (A-H,O-Z)
  EXTERNAL UT
  COMMON/INFO/ INSOL,SORCO(10),SORC1(10)
  DIMENSION AP(NA,NA),BP(NA,NB),F(NB,NF),XL(NC),
  2 BPF(NA,NF),ACL(NCL,NCL),GWORK(NWORK,NWORK),
  3 AWORK(NWORK,NWORK),BWORK(NWORK,NWORK),CWORK(NWORK,NWORK),
  4 DWORK(NWORK,NWORK),EWORK(NWORK,NWORK),FWORK(NWORK,NWORK),
  5 POUT(NWORK,NPNS),P(NWORK),SV(NWORK),WR(NWORK),WI(NWORK),
  6 TIMES(NPNS),WORK(5000),IPR(200),COEF(10,2),OUTVEC(NA,NPNS)
  CHARACTER ANS*1,TITLE*70
C-
C- IASH is a flag for the type of solution, 0=Runga Kutta, else ASH
C-
C- IPRT,JPRT are print flag - use as follows:
C- 0 = No printing except for output
C- 1 = Data printed and output and eigenvalues
C- 2 = More complete transient print
C- 3 = All pertinent matrices printed - input,working,output
C- JPRT = IPRT - 1 print level is reduced one for subroutines
C-
C- IPLOT is a plot flag, greater than zero plots that many states
C- a maximum of seven are printed to the plot file. The
C- format is 1X,E9.3
C-
C- IFORM is a flag for the form of the Q matrix used in POLES
C- for the arbitrary eigenvalue assignment.
C- 1=Companion, 2=Diagonal See POLES for more details
C-
C- NEPS is an accuracy parameter - see SOLVER documentation
C- MMAX is the maximum number of terms kept in ASH - see SOLVER
C-
  READ (1,103) TITLE
  READ (1,102) IASH,IPRT,IPLOT,IFORM
  READ (1,102) NEPS,MMAX
  WRITE (2,214) TITLE

```

```

IF (IPRT.GE.1) WRITE (*,214) TITLE
JPRT = IPRT - 1
IF (JPRT.LT.0) JPRT = 0
IF (IPLOT .GT. NA) IPLOT = NA
IF (IPLOT .GT. 6) IPLOT = 6
IF (IASH.EQ.0) THEN
  WRITE (2,221)
  IF (IPRT.GE.1) WRITE (*,221)
ELSE
  WRITE (2,222)
  IF (IPRT.GE.1) WRITE (*,222)
END IF
DO 360 I=1,NSOL
  READ (1,101,END=99,ERR=99) (COEF(I,J),J=1,2),SORCO(I),SORC1(I)
  TIMES(NPD*I) = COEF(I,2)
  WRITE (2,219) I,(COEF(I,J),J=1,2)
  WRITE (2,220) SORCO(I),SORC1(I)
  IF (IPRT.GE.1) THEN
    WRITE (*,219) I,(COEF(I,J),J=1,2)
    WRITE (*,220) SORCO(I),SORC1(I)
  END IF
360 CONTINUE
C-
C- Here we go
C- Get the AP matrix and BP matrices
C-
  CALL MAKEAB(NA,NREF,NGRP,NA,NB,AP,BP,JPRT)
C-
C- Print the created A matrix
C-
  IF (IPRT.GE.3) THEN
    CALL PRMAT(NA,NA,NA,AP,'Plant A matrix')
    CALL PRMAT(NA,NA,NB,BP,'Plant B matrix')
  END IF
C-
C- Read in the desired eigenvalues for the plant matrix and
C- set up the work matrices for POLES subroutine
C-
  DO 55 J=1,NWORK
    DO 55 I=1,NWORK
      AWORK(I,J) = 0.0DO
      BWORK(I,J) = 0.0DO
      CWORK(I,J) = 0.0DO
      DWORK(I,J) = 0.0DO
      EWORK(I,J) = 0.0DO
      FWORK(I,J) = 0.0DO
      GWORK(I,J) = 0.0DO
55 CONTINUE
  DO 70 I=1,NA
    READ (1,101) WR(I),WI(I)
    DO 75 J=1,NA
      AWORK(I,J) = AP(I,J)
      IF (J.LE.NB) BWORK(I,J) = BP(I,J)
75 CONTINUE
70 CONTINUE
  READ (1,101) WR(NA+1),WI(NA+1)
  READ (1,101) WR(NA+2),WI(NA+2)
C-
C- Set the integrators
C- This program will only track the first state...
C-
  AWORK(NA+1,NA+2) = 1.0DO
  AWORK(NA+2,1) = -1.0DO
C-
C- Read the G matrix
C-
  DO 80 I=1,NB
    READ (1,101) (GWORK(I,J),J=1,NF)
80 CONTINUE

```

```

C-
C- get the F matrix using POLES
C-
      CALL POLES(AWORK,BWORK,GWORK,DWORK,FWORK,ework,CWORK,
1  WR,WI,P,WORK,IPR,NWORK,NF,NWORK,NB,
2  IFORM,JPRT,IERR)
      IF (IERR .NE. 0) GO TO 99
C-
C- Set the F matrix equal to FWORK found above
C-
      DO 85 I=1,NB
        DO 85 J=1,NF
          F(I,J) = FWORK(I,J)
85    CONTINUE
      IF (IPRT .GE. 1) WRITE (*,*) 'Done finding F'
C-
C- Now prepare the work matrices for POLE in the observer case
C- this involves transposing A22, and A12 as A and B.
C-
      DO 90 J=1,NWORK
        P(J) = 0.0DO
        DO 90 I=1,NWORK
          AWORK(I,J) = 0.0DO
          BWORK(I,J) = 0.0DO
          CWORK(I,J) = 0.0DO
          DWORK(I,J) = 0.0DO
          EWORK(I,J) = 0.0DO
          FWORK(I,J) = 0.0DO
          GWORK(I,J) = 0.0DO
90    CONTINUE
      DO 115 I=1,NC
        READ (1,101) WR(I),WI(I)
        DO 95 J=1,NC
          AWORK(I,J) = AP(J+1,I+1)
95    CONTINUE
        BWORK(I,1) = -AP(1,I+1)
115   CONTINUE
C-
C- Read in the G matrix for the observer
C-
      READ (1,101) (GWORK(1,J),J=1,NC)
C-
C- Get the L gains
C-
      MPOLE = 1
      CALL POLES(AWORK,BWORK,GWORK,DWORK,FWORK,ework,CWORK,
1  WR,WI,P,WORK,IPR,NWORK,NC,NWORK,MPOLE,
2  IFORM,JPRT,IERR)
      IF (IERR .NE. 0) GO TO 99
C-
C- Set L equal to FWORK found above
C-
      DO 120 I=1,NC
        XL(I) = FWORK(1,I)
120   CONTINUE
      IF (IPRT .GE. 1) WRITE (*,*) 'Done finding L'
C-
C- Close the loop with the observer
C- and define the ACL matrix
C-
      CALL MACL(ACL,AP,BP,F,XL,BPF,NREF,NGRP,NA,NB,NC,NF,NCL)
      IF (IPRT.GE.3)
1    CALL PRMAT(NCL,NCL,NCL,ACL,'Closed loop Acl')
C-
C- Calculate the eigenvalues of the closed loop matrix
C-
      DO 125 I=1,NCL
        DO 125 J=1,NCL
          AWORK(J,I) = ACL(J,I)

```

```

125 CONTINUE
CALL RG(NWORK,NCL,AWORK,WR,WI,EWORK,IPR,P,IERR,0)
IF (IERR.NE.0) THEN
  IF (IPRT.GE.1) WRITE (*,218)
  WRITE (2,218)
  GO TO 99
END IF
IF (IPRT.GE.1) WRITE (*,208)
WRITE (2,208)
DO 130 I=1,NCL
  IF (IPRT.GE.1) WRITE (*,209) WR(I),WI(I)
  WRITE (2,209) WR(I),WI(I)
130 CONTINUE
IF (IPRT.GE.2) CALL PRMAT(NCL,NCL,NCL,EWORK,
1 'Eigenvectors of Acl')
IF (IPRT.GE.1) THEN
  WRITE (*,215)
  READ (*,216) ANS
  IF (ANS.EQ.'N' .OR. ANS.EQ.'n') GO TO 99
END IF
C-
C- Set the source vector so that the source enters the last integrator
C- and fix up matrix so that ASH calculates time and uses that to
C- incorporate a linear source term. Note that the work matrices used
C- by ash are of one more dimension than the closed loop system
C-
C-
C- Input initial conditions
C-
DO 10 I=1,NWORK
  SV(I) = 0.0DO
  DO 10 J=1,NWORK
    AWORK(J,I) = 0.0DO
    BWORK(J,I) = 0.0DO
    CWORK(J,I) = 0.0DO
    DWORK(J,I) = 0.0DO
    EWORK(J,I) = 0.0DO
    FWORK(J,I) = 0.0DO
    GWORK(J,I) = 0.0DO
10 CONTINUE
DO 30 I=1,NCL
  DO 20 J=1,NCL
    AWORK(J,I) = ACL(J,I)
20 CONTINUE
  READ (1,101) P(I)
30 CONTINUE
  BWORK(NCL,1) = 1.0DO
C-
C- Perform a runga kutta solution if IASH is zero
C-
IF (IASH.EQ.0) THEN
  IF (IPRT.GE.1) WRITE (*,*) 'Calling DERK for answer'
  IP = 1
  TIMES(1) = 0.0DO
  DO 370 I=1,NCALC
    POUT(I,1) = P(I)
370 CONTINUE
C-
C- First determine the times where the results will
C- be printed out.
C-
DO 311 J=1,NSOL
  IF (J.EQ.1) THEN
    NPDM1 = NPD - 1
    DT = COEF(J,2)/DBLE(NPDM1)
    PREVT = 0.0DO
  ELSE
    NPDM1 = NPD
    DT = (COEF(J,2)-COEF(J-1,2))/DBLE(NPDM1)

```

```

        PREVT = COEF(J-1,2)
        END IF
        DO 310 I=1,NPDM1
            IP = IP+1
            TIMES(IP) = PREVT + DT*DBLE(I)
310     CONTINUE
311     CONTINUE
C-
C- Note that SV is being used as Pdot in RK routine
C- and the input is gotten from the UT external function
C- Modified on 8/7/85 to do variable time steps and ramps
C-
        NSV = 1
        NCALC = NCL
        IP = 2
        TIME = 0.0DO
        DO 330 ICOEF=1,NSOL
            DT = COEF(ICOEF,1)
            TIMAX = COEF(ICOEF,2)
            INSOL = ICOEF
320     CONTINUE
            CALL LDDERK(UT,NWORK,NCALC,NSV,P,SV,TIME,DT,AWORK,BWORK)
            TEST = DABS(TIMES(IP) - TIME) / DT
            IF (TEST.LT. 0.9DO) THEN
                TIMES(IP) = TIME
                IF (IPRT.GE.1) THEN
                    WRITE (*,202) TIME
                    WRITE (*,217) 1,P(1)
                END IF
                DO 340 I=1,NCALC
                    POUT(I,IP) = P(I)
340     CONTINUE
                    IP = IP+1
                END IF
                IF (TIME.LT.TIMAX .AND. IP.LE.NPNS) GO TO 320
330     CONTINUE
            ELSE
C-
C- Get solution using ASH SOLVER. The routine ASV must be present in
C- version since it solves the linear source problem.
C-
                NCALC = NCL + 1
                P(NCALC) = 0.0DO
                SV(NCALC) = 1.0DO
                SV(NCL) = SORCO(1)
                AWORK(NCL,NCALC) = SORC1(1)
                NSV = 1
                IF (IPRT.GE.1) WRITE (*,*) 'Calling ASH for answer'
                CALL SOLVE2(AWORK,BWORK,CWORK,DWORK,EWORK,FWORK,
1                 NWORK,NCALC,NSV,SV,P,TIMES,POUT,
2                 NPD,NSOL,NPNS,NEPS,MMAX,JPRT)
                IF (IPRT.GE.1) WRITE (*,*) 'Done with ASH'
                END IF
C-
C- Calculate the new internally found inputs store in CWORK
C- note that the exact R(t) is also put in CWORK
C-
                K = 0
                DO 380 ICOEF=1,NSOL
                    INSOL = ICOEF
                    DO 380 KK=1,NPD
                        K=K+1
                        OUTVEC(1,K) = UT(TIMES(K),1)
                        DO 63 I=1,NB
                            OUTVEC(I+1,K) = 0.0DO
                            DO 62 J=1,NF
                                OUTVEC(I+1,K) = OUTVEC(I+1,K) + F(I,J)*POUT(J,K)
62     CONTINUE
63     CONTINUE

```

```

380 CONTINUE
C-
C- Write the plot output file, note that IPLOT dictates
C- how many of the states are included in the PLOT.PRN file
C- also it automatically prints the desired R(t) function
C- in the first column.
C-
      IF (IPLOT .GT. 0) THEN
        IP2 = IPLOT + 1
        WRITE (3,102) NPNS,IP2
        WRITE (4,102) NPNS,NB
        DO 35 I=1,NPNS
          WRITE (3,201) TIMES(I),OUTVEC(1,I),(POUT(J,I),J=1,IPLOT)
          WRITE (4,201) TIMES(I),(OUTVEC(J+1,I),J=1,NB)
35      CONTINUE
        END IF
64      CONTINUE
C-
C- Output the answers to the disk file
C- This was modified on 8/8/85 to print the desired R(t) and
C- the control inputs through time as specified by U = FX
C- Also the controller states are not printed unless IPRT >= 2
C-
      IL = 0
      IF (IPRT.GE.1) WRITE (*,*) 'Outputting answers'
40      CONTINUE
      JB = 5*IL+1
      JE = JB + 4
      IF (JE .GE. NPNS) JE = NPNS
      WRITE (2,202) (TIMES(J),J=JB,JE)
      WRITE (2,211)
      WRITE (2,212) (OUTVEC(1,J),J=JB,JE)
      WRITE (2,203) (POUT(1,J),J=JB,JE)
      DO 50 I=1,NREF
        IF (NREF.EQ.0) GO TO 50
        WRITE (2,204) I,(POUT(I+1,J),J=JB,JE)
50      CONTINUE
        WRITE (2,207) 'F',(POUT(NREF+2,J),J=JB,JE)
        WRITE (2,207) 'C',(POUT(NREF+3,J),J=JB,JE)
        DO 60 I=1,NGRP
          WRITE (2,205) I,(POUT(I+NREF+3,J),J=JB,JE)
60      CONTINUE
        DO 390 I=1,NB
          WRITE (2,213) I,(OUTVEC(I+1,J),J=JB,JE)
390     CONTINUE
        IF (IPRT .GE. 2) THEN
          NCON = NA + 1
          DO 61 I=NCON,NCALC
            WRITE (2,217) I,(POUT(I,J),J=JB,JE)
61      CONTINUE
          END IF
          IL = IL + 1
          IF (JE .LT. NPNS) GO TO 40
C-
99      CONTINUE
C-
C- Thats all
C-
      IF (IERR .NE. 0) THEN
        IF (IPRT.GE.1)
1       WRITE (*,*) 'Run aborted due to internal error #',IERR
        WRITE (2,*) 'Run aborted due to internal error #',IERR
        END IF
        RETURN
101     FORMAT(8F10.4)
102     FORMAT(8I4)
103     FORMAT(A)
201     FORMAT(8(1X,E9.3))
202     FORMAT(/,T2,'Time',T10,5(1PD12.4,1X))

```

```

203 FORMAT(T2,'Nc',T10,5(1PD12.4,1X))
204 FORMAT(T2,'Nr(',I1,')',T10,5(1PD12.4,1X))
205 FORMAT(T2,'C(',I1,')',T10,5(1PD12.4,1X))
207 FORMAT(T2,'T',A1,T10,5(1PD12.4,1X))
208 FORMAT(/,T7,'Closed loop eigenvalues',/,T7,23('-'))
209 FORMAT(T5,1PD12.4,' + j ',1PD12.4)
211 FORMAT(T2,73('-'),/)
212 FORMAT(T2,'R(t)',T10,5(1PD12.4,1X))
213 FORMAT(T2,'U(',I2,')',T10,5(1PD12.4,1X))
214 FORMAT(/,',',A70,/)
215 FORMAT('$Continue with calculation ? ')
216 FORMAT(A1)
217 FORMAT(T2,'X(',I2,')',T10,5(1PD12.4,1X))
218 FORMAT(' *** ERROR *** in RG eigenvalue finder',/)
219 FORMAT(T2,'Step ',I2,': Internal Dt =',D12.4,' to Time =',F10.4)
220 FORMAT(T2,'Source =',1PD12.4,' + ',1PD12.4,' t')
221 FORMAT(T2,'RUNGA KUTTA 4th order solution method',/)
222 FORMAT(T2,'ASH EXP OPERATOR solution method',/)
END
FUNCTION UT(TIME,I)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/INFO/ INSOL,SORCO(10),SORC1(10)
C-
C- This little short routine is a user supplied function that
C- goes to the Runga Kutta solution solver. This one is a modified lin
C- source input but can be changed to anything. Also note that
C- there is only one input in this problem so I is ignored. This
C- is not true for the open loop problem since I represents the
C- input that is being returned.
C-
UT = SORCO(INSOL) + SORC1(INSOL)*TIME
RETURN
END
SUBROUTINE ASV(A,SV,INSOL,NDIM,NC)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/INFO/ IDUM,SORCO(10),SORC1(10)
DIMENSION A(NDIM,NC),SV(NC)
C-
C- This is the equivalent of UT for ASH. It just updates
C- the SV and A matrices for doing different ramps and steps.
C-
A(NC-1,NC) = SORC1(INSOL)
SV(NC-1) = SORCO(INSOL)
A(NC,NC) = 0.0D0
SV(NC) = 1.0D0
RETURN
END
SUBROUTINE MAKEAB(NDIM,NREF,NGRP,NC,NB,A,B,IPRT)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(NDIM,NDIM),B(NDIM,NB),
1 BETA(6),XL(6),PFR(9),XLR(9),PRO(9)
DATA CONVF1,CONVF2 / 1.4022D-06,6.7050D-05 /
C-
C- The intent of this subroutine is to read in all pertinent
C- information and then calculate the system A and B matrices
C- for an averaged core region and coolant channel for the
C- reflected point kinetics model developed for
C- the air force research contract.
C-
C- This version was modified in that the precursor states were
C- placed at the end in order to simplify the process of designing
C- a minimum order controller for the unreachable concentrations
C-
C- Initialize matrix to zero
C-
DO 20 I=1,NDIM
DO 10 J=1,NDIM
A(J,I) = 0.0D0

```

```

          IF (I.LE. NB) B(J,I) = 0.000
10      CONTINUE
20      CONTINUE
C-
C- Read in and write out the problem information
C-
C- Core
C-
C- PHIO = Initial core flux = XNCO * VEL
C- XNCO = Initial neutron distribution
C- XKINF K - Infinity for the core initially
C-      this is now calculated from the critical condition since
C-      it was assumed that the core was initially critical.
C-      the K infinity value is not an independent thing. It is a
C-      function of the transfer probabilities.
C- XLC   Inverse of core lifetime
C- PCO   Initial prob of neutrons staying in the core
C- ENERGY Energy of the neutron spectrum in KeV
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Core input'
      READ (1,200) PHIO,XLC,PCO,ENERGY
C-
C- Calculate the velocity from the energy and use to get XNCO
C- or vice versa if energy is input as zero (defaults to 10 KeV)
C-
      IF (ENERGY .EQ. 0.000) THEN
          ENERGY = 10.000
          VEL = DSQRT(2.000*ENERGY/9.395493D5) * 3.0D+10
          XNCO = PHIO
          PHIO = XNCO * VEL
      ELSE
          VEL = DSQRT(2.000*ENERGY/9.395493D5) * 3.0D+10
          XNCO = PHIO / VEL
      END IF
C-
C- Reflectors
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Reflector input'
      IF (NREF.EQ.0) THEN
          XKINF = 1.000 / PCO
          WRITE (2,201) XNCO,XKINF,XLC,PCO,ENERGY
          WRITE (2,211)
      ELSE
          SUMR = 0.000
          DO 30 N=1,NREF
              READ (1,200) PFR(N),XLR(N),PRO(N)
              SUMR = SUMR + PFR(N)*PRO(N)
30          CONTINUE
          XKINF = 1.000 / (PCO + (1.000-PCO)*SUMR)
          WRITE (2,201) XNCO,XKINF,XLC,PCO,ENERGY
          WRITE (2,202) NREF
          DO 31 N=1,NREF
              XNRO = PFR(N)*(1.000-PCO)*XKINF*XNCO*XLC/XLR(N)
              WRITE (2,203) XNRO,PFR(N),XLR(N),PRO(N)
31          CONTINUE
      END IF
C-
C- Precursors
C-
C- BETA   Delayed neutron fraction for group i
C- XL     The decay constant for group i
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Precursor input'
      BEFF = 0.000
      WRITE (2,204) NGRP
      DO 40 I=1,NGRP
          READ (1,200) BETA(I),XL(I)
          BEFF = BEFF + BETA(I)
          CIO = BETA(I)*XKINF*XNCO*XLC/XL(I)

```



```

      WRITE (2,203) CIO,BETA(I),XL(I)
40  CONTINUE
      WRITE (2,205) BEFF
C-
C- Temperature equations in fuel and coolant - temperature feedback
C-
C- CPRF    Cp * Rho in the fuel region in BTU/(ft3-F)
C-         this is changed to KW-sec/(cm3-C)
C- CPRC    Cp * Rho in the coolant region in BTU/(ft3-F)
C-
C- HF      newton cooling coefficient at wall
C-
C- SIGF    Macroscopic fission cossection in region in 1/cm
C- CHI     Power produced per neutron in the core in KW/neutron
C-         CHI = SIGFission * sqrt(E) * Constants
C- CONV F1 Conversion factor for CHI to KW/neutron
C- CONV F2 Conversion factor for CP * Rho to KW-sec/Cm3-C
C-
C- ALPHAF  Feedback coefficient for fuel temperature (doppler)
C- ALPHAC  Feedback coefficient for coolant temperature
C- ALPHAE  Feedback coefficient for Pc, could be pos or neg
C-
C- TFO     Average initial core temperature (deg C)
C- TCO     Average initial coolant temperature (TCO > TINP)
C- TINP    Coolant inlet temperature
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Temperature parameters input'
      READ (1,200) ASURF,VOLF,VOLC
      READ (1,200) RHOF,CPF,ALPHAF,SIGF
      READ (1,200) RHOC,CPC,ALPHAC,HF
      READ (1,200) VCO,ALPHAE
      READ (1,200) TFO,TCO,TINP
C-
C- Calculate the needed parameters from the physical ones
C-
      CHI = SIGF * DSQRT(ENERGY) * CONV F1
      CPRF = CPF * RHOF
      CPRC = CPC * RHOC
C-
C- The term in brackets below is the surface area to volume
C- ratio for the fuel and coolant regions respectively
C- TAU values are heat time constants. See derivation for details
C-
      TAUF = CPRF * VOLF / (HF * ASURF) * 3600.0d0
      TAUC = CPRC * VOLC / (HF * ASURF) * 3600.0d0
      TAUL = RHOC * VOLC / (2.0D0 * VCO)
      WRITE (2,212) VCO,TFO,TCO,TINP
      WRITE (2,206)
      WRITE (2,207) TAUF,TAUC
      WRITE (2,209)
      WRITE (2,207) CPRF,CPRC
      WRITE (2,208) TAUL,CHI
      WRITE (2,210) ALPHAF,ALPHAC,ALPHAE
C-
C- Create the array entries
C-
      IF (IPRT.GE.1) WRITE (*,*) 'Creating A and B'
C-
C- Core to itself coupling term
C-
      A(1,1) = (PCO*(1.0D0-BEFF)*XKINF - 1.0D0)*XLC
      SAV1 = (1.0D0-PCO)*(1.0D0-BEFF)*XLC
      SAV2 = XKINF*XNCO*XLC
      SAV3 = (1.0D0-PCO)*SAV2
C-
C- Loop through the number of reflectors
C-
      DO 60 J=1,NREF
      IF (NREF.EQ. 0) GO TO 60

```

```

C-
C- Core to reflector
C-       A(1+J,1) = PFR(J)*XKINF*SAV1
C-
C- Reflector to core
C-       A(1,1+J) = PRO(J)*XLR(J)
C-
C- Reflector to reflector
C-       A(1+J,1+J) = -XLR(J)
C-
C- Temperature to reflector terms via k infinity and Pc
C-
C-       1 A(1+J,1+NREF+1) = - PFR(J) * (SAV1*XNCO*ALPHAF -
C-                                     SAV2*ALPHAE)
C-       A(1+J,1+NREF+2) = - PFR(J) * SAV1*XNCO*ALPHAC
C-
C- Input delta Pr term to core
C-       B(1,J) = PFR(J)*SAV3
C-
C- Precursors to reflectors
C-
C-       DO 50 I=1,NGRP
C-         A(1+J,1+NREF+2+I) = PFR(J)*(1.0DO-PCO)*XL(I)
50      CONTINUE
C-
C- Done looping through number of reflectors
C-
C- 60  CONTINUE
C-
C- Loop through number of precursor groups
C-
C-       DO 70 I=1,NGRP
C-
C- Precursors to the core
C-
C-       A(1,1+NREF+2+I) = PCO*XL(I)
C-
C- Core to precursors
C-
C-       A(1+NREF+2+I,1) = BETA(I)*XKINF*XLC
C-
C- Precursors to precursors
C-
C-       A(1+NREF+2+I,1+NREF+2+I) = -XL(I)
C-
C- Fuel and coolant temperature to precursors
C-
C-       A(1+NREF+2+I,1+NREF+1) = - BETA(I)*XNCO*XLC*ALPHAF
C-       A(1+NREF+2+I,1+NREF+2) = - BETA(I)*XNCO*XLC*ALPHAC
C-
C- Done with the precursors
C-
C- 70  CONTINUE
C-
C- Do the fuel and coolant temperature terms
C-
C-       NTEMP = 1 + NREF
C-
C- Core to temperature
C-
C-       A(NTEMP+1,1) = CHI/(CPRF * CONV F2)
C-
C- Temperature to core
C-
C-       A(1,NTEMP+1) = - PCO*(1.0DO-BEFF)*XNCO*XLC*ALPHAF - SAV2*ALPHAE

```

```

C-      A(1,NTEMP+2) = - PCO*(1.ODO-BEFF)*XNCO*XLC*ALPHAC
C- Do temp to fuel temp coupling
C-      A(NTEMP+1,NTEMP+1) = - 1.ODO/TAUF
C-      A(NTEMP+1,NTEMP+2) = 1.ODO/TAUF
C- Do temp to coolant temp coupling
C-      A(NTEMP+2,NTEMP+2) = - (1.ODO/TAUC) - (1.ODO/TAUL)
C-      A(NTEMP+2,NTEMP+1) = 1.ODO/TAUC
C- Coolant velocity input to the coolant temperature
C-      B(NTEMP+2,NREF+1) = (TINP - TCO) / TAUL
C- Done
C-
      RETURN
200  FORMAT(8(F10.4))
201  FORMAT(/,T5,'Core input information',/,T5,22('-'),/,T11,'Nc0',
1    T23,'K-inf(0)',T39,'1/Lc',T56,'Pc(0)',T67,'E-n(KeV)',//,
2    T5,1PD12.4,T20,1PD12.4,T35,1PD12.4,T50,OPF12.6,T65,1PD12.4)
202  FORMAT(/,T5,'Reflector input information for ',I1,' reflectors',
1    /,T5,46('-'),/,T11,'Nr0',
2    T24,'PFR(0)',T39,'1/Lr',T56,'Pr(0)',//)
203  FORMAT(T5,1PD12.4,T20,1PD12.4,T35,1PD12.4,T50,OPF12.6)
204  FORMAT(/,T5,I1,' Group delayed neutron precursor information',/,
1    T5,44('-'),/,T10,'Ci(0)',T23,'Beta(i)',T37,'Lambda(i)',//)
205  FORMAT(/,T5,44('-'),/,T5,' Beta - eff',T20,F10.6)
206  FORMAT(/,T5,'Tau Values',/,T5,10('-'))
207  FORMAT(T5,'Fuel ',1PD12.4,T40,'Coolant ',1PD12.4)
208  FORMAT(/,T5,'TAUL ',1PD12.4,T40,'CHI-Fuel ',1PD12.4,/)
209  FORMAT(/,T5,'Cp*Rho Values',/,T5,13('-'))
210  FORMAT(/,T5,'Temperature Feedback Coefficients',/,T5,32('-'),/,
1    T5,'Alpha - F = ',1PD12.4,/,T5,'Alpha - C = ',1PD12.4,
1    /,T5,'Alpha - E = ',1PD12.4)
211  FORMAT(/,T5,'No reflectors this run. Controlled by Vc only',/)
212  FORMAT(/,T5,'Initial coolant velocity (Vc0) : ',1PD12.4,/,
1    T5,'Initial fuel temperature (Tf0) : ',1PD12.4,/,
2    T5,'Initial coolant temp (Tc0) : ',1PD12.4,/,
3    T5,'Inlet channel temp (Tinp) : ',1PD12.4,/)
      END
      SUBROUTINE MACL(ACL,AP,BP,F,L,BPF,
1      NREF,NGRP,NA,NB,NC,NF,NCL)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L
      DIMENSION ACL(NCL,NCL),AP(NA,NA),BP(NA,NB),F(NB,NF),L(NC),
1      BPF(NA,NF)
C- NA      is the plant dimension = nref + ngrp + 3
C-          the 3 comes from 1 core and 2 temperatures
C- NB      is the number of inputs = nref + 1 (coolant velocity)
C- NC      is the dimension of the plant minus number of outputs
C-          this is the dimension of the estimator = nref + ngrp + 2
C- NF      is the dimension of the plant plus 2 integrators
C-          = 1 + nref + ngrp + 2 + 2 = nref + ngrp + 5
C- NCL     the final closed loop dimension that includes:
C-          the core, reflectors, temperatures, estimator, and
C-          two integrators to allow tracking of ramps
C-          NCL = 1 + 2*(nref + ngrp + 2) + 2
C- note that even though all but core power is estimated, only the
C- precursor estimation is used in the feedback law. The others were
C- only done to obtain enough information to be able to observe the

```

```

C- precursor states. If the observability index had been smaller,
C- I may have been able to reduce the size of the observer some.
C-
C- Set some pointer values
C-
      NP2 = NREF + 2
      NP3 = NP2 + 1
      NP4 = NP3 + 1
      NPG = NP2 + NGRP
C-
C- First copy Aplant to the upper left corner of ACL and set the
C- rest to zero
C-
      DO 10 I=1,NCL
        DO 20 J=1,NCL
          ACL(I,J) = 0.0DO
          IF (I.LE.NA .AND. J.LE.NA) ACL(I,J) = AP(I,J)
        20 CONTINUE
      10 CONTINUE
C-
C- Now multiply BP * F and keep temporarily in a matrix
C-
      DO 30 I=1,NA
        DO 40 J=1,NF
          TEMP = 0.0DO
          DO 50 K=1,NB
            TEMP = TEMP + BP(I,K) * F(K,J)
          50 CONTINUE
          BPF(I,J) = TEMP
        40 CONTINUE
      30 CONTINUE
C-
C- Now multiply Lu on the right side of Fp3 and add to upper left
C- corner of the closed loop matrix
C-
      DO 60 I=1,NA
        DO 70 J=2,NP3
          ACL(I,J) = ACL(I,J) + BPF(I,J)
        70 CONTINUE
        DO 80 J=NP4,NF
          ACL(I,NPG+J) = BPF(I,J)
        80 CONTINUE
        TEMP = 0.0DO
        DO 90 J=1,NGRP
          TEMP = TEMP + BPF(I,NP3+J)*L(NP2+J)
        90 CONTINUE
        ACL(I,1) = ACL(I,1) + BPF(I,1) + TEMP
      60 CONTINUE
C-
C- make the Ac matrix = A22 - L*A12 and the Dc vector
C-
      DO 15 I=1,NC
        TEMP = 0.0DO
        DO 25 J=1,NC
          ACL(NA+I,NA+J) = AP(1+I,1+J) - L(I)*AP(1,1+J)
          TEMP = TEMP + ACL(NA+I,NA+J)*L(J)
        25 CONTINUE
C-
C- Below is Dc vector
C-
      ACL(NA+I,1) = TEMP - L(I)*AP(1,1) + AP(1+I,1)
      15 CONTINUE
C-
C- make Bc * F from Bp * F and put in Acl
C-
      DO 35 I=1,NC
C-
C- first column : Bc*(Fp1 + Fp3*Lu)

```

```

TEMP = 0.0DO
DO 45 J=1,NGRP
  TEMP = TEMP + ( BPF(1+I,NP3+J)-L(I)*BPF(1,NP3+J) ) * L(NP2+J)
45 CONTINUE
  ACL(NA+I,1) = ACL(NA+I,1) + ( BPF(1+I,1)-L(I)*BPF(1,1) ) + TEMP
C-
C- next get reflectors and temp terms in the observer
C-
  DO 55 J=2,NP3
    ACL(NA+I,J) = BPF(1+I,J) - L(I)*BPF(1,J)
55 CONTINUE
C-
C- precursors and the two integrators in the observer
C-
  DO 65 J=NP4,NF
    ACL(NA+I,NPG+J) = ACL(NA+I,NPG+J) +
1      ( BPF(1+I,J)-L(I)*BPF(1,J) )
65 CONTINUE
35 CONTINUE
C-
C- Finally, add on the one and negative one for the integrator
C-
  ACL(NA+NC+1,NA+NC+2) = 1.0DO
  ACL(NA+NC+2,1) = -1.0DO
C-
C- done
C-
  RETURN
  END
1 SUBROUTINE POLES(A,B,G,Q,F,X,V,WR,WI,ORT,
  WORK,IPR,NDIM,N,MDIM,M,IFORM,IPRT,IERR)
  IMPLICIT REAL*8(A-H,O-Z)
  DIMENSION A(NDIM,N),B(NDIM,M),G(MDIM,N),Q(NDIM,N),
1      F(MDIM,N),X(NDIM,N),V(NDIM,N),WR(N),WI(N),
2      ORT(N),WORK(1),IPR(1)
  POLE ASSIGNMENT PROGRAM
C-
C- AUTHOR : LEE H. KEEL modified by Ken Washington
C-
C- TEXAS A & M UNIVERSITY
C-
C- PARAMETER DESCRIPTIONS
C- NDIM,MDIM : DEFINED DIMENSION SIZES IN MAIN PROGRAM
C- N : NUMBER OF STATUS OF GIVEN PLANT
C- M : NUMBER OF INPUT OF GIVEN PLANT
C- A (NxN) : GIVEN PLANT
C- B (NxM) : GIVEN PLANT INPUT
C- Q (NxN) : THIS MATRIX CONTAINS DESIRED POLES
C- : USER MUST PREPARE Q or have MAKEQ do it from WR,WI
C- V (NxN) : Working array for Hessenberg schur
C- G (MxN) : THIS MATRIX WILL BE FREELY CHOSEN BY USER, BUT IF
C- : MAKES SOLUTION MATRIX X SINGULAR, USER SHOULD CHOOSE
C- X (NxN) : SOLUTION OF MATRIX EQUATION AX - XQ = -BG
C- F (MxN) : SOLUTION OF MATRIX EQUATION FX=G
C-
C- OTHER ARRAYS : WORKING AREA FOR SUBROUTINES
C-
C- ORT : WORKING VECTOR FOR HESSENBERG-SCHUR PACKAGE
C- : DIMENSIONED AT LEAST MAX(M,N)
C- WORK : WORKING AREA FOR HESSENBERG-SCHUR PACKAGE
C- : THIS AREA MUST BE PREPARED BIGGER THAN 2M**2+7M
C- : ONE DIMENSION ARRAY
C- IPR : WORKING AREA FOR HESSENBERG-SCHUR PACKAGE
C- : THIS AREA MUST BE PREPARED BIGGER THAN 5N
C- : ONE DIMENSION AREA
C- : NOTE : IPR IS AN INTEGER AREA
C-
C- INPUT : A,B,Q,AND G
C- OUTPUT : F

```



```

        IF (IERR .NE. 0) GO TO 60
        CALL TRAN(F,B,MDIM,NDIM,M,N)
        IF (IPRT .GT. 1)
1      CALL PRMAT(MDIM,M,N,F,'F Gain matrix      ')
60     CONTINUE
        RETURN
        END
        SUBROUTINE MAKEQ(NDIM,N,Q,WR,WI,IFORM)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Q(NDIM,N),WR(N),WI(N)
C-
C- The purpose of this subroutine is to calculate either
C- a companion or phase variable form matrix Q that has the
C- eigenvalues passed in the vectors WR and WI being the
C- real and imaginary parts of the desired eigenvalues
C-
C- IFORM = 0   : ignore imaginary components and force diagonal
C- IFORM = 1   : companion form (the recommended form)
C- IFORM = 2   : phase variable form
C- IFORM = 3   : user decided form
C- IFORM <0 >3 : no action on Q
C-
C- Zero the array
C-
        IF (IFORM.LT.0 .OR. IFORM.GT.3) RETURN
        DO 10 I=1,N
        DO 10 J=1,N
            Q(J,I) = 0.0DO
10     CONTINUE
C-
C- Forced diagonal
C-
        IF (IFORM.EQ.0) THEN
            DO 15 I=1,N
                Q(I,I) = WR(I)
                WI(I) = 0.0DO
15     CONTINUE
C-
C- Companion form (this is the recommended form)
C-
        ELSE IF (IFORM.EQ.1) THEN
            I = 1
20     CONTINUE
C-
C- Add a real pole or last one which must be all real
C-
            IF (WI(I).EQ.0.0DO .OR. I.EQ.N) THEN
                Q(I,I) = WR(I)
                WI(I) = 0.0DO
                I = I+1
            ELSE
C-
C- This is the section to add a conjugate pair
C-
                Q(I,I+1) = 1.0DO
                Q(I+1,I+1) = 2.0DO*WR(I)
                Q(I+1,I) = - WR(I)*WR(I) - WI(I)*WI(I)
                WR(I+1) = WR(I)
                WI(I+1) = -WI(I)
                I = I+2
            END IF
            IF (I.LE.N) GO TO 20
C-
C- Phase variable form
C-
        ELSE IF (IFORM.EQ.2) THEN
            I = 1
30     CONTINUE
C-

```

```

C- Check for all real pole or last one which must be all real
C-
      IF (WI(I).EQ.0.0DO .OR. I.EQ.N) THEN
        Q(N,I) = 1.0DO
        WI(I) = 0.0DO
        IF (I.EQ.1) GO TO 50
        DO 40 JBACK=2,I
          J = I - JBACK + 2
          Q(N,J) = -WR(I)*Q(N,J) + Q(N,J-1)
40      CONTINUE
50      CONTINUE
        Q(N,1) = -WR(I) * Q(N,1)
        I = I+1
      ELSE
C-
C- this is the section to add a conjugate pair
C-
        CON1 = -2.0DO*WR(I)
        CON2 = WR(I)*WR(I) + WI(I)*WI(I)
        Q(N,I) = 1.0DO
        Q(N,I+1) = 0.0DO
        IF (I.EQ.1) GO TO 70
        DO 60 JBACK=2,I
          J = I - JBACK + 3
          Q(N,J) = CON2*Q(N,J) + CON1*Q(N,J-1) + Q(N,J-2)
60      CONTINUE
70      CONTINUE
        Q(N,2) = CON2*Q(N,2) + CON1*Q(N,1)
        Q(N,1) = CON2*Q(N,1)
        I = I+2
      END IF
      IF (I.LE.N) GO TO 30
C-
C- Fill in the rest of the companion matrix and take
C- the negative of the last row
C-
      DO 80 J=1,N
        IF (J.LT.N) Q(J,J+1) = 1.0DO
        Q(N,J) = -Q(N,J)
80      CONTINUE
C-
C- User interactive input Q
C-
      ELSE IF (IFORM.EQ.3) THEN
        CALL MATIN(NDIM,N,N,Q,'Enter the Q matrix from POLES')
      END IF
      RETURN
      END

```



```

PROGRAM RKMODAL
C-
C- Reflected Kinetics Modal Control program by Ken Washington
C-
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/SIZES/ NA,NCL,NUMT
      DIMENSION Z(3001)
      CHARACTER FNAME*8
      DATA NMAX /3000/
C-
C- Driver routine
C-
5     CONTINUE
      WRITE (*,103)
      READ (*,105) FNAME
      OPEN (UNIT=1,FILE=FNAME//'.INP',STATUS='OLD',ERR=5)
      OPEN (UNIT=2,FILE=FNAME//'.OUT',STATUS='NEW',ERR=5)
      OPEN (UNIT=3,FILE=FNAME//'.PRN',STATUS='NEW',ERR=5)
      OPEN (UNIT=4,FILE='U'//FNAME//'.PRN',STATUS='NEW',ERR=5)
10    CONTINUE
      READ (1,101,ERR=40,END=40) NGRP,NUMT,NM,NPD,NSOL
      NPNS = NPD*NSOL
      IF (NGRP.LT.0) GO TO 40
      NREF = 1
C-
C- Calculate the size of the matrix from the number of reflectors, etc
C-
      NA = NREF + NGRP + 3
      NB = NREF + 1
      IF (NB.GT.NA .OR. NC.GT.NA .OR. NGRP.GT.6) THEN
          WRITE (*,*) 'Invalid system size options'
          WRITE (*,*) 'Sorry.'
          GO TO 40
      END IF
C-
C- Fix up the sizes to accomodate internal model
C-
C- NUMT = number of outputs to track
C- NM   = dimension of the measureable output
C- NC   = size of the feedback output matrix (with internal model)
C-
      IF (NUMT .GT. NM) NUMT = NM
      NCL = NA + 2*NUMT
      NC = NM + 2*NUMT
      NWORK = NCL + 1
      NW2 = NWORK*NWORK
      N1 = 1
      N2 = N1 + NCL*NCL
      N3 = N2 + NCL*NB
      N4 = N3 + NB*NC
      N6 = N4 + NC*NCL
      N7 = N6 + NCL*NCL
      N8 = N7 + NW2
      N9 = N8 + NW2
      N10 = N9 + NW2
      N11 = N10 + NW2
      N12 = N11 + NW2
      N13 = N12 + NW2
      N14 = N13 + NW2
      N15 = N14 + NWORK*NPNS
      N16 = N15 + NWORK
      N17 = N16 + NWORK
      N18 = N17 + NWORK
      NVD = N18 + NWORK
      NVA = NVD + NCL*NC
      N19 = NVA + NCL*NC
      NLAST = N19 + NPNS
      NVECT = NLAST - 1
      IF (NLAST .GT. NMAX) THEN

```

```

        WRITE (*,*) 'System selected is too big.'
        WRITE (*,*) 'Modify input file and try again'
        GO TO 40
    END IF
C-
C- Initialize all vectors to zero
C-
    DO 30 I=1,NVECT
        Z(I) = 0.0D0
    30 CONTINUE
        Z(NLAST) = 999.999D0
C-
C- Here we go
C-
        CALL GORESM(NREF,NGRP,NA,NB,NC,NUMT,NM,NCL,NWORK,NPD,
    1 NSOL, NPNS, Z(N1), Z(N2), Z(N3), Z(N4), Z(N6),
    2 Z(N7), Z(N8), Z(N9), Z(N10), Z(N11), Z(N12), Z(N13),
    3 Z(N14), Z(N15), Z(N16), Z(N17), Z(N18),
    4 Z(NVD), Z(NVA), Z(N19))
C-
C- go back and get another set
C-
        GO TO 10
C-
C- end
C-
    40 CONTINUE
        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
        CLOSE(UNIT=3)
        CLOSE(UNIT=4)
        STOP
    101 FORMAT(5I4)
    103 FORMAT('$Input file root name ? ')
    105 FORMAT(A)
        END
        SUBROUTINE GORESM(NREF,NGRP,NA,NB,NC,NUMT,NM,NCL,NWORK,NPD,
    1 NSOL, NPNS, AP, BP, F, CP, ACL, AWORK, BWORK, CWORK, DWORK,
    2 EWORK, FWORK, GWORK, POUT, P, SV, WR, WI, VD, VACH, TIMES)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*4 R4
        COMMON/INFO/ SORCO(10),SORC1(10)
        DIMENSION AP(NCL,NCL),BP(NCL,NB),F(NB,NC),VD(NCL,NC),
    2 CP(NC,NCL),ACL(NCL,NCL),GWORK(NWORK,NWORK),
    3 AWORK(NWORK,NWORK),BWORK(NWORK,NWORK),CWORK(NWORK,NWORK),
    4 DWORK(NWORK,NWORK),EWORK(NWORK,NWORK),FWORK(NWORK,NWORK),
    5 POUT(NWORK,NPNS),P(NWORK),SV(NWORK),WR(NWORK),WI(NWORK),
    6 TIMES(NPNS),IPR(200),VACH(NCL,NC)
        CHARACTER ANS*1,TITLE*40,FNAME*12
C-
C- IASH is a flag for the type of solution, 0=Runga Kutta, else ASH
C-
C- IPRT,JPRT are print flag - use as follows:
C- 0 = No printing except for output
C- 1 = Data printed and output and eigenvalues
C- 2 = More complete transient print
C- 3 = All pertinent matrices printed - input,working,output
C- JPRT = IPRT - 1 print level is reduced one for subroutines
C-
C- IPLOT is a plot flag, greater than zero plots that many states
C- a maximum of seven are printed to the plot file. The
C- format is 1X,E9.3
C-
C- IFORM is a flag for the form of the Q matrix used in POLES
C- for the arbitrary eigenvalue assignment.
C- 1=Companion, 2=Diagonal See POLES for more details
C-
C- NEPS is an accuracy parameter - see SOLVER documentation
C- MMAX is the maximum number of terms kept in ASH - see SOLVER

```

```

C-
C- TIMAX is the maximum solution time, other times printed in
C-   powers of two down from TIMAX for the ASH solution
C-
C- Return point for reading another set
C-
C- 1 CONTINUE
C-
C- Input initial conditions
C-
  READ (1,103,END=99,ERR=99) TITLE
  IF (TITLE(1:3) .EQ. 'END') GO TO 99
  WRITE (2,214) TITLE
  WRITE (*,214) TITLE
  READ (1,102,END=99,ERR=99) IPRT, IPLOT
  JPRT = IPRT - 1
  IF (JPRT.LT.1) JPRT = 1
  WRITE (2,222)
  WRITE (*,222)
  READ (1,102,END=99,ERR=99) NEPS, MMAX
  DO 450 I=1, NPNS
    TIMES(I) = 0.000
450 CONTINUE
  DO 460 INSOL=1, NSOL
    J = (INSOL-1)*NUMT
    K = NPD*INSOL
    DO 360 I=1, NUMT
      READ (1,101,END=99,ERR=99) SORCO(J+I), SORC1(J+I)
      WRITE (2,219) I, SORCO(J+I), SORC1(J+I)
      WRITE (*,219) I, SORCO(J+I), SORC1(J+I)
360 CONTINUE
      READ (1,101) TIMES(K)
      WRITE (2,220) TIMES(K)
      WRITE (*,220) TIMES(K)
460 CONTINUE
C-
C- Here we go
C- Get the AP matrix and BP matrices
C-
  CALL MAKEAB(NCL, NREF, NGRP, NA, NB, AP, BP, JPRT)
C-
C- Now make the CP matrix which is just as many outputs
C- taken as states individually. Usually, there is only one output
C- which is state number one. But we can try track any combination
C- of the states which is specified in first row of CP below
C-
  DO 315 I=1, NCL
    DO 316 J=1, NM
      IF (I.EQ.J) THEN
        CP(J,I) = 1.000
      ELSE
        CP(J,I) = 0.000
      END IF
316 CONTINUE
315 CONTINUE
C-
C- Make the two integrators outputs for modal stabilization
C-
  NUMT2 = NUMT*2
  DO 503 I=1, NUMT2
    CP(NM+I, NA+I) = 1.000
503 CONTINUE
C-
C- one chance to change the output matrix
C-
  CALL MATIN(NC, NC, NCL, CP, 'Plant CP matrix')
C-
C- Set the two integrators for the tracking problem
C- for each of the output variables we want to track

```

```

C- note, the number of tracking outputs must be less than
C- or equal to the number of measureable outputs for feedback
C-
      DO 502 J=1,NUMT
        AP(NA+NUMT+J,NA+J) = 1.0DO
        DO 502 I=1,NCL
          AP(NA+J,I) = -CP(J,I)
502  CONTINUE
C-
C- Calculate the eigenvalues of the open loop matrix
C- and use this as the default settings for STRUCT
C-
      LBAL = 0
      DO 325 I=1,NCL
        DO 325 J=1,NCL
          AWORK(J,I) = AP(J,I)
325  CONTINUE
      CALL RG(NWORK,NCL,AWORK,WR,WI,GWORK,IPR,P,IERR,LBAL)
      WRITE (*,228)
      WRITE (2,228)
      DO 326 I=1,NCL
        WRITE (*,209) WR(I),WI(I)
        WRITE (2,209) WR(I),WI(I)
C-
C- Read in the desired eigenvectors, x are given by -999.0
C-
      READ (1,101) (VD(I,J),J=1,NC)
326  CONTINUE
      IF (IPRT.GE.3) CALL PRMAT(NWORK,NCL,NCL,GWORK,
1    'Eigenvectors of AP ')
      IF (IPRT.GE.1) WRITE (*,*) 'Done with MAKEAB and CPlant'
C-
C- Print the created A matrix
C-
      IF (IPRT.GE.3) THEN
        CALL PRMAT(NCL,NCL,NCL,AP,'Plant A matrix')
        CALL PRMAT(NCL,NCL,NB,BP,'Plant B matrix')
        CALL PRMAT(NC,NC,NCL,CP,'Plant C matrix')
      END IF
C-
C- begin interactive loop
C-
400  CONTINUE
C-
C- The Acl is set equal to AP here in case we want to
C- do an open loop analysis we answer the proceed question (N)o
C- also reset working arrays to zero just in case
C-
      DO 55 J=1,NCL
        DO 55 I=1,NCL
          ACL(I,J) = AP(I,J)
          AWORK(I,J) = 0.0DO
          BWORK(I,J) = 0.0DO
          CWORK(I,J) = 0.0DO
          DWORK(I,J) = 0.0DO
          EWORK(I,J) = 0.0DO
          FWORK(I,J) = 0.0DO
          GWORK(I,J) = 0.0DO
55  CONTINUE
      WRITE (*,*) 'Acl currently set as open loop system'
      WRITE (*,226) NC
      CALL ASK('Proceed with MODAL control',4,R4,R8,I,ANS)
      IF (ANS.EQ.'N' .OR. ANS.EQ.'n') GO TO 410
C-
C- Read in the desired eigenvalues and eigenvectors
C- for the closed loop plant matrix modal output control law
C- The number of outputs (NC) governs how many are assigned
C-
      CALL MATIN(NCL,NC,1,WR,'Real assignable eignvalue')

```

```

      CALL MATIN(NCL,NCL,NC,VD,'Assignable eigenvectors')
C-
C- set the default transformation matrix to best guess and
C- the achievable eigenvector to the desired before projection
C-
      DO 317 I=1,NCL
        DO 319 J=1,NC
          VACH(I,J) = VD(I,J)
319      CONTINUE
        DO 317 K=1,NCL
          IF (K.LE.NB) THEN
            ACL(I,K) = BP(I,K)
          ELSE
            ACL(I,K) = 0.0DO
          END IF
317      CONTINUE
        K = 0
        NAMNB = NB + 1
        DO 318 I=NAMNB,NCL
          K = K + 1
          IF (K.EQ.1 .OR. K.EQ.4) K = K + 1
          ACL(K,I) = 1.0DO
318      CONTINUE
C-
C- chance to change the T matrix to transform B into (I 0)
C-
C-      CALL MATIN(NCL,NCL,NCL,ACL,'B Transformation matrix, T')
C-
C- Calculate the F gains using the STRUCT routine
C- this also creates ACL for you and stores it
C- in the ACL matrix. First we have to calculate the achievable Va
C-
      CALL VA(AP,BP,VACH,WR,AWORK,BWORK,CWORK,DWORK,
1      NCL,NCL,NB,NC,NWORK,IPRT,IERR)
      IF (IERR.NE.0) THEN
        WRITE (*,*) 'Error in VA routine'
        GO TO 400
      END IF
      CALL STRUCT(AP,BP,F,CP,ACL,WR,VACH,AWORK,BWORK,CWORK,
1      NCL,NB,NC,NCL,NB,NC,NCL,NWORK,IPRT,IERR)
      IF (IERR.NE.0) THEN
        WRITE (*,*) 'Error in STRUCT routine'
        GO TO 400
      END IF
C-
C- whew, we made it now do some cleaning up
C- Calculate the eigenvalues of the closed loop matrix
C-
      DO 125 I=1,NCL
        DO 125 J=1,NCL
          AWORK(J,I) = ACL(J,I)
125      CONTINUE
      CALL RG(NWORK,NCL,AWORK,WR,WI,GWORK,IPR,P,IERR,LBAL)
      IF (IERR.NE.0) THEN
        WRITE (*,218)
        GO TO 400
      END IF
      WRITE (*,208)
      DO 130 I=1,NCL
        WRITE (*,209) WR(I),WI(I)
130      CONTINUE
410      CONTINUE
      CALL ASK('Satisfied (Y/N) or e(X)it',4,R4,R8,I,ANS)
      IF (ANS.EQ.'N' .OR. ANS.EQ.'n') GO TO 400
      IF (ANS.EQ.'X' .OR. ANS.EQ.'x') GO TO 99
      WRITE (2,208)
      DO 145 I=1,NCL
        WRITE (2,209) WR(I),WI(I)
145      CONTINUE

```

```

      IF (IPRT.GE.3) CALL PRMAT(NWORK,NCL,NCL,GWORK,
1      'Eigenvectors of Acl')
C-
C- Set the source vector so that the source enters the last integrator
C- and fix up matrix so that ASH calculates time and uses that to
C- incorporate a linear source term. Note that the work matrices used
C- by ash are of one more dimension than the closed loop system
C-
C-
C- Input initial conditions and zero working arrays again
C-
      DO 10 I=1,NWORK
        SV(I) = 0.0DO
        DO 10 J=1,NWORK
          AWORK(J,I) = 0.0DO
          BWORK(J,I) = 0.0DO
          CWORK(J,I) = 0.0DO
          DWORK(J,I) = 0.0DO
          EWORK(J,I) = 0.0DO
          FWORK(J,I) = 0.0DO
          GWORK(J,I) = 0.0DO
10      CONTINUE
      DO 30 I=1,NCL
        DO 20 J=1,NCL
          AWORK(J,I) = ACL(J,I)
20      CONTINUE
      READ (1,101) P(I)
30      CONTINUE
C-
C- Get solution using ASH SOLVER
C-
      NCALC = NCL + 1
      SV(NCALC) = 1.0DO
      P(NCALC) = 0.0DO
      DO 501 I=1,NUMT
        AWORK(NA+I,NCL+1) = SORC1(I)
        SV(NA+I) = SORCO(I)
501      CONTINUE
      IF (IPRT.GE.1) WRITE (*,*) 'Calling ASH for answer'
      NSV = 1
      CALL SOLVE2(AWORK,BWORK,CWORK,DWORK,EWORK,FWORK,
1      NWORK,NCALC,NSV,SV,P,TIMES,POUT,
2      NPD,NSOL,NPNS,NEPS,MMAX,JPRT)
      IF (IPRT.GE.1) WRITE (*,*) 'Done with ASH'
C-
C- Calculate the new internally found inputs store in CWORK
C- note that the exact R(t) is also put in CWORK
C-
      K = 0
      DO 383 INSOL=1,NSOL
        DO 382 KK=1,NPD
          K = K + 1
          DO 381 J=1,NUMT
            JJ = (INSOL-1)*NUMT + J
            CWORK(J,K) = SORCO(JJ) + SORC1(JJ)*TIMES(K)
381          CONTINUE
382          CONTINUE
383          CONTINUE
          DO 380 K=1,NPNS
            DO 63 I=1,NB
              CWORK(I+NUMT,K) = 0.0DO
              DO 62 J=1,NC
                DO 62 M=1,NCL
                  CWORK(I+NUMT,K) = CWORK(I+NUMT,K) +
1                  F(I,J)*CP(J,M)*POUT(M,K)
62          CONTINUE
63          CONTINUE
380          CONTINUE
C-

```

C- Write the plot output file, note that IPLOT dictates
 C- how many of the states are included in the PLOT.PRN file
 C- also it automatically prints the desired R(t) function
 C- in the first column. Also a file called INPUTS.PRN is
 C- produced that holds the inputs through time for plotting
 C-

```

      I = IPLOT + NUMT
      WRITE (3,102) NPNS,I
      WRITE (4,102) NPNS,NB
      DO 35 I=1,NPNS
        WRITE (3,201) TIMES(I),(CWORK(J,I),J=1,NUMT),
          1 (POUT(J,I),J=1,IPLOT)
        WRITE (4,201) TIMES(I),(CWORK(J+NUMT,I),J=1,NB)
      35 CONTINUE
      64 CONTINUE
  C-
  C- Output the answers to the disk file
  C- This was modified on 8/8/85 to print the desired R(t) and
  C- the control inputs through time as specified by U = FX
  C-
```

```

      IL = 0
      40 CONTINUE
      JB = 5*IL+1
      JE = JB + 4
      IF (JE .GE. NPNS) JE = NPNS
      WRITE (2,202) (TIMES(J),J=JB,JE)
      WRITE (2,211)
      DO 509 I=1,NUMT
        WRITE (2,212) I,(CWORK(I,J),J=JB,JE)
      509 CONTINUE
      WRITE (2,203) (POUT(1,J),J=JB,JE)
      DO 50 I=1,NREF
        IF (NREF.EQ.0) GO TO 50
        WRITE (2,204) I,(POUT(I+1,J),J=JB,JE)
      50 CONTINUE
      WRITE (2,207) 'F',(POUT(NREF+2,J),J=JB,JE)
      WRITE (2,207) 'C',(POUT(NREF+3,J),J=JB,JE)
      DO 60 I=1,NGRP
        WRITE (2,205) I,(POUT(I+NREF+3,J),J=JB,JE)
      60 CONTINUE
      DO 390 I=1,NB
        WRITE (2,213) I,(CWORK(I+NUMT,J),J=JB,JE)
      390 CONTINUE
      IF (NCALC.GT.NA) THEN
        NCON = NA + 1
        DO 61 I=NCON,NCALC
          WRITE (2,217) I,(POUT(I,J),J=JB,JE)
        61 CONTINUE
      END IF
      IL = IL + 1
      IF (JE .LT. NPNS) GO TO 40
      99 CONTINUE
```

C-
 C- That's all
 C-

```

      IF (IERR .NE. 0) THEN
        WRITE (*,*) 'Program aborted due to an internal error'
        WRITE (2,*) 'Program aborted due to an internal error'
      END IF
      RETURN
      101 FORMAT(8F10.4)
      102 FORMAT(8I4)
      103 FORMAT(A40)
      201 FORMAT(8(1X,E9.3))
      202 FORMAT(/,T2,'Time',T10,5(1PD12.4,1X))
      203 FORMAT(T2,'Nc',T10,5(1PD12.4,1X))
      204 FORMAT(T2,'Nr(',I1,')',T10,5(1PD12.4,1X))
      205 FORMAT(T2,'C(',I1,')',T10,5(1PD12.4,1X))
      207 FORMAT(T2,'T',A1,T10,5(1PD12.4,1X))
```

```

208 FORMAT(//,T7,'Closed loop eigenvalues',/,T7,23('-'))
209 FORMAT(T5,1PD12.4,' + j ',1PD12.4)
211 FORMAT(T2,73('-'),/)
212 FORMAT(T2,'R',I1,'(t)',T10,5(1PD12.4,1X))
213 FORMAT(T2,'U(',I2,')',T10,5(1PD12.4,1X))
214 FORMAT(/,' ',A40,/)
215 FORMAT(5(1PD12.4))
216 FORMAT(A1)
217 FORMAT(T2,'X(',I2,')',T10,5(1PD12.4,1X))
218 FORMAT(' *** ERROR *** in RG eigenvalue finder',/)
219 FORMAT(T2,'R(',I2,') =',1PD12.4,' + T* =',1PD12.4)
220 FORMAT(T2,'Time = ',1PD12.4)
221 FORMAT(T2,'RUNGA KUTTA 4th order solution method',/)
222 FORMAT(T2,'ASH EXP OPERATOR solution method',/)
224 FORMAT(//,'$Enter Lambda(',I2,') ? ')
225 FORMAT('$Mode(',I2,')', Row(',I2,') ? ')
226 FORMAT(//,' Interactive eigenspectrum assignment ',/,
1 ' For ',I2,' modes',/)
227 FORMAT(T5,'Eigenvector matrix',/,T5,18('-'),/)
228 FORMAT(//,T7,'Open loop eigenvalues',/,T7,21('-'))
END
SUBROUTINE ASV(A,SV,INSOL,NDIM,NC)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/INFO/ SORCO(10),SORC1(10)
COMMON/SIZES/ NA,NCL,NUMT
DIMENSION A(NDIM,NC),SV(NC)
J = (INSOL-1)*NUMT
DO 10 I=1,NUMT
  A(NA+I,NCL+1) = SORC1(J+I)
  SV(NA+I) = SORCO(J+I)
10 CONTINUE
RETURN
END
SUBROUTINE STRUCT(AP,BP,F,CP,T,WR,V,AWORK,CWORK,EWORK,
1 NDIM,LDIM,MDIM,NP,NIN,NOUT,NCL,NWORK,IPRT,IERR)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION AP(NDIM,NP),BP(NDIM,NIN),CP(MDIM,NP),T(NCL,NP),
1 V(NDIM,NOUT),F(LDIM,NOUT),WR(NOUT),
2 AWORK(NWORK,NP),CWORK(NWORK,NP),EWORK(NWORK,NP)
C-
C- NDIM = row dimension of AP, BP, and V in main (max # states)
C- MDIM = row dimension of CP in main (max # outputs)
C- LDIM = row dimension of F in main (max # inputs)
C- NCL = row dimension of T in main (max size of closed loop)
C- this is different than AP because the user may
C- want to have some integrators thrown in later
C-
C- A : NP x NP
C- B : NP x NIN
C- C : NOUT x NP
C- T : NP x NP (also used as a work matrix)
C- on input it is the transformation matrix
C- on return it is Acl closed loop. It can later
C- be enlarged to NCL x NCL since the col dim is passed here
C- V : NP x NOUT (passed as a working array with col dim = nwork)
C- F : NIN x NOUT
C- A1: NIN x NP (never really computed rather kept in T)
C-
C- This routine calculates the F matrix for output feedback
C- and the closed loop matrix and stores it in T.
C- eigenvalue/eigenvector structure assignment problem
C- The input to the subroutine is the plant matrices AP,BP,CP
C- and the desired eigenvectors V and eigenvalues WR
C- Also the T matrix must be provided such that Tinverse * B
C- has the identity matrix on the top NIN rows and zero on the bottom
C- Note that T must be invertible.
C-
C- on output: T contains the closed loop matrix ACL
C-

```



```

C- This version only handles real eigenvalues right now
C-
C- First transform AP and CP in the system using the T matrix
C-
      IERR = 0
      IF (IPRT.GE.4) CALL PRMAT(NCL,NP,NP,T,
1      'Transformation mat ')
      CALL GMULT(CWORK,CP,T,NWORK,MDIM,NCL,NOUT,NP,NP)
      CALL GMULT(EWORK,AP,T,NWORK,NDIM,NCL,NP,NP,NP)
      CALL INVDET(NCL,NP,T,1.0DO,DET,DTRM)
      IF (IPRT.GE.2) WRITE (*,201) DET
      WRITE (2,201) DET
      IF (DET.EQ.0.0DO) THEN
        IF (IPRT.GE.1) WRITE (*,*) 'Singular T matrix'
        IERR = 1
        RETURN
      END IF
      CALL GMULT(AWORK,T,EWORK,NWORK,NCL,NWORK,NP,NP,NP)
      CALL GMULT(EWORK,T,V,NWORK,NCL,NDIM,NP,NP,NOUT)
      IF (IPRT.GE.4) THEN
        CALL PRMAT(NWORK,NP,NP,AWORK,'Plant transformed Ap ')
        CALL PRMAT(NWORK,NOUT,NP,CWORK,'Plant transformed Cp ')
        CALL PRMAT(NWORK,NP,NOUT,EWORK,'Transformed V matrix ')
      END IF
C-
C- AWORK - has in it the transformed Ap matrix
C- CWORK - has in it the transformed Cp matrix
C- EWORK - has in it the transformed desired eigenvectors
C-
C- note that we are done with T as a transformation matrix so we
C- can use it to store things temporarily
C-
C- Now make the Z - A1*V matrix and store in T temporarily
C-
      DO 10 J=1,NOUT
        DO 20 I=1,NIN
          T(I,J) = EWORK(I,J) * WR(J)
          DO 30 K=1,NP
            T(I,J) = T(I,J) - AWORK(I,K) * EWORK(K,J)
          30 CONTINUE
        20 CONTINUE
      10 CONTINUE
      IF (IPRT.GE.4) CALL PRMAT(NCL,NIN,NOUT,T,
1      'Z - A1*V matrix ')
C-
C- We are all done with AWORK so use it now as a storage array
C- Now make CV (store in AWORK) and invert it
C-
      CALL GMULT(AWORK,CWORK,EWORK,NWORK,NWORK,NWORK,NOUT,NP,NOUT)
      CALL INVDET(NWORK,NOUT,AWORK,1.0DO,DET,DTRM)
      IF (IPRT.GE.1) WRITE (*,202) DET
      WRITE (2,202) DET
      IF (DET.EQ.0.0DO) THEN
        IERR = 2
        RETURN
      END IF
      IF (IPRT.GE.4) CALL PRMAT(NWORK,NOUT,NOUT,AWORK,
1      'Inverse of C*V mat ')
C-
C- Now multiply the inverse on the right hand side of Z - A1*V to get F
C- note that T is used to store Z-A1*V
C-
      CALL GMULT(F,T,AWORK,LDIM,NCL,NWORK,NIN,NOUT,NOUT)
C-
C- now we are done with CWORK and EWORK so use as storage arrays
C- Now multiply B*F*C (store FC in CWORK, BFC in EWORK) and add
C- it to Ap to get ACL the closed loop matrix returned in T as ACL.
C-
      CALL GMULT(CWORK,F,CP,NWORK,LDIM,MDIM,NIN,NOUT,NP)

```

```

CALL GMULT(EWORK,BP,CWORK,NWORK,NDIM,NWORK,NP,NIN,NP)
DO 40 I=1,NP
  DO 40 J=1,NP
    T(J,I) = AP(J,I) + EWORK(J,I)
40 CONTINUE
IF (IPRT.GE.2) THEN
  CALL PRMAT(LDIM,NIN,NOUT,F,'F Gains for feedback')
  CALL PRMAT(NCL,NP,NP,T, 'Acl = A + BFC matrix')
END IF
C-
C- Done
C-
RETURN
201 FORMAT(/,T5,'Determinant of T transorm matrix = ',1PD12.4,/)
202 FORMAT(/,T5,'Determinant of CV matrix = ',1PD12.4,/)
END
SUBROUTINE VA(AP,BP,VD,WR,AWORK,BWORK,CWORK,DWORK,
1          NDIM,NA,NB,NC,NWORK,IPRT,IERR)
  IMPLICIT REAL*8 (A-H,O-Z)
  DIMENSION AP(NDIM,NA),BP(NDIM,NB),VD(NDIM,NC),WR(NC),
1          AWORK(NWORK,NA),BWORK(NWORK,NA),
2          CWORK(NWORK,NA),DWORK(NWORK,NA)
C-
C- This routine takes the desired eigenvectors and turns
C- them into achievable ones. An entry which is unspecified
C- must be entered as -999.000 and then will be treated as
C- such leaving the other components as more achievable.
C- This may take a long time to compute especially if there
C- are a lot of outputs so NC is large. (Note that NC is also
C- equal to the number of assignable eigenvalue/vector pairs)
C-
  IERR = 0
  IF (IPRT.GE.2) THEN
    CALL PRMAT(NC,NC,1,WR, 'Desired real eigvals')
    CALL PRMAT(NDIM,NA,NC,VD,'Desired eigenvcs ')
  END IF
  DO NEIG=1,NC
    IF (IPRT.GE.1) THEN
      WRITE (*,*) 'Processing mode ',NEIG
      WRITE (2,*) 'Processing mode ',NEIG
    END IF
C-
C- First compute the inv(lambda*I - A ) * B matrix
C-
    DO I=1,NA
      DO J=1,NA
        AWORK(I,J) = -AP(I,J)
      END DO
      AWORK(I,I) = AWORK(I,I) + WR(NEIG)
    END DO
    CALL INVDET(NWORK,NA,AWORK,1.000,DET,DTNRM)
    IF (DET.EQ.0.000) THEN
      WRITE (2,*) 'Duplicate eigenvalue of system found'
      WRITE (*,*) 'Duplicate eigenvalue of system found'
      WRITE (2,*) 'Routine aborted on mode # ',NEIG
      WRITE (*,*) 'Routine aborted on mode # ',NEIG
      IERR = -NEIG
      GO TO 99
    END IF
    CALL GMULT(BWORK,AWORK,BP,NWORK,NWORK,NDIM,NA,NA,NB)
    DO I=1,NB
      DO J=1,NA
        AWORK(J,I) = BWORK(J,I)
      END DO
    END DO
    IF (IPRT.GE.3) CALL PRMAT(NWORK,NA,NA,AWORK,
1          'inverse(Lam*I - A)*B')
C-
C- Now go through the desired eigenvector see if we need to

```

C- reorder the rows due to not specifying certain entries.

```
C-
      LDIM = 0
      DO I=1,NA
        TESTX = DABS(VD(I,NEIG) + 999.0DO)
        IF (TESTX .GT. 0.5DO) THEN
          LDIM = LDIM + 1
          CWORK(LDIM,1) = VD(I,NEIG)
          DO J=1,NB
            BWORK(LDIM,J) = AWORK(I,J)
          END DO
        ELSE
          CWORK(NA-I+LDIM+1,1) = 1.0DO
          DO J=1,NB
            BWORK(NA-I+LDIM+1,J) = AWORK(I,J)
          END DO
        END IF
      END DO
      DO J=1,NA
        VD(J,NEIG) = CWORK(J,1)
      END DO
```

C-

C- Now compute the zi based on the size of LDIM

C-

```
      CALL TRAN(CWORK,BWORK,NWORK,NWORK,NB,LDIM)
      IF (LDIM.GE.NB) THEN
        CALL GMULT(DWORK,CWORK,BWORK,NWORK,NWORK,NWORK,NB,LDIM,NB)
        CALL SIMEQ(DWORK,CWORK,NWORK,NB,LDIM,IERR)
        IF (IPRT.GE.1) THEN
          WRITE (*,201)
          WRITE (2,201)
        END IF
      ELSE
        CALL GMULT(DWORK,BWORK,CWORK,NWORK,NWORK,NWORK,LDIM,NB,LDIM)
        CALL TRAN(CWORK,DWORK,NWORK,NWORK,LDIM,LDIM)
        CALL SIMEQ(CWORK,BWORK,NWORK,LDIM,NB,IERR)
        CALL TRAN(CWORK,BWORK,NWORK,NWORK,NB,LDIM)
        IF (IPRT.GE.1) THEN
          WRITE (*,202)
          WRITE (2,202)
        END IF
      END IF
      IF (IERR.NE.0) GO TO 99
201 1  FORMAT(' The # of specified modes is >= number of inputs',
202 1  /,' Calculating Achievable eigenvectors by method 1',/)
      IF (IERR.NE.0) GO TO 99
202 1  FORMAT(' The # of specified modes is < number of inputs',
203 1  /,' Calculating Achievable eigenvectors by method 2',/)
```

C-

C- Note that both ways gives rise to the NB x LDIM matrix CWORK

C- which used to the the Li transformed matrix

C-

C- Now multiply this matrix by the li vector on the right and the

C- resulting vector by the original Li matrix on the left

C-

```
      CALL GMULT(BWORK,AWORK,CWORK,NWORK,NWORK,NWORK,NA,NB,LDIM)
      DO I=1,LDIM
        DO J=1,NA
          IF (I.EQ.1) DWORK(J,1) = 0.0DO
          DWORK(J,1) = DWORK(J,1) + BWORK(J,I)*VD(I,NEIG)
        END DO
      END DO
      EVMAX = 0.0DO
      DO J=1,NA
        IF (DABS(DWORK(J,1)).GT.EVMAX) EVMAX = DABS(DWORK(J,1))
      END DO
      IF (EVMAX.EQ.0.0DO) EVMAX = 1.0DO
      DO J=1,NA
        VD(J,NEIG) = DWORK(J,1)/EVMAX
      END DO
```

C-
C- Done with this eigenvector
C-

```

99      END DO
        CONTINUE
        IF (IPRT.GE.4) THEN
          IF (IERR.NE.0) WRITE (2,*)
            1      ' Eigenvectors below may be useless due to IERR'
                CALL PRMAT(NDIM,NA,NC,VD,'Achievable eigenvecs')
          END IF
        RETURN
        END

```

```

SUBROUTINE SOLVE2(A,B,C,D,E,F,NDIM,NC,NSV,SV,
1  P,TIMES,POUT,NPD,NSOL,NPNS,NEPS,MMAX,IPRT)
  IMPLICIT REAL*8 (A-H,O-Z)
  DIMENSION A(NDIM,NC),B(NDIM,NC),C(NDIM,NC),D(NDIM,NC),
1  E(NDIM,NC),F(NDIM,NC),SV(NC),P(NC),
2  TIMES(NPNS),POUT(NDIM,NPNS)

```

C-
C- purpose: give the solution to $\dot{x} = Ax + S$
C- where A and S are time invariant matrices
C- and the initial condition X0 is supplied
C-

C- This version modified 1/86 to handle multiple
C- time steps specified every NPD elements in TIMES array
C- It was also forced to take as many powers of two as it
C- needs to give the results.
C-

C- variables:
C-

C- A, SV	input system matrix and source respectively
C- B,C,D,E,F	on input: work matrices
C-	on output: $E = I + C D(C)$ $F = t D(C)$
C-	such that $x = E*x(0) + F*SV$ (see POUT)
C- NC	size of the system A (NCxNC) matrix
C- NSV	flag for SV (=0 if all SV=0 this saves time)
C-	otherwise NSV = 1 for the general case
C- P	initial condition at time = 0
C- NPD	this is an important integer variable:
C-	= number of solution at powers of two desired
C-	for example: if only the answer at time = t
C-	is wanted then NPD = 2. one is added because
C-	the initial condition is always included in the res
C-	this is only done for the first time group
C-	For successive ones, the initial condition is not
C-	repeated.
C- NSOL	Number of time solutions that are desired.
C-	The times are held in TIMES at every NPD grouping
C-	Recall that NPD = NP2 for all but the first group
C- NPNS	This is the product of NPD and NSOL used to dimension
C-	the TIMES and POUT arrays.
C- POUT	the answers are stored in this matrix. Note that the
C-	second dimension is NPNS. POUT(i,1) = P the initial c
C-	The rest of the entries of POUT(i,2..NPD) contain the
C-	answers to $\dot{x} = Ax + SV$ $x(0) = P$. The last entry
C-	POUT(i,NPD) is the result at time = t NPD times.
C-	That's the way it works on the first INSOL loop
C-	from then on it stores POUT(i,1..NPD) straightforward
C-	POUT is calculated as $E*x(0) + F*SV$
C-	i = 1..NC
C- TIMES	this variable contains the time information.
C-	on input: t = times(NPD), times(NPD*2), ...
C-	on output: times(1) = 0.0
C-	times(npd)=t1 times(npd-1)=t1/2 ... times(2)=t1/2^p
C-	times(npd*2) = t2 , times(npd*2-1)=t2/2 ...
C-	note also that the array times has the dimension NPNS

```

C-          if the p calculated to make norm(H)<1/2 is smaller
C-          than NP2 then P is adjusted accordingly.
C-
C- NEPS      this specifies the epsilon in the series solution of
C-          D(H) where eps = 10^-NEPS.
C-          the bigger NEPS, then the bigger the code will determ
C-          M must be in the series, where M = number of terms.
C-
C- MMAX      this is the maximum allowable value of M that you want
C-          the code to ever determine for calculating the series
C-
C- IPRT      print flag: 0=no printing, 1=print p,M, 2=print mat
C-
C- Determine M dynamically, the maximum is MMAX
C-
      Y = DBLE(NEPS)*DLOG(10.0DO)
      TLOG = DLOG(2.0DO)
      FACT = 2.0DO
      M = 1
      IF (MMAX .EQ. 1) GO TO 2
      IF (MMAX .LE. 0) MMAX = 24
C#
      DO 1 M=1,MMAX
          FACT = FACT*(M+2)
          X = DBLE(M+1)*TLOG + DLOG(FACT)
          IF (X .GT. Y) GO TO 2
1      CONTINUE
2      CONTINUE
      TIMES(1) = 0.0DO
      PREVT = 0.0DO
      IP = 1
      DO 1000 INSOL=1,NSOL
          POSTT = TIMES(INSOL*NPD)
          TIME = POSTT - PREVT
          IF (IPRT.GE.1) WRITE (*,*) 'Process to T = ',POSTT
C-
C- Adjust the matrix and the source
C- for this iteration
C-
      CALL ASV(A,SV,INSOL,NDIM,NC)
      SUM = 0.0DO
C- Create the sum of the squares
      DO 4 I=1,NC
          DO 3 J=1,NC
              SUM = SUM + A(J,I)*A(J,I)
3          CONTINUE
          IF (INSOL.EQ.1) POUT(I,1) = P(I)
4          CONTINUE
      NP2 = NPD
      IF (INSOL.EQ.1 .AND. NP2.GT.1) NP2 = NPD - 1
C-
C- Calculate NP power of 2 scaling
C-
      PP = (0.5DO*DLOG(SUM) + DLOG(TIME))/TLOG
      NP = PP + 1
      IF (NP .LT. NP2) NP = NP2
      IF (NP .LT. 1) NP = 1
      IF (IPRT.GT.1) WRITE (2,100) INSOL,M,NP,SUM
      TWON = 1.0DO
      DO 5 I=1,NP
          TWON = TWON * 2.0DO
5      CONTINUE
      T = TIME / TWON
      CALL SCALAR(A,T,C,NDIM,NC)
      IF (IPRT.GT.2) CALL PRMAT(NDIM,NC,NC,C,'C = AT Matrix (ASH)')
C- Use taylor series in special form
      CALL GENID(D,NDIM,NC)
      DO 7 I=1,M

```

```

        FM = 1.0DO / (2.0DO + DBLE(M-I))
        CALL SCALAR(D,FM,F,NDIM,NC)
        CALL MULTI(C,F,D,NDIM,NC)
        DO 6 J=1,NC
            D(J,J) = D(J,J) + 1.0DO
6       CONTINUE
7       CONTINUE
        CALL MULTI(C,D,E,NDIM,NC)
        DO 8 I=1,NC
            E(I,I) = E(I,I) + 1.0DO
8       CONTINUE
C-
C- Rescale using recursion relation and keep NP2 powers of 2 solutions
C-
        CALL GENID(C,NDIM,NC)
        SP = 1.0DO
        DO 10 I=1,NP
            NN = NP-I
            SP = SP/2.0DO
            T = T*2.0DO
C- Create the I + A*D(A) Matrix
        CALL EQUAL(E,F,NDIM,NC)
        CALL MULTI(E,F,B,NDIM,NC)
        CALL EQUAL(B,E,NDIM,NC)
        IF (NSV .EQ. 0) GO TO 12
C- Skip over source if NSV = 0
        DO 11 ID=1,NC
            F(ID,ID) = F(ID,ID) + 1.0DO
11       CONTINUE
        CALL MULTI(C,F,B,NDIM,NC)
        CALL EQUAL(B,C,NDIM,NC)
12       CONTINUE
C- Skip over rest if NP2 is not within range
        IF (NN .GE. NP2) GO TO 10
        IF (NSV .EQ. 0) GO TO 13
C- Compute the D(C) matrix
        CALL SCALAR(D,SP,F,NDIM,NC)
        CALL MULTI(F,C,B,NDIM,NC)
        CALL SCALAR(B,T,F,NDIM,NC)
13       CONTINUE
        IP = IP + 1
        TIMES(IP) = T + PREVT
C- Compute the POUT solution at time=t
C- POUT = E*P + F*SV
        CALL MVMUL3(E,F,P,SV,POUT,IP,NDIM,NC,NPNS)
10       CONTINUE
        PREVT = TIMES(IP)
        DO 1001 I=1,NC
            P(I) = POUT(I,IP)
1001      CONTINUE
        IF (IPRT.GT.2) THEN
            CALL PRMAT(NDIM,NC,NC,E,'E matrix (ASH)')
            IF (NSV.NE.0) CALL PRMAT(NDIM,NC,NC,F,'F matrix (ASH)')
        END IF
1000      CONTINUE
C- End of ash solver routine
        RETURN
100     FORMAT(/,' Loop ',I4,
1         /,' M   = ',I4,5X,'(# terms in sum)',
1         /,' NP  = ',I4,5X,'(# times scaled by 2)',
2         /,' SUM = ',1PD16.6,5X,'(Sum of squares of A)',/)
        END
        SUBROUTINE SCALAR(A,S,B,NDIM,NC)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(NDIM,NC),B(NDIM,NC)
C- Multiplies B = A * S (S is a scalar)
        DO 20 J=1,NC
            DO 10 JJ=1,NC
                B(JJ,J) = A(JJ,J) * S

```

```

10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE EQUAL(A,B,NDIM,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NDIM,NC),B(NDIM,NC)
C- Equates B to A
      DO 20 J=1,NC
        DO 10 JJ=1,NC
          B(JJ,J) = A(JJ,J)
10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE GENID(A,NDIM,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NDIM,NC)
C- Creates the identity matrix in A
      DO 20 J=1,NC
        DO 10 JJ=1,NC
          A(JJ,J) = 0.0DO
10    CONTINUE
          A(J,J) = 1.0DO
20    CONTINUE
      RETURN
      END
      SUBROUTINE MULTI(A,B,C,NDIM,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NDIM,NC),B(NDIM,NC),C(NDIM,NC)
C-
C- Multiplies two square matrices C = A*B
C- Modified on 7/26/85 to not page fault to speed things up some
C-
      DO 30 K=1,NC
        DO 20 I=1,NC
          C(I,K) = 0.0DO
          DO 10 J=1,NC
            C(I,K) = C(I,K) + A(I,J)*B(J,K)
10    CONTINUE
20    CONTINUE
30    CONTINUE
      RETURN
      END
      SUBROUTINE MVMUL3(AM,BM,AV,BV,C,IT,NDIM,NC,NPNS)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AM(NDIM,NC),BM(NDIM,NC),AV(NDIM),BV(NDIM),C(NDIM,NPNS)
C-
C- Multiplies P(it) = AM*AV + BM*Bv
C- and stores result in IT th column of C matrix
C- Modified on 7/26/85 to not page fault
C-
      DO 5 I=1,NC
        C(J,IT) = 0.0DO
5    CONTINUE
        DO 20 K=1,NC
          DO 10 J=1,NC
            C(J,IT) = C(J,IT) + AM(J,K)*AV(K) + BM(J,K)*BV(K)
10    CONTINUE
20    CONTINUE
      RETURN
      END

```

VITA

Kenneth Edward Washington is the son of Dr. Roosevelt Washington Jr. and Mrs. Lillian Washington. Upon Graduaton from Denton High School in 1978, he enrolled in Texas A&M University where he proceeded to graduate Summa Cum Laude in 1982 with a B.S. degree in Nuclear Engineering. After graduating in 1982, he married his wife, Connie, and enrolled in graduate school at Texas A&M University. He also received his M.S. and Ph.D. degrees in Nuclear Engineering at Texas A&M University in 1983 and 1986 respectively. His research interests include numerical methods. computer simulation, space reactors. and nuclear reactor kinetics and control. Mr. Washington can be reached in care of Dr. Roosevelt Washington, Jr. at 2125 Woodbrook, Denton. Tx 76201.