COO-3052-1
MITNE-136

TIME INTEGRATION METHODS FOR REACTOR KINETICS

by

J. A. W. da Nóbrega, A. F. Henry

December, 1971

Massachusetts Institute of Technology
Department of Nuclear Engineering
Cambridge, Massachusetts 02139

AEC Research and Development Report
Contract AT(11-1)-3052
U.S. Atomic Energy Commission

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF NUCLEAR ENGINEERING
Cambridge, Massachusetts 02139

TIME INTEGRATION METHODS FOR REACTOR KINETICS

by

J. A. W. da Nóbrega, A. F. Henry

December, 1971

COO - 3052 - 1
MITNE - 136
AEC Research and Development Report

Contract AT(11-1)-3052
U. S. Atomic Energy Commission

TIME INTEGRATION METHODS
FOR REACTOR KINETICS

by

José de Anchieta Wanderley da Nóbrega

Submitted to the Department of Nuclear Engineering on December 30, 1971, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

A technique based on the Padé approximations is applied to the solution of the point kinetics equations. The method consists of treating explicitly the roots of the inhour formula which would make the Padé approximations inaccurate. Also, an analytic method is developed which permits a fast inversion of polynomials of the point kinetics matrix and has direct applicability to the Padé approximations.

Results are presented for several cases using various options of the method. It is concluded that the technique provides a fast and accurate computational method for the point kinetics equations.

Also, an implicit solution method for the time-dependent multigroup diffusion equations known as the "theta method" is studied. Both the usual method and a variation of it, derived from the precursor integrated equations, are considered. Several properties of both versions of the theta method are demonstrated.

An attempt is made to find better integration parameters (thetas) for the method, based on corresponding point kinetics calculations. Calculations are done for several test cases, leading to the conclusion that the improvements obtained are of limited value.

Thesis Supervisor: Allan F. Henry
Title: Professor of Nuclear Engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author wishes to express his deep appreciation to Professor Allan F. Henry for his valuable guidance and suggestions during the course of this work.

Thanks are also extended to Professor Kent F. Hansen for some helpful comments and discussions.

The author's wife, Else, provided encouragement and help during this work, particularly in preparing the first draft. Her effort is very much appreciated.

All computations were done at the M.I.T. Information Processing Center. This work was performed under the auspices of the U. S. Atomic Energy Commission.

BIOGRAPHICAL NOTE

José de Anchieta Wanderley da Nóbrega was born on April 2, 1941, in Campina Grande, Paraíba, Brazil. He received his primary and secondary education in Campina Grande, and was graduated from the Colégio Estadual de Campina Grande in December, 1958.

He enrolled at the Escola Nacional de Engenharia of the Universidade do Brasil (now Universidade Federal do Rio de Janeiro) in March, 1959. He received his degree in Electrical Engineering (S.B. equivalent) in December, 1963.

After graduation, he participated of a two - month technical tour through several European countries. He worked at the Instituto de Engenharia Nuclear, Rio de Janeiro, Brazil until August, 1967. During the first half of 1967 he took, concurrently, a few graduate courses in Electrical Engineering at the Universidade Federal do Rio de Janeiro.

In September, 1967, he entered the Massachusetts Institute of Technology with an I.A.E.A. fellowship, having received a Master of Science degree in Nuclear Engineering in September, 1968. He continued his graduate studies at M.I.T.

Mr. da Nóbrega is married to the former Else Joana Schumacher of Petrópolis, Rio de Janeiro, Brazil and they have two children, Ricardo and Alessandra.

Chapter 1

INTRODUCTION

In a nuclear reactor, several kinds of time dependent calculations must be done to assure an economical design and a safe operation. These calculations span time scales several orders of magnitude apart and can be grouped in some general categories.

First, there are calculations with a time scale of weeks or months. These include fuel depletion, buildup of long lived fission products, breeding of new fissile material and poison management, if any. Second, there are calculations with a time scale of hours, namely, overall xenon concentration changes and xenon spatial oscillations. Finally, a third class of calculations is concerned with short-term transients, with time scales of minutes to fractions of a second. The latter involves variations in power, temperature, pressure, and coolant flow and some of their consequences, like fuel expansion, moderator and coolant concentration changes, coolant voiding, Doppler effects, and control rod motion. Of course, some of the aforementioned problems are important or existent for only certain types of reactors. This thesis is concerned with the solution of problems in the third class.

To treat the short-term transients, the simplest approach is provided by the point kinetics model, described by the point kinetics equations, in which only the time dependence of the total neutron population (or some related quantity) is followed. This model is useful to describe the effects of perturbations which involve small changes in the flux shape and is also widely used for stability analysis in reactor control theory. In addition, the point kinetics equations are part of the solution of some space-dependent models, and will in fact be also used in this thesis in such a role.

It is therefore of some interest to have an efficient computational method for the solution of the point kinetics equations. In Chap. 2 we develop a new numerical method for the solution of these equation which is fast and accurate, when compared to some previous methods, particularly for perturbations resulting from a fixed insertion of reactivity.

The point kinetics model is inadequate, however, for certain types of short-term transient analyses especially in large loosely-coupled cores. These involve mainly cases in which spatial changes in the flux shape are considerable so that the underlying assumption of the point kinetics model (that only the amplitude and not the shape of the flux changes during a transient) is made invalid. For this reason, the more accurate space-dependent kinetics models are being used increasingly. So far, spatial kinetics has been based almost entirely on the multigroup diffusion model, because of its relative computational simplicity.

In Chapter 3 we study a time integration technique for the time dependent multigroup diffusion equations based on the approach used in the "theta method".[1,2] This is an implicit method which permits comparatively large time steps to be taken, although at the cost of requiring a considerable amount of computation time per step, because of its implicit nature. In this work we attempt to use the point kinetics equations in order to find better integration parameters (thetas) for the space-time equations. The ultimate goal is to improve the accuracy of a given space-time calculation or to increase the time step size demanded for a given accuracy. A variation of the theta method, which uses the precursor integrated equations, is also considered in Chap. 3. Several properties of the two versions are demonstrated in that chapter.

In both the point and space-dependent kinetics equations to be considered, the time dependent behavior of the reactor parameters is assumed given, whether such behavior is externally induced or is due to feedback effects. The latter are supposed to have been calculated by some approximate method not explicitly discussed in this work.

In Chapter 4, results are presented for some point kinetics problems, using the method of Chap. 2, as well as for some space-dependent problems, using the techniques considered in Chap. 3.

Finally, Chap. 5 presents the conclusions reached in this work and offers some recommendations for further research. The technique of Chap. 2 is found to be, in fact, a fast and accurate computational method for point kinetics. Also, it is concluded that only limited improvements are obtained by the use of point kinetics as a means to find better time integration parameters for the space-dependent kinetics equations.

Chapter 2

SOLUTION OF THE POINT KINETICS EQUATIONS

2.1 The Point Kinetics Equations

For various reasons, the point kinetics equations are still an important set of equations in this era of space-dependent kinetics. First, they can be derived formally from a transport theory formulation[3] and will lead to quite accurate results if a good approximation for the flux shape is available. Thus they are useful for describing the effects of small perturbations or in rough preliminary reactor calculations, particularly for tightly coupled cores. Secondly, they are widely applied in the control theory aspects of reactor analysis, where it is difficult to perform a study using a full space-time description of the neutronics.[4,5] In addition, some of the more elaborate space-time methods rely on the solution of the point kinetics equations, namely the family of quasi-static methods[6] and some synthesis schemes.[7] In some of these methods a generalized point kinetics formulation results in which the elements of the ordinary point kinetics equations are replaced by matrices having a similar, but more generalized, physical meaning.[7,8]

We feel therefore justified in developing an efficient solution method for the point kinetics equations, even recognizing that they do not offer an insurmountable barrier for present-day computers, whatever the solution technique adopted.

The preferred form of the point kinetics equations is written in terms of the neutron generation time.[9] They are:

$$\frac{dn}{dt} = \frac{\rho-\beta}{\Lambda} n + \sum_{i=1}^{I} \lambda_i c_i + f$$

$$\frac{dc_i}{dt} = \frac{\beta_i}{\Lambda} n - \lambda_i c_i \qquad (i=1,2,\ldots,I)$$

where n and the c_i are weighted integrals of the neutron and i^{th} precursor concentrations; f is a source term; ρ is the reactivity; β_i is the effective delayed neutron fraction for group i; λ_i is the decay constant for precursor i; Λ is the neutron generation time; and I is the total number of delayed neutron groups. The quantities n, c_i, f and ρ are, in general, functions of the time t and β_i, λ_i and Λ are assumed constant. In addition, ρ may be a function of n in feedback problems.

The above equations can be written in matrix form as

$$\frac{d\underline{\Psi}}{dt} = A\,\underline{\Psi} + \underline{f}, \tag{1}$$

where $\underline{\Psi} = \mathrm{col}[n\ c_1\ c_2\ \ldots\ c_I]$, $\underline{f} = \mathrm{col}[f\ 0\ 0\ \ldots\ 0]$, and

$$A = \begin{bmatrix} \alpha & \lambda_1 & \lambda_2 & \cdots & \lambda_I \\ \mu_1 & -\lambda_1 & 0 & \cdots & 0 \\ \mu_2 & 0 & -\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_I & 0 & 0 & \cdots & -\lambda_I \end{bmatrix},$$

with $\alpha = (\rho-\beta)/\Lambda, (\beta=\Sigma\beta_i)$ and $\mu_i = \beta_i/\Lambda$. A is usually called the point kinetics matrix.

When ρ and f vary with time, Eq. (1) is usually solved in a series of time steps, the assumption being made that ρ and f are constant and equal to their average values during the time step under consideration. Later we shall analyze the implications of this assumption.

For constant A the formal solution of Eq. (1) is given by

$$\underline{\Psi}_1 = e^{hA}\,\underline{\Psi}_0 + A^{-1}[e^{hA}-I]\,\underline{f}, \tag{2}$$

where $h = t_1-t_0$ and $\underline{\Psi}_0$, $\underline{\Psi}_1$ are the values of the vector $\underline{\Psi}$ at the beginning and end of the time step, respectively. Note that the matrix function multiplying $\underline{f}$ is always well defined, even if A is singular.

Because this system of equation is a relatively simple one, its solution can be done in practice by "brute-force" methods, that is, calculating all the eigenvalues of the matrix A and performing straightforward computations. However, this is an expensive scheme, when the reactivity varies with time, since the calculation of the eigenvalues amounts to solving and $(I+1)^{th}$ order algebraic equation (the inhour formula) for all its roots at every time step.

Thus a wealth of methods have been devised to solve Eq. (1) in a more economical way. Among them we have: i) methods based essentially on Taylor series expansions;[10,11] ii) methods based on convolution integrals using numerical integration;[12,13] iii) methods based on integral equation formulations and approximation of the integrand;[14-17] iv) methods based on some approximation of matrix exponentials;[18,19] v) methods based on extrapolation of low order approximations;[20] vi) methods using spline functions.[17] Of course there are many other methods and we do not intend the above list to be exhaustive.

In what follows, we propose to apply yet another method which is expected to be very fast and accurate and which has the ability to reproduce all the features of transients, including the prompt jump, which is not very well represented in some of the other methods.

Our method is based on an approximate expression for e^{hA}, which was suggested by an earlier scheme called the "purification method".[21] Also important in the present work is a procedure to invert polynomials of the point kinetics matrix A, which might be considered a good method of solution for Eq. (1) in its own right, as we shall see later.

2.2 An Approximation for exp(hA)

As seen in the last section there is a need for a simple and accurate computational method to evaluate exp(hA). In this section we derive a method which we believe to possess these characteristics. The essential idea will be to replace exp(hA) by an approximation f(hA) plus a correction term to account properly for those eigenvalues of A for which the approximation is inaccurate.

Let the matrix A have eigenvalues ω_i and eigenvectors $\underline{u}_i$. Also let A^T have eigenvectors $\underline{v}_i$, corresponding to eigenvalues ω_i which are the same as those of A. The eigenvalues ω_i are the roots of the inhour formula

$$\rho = \omega\Lambda + \omega \sum_{i=1}^{I} \frac{\beta_i}{\lambda_i+\omega} \tag{3}$$

and can be ordered as

$$\omega_I < -\lambda_I < \omega_{I-1} < -\lambda_{I-1} < \ldots < \omega_1 < -\lambda_1 < \omega_0. \tag{4}$$

We now state the following

THEOREM: If $A\underline{u}_i = \omega_i\underline{u}_i$ and $A^T\underline{v}_i = \omega_i\underline{v}_i$ with $\underline{u}_i^T\underline{v}_i = 1$, that is, normalized to unity, the following expression holds for any f(A) for which $f(\omega_i)$ is bounded for all i:

$$e^A = f(A) + \sum_{i=0}^{I} [e^{\omega_i} - f(\omega_i)]\underline{u}_i\underline{v}_i^T. \tag{5}$$

Proof: Since $e^A\underline{u}_k = e^{\omega_k}\underline{u}_k$ for all $\underline{u}_k$ we have only to show that this same relationship holds for the right hand side of Eq. (5), since two matrices with the same eigenvalues and eigenvectors are identical.

Thus we write

$$e^A\,\underline{u}_k = f(A)\underline{u}_k + \sum_{i=0}^{I} [e^{\omega_i} - f(\omega_i)]\underline{u}_i\underline{v}_i^T\,\underline{u}_k.$$

But $\underline{v}_i^T\underline{u}_k = \delta_{ik}$, since the eigenvalues of A and A^T form a biorthonormal set when properly normalized.[22] As a result,

$$e^{A}\,\underline{u}_k = f(\omega_k)\underline{u}_k + \sum_{i=0}^{I} [e^{\omega_i} - f(\omega_i)]\underline{u}_i\delta_{ik}$$

$$= f(\omega_k)\underline{u}_k + [e^{\omega_k} - f(\omega_k)]\underline{u}_k = e^{\omega_k}\,\underline{u}_k \qquad \text{Q.E.D.}$$

To suit our particular needs we introduce the factor h. Equation (5) then becomes

$$e^{hA} = f(hA) + \sum_{i=0}^{I} [e^{h\omega_i} - f(h\omega_i)]\underline{u}_i\underline{v}_i^T \tag{6}$$

with $\underline{u}_i$ and $\underline{v}_i$ unchanged, since $(hA)\underline{u}_i = (h\omega_i)\underline{u}_i$ and analogously for $\underline{v}_i$.

Equation (6) is so far a mere mathematical manipulation. It has, however, a form which will permit us to approximate e^{hA} in an economical fashion. We only have to note that, if $f(h\omega_i)$ is a good approximation for $e^{h\omega_i}$, we are justified in dropping the i^{th} term from the summation. It will have a very small coefficient, namely

$$[e^{h\omega_i} - f(h\omega_i)] \ll 1, \tag{7}$$

since $e^{h\omega_i} \simeq f(h\omega_i)$.

Thus, to a high degree of accuracy, we have

$$e^{hA} \simeq g(hA) \equiv f(hA) + \sum_{k}{}' \; [e^{h\omega_k} - f(h\omega_k)]\underline{u}_k\underline{v}_k^T \quad , \tag{8}$$

where the sum Σ' is over only those k for which Eq. (7) does not hold.

Another way to see the small size of the errors involved is to calculate the Euclidean norm of the difference between the exact exponential and the approximation g(hA). From Eqs. (6) and (8), we have

$$\| e^{hA} - g(hA) \| = \| \sum_i'' \, [e^{h\omega_i} - f(h\omega_i)] \underline{u}_i \underline{v}_i^T \| \quad ,$$

where Σ'' runs over all i for which Eq. (7) holds, that is, those excluded from the summation Σ' in Eq. (8). Using the properties of norms[23] we have:

$$\| e^{hA} - g(hA) \| \leq \sum_i'' | e^{h\omega_i} - f(h\omega_i) | \cdot \| \underline{u}_i \| \cdot \| \underline{v}_i \| \quad .$$

Since $\underline{u}_i^T \underline{v}_i = 1$, $\| \underline{u}_i \| \cdot \| \underline{v}_i \| \geq 1$ but is expected not to be very large. From Eq. (7) we can then conclude that the norm in question is indeed small and that, accordingly, g(hA) is a good approximation for the exponential.

In order to implement the approximation (8) we need an expression for the eigenvectors $\underline{u}_k$ and $\underline{v}_k$ of the matrix A and its transpose. These are easily calculated from their defining equations. They are:

$$\underline{u}_k = \mathrm{col}\,[1 \;\; \frac{\mu_1}{\lambda_1+\omega_k} \;\; \frac{\mu_2}{\lambda_2+\omega_k} \;\cdots\; \frac{\mu_I}{\lambda_I+\omega_k}]$$

$$\underline{v}_k = \nu_k \cdot \mathrm{col}\,[1 \;\; \frac{\lambda_1}{\lambda_1+\omega_k} \;\; \frac{\lambda_2}{\lambda_2+\omega_k} \;\cdots\; \frac{\lambda_I}{\lambda_I+\omega_k}] \quad ,$$

where ν_k is a normalization factor given by

$$\nu_k = \left(1 + \sum_{i=1}^{I} \frac{\mu_i \lambda_i}{(\lambda_i+\omega_k)^2}\right)^{-1} < 1.$$

The present method is thus seen to require calculation of only those ω_k which are included in Σ' of Eq. (8). In practice, only ω_0 and ω_I are ever computed. Appendix A describes a procedure to find ω_0 and ω_I from the inhour formula, Eq. (3), using the Newton-Raphson method.

2.3 The Padé Approximations and Related Inversions

In order to use the method derived in the previous section, we need to find an approximation f(x) for e^x such that Eq. (7) holds for most eigenvalues ω_i, so that only one, or at most two terms have to be included in Σ' of Eq. (8). Another requirement for f(x) is that the computation of f(hA) can be done without too much effort.

A particular class of approximations that fulfills these conditions are the Padé rational approximations. These approximations are known to be consistent and unconditionally stable when the numerator is a polynomial of the same degree as the denominator or smaller.[24] For any of these approximations for which the degree of the polynomial of the denominator is larger than unity we will have a full square matrix of order (I+1) to invert, which is a task one normally tries to avoid, particularly for the case of varying reactivity when such inversion needs to be done at every time step.

We have, however, developed a method in which, by going temporarily to the complex plane, we obtain simple analytic expressions for such inverses. As a result, essentially the same number of arithmetic operations suffice to evaluate the product of the inverse of a polynomial of the matrix A and a vector as are required to compute the product of the polynomial itself and a vector. This fact makes the computational effort involved in using implicit methods of any order equal to that used for explicit methods of the same order (Taylor series expansions). However, instabilities associated with the latter are avoided.

The method is based on an expression for the inverse of $[I-\varepsilon A]$ where ε is any scalar such that $[I-\varepsilon A]$ is non-singular. This expression is

$$[I-\varepsilon A]^{-1} = \gamma \underline{a}\, \underline{b}^T + D, \tag{9}$$

where
$$\gamma = \left(1 - \varepsilon \frac{\rho}{\Lambda} + \sum_{i=1}^{I} \frac{\varepsilon \mu_i}{1+\varepsilon \lambda_i}\right)^{-1}$$

$$\underline{a} = \mathrm{col}\left[1 \; \frac{\varepsilon\mu_1}{1+\varepsilon\lambda_1} \; \frac{\varepsilon\mu_2}{1+\varepsilon\lambda_2} \; \cdots \; \frac{\varepsilon\mu_I}{1+\varepsilon\lambda_I}\right]$$

$$\underline{b} = \mathrm{col}\left[1 \; \frac{\varepsilon\lambda_1}{1+\varepsilon\lambda_1} \; \frac{\varepsilon\lambda_2}{1+\varepsilon\lambda_2} \; \cdots \; \frac{\varepsilon\lambda_I}{1+\varepsilon\lambda_I}\right]$$

$$D = \mathrm{diag}\left[0 \; \frac{1}{1+\varepsilon\lambda_1} \; \frac{1}{1+\varepsilon\lambda_2} \; \cdots \; \frac{1}{1+\varepsilon\lambda_I}\right].$$

The validity of Eq. (9) can be verified directly by multiplication by $[I-\varepsilon A]$.

Notice that only γ, a scalar, depends on the reactivity. Also it is worth noting the computational advantages of multiplying an N-dyad by a vector, as compared to a general N×N matrix. The former involves only 3N arithmetic operations as compared to $2N^2$ in the latter. We further observe that, for this particular matrix A, essentially the same number of operations are required to effect the product of either $[I-\varepsilon A]^{-1}$ or $[I+\varepsilon A]$ and some vector.

The expression (9) is of no great advantage by itself since we can always solve directly the system of equations implied by the inverse shown, spending essentially the same computational effort, in this case. The utility of the analytical inversion is evident, however, when one tries to invert a general polynomial of the matrix A which can be expressed as a product of factors having the form $[I-\varepsilon A]$. To see this utility in more detail, consider the matrix polynomial

$$P_N(A) = \sum_{n=0}^{N} c_n A^n$$

with $\{c_n\}$ being real numbers and $c_0=1$.

We factor $P_N(A)$ as

$$P_N(A) = \prod_{n=1}^{N} [I-\varepsilon_n A],$$

where $\{\varepsilon_n\}$ are, in general, complex numbers. Then,

$$[P_N(A)]^{-1} = \prod_{n=1}^{N} [I-\varepsilon_n A]^{-1}.$$

Of course, the $\{\varepsilon_n\}$ will either be real numbers or form complex conjugate pairs. For ε_n real we use Eq. (9) to do the inversion or directly solve the pertinent system of equations. For a complex conjugate pair we consider the pair of factors

$$[I-\varepsilon A]^{-1}[I-\bar{\varepsilon}A]^{-1} = I - 2\mathrm{Re}(\varepsilon)A + |\varepsilon|^2 A^2,$$

which is a real matrix and thus has a real inverse. By using Eq. (9) with complex ε for both factors and combining the complex elements of the resulting expression we will get a sum of two real dyads and one real diagonal matrix as the desired inverse. Almost all of the elements involved can be precomputed since only the reactivity changes with time and it appears in a simple fashion in the coefficients of the dyads as we shall see later in a specific example.

For the Padé approximations Eq. (7) will be satisfied, in general, whenever $|h\omega_i| << 1$ and therefore we can use this as a preliminary criterion to decide which terms should be included in Eq. (8). From Eq. (4) we see that, with the exception of ω_I and, possibly, ω_0, all the eigenvalues of the matrix A have magnitudes smaller than λ_I. Therefore, the condition $h\lambda_I << 1$ assures that Eq. (7) will be satisfied for most eigenvalues ω_i. For practical values of λ_I, this allows that relatively large values of h be taken with good accuracy.

As an example, we consider two of the Padé approximations, namely the Crank-Nicholson and the second order implicit approximations.

2.3.1 The Padé (1,1) Approximation

This approximation, commonly known as the Crank-Nicholson approximation, is given by

$$f_1(hA) = [I-\tfrac{1}{2}hA]^{-1}[I+\tfrac{1}{2}hA]$$

and involves no particular difficulty, except that one must always choose h so that $h\omega_i \neq 2$ to keep $f(h\omega_i)$ bounded. Since only the eigenvalue ω_0 of A can be positive we simply must have $h\omega_0 \neq 2$. For small h, the error of this aproximation is given by

$$e^{hA} - f_1(hA) = -\tfrac{1}{12}h^3A^3 + 0(h^4).$$

Since all eigenvalues of A have small magnitudes except possibly ω_0 and ω_6 these are the only ones we will ever need to calculate explicitly for use in Eq. (8), if we do not want to take h very small. Also, we have that $|e^{h\omega_i}-f_1(h\omega_i)| < 0.035$ or 0.14 for negative $h\omega_i$ down to -1 or -2, respectively. These figures are low enough to require the inclusion of the ω_I term in Eq. (8) only when $h\omega_I < -1$ or, in many cases, when $h\omega_I < -2$. Such conditions are clearly much less stringent than the general condition $|h\omega_I| << 1$.

The above considerations, coupled with the fact that for most practical cases only one of ω_0 and ω_I is of large magnitude, indicate that, in many problems, satisfactory results will be obtained by treating explicitly only one term in Eq. (8).

2.3.2 The Padé (2,0) Approximation

This approximation, which we call the second order implicit approximation, is given by

$$f_2(hA) = [I-hA+\tfrac{1}{2}h^2A^2]^{-1},$$

and $f_2(h\omega_i) \neq 0$ for any real value of $h\omega_i$. For small h the error of this approximation is given by

$$e^{hA} - f_2(hA) = \tfrac{1}{6}h^3A^3 + 0(h^4).$$

We will show here an explicit application of the method which we developed, to find the inverse in $f_2(hA)$. We factor $f_2(hA)$ as

$$f_2(hA) = [I-\varepsilon A]^{-1}[I-\bar{\varepsilon}A]^{-1},$$

where $\varepsilon = h(1+i)/2$ with $i = \sqrt{-1}$.

Using Eq. (9) for these values of ε and carrying out some involved but straightforward algebra we eliminate all imaginary numbers in the final expression which is given by

$$f_2(hA) = \gamma_1(\underline{a}_1\underline{b}_2^T+\underline{a}_2\underline{b}_1^T) + \gamma_2(\underline{a}_1\underline{b}_1^T-\underline{a}_2\underline{b}_2^T) + D',$$

where

$$\gamma_1 = 2r/(r^2+s^2); \qquad \gamma_2 = 2s/(r^2+s^2);$$

$$r = 1 - \frac{h\rho}{\Lambda} + h\sum_{i=1}^{I}\mu_i p_i(2+h\lambda_i);$$

$$s = 1 + h^2\sum_{i=1}^{I}\mu_i p_i \lambda_i; \qquad p_i = [1 + (1+h\lambda_i)^2]^{-1};$$

$$\underline{a}_1 = \begin{bmatrix} 1 \\ h\mu_1 p_1(1+h\lambda_1) \\ h\mu_2 p_2(1+h\lambda_2) \\ \cdots \\ h\mu_I p_I(1+h\lambda_I) \end{bmatrix}; \qquad \underline{a}_2 = \begin{bmatrix} 0 \\ h\mu_1 p_1 \\ h\mu_2 p_2 \\ \cdots \\ h\mu_I p_I \end{bmatrix};$$

$$\underline{b}_1 = \begin{bmatrix} 1 \\ h\lambda_1 p_1(1+h\lambda_1) \\ h\lambda_2 p_2(1+h\lambda_2) \\ \cdots \\ h\lambda_I p_I(1+h\lambda_I) \end{bmatrix}; \qquad \underline{b}_2 = \begin{bmatrix} 0 \\ h\lambda_1 p_1 \\ h\lambda_2 p_2 \\ \cdots \\ h\lambda_I p_I \end{bmatrix};$$

$$D' = \mathrm{diag}[0\ 2p_1\ 2p_2\ \cdots\ 2p_I].$$

We note that only the scalars r, γ_1 and γ_2 depend on the reactivity and therefore almost everything can be precomputed. Also, although we have displayed four dyads in the expression for $f_2(hA)$ we could combine them into two. However, the form shown provides some computational convenience in that we can group the four dyads in two ways keeping either $(\underline{a}_1,\underline{a}_2)$ or $(\underline{b}_1,\underline{b}_2)$ explicit.

For this approximation we have $|e^{h\omega_i} - f_2(h\omega_i)| < 0.07$ for all negative $h\omega_i$ (maximum 0.069 at $h\omega_i = -2.62$). This figure is small enough that results of high accuracy can be obtained by treating explicitly only the ω_0 term in Eq. (8). Moreover if $|h\omega_0|$ is sufficiently below unity, none of the ω_k terms in Eq. (8) need be considered explicitly.

The same complex factorization idea used here could be applied to the denominator of any higher order Padé approximation but there is not much advantage in going beyond $0(h^3)$, since there is an $0(h^3)$ error inherent in the assumption of constant reactivity and source during the time step as we will see in the next section. For this reason, only the two approximations above have been considered.

In summary, we developed in this section an efficient analytical method to invert polynomials of the matrix A which has direct applicability to the Padé approximations. The method was applied to two specific cases, the Padé (1,1) and the Padé (2,0) approximations. Numerical characteristics of these approximations pertinent to their utilization in Eq. (8) were discussed. A concrete implementation for the method of Sec. (2.3) is thus provided.

2.4 Analysis of the Assumption of Constant Parameters

The analysis done previously assumes constant reactivity and source during a time step. Taking them equal to their average value during the time interval h yields the smallest error. To see how good an approximation this is, and to support the statements made near the end of the previous section, an order of magnitude error analysis is now undertaken. Consider the following exact expansion for $\underline{\Psi}(t+h) = \underline{\Psi}_1$ in terms of $\underline{\Psi}(t) = \underline{\Psi}_0$:

$$\underline{\Psi}_1^{ex} = \underline{\Psi}_0 + h\dot{\underline{\Psi}}_0 + \frac{h^2}{2!}\ddot{\underline{\Psi}}_0 + \frac{h^3}{3!}\dddot{\underline{\Psi}}_0 + \cdots \quad . \tag{10}$$

By repeated use of $\dot{\underline{\Psi}}_0 = A_0\underline{\Psi}_0 + \underline{f}_0$, Eq. (10) becomes

$$\underline{\Psi}_1^{ex} = [I+hA_0+\frac{h^2}{2!}(A_0^2+\dot{A}_0)+\frac{h^3}{3!}(A_0^3+2\dot{A}_0A_0+A_0\dot{A}_0+\ddot{A}_0)+\ldots]\underline{\Psi}_0 + + [h\underline{f}_0+\frac{h^2}{2!}(\dot{\underline{f}}_0+A_0\underline{f}_0)+\frac{h^3}{3!}(\ddot{\underline{f}}_0+A_0\dot{\underline{f}}_0+2\dot{A}_0\underline{f}_0+A_0^2\underline{f}_0)+\ldots]. \tag{11}$$

With the assumption of constant parameters we obtain the solution

$$\underline{\Psi}_1^{ap} = e^{h\bar{A}}\underline{\Psi}_0 + \bar{A}^{-1}(e^{h\bar{A}}-I)\underline{\bar{f}},$$

as indicated in Eq. (2). Here, the bar indicates an average over the time step.

By using the series expansion for $e^{h\bar{A}}$ we then get

$$\underline{\Psi}_1^{ap} = [I+h\bar{A}+\frac{h^2}{2!}\bar{A}^2+\frac{h^3}{3!}\bar{A}^3+\ldots]\underline{\Psi}_0 + h[I+\frac{h}{2!}\bar{A}+\frac{h^2}{3!}\bar{A}^2+\ldots]\underline{\bar{f}}. \tag{12}$$

But

$$\bar{A} = \frac{1}{h}\int_{t_0}^{t_0+h} A(t)dt = A_0 + \frac{h}{2!}\dot{A}_0 + \frac{h^2}{3!}\ddot{A}_0+\ldots,$$

and

$$\bar{f} = \frac{1}{h}\int_{t_0}^{t_0+h} f(t)dt = f_0 + \frac{h}{2!}\dot{f}_0 + \frac{h^2}{3!}\ddot{f}_0+\ldots \quad .$$

Substituting these expansions into Eq. (12) gives us

$$\underline{\Psi}_1^{ap} = [I+hA_0+\frac{h^2}{2!}(A_0^2+\dot{A}_0)+\frac{h^3}{3!}(A_0^3+\frac{3}{2}\dot{A}_0A_0+\frac{3}{2}A_0\dot{A}_0+\ddot{A}_0)+\ldots]\underline{\Psi}_0 + + [h\underline{f}_0+\frac{h^2}{2!}(\dot{\underline{f}}_0+A_0\underline{f}_0)+\frac{h^3}{3!}(\ddot{\underline{f}}_0+\frac{3}{2}A_0\dot{\underline{f}}_0+\frac{3}{2}\dot{A}_0\underline{f}_0+A_0^2\underline{f}_0)+\ldots].$$

Then, from this result and Eq. (11) we obtain

$$\underline{\Psi}_1^{ex} - \underline{\Psi}_1^{ap} = \frac{h^3}{12}[\dot{A}_0A_0-A_0\dot{A}_0]\underline{\Psi}_0 + \frac{h^3}{12}[\dot{A}_0\underline{f}_0-A_0\dot{\underline{f}}_0] + 0(h^4).$$

Note that this expression for the error still holds if A(t) and f(t) are known only through a three-term Taylor expansion, which requires only knowledge of A, f and their first and second derivatives at the beginning of the time step.

Thus, the assumption of constant parameters has a local error of $0(h^3)$ as stated in Sec. (2.3) and, correspondingly, a global error of $0(h^2)$. This analysis is significant only for small h. It has been found that, in some practical cases where $\rho<\beta$, the error in n(t) is essentially 0(h) for h sufficiently larger than $1/\omega_I$.

Chapter 3

SOLUTION OF THE SPACE-DEPENDENT KINETICS EQUATIONS

3.1 The Space-Dependent Kinetics Equations

For problems in which changes in the shape of the neutron flux can no longer be neglected, the need arises to adopt a model that includes space and energy dependence. The multigroup diffusion model has been found adequate to represent the spatial effects for most practical problems. The more elaborate transport model has not been widely used for time dependent calculations, so far, due to its computational complexity.

In this chapter, we shall direct our attention to an implicit solution technique for the diffusion model, based on a method known as the "theta method", which we shall describe in Sec. (3.4). The properties of this method are considered in Sec. (3.6).

The model assumed to represent the space-time behavior of the neutron flux in a reactor is described by the time dependent multigroup diffusion equations and the associated delayed neutron precursor equations. In matrix form these equations are

$$
\begin{aligned}
&\vec{\nabla}\cdot D\vec{\nabla}\underline{\phi} - \Sigma\underline{\phi} + \sum_{i=1}^{I} \lambda_i \underline{\chi}_i C_i = V_0^{-1}\,\dot{\underline{\phi}} \\
&\beta_i \underline{f}^T\underline{\phi} - \lambda_i C_i = \dot{C}_i \quad , \qquad (i=1,2,\ldots,I)
\end{aligned}
\qquad (1)
$$

where

$$D = \mathrm{diag}[D_g(\vec{r},t)], \quad V_0 = \mathrm{diag}[v_g], \quad \Sigma = [\Sigma_{gg'}(\vec{r},t)],$$

are G×G square matrices and

$$\underline{f} = \mathrm{col}[\nu\Sigma_{fg}(\vec{r},t)], \quad \underline{\phi} = \mathrm{col}[\phi_g(\vec{r},t)], \quad \underline{\chi}_i = \mathrm{col}[\chi_{ig}],$$

are G-vectors, with the following notation:

g = index number for a neutron energy group

i = index number for a delayed neutron group

G = total number of neutron energy groups

I = total number of delayed neutron groups

ϕ_g = total scalar flux in group g

C_i = $C_i(\vec{r},t)$ = precursor concentration for group i

v_g = characteristic neutron velocity for group g

D_g = diffusion constant for group g

$\Sigma_{gg'} = \Sigma_{Tg}(\vec{r},t)\delta_{gg'} - \Sigma_{sgg'}(\vec{r},t) - (1-\beta)\chi_{0g}\nu\Sigma_{fg'}(\vec{r},t)$

Σ_{Tg} = total macroscopic cross section for group g

$\delta_{gg'}$ = Kroenecker delta

$\Sigma_{sgg'}$ = macroscopic scattering cross section from group g' to group g

$\nu\Sigma_{fg}$ = neutrons per fission × macroscopic fission cross section for group g

χ_{0g} = fractional yield of neutrons in group g, per prompt neutron

χ_{ig} = fractional yield of neutrons in group g, per decay of precursor i

β_i = fractional yield of delayed neutrons in group i, per fission

$\beta \quad = \sum_{i=1}^{I} \beta_i$ = total fractional yield of delayed neutrons, per fission

$\lambda_i \quad$ = decay constant for precursor i.

We are assuming only one fissionable isotope and no external source present, for simplicity. However, extension of the method and its formalism for more than one fissionable isotope and an external source presents no particular difficulty.

Equation (1) can be written in the general matrix form

$$\frac{d}{dt}\,\underline{\Psi} = A(\vec{r},t)\,\underline{\Psi}(\vec{r},t) \quad , \tag{2}$$

where $\underline{\Psi} = \mathrm{col}[\underline{\phi}\ C_1\ C_2\ \ldots\ C_I]$, and

$$A = \begin{bmatrix} V_0(\vec{\nabla}\cdot D\vec{\nabla}-\Sigma) & \lambda_1 V_0 \underline{\chi}_1 & \lambda_2 V_0 \underline{\chi}_2 & \cdots & \lambda_I V_0 \underline{\chi}_I \\ \beta_1 \underline{f}^T & -\lambda_1 & 0 & \cdots & 0 \\ \beta_2 \underline{f}^T & 0 & -\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_I \underline{f}^T & 0 & 0 & \cdots & -\lambda_I \end{bmatrix} .$$

3.2 Spatial Discretization

Our main interest in this work is the study of a method to treat the time dependence. Therefore, we shall consider only the one-dimensional case, in which the position vector $\vec{r}$ is replaced by the coordinate r. The space dependence is treated by a simple finite-difference method. To do this, we divide the reactor in regions delimited by points r_k in the r-axis and use the box integration procedure[25] and thus obtain the finite-difference equations

$$B_k'\underline{\phi}_{k-1} + B_k\underline{\phi}_k + B_k''\underline{\phi}_{k+1} + \sum_{i=1}^{I} \lambda_i\underline{\chi}_i C_{ik} = V_0^{-1}\dot{\underline{\phi}}_k$$

$$\beta_i\underline{f}_k^T\underline{\phi}_k - \lambda_i C_{ik} = \dot{C}_{ik}\ , \quad (i=1,2,\ldots,I) \tag{3}$$

for k = 1,2,...,K, where K is the total number of inner mesh points and

$$\underline{\phi}_k = \underline{\phi}(r_k,t)\ ; \qquad C_{ik} = C_i(r_k,t)\ ;$$

$$B_{k+1}' = B_k'' = \frac{r_{k+1/2}^q}{\Delta r_k \Delta v_k} D(r_{k+1/2},t)\ ; \quad B_k = -[B_k'+B_k''+\Sigma_k]\ ;$$

$$\Sigma_k = \frac{r_k^q}{2\Delta v_k}[\Sigma(r_k^+,t)\Delta r_k + \Sigma(r_k^-,t)\Delta r_{k-1}]\ ;$$

$$\underline{f}_k = \frac{r_k^q}{2\Delta v_k}[\underline{f}(r_k^+,t)\Delta r_k + \underline{f}(r_k^-,t)\Delta r_{k-1}]\ ;$$

$$\Delta r_k = r_{k+1}-r_k\ ; \quad \Delta v_k = \frac{r_k^q}{2}(\Delta r_k+\Delta r_{k-1})\ ; \quad r_{k+1/2} = r_k+\tfrac{1}{2}\Delta r_k\ .$$

We have q = 0, 1, and 2, for slab, cylindrical and spherical geometry, respectively. The + and - superscripts denote right and left limits of functions of r at r_k, which may be different at material interfaces.

Special relations apply to the boundary points r_0 and

r_{K+1}, which are derived from the boundary conditions at these points. These boundary conditions are usually homogeneous and of the Neumann or Dirichlet type.

We now group the elements of Eq. (3) in larger vectors and matrices by defining

$$\underline{\phi} = \begin{bmatrix} \underline{\phi}_1 \\ \underline{\phi}_2 \\ \vdots \\ \underline{\phi}_K \end{bmatrix}, \quad \underline{C}_i = \begin{bmatrix} C_{i1} \\ C_{i2} \\ \vdots \\ C_{iK} \end{bmatrix}, \quad V = \begin{bmatrix} V_0 & & & \\ & V_0 & & \\ & & \ddots & \\ & & & V_0 \end{bmatrix},$$

$$L = \begin{bmatrix} B_1 & B_1'' & & \\ B_2' & B_2 & B_2'' & \\ & \ddots & \ddots & \ddots B_{K-1}'' \\ & & B_K' & B_K \end{bmatrix}$$

$$\chi_i = \begin{bmatrix} \underline{\chi}_i & & & \\ & \underline{\chi}_i & & \\ & & \ddots & \\ & & & \underline{\chi}_i \end{bmatrix}, \quad F = \begin{bmatrix} \underline{f}_1^T & & & \\ & \underline{f}_2^T & & \\ & & \ddots & \\ & & & \underline{f}_K^T \end{bmatrix}.$$

With these definitions, Eq. (3) becomes

$$L\underline{\phi} + \sum_{i=1}^{I} \lambda_i \chi_i \underline{C}_i = V^{-1}\dot{\underline{\phi}} \tag{4-a}$$

$$\beta_i F^T \underline{\phi} - \lambda_i \underline{C}_i = \dot{\underline{C}}_i \qquad (i=1,2,\ldots,I). \tag{4-b}$$

Note that $\underline{\phi}$ is a GK-vector and $\underline{C}_i$ is a K-vector. L and V are GK×GK matrices, χ_i is a GK×K matrix, and F is a K×GK matrix. The blocks of L are G×G matrices, its off-diagonal blocks being diagonal. In the form shown above for L, we tacitly assumed the boundary conditions $\underline{\phi}_0 = \underline{\phi}_{K+1} = 0$, since all test cases will be for slab reactors and most of them will involve asymmetric transients.

In what follows, we shall prove several properties of the method adopted for the solution of Eqs. (4). Use of an even more compact notation will greatly facilitate such a task. Therefore, we write Eqs. (4) in the form

$$\frac{d}{dt}\underline{\Psi} = H(t)\ \underline{\Psi}(t) \tag{5}$$

where

$$\underline{\Psi} = \begin{bmatrix} \underline{\phi} \\ \underline{C}_1 \\ \underline{C}_2 \\ \vdots \\ \underline{C}_I \end{bmatrix} \qquad H = \begin{bmatrix} VL & \lambda_1 V\chi_1 & \lambda_2 V\chi_2 & \cdots & \lambda_I V\chi_I \\ \beta_1 F & -\lambda_1 I & 0 & \cdots & 0 \\ \beta_2 F & 0 & -\lambda_2 I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_I F & 0 & 0 & \cdots & -\lambda_I I \end{bmatrix} .$$

Note that the symbols $\underline{\Psi}$ and $\underline{\phi}$ are now supervectors that represent the vectors $\underline{\Psi}(\vec{r},t)$ and $\underline{\phi}(\vec{r},t)$ of Eq. (2) for all discrete points r_k. The use of the same symbols is intended to show the analogy between them, their precise meaning being clear from the context.

3.3 Survey of Solution Methods

The numerical solution of the kinetics equations is made very difficult by the fact that they form a "stiff" system, namely the characteristic times (or eigenvalues) involved span several orders of magnitude.[26] This eliminates from consideration explicit methods based on Taylor series expansions in the time variable.

Several practical methods of solution for Eq. (2) exist, most based on its finite-difference formulation, given by Eq. (5) or by a higher-dimensional version of it. Here we mention a few of the most important.

To begin with, there are the nodal methods,[27] in which the reactor is divided into a few large regions, with some form of coupling between them specified. Another class of methods are the modal methods, in which the flux is expanded in a set of known modes or functions with coefficients determined by a variety of methods such as weighted residuals, least-squares or variational principles.[28,29] Synthesis techniques are also widely used. They are actually special cases of the modal approach and emcompass a variety of schemes like time synthesis,[7] space-time synthesis,[30] and multichannel space-time systhesis.[31] Quasi-static methods[6] also form an important family of methods. These may be viewed as including the adiabatic and the point kinetics models, as special cases.

Finally we have the pure finite-difference methods. These lead to several different procedures for solving Eq. (5). Among them are the matrix decomposition (GAKIN) method,[32,33] alternating-direction methods,[34-36] for two and three dimensions, and the "theta" differencing (WIGLE, TWIGL) method.[1,2,37,38] As mentioned before, it is to this last method that we shall direct our attention in the present work.

A good survey of the above methods, as well as others, is provided by Ref. (39). It includes both descriptions and references.

3.4 Derivation of the Theta-Differenced Equations

The theta method is an implicit method involving the use of special parameters (thetas) intended to increase the accuracy of the difference equations for the time step at hand. It can be viewed as a generalization of the Crank-Nicholson method and includes, as special cases, both the totally explicit and the totally implicit methods. Being in general implicit, it involves inversion of large matrices, which, in turn, requires relatively large amounts of computation time, per time step. On the other hand, if appropriate values of the thetas can be found, the method is able to take comparatively large time steps with a reasonably good accuracy. One purpose of the present thesis is to improve the current methods[1,2,37,38] by which thetas are found. We shall attempt to find some efficient systematic procedure for generating these parameters, which will be applicable under all transient conditions.

The equations for the "theta method" can be derived by integrating both sides of Eqs. (4) from t to t+h, thus obtaining

$$\underline{V}^{-1}(\underline{\phi}_1-\underline{\phi}_0) = \int_0^h L(t+s)\underline{\phi}(t+s)\,ds + \sum_{i=1}^{I}\lambda_i\chi_i\int_0^h \underline{C}_i(t+s)\,ds$$

$$\underline{C}_{i1} - \underline{C}_{i0} = \beta_i\int_0^h F(t+s)\underline{\phi}(t+s)\,ds - \lambda_i\int_0^h \underline{C}_i\,ds,\quad (i=1,2,\ldots,I) \tag{6}$$

where the subscripts 0 and 1 refer to functions at t and t+h, respectively. Then the approximations

$$\frac{1}{h}\int_0^h L(t+s)\underline{\phi}(t+s)\,ds = \theta_0 L_1\underline{\phi}_1 + (1-\theta_0)L_0\underline{\phi}_0$$

$$\frac{1}{h}\int_0^h F(t+s)\underline{\phi}(t+s)\,ds = \theta_0 F_1\underline{\phi}_1 + (1-\theta_0)F_0\underline{\phi}_0 \tag{7}$$

$$\frac{1}{h}\int_0^h \underline{C}_i(t+s)\,ds = \theta_i\underline{C}_{i1} + (1-\theta_i)\underline{C}_{i0}\qquad (i=1,2,\ldots,I)$$

are made, where the thetas will be determined so as to yield a good solution under the condition that $0<\theta_i\leq 1\,(i=0 \text{ to } I)$.

Substituting the approximations (7) in Eq. (6) yields

$$\frac{1}{h}V^{-1}(\underline{\phi}_1-\underline{\phi}_0) = [\theta_0 L_1\underline{\phi}_1+(1-\theta_0)L_0\underline{\phi}_0] + \sum_{i=1}^{I}\lambda_i\chi_i[\theta_i\underline{C}_{i1}+(1-\theta_i)\underline{C}_{i0}] \tag{8-a}$$

$$\frac{1}{h}(\underline{C}_{i1}-\underline{C}_{i0}) = \beta_i[\theta_0 F_1\underline{\phi}_1+(1-\theta_0)F_0\underline{\phi}_0] - \lambda_i[\theta_i\underline{C}_{i1}+(1-\theta_i)\underline{C}_{i0}]. \tag{8-b}$$

In principle, Eqs. (8) would yield the exact solution, if we used the proper thetas, one for each space point and neutron energy group. However, knowledge of such thetas would imply knowledge of the exact solution, which is already what we are looking for. Thus approximations have to be made in order to implement Eqs. (8). Specifically, we assume a single theta parameter for all spatial points and, for the flux, the same parameter also for all neutron energy groups.

In the WIGLE approach[1,37] a different theta is used for each neutron energy group, but only one theta is assumed for all precursors, namely $\theta_i=\theta_d$ $(i=1,2,...,I)$. The thetas are assumed to be space-independent and are estimated from a two-group point calculation (WIGLE-type codes are limited to two energy groups). Drastic approximations are made in the point equations, in order to compute these thetas.

In TWIGL,[2,38] the thetas for the precursors are all chosen as $\theta_i=1/2$ $(i=1,2,...,I)$ and θ_0 is left as in input parameter, the same for all energy groups and space points.

In the present work, the idea is to use an independent θ_i for each precursor and the flux, since they have distinctly different characteristic times. As mentioned before, the same theta is used for all space points (as in WIGLE and TWIGL) and, in the case of the flux, also for all neutron energy groups. It is felt that, since the energy coupling is stronger than the spatial coupling, at least for thermal reactors, there is no justification in taking a different integration parameter for each energy group while using only one for all the points of space.

We shall examine, in this thesis, the feasibility of using an accurate point kinetics calculation as a means of finding the thetas so that Eqs. (8) will be accurate in an integral sense. The details of this procedure are spelled out in the next section.

To get the time advancement algorithm we first define the total (prompt + delayed) net production operator

$$M(t) = L(t) + \sum_{i=1}^{I} \beta_i \chi_i F(t) \tag{9}$$

Solving Eq. (8-b) for $\underline{C}_{i1}$ and substituting in Eq. (8-a) yields to new equations,

$$[M_1 - \frac{1}{h\theta_0} V^{-1} - \sum_{i=1}^{I} \beta_i \chi_i \frac{\theta_i'}{\theta_0} F_1] \underline{\phi}_1 + [\frac{1-\theta_0}{\theta_0} M_0 + \frac{1}{h\theta_0} V^{-1} -$$

$$- \sum_{i=1}^{I} \beta_i \chi_i \frac{\tau_i - \theta_i'}{\theta_0} F_0] \underline{\phi}_0 + \sum_{i=1}^{I} \lambda_i \frac{\tau_i}{\theta_0} \chi_i \underline{C}_{i0} = 0 \tag{10-a}$$

$$\underline{C}_{i1} = (1-h\lambda_i \tau_i) \underline{C}_{i0} + h\beta_i \theta_i' F_1 \underline{\phi}_1 + h\beta_i (\tau_i - \theta_i') F_0 \underline{\phi}_0 \ , \tag{10-b}$$

where

$$\tau_i = \frac{1}{1+h\lambda_i \theta_i}$$
$$\theta_i' = \frac{\theta_0}{1+h\lambda_i \theta_i} \ . \tag{11}$$

This procedure will henceforth be called Method 1.

We can also take a slightly different approach which we shall call Method 2. Instead of first approximating the equations and then solving for $\underline{C}_{i1}$ we can first solve Eq. (4-b) for $\underline{C}_i$, substitute in Eq. (4-a), and then integrate, thus obtaining

$$V^{-1}(\underline{\phi}_1-\underline{\phi}_0) = \int_0^h M(t+s)\underline{\phi}(t+s)\,ds - \sum_{i=1}^{I}\beta_i\chi_i \int_0^h e^{-\lambda_i(h-s)}\cdot$$

$$\cdot F(t+s)\underline{\phi}(t+s)\,ds + \sum_{i=1}^{I}\chi_i(1-e^{-h\lambda_i})\underline{C}_{i0}$$

$$\underline{C}_{i1} = e^{-h\lambda_i}\underline{C}_{i0} + \beta_i \int_0^h e^{-\lambda_i(h-s)} F(t+s)\underline{\phi}(t+s)\,ds, \tag{12}$$

which we call the precursor integrated equations.

We now make the approximations

$$L(t+s)\underline{\phi}(t+s) = \theta(s)L_1\underline{\phi}_1 + [1-\theta(s)]L_0\underline{\phi}_0$$

$$F(t+s)\underline{\phi}(t+s) = \theta(s)F_1\underline{\phi}_1 + [1-\theta(s)]F_0\underline{\phi}_0 \tag{13}$$

for $0\leq s\leq h$, where $\theta(s)$ is a space-independent function to be determined so as to yield good results, subject to the conditions $\theta(0)=0$, $\theta(h)=1$, and $0\leq\theta(s)\leq 1$ in the above interval for s. Taking $\theta(s)=s/h$ means that all reaction rates are assumed to vary linearly between their initial and final values. This approach has been used in Ref. (47).

By introducing Eq. (13) in Eq. (12) and carrying out the integrations we get again Eqs. (10) where now

$$\tau_i = \frac{1}{h}\int_0^h e^{-\lambda_i(h-s)}\,ds = \frac{1-e^{-h\lambda_i}}{h\lambda_i}$$

$$\theta_i' = \frac{1}{h}\int_0^h \theta(s)e^{-\lambda_i(h-s)}\,ds \tag{14}$$

$$\theta_0 = \frac{1}{h}\int_0^h \theta(s)\,ds.$$

We impose an additional condition for $\theta(s)$, namely that $0<\theta_0\leq 1$.

As in Method 1, we shall, in the next section, try to apply point kinetics to find $\theta(s)$ or directly the parameters of Eq. (14).

Note that we have from either Eq. (11) or (14) the following results

$$\begin{aligned} \theta_i' &< \theta_0 \\ \theta_i &\leq \tau_i \end{aligned} \tag{15}$$

as can easily be inferred from their definitions and the restrictions placed on the thetas. These results will be useful later.

From Eqs. (11) and (14) it can be shown that the two methods are identical if

$$\frac{1}{h}\int_0^h \theta(s)\ e^{\lambda_i s}\ ds = [\frac{1}{h}\int_0^h \theta(s)ds]\,[\frac{1}{h}\int_0^h e^{\lambda_i s}ds]\ , \tag{16}$$

provided θ_0 is the same for both methods and provided we take

$$\theta_i = \frac{1}{(-h\lambda_i)} - \frac{1}{e^{(-h\lambda_i)}-1}\ .$$

Equation (16) holds exactly only for constant $\theta(s)$, but it becomes an increasingly better approximation as $h\lambda_i$ becomes small.

3.5 Determination of the Parameters Theta

As indicated in the previous section, a possible way to get a set $\{\theta_i\}$ or a function $\theta(s)$ is through a corresponding point kinetics calculation, which makes Eqs. (8) or (12) "exact" in an integral sense.

The point kinetics problem is obtained by reducing all space-energy dependent quantities to their point equivalent counterparts. This is done simply by multiplying the quantities of interest by a weight function of space and energy and integrating over the domain of interest of these variables. In the multigroup, space-discretized formulation such a procedure corresponds to a scalar product of a weight vector $\underline{w}$ and the vector quantity under consideration. We thus define

$$n(t) = \underline{w}^T V^{-1} \underline{\phi} , \qquad c_i = \underline{w}^T \chi_i \underline{C}_i ,$$
$$\alpha(t) = \frac{1}{n} \underline{w}^T L \underline{\phi} , \qquad \mu_i(t) = \frac{\beta_i}{n} \underline{w}^T \chi_i F \underline{\phi} , \tag{17}$$

where $\underline{w}$ is a GK-vector given by

$$\underline{w} = \mathrm{col}[\underline{w}_1 \; \underline{w}_2 \; \cdots \; \underline{w}_K]$$

with the G-vectors

$$\underline{w}_k = \Delta v_k \times \mathrm{col}[w_{1k} \; w_{2k} \; \cdots \; w_{Gk}] .$$

With this choice of $\underline{w}$, the scalar products in Eq. (17) correspond to a trapezoidal integration in space. One choice for the elements w_{gk} can be the positive solution

$\phi_g^*(r_k)$ of the equation

$$M^T(0)\ \underline{\phi}^* = 0$$

which is the discretized equation for the adjoint in the initial steady-state. More simply we could also take $\underline{w}$ = constant. Other possibilities would be to use $\underline{w} = \underline{\phi}(t)$ itself or to use at every time the adjoint of the perturbed state, made critical by dividing the matrix F by the appropriate scalar. This last procedure would probably yield the best results but would be too time-consuming.

An approximation has to be introduced at this point in order to calculate $\alpha(t)$ and $\mu(t)$. We assume that the <u>shape</u> of the flux $\underline{\phi}$ remains constant during the time step h, and is given by its value at the beginning of that time step.

Note that this approximation of constant shape is used only to calculate the thetas and does not imply a similar assumption for the space-dependent calculations. We thus have that

$$\alpha(t) = \frac{1}{n(t)}\ \underline{w}^T L(t)\underline{\phi}_0\ , \qquad \mu_i(t) = \frac{\beta_i}{n(t)}\ \underline{w}^T \chi_i F(t)\underline{\phi}_0\ .$$

With these definitions, Eq. (5) will reduce to the point kinetics Eq. (2.1). This equation can be solved by the method of Chap. 2 yielding n_1 and c_{i1} as functions of n_0 and c_{i0}.

Introducing Eqs. (17) into the theta-differenced Eqs. (8) yields

$$\frac{n_1-n_0}{h} = [\theta_0\alpha_1 n_1+(1-\theta_0)\alpha_0 n_0] + \sum_{i=1}^{I}\lambda_i[\theta_i c_{i1}+(1-\theta_i)c_{i0}]$$

$$\frac{c_{i1}-c_{i0}}{h} = [\theta_0\mu_{i1}n_1+(1-\theta_0)\mu_{i0}n_0] - \lambda_i[\theta_i c_{i1}+(1-\theta_i)c_{i0}]. \tag{18}$$

Calculating the n_1 and c_{i1} from Eq. (2.1) and using these values in Eq. (18) allows us to find the desired theta parameters. A detailed account of this procedure is given in Appendix B. For the second approach, using the precursor integrated equations, we assume that $L\underline{\phi}$ and $F\underline{\phi}$ go from their initial to final values at the same rate with which the amplitude function n(t) foes from n_0 to n_1. Thus, $\theta(s)$ is such that

$$n(t+s) = \theta(s)n_1 + [1-\theta(s)]n_0 ,$$

from which we get

$$\theta(s) = \frac{n(t+s) - n_0}{n_1 - n_0} . \tag{19}$$

The function n(t) is obtained by solving Eq. (2.1). With this $\theta(s)$ we can then find the parameters θ_0 and θ_i' of Eq. (14) (see Appendix B for a detailed description of the procedure used.)

Of course the thetas may also be chosen without the use of a point kinetics formulation. For instance we may simply pick any of the thetas to be $\theta_i=1$ or $\theta_i=\frac{1}{2}$ or

$$\theta_i = \frac{1}{h\omega_0} - \frac{1}{e^{h\omega_0}-1} \quad , \tag{20}$$

if the reactor is on an asymptotic period ω_0. Also $\theta(s)$ could be picked as $\theta(s) = s/h$ or $\theta(s) = 1$ (step function) or, in the asymptotic period case,

$$\theta(s) = \frac{e^{s\omega_0} - 1}{e^{h\omega_0} - 1} \quad . \tag{21}$$

3.6 Properties of the Solution Method

In this section, we shall analyze several properties of the two methods of solution, as presented in Sec. (3.4). The methods will first be shown to be consistent and stable, for the case in which the reactor parameters are constant in time.

Consistency and stability are necessary and sufficient conditions for the convergence of the finite-difference solution, by Lax's theorem.[40] These two properties have been shown to hold for Method 1, in the case of constant spatial mesh[41] and can easily be extended to Method 2 for such a case. Here we show that they also hold when the spatial mesh size is shrunk to zero, together with the time step size, provided that

$$\frac{h}{(\Delta r)^2} = \text{const.} \;, \qquad\qquad \theta_0 \geq \frac{1}{2} \quad .$$

The two methods will also be shown to conserve the steady-state for any thetas, and to be asymptotically exact, if asymptotic thetas are used. Asymptotic stability is also examined. Finally, the solvability of the time advancement equations (10) will be analyzed.

In the case of constant spatial mesh it has been shown [41] that the global truncation error for Method 1 is $0(h)$ for a general theta matrix and is $0(h^2)$ if $\Theta = \frac{1}{2}I$. Similarly, for Method 2 it can be readily shown that the global truncation error is $0(h)$ for a general $\theta(s)$ and $0(h^2)$ if $\theta(s)$ is such that $\theta_0 = \frac{1}{2}$.

The proofs of consistency and stability will be carried in the L_2 norm given by

$$\|\underline{u}(r)\| = [\int_V |\underline{u}(r)|^2 dV]^{1/2} .$$

For functions defined only at certain discrete points r_k, some interpolation scheme is assumed for the intermediate points. For an operator A the L_2 norm is defined as

$$\|A\| = \max_{\|\underline{u}\|=1} \|A\underline{u}\| .$$

In the Appendix C we show that, for a discrete operator or matrix H,

$$k\|H\|' \leq \|H\| \leq K\|H\|' ,$$

where $\| \; \|'$ is the Euclidean norm,[23] and k, K are fixed, positive constants. The above relation means that the L_2 norm and the Euclidean norm of a matrix are equivalent so that all properties of boundedness of matrices in one norm hold for the other norm.

When stating that certain matrices have bounded norms we shall be tacitly using the following

THEOREM:[34] A family of matrices H_N of varying dimension N having at most n<N nonzero elements in each row or column, n being constant for all N, has a uniformly bounded Euclidean norm if the individual elements of the matrices H_N are uniformly bounded for all N.

It will be convenient to rederive the iteration algorithms in a more compact form to carry out the proofs that follow. To this end we consider H as composed of two terms

$$H(t) = Q + R(t) \ . \tag{22}$$

We then write Eq. (5) in the form

$$\frac{d}{dt}\underline{\Psi} - Q\underline{\Psi}(t) = R(t)\underline{\Psi}(t) \ ,$$

a solution of which is given by

$$\underline{\Psi}_1 = e^{hQ}\underline{\Psi}_0 + \int_0^h e^{(h-s)Q} R(t+s)\underline{\Psi}(t+s)ds \ , \tag{23}$$

where the subscripts 0 and 1 denote quantities evaluated at t and t+h, respectively.

For Method 1 we have Q = 0 and R = H, and we make the approximation

$$\frac{1}{h}\int_0^h R(t+s)\underline{\Psi}(t+s)ds = \Theta R_1\underline{\Psi}_1 + (I-\Theta)R_0\underline{\Psi}_0 \tag{24}$$

of Eqs. (7), where

$$\Theta = \begin{bmatrix} \theta_0 I & & & & \\ & \theta_1 I & & & \\ & & \theta_2 I & & \\ & & & \ddots & \\ & & & & \theta_I I \end{bmatrix}, \tag{25}$$

with the first I being GK×GK and the others being K×K.

For Method 2, in order to get Eq. (12), we must have

$$Q = \begin{bmatrix} 0 & \lambda_1 V\chi_1 & \lambda_2 V\chi_2 & \cdots & \lambda_I V\chi_I \\ 0 & -\lambda_1 I & 0 & \cdots & 0 \\ 0 & 0 & -\lambda_2 I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\lambda_I I \end{bmatrix},$$

(26)

$$R = \begin{bmatrix} VL & 0 & 0 & \cdots & 0 \\ \beta_1 F & 0 & 0 & \cdots & 0 \\ \beta_2 F & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_I F & 0 & 0 & \cdots & 0 \end{bmatrix},$$

and the approximation is made that (see Eq. (13))

$$R(t+s)\underline{\Psi}(t+s) = \theta(s) R_1 \underline{\Psi}_1 + [1-\theta(s)] R_0 \underline{\Psi}_0. \tag{27}$$

Introducing either of these approximations, Eq. (24) or (27), in Eq. (23) yields

$$\begin{aligned} \underline{\Psi}_1 &= [I - h\Theta R_1]^{-1}[e^{hQ} + h(\Lambda-\Theta)R_0]\underline{\Psi}_0 \\ &= E(h)\ \underline{\Psi}_0 \ , \end{aligned} \tag{28}$$

where E(h) is the time advancement matrix. For Method 1, Θ is given by Eq. (25) and for Method 2, we have

$$\Theta = \frac{1}{h}\int_0^h e^{(h-s)Q}\ \theta(s)ds \quad . \tag{29}$$

For both methods, Λ is given by

$$\Lambda = \frac{1}{h}\int_0^h e^{(h-s)Q}\ ds, \tag{30}$$

which reduces to $\Lambda=I$ for Method 1. Equation (28) is equivalent to the system of Eqs. (10), as can be verified for both methods. From Eq. (30) we can readily derive the identity

$$e^{hQ} - I - h\Lambda Q = 0 \tag{31}$$

valid for any Q.

Equation (28) will be the one used for the demonstration of the several properties of both solution methods presented here. We shall assume constant properties during a time step in most of the remaining of this section either for simplicity, to show consistency and stability, or by necessity, when analyzing steady-state and asymptotic behavior.

3.6.1 Consistency

The property of consistency simply implies that the truncation error of the finite-difference formulation tends to zero, as the time step size tends to zero. Mathematically we must have[40]

$$\lim_{h\to 0} \left\| \frac{\underline{\Psi}_1 - E(h)\underline{\Psi}_0}{h} \right\| = 0$$

for any $\underline{\Psi}(\vec{r},t)$ which is a genuine solution of Eq. (2). The convention is made that a matrix operating on a continuous function of r is actually operating on the corresponsing vector obtained from the function values at the discrete points r_k. From the properties of norms we have, using Eq. (28),

$$\left\| \frac{\underline{\Psi}_1 - E(h)\underline{\Psi}_0}{h} \right\| \leq \| [I-h\Theta R]^{-1} \| \cdot \left\| \frac{[I-h\Theta R]\underline{\Psi}_1 - [e^{hQ}+h(\Lambda-\Theta)R]\underline{\Psi}_0}{h} \right\| \quad .$$

We shall prove afterwards that

$$\lim_{h\to 0} \| [I - h\Theta R]^{-1} \|$$

exists, if $h/(\Delta r)^2$ = const as $h\to 0$.

Now, it remains to be shown that

$$\lim_{h\to 0} \left\| \frac{[I - h\Theta R]\underline{\Psi}_1 - [e^{hQ} + h(\Lambda-\Theta)R]\underline{\Psi}_0}{h} \right\| = 0 \ . \tag{32}$$

We have $\underline{\Psi}_1 = \underline{\Psi}_0 + h\dot{\underline{\Psi}}(t+\varepsilon h)$, for some ε in the interval [0,1]. Recalling that H = Q + R and using Eq. (31), the expression whose norm is being taken in Eq. (32) becomes

$$[I - h\Theta H + h\Theta Q]\dot{\underline{\Psi}}(t+\varepsilon h) - \Lambda H\underline{\Psi}_0.$$

Now we use Eq. (2) and the result $H = A + 0(\Delta r^n)$, where n is the order of accuracy of the spatial discretization, usually $n = 2$. Then the expression above becomes

$$[I - h\Theta A - 0(\Delta r^n)h\Theta + h\Theta Q]\dot{\underline{\Psi}}(t+\varepsilon h) - \Lambda\dot{\underline{\Psi}}_0 - 0(\Delta r^n)\Lambda\underline{\Psi}_0 =$$

$$= [I - 0(h)A + 0(h)]\dot{\underline{\Psi}}(t+\varepsilon h) - [I + 0(h)]\dot{\underline{\Psi}}_0 + 0(h^{n/2})\underline{\Psi}_0 ,$$

where we have used the facts that Θ and Q have bounded norms, $\Lambda = I+0(h)$ and $\Delta r^2/h = \text{const}$. From Eq. (2), $A\dot{\underline{\Psi}} = \ddot{\underline{\Psi}}$, for A constant in time. Also we have

$$\dot{\underline{\Psi}}(t+\varepsilon h) = \dot{\underline{\Psi}}_0 + \varepsilon h\ddot{\underline{\Psi}}(t+\varepsilon' h) ,$$

for some ε' in the interval $[0,\varepsilon]$. Using these two equations and the fact that $\underline{\Psi}$, $\dot{\underline{\Psi}}$, and $\ddot{\underline{\Psi}}$ have bounded L_2 norms in the interval $(t,t+h)$, we can reduce the expression under consideration to $0(h) + 0(h^{n/2})$. Thus Eq. (32) is proved.

We now show that $\|[I - h\Theta R]^{-1}\|$ is bounded as $h\to 0$. Let us write $R = R_D + R'$ where R_D contains only the diffusion terms (of the VL matrix) and R' contains all the other elements of R. Then

$$[I - h\Theta R]^{-1} = [I - (I-h\Theta R_D)^{-1}h\Theta R']^{-1}\,[I - h\Theta R_D]^{-1}$$

$$= [I - (I-h\theta_0 R_D)^{-1}h\Theta R']^{-1}\,[I - h\theta_0 R_D]^{-1} ,$$

since $\Theta R_D = \theta_0 R_D$.

The matrix R_D is real, symmetric, diagonally dominant, and has non-positive diagonal elements, therefore, the eigenvalues of R_D are all real and non-positive.[42] Thus, since the Euclidean norm of a symmetric matrix is equal to its spectral radius,[23]

$$\|[I - h\theta_0 R_D]^{-1}\|' = \rho([I-h\theta_0 R_D]^{-1}) = \max_n \left|\frac{1}{1-h\theta_0\gamma_n}\right| = 1 \quad ,$$

since zero is also an eigenvalue of R_D. Here, ρ denotes the spectral radius, and γ_n are the eigenvalues of R_D.

With this result we have

$$\|[I - h\Theta R]^{-1}\|' \leq \|[I - (I-h\theta_0 R_D)^{-1} \, h\Theta R']^{-1}\|' \leq \frac{1}{1-\beta}$$

where

$$\beta = \|[I - h\theta R_D]^{-1} \, h\Theta R'\|' \leq h\|\Theta R'\|' < 1,$$

for h sufficiently small, since Θ and R' are bounded as $h\to 0$.

Using the equivalence of the L_2 and the Euclidean norms we then conclude that $\|[I - h\Theta R]^{-1}\|$ is indeed bounded as $h\to 0$.

3.6.2 Stability

The property of stability assures us that the error in the solution will not grow without bounds as $h\to 0$. Recalling the equivalence of the L_2 and the Euclidean norms, a sufficient condition for stability is that[40]

$$\|E(h)\|' = 1 + 0(h) \quad .$$

We again factor R in two terms $R_D + R'$ as seen in Sec. (3.6.1), where R_D contains only the diffusion terms. Then we write

$$E(h) = [I - (I-h\Theta R_D)^{-1} h\Theta R']^{-1}\ [I - h\Theta R_D]^{-1} \cdot$$
$$\cdot [I + h(I-\Theta)R_D + (e^{hQ}-I) + h(\Lambda-\Theta)R' - h(I-\Lambda)R_D].$$

Now we use the following equations

$$e^{hQ} - I = 0(h)$$
$$h[\Lambda-\Theta]\ R' = 0(h)$$
$$h[I-\Lambda]\ R_D = 0(h^2)R_D = 0(h)\quad ,$$

which can be seen to hold since Q, Λ, Θ, R', and hR_D have bounded norms and $\|\Lambda-I\| = 0(h)$. Thus, since $[I-h\Theta R_D]^{-1}$ is also bounded, as shown in Sec. (3.6.1), we have

$$E(h) = [I-0(h)]^{-1}[I-h\theta_0 R_D]^{-1}[I+h(1-\theta_0)R_D+0(h)]$$
$$= [I-h\theta_0 R_D]^{-1}[I+h(1-\theta_0)R_D] + 0(h)\quad ,$$

so that

$$\|E(h)\|' = \|[I-h\theta_0 R_D]^{-1}[I+h(1-\theta_0)R_D]\|' + 0(h)$$

But since R_D is symmetric and has only real non-positive eigenvalues, including zero eigenvalues, as we have seen in Sec. (3.6.1), we may write

$$\|E(h)\|' = \rho([I-h\theta_0 R_D]^{-1}[I+h(1-\theta_0)R_D]) + 0(h) =$$
$$= \max_n \left|\frac{1+h(1-\theta_0)\gamma_n}{1-h\theta_0\gamma_n}\right| + 0(h) = 1 + 0(h)\quad ,$$

provided $\theta_0 \geq \frac{1}{2}$. Again, ρ denotes the spectral radius, and γ_n are the eigenvalues of R_D.

3.6.3 Steady-State Behavior

We will now show that, given steady-state properties and a steady-state initial solution, the method reproduces the initial solution. In symbols it will be shown that

$$\underline{\Psi}_1 = E(h)\underline{\Psi}_0 = \underline{\Psi}_0$$

for any thetas and any time step size. From Eq. (22) and the criticality condition $H\underline{\Psi}_0 = 0$ we have

$$R\underline{\Psi}_0 = -Q\underline{\Psi}_0 \quad ,$$

so that Eq. (28) becomes

$$\begin{aligned} \underline{\Psi}_1 &= [I - h\Theta R]^{-1}[e^{hQ} - h\Lambda Q - h\Theta R]\underline{\Psi}_0 \\ &= [I - h\Theta R]^{-1}[I - h\Theta R]\underline{\Psi}_0 = \underline{\Psi}_0 \quad , \end{aligned}$$

by use of Eq. (31).

3.6.4 Asymptotic Behavior

The solution methods under consideration will yield the exact asymptotic solution, when the corresponsing asymptotic thetas are used. We have, for an asymptotic reactor,

$$H\underline{\Psi}_0 = \omega_0\underline{\Psi}_0 \quad , \tag{33}$$

where ω_0 is the asymptotic period. We have to show that

$$\underline{\Psi}_1 = E(h)\underline{\Psi}_0 = e^{h\omega_0}\underline{\Psi}_0$$

for any h, and the proper Θ matrix. From Eqs. (33) and (22) we can write

$$R\underline{\Psi}_0 = [\omega_0 I - Q]\underline{\Psi}_0 .$$

Using this result and Eq. (31) in Eq. (28) we get

$$\begin{aligned}\underline{\Psi}_1 &= [I - h\Theta R]^{-1}[e^{hQ} + h\Lambda(\omega_0 I-Q) - h\Theta R]\underline{\Psi}_0 \\ &= e^{h\omega_0}\underline{\Psi}_0 + [I - h\Theta R]^{-1}[h\omega_0\Lambda + (1-e^{h\omega_0})\{I-h\Theta(\omega_0 I-Q)\}]\underline{\Psi}_0 .\end{aligned}$$

Now, it is sufficient to show that

$$[h\omega_0\Lambda + (1-e^{h\omega_0})\{I-h\Theta(\omega_0 I-Q)\}] = 0 . \qquad (34)$$

From Eqs. (20) and (25) we have for Method 1

$$\Theta = \{\frac{1}{h\omega_0} - \frac{1}{e^{h\omega_0}-1}\}I ,$$

for an asymptotic reactor, $Q = 0$, and $\Lambda = I$. Equation (34) is readily seen to hold. For Method 2, from Eqs. (21) and (29) we get

$$\Theta = \frac{1}{e^{h\omega_0}-1}\,[\frac{1}{h}(\omega_0 I-Q)^{-1}(e^{h\omega_0}I-e^{hQ}) - \Lambda] .$$

With this result and Eq. (31) we find that Eq. (34) is again true.

We can also show that, under certain conditions, Method 1 is asymptotically stable, meaning that the eigenvalue of largest magnitude of the advancement matrix E(h) corresponds to the non-negative eigenvector of H. This property can be demonstrated for the case of a scalar Θ matrix, $\Theta=\theta I$ in which case

$$E(h) = [I - h\theta H]^{-1}[I + h(1-\theta)H]$$

has the same eigenvectors as H. We assume further that the eigenvalues ω_n of H are real and all $\omega_n<\omega_0$ where ω_0 is the eigenvalue corresponding to the non-negative eigenvector of H. The eigenvalues of E(h) are

$$\nu_n = \frac{1 + h(1-\theta)\omega_n}{1 - h\theta\omega_n} .$$

We can infer from this expression that for a fixed θ, we have $|\nu_n| < \nu_0$ if

$$\frac{1}{2}[\frac{1}{\theta} - \frac{1}{1-\theta}] \leq h\omega_0 < \frac{1}{\theta} .$$

If we pick θ from Eq. (20) then $\nu_0 = e^{h\omega_0}$ and the condition for asymptotic convergence becomes

$$\left|\frac{1+h(1-\theta)\omega_n}{1-h\theta\omega_n}\right| < e^{h\omega_0} \tag{35}$$

for $\omega_n < \omega_0$.

The condition (35) holds for any $\omega_n<\omega_0$ if $\omega_0>0$, but fails for negative ω_0 if ω_n is of very large magnitude.

3.6.5 Solvability

Solution of the time advancement equation (28) requires the existence of the inverse matrix in that equation. Equivalently, we must be able to calculate a unique $\underline{\Psi}_1$ from the linear system of equations implied by Eq. (28). A necessary and sufficient condition for the existence of this inverse is then that

$$G = M_1 - \frac{1}{h\theta_0}V^{-1} - \sum_{i=1}^{I} \beta_i \chi_i \frac{\theta_i'}{\theta_0} F_1$$

has an inverse, as can be concluded from the equivalent Eqs. (10). Equation (10-a) can be written as

$$G\underline{\phi}_1 + \underline{S} = 0 \ , \tag{36}$$

which has the form of a steady-state equation with an external source. The operator G corresponds to M(t+h) given by Eq. (9), with an added absorption and a reduced delayed neutron production (since $\theta_i'/\theta_0 < 1$, from Eq. (15)).

To find conditions under which G has an inverse we use the properties of essentially positive matrices[43] which we now define: a real matrix P is essentially positive if it is irreducible and has non-negative off-diagonal elements. Such a matrix has a real eigenvalue π_0 which is algebraically larger than $\text{Re}(\pi_i)$ for any other eigenvalue π_i. To π_0 there corresponds a positive eigenvector which is the only non-negative eigenvector of the matrix. If any element of P is increased π_0 also increases.

Let us now, consider the eigenvalue problem

$$H_1\underline{u}_n = \omega_n\underline{u}_n \ . \tag{37}$$

The matrix H as defined after Eq. (5) is irreducible only if $\nu\Sigma_f \neq 0$ for all points. However, in case $\nu\Sigma_f = 0$ for a particular region of the reactor, we can always redefine H so as to make it always irreducible. This is done simply by deleting the k^{th} column from the matrices χ_i and the k^{th} row from the matrix F, whenever $\underline{f}_k = 0$ (see Sec. (3.2), for notation). Corresponding deletions are made in vectors on which H operates. Such a procedure has the effect of eliminating elements C_{ik} from the vectors $\underline{C}_i$, whenever $\underline{f}_k = 0$, and expresses the recognition that, physically, the precursor concentrations are zero for points at which no fission takes place. The above redefinition of H is tacitly assumed below.

An examination readily shows that H is an essentially positive matrix. Let ω_0 be its eigenvalue corresponding to $\underline{u}_0 > 0$, and let

$$\underline{u}_0 = \text{col}[\underline{u}_{00}\ \underline{u}_{01}\ \underline{u}_{02} \cdots \underline{u}_{0I}] \ .$$

Solving Eq. (37) with n = 0 for $\underline{u}_{0i}$ in terms of $\underline{u}_{00}$ gives

$$\underline{u}_{0i} = \frac{\beta_i}{\omega_0 + \lambda_i} F_1 \underline{u}_{00} \ .$$

Since $\underline{u}_{0i}$ and $\underline{u}_{00}$ have only positive elements, F_1 is a positive matrix, and $\beta_i > 0$, we must necessarily have $(\omega_0 + \lambda_i) > 0$, a result which will be implicitly assumed below. By using the expression above for $\underline{u}_{0i}$ we can solve Eq. (37)

with n = 0 for $\underline{u}_{00}$, obtaining

$$G'\underline{u}_{00} = [M_1 - \omega_0 V^{-1} - \sum_{i=1}^{I} \frac{\omega_0}{\omega_0+\lambda_i} \beta_i \chi_i F_1]\underline{u}_{00} = 0.$$

Upon inspection it can be seen that G' is an essentially positive matrix with zero as the eigenvalue corresponding to the positive eigenvector $\underline{u}_{00}$. The matrix G is also essentially positive. If

$$\omega_0 \leq \frac{1}{h\theta_0} \quad \text{and} \quad \frac{\omega_0}{\omega_0+\lambda_i} \leq \frac{\theta_i'}{\theta_0} \quad (i=1,2,\ldots,I) \qquad (38)$$

with at least one strict inequality, G can be obtained from G' by decreasing some elements of the latter. From the properties of essentially positive matrices we can then conclude that, under the conditions (38) the real parts of all eigenvalues of G are less than zero and thus G will have a negative inverse.[44] With that, a sufficient condition to get $\underline{\Psi}_1 > 0$ is that $\underline{S} > 0$, as can be gathered from Eqs. (36) and (10), by using Eq. (15).

For Method 1, Eq. (38) reduces to $h\omega_0\theta_i \leq 1$ (i=0,1, 2,...,I) with at least one strict inequality. It can be shown that the conditions (38) are always obeyed for both methods, if asymptotic thetas corresponding to ω_0 (as given by Eqs. (20) and (21)) are used. This is easily seen to be the case, for Method 1. For Method 2, again the proofs are straightforward but somewhat more involved.

Chapter 4

NUMERICAL RESULTS

4.1 Results for Point Kinetics

We now present some numerical results for two representative reactors, a thermal and a fast reactor. The values for Λ, β_i and λ_i (in $\sec^{-1}$) for these reactors are shown in Table I, for six delayed neutron groups. We shall consider here five different transients, all starting from equilibrium conditions and with $n(0) = 1$. In every case the source term f is taken to be zero. The Tables II to VI show the exact n(t) and the relative percent errors of the calculations for several options of the method presented in this work, the errors being defined as

$$\text{Error} = \frac{n_{calc} - n_{exact}}{n_{exact}} \times 100.$$

The results are shown for selected times t during the transient and for several values of the time step size h used in the calculations. The numbers in each square are in exponential notation and correspond, respectively, to the methods described in each of the cases that follow.

Case 1

This case corresponds to a step reactivity insertion of +0.5 dollars in the fast reactor, calculations having been done by three methods: (a) Padé (2,0) with no explicit treatment of roots; (b) Padé (1,1) with ω_6 treated explicitly;

TABLE I

Parameters for Two Typical Reactors

NEUTRON GROUP	THERMAL REACTOR		FAST REACTOR	
	λ_i	β_i	λ_i	β_i
1	0.0127	2.850×10^{-4}	0.0129	1.672×10^{-4}
2	0.0317	1.5975×10^{-3}	0.0311	1.232×10^{-3}
3	0.115	1.410×10^{-3}	0.134	9.504×10^{-4}
4	0.311	3.0525×10^{-3}	0.331	1.443×10^{-3}
5	1.40	9.600×10^{-4}	1.26	4.534×10^{-4}
6	3.87	1.950×10^{-4}	3.21	1.540×10^{-4}
	$\beta_{tot} = 0.00750$ $\Lambda = 5 \times 10^{-4}$ sec		$\beta_{tot} = 0.00440$ $\Lambda = 1 \times 10^{-7}$ sec	

(c) Crank-Nicholson. Results are presented in Table II.

The errors for this case are of the same order of magnitude for the first two methods, (a) and (b), method (b) having an error usually about half of the error of method (a). The errors of the Crank-Nicholson method, (c), are in general unacceptably large, except for large t and small h. Comparison of the methods (b) and (c) shows the large correction effect obtained by treating ω_6 explicitly, a feature to be shared by some of the following cases. We note that the errors for the first two methods are quite small (less than 1%) for h as large as 1 second.

Case 2

In this case, we perturb the thermal reactor by inserting a -0.5 dollars step reactivity. Calculations were done using the same three methods as for Case 1. Table III shows the results.

For this case, we again have that the errors for the methods (a) and (b) are of the same order of magnitude, except for the early part of the transient ($t \sim 0.1$ sec) when the second method yields much smaller errors. The errors for method (a) are nevertheless also quite small. In this example, the Crank-Nicholson method, (c), fares better than for Case 1, but the errors become too large for time steps of 0.5 seconds or larger, in contrast with the first two methods.

TABLE II

Percent Errors and Exact n(t) for Case 1.

h(sec)	Method	t=0.1 sec	t=1 sec	t=10 sec
0.01	a	-9.64(-5)	-7.53(-5)	-7.85(-5)
	b	+4.82(-5)	+3.77(-5)	0.0
	c	-4.02(+1)	-6.11(+0)	0.0
0.10	a	-7.32(-3)	-7.34(-3)	-6.28(-3)
	b	+4.53(-3)	+4.18(-3)	+3.14(-3)
	c	+4.81(+1)	-3.70(+1)	-6.54(+0)
0.25	a	--	-3.93(-2)	-3.99(-2)
	b		+2.62(-2)	+1.94(-2)
	c		-3.75(+1)	-7.60(+0)
0.50	a	--	-1.29(-1)	-1.64(-1)
	b		+1.08(-1)	+7.77(-2)
	c		-3.75(+1)	-7.71(+0)
1.00	a	--	-3.91(-1)	-6.93(-1)
	b		+6.88(-1)	+3.12(-1)
	c		+3.83(+1)	-7.52(+0)
Exact n(t)		2.075317	2.655853	12.74654

Method: (a) Padé (2,0);
(b) Padé (1,1) + ω_6 term;
(c) Crank-Nicholson.

TABLE III

Percent Errors and Exact n(t) for Case 2.

h(sec)	Method	t=0.1 sec	t=1 sec	t=10 sec
0.01	a	+7.99(-2)	+6.59(-5)	0.0
	b	-2.86(-5)	-4.94(-5)	0.0
	c	-4.70(-2)	-4.94(-5)	0.0
0.10	a	+3.15(+0)	+6.44(-3)	+7.07(-4)
	b	-3.88(-3)	-3.71(-3)	-3.53(-4)
	c	-7.78(+0)	-3.71(-3)	-3.53(-4)
0.25	a	--	+3.44(-2)	+4.37(-3)
	b		-2.32(-2)	-2.27(-3)
	c		+2.83(+0)	-2.27(-3)
0.50	a	--	+1.19(-1)	+1.69(-2)
	b		-9.05(-2)	-9.09(-3)
	c		+2.62(+1)	+5.92(-2)
1.00	a	--	+5.01(-1)	+6.34(-2)
	b		-6.86(-1)	-3.64(-2)
	c		-4.55(+1)	+1.41(+1)
Exact n(t)		0.6989252	0.6070536	0.3960777

Method: (a) Padé (2,0);
(b) Padé (1,1) + ω_6 term;
(c) Crank-Nicholson.

It is to be noted that the transient is very accurately represented in method (b) because of the explicit treatment of ω_6.

Again the errors with the first two methods are less than 1% even for time steps up to 1 second.

Case 3

This case is that of a step reactivity of +1 dollar introduced in the thermal reactor, four methods being used to do the calculations: (a) Padé (2,0) with explicit treatment of ω_0; (b) Padé (2,0) without explicit treatment of roots; (c) Padé (1,1) with ω_0 treated explicitly; (d) Crank-Nicholson. Results are displayed in Table IV.

As one would expect, methods (a) and (c) have smaller errors than the other two, a situation which again shows the improvement obtained by the explicit treatment of the dominant root, in this case ω_0. The errors are about the same for both methods, but method (c) fares better than method (a) for small h. A parallel behavior takes place for methods (b) and (d). Note that the errors for methods (a) and (c) are of the order of 1% for h as large as 1 second.

Case 4

This case corresponds to a positive ramp insertion of reactivity of +1 dollar/sec in the fast reactor. The calculations were done using two methods, namely: (a) Padé (2,0)

TABLE IV

Percent Errors and Exact n(t) for Case 3.

h(sec)	Method	t=0.1 sec	t=0.5 sec	t=1 sec
0.01	a	-2.70(-3)	-7.72(-4)	-1.24(-4)
	b	-5.76(-3)	-9.17(-3)	-1.50(-2)
	c	+1.39(-3)	+4.83(-4)	+6.21(-5)
	d	+2.90(-3)	+4.54(-3)	+7.40(-3)
0.10	a	-2.16(-1)	-6.85(-2)	-1.03(-2)
	b	-5.64(-1)	-1.02(+0)	-1.71(+0)
	c	+1.43(-1)	+4.32(-2)	+6.21(-3)
	d	+2.93(-1)	+4.57(-1)	+7.48(-1)
0.25	a	--	-3.25(-1)	-5.22(-2)
	b		-7.53(+0)	-1.25(+1)
	c		+2.83(-1)	+3.89(-2)
	d		+2.98(+0)	+4.93(+0)
0.50	a	--	-9.04(-1)	-1.59(-1)
	b		-3.33(+1)	-4.99(+1)
	c		+1.46(+0)	+1.47(-1)
	d		+1.44(+1)	+2.46(+1)
1.00	a	--	--	-4.28(-1)
	b			-9.08(+1)
	c			+1.42(+0)
	d			-1.32(+3)
Exact n(t)		2.515766	10.36253	32.18354

Method: (a) Padé (2,0) + ω_0 term;
(b) Padé (2,0);
(c) Padé (1,1) + ω_0 term;
(d) Crank-Nicholson.

with automatic inclusion of the ω_0 term; (b) Padé (2,0) with no explicit treatment of roots. By automatic inclusion of the ω_0 term we mean that this root is treated explicitly whenever $h\omega_0$ is larger than a certain value, which we chose as 0.05. Otherwise, the ω_0 term is not included. Table V shows the results for this case.

The results are exactly the same for t = 0.5 sec and approximately the same for t = 1 sec, which indicates that the main source of error here is the non-constancy of the reactivity. The errors become very large in the vicinity of prompt-criticality (and beyond) for time steps of the order of 0.01 seconds or larger, especially for the results at t = 1 sec. This behavior, however, represents no significant limitation to the method since, in most practical problems, feedback considerations will limit the acceptable time step size to a small value if results of a reasonable accuracy are desired.

Case 5

This case is that of a more complex reactivity insertion pattern in the thermal reactor: a +1 dollar/sec ramp up to 0.5 seconds followed by a -1 dollar/sec ramp up to 1 second; then a +1 dollar/sec ramp up to 1.5 seconds followed by a constant 0.5 dollars reactivity from then on. Note that $\rho(t)$ is a continuous function. Table VI shows the results obtained for this case by three methods: (a) Padé (2,0) with no explicit treatment of roots; (b) Padé (1,1)

TABLE V

Percent Errors and Exact n(t) for Case 4.

h(sec)	Method	t=0.5 sec	t=1 sec
0.001	a	-9.15(-2)	-5.07(-1)
	b	-9.15(-2)	-5.96(-1)
0.01	a	-9.82(-1)	-2.92(+1)
	b	-9.82(-1)	-3.34(+1)
0.10	a	-9.15(+0)	-9.48(+1)
	b	-9.15(+0)	-9.52(+1)
0.25	a	-2.03(+1)	-9.85(+1)
	b	-2.03(+1)	-9.86(+1)
Exact n(t)		2.136407	1207.813

Method: (a) Padé (2,0) + ω_0 term [$h\omega_0 > 0.05$];

(b) Padé (2,0).

TABLE VI

Percent Errors and Exact n(t) for Case 5.

h(sec)	Method	t=0.5 sec	t=1 sec	t=1.5 sec	t=2 sec	t=10 sec
0.01	a	-2.69(-2)	+3.74(-2)	-2.76(-2)	-2.97(-3)	-1.25(-3)
	b	-4.24(-3)	-3.30(-3)	-4.76(-3)	-4.36(-4)	-5.81(-4)
	c	-4.24(-3)	-3.30(-3)	-4.76(-3)	-4.36(-4)	-5.81(-4)
0.10	a	-1.99(+0)	+2.25(+0)	-2.03(+0)	-2.31(-1)	-9.80(-2)
	b	-4.22(-1)	-3.50(-1)	-4.70(-1)	-4.28(-2)	-5.10(-2)
	c	-4.22(-1)	-3.50(-1)	-4.70(-1)	-4.28(-2)	-5.10(-2)
0.25	a	-8.43(+0)	+8.40(+0)	-8.55(+0)	-1.15(+0)	-4.67(-1)
	b	-6.64(+0)	+6.07(+0)	-6.94(+0)	-6.42(-1)	-3.85(-1)
	c	-2.39(+0)	-4.61(+0)	-2.60(+0)	-3.37(-1)	-3.19(-1)
0.50	a	-2.03(+1)	+1.98(+1)	-2.02(+1)	-3.66(+0)	-1.41(+0)
	b	-1.94(+1)	+2.00(+1)	-2.01(+1)	-1.77(+0)	-1.21(+0)
	c	-1.08(+1)	+1.40(+1)	-1.82(+1)	+6.25(+0)	-1.09(+0)
Exact n(t)		1.721422	1.211127	1.892226	2.521601	12.04711

Method: (a) Padé (2,0);

(b) Padé (1,1) + ω_6 term [$h\omega_6 < \min(-2,-1.2h\lambda_6)$];

(c) Crank-Nicholson.

with automatic inclusion of the ω_6 term; (c) Crank-Nicholson. In the second method, (b), the ω_6 term is included whenever $h\omega_6 < \min(-2,-1.2h\lambda_6)$.

The errors are, in general, the same for all three methods. For small h the results show a larger error for method (a) which should be expected from the theory and was noted in previous cases. From the methods (b) and (c) it can be concluded that the better results obtained for the Crank-Nicholson method are accidental, since method (b) is an improvement over Crank-Nicholson.

Notice also that the inaccurate results obtained for larger time steps stem from the very poor representation of the ramp as a series of steps. In particular, for h = 0.5 sec, we simply have the transient represented as a 0.25 dollar step from 0 to 1.5 seconds followed by another step totalling 0.5 dollars from then on.

4.2 Analysis of Point Kinetics Results

The results of Sec. (4.1) for the first three cases show the great improvement of the Padé (1,1) method with explicit treatment of ω_0 or ω_6 over the simple Crank-Nicholson method. This improvement is apparent not only in that greater accuracy is obtained for all time steps, but also in that larger time steps can be taken without incurring large errors.

The Padé (2,0) method is also much better than the

Crank-Nicholson method, for the cases of constant reactivity, as can be concluded from the results for the first three cases. Except for near or above prompt-criticality, the Padé (2,0) method does quite well without the need for inclusion of the ω_0 term and does not suffer from the oscillations present in the Crank-Nicholson method when h is large.

In general, the Padé (1,1) method yields better results than the Padé (2,0) method. A glance through Tables II to IV shows that halving the time step size h for the first of these methods will produce better results than the second method with time step h. Note, however, that in many cases the Padé (1,1) method will require one more root treated explicitly than will the Padé (2,0) method. Also, such extra calculation will normally more than double the amount of computation per time step. It can thus be seen that the Padé (2,0) method will be more economical, in general, even if the time step necessary to attain a given accuracy is smaller.

The results for the last two cases show that, for varying reactivity, inclusion of roots in the basic approximations does not make any significant improvement in accuracy.

Calculations for other cases, in addition to those presented, have been done and confirmed the trends observed here which seem to hold in general and which agree with the theoretical expectations.

4.3 Results for Space-Dependent Kinetics

Next, results are presented for two slab reactors without external source, one a thermal reactor,[45] and the other a fast reactor.[37,46] The two-group parameters for these reactors are presented in Tables VII and VIII, and their geometries are shown in Figs. 1 and 2. The reactors are made critical initially by dividing $\nu\Sigma_f$ by k_{eff}. Although these reactor configurations are by no means typical of real reactors, they are expected to display similar numerical properties. We shall consider several different transients all starting from equilibrium conditions. Except for the first test case, all transients are highly asymmetric.

For all cases in which point kinetics calculations were used to find the thetas, the method described in Appendix B was applied, with inclusion of ω_0 and ω_I terms in Σ' for $h\omega_0>0.05$ and $h\omega_I<\min(-2,-1.2h\lambda_I)$, respectively. When point kinetics was not used in Method 1, θ_i was always taken to be 0.5 (i=1 to I) and θ_0 chosen as described in each case. For Method 2, when point kinetics calculations were not used, $\theta(s)$ was taken to be either s/h or 1. For simplicity, these choices of $\theta(s)$ will be indicated by $\theta_0=0.5$ and $\theta_0=1$, respectively. For ramps, θ_i was chosen as 0.5 (i=0 to I) in Method 1, and $\theta(s)$ as s/h in Method 2, for all cases. The reasons for these choices will be given later. The time step was doubled at the end of any ramp that was followed by a period during which reactor parameters remained constant.

The motivation for choosing $\theta_i=0.5$ (i=0 to I) or

TABLE VII

Parameters for a Thermal Reactor

(all units in the CGS system).

Parameter	Regions 1 & 3	Region 2
D_1	1.50	1.00
D_2	0.50	0.50
Σ_{r1}	0.026	0.020
Σ_{a2}	0.180	0.080
Σ_{s21}	0.015	0.010
$\nu\Sigma_{f1}$	0.010	0.005
$\nu\Sigma_{f2}$	0.200	0.099
$\chi_1 = 1.0$	$v_1 = 1.0\times10^7$	$\Sigma_{r1} = \Sigma_{T1} - \Sigma_{s11}$
$\chi_2 = 0.0$	$v_2 = 3.0\times10^5$	$\Sigma_{a2} = \Sigma_{T2} - \Sigma_{s22}$
Group i	β_i	λ_i
1	0.00025	0.0124
2	0.00164	0.0305
3	0.00147	0.111
4	0.00296	0.301
5	0.00086	1.14
6	0.00032	3.01

Regions 1 & 3: width = 40 cm, mesh size = 2.5 cm
Region 2 : width = 160 cm, mesh size = 2.5 cm
$k_{eff} = 0.901527855$

TABLE VIII

Parameters for a Fast Reactor

(all units in the CGS system).

Parameter	Core	Blanket
D_1	2.0770	1.5480
D_2	0.9322	0.9484
Σ_{r1}	0.020010	0.016137
Σ_{a2}	0.015030	0.008703
Σ_{s21}	0.014730	0.013070
$\nu\Sigma_{f1}$	0.008916	0.000216
$\nu\Sigma_{f2}$	0.014230	0.000446
$\chi_1 = 1.0$ $\chi_2 = 0.0$	$v_1 = 1.12\times10^{10}$ $v_2 = 1.18\times10^{9}$	$\Sigma_{r1} = \Sigma_{T1}-\Sigma_{s11}$ $\Sigma_{a2} = \Sigma_{T2}-\Sigma_{s22}$
I=1	$\beta = 0.003$	$\lambda = 0.065$
Core:	width = 70 cm,	mesh size = 5 cm
Blanket:	width = 48 cm,	mesh size = 8 cm
$k_{eff} = 0.999996149$		

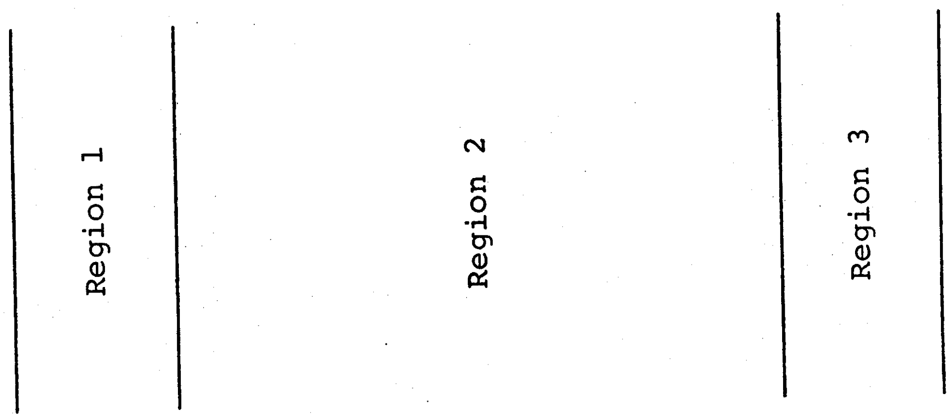

Fig. 1. Geometry for a Thermal Reactor.

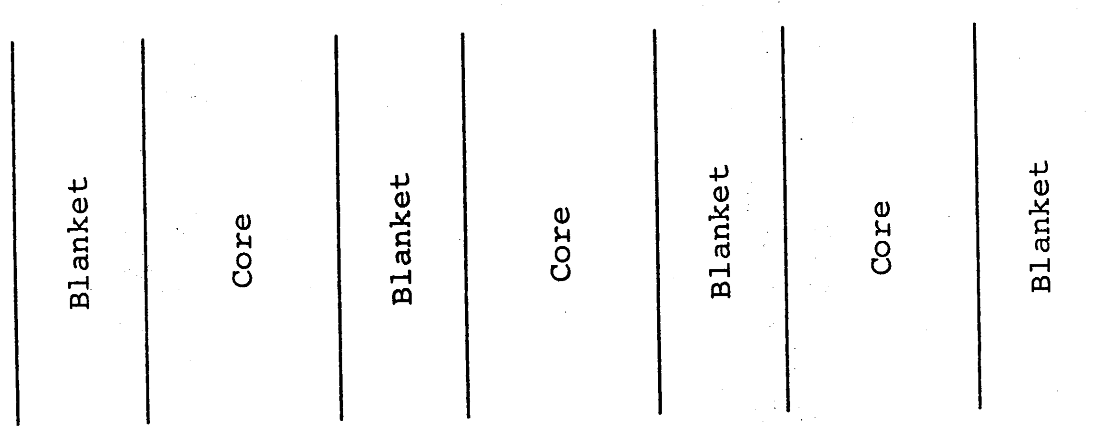

Fig. 2. Geometry for a Fast Reactor.

$\theta(s)=s/h$ lies in the fact that, as seen in Sec. (3.6), they yield a $0(h^2)$ global accuracy. Other choices will be accurate only to $0(h)$, globally. In practice, these choices of thetas usually yields the best results, if fixed thetas are to be used, except after steps or steep ramps in prompt-subcritical reactors. In such cases, use of $\theta_0=1$ or $\theta(s)=1$ will rapidly attenuate modes with large negative time constants which would otherwise make the solution oscillatory.

Case A

In this first case $\nu\Sigma_f$ was increased everywhere by 0.4% in a step fashion, in the thermal reactor. About 0.5 dollars of reactivity was thereby added. Several methods were used to perform the calculations. Some of the results are presented in Table IX, which shows for several time step sizes h, the fast flux in the middle of Region 1 or 3, at t=0.5 sec and t=5 sec.

To obtain column A, Method 1 was used with thetas calculated from point kinetics. Method 2 was also applied under the same conditions, and led to essentially the same results. Column B shows the same calculation as in column A but with θ_0 taken as 1 for the first step. In column C we used Method 2 with $\theta_0=1$ for the first two steps and $\theta_0=0.5$ from then on. Column D shows the results using Method 1 with $\theta_0=1$, for the first two steps, and $\theta_0=0.5$ thereafter.

TABLE IX

Flux Values for Case A.

h (sec)	(A) Method 1 (θ=p.k.)†		(B) Method 1* (θ=p.k.)†		(C) Method 2** (θ_0=0.5)		(D) Method 1** (θ_0=0.5)		(E) Method 1 (θ_0 = 1)	
	.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec
1.00	-	7.5669	-	7.5699	-	7.7400	-	7.9602	-	6.9793
0.50	3.1673	7.5658	2.9281	7.5667	2.9445	7.5142	2.9281	7.4253	2.9281	7.2591
0.25	3.1618	7.5664	3.1620	7.5668	3.0562	7.5578	3.0532	7.5429	3.0532	7.4090
0.10	3.1622	7.5654	3.1612	7.5660	3.1848	7.5653	3.1873	7.5655	3.1186	7.5023
0.05	3.1589	7.5652	3.1608	7.5659	3.1596	7.5658	3.1595	7.5658	3.1398	7.5339
EXACT FLUX	$\phi(0.0)$ = 1.4202, $\phi(0.5)$ = 3.1609, $\phi(5.0)$ = 7.5659									

* θ_0 = 1 for first step

** θ_0 = 1 for first two steps

† p.k. = point kinetics thetas

This last calculation was repeated taking θ_0=1 all the way, results being presented in column E.

Not shown in the table are calculations done with the Crank-Nicholson method (θ_0=0.5) and with Method 2 taking θ_0=0.5, as well as calculations analogous to those of columns C and D but with θ_0=1 for only the first step. The Crank-Nicholson results and their counterparts for Method 2 were oscillatory, as expected from theory, yielding large errors. The last two calculations tended to be slightly less accurate than their counterparts, in C and D.

Examining Table IX we see that column A displays the most accurate results, except when h is quite small. This behavior is probably due to some very small oscillations in the calculated thetas. The solution itself oscillates slightly, but oscillations are small enough that the first difference does not change sign. The theta oscillations are greatly reduced by the procedure described above leading to column B. Taking θ_0=1 for the first time step makes the errors decrease for small h but increase for large h as can be seen by comparison of columns A and B.

In spite of these anomalies, the results in column A are very good. This is no surprise, of course, since in this case the assumption of constant flux shape is valid. Comparison of columns C and D shows that Method 1 is slightly better than Method 2 for this choice of thetas. The procedure used in TWIGL of choosing θ_0=1 all the way is clearly the worse, except for the first two time steps when columns D and E coincide.

Case B

A transient was induced in the thermal reactor by a step reduction of Σ_{a2} in Region 1 by 1%. This is a transient in the delayed supercritical range. Results using various methods are presented in Table X which shows, for several values of h, the fast flux in the middle of Regions 1 and 3 for t=0.5 sec and t=5 sec.

Columns A and B correspond to calculations with Method 2 and Method 1, respectively, with $\theta_0=1$ for the first time step, and thetas calculated from point kinetics for subsequent time steps. Columns C and D display similar results by Method 2 and Method 1 but with $\theta_0=0.5$ after the first time step. In column E we took $\theta_0=1$ for all time steps and used Method 1.

Calculations corresponding to columns A and B were also done without taking $\theta_0=1$ for the first step. While the calculated value for the first step was much better, all subsequent steps had inferior results. Also oscillations of the theta values took place; these were not present in the calculations for columns A and B. Such behavior is understandable, in view of the large change in shape that takes place in the first time step due to the fast modes (prompt jump). The point kinetics estimation of thetas for this time step is thus of little value.

TABLE X

Flux Values for Case B.

h (sec)	Region	(A) Method 2* (θ=p.k.)†		(B) Method 1* (θ=p.k.)†		(C) Method 2* (θ_0=0.5)		(D) Method 1* (θ_0=0.5)		(E) Method 1 (θ_0=1.0)	
		.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec	.5 sec	5 sec
1.00	1	-	9.5801	-	9.5858	-	9.3946	-	9.0746	-	8.7727
	3	-	2.3549	-	2.3568	-	2.3364	-	2.2981	-	2.2454
0.50	1	3.3579	9.5795	3.3366	9.5836	3.3579	9.6354	3.3366	9.7942	3.3366	9.1555
	3	1.5607	2.3527	1.5583	2.3529	1.5607	2.3562	1.5583	2.3747	1.5583	2.2963
0.25	1	3.6310	9.5800	3.6333	9.5809	3.7867	9.5891	3.8036	9.6177	3.4972	9.3622
	3	1.5903	2.3532	1.5906	2.3533	1.6071	2.3527	1.6092	2.3551	1.5757	2.3240
0.10	1	3.6352	9.5793	3.6355	9.5795	3.5845	9.5803	3.5783	9.5809	3.5815	9.4914
	3	1.5914	2.3532	1.5914	2.3532	1.5874	2.3533	1.5867	2.3534	1.5852	2.3414
0.05	1	3.6355	9.5794	3.6356	9.5794	3.6399	9.5800	3.6404	9.5802	3.6090	9.5354
	3	1.5912	2.3532	1.5912	2.3532	1.5907	2.3533	1.5906	2.3533	1.5883	2.3473
EXACT FLUX	1	$\phi(0.0)$ = 1.4202,				$\phi(0.5)$ = 3.6363,			$\phi(5.0)$ = 9.5799		
	3	= 1.4202,				= 1.5914,			= 2.3533		

* θ_0 = 1 for first step † p.k. = point kinetics thetas

Calculations using the Crank-Nicholson method and its equivalent for Method 2 were also performed. Solutions had large errors and were oscillatory in nature.

Finally, columns D and E were recalculated using $\theta_0=1$ for the first two steps. The accuracy of the results was similar to that shown for columns D and E in Table X.

Columns A and B of Table X are the most accurate results for the larger time step sizes, with column A only slightly better, overall. Thus the two methods have about the same accuracy, for thetas calculated from point kinetics. Column C, however, displays generally better results than column D, so that, at least for this case, when $\theta_0=0.5$ is used, Method 2 fares better. Comparing columns D and E, one again sees the great improvement obtained by using $\theta_0=1$ only at the first one or two steps.

Case C

This case involves a step reduction of Σ_{a2} in Region 1 of the thermal reactor by 5%. Such a perturbation puts the reactor in a prompt-critical state. Table XI shows the fast flux at t=0.001 and 0.005 sec in the middle of Regions 1 and 3, calculated for several values of the time step h.

The results were obtained by two methods, the Crank-Nicholson method, and Method 1 with thetas calculated from point kinetics. The corresponding results obtained from Method 2, with $\theta_0=0.5$ and point kinetics thetas, differ from the Method 1 results by at most one unit in the last digit.

TABLE XI

Flux Values for Case C.

h (sec)	Region	CRANK-NICHOLSON 0.001 sec	CRANK-NICHOLSON 0.005 sec	THETA METHOD 0.001 sec	THETA METHOD 0.005 sec
1×10^{-3}	1	4.5981	130.06	4.5611	112.79
	3	1.4242	2.4403	1.4239	2.2887
5×10^{-4}	1	4.3711	111.49	4.3552	107.68
	3	1.4220	2.2782	1.4219	2.2445
2.5×10^{-4}	1	4.3448	107.61	4.3417	106.68
	3	1.4214	2.2444	1.4214	2.2362
1×10^{-4}	1	4.3274	106.57	4.3274	106.44
	3	1.4212	2.2354	1.4212	2.2342
5×10^{-5}	1	4.3236	106.42	4.3236	106.42
	3	1.4212	2.2341	1.4212	2.2341
EXACT	1	$\phi(0)=1.4202$, $\phi(0.001)=4.3231$, $\phi=(0.005)=106.37$			
FLUX	3	$=1.4202$, $=1.4212$, $= 2.2337$			

This behavior was expected because of the small h required for the calculation of a prompt-supercritical transient.

Here we have a case in which the fundamental mode is strongly dominant, and the point kinetics thetas produce sizable improvements in accuracy. This improvement is particularly evident for the values at t=0.005 sec. The values for t=0.001 sec with $h=10^{-4}$ sec, and all values for $h=5\times10^{-5}$ sec coincide for both methods because, as stated previously, the ω_0 terms were included in the point kinetics calculation only for $h\omega_0>0.05$. Continued inclusion of such terms would have kept the superiority of the theta method over the Crank-Nicholson method. It can be seen from Table XI that, for roughly comparable accuracy, it is necessary to use for the Crank-Nicholson method time steps about half as large as those used with the theta method.

Case D

In this case a transient is provoked by reducing the absorption cross sections linearly in the right core of the fast reactor by 0.5% during the time interval from t=0 to t=0.5 sec, and holding these values constant thereafter. The result is a delayed supercritical transient. Table XII shows, for several values of h, the fast flux at t=0.5 sec for the middle points of the outer cores, for Methods 1 and 2 using $\theta_0=0.5$.

For Method 1 this amounts to a modified Crank-Nicholson scheme given by

TABLE XII

Flux Values for Case D.

h (sec)	Core	Method 1	Method 2
0.50	Left	1.7667	1.7481
	Right	7.2318	7.0902
0.25	Left	1.7465	1.7400
	Right	7.0773	7.0279
0.10	Left	1.7383	1.7366
	Right	7.0149	7.0019
0.05	Left	1.7368	1.7364
	Right	7.0038	7.0002
0.025	Left	1.7364	1.7363
	Right	7.0006	6.9997
EXACT	Left	$\phi(0.0)=1.1443,$	$\phi(0.5)=1.7363$
FLUX	Right	$=1.1443,$	$=6.9995$

$$\underline{\Psi}_1 = [I-\frac{1}{2}hH_1]^{-1} \ [I+\frac{1}{2}hH_0]\underline{\Psi}_0.$$

It is seen from Table XII that the results using Method 2 are much better, in that the errors are two to three times smaller for the same h, or in that one must roughly halve the time step in Method 1 to match the accuracy of Method 2.

Methods 1 and 2 with θ_0=0.5 are of comparable accuracy in the constant parameter part of the transient, after t=0.5 sec. For this part we also used point kinetics thetas, in which case Method 2 fared slightly better than Method 1. Comparing values obtained for this case using point kinetics thetas and θ_0=0.5, one concludes that, for both methods, the θ_0=0.5 choice is generally better for smaller h, the converse being true for larger h.

Performing the same calculation as in Ref. (37) with Method 1 shows that use of θ_0=0.5 yields better results than the ones shown in that reference, using the thetas described there.

A similar computation was performed for the thermal reactor in which we linearly reduced Σ_{a2} in Region 1 by 1% in 1 sec, and held the parameters constant from then on. The results at the end of the ramp section, at t=1 sec, yielded the same conclusion about the higher accuracy of Method 2 over Method 1, with θ_0=0.5. For the constant parameter part of the transient, after t=1 sec, we found that again Methods 1 and 2 produced results of comparable accuracy. When point kinetics thetas were used, Method 2 was found again slightly better than Method 1, which suffered from oscillation in its thetas, for the larger h. At the early part of the constant section ($t\sim2$ sec), the use of point kinetics thetas improved results. However, later in time ($t\sim5$ sec), values obtained with $\theta_0=0.5$ all the way were more accurate.

Case E

In this case we introduce a subcritical transient in the thermal reactor by increasing Σ_{a2} linearly in Region 1 by 3% from t=0 to t=1 sec, and keeping parameters constant thereafter. In Table XIII we show, for several h, the values of the fast flux in the middle point of Regions 1 and 3 at t=1 sec, using Methods 1 and 2 with $\theta_0=0.5$.

It can be seen from Table XIII that with Method 1, the modified Crank-Nicholson method, results are better than with Method 2, except for the smaller h when they both tend to agree, as expected. Results for the constant reactivity part of the transient, after t=1 sec, are not shown here. They indicate however, that the Methods 1 and 2 give overall results of comparable accuracy, either using $\theta_0=0.5$ or thetas from point kinetics.

Use of point kinetics thetas again improved the results early in the constant parameter portion of the transient but, as time increases and h decreases, they tend to have a lower accuracy in comparison with the $\theta_0=0.5$ results. The thetas

TABLE XIII

Flux Values for Case E.

h (sec)	Region	Method 1	Method 2
0.50	1	0.49608	0.49077
	3	1.3451	1.3438
0.25	1	0.49339	0.49208
	3	1.3443	1.3440
0.10	1	0.49335	0.49328
	3	1.3444	1.3444
0.05	1	0.49418	0.49418
	3	1.3445	1.3445
0.025	1	0.49433	0.49432
	3	1.3445	1.3445
EXACT	1	$\phi(0.0)=1.4202$,	$\phi(0.5)=0.49432$
FLUX	3	$=1.4202$,	$=1.3445$

tend to oscillate after a while, the oscillations becoming larger as h decreases, even yielding thetas outside the range (0,1). The computer program used in the calculation limits them to $0.1<\theta\leq 1$. Those thetas oscillations probably account for the better accuracy of the calculations with a fixed θ_0 of 0.5 for small h. Use of $\theta_0=1$ for the first step after t=1 sec would probably eliminate the theta oscillations and yield better results using point kinetics thetas for the subsequent time steps, as was the case in the step problems below prompt-critical analyzed before.

An analogous calculation was done for the fast reactor, by linearly increasing the absorption cross sections of the right core by 0.5% in 0.5 sec, and then holding it constant. The value found at t=0.5 sec again shows Method 1 to be more accurate than Method 2, with $\theta_0=0.5$. Calculations using both methods and either $\theta_0=0.5$ or point kinetics thetas were all of similar accuracy for the flat portion of the transient.

It should be noted here that the conclusion about the accuracy of Methods 1 and 2, for this subcritical ramp, is the reverse of the one reached in the previous case of the delayed supercritical ramp, for both the thermal and the fast reactor.

4.4 Analysis of Space Kinetics Results

From the results of Sec. (4.2) it is hard to see any general trend which would permit us to draw conclusions about which method is better. We saw in Case A that, for the supposedly ideal case of constant shape, some form of weak instability is reflected in the oscillations found in the computed theta values. While this did not prove to be a serious problem it is certainly an unpleasant fact, and we found it useful to deal with it by the special procedure of taking $\theta_0=1$ for the first one or two steps.

When there is non-uniform reactivity insertion below prompt-critical, either in step problems or in the constant reactivity part after initial ramps, the use of point kinetics thetas did improve the results for the larger values of h as can be seen from the results for Cases B, D, E and F. The additional computing time to find these parameters can, however, be appreciable, and might be better put to use by taking a smaller time step especially since, in some cases the results based on computed thetas are less accurate then those using $\theta_0=0.5$. The above argument is even stronger for ramp insertions and for this reason (as discussed in Sec. (4.3)) it was decided to abandon the point kinetics thetas for ramps in all cases calculated.

With respect to the relative merits of Methods 1 and 2 we found that, in delayed supercritical problems (Case D), Method 2 produced better results for the ramp section of the transient, for both the fast and the thermal reactor. That would be a result of general interest if it were not for the fact that the conclusion is reversed for the subcritical problems considered in Case E. It may be, however, that the contrary behavior observed holds in general for the two separate classes of problems. We did not investigate this possibility in sufficient depth to justify such a general conclusion, however.

In prompt-supercritical problems, because of the necessarily small time steps, Methods 1 and 2 essentially agree. It was found, in Case C, that point kinetics thetas did produce a significant improvement in accuracy. This is a case in which a single mode is strongly dominant. In these cases one might try to find the asymptotic period by some simpler method and thus avoid the penalty imposed by the point kinetics related calculations.

4.5 Additional Results for Space-Dependent Kinetics

We quote here other interesting results of our study in which we compare the chosen methods with other variations. We shall use the fast reactor with the same ramp as that described in Case D.

First we compare Method 2 using $\theta(s)=s/h$ (that is assuming $L\underline{\phi}$ and $F\underline{\phi}$ to vary linearly between their initial and final values) with another calculation using Eq. (3.12) in which L(t) and F(t) are given their real values and only $\underline{\phi}(t)$ is assumed to vary linearly. The latter is equivalent to a linear spline approximation for the flux.

The errors for the fast flux in the middle points of the three cores (left, middle, and right, respectively) for the two methods are shown in Table XIV, for t=0.5 sec. They indicate that the linear spline method is far less accurate.

In trying to understand the reason for this, we consider a linear L(t) and F(t) and take a linear flux approximation. This gives, for instance,

$$\frac{1}{h}\int_0^h M(t+s)\underline{\phi}(t+s)\,ds = \frac{1}{2}\,[(M_0+\tfrac{1}{3}\Delta M)\underline{\phi}_0+(M_0+\tfrac{2}{3}\Delta M)\underline{\phi}_1]$$

where $\Delta M=M_1-M_0$. Thus, the operator at one third of the interval operates on the initial flux, and the operator at two thirds of the interval operates on its final value. In the method we adopted one would have

$$\frac{1}{h}\int_0^h M(t+s)\underline{\phi}(t+s)\,ds = \frac{1}{2}[M_0\underline{\phi}_0+M_1\underline{\phi}_1]$$

so that the operator at a given time t acts on the flux for the same time t. This fact, we conjecture, accounts for the greater accuracy of Method 2 as compared to a linear spline approximation, at least for cases below prompt-critical.

Another variation studied was to assume the varying properties to be replaced by their average values during the time step. It was also found to be much less accurate than Methods 1 or 2 with θ_0=0.5. For the case of a ramp, this amounts to substituting for the ramp a series of steps with the result that the global error, for time steps large in comparison with the prompt mode time constant, becomes 0(h) instead of $0(h^2)$ as shown in Sec. (2.4). The flux tends to go through a series of prompt jumps at every time step, in-

TABLE XIV

Percent Errors for a Delayed Critical Ramp.

h (sec)	Core	Linear $L\phi$, $F\phi$	Linear ϕ
0.10	Left	0.0181	4.31
	Middle	0.0260	6.93
	Right	0.0343	10.63
0.05	Left	0.00507	1.42
	Middle	0.00730	2.28
	Right	0.00960	3.50
0.02	Left	<0.001	0.267
	Middle	<0.001	0.430
	Right	0.00124	0.660

TABLE XV

Ratios of Relative Error and Time Step Size.

h (sec)	Left Core	Middle Core	Right Core
0.020	1.82	2.92	4.47
0.010	1.89	3.04	4.67
0.005	1.92	3.09	4.73
0.0025	1.92	3.08	4.72

stead of behaving smoothly as it should, and this in turn introduces an 0(h) error as a qualitative graphical analysis would show. Thus, one concludes that the analysis of Sec. (2.4) is relevant only for h extremely small.

Table XV shows the ratio of the relative error and the time step size for the fast flux at the middle points of the three cores of the fast reactor, at t=0.5 sec, for several values of h. The conclusion is immediate that the errors are indeed 0(h), for the time step sizes under consideration. In these calculations we used Method 1 with point kinetics thetas.

From the above results we infer that, for the point kinetics problem, the modified Crank-Nicholson method is probably also more accurate than the method of Chap. 2, for the case of a ramp in the range below prompt-critical except, perhaps, for slow ramps. This is because the latter method uses constant average parameters during a time step.

Calculations were also done for a limited ramp in the thermal reactor, reaching into the prompt-supercritical range. Because of the small time steps needed to achieve a significant accuracy, Methods 1 and 2 essentially coincide. The ramp section was thus effectively treated by the modified Crank-Nicholson method, so there is nothing new here to compare results with. In the constant parameter section, improvement in accuracy by use of point kinetics thetas is comparable with the results shown in Case 3, as would be expected.

We repeated some of the calculations mentioned before, using a constant weight function, instead of the adjoint flux for the initial state, to obtain the point kinetics parameters to find the thetas. The results for the cases explored did not differ greatly from the corresponding results using adjoint weighting. We found that while overall accuracy is somewhat better for adjoint weighting in some instances, this was not a systematic trend. In addition, we observed that the oscillations in the theta values tend to diminish with a constant weight function. Computation time savings by using a constant weight function might be appreciable, since it would eliminate all of the multiplications required in the scalar products needed to find the point kinetics parameters.

The use of point kinetics to calculate thetas for the ramp section in the cases analyzed did not produce good results. In general we found $\theta_0 \simeq 0.5$, and deviations from this value were due more to the numerical procedure adopted than to a genuine deviation associated with the true point kinetics solution. Methods to improve accuracy of the numerical point kinetics calculation are straightforward but the computational effort spent in such a task is hardly justified. This prompted us to abandon point kinetics as a means of finding thetas, and to use $\theta_0 = 0.5$ under all ramp conditions.

Good spectral coupling was observed for all transients. That is, the ratios of fast to slow fluxes for all points in both the thermal and the fast reactors deviated at most a few percent from steady-state values. This amount is quite small when compared with the large asymmetry factors encountered in the transients analyzed. This fact confirms our contention, made in Sec. (3.4), that the energy coupling is much stronger than spatial coupling and justifies our use of only one theta for all neutron energy groups, especially since only one theta is used for all spatial points.

Finally, we made special modifications in the computer code written to perform the calculations, in order to allow a comparison between the computation time using point kinetics thetas with that using predetermined thetas. It was found that, with adjoint weighting, use of point kinetics increased the computation time during a time step by about 50% for the fast reactor (I=1) and by about 75% for the thermal reactor (I=6). These numbers reflect both the time spent to do the point kinetics calculations (a function of I) and the time spent to obtain the point kinetics parameters, which depends not only on I but also on G and K (see Chap. 3 for notation). This extra time would be somewhat reduced by use of a constant weight function.

For higher dimensions and more neutron energy groups, the fraction of time spent in point kinetics related computations may decrease, since the time spent for the matrix inversions required by implicit methods, will probably increase at a faster rate than the time spent for computing point kinetics thetas.

Chapter 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions for Point Kinetics

As the analysis of the point kinetics results in Sec. (4.2) indicates, a great improvement is achieved in the simple Crank-Nicholson method by the addition of terms to treat the eigenvalues ω_0 and/or ω_I explicitly, at least in cases of constant reactivity. This procedure allows much larger time steps to be taken while maintaining errors acceptably low, and yields a far superior accuracy for a given time step size.

The Padé (2,0) method is also definitely superior to the Crank-Nicholson method, for constant reactivity, and does fairly well without an explicit treatment of any eigenvalue in many practical cases. The ω_I term is not essentially needed for the Padé (2,0) approximation and the ω_0 term need be included only for cases near or above prompt-criticality. The simple Padé (2,0) approximation does not exhibit the oscillatory behavior sometimes observed in the Crank-Nicholson method.

In most instances, the method based on the Padé (1,1) approximation produces better results than when the Padé (2,0) expression is used in the present method. However, halving the time step size h for the latter method will make it more accurate than the first method with time step h.

Since the Padé (1,1) method will, in many cases, require one more eigenvalue (ω_I) treated explicitly, and since this addition will normally more than double the computation time per step, we conclude that, usually, the method based on the Padé (2,0) approximation will be faster, for the same accuracy.

For cases in which the reactivity changes with time, addition of eigenvalue correction terms to the basic Padé approximations did not produce significant improvements in accuracy.

From the above, we can conclude that, for cases of practical interest, the Padé (2,0) method with a criterion to decide whether the ω_0 term should be treated explicitly, plus some strategy for the choice of the time step size will, in general, provide a very fast and accurate computational method for point kinetics calculations in general, when the inversion scheme developed in Sec. (2.3) is used.

For cases in which a very precise treatment of the beginning of transients is desired, the Padé (1,1) method with inclusion of the ω_6 term is to be preferred, when ω_6 has a large magnitude. Although one could also include the ω_6 term in the Padé (2,0) method, the error for the previous method will be smaller for most cases, as can also be concluded from the underlying theory.

The final conclusion is that the general method presented in Chap. 2, including the inversion scheme developed, is particularly good for cases in which the reactivity can be represented by a series of steps and performs quite well for more general cases.

5.2 Conclusions for Space-Dependent Kinetics

As stated in Sec. (4.3) it is difficult to draw general conclusions about which of the methods used in the calculations is the best, because of the lack of a general trend. On the other hand, some specific conclusions have been reached as summarized below.

The theta method based on point kinetics, for either Method 1 or Method 2, provides only a marginal improvement, in most cases, over these same methods with properly chosen fixed thetas. In some problems, the calculated thetas exhibit an oscillatory behavior that degrades the accuracy. Such behavior was even observed in a case of constant flux shape.

The point kinetics thetas did improve the accuracy (with respect to suitably chosen fixed thetas) in certain problems, when reactor parameters were constant for a period of time, but that improvement was, in many cases, restricted to the beginning of such period. Also, as the time step size decreased, point kinetics thetas tended to make results less accurate as compared to those obtained from properly chosen fixed thetas.

Methods 1 and 2 produce results of comparable accuracy. In subcritical problems, Method 1 was always better, the converse being true for delayed supercritical transients. The two Methods (1 and 2) essentially agree for prompt-supercritical transients, because of the necessarily small time steps one has to take in such cases.

A case in which there was a significant improvement in accuracy with the point kinetics thetas was that of a prompt-supercritical step. This is a case in which one mode is strongly dominant. We feel, however, that one might try to find the asymptotic period in such cases by some simpler method and then use it to determine asymptotic thetas, to get a similar improvement without the need for the point kinetics calculations.

One definite conclusion reached was that use of $\theta_0=1$ for only the first one or two time steps of the calculation yields much better results than using $\theta_0=1$ for all time steps. This conclusion applies to cases in which using $\theta_0=0.5$ all the way would produce large errors and oscillations, as in cases below prompt-criticality where a constant reactivity section of a transient is preceded by a jump (step) or steep ramp. The decision of how steep the ramp is will depend on the problem at hand, as will the choice of how many steps to take with $\theta_0=1$.

Another interesting conclusion, valid at least for the range below prompt-criticality, is that the approximations of linear $L\underline{\phi}$ and $F\underline{\phi}$ produce far better results than those obtained using the real L(t) and F(t) and then approximating

$\underline{\phi}(t)$ by a straight line. This result suggests that higher order spline approximations for the time variable might become more accurate if applied to $L\underline{\phi}$ and $F\underline{\phi}$ rather than to $\underline{\phi}$ alone.

For reactors below prompt-critical at least, use of the constant parameter approximation, in which varying quantities are replaced by their average during the time step, also fared much worse when compared to the linear $L\phi$ and $F\phi$ approximations. The constant parameter approximation turns out to be $O(h)$ in the prompt-subcritical range, instead of $O(h^2)$, and becomes $O(h^2)$ only for very small values of h.

The computation of the thetas using point kinetics took about 50% to 75% longer during the time step calculations, when adjoint weighting was used. Use of a constant instead of the adjoint flux as a weight function was also explored and found to yield results of comparable accuracy, with some savings in computation time.

Therefore, it is our final conclusion that the use of methods based on point kinetics calculations to find better integration parameters θ for the theta method are not worth the computational effort required.

5.3 Recommendations for Further Research

The time step size that both the theta method and our point kinetics method can take, with satisfactory accuracy, is already quite large, in comparison with some other methods. While it can be argued that extending this time step much longer might produce the unwanted effect of having the flux grow so much that the feedback effects will be imprecisely accounted for, it must not be forgotten that, usually, methods that can take larger time steps mantaining a certain accuracy, have smaller errors for a given time step. Having smaller errors is, of course, a desirable feature. In this vein, we make the recommendations that follow.

For point kinetics, one direction in which some additional study could be done is that of finding a better treatment for the case of reactivity varying in a general manner. A more difficult task would be to include feedback directly into the solution method. It appears to us, however, that an extension of the general method of Chap. 2 in these directions does not seem to offer much promise.

The theta method, as considered in this thesis, is based on the idea of finding space-independent θ-parameters that would improve the accuracy of the time integration of the space-dependent equations. It is our recommendation that the theta method as such should not be pursued further, as there appears to be little room for improvement over simpler choices of thetas like 0.5 and 1.

A possible extension of the theta method, that does deserve some exploring, is that of finding space (and energy) dependent thetas that could improve the accuracy, along the lines described in Ref. (47) for the heat conduction equation. There, a θ-matrix is found and incorporated directly in the differencing scheme to produce a difference equation of higher accuracy. If this could be done for the reactor case without too much extra computational effort it would certainly be worthwhile.

REFERENCES

1. A. F. HENRY and A. V. VOTA, "WIGL-2 - A Program for the Solution of the One-Dimensional, Two-Group Space-Time Diffusion Equations Accounting for Temperature, Xenon, and Control Feedback", WAPD-TM-532, Bettis Atomic Power Laboratory (1965).

2. J. B. YASINSKY, M. NATELSON and L. A. HAGEMAN, "TWIGL - A Program to Solve the Two-Dimensional, Two-Group Space-Time Neutron Diffusion Equations with Temperature Feedback", WAPD-TM-743, Bettis Atomic Power Laboratory (1968).

3. A. F. HENRY, Nucl. Sci. Eng., 3, 52 (1958).

4. E. P. GYFTOPOULOS, "General Reactor Dynamics", in The Technology of Nuclear Reactor Safety, Vol. 1, Chap. 3, T. J. THOMPSON and J. G. BECKERLEY, Eds., The MIT Press, Cambridge, Massachusetts (1964).

5. Z. AKCASU, G. S. LELLOUCHE and L. M. SHOTKIN, Mathematical Methods in Nuclear Reactor Dynamics, Academic Press, New York and London (1971).

6. K. O. OTT and D. A. MENELEY, Nucl. Sci. Eng., 36, 402 (1969).

7. S. KAPLAN, "Synthesis Methods in Reactor Analysis", in Advances in Nuclear Science and Technology, Vol. 3, p. 233, P. GREEBLER and E. J. HENLEY, Eds., Academic Press, New York and London (1966).

8. T. A. PORSCHING, SIAM J. Appl. Math., 16, 301 (1968).

9. J. LEWINS, Nucl. Sci. Eng., 7, 122 (1960).

10. J. C. VIGIL, Nucl. Sci. Eng., 29, 392 (1967).

11. T. AUERBACH, J. MENNIG and G. SARLOS, " The Application of Lie Series to Time Dependent Reactor Theory", in Proc. Brookhaven Conf. on Ind. Needs and Acad. Research

in Reactor Kinetics, BNL-50117, p. 306, Brookhaven National Laboratory (1968).

12. G. R. KEEPIN and C. W. COX, Nucl. Sci. Eng., 8, 670 (1960).

13. W. R. RHYNE and A. C. LAPSLEY, Nucl. Sci. Eng., 40, 91 (1970).

14. J. J. KAGANOVE, "Numerical Solution of the One-Group, Space-Independent Reactor Kinetics Equations for Neutron Density Given the Excess Reactivity", ANL-6132, Argonne National Laboratory (1960).

15. E. R. COHEN, "Some Topics in Reactor Kinetics", in Proc. 2nd U. N. Intern. Conf. Peaceful Uses At. Energy, 11, 302 (1958).

16. K. F. HANSEN, B. V. KOEN and W. W. LITTLE, Jr., Nucl. Sci. Eng., 22, 51 (1965).

17. C. M. KANG, Trans. Am. Nucl. Soc., 14, 201 (1971).

18. T. A. PORSCHING, Nucl. Sci. Eng., 25, 183 (1966).

19. K. F. HANSEN, E. R. KLEBAN and I. J. CHANG, Trans. Am. Nucl. Soc., 12, 617 (1969).

20. M. IZUMI and T. NODA, Nucl. Sci. Eng., 41, 299 (1970).

21. A. F. HENRY, Personal Communication (January 1970).

22. M. CLARK, Jr. and K. F. HANSEN, Numerical Methods of Reactor Analysis, Sec. 1.12, Academic Press, New York and London (1964).

23. E. ISAACSON and H. B. KELLER, Analysis of Numerical Methods, Sec. 1, John Wiley & Sons, Inc., New York, London and Sidney (1966).

24. R. S. VARGA, Matrix Iterative Analysis, Sec. 8.3, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1962).

25. M. CLARK, Jr., and K. F. HANSEN, Op. Cit., Sec. 4.5.

26. K. F. HANSEN, "A Comparative Review of Two-Dimensional Kinetics Methods", GA-8169, General Atomic Division of General Dynamics (1967).

27. C. G. CHEZEM and W. H. KOEHLER, Eds., "Coupled Reactor Kinetics", Proc. Nat. Topical Meeting Am. Nucl. Soc., Texas A&M Press, College Station, Texas (1967).

28. S. KAPLAN, Nucl. Sci. Eng., 9, 357 (1961).

29. W. M. STACEY, Jr., Modal Approximations, The MIT Press, Cambridge, Massachusetts (1967).

30. S. KAPLAN, O. J. MARLOWE, and J. BEWICK, Nucl. Sci. Eng., 18, 163 (1964).

31. W. M. STACEY, Jr., Nucl. Sci. Eng., 34, 45 (1968).

32. J. B. ANDREWS, II, and K. F. HANSEN, Nucl. Sci. Eng., 31, 304,(1968).

33. W. T. McCORMICK, Jr., and K. F. HANSEN, "Numerical Solution of the Two-Dimensional Time-Dependent Multigroup Equations", in Proc. Conf. on Effective Use of Computers in Nucl. Ind., CONF-690401, p. 76,USAEC (1969).

34. W. H. REED and K. F. HANSEN, Nucl. Sci. Eng., 41, 431 (1970).

35. A. L. WIGHT, K. F. HANSEN and D. R. FERGUSON, Nucl Sci. Eng., 44, 239 (1971).

36. D. R. FERGUSON and K. F. HANSEN, "Solution of the Space-Dependent Reactor Kinetics Equations in Three Dimensions", MIT-3903-4 (MITNE-132), Ph. D. Thesis, Department of Nuclear Engineering, Massachusetts Institute of Technology (1971).

37. T. A. PORSCHING and A. F. HENRY, "Some Numerical Methods for Problems in Reactor Kinetics", in Numerical Solution of Field Problems in Continuum Physics, SIAM-AMS Proc., Vol. 2, p. 95, G. BIRKHOFF and R. S. VARGA, Eds., American Mathematical Society, Providence, R.I. (1970).

38. J. B. YASINSKY and L. A. HAGEMAN, "On the Solution of the Time-Dependent Group Diffusion Equations by an Implicit Time Differenced Iterative Method", in Proc. Conf. on Effective use of Computers in Nucl. Ind., CONF-690401, p. 55, USAEC (1969).

39. W. M. STACEY, Jr., Space-Time Nuclear Reactor Kinetics, Chaps. 1-3, Academic Press, New York and London (1969).

40. R. D. RICHTMYER and K. W. MORTON, Difference Methods for Initial-Value Problems, 2nd. Ed., Chap. 3, Interscience Publishers (Wiley), New York, London and Sydney (1967).

41. L. A. HAGEMAN and J. B. YASINSKY, Nucl. Sci. Eng., 38, 8 (1969).

42. R. S. VARGA, Op. Cit., Sec. 1.5.

43. R. S. VARGA, Op. Cit., Sec. 8.2.

44. E. L. WACHSPRESS, Iterative Solution of Elliptic Systems, Sec. 1.3, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1966).

45. W. M. STACEY, Jr., to be published.

46. J. B. YASINSKY, Nucl. Sci. Eng., 32, 425 (1968).

47. W. M. STACEY, Jr., and C. H. ADAMS, Trans. Am. Nucl. Soc., 10, 251 (1967).

Appendix A

CALCULATION OF THE EIGENVALUES OF LARGE MAGNITUDE

In what follows, we present a procedure to find the values of one or both of the two extreme eigenvalues of the point kinetics matrix A, which are the algebraically largest and smallest roots of the inhour formula, Eq. (2.3).

The calculation of ω_0 or ω_I can be done by table lookup and interpolation, by using the Newton-Raphson method, or by a combination of both schemes. In this last method, a first guess is obtained from a table lookup and the Newton-Raphson method is used to improve upon the initial guess to any desired degree of accuracy.

The Newton-Raphson method can be shown to converge in the following cases:

a) For ω_0, with some initial guess $\omega_0^{(0)}$ such that

$$-\lambda_1 < \omega_0^{(0)} < \omega_0.$$

b) For ω_I, with some initial guess $\omega_I^{(0)}$ such that

$$\omega_I < \omega_I^{(0)} < -\lambda_I.$$

It can also be shown that $\omega_I<\alpha<\omega_0$ and that, if α has a large enough magnitude, it is already a good approximation for one of the roots and can, thus, be used as a good first guess for that root.

The Newton-Raphson expression is derived from Eq. (2.3) and is given by

$$\omega^{(j+1)} = \frac{\rho - \sum_{i=1}^{I} \beta_i \left(\frac{\omega^{(j)}}{\lambda_i+\omega^{(j)}}\right)^2}{\Lambda + \sum_{i=I}^{I} \beta_i \lambda_i \left(\frac{1}{\lambda_i+\omega^{(j)}}\right)^2} \; .$$

In practice, with reasonable first guesses, two to five iterations are enough to achieve an accuracy of one part in 10^5. In order to appreciate the amount of work involved in the above calculations we estimated the computer time necessary to do one iteration, which, by efficient programming, can be reduced to 3I additions and subtractions, 4I multiplications and (I+1) divisions, for I precursor groups. If we take I=6 and assume $t_{\pm} = 2$ µsec, $t_{\times} = 4$ µsec and $t_{\div} =$ 7 µsec (approximately the IBM 360/65 times) one would spend about 180 µsec per iteration which, for comparison, is about the time needed to calculate two exponentials in single precision (in the IBM 360/65).

We can see, therefore, that the amount of work spent in the calculation of the desired roots is not unduly large, particularly in view of the relatively large time step sizes one will be able to take with the present method.

Appendix B

CALCULATION OF THE PARAMETERS THETA

In this appendix we describe the procedure used to find the thetas for use in Eq. (3.10) when based on a point kinetics calculation. The notation adopted in Chap. 2 is used throughout except when otherwise noted.

The case of constant parameters is dealt with by direct use of the methods of Chap. 2. For Method 1 we write Eq. (3.18) in the form

$$\frac{1}{h}(\underline{\Psi}_1-\underline{\Psi}_0) = A[\theta\underline{\Psi}_1+(I-\theta)\underline{\Psi}_0]$$

where $\theta=\mathrm{diag}[\theta_0\ \theta_1\ \theta_2\ \ldots\ \theta_I]$. From this and Eq. (2.2) without source we get

$$[(hA)^{-1}(e^{hA}-I) - I]\underline{\Psi}_0 = \theta[e^{hA}-I]\underline{\Psi}_0 \tag{1}$$

which, as we shall see, is defined even if A does not have an inverse. Using Eq. (2.8) with f(hA) given by the Padé (1,1) approximation, $f_1(hA)$, we get

$$[\tfrac{1}{2}g(hA) + \sum_k{}'\{\frac{e^{h\omega_k} - f(h\omega_k)}{h\omega_k}\}\underline{u}_k\underline{v}_k^T]\underline{\Psi}_0 =$$

$$\theta[g(hA) + \sum_k{}'\{e^{h\omega_k} - f(h\omega_k)\}\underline{u}_k\underline{v}_k^T]\underline{\Psi}_0 \tag{2}$$

with $g(hA)=[I-\frac{1}{2}hA]^{-1}hA$, from which θ can be calculated. Note that, if no terms are included in Σ', $\theta=\frac{1}{2}I$.

If the Padé (2,0) approximation, $f_2(hA)$, is used, there will, in general, be one less term in Σ'. But the computation of the expressions that replace $g(hA)$ will be more complicated. Since the error is smaller for the Padé (1,1) approximation, we decided to use it here.

We adopt the above procedure (i.e. Eq. (2)) to find the thetas since it is computationally simpler, and since it eliminates much of the error arising from the fact that Eq. (3.18) sometimes forms a quasi-singular system for the thetas. Equation (2) mathematically cancels the singularities due to small h or to the fact that A is singular when $\rho=0$.

For the case of constant parameters, Eq. (1) can also be intepreted as representing

$$\frac{1}{h}\int_0^h \underline{\Psi}(t+s)\,ds - \underline{\Psi}_0 = \theta(\underline{\Psi}_1-\underline{\Psi}_0) \qquad (3)$$

as can readily be verified by use of Eq. (2.2).

For Method 2 we determine θ_0 and θ_i' directly, rather than finding $\theta(s)$ and then using Eq. (3.14). When the definition of $\theta(s)$ in Eq. (3.19) and the expression for θ_0 in Eq. (3.14) are applied, Eq. (3) shows that θ_0 is the same for both Methods 1 and 2, and can thus be calculated using part of Eq. (2).

Equation (3.14) for θ_i' and τ_i leads to

$$\theta_i' = \frac{\frac{1}{h}\int_0^h e^{-\lambda_i(h-s)}\,n(t+s)\,ds - \tau_i n_0}{n_1 - n_0} = \frac{\dfrac{c_{i1}-e^{-h\lambda_i}c_{i0}}{h\mu_i} - \tau_i n_0}{n_1 - n_0}$$

where $(n_1 - n_0)$ can be obtained in the process of calculating θ_0.

Note that we could try also to decrease the error in the numerator by writing it in the form

$$\frac{1}{\mu_i}\left\{\left(\frac{c_{i1}-c_{i0}}{h} - \dot{c}_{i0}\right) + (1-\tau_i)\dot{c}_{i0}\right\}$$

where the first part could be obtained by evaluating $[e^{hA}-I-hA]\underline{\Psi}_0$ by a procedure similar to that used in Eq.(2). We have not implemented this procedure, however.

Appendix C

PROOF OF EQUIVALENCE OF NORMS

The L_2 norm was defined in Sec. (3.6) as being

$$\|\underline{u}\| = \left[\int_V |\underline{u}(r)|^2 dV\right]^{1/2} = \left[\sum_n u_n^2 w_n^2\right]^{1/2}$$

the last form applying to spatially discrete functions or vectors (supervectors). The supervector index n depends both on the spatial index k and on the position index in $\underline{u}(r_k)$. Here, w_n^2 are integration weights, proportional to $\Delta\mathbf{r}$ and related to the interpolation procedure chosen to make the discrete functions continuous.

The Euclidean norm of a vector (or supervector) is simply

$$\|\underline{v}\|' = [\underline{v}^T\underline{v}]^{1/2} = \left[\sum_n v_n^2\right]^{1/2}$$

so that, by defining $\Omega=\text{diag}[w_n]$, we can write the relation between the L_2 norm and the Euclidean norm as

$$\|\underline{u}\| = \|\Omega\underline{u}\|'$$

Now, for a matrix H we have, by definition,

$$\|H\| = \max_{\|\underline{u}\|=1} \|H\underline{u}\| = \max_{\|\Omega\underline{u}\|'=1} \|\Omega H\underline{u}\|' =$$

$$= \max_{\|\underline{x}\|'=1} \|\Omega H\Omega^{-1}\underline{x}\|' = \|\Omega H\Omega^{-1}\|'$$

where we made $\Omega\underline{u}=\underline{x}$ and used the definition of the Euclidean norm of a matrix[23].

In general

$$\|H\| = \|\Omega H \Omega^{-1}\|' \le \|\Omega\|' \|H\|' \|\Omega^{-1}\|' = \frac{w_{max}}{w_{min}} \|H\|'$$

and

$$\|H\|' = \|\Omega^{-1} \Omega H \Omega^{-1} \Omega\|' \le \|\Omega^{-1}\|' \|\Omega H \Omega^{-1}\|' \|\Omega\|' = \frac{w_{max}}{w_{min}} \|H\|$$

so that

$$k\|H\|' \le \|H\| \le K\|H\|'$$

where $K=1/k \ge w_{max}/w_{min}$ is a fixed constant, if the spatial mesh size is shrunk to zero uniformly, that is, if $\Delta r_{max}/\Delta r_{min}$ is bounded by some constant.

Appendix D

INPUT CARDS FOR COMPUTER PROGRAMS

D.1 Point Kinetics Programs

Card 1: FORMAT(I10,4D10.0). For the method based on Padé (1,1).

FORMAT(I10,3D10.0). For the method based on Padé (2,0).

NPG = number of neutron precursor groups, I.

HWS(1) = HWS = number such that, if $h\omega_0$>HWS, the ω_0 term is included in Σ', Eq. (2.8).

HWS(2) = number such that, if $h\omega_I<\min(\text{HWS}(2),-1.2h\lambda_I)$, the ω_I term is included in Σ', Eq. (2.8). Not used with Padé (2,0).

OMTOL = relative error for the computation of ω_0 and ω_I.

LAMBDA = neutron generation time, Λ.

Card 2: FORMAT(6D10.0)

(BETA(I),I=1,NPG) = delayed neutron fractions, β_i.

Card 3: FORMAT(6D10.0)

(LAM(I),I=1,NPG) = decay constants, λ_i.

Card 4: FORMAT(I10,4D10.0). One card for each time zone.

NPRINT = solution printed every NPRINT steps.

H = time step size, h.

TZ = end time for the time zone.

RHOØ = initial value of the reactivity for the time zone, ρ_0.

DRHO = derivative of the reactivity with respect to time for the time zone, $\dot{\rho}_0$.

Observations: (a) $\rho(t)=\rho_0+\dot{\rho}_0 t$.

(b) the programs treat only problems without external source, in their present version.

(c) to run several cases the whole set of cards described before (1 to 4) is repeated as many times as there are cases, with a blank card separating the set of cards for consecutive cases.

D.2 One-Dimensional Space-Kinetics Program

Card 1: FORMAT(5I5)

NREG = number of homogeneous reactor regions (≤ 10).

NPTS = number of spatial points whose fluxes are to be printed out (≤ 16).

II = number of delayed neutron groups, I (≤ 6).

IREAD = 1, if initial flux and adjoint are to be read in; blank, otherwise.

IPUNCH = 1, if calculated initial flux and adjoint are to be punched out; blank, otherwise.

Card 2: FORMAT(16I5)

(KPRINT(N) ,N=1,NPTS) = spatial mesh point numbers of points whose fluxes will be printed out.

Card 3: FORMAT(2D10.0)

CRITOL = relative errors for eigenvalue (k_{eff}) and for pointwise flux and adjoint.

OVER = overrelaxation parameter, between 1 and 2.

Card 4: FORMAT(4D10.0)

(V(G),G=1,2) = group velocities, v_g.

(CHIO(G),G=1,2) = prompt neutron spectrum, χ_{0g}.

Card 5: FORMAT(6D10.0)

(BETA(I),I=1,II) = delayed neutron fractions, β_i.

Card 6: FORMAT(6D10.0)

(LAM(I),I=1,II) = decay constants, λ_i.

Card 7: FORMAT(6D10.0). One card for each neutron energy group.

(CHII(G,I),I=1,II) = delayed neutron energy spectrum χ_{ig}.

Card 8: A set of two cards for each reactor region.

Card 8-A: FORMAT(8D10.0)

(D(G),G=1,2) = diffusion constants, D_g.

((SIG(G,G'),G'=1,2),G=1,2) = cross section matrix, $\Sigma'_{gg'}$ =

$= \Sigma_{Tg}\ \delta_{gg'} - \Sigma_{sgg'}$.

(SFN(G),G=1,2) = neutron production cross sections, $\nu\Sigma_{fg}$.

Card 8-B: FORMAT(2D10.0)

X = width of the region.

KINT = number of mesh intervals in the region.

Card 9: FORMAT(4D20.0). One card for each interior mesh point, if IREAD=1.

(FLUX(G,K),G=1,2) = initial flux, ϕ_{gk}.

(ADJ(G,K),G=1,2) = initial adjoint, ϕ^*_{gk}.

Card 10: FORMAT(I5)

NTZ = number of time zones.

Card 11: A set of (NREG+1) cards for each time zone.

Card 11-A: FORMAT(I5,D10.0)

IRAMP = 0 if step, 1 if ramp.

TZ = end time for the time zone.

Card 11-B: FORMAT(4D10.0). One card for each reactor region.

(DSA(G),G=1,2) = $\Delta(\Sigma_{ag})$ if step, or $d(\Sigma_{ag})/dt$, if ramp.

(DSFN(G),G=1,2) = $\Delta(\nu\Sigma_{fg})$ if step, or $d(\nu\Sigma_{fg})/dt$, if ramp.

Card 12: FORMAT(2D10.0,I10). One for each time zone.

H = time step size.

HPRINT = printing time interval.

NT1 = method selection number: if $NT1 \geq 0$ we have the first NT1 steps with $\theta_0=1$ and subsequent steps using point kinetics thetas; if $NT1<0$ we have first $-(1+NT1)$ steps using $\theta_0=1$ and subsequent steps using $\theta_0=0.5$.

Observations: (a) $\Sigma(0^+)=\Sigma(0^-)+\Delta\Sigma$, if step and $\Sigma(t)=\Sigma(0)+t\dot{\Sigma}$, if ramp. For constant parameter zone without initial jump it is more efficient to take step with $\Delta\Sigma=0$ instead of ramp with $\dot{\Sigma}=0$.

(b) the maximum number of inner mesh points is 100.

(c) the program treats only two-group slab reactors with zero flux boundary conditions at both boundaries.

(d) to run the same problem with different H, HPRINT, or NT1 repeat card set 12 (NTZ cards) as many times as desired, separating consecutive sets by a blank card.

(e) to run different transients for the same reactor, repeat card sets 10 to 12 as many times as desired, separating consecutive transients by a card with 1 punched in the first column.

(f) to run different transients in different reactors, repeat all cards sets from 1 to 12, as many times as desired, separating consecutive cases by a card with 2 punched in the first column.

(g) any combination of possibilities described in items (d), (e), and (f) is permitted.

(h) logical unit 8 is used for input/output so that a control card for FT08F001 has to be included.

Appendix E

LISTINGS OF COMPUTER PROGRAMS

(only in first six copies)

The following pages contain the source listings of the computer programs enumerated below, which were used to do the calculations in this thesis.

E.1 Point Kinetics Programs

E.1.1 Method Based on the Padé (2,0) Approximation

a) MAIN PROGRAM
b) FUNCTION REACT
c) SUBROUTINE ROOT

E.1.2 Method Based on the Padé (1,1) Approximation

a) MAIN PROGRAM
b) FUNCTION SUM
c) SUBROUTINE ROOT
d) SUBROUTINE NEWRAP

E.2 One-Dimensional Space-Kinetics Program

a) MAIN PROGRAM
b) SUBROUTINE INPUT
c) SUBROUTINE PREP
d) SUBROUTINE CRITIC
e) SUBROUTINE INVERT
f) SUBROUTINE PQP
g) SUBROUTINE THETA (for Method 1)
SUBROUTINE THETA (for Method 2)
h) SUBROUTINE POINTK (for Method 1)
SUBROUTINE POINTK (for Method 2)
i) SUBROUTINE ROOT
j) SUBROUTINE SOURCE

```
C      MAIN PROGRAM
C
C      METHOD BASED ON THE PADE (2,0) APPROXIMATION
C
       IMPLICIT REAL*8 (A-H,O-Z)
       COMMON LAMBDA,LAM(6),BETA(6),M(6),RO,ALFA,OMTOL,NPG
       COMMON /TABLE/ W,WS,WT(20),ROT(20)
       DIMENSION PHIO(6),PHI1(6),P(6),A1(6),A2(6),B1(6),B2(6),B(6)
       REAL*8 LAMBDA, LAM, M
       LOGICAL RAMP, DYAD
C      ARITHMETIC STATEMENT FUNCTION
       FEXP(X)=1.0/(1.0-X+0.5*X**2)
C      INPUT DATA.
 101   WRITE (6,102)
 102   FORMAT ('1*********INPUT DATA**********')
       READ(5,3) NPG,HWS,OMTOL,LAMBDA,BETA,LAM
 3     FORMAT(I10,3D10.0/(6D10.0))
       WRITE(6,4) NPG,HWS,OMTOL,LAMBDA,(BETA(I), I=1,NPG)
 4    0FORMAT ('0NPG= 'I1,8X,'HW*=',1PD8.2,8X,'OMEGA ERROR=',D8.2,8X,'LAM
      1BDA=',D10.4/'0BETAS'/6G15.5)
       WRITE(6,5) (LAM(I), I=1,NPG)
 5     FORMAT('0LAMBDAS'/1P6G15.5)
C      INITIAL CALCULATIONS AND CONDITIONS.
       W=40.0
       DO 6 N=1,20
       WT(N)=W
       ROT(N)=REACT(W)
 6     W=0.7*W
       PHIOO=1.0
       BETAT=0.0
       DO 8 I=1,NPG
       BETAT=BETAT+BETA(I)
       M(I)=BETA(I)/LAMBDA
 8     PHIO(I)=M(I)/LAM(I)
       WRITE(6,9) BETAT,NPG,PHIOO,(PHIO(I),I=1,NPG)
 9    0FORMAT('0BETA='1PD13.5/'0INITIAL VALUES, PHI(0) TO PHI(',I1,')'/7D
```

```
     1 15.6)
C     PREPARATION OF TIME ZONE.
      T=0.0
      HOLD=0.0
  106 READ(5,1,END=21) NPRINT,H,TZ,RHOO,DRHO
    1 FORMAT(I10,4D10.0)
      IF(NPRINT.EQ.0) GO TO 101
      WRITE(6,2) H,TZ,RHOO,DRHO
    2 CFORMAT('ONEW TIME ZONE'/'OH=',1PD10.4,5X,'END TIME=',G10.4,5X,'RHO
     10=',G11.4,5X,'DRHO/DT='G11.4/)
      IT=(TZ-T)/H+0.1
      KPRINT=0
      IF(H.EQ.HOLD) GO TO 104
      HOLD=H
      R1=1.0
      S=1.0
      DO 103 I=1,NPG
      TEMP=H*LAM(I)
      TEMP1=TEMP+1.0
      P(I)=1.0/(TEMP1**2+1.0)
      A2(I)=H*M(I)*P(I)
      A1(I)=A2(I)*TEMP1
      R1=R1+A2(I)*(TEMP+2.0)
      S=S+A2(I)*TEMP
      B2(I)=TEMP*P(I)
      B1(I)=B2(I)*TEMP1
  103 P(I)=2.0*P(I)
      S2=S**2
      HH=H/LAMBDA
      WS=HWS/H
      RHOS=REACT(WS)
  104 DRHO=DRHO*H
      RAMP=DRHO.NE.0.0
      RO=RHOO
      IF(RAMP) RO=RO-0.5*DRHO
      DO 20 K=1,IT
```

```
C     INITIATE TIME STEP CALCULATIONS.                                  MAIN0073
      IF(RAMP) RO=RO+DRHO                                               MAIN0074
 10   IF(.NOT.RAMP.AND.K.GT.1) GO TO 110                                MAIN0075
      ALFA=(RO-BETAT)/LAMBDA                                            MAIN0076
      R=R1-HH*RO                                                        MAIN0077
      R2=R**2                                                           MAIN0078
      DYAD=RO.GT.RHOS                                                   MAIN0079
      IF(DYAD) CALL ROOT                                                MAIN0080
 110  T=T+H                                                             MAIN0081
C     CALCULATION OF "EXP"(A)*PHIO(I).                                  MAIN0082
      TEMP=2.0/(R2+S2)                                                  MAIN0083
      C1=R*TEMP                                                         MAIN0084
      C2=S*TEMP                                                         MAIN0085
      TEMP1=PHI00                                                       MAIN0086
      TEMP2=0.0                                                         MAIN0087
      DO 11 I=1,NPG                                                     MAIN0088
      TEMP1=TEMP1+B1(I)*PHIO(I)                                         MAIN0089
 11   TEMP2=TEMP2+B2(I)*PHIO(I)                                         MAIN0090
      PHI10=C2*TEMP1+C1*TEMP2                                           MAIN0091
      TEMP3=C1*TEMP1-C2*TEMP2                                           MAIN0092
      DO 12 I=1,NPG                                                     MAIN0093
 12   PHI1(I)=PHI10*A1(I)+TEMP3*A2(I)+P(I)*PHIO(I)                      MAIN0094
      IF(.NOT.DYAD) GO TO 17                                            MAIN0095
      HW=H*W                                                            MAIN0096
      TEMP1=PHIOO                                                       MAIN0097
      TEMP=DEXP(HW)-FEXP(HW)                                            MAIN0098
      TEMP2=1.0                                                         MAIN0099
      DO 13 I=1,NPG                                                     MAIN0100
      A=LAM(I)+W                                                        MAIN0101
      B(I)=M(I)/A                                                       MAIN0102
      TEMP3=LAM(I)/A                                                    MAIN0103
      TEMP2=TEMP2+TEMP3*B(I)                                            MAIN0104
 13   TEMP1=TEMP1+PHIO(I)*TEMP3                                         MAIN0105
      TEMP=TEMP1*TEMP/TEMP2                                             MAIN0106
      TEMP1=TEMP/HW                                                     MAIN0107
      PHI10=PHI10+TEMP                                                  MAIN0108
```

```
      DO 14 I=1,NPG                                                     MAIN0109
   14 PHI1(I)=PHI1(I)+TEMP*E(I)                                         MAIN0110
   17 PHI00=PHI10                                                       MAIN0111
      DO 16 I=1,NPG                                                     MAIN0112
   16 PHI0(I)=PHI1(I)                                                   MAIN0113
      KPRINT=KPRINT+1                                                   MAIN0114
      IF(KPRINT.NE.NPRINT) GO TO 20                                     MAIN0115
      KPRINT=0                                                          MAIN0116
      WRITE(6,18) T,PHI10,(PHI1(I),I=1,NPG)                             MAIN0117
   18 FORMAT(' T=',F7.3,' SEC;',3X,'N=',1PD14.6,';   CI=',6D12.4)       MAIN0118
C     END OF TIME STEP CALCULATIONS                                     MAIN0119
   20 CONTINUE                                                          MAIN0120
      GO TO 106                                                         MAIN0121
   21 STOP                                                              MAIN0122
      END                                                               MAIN0123
```

```
      FUNCTION REACT(W)
C     CALCULATION OF THE REACTIVITY AS A FUNCTION OF OMEGA.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON LAMBDA,LAM(6),BETA(6),M(6),RO,ALFA,OMTOL,NPG
      REAL*8 LAMBDA,LAM,M
      REACT=LAMBDA
      DO 1 I=1,NPG
    1 REACT=REACT+BETA(I)/(LAM(I)+W)
      REACT=REACT*W
      RETURN
      END
```

```
      SUBROUTINE ROOT
C
C     FOR USE IN THE PADE(2,0) BASED METHOD
C     CALCULATION OF OMEGA(C).
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON LAMBDA,LAM(6),BETA(6),M(6),RO,ALFA,OMTOL,NPG
      COMMON /TABLE/ W,WS,WT(20),ROT(20)
      REAL*8 LAM,LAMBDA,M
      DO 2 N=1,20
      IF(RO.LT.ROT(N)) GO TO 2
      W=WT(N)
      GO TO 3
 2    CONTINUE
      W=0
 3    W=DMAX1(ALFA,WS,W)
 4    WO=W
      SUM1=ALFA
      SUM2=1.0
      DO 1 I=1,NPG
      TEMP1=W+LAM(I)
      TEMP=LAM(I)*M(I)/TEMP1**2
      SUM1=SUM1+(TEMP1+W)*TEMP
 1    SUM2=SUM2+TEMP
      W=SUM1/SUM2
      IF((W-WO)/W.GT.OMTOL) GO TO 4
      RETURN
      END
```

```
C      MAIN PROGRAM
C
C      METHOD BASED ON THE PADE (1,1) APPROXIMATION
C
       IMPLICIT REAL*8 (A-H,O-Z)
      CCOMMON /CM1/WS(2),RHOS(2),RO,ALFA,OMTOL /CM2/BETA(6),LAM(6),LAMBDA
      1,NPG
       DIMENSION A(6),B(6),W(2),PHIO(6),PHI1(6),THETA(6),IND(2),HWS(2)
       REAL*8 LAMBDA, LAM, MU(6)
C      ARITHMETIC STATEMENT FUNCTION
       FEXP(X)=(2.0+X)/(2.0-X)
C      INPUT DATA.
 101   WRITE (6,102)
 102   FORMAT ('1*********INPUT DATA*********')
       READ(5,3) NPG,HWS,OMTOL,LAMBDA,BETA,LAM
 3     FORMAT(I10,4D10.0/(6D10.0))
       WRITE(6,4) NPG,HWS,OMTOL,LAMBDA,(BETA(I), I=1,NPG)
 4    OFORMAT('ONPG= ',I1,8X,'HW1*=',1PD10.2,5X,'HW2*=',D10.2,8X,'OMEGA E
      1RROR=',D8.2,8X,'LAMBDA=',D10.4/'OBETAS'/6D15.5)
       WRITE(6,5) (LAM(I), I=1,NPG)
 5     FORMAT('OLAMBDAS'/1P6G15.5)
C      INITIAL CALCULATIONS AND CONDITIONS.
       PHIO0=1.0
       DO 8 I=1,NPG
       MU(I)=BETA(I)/LAMBDA
 8     PHIO(I)=MU(I)/LAM(I)
       BETAT=SUM(NPG,BETA)
       WRITE(6,9) BETAT,NPG,PHIO0,(PHIO(I),I=1,NPG)
 9    OFORMAT('OBETA='1PG13.5/'OINITIAL VALUES, PHI(0) TO PHI(',I1,')'/7D
      115.6)
C      PREPARATION OF TIME ZONE.
       T=0.0
 106   READ(5,1,END=21) NPRINT,H,TZ,RHO0,DRHO
 1     FORMAT(I10,4D10.0)
       IF(NPRINT.EQ.0) GO TO 101
       WRITE(6,2) H,TZ,RHO0,DRHO
```

```
  2   0FORMAT('ONEW TIME ZONE'/'0H=',1PD10.4,5X,'END TIME=',G10.4,5X,'RHO
     10=',G11.4,5X,'DRHO/DT=',G11.4/)
      IT=(TZ-T)/H+0.1
      HH=0.5*H
      ETA=1.0/HH
      DO 7 N=1,2
      WS(N)=HWS(N)/H
      IF(N.EQ.2) WS(2)=DMIN1(-1.2*LAM(NPG),WS(2))
      DO 6 I=1,NPG
  6   A(I)=BETA(I)/(LAM(I)+WS(N))
  7   RHOS(N)=WS(N)*(LAMBDA+SUM(NPG,A))
      DO 20 K=1,IT
C     INITIATE TIME STEP CALCULATIONS.
      RO=RHO0
      IF (DRHO.EQ.0.0) GO TO 10
      RHO1=RHO0+H*DRHO
      RO=(RHO1+RHO0)/2.0
      RHO0=RHO1
 10   IF(DRHO.EQ.0.0.AND.K.GT.1) GO TO 110
      ALFA=(RO-BETAT)/LAMBDA
      CALL ROOT(1,IND(1),W(1))
      CALL ROOT(2,IND(2),W(2))
 110  T=T+H
C     CALCULATION OF "EXP"(A)*PHI0(I).
      TEMP=1.0-RO/(LAMBDA*ETA)
      TEMP1=PHI00
      DO 11 I=1,NPG
      A(I)=ETA+LAM(I)
      PHI1(I)=ETA/A(I)
      B(I)=MU(I)/A(I)
      TEMP=TEMP+B(I)
 11   TEMP1=TEMP1+LAM(I)/A(I)*PHI0(I)
      TEMP=TEMP1/TEMP
      DO 12 I=1,NPG
 12   PHI1(I)=PHI1(I)*PHI0(I)+TEMP*B(I)
      THETA0=ALFA*TEMP
```

```
      DO 13 I=1,NPG
      TEMP1=PHI1(I)*LAM(I)
      THETA0=THETA0+TEMP1
      THETA(I)=(MU(I)*TEMP-TEMP1)*HH
   13 PHI1(I)=PHI1(I)+THETA(I)
      THETA0=THETA0*HH
      PHI10=THETA0+TEMP
      DO 15 N=1,2
      IF(IND(N).EQ.0) GO TO 15
      HW=H*W(N)
      IF(DABS(HW-2.0).GT.1.0E-5) GO TO 113
      WRITE (6,112)
  112 FORMAT('0***** ERROR: FEXP IS SINGULAR, HW=2. *****')
      GO TO 101
  113 TEMP=DEXP(HW)-FEXP(HW)
      TEMP1=PHI00
      TEMP2=1.0
      DO 14 I=1,NPG
      A(I)=LAM(I)+W(N)
      B(I)=MU(I)/A(I)
      TEMP3=LAM(I)/A(I)
      TEMP2=TEMP2+TEMP3*B(I)
   14 TEMP1=TEMP1+PHI0(I)*TEMP3
      TEMP=TEMP1*TEMP/TEMP2
      TEMP1=TEMP/HW
      THETA0=THETA0+TEMP1
      PHI10=PHI10+TEMP
      DO 115 I=1,NPG
      THETA(I)=THETA(I)+TEMP1*B(I)
  115 PHI1(I)=PHI1(I)+TEMP*B(I)
   15 CONTINUE
C     CALCULATION OF THETAS.
      TEMP=PHI10-PHI00
      IF(DABS(TEMP/PHI10).LT.1.E-7) TEMP=2.0*THETA0
      THETA0=THETA0/TEMP
      PHI00=PHI10
```

```
      DO 16 I=1,NPG                                                     MAIN0109
      TEMP=PHI1(I)-PHI0(I)                                              MAIN0110
      IF(DABS(TEMP/PHI1(I)).LT.1.E-5) TEMP=2.0*THETA(I)                 MAIN0111
      THETA(I)=THETA(I)/TEMP                                            MAIN0112
   16 PHI0(I)=PHI1(I)                                                   MAIN0113
      IF(MOD(K,NPRINT).NE.0) GO TO 20                                   MAIN0114
      WRITE(6,18) T,PHI10,(PHI1(I), I=1,NPG),THETA0,(THETA(I), I=1,NPG) MAIN0115
   18 0FORMAT(' T=',F7.3,' SEC;',3X,'N=',1PD14.6,';   CI=',6D12.4/10X,'THMAIN0116
     1ETA0=',0PF10.6,';   THETA(I)=',6F10.6)                            MAIN0117
C     END OF TIME STEP CALCULATIONS                                     MAIN0118
   20 CONTINUE                                                          MAIN0119
      GO TO 106                                                         MAIN0120
   21 STOP                                                              MAIN0121
      END                                                               MAIN0122
```

```
      FUNCTION SUM(N,A)
      IMPLICIT REAL*8 (A-H,O-Z)
C     SUMMATION FUNCTION, INDEX 1 TO N.
      DIMENSION A(N)
      SUM=A(1)
      IF(N.EQ.1) RETURN
      DO 1 I=2,N
    1 SUM=SUM+A(I)
      RETURN
      END
```

```
      SUBROUTINE ROOT(K,I,W)
C
C     FOR USE IN THE PADE(1,1) BASED METHOD
C
C     CALCULATION OF EXTREME ROOTS OF INHOUR FORMULA.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /CM1/WS(2),RHOS(2),RO,ALFA,OMTOL
      I=0
      IF(RHOS(1).GT.RO.AND.K.EQ.1.OR.RHOS(2).LT.RO.AND.K.EQ.2) RETURN
      W=WS(K)
      I=1
      IF(ALFA.GT.W.AND.K.EQ.1.OR.ALFA.LT.W.AND.K.EQ.2) W=ALFA
      NIT=0
    1 WO=W
      NIT=NIT+1
      CALL NEWRAP (RO,WO,W)
      IF ((W-WO)/W.GT.OMTOL) GO TO 1
      RETURN
      END
```

```
      SUBROUTINE NEWRAP (RO,WO,W)                                       NEWR0001
      IMPLICIT REAL*8 (A-H,O-Z)                                         NEWR0002
      COMMON /CM2/ BETA(6),LAM(6),LAMBDA,NPG                            NEWR0003
      REAL*8 LAMBDA, LAM, ML(6)                                         NEWR0004
      SUM1=RO                                                           NEWR0005
      SUM2=LAMBDA                                                       NEWR0006
      DO 1 I=1,NPG                                                      NEWR0007
      TEMP=BETA(I)/(WO+LAM(I))**2                                       NEWR0008
      SUM1=SUM1-TEMP*WO*WO                                              NEWR0009
    1 SUM2=SUM2+TEMP*LAM(I)                                             NEWR0010
      W=SUM1/SUM2                                                       NEWR0011
      RETURN                                                            NEWR0012
      END                                                               NEWR0013
```

```
C     MAIN PROGRAM
C
      IMPLICIT REAL*8 (A-H,O-Z)
     0COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETAO,TETA(6),TAU(6),EHL
     1(6),DALFA,DML(6),H,II
      COMMON /INTEG/KK,NREG,KINT(10)
      COMMON /ARRAY/B(2,2,100),C(2,100),F(2,100),P(100),PREC(6,100)
      COMMON /CONST/D(2,10),SIG(2,2,10),CHI(2),SFN(2,10)
      COMMON /VCHII/V(2),CHII(2,6),DX(10)
      COMMON /CRIT/CRITOL,CVER,NIT,IREAD
      COMMON /DELTA/DB(2,2,10),DSFN(2,10)
      COMMON /RDINP/X(10),CHIO(2),KPRINT(16),NPTS,IPUNCH
      COMMON /PKP/ S(2,100),B1(2,2,100)
      REAL*8 LAM(6),MU(6)
      LOGICAL RAMP,LFLUX
      DATA LITO/'STEP'/, LIT1/'RAMP'/
      DIMENSION FLUX(2,100),ADJ(2,100),VI(2),DSA(2,10),BCHI(2)
C
C     READING OF INPUT DATA
C     NPTS<17, II<7, KK<101, NREG<11, NUMBER OF MESH INTERVALS/REGION >1
C
 10   CALL INPUT
C
C     INITIAL CRITICALITY CALCULATIONS & PREPARATION OF BASIC CONSTANTS
C
      DO 13 N=1,NREG
      DX(N)=X(N)/KINT(N)
      IF(N.GT.1) KINT(N)=KINT(N)+KINT(N-1)
      DO 13 J=1,2
      D(J,N)=D(J,N)/DX(N)
      SIG(J,J,N)=SIG(J,J,N)*DX(N)
      JC=3-J
      SIG(J,JC,N)=SIG(J,JC,N)*DX(N)
 13   SFN(J,N)=SFN(J,N)*DX(N)
      KK=KINT(NREG)-1
      CHI(1)=0.0
```

```
      CHI(2)=0.0
      CALL PREP
      DO 11 J=1,2
      CHI(J)=CHIO(J)
      DO 11 I=1,II
   11 CHI(J)=CHI(J)+BETA(I)*(CHII(J,I)-CHIO(J))
     0IF(IREAD.EQ.1) READ(5,14) (FLUX(1,K),FLUX(2,K),ADJ(1,K),ADJ(2,K),
     1K=1,KK)
   14 FORMAT(1P4D20.13)
      LFLUX=.TRUE.
      CALL CRITIC(FLUX,EIGF,LFLUX)
      WRITE(6,17) EIGF,NIT,(FLUX(1,K),K=1,KK)
   17 0FORMAT('1FLUX EIGENVALUE=',1PG16.9,I20,' ITERATIONS'/'0INITIAL FLU
     1X'/(6D17.7))
      WRITE(6,18) (FLUX(2,K),K=1,KK)
   18 FORMAT(/(1P6D17.7))
      LFLUX=.FALSE.
      CALL CRITIC(ADJ,EIGA,LFLUX)
      WRITE(6,20) EIGA,NIT,(ADJ(1,K),K=1,KK)
   20 0FORMAT(//'0ADJOINT EIGENVALUE=',1PG16.9,I20,' ITERATIONS'/'0ADJOIN
     1T FOR INITIAL STATE'/(6D17.7))
      WRITE(6,18) (ADJ(2,K),K=1,KK)
     0IF(IPUNCH.EQ.1) WRITE(7,14) (FLUX(1,K),FLUX(2,K),ADJ(1,K),ADJ(2,K)
     1, K=1,KK)
      EIGI=2.0/(EIGF+EIGA)
      DO 15 N=1,NREG
      DO 15 J=1,2
   15 SFN(J,N)=SFN(J,N)*EIGI
C
C     CALCULATION OF INITIAL PRECURSOR CONCENTRATION
C
      CALL PREP
      DO 21 K=1,KK
      P(K)=F(1,K)*FLUX(1,K)+F(2,K)*FLUX(2,K)
      DO 21 I=1,II
   21 PREC(I,K)=BETA(I)*P(K)/LAM(I)
```

```
C                                                                        MAIN0073
C     TIME CALCULATIONS. PREPARATION FOR TIME ZONE.                      MAIN0074
C                                                                        MAIN0075
     0WRITE(8) ((SIG(J,1,N),SIG(J,2,N),SFN(J,N),N=1,NREG),CHI(J),J=1,2), MAIN0076
     1((FLUX(J,K),F(J,K),B(J,1,K),B(J,2,K),J=1,2),(PREC(I,K),I=1,II),P(K MAIN0077
     2),K=1,KK)                                                          MAIN0078
 5    READ(5,1) NTZ                                                      MAIN0079
      WRITE(6,6)                                                         MAIN0080
 6    FORMAT('1')                                                        MAIN0081
      DO 2 L=1,NTZ                                                       MAIN0082
      READ(5,1)IRAMP,TZ,(DSA(1,J),DSA(2,J),DSFN(1,J),DSFN(2,J),J=1,NREG)  MAIN0083
 1    FORMAT(I5,D10.0/(4D10.0))                                          MAIN0084
      RAMP=IRAMP.EQ.1                                                    MAIN0085
      LIT=LIT0                                                           MAIN0086
      IF(RAMP) LIT=LIT1                                                  MAIN0087
      WRITE(8) RAMP,TZ,(DSA(1,J),DSA(2,J),DSFN(1,J),DSFN(2,J),J=1,NREG)   MAIN0088
 2   0WRITE(6,22) L,TZ,LIT,(J,DSA(1,J),DSA(2,J),DSFN(1,J),DSFN(2,J),J=1, MAIN0089
     1NREG)                                                              MAIN0090
 22  0FORMAT('0TIME ZONE',I5,'; UP TO  T=',F11.6,' SECONDS',10X,A4,' PER MAIN0091
     1TURBATION'/'0REGION',6X,'DSA1',11X,'DSA2',10X,'DSFN1',10X,'DSFN2'/ MAIN0092
     2(I5,1P4D15.4))                                                     MAIN0093
      REWIND 8                                                           MAIN0094
      READ(8)                                                            MAIN0095
 119  TIME=0.0                                                           MAIN0096
      WRITE(6,7)                                                         MAIN0097
 7    FORMAT('1STARTING CALCULATION OF THE TRANSIENT')                   MAIN0098
      DO 100 L=1,NTZ                                                     MAIN0099
      READ(5,4) H,HPRINT,NT1                                             MAIN0100
 4    FORMAT(2D10.0,I10)                                                 MAIN0101
      WRITE(6,24) L,H,NT1                                                MAIN0102
 24   FORMAT('0*** TIME ZONE',I5,': H=',1PD11.4,I10,' THETAS=1 ***'/'0') MAIN0103
      READ(8) RAMP,TZ,(DSA(1,J),DSA(2,J),DSFN(1,J),DSFN(2,J),J=1,NREG)    MAIN0104
      NSTEP=(TZ-TIME)/H+0.1                                              MAIN0105
      NPRINT=HPRINT/H+0.1                                                MAIN0106
      DO 23 N=1,NREG                                                     MAIN0107
      TEMP=DX(N)                                                         MAIN0108
```

```
      IF(RAMP) TEMP=TEMP*H
      DO 23 J=1,2
      DSA(J,N)=DSA(J,N)*TEMP
      DSFN(J,N)=DSFN(J,N)*TEMP
      JC=3-J
      IF(RAMP) GO TO 19
      SIG(J,J,N)=SIG(J,J,N)+DSA(J,N)
      SFN(J,N)=SFN(J,N)+DSFN(J,N)
      DSFN(J,N)=0.0
      DB(J,J,N)=0.0
      DB(J,JC,N)=0.0
      GO TO 23
 19   DB(J,J,N)=CHI(J)*DSFN(J,N)-DSA(J,N)
      DB(J,JC,N)=CHI(J)*DSFN(JC,N)
 23   CONTINUE
C
C     CALCULATION OF TAU(I) AND P(K).
C
      DO 12 I=1,II
      TEMP=H*LAM(I)
      EHL(I)=DEXP(-TEMP)
 12   TAU(I)=(1.0-EHL(I))/TEMP
      IF(RAMP) GO TO 124
      CALL PREP
      DO 123 K=1,KK
 123  P(K)=F(1,K)*FLUX(1,K)+F(2,K)*FLUX(2,K)
C
C     TIME STEP CALCULATIONS.
C
 124  DO 100 N=1,NSTEP
      CALL PQP(ADJ,FLUX)
      CALL THETA(RAMP,NT1,N)
      CALL SOURCE
C
C     PARTIAL PRECURSOR CALCULATION
C
```

```
      BCHI(1)=0.0
      BCHI(2)=0.0
      DO 128 I=1,II
      DO 29 J=1,2
 29   BCHI(J)=BCHI(J)+BETA(I)*TETA(I)/TETAO*CHII(J,I)
      TEMP=H*BETA(I)*(TAU(I)-TETA(I))
      DO 128 K=1,KK
 128  PREC(I,K)=EHL(I)*PREC(I,K)+TEMP*P(K)
C
C     PREPARING MATRICES FOR INVERT
C
      IF(.NOT.RAMP) GO TO 27
      DO 129 M=1,NREG
      DO 129 J=1,2
      SIG(J,J,M)=SIG(J,J,M)+DSA(J,M)
 129  SFN(J,M)=SFN(J,M)+DSFN(J,M)
      CALL PREP
 27   TEMP=1.0/(H*TETAO)
      VI(1)=TEMP/V(1)
      VI(2)=TEMP/V(2)
      M=1
      DO 26 K=1,KK
      TEMP=DX(M)
      IF(K.NE.KINT(M)) GO TO 126
      M=M+1
      TEMP=0.5*(DX(M)+TEMP)
 126  DO 26 J=1,2
      JC=3-J
      B1(J,J,K)=B(J,J,K)-VI(J)*TEMP-BCHI(J)*F(J,K)
 26   B1(J,JC,K)=B(J,JC,K)-BCHI(J)*F(JC,K)
      CALL INVERT(B1,C,FLUX,S)
      DO 127 K=1,KK
 127  P(K)=F(1,K)*FLUX(1,K)+F(2,K)*FLUX(2,K)
      DO 125 I=1,II
      TEMP=H*BETA(I)*TETA(I)
      DO 125 K=1,KK
```

```
 125  PREC(I,K)=PREC(I,K)+TEMP*P(K)
C
C     PRINTING OF CALCULATED VALUES
C
      TIME=TIME+H
      IF(MOD(N,NPRINT).NE.0) GO TO 100
      WRITE(6,28) TIME,(KPRINT(I),(FLUX(J,KPRINT(I)),J=1,2),I=1,NPTS)
 28  0FORMAT('0T=',1PD11.4,' SECONDS  (FLUX VALUES AT SELECTED POINTS)'/
     1'0(K)',8X,'G=1',13X,'G=2'/(I3,1X,2D16.6))
      WRITE(6,16)
 16   FORMAT('0')
 100  CONTINUE
      REWIND 8
      READ(5,3,END=25) INDEX
 3    FORMAT(I1)
      IF(INDEX.EQ.2) GO TO 10
     0READ(8) ((SIG(J,1,N),SIG(J,2,N),SFN(J,N),N=1,NREG),CHI(J),J=1,2),
     1((FLUX(J,K),F(J,K),B(J,1,K),B(J,2,K),J=1,2),(PREC(I,K),I=1,II),P(K
     2),K=1,KK)
      IF(INDEX.EQ.1) GO TO 5
      GO TO 119
 25   STOP
      END
```

```
      SUBROUTINE INPUT
      IMPLICIT REAL*8 (A-H,O-Z)
     0COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DMU(6),H,II
      COMMON /INTEG/KK,NREG,KINT(10)
      COMMON /CONST/D(2,10),SIG(2,2,10),CHI(2),SFN(2,10)
      COMMON /VCHII/V(2),CHII(2,6),DX(10)
      COMMON /CRIT/CRITOL,OVER,NIT,IREAD
      COMMON /RDINP/X(10),CHIO(2),KPRINT(16),NPTS,IPUNCH
      REAL*8 LAM(6),MU(6)
      READ(5,1) NREG,NPTS,II,IREAD,IPUNCH,KPRINT
1     FORMAT(5I5/16I5)
      READ(5,2) CRITOL,OVER
      READ(5,2) V,CHIO
      READ(5,2) BETA
      READ(5,2) LAM
      READ(5,2) (CHII(1,I),I=1,II)
      READ(5,2) (CHII(2,I),I=1,II)
2     FORMAT(8D10.0)
     0READ(5,3) (D(1,J),D(2,J),SIG(1,1,J),SIG(1,2,J),SIG(2,1,J),SIG(2,2,
     1 J),SFN(1,J),SFN(2,J),X(J),KINT(J), J=1,NREG)
3     FORMAT(8D10.0/D10.0,I5)
      WRITE(6,4) CRITOL,OVER
4    0FORMAT('1INPUT DATA'/'0CRITICALITY ERROR=',1PD8.2,5X,'RELAXATION P
     1ARAMETER=',0PF5.3)
      WRITE(6,5) V,CHIO,(BETA(I),I=1,II)
5    0FORMAT('0V(G)=    ',1P2D15.5,7X,'CHIO(G)=',2D15.5/'0BETA(I)=',6D15.
     1 5)
      WRITE(6,6) (LAM(I),I=1,II)
6     FORMAT('0LAM(I)= ',1P6D15.5)
      WRITE(6,7) (CHII(1,I),I=1,II)
7     FORMAT('0CHI(G,I), I=1 TO II'/' G=1',5X,1P6D15.5)
      WRITE(6,8) (CHII(2,I),I=1,II)
8     FORMAT(' G=2',5X,1P6D15.5)
     0WRITE(6,9) (J,X(J),KINT(J),D(1,J),SIG(1,1,J),SIG(1,2,J),SFN(1,J),D
     1(2,J),SIG(2,1,J),SIG(2,2,J),SFN(2,J), J=1,NREG)
```

```
9     CFORMAT(//4X,'REGION',I3,10X,'WIDTH=',0PF12.5,I12,' MESH INTERVALS'        INPU0037
     1/'0',6X,'D(G)',16X,'SIG(G,G'')',13X,'NU*SIGF(G)'/1P4D15.5/4D15.5)        INPU0038
      RETURN                                                                    INPU0039
      END                                                                       INPU0040
```

```
      SUBROUTINE PREP                                                   PREP0001
      IMPLICIT REAL*8 (A-H,O-Z)                                         PREP0002
      COMMON /INTEG/KK,NREG,KINT(10)                                    PREP0003
      COMMON /ARRAY/B(2,2,100),C(2,100),F(2,100),P(100),PREC(6,100)     PREP0004
      COMMON /CONST/D(2,10),SIG(2,2,10),CHI(2),SFN(2,10)                PREP0005
      DIMENSION BT(2,2,10)                                              PREP0006
      DO 1 N=1,NREG                                                     PREP0007
      DO 1 J=1,2                                                        PREP0008
      BT(J,J,N)=CHI(J)*SFN(J,N)-SIG(J,J,N)-2.0*D(J,N)                   PREP0009
      JC=3-J                                                            PREP0010
    1 BT(J,JC,N)=CHI(J)*SFN(JC,N)-SIG(J,JC,N)                           PREP0011
      N=1                                                               PREP0012
      DO 4 K=1,KK                                                       PREP0013
      IF(K.EQ.KINT(N)) GO TO 2                                          PREP0014
      DO 3 J=1,2                                                        PREP0015
      C(J,K)=D(J,N)                                                     PREP0016
      F(J,K)=SFN(J,N)                                                   PREP0017
      B(J,J,K)=BT(J,J,N)                                                PREP0018
      JC=3-J                                                            PREP0019
    3 B(J,JC,K)=BT(J,JC,N)                                              PREP0020
      GO TO 4                                                           PREP0021
    2 NM1=N                                                             PREP0022
      N=N+1                                                             PREP0023
      DO 5 J=1,2                                                        PREP0024
      C(J,K)=D(J,N)                                                     PREP0025
      F(J,K)=0.5*(SFN(J,NM1)+SFN(J,N))                                  PREP0026
      B(J,J,K)=0.5*(BT(J,J,NM1)+BT(J,J,N))                              PREP0027
      JC=3-J                                                            PREP0028
    5 B(J,JC,K)=0.5*(BT(J,JC,NM1)+BT(J,JC,N))                           PREP0029
    4 CONTINUE                                                          PREP0030
      RETURN                                                            PREP0031
      END                                                               PREP0032
```

```
      SUBROUTINE CRITIC(FLUX,EIG,LFLUX)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /INTEG/KK,NREG,KINT(10)
      COMMON /ARRAY/B(2,2,100),C(2,100),F(2,100),P(100),PREC(6,100)
      COMMON /CONST/D(2,10),SIG(2,2,10),CHI(2),SFN(2,10)
      COMMON /CRIT/CRITOL,CVER,NIT,IREAD
      DIMENSION FLUX(2,KK),S(2,100),FLUXO(2,100),FLUX1(2,100)
     0EQUIVALENCE (S(1),PREC(1)),(FLUXO(1),PREC(201)),(FLUX1(1),PREC(401
     1))
      LOGICAL LFLUX
      IF(LFLUX) GO TO 9
      DO 4 K=1,KK
      TEMP=B(1,2,K)
      B(1,2,K)=B(2,1,K)
    4 B(2,1,K)=TEMP
    9 NIT=0
      EIGO=0.0
      IF(IREAD.EQ.1) GO TO 15
      DO 8 K=1,KK
      DO 8 J=1,2
    8 FLUX(J,K)=1.0
   15 DO 16 K=1,KK
      DO 16 J=1,2
   16 FLUX1(J,K)=FLUX(J,K)
   10 TEMP=0.0
      DO 11 K=1,KK
      DO 11 J=1,2
   11 TEMP=TEMP+FLUX(J,K)**2
      TEMP=DSQRT(1.D0/TEMP)
      DO 12 K=1,KK
      DO 12 J=1,2
      FLUX(J,K)=FLUX(J,K)*TEMP
   12 FLUXO(J,K)=FLUX(J,K)
      DO 1 K=1,KK
      IF(LFLUX) GO TO 2
      TEMP=CHI(1)*FLUX(1,K)+CHI(2)*FLUX(2,K)
```

```
      DO 7 J=1,2                                                        CRIT0037
    7 S(J,K)=-TEMP*F(J,K)                                               CRIT0038
      GO TO 1                                                           CRIT0039
    2 TEMP=F(1,K)*FLUX(1,K)+F(2,K)*FLUX(2,K)                            CRIT0040
      DO 13 J=1,2                                                       CRIT0041
   13 S(J,K)=-TEMP*CHI(J)                                               CRIT0042
    1 CONTINUE                                                          CRIT0043
      CALL INVERT (B,C,FLUX,S)                                          CRIT0044
      NIT=NIT+1                                                         CRIT0045
      EIG=0.0                                                           CRIT0046
      DO 3 K=1,KK                                                       CRIT0047
      DO 3 J=1,2                                                        CRIT0048
    3 EIG=EIG+FLUX(J,K)**2                                              CRIT0049
      EIG=DSQRT(EIG)                                                    CRIT0050
      TEMP=1.0/EIG                                                      CRIT0051
      DO 6 K=1,KK                                                       CRIT0052
      DO 6 J=1,2                                                        CRIT0053
      FLUX(J,K)=FLUX(J,K)*TEMP                                          CRIT0054
      DFLUX=FLUX(J,K)-FLUX0(J,K)                                        CRIT0055
    6 FLUX(J,K)=FLUX0(J,K)+OVER*DFLUX                                   CRIT0056
      IF(MOD(NIT,10).NE.0) GO TO 10                                     CRIT0057
      IF(DABS(EIG-EIG0)/EIG.LT.CRITOL) GO TO 14                         CRIT0058
      EIG0=EIG                                                          CRIT0059
      GO TO 10                                                          CRIT0060
   14 TEMP=0.0                                                          CRIT0061
      DO 5 K=1,KK                                                       CRIT0062
      DO 5 J=1,2                                                        CRIT0063
    5 TEMP=DMAX1(TEMP,DABS(FLUX(J,K)-FLUX1(J,K))/FLUX(J,K))             CRIT0064
      IF(TEMP.GE.CRITOL) GO TO 15                                       CRIT0065
      INDEX=DSIGN(1.D0,FLUX(1,KK/2))                                    CRIT0066
      IF(INDEX.EQ.1) RETURN                                             CRIT0067
      DO 17 K=1,KK                                                      CRIT0068
      DO 17 J=1,2                                                       CRIT0069
   17 FLUX(J,K)=-FLUX(J,K)                                              CRIT0070
      RETURN                                                            CRIT0071
      END                                                               CRIT0072
```

```
      SUBROUTINE INVERT(B,C,V,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /INTEG/KK,NREG,KINT(10)
     CDIMENSION B(2,2,KK),C(2,KK),V(2,KK),S(2,KK),G(2,2,100),H(2,2),HI(2
     1,2),W(2,100)
      DO 7 K=1,KK
      IF(K.NE.1) GO TO 2
      DO 1 I=1,2
      DO 1 J=1,2
1     H(I,J)=B(I,J,K)
      GO TO 4
2     KM1=K-1
      DO 3 I=1,2
      DO 13 J=1,2
13    H(I,J)=B(I,J,K)-C(I,KM1)*G(I,J,KM1)
3     S(I,K)=S(I,K)-C(I,KM1)*W(I,KM1)
4     TEMP=1.0/(H(1,1)*H(2,2)-H(2,1)*H(1,2))
      HI(1,1)=H(2,2)*TEMP
      HI(2,2)=H(1,1)*TEMP
      HI(2,1)=-H(2,1)*TEMP
      HI(1,2)=-H(1,2)*TEMP
      IF(K.EQ.KK) GO TO 6
      DO 5 I=1,2
      DO 5 J=1,2
5     G(I,J,K)=HI(I,J)*C(J,K)
6     DO 7 I=1,2
7     W(I,K)=HI(I,1)*S(1,K)+HI(I,2)*S(2,K)
      V(1,KK)=W(1,KK)
      V(2,KK)=W(2,KK)
      KM1=KK-1
      DO 8 K=1,KM1
      KI=KK-K
      KP1=KI+1
      DO 8 I=1,2
8     V(I,KI)=W(I,KI)-G(I,1,KI)*V(1,KP1)-G(I,2,KI)*V(2,KP1)
      RETURN
```

END

INVE0037

```
      SUBROUTINE PQP(W,FLUX)
      IMPLICIT REAL*8 (A-H,O-Z)
     0COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DMU(6),H,II
      COMMON /INTEG/KK,NREG,KINT(10)
      COMMON /ARRAY/B(2,2,100),C(2,100),F(2,100),P(100),PREC(6,100)
      COMMON /PKP/V1(2,100),V2(2,6,100),V3(2,100),V4(2,6,100)
      COMMON /VCHII/V(2),CHII(2,6),DX(10)
      COMMON /DELTA/DB(2,2,10),DSFN(2,10)
      REAL*8 LAM(6),MU(6)
      DIMENSION W(2,KK),FLUX(2,KK),DBF(2)
      N=1
      T=0.0
      ALFA=0.0
      DALFA=0.0
      DO 8 I=1,II
      CI(I)=0.0
      MU(I)=0.0
    8 DMU(I)=0.0
      DO 3 K=1,KK
      TEMP=DX(N)
      DO 10 J=1,2
   10 DBF(J)=DB(J,1,N)*FLUX(1,K)+DB(J,2,N)*FLUX(2,K)
      DP=DSFN(1,N)*FLUX(1,K)+DSFN(2,N)*FLUX(2,K)
      IF(K.NE.KINT(N)) GO TO 4
      N=N+1
      TEMP=0.5*(DX(N)+TEMP)
      DO 9 J=1,2
    9 DBF(J)=0.5*(DBF(J)+DB(J,1,N)*FLUX(1,K)+DB(J,2,N)*FLUX(2,K))
      DP=0.5*(DP+DSFN(1,N)*FLUX(1,K)+DSFN(2,N)*FLUX(2,K))
    4 DO 3 J=1,2
      V1(J,K)=TEMP*FLUX(J,K)/V(J)
      V3(J,K)=B(J,1,K)*FLUX(1,K)+B(J,2,K)*FLUX(2,K)
      IF(K.NE.KK) V3(J,K)=V3(J,K)+C(J,K)*FLUX(J,K+1)
      IF(K.NE.1) V3(J,K)=V3(J,K)+C(J,K-1)*FLUX(J,K-1)
      T=T+W(J,K)*V1(J,K)
```

```
      ALFA=ALFA+W(J,K)*V3(J,K)
      DALFA=DALFA+W(J,K)*DEF(J)
      WDP=W(J,K)*DF
      DO 3 I=1,II
      V2(J,I,K)=CHII(J,I)*FREC(I,K)
      V4(J,I,K)=CHII(J,I)*P(K)
      CI(I)=CI(I)+W(J,K)*V2(J,I,K)
      MU(I)=MU(I)+W(J,K)*V4(J,I,K)
3     DMU(I)=DMU(I)+WDP*CHII(J,I)
      ALFA=ALFA/T
      DALFA=DALFA/T
      DO 7 I=1,II
      TEMP=BETA(I)/T
      MU(I)=MU(I)*TEMP
      DMU(I)=DMU(I)*TEMP
      ALFA=ALFA-MU(I)
7     DALFA=DALFA-DMU(I)
      RETURN
      END
```

```
      SUBROUTINE THETA(RAMP,NT1,N)
C
C     THIS SUBROUTINE IS TO BE USED WITH METHOD 1 ONLY
C
      IMPLICIT REAL*8 (A-H,O-Z)
     COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DMU(6),H,II
      DIMENSION DC(6),DC1(6)
      REAL*8 LAM(6),MU(6)
      LOGICAL RAMP
      IF(RAMP) GO TO 7
      IF(NT1.GE.0) GO TO 3
      IF(N.LT.-NT1) GO TO 8
      GO TO 7
 3    IF(N.LE.NT1) GO TO 8
      CALL POINTK(H,DN,DN1,DC,DC1)
      TETA0=DN1/DN
      DO 1 I=1,II
 1    TETA(I)=DC1(I)/DC(I)
      WRITE(6,12) TETA0,(TETA(I),I=1,II)
 12   FORMAT(' THETAS=',7F12.6)
      IF(TETA0.LT.0.1) GO TO 7
      IF(TETA0.GT.1.0) GO TO 8
      GO TO 10
 7    TETA0=0.5
      GO TO 9
 8    TETA0=1.0
 9    DO 11 I=1,II
 11   TETA(I)=0.5
 10   DO 13 I=1,II
      TEMP=H*LAM(I)
      TAU(I)=1.0/(1.0+TEMP*TETA(I))
      TETA(I)=TETA0*TAU(I)
 13   EHL(I)=1.0-TEMP*TAU(I)
      RETURN
      END
```

```
      SUBROUTINE THETA(RAMP,NT1,N)
C
C     THIS SUBROUTINE IS TO BE USED WITH METHOD 2 ONLY
C
      IMPLICIT REAL*8 (A-H,O-Z)
     COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DMU(6),H,II
      DIMENSION CI1(6),EHL1(6)
      REAL*8 LAM(6),MU(6)
      LOGICAL RAMP
      IF(RAMP) GO TO 7
      IF(NT1.GE.0) GO TO 3
      IF(N.LT.-NT1) GO TO 8
      GO TO 7
 3    IF(N.LE.NT1) GO TO 8
      T0=T
      DO 2 I=1,II
 2    TETA(I)=EHL(I)*CI(I)
      CALL POINTK(H,DN,DN1)
      TETA0=DN1/DN
      DO 1 I=1,II
 1    TETA(I)=((CI(I)-TETA(I))/(H*MU(I))-TAU(I)*T0)/DN
      WRITE(6,12) TETA0,(TETA(I),I=1,II)
 12   FORMAT(' THETAS=',7F12.6)
      IF(TETA0.LT.0.1) GO TO 7
      IF(TETA0.GT.1.0) GO TO 8
      RETURN
 7    TETA0=0.5
      DO 9 I=1,II
 9    TETA(I)=(1.0-TAU(I))/(H*LAM(I))
      RETURN
 8    TETA0=1.0
      DO 11 I=1,II
 11   TETA(I)=TAU(I)
      RETURN
      END
```

```
      SUBROUTINE POINTK(H1,DN,DN1,DC,DC1)
C
C     THIS SUBROUTINE IS TO BE USED WITH METHOD 1 ONLY
C
      IMPLICIT REAL*8 (A-H,O-Z)
     0COMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DMU(6),H,II
      DIMENSION HWS(2),IND(2),W(2),WS(2),A(6),D(6),DC(6),DC1(6)
      REAL*8 LAM(6),MU(6)
      HWS(1)=0.05
      HWS(2)=DMIN1(-2.D0,-1.2D0*H1*LAM(II))
      R=ALFA
      DO 10 I=1,II
   10 R=R+MU(I)
      DO 2 N=1,2
      TEMP=1.0
      WS(N)=HWS(N)/H1
      DO 1 I=1,II
    1 TEMP=TEMP+MU(I)/(WS(N)+LAM(I))
      TEMP=WS(N)*TEMP
      IND(N)=0
    2 IF(TEMP.LT.R.AND.N.EQ.1.OR.TEMP.GT.R.AND.N.EQ.2) IND(N)=1
      ETA=2.0/H1
      TEMP=T
      G=1.0-R/ETA
      DO 3 I=1,II
      TEMP1=ETA+LAM(I)
      A(I)=MU(I)/TEMP1
      BI=LAM(I)/TEMP1
      D(I)=ETA/TEMP1
      G=G+A(I)
    3 TEMP=TEMP+BI*CI(I)
      G=TEMP/G
      TEMP2=G
      TEMP1=ALFA*TEMP2
      DO 4 I=1,II
```

```
      D(I)=G*A(I)+D(I)*CI(I)                                            POIN0037
      TEMP=LAM(I)*D(I)                                                  POIN0038
      A(I)=MU(I)*TEMP2-TEMP                                             POIN0039
4     TEMP1=TEMP1+TEMP                                                  POIN0040
      DN=H1*TEMP1                                                       POIN0041
      DN1=0.5*DN                                                        POIN0042
      TEMP2=TEMP2+DN1                                                   POIN0043
      DO 14 I=1,II                                                      POIN0044
      DC(I)=H1*A(I)                                                     POIN0045
      DC1(I)=0.5*DC(I)                                                  POIN0046
14    D(I)=D(I)+DC1(I)                                                  POIN0047
      DO 5 N=1,2                                                        POIN0048
      IF(IND(N).EQ.0) GO TO 5                                           POIN0049
      TEMP=T                                                            POIN0050
      G=1.0                                                             POIN0051
      CALL ROOT(W(N),WS(N))                                             POIN0052
      DO 6 I=1,II                                                       POIN0053
      TEMP1=LAM(I)+W(N)                                                 POIN0054
      A(I)=MU(I)/TEMP1                                                  POIN0055
      BI=LAM(I)/TEMP1                                                   POIN0056
      TEMP=TEMP+BI*CI(I)                                                POIN0057
6     G=G+A(I)*BI                                                       POIN0058
      HW=H1*W(N)                                                        POIN0059
      EXPON=0.0                                                         POIN0060
      IF(HW.GT.-20.0) EXPON=DEXP(HW)                                    POIN0061
      G=TEMP*(EXPON-(2.0+HW)/(2.0-HW))/G                                POIN0062
      TEMP2=TEMP2+G                                                     POIN0063
      DN=DN+G                                                           POIN0064
      DN1=DN1+G/HW                                                      POIN0065
      DO 7 I=1,II                                                       POIN0066
      TEMP=G*A(I)                                                       POIN0067
      D(I)=D(I)+TEMP                                                    POIN0068
      DC(I)=DC(I)+TEMP                                                  POIN0069
7     DC1(I)=DC1(I)+TEMP/HW                                             POIN0070
5     CONTINUE                                                          POIN0071
      T=TEMP2                                                           POIN0072
```

```
      DO 8 I=1,II                                                       POIN0073
8     CI(I)=D(I)                                                        POIN0074
      RETURN                                                            POIN0075
      END                                                               POIN0076
```

```
      SUBROUTINE POINTK(H1,DN,DN1)
C
C     THIS SUBROUTINE IS TO BE USED WITH METHOD 2 ONLY
C
      IMPLICIT REAL*8 (A-H,O-Z)
     0COMMON /PTKIN/MU,LAM,EETA(6),ALFA,CI(6),T,TETAO,TETA(6),TAU(6),EPL
     1(6),DALFA,DML(6),H,II
      DIMENSION HWS(2),IND(2),W(2),WS(2),A(6),C(6)
      REAL*8 LAM(6),MU(6)
      HWS(1)=0.05
      HWS(2)=DMIN1(-2.D0,-1.2D0*H1*LAM(II))
      R=ALFA
      DO 10 I=1,II
   10 R=R+MU(I)
      DO 2 N=1,2
      TEMP=1.0
      WS(N)=HWS(N)/H1
      DO 1 I=1,II
    1 TEMP=TEMP+MU(I)/(WS(N)+LAM(I))
      TEMP=WS(N)*TEMP
      IND(N)=0
    2 IF(TEMP.LT.R.AND.N.EQ.1.OR.TEMP.GT.R.AND.N.EQ.2) IND(N)=1
      ETA=2.0/H1
      TEMP=T
      G=1.0-R/ETA
      DO 3 I=1,II
      TEMP1=ETA+LAM(I)
      A(I)=MU(I)/TEMP1
      BI=LAM(I)/TEMP1
      C(I)=ETA/TEMP1
      G=G+A(I)
    3 TEMP=TEMP+BI*CI(I)
      G=TEMP/G
      TEMP2=G
      TEMP1=ALFA*TEMP2
      DO 4 I=1,II
```

```
      D(I)=G*A(I)+D(I)*CI(I)                                            POIN0037
      TEMP=LAM(I)*D(I)                                                  POIN0038
      A(I)=MU(I)*TEMP2-TEMP                                             POIN0039
    4 TEMP1=TEMP1+TEMP                                                  POIN0040
      DN=H1*TEMP1                                                       POIN0041
      DN1=0.5*DN                                                        POIN0042
      HH=0.5*H1                                                         POIN0043
      TEMP2=TEMP2+HH*TEMP1                                              POIN0044
      DO 14 I=1,II                                                      POIN0045
   14 D(I)=D(I)+HH*A(I)                                                 POIN0046
      DO 5 N=1,2                                                        POIN0047
      IF(IND(N).EQ.0) GO TO 5                                           POIN0048
      TEMP=T                                                            POIN0049
      G=1.0                                                             POIN0050
      CALL ROOT(W(N),WS(N))                                             POIN0051
      DO 6 I=1,II                                                       POIN0052
      TEMP1=LAM(I)+W(N)                                                 POIN0053
      A(I)=MU(I)/TEMP1                                                  POIN0054
      BI=LAM(I)/TEMP1                                                   POIN0055
      TEMP=TEMP+BI*CI(I)                                                POIN0056
    6 G=G+A(I)*BI                                                       POIN0057
      HW=H1*W(N)                                                        POIN0058
      EXPON=0.0                                                         POIN0059
      IF(HW.GT.-20.0) EXPON=DEXP(HW)                                    POIN0060
      G=TEMP*(EXPON-(2.0+HW)/(2.0-HW))/G                                POIN0061
      TEMP2=TEMP2+G                                                     POIN0062
      DN=DN+G                                                           POIN0063
      DN1=DN1+G/HW                                                      POIN0064
      DO 7 I=1,II                                                       POIN0065
    7 D(I)=D(I)+G*A(I)                                                  POIN0066
    5 CONTINUE                                                          POIN0067
      T=TEMP2                                                           POIN0068
      DO 8 I=1,II                                                       POIN0069
    8 CI(I)=D(I)                                                        POIN0070
      RETURN                                                            POIN0071
      END                                                               POIN0072
```

```
      SUBROUTINE ROOT(W,WS)
C
C     FOR USE IN SPACE-DEPENDENT KINETICS PROGRAM
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 LAM(6),MU(6)
     OCOMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETA0,TETA(6),TAU(6),EHL
     1(6),DALFA,DML(6),H,II
      IF(WS.GT.0.0) W=DMAX1(ALFA,WS)
      IF(WS.LT.0.0) W=DMIN1(ALFA,WS)
    2 W0=W
      SUM1=ALFA
      SUM2=1.0
      DO 1 I=1,II
      TEMP1=W+LAM(I)
      TEMP=LAM(I)*MU(I)/TEMP1**2
      SUM1=SUM1+(TEMP1+W)*TEMP
    1 SUM2=SUM2+TEMP
      W=SUM1/SUM2
      IF((W-W0)/W.GT.1.D-5) GO TO 2
      RETURN
      END
```

```
      SUBROUTINE SOURCE
      IMPLICIT REAL*8 (A-H,O-Z)
     CCOMMON /PTKIN/MU,LAM,BETA(6),ALFA,CI(6),T,TETAO,TETA(6),TAU(6),EHL
     1(6),CALFA,DML(6),H,II
      COMMON /INTEG/KK,NREG,KINT(10)
      COMMON /PKP/V1(2,100),V2(2,6,100),V3(2,100),V4(2,6,100)
      DIMENSION S(2,100),COEF2(6),COEF4(6)
      EQUIVALENCE (S(1),V1(1))
      REAL*8 LAM(6),MU(6)
      TETAI=1.0/TETAO
      COEF1=TETAI/H
      COEF3=TETAI-1.0
      DO 2 I=1,II
      COEF2(I)=TETAI*LAM(I)*TAU(I)
    2 COEF4(I)=TETAI*BETA(I)*(TETA(I)-TAU(I))
      DO 1 K=1,KK
      DO 1 J=1,2
      S2=0.0
      S4=0.0
      DO 3 I=1,II
      S2=S2+COEF2(I)*V2(J,I,K)
    3 S4=S4+COEF4(I)*V4(J,I,K)
    1 S(J,K)=-COEF1*V1(J,K)-S2-COEF3*V3(J,K)-S4
      RETURN
      END
```