Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

Alternative Methods in Criticality

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

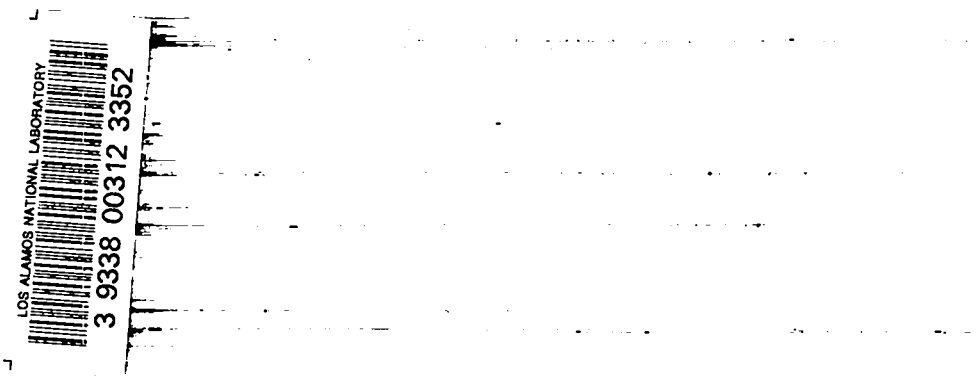
This thesis was accepted by the University of Illinois at Urbana-Champaign in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Nuclear Engineering. It is the independent work of the author and has not been edited by the Technical Information staff.

LA-9181-T
Thesis

UC-46
Issued: January 1982

Alternative Methods in Criticality

John Michael Pedicini



Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

CONTENTS

ALTERNATIVE METHODS IN CRITICALITY

by

John Michael Pedicini, Ph.D.
Nuclear Engineering Program
University of Illinois at Urbana-Champaign, 1981

ABSTRACT

In this thesis two new methods of calculating the criticality of a nuclear system are introduced and verified.

Most methods of determining the criticality of a nuclear system depend implicitly upon knowledge of the angular flux, net currents, or moments of the angular flux, on the system surface in order to know the leakage. For small systems, leakage is the predominant element in criticality calculations. Unfortunately, in these methods the least accurate fluxes, currents, or moments are those occuring near system surfaces or interfaces. This is due to a mathematical inability to satisfy rigorously with a finite order angular polynomial expansion or angular difference technique the physical boundary conditions which occur on these surfaces. Consequently, one must accept large computational effort or less precise criticality calculations.

The methods introduced in this thesis, including a direct leakage operator and an indirect multiple scattering leakage operator, obviate the need to know angular fluxes accurately at system boundaries. Instead, the system wide scalar flux, an integral quantity which is substantially easier to obtain with good precision, is sufficient to obtain production, absorption, scattering, and leakage rates.

I. INTRODUCTION

A. Opening Remarks

From the beginning of Reactor Physics as a science, an important question has been that of criticality. Any system with a neutron multiplicity (defined as the average number of neutrons emitted per neutron interaction), c, greater than unity can be made to go critical if the dimensions are appropriate. The determination of this critical size is the primary problem discussed in this thesis.

Methods for the calculation of critical size can be divided into two classes of techniques. The first class will be referred to as Mathematical Criticality (MC), in that the problems solvable by these techniques are highly idealized and physically unreal. The second class will be referred to as Physical Criticality (PC), since the techniques used, although benchmarked against MC techniques in MC solvable problems, are applicable to a wide range of physically real and interesting problems.

Mathematical Criticality applies to those cases in which the system to be studied is so simplified that an analytic and mathematically complete formulation is possible. These simple models are then treated as eigenvalue problems for some system parameter, such as the critical dimension or neutron multiplicity. Such eigenvalues are determined exactly or they are approximated with a high degree of precision. These values are most useful as benchmarks to test PC techniques applied to the simple systems.

Physical Criticality techniques implicitly equate the neutron production rate to the sum of the neutron leakage and destruction rates. This balance condition is often masked through the use of a source iteration technique. In typical problems there is little imprecision in the values of the production and destruction rates, since these depend on the system wide scalar flux, an integral quantity that is fairly accurate despite boundary perturbations. Unfortunately, for these techniques, the leakage rates depend upon the angular fluxes, net currents, or moments of the angular flux, on the system boundaries. These are the least accurate fluxes, currents, or moments in the system due to a mathematical inability to satisfy rigorously, with a finite order angular polynomial expansion or angular difference technique, the physical conditions which apply at the system boundaries. Consequently, approximations which show adequate precision in the system-wide parameters are inadequate to the task of accurately calculating leakage. Approximations of high enough order to yield accurate leakages are difficult to solve and are more complex than are necessary to calculate properly production and destruction rates.

Reduction of the calculational effort necessary for an accurate leakage calculation to the level which is adequate for the determination of the scalar flux is the central motivation of this thesis. To achieve this objective two new techniques are introduced, namely, the direct leakage operator and the indirect multiple scattering leakage operator. These techniques obviate the need to know the angular fluxes accurately at the system boundaries, since the operators require only the system-wide scalar flux for initiation, and are intended to provide

completely independent leakage calculations. Because of their formulation they should be easily adaptable to existing neutron transport computations. The direct leakage operator differs from the indirect leakage operator in that it concerns itself only with neutrons which leak from the point of origin with no interactions within the system. Thus, the direct leakage operator requires that all collisions be treated as neutron deaths and that any collisions from which neutrons emerge be treated as neutron births. Hence, all transport of neutrons within the system is handled by the scalar flux calculation. On the other hand, the indirect leakage operator handles internally all transport of the neutrons which eventually leak out of the system and uses the scalar flux calculation only to obtain a fission source distribution. For the indirect leakage operator the standard definitions of birth in fission and death in absorption apply.

For the purposes of this thesis results obtained with these two new operator techniques are tested against only those problems solvable by MC techniques. The reasons for this are twofold. One reason is that the most precise criticality calculations available are those for MC solvable problems. The other reason is that highly accurate analytic scalar fluxes are available for these problems, which enables the precision of the operator techniques themselves to be determined. In theory the operator techniques are not limited in any respect for any calculation for which an initiating scalar flux is available. However, the direct leakage operator technique is presently not developed enough to handle anisotropic neutron sources. The state of the indirect leakage operator is somewhat less advanced, due to the internal

transport, and is limited to steady state, monoenergetic neutrons, isotropic scatterings, and isotropic fission. In this thesis, the direct leakage operator has proved fully as accurate as the available scalar fluxes, while the indirect leakage operator has proved only slightly less accurate.

A further capability obtained from the extra computational effort involved in the indirect leakage operator is the ability to calculate the spatial leakage probability distribution for neutrons born in fission. This type of calculation, unique to this thesis, is carried out in such detail that the probability of a neutron born in fission at a specified location scattering n times and then leaking out of the assembly can be determined.

B. Overview of Mathematical Criticality Calculations

Not surprisingly, most of the MC techniques available are applied to the integral form of the Boltzmann transport equation. This is probably related to the fact that eigenvalue theory is well developed for differential equations and also for integral equations but is not readily available for integro-differential equations, such as the Boltzmann transport equation. The Boltzmann equation can be formulated in a purely integral form, but it cannot be formulated in a purely differential form due to discontinuous changes in energy and angle required by the physics[1]. Typically, the assumptions added to the integral transport equation for an MC calculation are steady state, monoenergetic neutrons, spatially constant cross sections, isotropic fission, isotropic scattering, and no external sources.

Variational theory as applied to the transport equation has long been used as an MC technique[1,2,3,4]. The detailed application of this technique will appear in Chapter Two, where it is used to obtain the scalar fluxes needed to initiate the leakage operators. The technique involves obtaining a variational principle whose necessary condition for an extremal is the integral transport equation. The functional of the variational principle can be interpreted as an eigenvalue. Polynomials comprised of the first n terms of a series expansion of the asymptotic solution to the transport equation, with each term muliplied by an independently variable parameter, are inserted into the functional as trial functions, and the functional is then extremized with respect to the parameters. Once the parameters are found, they can be reinserted into the functional in order to obtain an estimate of the eigenvalue. The abbreviation for this approach is V_n, where n is the order of the polynomial used as a trial function. Previously, V_2 slab, V_2 spherical, V_2 cylindrical, and V_4 spherical results have been reported in the literature[1,4]. This technique is extended considerably in this thesis and now includes V_4 slab and V_4 cylindrical results. In addition, the author has standardized the derivation and solution algorithms associated with the V_n technique.

The Extrapolated Endpoint Method (EEM), or simply the endpoint method, is as old as the previously discussed variational technique[1,3,4,5,6,7,8]. It arises from approximations made in order to utilize the Wiener-Hopf Technique, which is customarily applied to integral equations in a semi-infinite planar geometry (single boundary system), on the integral transport equation in a finite planar geometry

(two-boundary system). With more than one boundary, no exact solution is known, but an approximation can be made by treating the behavior of the solution near each boundary as if no other boundary existed. It is assumed that the solution of a two-boundary problem in which the boundaries are very far apart will behave in the central region like the solution in a semi-infinite system[5]. With this assumption, the solution of a two-boundary problem is constructed as a combination of two one-boundary solutions such that the asymptotic components coincide[5]. This sort of approximation is, of course, more accurate for larger systems than for smaller systems[4]. The point beyond the system boundary where the asymptotic flux goes to zero is called the extrapolated endpoint, which is the origin of the name of the method. These endpoint distances are also often used in diffusion theory calculations as the extrapolation distances. This distance is tabulated as a function of the neutron multiplicity, c. To determine the critical radius requires the use of two criticality conditions obtained from the integral transport equation in an infinite system. These are conditions that must be satisfied in order to have a nontrivial solution. One condition is found by Fourier transforming the integral transport equation. The other condition is found by expanding the flux, contained in the integrand of the integral equation, in a Taylor's series about the variable of integration and then integrating to obtain an infinite moments series. These two conditions are identified as equivalent. The solution of the Taylor's expansion form is assumed to be in the form of a solution of the scalar Helmholtz equation. The solution to the Fourier transform condition is actually a qualifier, as a function of c,

upon the admissable values of the Fourier transform variables. These admissable Fourier transform variables are identified with the constants that appear in the scalar Helmholtz equation, which is subsequently solved. The solution function is equated to zero, the first root is found, and the value of the position where this occurs is calculated[8]. This distance is assumed to be the sum of the critical radius and previously tabulated extrapolation distance. The spherical integral transport equation is reducible to the slab form and presents no problems. The cylindrical case is handled by analogy and assumption rather than by the approach specified above mainly because cylindrical geometry endpoints have not been obtained. In practice, the critical radii calculated have less than one percent error for boundary separations greater than three tenths of a mean free path[5], and are considered exact for boundary separations greater than six mean free paths[4]. Due to its complexity, this technique is rarely used.

A somewhat newer approach is the method of singular eigenfunction expansions, also known as Case's Method[6,9,10]. This approach is based on the fact that, under certain conditions, the integro-differential transport equation is separable. The previous assumptions associated with MC, namely steady state, monoenergetic neutrons, spatially constant cross sections, isotropic fission, isotropic scattering, and no external sources, are also applied in this method. Using essentially the standard separation of variables technique, the angular flux is separated into a series of spatially dependent exponentials multiplied by angle-dependent coefficients. It is shown that there are two discrete eigenvalues outside the angular domain of integration and a

continuous spectrum inside. Inserting the separated form into the original equation and applying normalization conditions yield a transcendental equation for the two discrete eigenvalues. By using the boundary conditions of the problem, conditions on the eigenfunctions are found and reduced to a single inhomogeneous Fredholm integral equation of the second kind for the continuous eigenvalue spectrum. Therefore, the angular flux is expanded in a pair of terms plus an integral of the continuum eigenfunctions. Solving the integral equation for the continuum eigenvalues is difficult, but Kaper, Leaf, and Lindeman[10] claim to have done it numerically to great accuracy and, thereby, claim to have calculated the benchmark critical dimensions for the slab and sphere.

Two other techniques are mentioned mainly because there are no accurate independent data to compare to in cylindrical geometry. These are the integral transform method of Hembd[11] and the matrix eigenvalue method of Milgram[12]. These have been shown to be essentially equivalent[13], and, therefore, only the integral transform method will be discussed. The integral transport equation limited by the assumptions discussed earlier is Fourier transformed, and the resulting integral equation for the transformed flux has its kernel replaced by a bilinear expansion. The bilinear expansion is determined to be a representation of the kernel in terms of its real and complex conjugate eigenfunctions. An infinite system of linear algebraic equations for the coefficients which appear in the transformed equation with expanded kernel is obtained. Once these coefficients are known, the transformed equation with expanded kernel can be inverted to yield the scalar flux. The

linear algebraic system is shown to represent an eigenvalue problem for the fundamental and higher decay constants of a neutron pulse decaying in the geometry of interest. The fundamental decay problem is stated to be mathematically equivalent to the critical problem.

C. Overview of Physical Criticality Calculations

Physical Criticality techniques are mostly applied to the integro-differential form of the transport equation. There are exceptions to this, as well as hard to classify techniques, such as the method of Dahl and Sjöstrand[14] and the Serber-Wilson technique[3,7,15,16,17]. Most PC techniques are based on an iterative solution called the source iteration technique. In source iteration, a comparison is made between the total fission source (integral over all space, energies, and angles) of the Kth and (K-1)th iterates. As the number of iterations increases, the ratio approaches K_{eff} if each iterate was appropriately normalized. The method iterates, after K_{eff} is converged, on some system parameter, such as the radius, to drive K_{eff} to the desired value (usually one). Implicit in this technique, for any self-consistent formulation, is the fact that the neutron fission rate is equal to the destruction rate plus the leakage rate. The value of the fission and destruction rates are dependent upon the system-wide scalar flux, but the leakage rate is dependent, in the source iteration formalism, upon the surface angular fluxes, net currents, or angular moments. This is due to the inherent neutron balance condition present in the starting equation. As discussed

earlier, the system-wide scalar flux is much more accurate than the surface angular fluxes, net currents, or angular moments.

The technique of Dahl and Sjöstrand is included in this section mainly because of their successful treatment of anisotropic scattering[14], a problem usually not considered by MC methods. For the purpose of this thesis, the only interest is in their isotropic treatment. Starting with the integral transport equation, they expand the spatial distribution of the scalar flux in a series of Legendre polynomials. The series is truncated at nine terms and inserted into the integral transport equation to yield a standard matrix eigenvalue problem which is solved for the eigenvalues. It should be noted that they apply their expansion only to the scalar flux in an equation that already contains the boundary conditions. Therefore, the previously discussed limitations of PC should not apply, and the error present should be attributable only to truncation and numerics.

Some of the earliest, accurate criticality calculations were done by diffusion theory rather than transport theory. The most important difference between the two theories is that the transport equation, through the use of a streaming operator acting upon the angular flux, handles collisionless transport exactly, while the diffusion equation handles leakage from a differential volume element by Fick's law of diffusion, which yields a Laplacian acting upon the scalar flux. The diffusion equation is far more tractable and, therefore, better known and more often used. The diffusion equation is inapplicable to small systems due to the fact that the desired solution is selected from the class of all solutions by boundary conditions applied just where the

Fick's law assumption fails, namely at the boundaries[3]. The Serber-Wilson Method (SWM)[15,16,17] is an ingenious attempt to improve this situation. In the SWM, the general solution of the diffusion equation is substituted into the integral transport equation and required to satisfy exactly the integral transport equation at one point. The SWM has been applied only to spheres and the point selected is usually the center. By using the physical constants present in the diffusion solution and in the integral transport equation, an equation can be obtained for the critical radius of a sphere. The SWM represents the best available criticality technique for one-group diffusion theory fluxes. The SWM is mentioned not because of its accuracy (diffusion theory compares poorly with transport theory in any case) but because a diffusion theory scalar flux is used as one of the test cases in this thesis.

Expansion of an unknown function in a complete set of basis functions followed by appropriate truncation has always been a popular technique in mathematical physics. The earliest example of this in neutron transport theory is the P_n method, so called because the spherical harmonics basis functions frequently reduce to Legendre polynomials[18,19]. Typically, the angular flux is expanded in an infinite set of spherical harmonics in angle with spatially dependent expansion coefficients. The expansion is inserted into the integro-differential transport equation and the integrations are carried out. After use of suitable recurrence relations, the result is multiplied by the complex conjugate of each of the basis functions and integrated. The series is truncated at n terms and the spatial

derivative of the expansion coefficient of the (n+1) term is set to zero. The result is a finite set of linear differential equations for the expansion coefficients which can then be finite differenced and source iterated[1]. Several choices are available to approximate vacuum boundary conditions. The Marshak boundary conditions require that the integral over angle of the angular flux with each odd term of the expansion vanish on the boundary[1]. The Mark boundary conditions require the angular flux to go to zero on the boundary for a select set of angular points. The points chosen are usually the positive zeros of the (n+1)st term of the expansion[1]. Although the Marshak conditions are superior, neither is very accurate for low orders of approximation[4].

Syros and Theocharopoulos[20] claim to have eliminated the chief defect of PC techniques, namely that boundary conditions applied to PC methods don't match the physical boundary conditions present except at large orders of approximation. Syros and Theocharopoulos[21] were quick to point this out in their recent criticism of Dahl and Sjöstrand. Basically, they expand the integro-differential form of the neutron transport equation in a set of polynomials which can be made to satisfy exactly either homogeneous or inhomogeneous boundary conditions. The sets of polynomials are constructed for each problem in a manner too complex to detail here. The interested reader will find appropriate citations in Ref. 20 and Ref. 21. The eigenvalues of Syros and Theocharopoulos are consistently lower then the eigenvalues obtained by other methods.

Finite Element techniques, popular research items a few years ago, have mostly been abandoned. There is one approach of interest to this thesis, that of Ackroyd and associates[22,23,24,25]. The purely finite element formulation with which they started[22] has evolved into a spherical harmonics angular expansion coupled with a spatial finite element method to approximate the angular flux[24]. The equation solved is the second-order self-adjoint form of the transport equation. Unfortunately, the only eigenvalue result[25] obtained so far is of insufficient accuracy to warrant consideration in the eigenvalue controversy between Dahl and Sjöstrand and Syros and Theocharopoulos.

Finite differencing of the integro-differential form of the Boltzmann transport equation has resulted in the S_n technique, the most popular and widely accepted neutron transport technique extant[1,4,19,26,27]. As currently implemented, S_n techniques expand the inhomogeneous sources, the fission kernel, and the scattering kernel in a truncated series of spherical harmonics with known, usually physically based, coefficients. These expansions enable the collection of all "source" terms into a single term. The resulting equation is then: streaming operator plus absorption operator operating upon angular flux equals an angular source term, which is assumed known from a previous flux iteration. A set of quadrature points and weights is determined and the angular cell-centered fluxes are assumed to be represented by the angular fluxes at these points. These points and weights are then used to discretize the streaming and absorption operators. In curved geometries, "alpha" coefficients are introduced to force neutron conservation which was destroyed by the angular differencing of the

streaming operator. The boundary conditions used are the obvious ones. The resulting set of partial differential equations can then be finite differenced. Relations between phase space cell-edge and cell-centered fluxes are needed and are usually handled by an artifice known as the diamond difference relation. This relation states simply that the arithmatic average of the cell surface fluxes is the same as the average flux over the entire cell. Eigenvalue calculations are handled by source iteration.

D. Analytic Similarities

Somewhat similar to the direct leakage operator that appears in Chapter Three is the escape probability treatment beginning with Case, de Hoffman, and Placzek[1,6,28]. The basic problem addressed by the escape probability formalism is to determine the average probability that neutrons born uniformly in a purely absorbing body will escape. Several of the distinctions between the direct leakage operator and the escape probability formalism will be discussed. Escape probability methodology deals only with purely absorbing bodies, while the direct leakage operator is applicable to bodies in which both scattering and absorption occur. Escape probability analysis is capable of handling only spatially constant neutron sources. Neutron sources in multiplying assemblies are spatially varying and linearly proportional to the flux. It is this physically occuring spatially dependent source that is used by the direct leakage operator. Although such restrictions appear not to hamper adequate results for fast fission and resonance escape calculations in large thermal lattices, they do eliminate a large amount

of the physics and would be useless for criticality calculations. The detailed derivation of the escape probability method does not contain within it the development of the pointwise, or spatially distributed, escape probability except, incidentally, in the slab case. Without the pointwise escape probability, the direct leakage operator cannot be formed and, although the escape probability is equal to the direct leakage operator operating upon a uniform source in a purely absorbing medium, the escape probability cannot be considered equivalent to the direct leakage operator.

The derivation of the escape probabilty[28] evades the tricky integrations involved in the direct leakage approach by use of the chord functions introduced by Dirac[29]. These chord functions enabled Case, de Hoffman, and Placzek to obtain exact numerical results for some simple geometries. However, they have not proved very tractable in practical usage. This has instigated a number of approximations for the escape probability from a number of authors beginning with the Wigner rational approximation[30,31,32,33,34,35]. It should be mentioned that the starting point for both the kernel of the direct leakage operator and for the kernel of the escape probability is the same, namely, an integral over angle of the surface penetration probability of a neutron emitted at a specified point. The assumptions and method of solution used for the escape probability formalism serve only to restrict severely its potential application and accuracy. The direct leakage operator is subject to none of the limitations of the escape probability formalism and is, therefore, applicable to a much wider class of

problems. These include criticality and those problems now handled by escape probability.

E. Summary of Thesis

The organization of this thesis entails four working chapters, one introductory chapter, and nine appendices. The first working chapter, Chapter Two, contains detailed V_n calculations, which are used both to obtain critical size estimates for comparison purposes and for the determination of scalar flux distributions. The polynomial type distributions obtained from the V_n calculations are highly accurate and convenient for initiating the direct and indirect leakage operators. For use in this thesis the V_n formalism is standardized, in derivation and solution algorithm, and extended to include the V_4 cylindrical and V_4 slab results. Chapter Two also contains a discussion of the recent eigenvalue controversy between Dahl and Sjöstrand[36] and Syros and Theocharopoulos[21].

Chapter Three introduces the direct leakage operator in analytic form for the three standard one-dimensional geometries. V_2 and V_4 scalar fluxes are used, and the direct leakage operators are proved exact within the limitations imposed by such fluxes. Surprisingly accurate critical size estimates are obtained when the direct leakage operator is used in conjunction with a diffusion theory scalar flux.

The indirect leakage operator is introduced in Chapter Four. The operator integrand is derived as the solution function of an inhomogeneous singular Fredholm integral equation of the second kind. The kernel of that integral equation is shown to be identical to the

kernel of the appropriate integral transport equation. Due to the inhomogeneity, solution of the indirect leakage operator integral equation is substantially easier than that of the transport integral equation. Detailed calculations of the spatial leakage probability distribution are performed both for the total leakage probability and for the discrete scattering order leakage probability.

Chapter Five contains conclusions and recommendations for future work. Appendices A and B contain mathematical data on exponential integral functions and modified Bessel functions. Appendix B also adapts a form of the Graf's addition formula for modified Bessel functions, which permits some difficult integrations in cylindrical geometry to be performed. Appendix C is devoted to the derivation of the cylindrical geometry integral transport equation which does not appear in the literature. Tedious manipulations necessary for the V_n calculations are included in Appendices D, E, F, and G. Extrapolation distances appear in Appendix H. Appendix I contains code listings along with appropriate comments.

Note that in this thesis the total cross section, σ_t, has usually been normalized to unity except where noted. This conveniently allows suppression of the σ_t and implies distance measured in units of the total mean free path.

II. VARIATIONAL TECHNIQUES

A. Integral Transport Equations

The starting point for the V_n method is the integral transport equation appropriate to each coordinate system. The V_n method is used not only to obtain critical radii for comparison purposes but also to obtain the scalar fluxes utilized to initiate the direct and indirect leakage calculations. The derivation of the equations in the forms used in this thesis are available in the literature[1,6,19] for slab and spherical geometries. A derivation of the cylindrical integral transport equation in a useful form is not readily available in the literature. Weinberg and Wigner[7] mention that the sum of a diffusion kernel and a first collision kernel is the approximate kernel of the integral transport equation. However, the first collision kernel given in cylindrical geometry is the exact kernel for the cylindrical integral transport equation, as shall be proven. Greenspan, Kelber, and Okrent[19] give a very detailed derivation for the kernels of the integral transport equation in slab and spherical geometry, but don't even mention cylindrical geometry other than to tabulate a permutation of the Weinberg Wigner first collision kernel. Neither Ref. 7 nor Ref. 19 give a derivation for the cylindrical geometry results shown. Accordingly, a rigorous derivation is given in Appendix C for cylindrical geometry. The integral transport equations to be given are subject to the following limitations: steady state, monoenergetic neutrons, isotropic fission, isotropic scattering, no external sources, and the total macroscopic cross section normalized to unity.

The integral transport equation for the scalar flux in a one-dimensional slab of half-thickness, b, infinite in the y and z coordinates, is given by Bell and Glasstone[1] as

$$\Phi(x) = \frac{c}{2}\int_{-b}^{b} dx'\Phi(x')E_1(|x-x'|). \tag{2.1}$$

The E_n functions are the exponential integral functions discussed in Appendix A.

For a one-dimensional sphere of radius, b, the integral transport equation for the scalar flux is[1]

$$r\Phi(r) = \frac{c}{2}\int_{-b}^{b} dr'r'\Phi(r')E_1(|r-r'|), \tag{2.2}$$

subject to r in the interval [0,b], r′ in the interval [-b,b], and $\Phi(-r')=\Phi(r')$.

In one-dimensional cylindrical geometry of infinite axial extent and radius, b, the integral transport equation for the scalar flux is given by Appendix C as

$$\Phi(r) = c\int_{0}^{b} dr'r'\Phi(r')K(r,r'); \tag{2.3a}$$

$$K(r,r') = \int_{1}^{\infty} dyK_0(yr)I_0(yr') \text{ for } r'<r, \tag{2.3b}$$

$$K(r,r') = \int_{1}^{\infty} dyK_0(yr')I_0(yr) \text{ for } r'>r. \tag{2.3c}$$

B. Application of Variational Techniques

Variational techniques as applied to the transport equation seek to estimate the lowest neutron multiplicity eigenvalue for an otherwise fixed system[1,2,3,4]. The lowest eigenvalue is sought, since this corresponds to the non-negative, or physically acceptable, eigenfunction[3]. If the one-dimensional neutron transport equation as represented by (2.1), (2.2), and (2.3a) is generalized to

$$f(r) = \gamma \int_{a_1}^{a_2} dr' K(r,r') f(r'), \tag{2.4}$$

subject to the definitions in Table I, then only one derivation need be done instead of one for each geometry.

Rewrite (2.4) in terms of an eigenvalue equation with eigenvalues γ_i corresponding to eigenfunctions $f_i(r)$:

$$f_i(r) = \gamma_i \int_{a_1}^{a_2} dr' K(r,r') f_i(r'). \tag{2.5}$$

Since this is a linear integral equation with a real, symmetric, continuous, and nondegenerate, kernel it is known that there exists an infinite number of real positive eigenvalues with orthogonal eigenfunctions[37]. Assume that these eigenfunctions are normalized such that

$$\int_{a_1}^{a_2} dr f_i(r) f_j(r) = \delta_{ij} \tag{2.6}$$

and order the eigenvalues $\gamma_0 < \gamma_1 < \gamma_2 < \ldots < \gamma_\infty$. Now, since any well-behaved trial function may be expanded as a series of eigenfunctions, let

Table I

Variable Identification for the

Generalized Variational Development

Generalized Variable	Slab Variable	Sphere Variable	Cylinder Variable
$f(r)$	$\Phi(x)$	$r\Phi(r)$	$\Phi(r)$
$f_i(r)$	$\Phi_i(x)$	$r\Phi_i(r)$	$\Phi_i(r)$
γ	$\frac{c}{2}$	$\frac{c}{2}$	c
γ_i	$\frac{c_i}{2}$	$\frac{c_i}{2}$	c_i
a_1	$-b$	$-b$	0
a_2	b	b	b
dr	dx	dr	rdr
$K(r,r')$	$E_1(\lvert x-x'\rvert)$	$E_1(\lvert r-r'\rvert)$	$\int_1^\infty dy K_0(yr)I_0(yr'),\ r'<r$ $\int_1^\infty dy K_0(yr')I_0(yr),\ r'>r$

$$f(r) = \sum_{i=0}^{\infty} M_i f_i(r). \tag{2.7}$$

Insert (2.7) into the variational principle (2.8),

$$\gamma = \int_{a_1}^{a_2} dr f^2(r) \div$$

$$\int_{a_1}^{a_2} dr f(r) \int_{a_1}^{a_2} dr' K(r,r') f(r'), \tag{2.8}$$

and note that

$$\int_{a_1}^{a_2} dr' K(r,r') f(r') = \frac{f(r)}{\gamma} \tag{2.9}$$

therefore

$$\gamma = \frac{\sum_{i=0}^{\infty} M_i^2}{\sum_{i=0}^{\infty} \frac{M_i^2}{\gamma_i}} \tag{2.10}$$

Now since γ_0 is the smallest eigenvalue,

$$\gamma \geq \gamma_0 . \tag{2.11}$$

Therefore, the variational principle will always yield a value greater than or equal to the eigenvalue sought. That (2.8) is a variational principle for (2.4) is shown below.

Multiply both sides of (2.8) through by the denominator of the right hand side to yield

$$\gamma\int_{a_1}^{a_2} drf(r)\int_{a_1}^{a_2} dr'K(r,r')f(r') = \int_{a_1}^{a_2} drf^2(r). \tag{2.12}$$

Take the variation of (2.12) to get

$$\delta\gamma\int_{a_1}^{a_2} drf(r)\int_{a_1}^{a_2} dr'K(r,r')f(r')+\gamma\int_{a_1}^{a_2} dr\delta f(r)\int_{a_1}^{a_2} dr'K(r,r')f(r')+$$

$$\gamma\int_{a_1}^{a_2} drf(r)\int_{a_1}^{a_2} dr'K(r,r')\delta f(r') = \int_{a_1}^{a_2} dr2f(r)\delta f(r). \tag{2.13}$$

Noting that $K(r,r')$ is real and symmetric yields

$$\int_{a_1}^{a_2} drf(r)\int_{a_1}^{a_2} dr'K(r,r')\delta f(r') = \int_{a_1}^{a_2} dr'\delta f(r')\int_{a_1}^{a_2} drK(r,r')f(r)$$

$$= \int_{a_1}^{a_2} dr\delta f(r)\int_{a_1}^{a_2} dr'K(r,r')f(r'). \tag{2.14}$$

Now for (2.8) to be a variational principle, $\delta\gamma$ must vanish. Thus, through the use of (2.14), (2.13) becomes

$$2\gamma\int_{a_1}^{a_2} dr\delta f(r)\int_{a_1}^{a_2} dr'K(r,r')f(r') = \int_{a_1}^{a_2} dr2f(r)\delta f(r) \tag{2.15}$$

or

$$\int_{a_1}^{a_2} dr\delta f(r)\left[f(r)-\gamma\int_{a_1}^{a_2} dr'K(r,r')f(r')\right] = 0. \tag{2.16}$$

This can be true iff

$$f(r) = \gamma \int_{a_1}^{a_2} dr' K(r,r') f(r') \tag{2.17}$$

which is (2.4), and, therefore, (2.8) is a variational principle for (2.4).

In order to obtain an estimate of the eigenvalue for a system of a given size, insert a trial function for the flux containing adjustable parameters into (2.8) and minimize (2.8) with respect to the parameters. Algebraically determine the parameters from the resulting system and reinsert them into (2.8) to estimate the eigenvalue. To estimate the critical size for a given eigenvalue, guess the size, insert the parametric trial function, and determine the eigenvalue estimate for that size. If the estimate is larger than the desired eigenvalue, increase the size. If the estimate is less than the desired eigenvalue, decrease the size. Iterate upon the size until the eigenvalue estimate converges satisfactorily to the eigenvalue for which the critical size is desired.

C. Variational Results

The procedure outlined in Section B above has been implemented in the three common one-dimensional geometries. Trial functions of the V_2 type (called quadratic trial functions) as in (2.18), and of the V_4 type (called quartic trial functions) as in (2.19), were used in each geometry:

$$\Phi(r) = 1-a_1 r^2, \tag{2.18}$$

$$\Phi(r) = 1-a_1 r^2-a_2 r^4. \tag{2.19}$$

It should be noted that the trial functions given above, with the exception that a_1 and a_2 are variable parameters and not fixed constants, have the same form as the leading terms of the expansion of the asymptotic solution to the transport equation. This holds true for the cosine spatial flux distribution in slab geometry, sine divided by r spatial flux distribution in spherical geometry, and the zero order Bessel function of the first kind spatial flux distribution in cylindrical geometry.

The critical size estimates, for various values of the neutron multiplicity, are compared to other methods in Table II for slabs, Table III for spheres, and Table IV for cylinders. Appendix D contains details of the implementation of the variational technique for the slab, Appendix E for the sphere, and Appendix F for the cylinder. Appendix G contains details of the minimization of the variational principle for both quadratic and quartic trial functions. The V_4 cylindrical results are unique and are among the best cylindrical critical radii estimates available. The V_4 results for slab and sphere differ by no more than one fortieth of a percent from the "benchmark" values of Kaper, Leaf, and Lindeman[10]. Therefore, it will be assumed that the scalar fluxes, (2.19), resulting from the variational techniques, are very precise and any error in the leakage operator calculations must result from the leakage operator formalism itself and not from inadequate initiating fluxes.

Table II

Critical Half-Thickness Estimates, in Units of the Total Mean Free Path, for Infinite Slabs as a Function of c

c	V_2	V_4	EEP^4	$P_1{}^4$	$P_3{}^4$	$P_5{}^4$	$S_2{}^4$
1.6	.5120	.5120	.5152	.6796	.5590	.5299	.6197
1.4	.7366	.7366	.7384	.9191	.7793	.7501	.8455
1.2	1.2894	1.2894	1.2898	1.4850	1.3185	1.2985	1.3896
1.1	2.1134	2.1134	2.1133	2.3087	2.1354	2.1210	2.1984
1.05	3.3010	3.3005	3.3002	3.4878	3.3191	3.3073	3.3718
1.02	5.6706	5.6661	5.6655	5.8391	5.6828	5.6723	5.7262

Table II

continued

c	S_4[4]	S_8[4]	S_{16}[4]	Case's[10]	IT_2[11]	IT_4[11]
1.6	.5223	.5138	.5125	.5120	.5120	.5120
1.4	.7435	.7383	.7372	.7366	.7366	.7366
1.2	1.2949	1.2912	1.2902	1.2894	1.2894	1.2894
1.1	2.1198	2.1155	2.1146	2.1133	2.1134	2.1134
1.05	3.3078	3.3032	3.3023	3.3003	3.3010	3.3004
1.02	5.6747	5.6702	5.6694	5.6655	5.6708	5.6660

Table III

Critical Radii Estimates, in Units of the

Total Mean Free Path, for Spheres as a Function of c

c	V_2	V_4	EEP^4	$P_1{}^4$	$P_3{}^4$	$P_5{}^4$	$S_2{}^4$
1.6	1.4768	1.4761	1.4759	1.8504	1.5502	1.4969	1.4197
1.4	1.9870	1.9854	1.9853	2.3530	2.0394	1.9994	1.9198
1.2	3.1785	3.1723	3.1720	3.5129	3.2041	3.1813	3.0963
1.1	4.8934	4.8733	4.8727	5.1766	4.8953	4.8805	4.7960
1.05	7.3326	7.2784	7.2772	7.5435	7.2961	7.2842	7.2084
1.02	12.1835	12.0305	12.0270	12.2520	12.0450	12.0340	11.9780

Table III

continued

c	S_4[4]	S_8[4]	S_{16}[4]	Case's[10]	IT_2[11]	IT_4[11]
1.6	1.4613	1.4716	1.4742	1.4761	1.4768	1.4761
1.4	1.9685	1.9801	1.9830	1.9853	1.9870	1.9854
1.2	3.1530	3.1659	3.1690	3.1721	3.1785	3.1721
1.1	4.8535	4.8659	4.8688	4.8727	4.8935	4.8728
1.05	7.2591	7.2699	7.2723	7.2772	7.3327	7.2775
1.02	12.0120	12.0200	12.0210	12.0275	12.1845	12.0289

Table IV

Critical Radii Estimates, in Units of the

Total Mean Free Path, for Infinite Cylinders as a Function of c

c	V_2	V_4	EEP^4	$S_2{}^4$	$S_4{}^4$	$S_8{}^4$	$IT_4{}^{11}$
1.6	1.0209	1.0209	1.0253	1.0647	1.0230	1.0194	1.0209
1.4	1.3971	1.3970	1.4004	1.4411	1.3958	1.3950	1.3970
1.2	2.2882	2.2873	2.2892	2.3297	2.2824	2.2850	2.2872
1.1	3.5819	3.5776	3.5784	3.6211	3.5725	3.5757	3.5774
1.05	5.4265	5.4119	5.4118	5.4654	5.4097	5.4108	5.4115
1.02	9.0960	9.0445	9.0433	9.1351	9.0495	9.0454	9.0446

D. Eigenvalue Controversy

The technique created by Syros and Theocharopoulos[20], if it performed as claimed, would obviate the need for the leakage operators derived in this thesis. As noted before, Syros and Theocharopoulos[20] have published a neutron transport technique based upon expansion of the angular flux in a series of polynomials. These polynomials can be made to satisfy exactly homogeneous boundary conditions for any order of angular approximation. Presumably such an approximation would not be subject to the boundary perturbations caused by poor satisfaction of boundary conditions and would, therefore, be able to calculate leakages accurately. There is, however, reason to doubt the accuracy of their technique.

Syros and Theocharopoulos published a criticism[21] of Dahl and Sjöstrand[14] questioning the accuracy of the eigenvalues in Ref. 14. Syros and Theocharopoulos claim that the expansion of the scalar flux in Ref. 14 is an integral of a specified angular flux expansion. The specified angular flux expansion is shown not to satisfy homogeneous boundary conditions and, therefore, Syros and Theocharopoulos state that the eigenvalues of Dahl and Sjöstrand are inaccurate. Dahl and Sjöstrand's counter criticism[36] is that their approximation is made on the scalar flux in an equation that already contains the homogeneous boundary conditions, namely, the integral transport equation. Splawski, Ziver, and Galliara[25] have commented upon this controversy but present only one eigenvalue. Since their eigenvalue is less accurate than the other eigenvalues to be presented here, it will not be tabulated. In addition to the eigenvalues of Dahl and Sjöstrand there are at least two

other sets of neutron multiplicity eigenvalues which are relevant to this controversy. Such eigenvalues are the Case's Method eigenvalues of Kaper, Leaf, and Lindeman[10] and the V_4 eigenvalues obtained by the techniques developed in this chapter. Since the V_4 techniques do not involve the angular flux, they are not subject to the criticisms of Syros and Theocharopoulos. Since the controversy over eigenvalues has pertained, so far, only to the infinite slab geometry, only slab geometry eigenvalues will be listed in Table V. Close examination of the eigenvalues in Table V reveals that Dahl and Sjöstrand agree exactly with Kaper, Leaf, and Lindeman and that the V_4 slab results are only just slightly higher. The eigenvalues generated by the V_4 slab code were used as input data for a critical size search utilizing the direct leakage operator. These sizes are a check upon the V_4 slab variational eigenvalues and are very close to the initial dimensions. Note that the eigenvalues of Syros and Theocharopoulos are consistently lower. It is for this reason that the accuracy of their transport techniques seems questionable.

Table V

Neutron Multiplicity Eigenvalues for the Infinite Slab as a Function of Half-Thickness

half thickness in mfp	Syros and Theocharopoulos[20]	Dahl Sjöstrand[14]	Kaper, Leaf and Lindeman[10]	V_4	direct leakage operator critical half thickness in mfp
.5	1.61384	1.6153785	1.61537852	1.61538482	.4999999996
1.0	1.27625	1.2771018	1.277101824	1.27710861	.9999999988
1.5	1.16257	1.1631293	---	1.16313622	1.4999999970
2.0	1.10799	1.1084678	1.1084678323	1.10847459	2.0000000014
2.5	1.07724	1.0775728	---	1.07757921	2.5000000065

III. DIRECT LEAKAGE OPERATOR

A. Introductory Remarks

A system with no external sources is just critical when the neutron production rate equals the neutron absorption rate plus the neutron leakage rate. This balance condition is normally reflected in an empirical system eigenvalue known as the effective multiplication factor, or K_{eff}, which is the eigenvalue by which the fission source term would have to be divided to yield a critical system. For this reason K_{eff} is defined as[38]

$$K_{eff} = \frac{\text{production rate}}{\text{absorption rate} + \text{leakage rate}} \quad . \tag{3.1}$$

Obviously, from (3.1), the balance condition is satisfied and the system is just critical when K_{eff} equals unity.

Alternatively, and more convenient for this chapter, is a balance condition defined as follows; neutron production rate plus neutron scattering rate equals neutron absorption rate plus neutron scattering rate plus neutron leakage rate. This leads to an alternative eigenvalue, namely, the gamma eigenvalue[39] defined by

$$\gamma = \frac{\text{production rate} + \text{scattering rate}}{\text{absorption rate} + \text{scattering rate} + \text{leakage rate}} \quad . \tag{3.2}$$

Gamma also goes to unity when the system is just critical. The definition of γ, (3.2), has a different off-critical value for a given

set of parameters than (3.1) but is just as useful for determining the critical size.

Equation (3.2) also does not have the physical interpretation that (3.1) has, namely, that division of the fission source term by (3.1) will yield a critical system (or, more accurately, will cause the mathematical model involved to look critical). For a steady state system with monoenergetic neutrons, isotropic fission, isotropic scattering, and spatially constant cross sections, the following definitions apply:

$$\text{production rate} = \nu\sigma_f \int_V \Phi(r)dV; \tag{3.3a}$$

$$\text{scattering rate} = \sigma_s \int_V \Phi(r)dV; \tag{3.3b}$$

$$\text{absorption rate} = \sigma_a \int_V \Phi(r)dV; \tag{3.3c}$$

$$\sigma_t = \sigma_a + \sigma_s; \tag{3.4}$$

and

$$c = \frac{\nu\sigma_f + \sigma_s}{\sigma_t} . \tag{3.5}$$

The total macroscopic cross section is σ_t, the macroscopic fission cross section is σ_f, the macroscopic absorption cross section is σ_a, the macroscopic scattering cross section is σ_s, c is the neutron multiplicity, and ν is the mean number of fission neutrons produced per fission. Substituting (3.3), (3.4), and (3.5) into (3.2), and dividing both numerator and denominator by σ_t yields

$$\gamma = \frac{c\int\Phi(r)dV}{\int\Phi(r)dV + \frac{1}{\sigma_t}\text{leakage rate}} . \qquad (3.6)$$

To obtain a critical size estimate for a given neutron multiplicity, say c_g, requires, as input to (3.6), a scalar flux shape and a leakage rate. In this chapter, the leakage rate is determined from the scalar flux shape through the use of the direct leakage operator. The procedure used to determine the critical size for a given neutron multiplicity, c_g, is as follows. First a size guess is used to determine a scalar flux approximation from an appropriate V_n technique. This scalar flux is used to estimate γ from (3.6). If γ is less than unity, an increment in the size guess is made; γ greater than unity requires a decrement in the size guess. Once γ has converged to unity, the critical size is known. From Chapter Two it is known that the V_n scalar fluxes (which implicitly correspond to K_{eff} equaling unity in (3.1)) are the lowest eigenfunctions for a system of a given size and correspond to a definite neutron multiplicity eigenvalue, say c_v. Therefore, it is obvious that c_g, the eigenvalue for which a critical size is sought, and c_v, the eigenvalue corresponding to the current V_n scalar flux guess, should not be equal unless (3.6) predicts γ is identically unity. Once γ is identically unity, then c_g should equal c_v, and any difference is a measure of the accuracy of the direct leakage operator. A more straightforward accuracy comparison is to compare the critical size estimates for a specified c as calculated by both the V_n method and by (3.6) through the use of the direct leakage operator.

B. General Direct Leakage Operator

Let V be a convex region of space for which it is desired to know the number of neutrons leaking out per unit time, N(t), given by

$$N(t) = \hat{L}_D \, S(\underline{r},\underline{\Omega},E,t-\frac{d(\underline{r},\underline{\Omega})}{v}) \, , \tag{3.7}$$

where $S(\underline{r},\underline{\Omega},E,t-\frac{d(\underline{r},\underline{\Omega})}{v})$ is the number of neutrons created from collisions or inhomogeneous sources within dV of position $\underline{r}$, within $d\underline{\Omega}$ of direction $\underline{\Omega}$, within dE of energy E, at time $t-\frac{d(\underline{r},\underline{\Omega})}{v}$, v is the velocity of a neutron of energy E, and $d(\underline{r},\underline{\Omega})$ is the distance along $\underline{\Omega}$ from $\underline{r}$ to the surface of region V. The direct leakage operator $\hat{L}_D$ is written as

$$\hat{L}_D = \int_{4\pi} d\underline{\Omega} \int_V dV \int_0^\infty dE e^{-\tau(E;\underline{r}_s - d(\underline{r},\underline{\Omega})\underline{\Omega}\rightarrow\underline{r}_s)} \, , \tag{3.8}$$

where τ is the optical depth,

$$\tau(E;\underline{r}_s - d(\underline{r},\underline{\Omega})\underline{\Omega}\rightarrow\underline{r}_s) = \int_0^{d(\underline{r},\underline{\Omega})} ds'' \sigma_t(\underline{r}_s - s''\underline{\Omega},E) \, , \tag{3.9}$$

and $\underline{r}_s$ is the position vector at a point on the surface of V where it is penetrated by a ray along $\underline{\Omega}$ from $\underline{r}$. The source, S, can be written as

$$S(\underline{r},\underline{\Omega},E,t) = Q(\underline{r},\underline{\Omega},E,t) + \int_{4\pi} d\underline{\Omega}' \int_{t'=-\infty}^{t'=t} dt' \int_0^\infty dE'$$

$$\sum_i \left[\nu_i(E')\sigma_i(\underline{r},E')f_i(t',\underline{\Omega}',E'\rightarrow t,\underline{\Omega},E)\right]\Psi(\underline{r},\underline{\Omega}',E',t') \, , \tag{3.10}$$

where Q is the inhomogeneous source term, the sum over i is a sum over all interactions, $\nu_i(E')$ is the mean number of secondaries for an

interaction of type i at energy E′, $\sigma_i(\underline{r},E')$ is the macroscopic interaction cross section for an interaction of type i at position $\underline{r}$ and energy E′, $f_i(t',\underline{\Omega}',E'\rightarrow t,\underline{\Omega},E)$ is the transfer probability of an interaction of type i from the primed coordinates to the unprimed coordinates, and $\Psi(\underline{r},\underline{\Omega}',E',t')$ is the angular flux of neutrons within dV of $\underline{r}$, within $d\underline{\Omega}'$ of $\underline{\Omega}'$, within dE′ of E′, at time t′. Now assume steady state, integrate (3.9) and (3.10) from E_g to E_{g-1}, and use the following multigroup definitions;

$$S_g(\underline{r},\underline{\Omega}) = \int_{E_g}^{E_{g-1}} dE S(\underline{r},\underline{\Omega},E), \tag{3.11a}$$

$$Q_g(\underline{r},\underline{\Omega}) = \int_{E_g}^{E_{g-1}} dE Q(\underline{r},\underline{\Omega},E), \tag{3.11b}$$

$$\Psi_g(\underline{r},\underline{\Omega}) = \int_{E_g}^{E_{g-1}} dE \Psi(\underline{r},\underline{\Omega},E), \tag{3.11c}$$

$$\sum_{g'=1}^{G} \sum_i \nu_{ig'}\sigma_{ig'\rightarrow g}(\underline{r}) f_{ig'}(\underline{\Omega}'\rightarrow\underline{\Omega})\Psi_{g'}(\underline{r},\underline{\Omega}')$$

$$= \int_0^{\infty} dE' \int_{E_g}^{E_{g-1}} dE \sum_i \left[\nu_i(E')\sigma_i(\underline{r},E') f_i(\underline{\Omega}',E'\rightarrow\underline{\Omega},E)\right]\Psi(\underline{r},\underline{\Omega}',E'), \tag{3.11d}$$

$$\tau_g\left(\underline{r}_s - d(\underline{r},\underline{\Omega})\underline{\Omega}\rightarrow\underline{r}_s\right) = \int_{E_g}^{E_{g-1}} dE \int_0^{d(\underline{r},\underline{\Omega})} ds'' \sigma_t(\underline{r}_s - s''\underline{\Omega},E)$$

$$= \int_0^{d(\underline{r},\underline{\Omega})} ds'' \sigma_{tg}(\underline{r}_s - s''\underline{\Omega}). \tag{3.11e}$$

Then it follows that

$$N = \hat{L}_D S_g(\underline{r},\underline{\Omega}), \tag{3.12}$$

$$\hat{L}_D = \sum_{g=1}^{G} \int_{4\pi} d\underline{\Omega} \int_V dV \; e^{-\tau_g(\underline{r}_s - d(\underline{r},\underline{\Omega})\underline{\Omega} \to \underline{r}_s)}, \tag{3.13}$$

and

$$S_g(\underline{r},\underline{\Omega}) = Q_g(\underline{r},\underline{\Omega}) + \sum_{g'=1}^{G} \sum_i \nu_{ig'} \sigma_{ig' \to g}(\underline{r}) \int_{4\pi} d\underline{\Omega}' f_{ig'}(\underline{\Omega}' \to \underline{\Omega}) \Psi_{g'}(\underline{r},\underline{\Omega}'). \tag{3.14}$$

Now consider only monoenergetic neutrons, isotropic scattering, isotropic fission, spatially constant cross sections, and no external sources. Then (3.12), (3.13), and (3.14) reduce to

$$N = \hat{L}_D S(\underline{r}), \tag{3.15}$$

$$\hat{L}_D = \int_{4\pi} d\underline{\Omega} \int_V dV e^{-\sigma_t d(\underline{r},\underline{\Omega})}, \tag{3.16}$$

and

$$S(\underline{r}) = \sum_i \nu_i \sigma_i \int_{4\pi} d\underline{\Omega}' \frac{1}{4\pi} \Psi(\underline{r},\underline{\Omega}'), \tag{3.17}$$

If only fission and elastic scattering are taken into account, then

$$N = \hat{L}_D \; S(\underline{r}), \tag{3.18}$$

$$\hat{L}_D = \int_{4\pi} d\underline{\Omega} \int_V dV e^{-\sigma_t d(\underline{r},\underline{\Omega})}, \tag{3.19}$$

$$S(\underline{r}) = (\nu\sigma_f + \sigma_s)\frac{1}{4\pi}\Phi(\underline{r}), \tag{3.20}$$

and

$$\Phi(\underline{r}) = \int d\underline{\Omega}'\Psi(\underline{r},\underline{\Omega}'). \tag{3.21}$$

Equations (3.18) and (3.19) are used to calculate the leakage from three one-dimensional systems in Sections C, D, E, and F of this chapter.

C. Direct Leakage Operator in Slab Geometry

The geometry of interest is shown in Fig. 1. This geometry is a semi-infinite slab, which is infinite in the y and z coordinates and of half-thickness b in the x coordinate. The angular coordinates in this system are the polar angle Θ, which is the angle the neutron makes with the positive x-axis, and the azimuthal angle χ, with respect to which there is symmetry. A convenient notation is to take $\mu=\cos\Theta$. In this geometry, (3.18), (3.19), and (3.20) combine to yield

$$N = \frac{\nu\sigma_f + \sigma_s}{4\pi}\int_{-b}^{b} dx\Phi(x)\int_0^{2\pi} d\chi\int_{-1}^{1} d\mu e^{-\sigma_t d(x,\mu)}. \tag{3.22}$$

For $\mu>0$ the neutron can only leak through the right hand side, and for $\mu<0$ the neutron can only leak through the left hand side. Therefore, it follows that

$$d(x,\mu) = \frac{b-x}{\mu}, \qquad \text{for } \mu>0, \tag{3.23a}$$

and

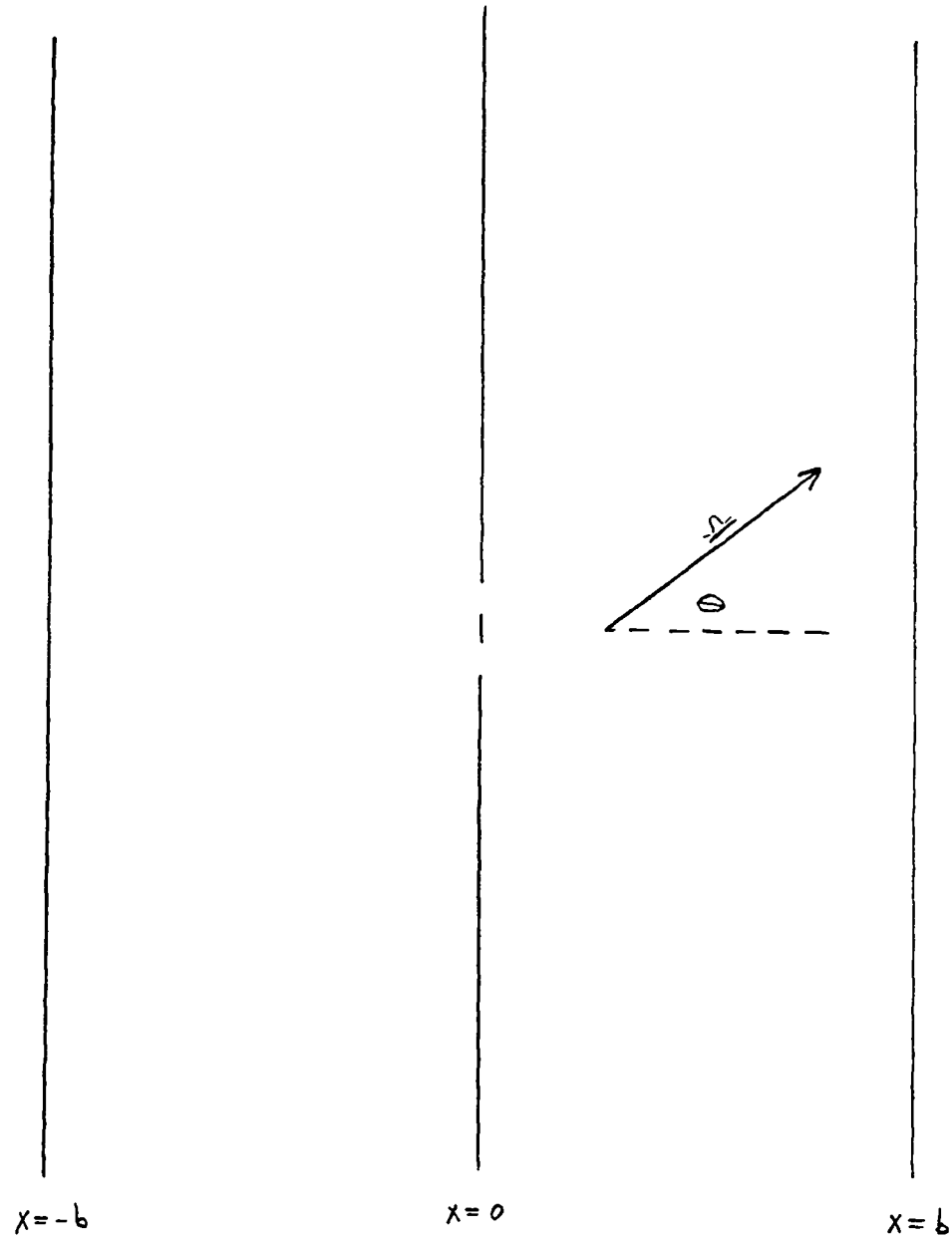


Figure 1.

One-dimensional slab geometry.

$$d(x,\mu) = \frac{b+x}{-\mu}, \qquad \text{for } \mu<0. \tag{3.23b}$$

Using (3.22) and (3.23), and noting the azimuthal symmetry, yields

$$N = \frac{\nu\sigma_f + \sigma_s}{2} \int_{-b}^{b} dx\Phi(x)\Big[\int_{-1}^{0} d\mu e^{+\sigma_t\left(\frac{b+x}{\mu}\right)}$$

$$+ \int_{0}^{1} d\mu e^{-\sigma_t\left(\frac{b-x}{\mu}\right)}\Big]. \tag{3.24}$$

Some additional definitions will provide convenience and insight. Therefore, let

$P_n^r(x)$ = the probability that a neutron emitted isotropically at position x escapes through the right hand side with n additional scatterings inside; (3.25)

$P_n^l(x)$ = the probability that a neutron emitted isotropically at position x escapes through the left hand side with n additional scatterings inside. (3.26)

Now note that isotropic emission in the geometry of Fig. 1, which has azimuthal symmetry, implies that

$\frac{d\mu}{2}$ = the probability that a neutron emitted isotropically ends up within $d\mu$ of μ. (3.27)

Combine (3.23), (3.25), (3.26), and (3.27) to get

$$P_0^r = \int_0^1 \frac{d\mu}{2} e^{-\sigma_t(\frac{b-x}{\mu})}, \tag{3.28}$$

$$P_0^l = \int_{-1}^0 \frac{d\mu}{2} e^{+\sigma_t(\frac{b+x}{\mu})}, \tag{3.29}$$

or, by using (A.2) and setting $\mu=-\mu$ in P_0^l above, it follows that

$$P_0^r = \frac{1}{2} E_2[\sigma_t(b-x)], \tag{3.30}$$

and

$$P_0^l = \frac{1}{2} E_2[\sigma_t(b+x)]. \tag{3.31}$$

Use of (3.28), (3.29), (3.30), and (3.31) reduces (3.24) to

$$N = (\nu\sigma_f + \sigma_s) \int_{-b}^{b} dx\Phi(x)[P_0^r(x)+P_0^l(x)], \tag{3.32}$$

or

$$N = \frac{\nu\sigma_f + \sigma_s}{2} \int_{-b}^{b} dx\Phi(x)(E_2[\sigma_t(b-x)] + E_2[\sigma_t(b+x)]). \tag{3.33}$$

Combine (3.5), (3.6), and (3.33), and normalize σ_t to unity to obtain

$$\gamma = c\int_{-b}^{b} dx\Phi(x) \div$$

$$\left[\int_{-b}^{b} dx\Phi(x) + \frac{c}{2}\int_{-b}^{b} dx\Phi(x)\left(E_2(b-x)+E_2(b+x)\right)\right]. \qquad (3.34)$$

The scalar flux, $\Phi(x)$, used in (3.34) is taken from variational theory. Since the form of the V_2 scalar flux is the same as the form of the V_4 scalar flux, except that a_2 equals zero, the terms in (3.34) need be determined only once. The production term in the numerator of (3.34) leads to

$$\text{prod.} = c\int_{-b}^{b} dx\Phi(x) = c\int_{-b}^{b} dx(1-a_1x^2-a_2x^4), \qquad (3.35)$$

$$\text{prod.} = c(2b-\frac{2}{3}a_1b^3-\frac{2}{5}a_2b^5). \qquad (3.36)$$

Similarly, the collision term in the denominator of (3.34) is

$$\text{coll.} = 2b-\frac{2}{3}a_1b^3-\frac{2}{5}a_2b^5. \qquad (3.37)$$

The leakage term in the denominator of (3.34) is

$$\text{leak.} = \frac{c}{2}\int_{-b}^{b} dx(1-a_1x^2-a_2x^4)\left[E_2(b-x)+E_2(b+x)\right]. \qquad (3.38)$$

To do this integral, use (A.2), the definition of $E_n(x)$, and interchange the order of integration. Perform the integration over x, collect terms, and then perform the integration over μ by using (A.2). This leads to

$$\text{leak.} = \frac{c}{2}\left[1-2E_3(2b)-a_1\left(1-\frac{4}{3}\,b+b^2-2b^2E_3(2b)-4bE_4(2b)\right.\right.$$

$$-4E_5(2b))-a_2\left(b^4-\frac{8}{3}\,b^3+6b^2-\frac{48}{5}\,b+8-2b^4E_3(2b)\right.$$

$$\left.\left.-8b^3E_4(2b)-24b^2E_5(2b)-48bE_6(2b)-48E_7(2b)\right)\right]. \qquad (3.39)$$

Now that all the terms in (3.34) are known for a V_2 or V_4 scalar flux shape, some calculations will be performed using the iterative technique outlined in Section A. The results are presented in Table VI. Note that the direct leakage approach yielded the exact same critical half-thickness as the appropriate V_n technique. This is positive verification of the accuracy of the direct leakage operator. More comparison values will be found in Chapter Two, Table II.

Table VI

Critical Half-Thickness Estimates, in Units of the Total Mean Free Path, for Infinite Slabs as a Function of c

c	V_2	V_4	Direct leakage with V_2 scalar flux	Direct leakage with V_4 scalar flux	S_{16}[4]	Case's[10]	IT_4[11]
1.6	.5120	.5120	.5120	.5120	.5125	.5120	.5120
1.4	.7366	.7366	.7366	.7366	.7372	.7366	.7366
1.2	1.2894	1.2894	1.2894	1.2894	1.2902	1.2894	1.2894
1.1	2.1134	2.1134	2.1134	2.1134	2.1146	2.1133	2.1134
1.05	3.3010	3.3005	3.3010	3.3005	3.3023	3.3003	3.3004
1.02	5.6706	5.6661	5.6706	5.6661	5.6694	5.6655	5.6660

D. Direct Leakage Operator in Spherical Geometry

The geometry of interest is a one-dimensional sphere of radius b. This geometry is shown in Fig. 2. The position coordinates are α, δ, and r, and μ and w are angular (direction) coordinates; μ is the cosine of the polar angle, the angle the neutron makes with the position vector, and w is the azimuthal angle with respect to which there is symmetry. Therefore, (3.18), (3.19), and (3.20) combine to yield

$$N = \frac{(\nu\sigma_f+\sigma_s)}{4\pi} \int_0^b dr 4\pi r^2 \Phi(r) \int_0^{2\pi} dw \int_{-1}^{1} d\mu e^{-\sigma_t d(r,\mu)}, \tag{3.40}$$

which reduces to

$$N = 2\pi(\nu\sigma_f+\sigma_s) \int_0^b dr r^2 \Phi(r) \int_{-1}^{1} d\mu e^{-\sigma_t d(r,\mu)}. \tag{3.41}$$

Now, due to the azimuthal symmetry, the distance to the surface can be calculated from Fig. 3. Note the following identities;

$$d(r,\mu)\sin(\Theta) = b \sin(\Phi); \tag{3.42a}$$

$$\beta = \Theta - \Phi; \tag{3.42b}$$

$$b = r \cos(\Phi)+d(r,\mu) \cos\beta; \tag{3.42c}$$

$$\mu = \cos(\Theta). \tag{3.43}$$

Use (3.42b) to remove β from (3.42c) and subsequently remove Φ through the use of (3.42a). Multiply through by b, divide through by $(b^2-d^2(r,\mu) \sin^2(\Theta))^{.5}$, and square the result to get

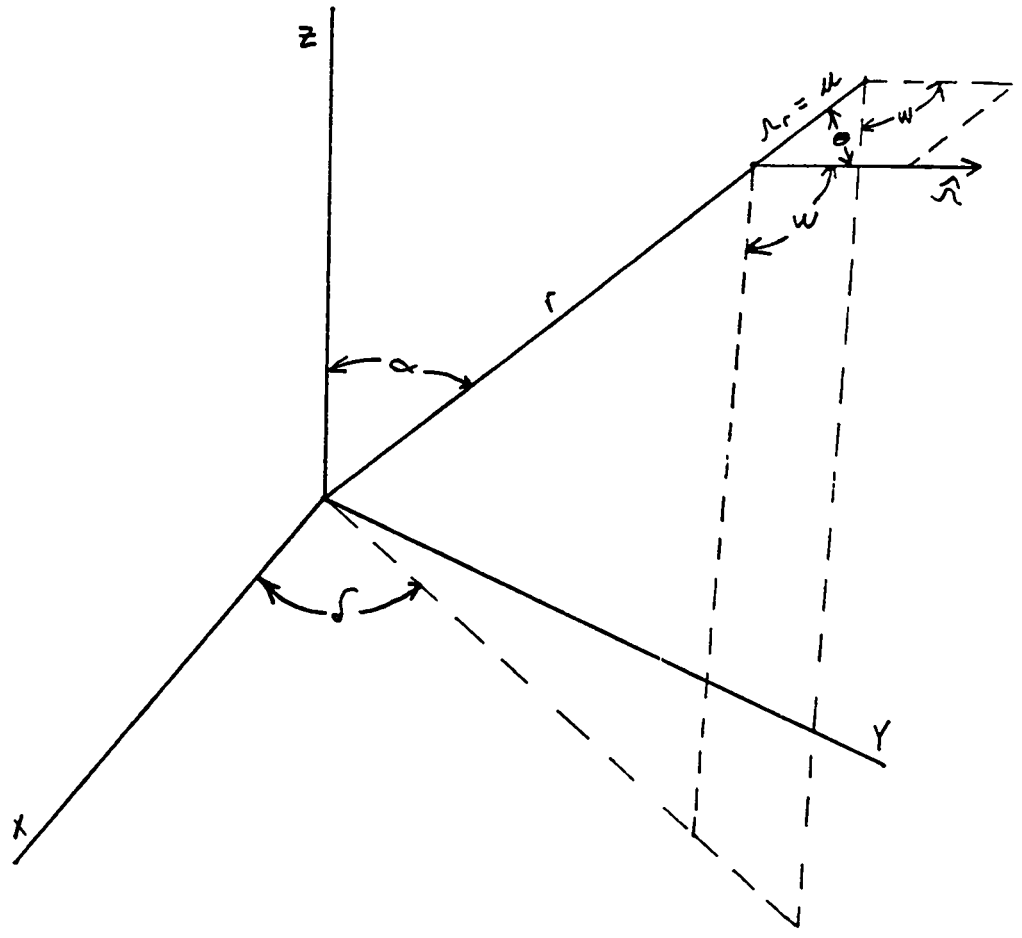


Figure 2.

One-dimensional spherical geometry.

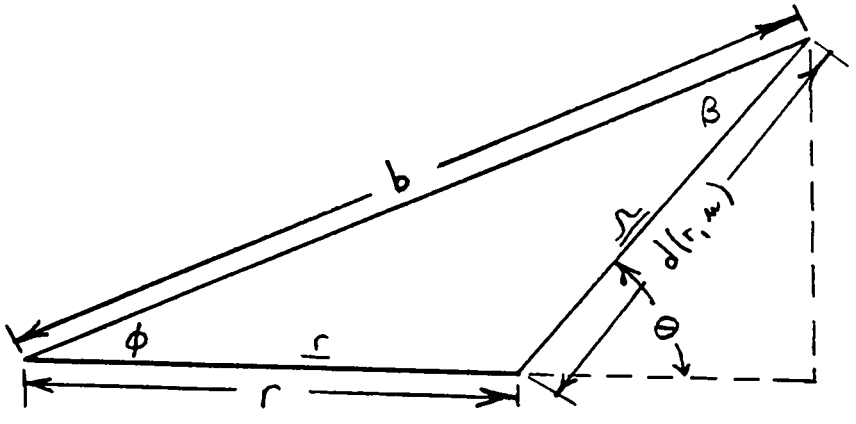


Figure 3.

Geometry of a plane formed by the position and direction vectors of a neutron in spherical geometry.

$$d^2(r,\mu) + d(r,\mu)2r\cos(\Theta) + r^2 - b^2 = 0. \tag{3.44}$$

Use (3.43) and solve to obtain

$$d(r,\mu) = -r\mu + (b^2-r^2+r^2\mu^2)^{.5} . \tag{3.45}$$

The positive root was taken because $d(r,\mu)$ is a distance and must, therefore, be positive. Equation (3.45) leads to

$$N = 2\pi(\nu\sigma_f+\sigma_s)\int_0^b drr^2\Phi(r)\int_{-1}^{1} d\mu$$

$$e^{-\sigma_t[-r\mu+(b^2-r^2+r^2\mu^2)^{.5}]} . \tag{3.46}$$

Now introduce the definition,

$P_n(r)$ = the probability that a neutron emitted isotropically at position r escapes from the system with n additional scatterings inside, (3.47)

and note that isotropic emission in the geometry of Fig. 2, which has azimuthal symmetry, implies that

$\frac{d\mu}{2}$ = the probability that a neutron emitted isotropically ends up within $d\mu$ of μ. (3.48)

Combining (3.45), (3.47), and (3.48) yields

$$P_0(r) = \frac{1}{2}\int_{-1}^{1} d\mu e^{-\sigma_t[-r\mu+(b^2-r^2+r^2\mu^2)\cdot 5]} . \tag{3.49}$$

To integrate (3.49), use the substitution

$$a = \sigma_t[(b^2-r^2+r^2\mu^2)\cdot 5-r\mu], \tag{3.50}$$

where

$$\mu = \frac{\sigma_t^2(b^2-r^2)-a^2}{\sigma_t 2ar} \tag{3.51}$$

and

$$d\mu = -\frac{1}{2r\sigma_t}\left[\frac{\sigma_t^2(b^2-r^2)+a^2}{a^2}\right]d\mu, \tag{3.52}$$

to get

$$P_0(r) = \frac{1}{4r\sigma_t}\int_{(b-r)\sigma_t}^{(b+r)\sigma_t} da\left[\frac{(b^2-r^2)\sigma_t^2+a^2}{a^2}\right]e^{-a}, \tag{3.53}$$

or

$$P_0(r) = \frac{1}{4r\sigma_t}[\sigma_t^2(b^2-r^2)(\int_{(b-r)\sigma_t}^{\infty} \frac{da}{a^2} e^{-a}$$

$$-\int_{(b+r)\sigma_t}^{\infty} \frac{da}{a^2} e^{-a}) + \int_{(b-r)\sigma_t}^{(b+r)\sigma_t} da e^{-a}]. \tag{3.54}$$

Integration and the use of (A.3) yield

$$P_0(r) = \frac{1}{4r\sigma_t}(\sigma_t(b+r)E_2[\sigma_t(b-r)] - \sigma_t(b-r)E_2[\sigma_t(b+r)]$$

$$-e^{-(b+r)\sigma_t} + e^{-(b-r)\sigma_t}); \tag{3.55}$$

$$P_0(r) = \frac{1}{4r}((b+r)E_2[\sigma_t(b-r)] - (b-r)E_2[\sigma_t(b+r)]$$

$$- \frac{e^{-(b+r)\sigma_t}}{\sigma_t} + \frac{e^{-(b-r)\sigma_t}}{\sigma_t}) . \tag{3.56}$$

From (3.46), (3.49), and (3.56),

$$N = 4\pi(\nu\sigma_f + \sigma_s)\int_0^b dr r^2 \Phi(r) P_0(r). \tag{3.57}$$

Combine (3.5), (3.6), (3.57), cancel the 4π, and normalize σ_t to unity to find

$$\gamma = c\int_0^b dr r^2 \Phi(r) \div$$

$$[\int_0^b drr^2\Phi(r)+c\int_0^b drr^2\Phi(r)P_0(r)]. \tag{3.58}$$

The scalar fluxes to be used in (3.58) are conveniently taken as the variational scalar fluxes. Since the form of the V_2 scalar flux is the same as the form of the V_4 scalar flux, except that a_2 equals zero, the terms in (3.58) need be done only once. The production term in the numerator of (3.58) is:

$$\text{prod.} = c\int_0^b drr^2\Phi(r) = c\int_0^b drr^2(1-a_1r^2-a_2r^4); \tag{3.59}$$

$$\text{prod.} = c(\frac{b^3}{3} - \frac{a_1}{5}\, b^5 - \frac{a_2}{7}\, b^7). \tag{3.60}$$

Similarly, the collision term in the denominator of (3.58) is

$$\text{coll.} = \frac{b^3}{3} - \frac{a_1}{5}\, b^5 - \frac{a_2}{7}\, b^7. \tag{3.61}$$

The leakage term in the denominator of (3.58) is

$$\text{leak.} = c\int_0^b drr^2(1-a_1r^2-a_2r^4)\frac{1}{4r}[(b+r)E_2(b-r)$$

$$-(b-r)E_2(b+r)-e^{-(b+r)} + e^{-(b-r)}]. \tag{3.62}$$

To do this integral, use (A.2), the definition of $E_n(x)$, and interchange the order of integration. Perform the integration over x, collect

terms, and then perform the integration over μ by using (A.2). This leads to

$$\text{leak.} = \frac{c}{4}\left[b^2 - \frac{1}{2} - bE_4(2b) - 2E_5(2b) + (b+1)e^{-2b}\right]$$

$$- \frac{a_1 c}{4}\left[b^4 - \frac{4}{3}b^3 + \frac{3}{2}b^2 - 2 - b^3E_4(2b) - 6b^2E_5(2b)\right.$$

$$\left. - 18bE_6(2b) - 24E_7(2b) + e^{-2b}(b^3+3b^2+6b+6)\right]$$

$$- \frac{a_2 c}{4}\left[b^6 - \frac{8}{3}b^5 + \frac{15}{2}b^4 - 16b^3 + 20b^2 - 30 - b^5E_4(2b)\right.$$

$$- 10b^4E_5(2b) - 60b^3E_6(2b) - 240b^2E_7(2b) - 600bE_8(2b)$$

$$\left. - 720E_9(2b) + e^{-2b}(b^5+5b^4+20b^3+60b^2+120b+120)\right]. \qquad (3.63)$$

Now that all the terms in (3.58) are known for a V_2 or V_4 scalar flux shape, some calculations will be performed using the iterative technique outlined in section A. The results are presented in Table VII. Note that the direct leakage approach yielded the exact same critical radii as the appropriate V_n technique. This is, of course, justification for considering the direct leakage operator exact. More comparison values will be found in Chapter Two, Table III.

Table VII

Critical Radii Estimates, in Units of the Total Mean Free Path, for Spheres as a Function of c

c	V_2	V_4	Direct leakage with V_2 scalar flux	Direct leakage with V_4 scalar flux	S_{16}[4]	Case's[10]	IT_4[11]
1.6	1.4768	1.4761	1.4768	1.4761	1.4742	1.4761	1.4761
1.4	1.9870	1.9854	1.9870	1.9854	1.9830	1.9853	1.9854
1.2	3.1785	3.1723	3.1785	3.1723	3.1690	3.1721	3.1721
1.1	4.8934	4.8733	4.8934	4.8733	4.8688	4.8727	4.8728
1.05	7.3326	7.2784	7.3326	7.2784	7.2723	7.2772	7.2775
1.02	12.1835	12.0305	12.1835	12.0305	12.0210	12.0275	12.0289

E. Direct Leakage Operator in Cylindrical Geometry

The one-dimensional cylindrical geometry of interest is shown in Fig. 4. The position coordinates are Φ, r, and z, while μ and χ are angular (direction) coordinates; μ is the cosine of the polar angle, the angle which the neutron makes with the $\underline{z}$ vector, and χ is the azimuthal angle. This azimuthal angle is the angle between the planes formed by the $\underline{r}$ and $\underline{z}$ vectors and the $\underline{\Omega}$ and $\underline{z}$ vectors. Therefore, (3.18), (3.19), and (3.20) combine to yield

$$N = \frac{(\nu\sigma_f+\sigma_s)}{4\pi}\int_0^b dr 2\pi r\Phi(r)\int_0^{2\pi} d\chi\int_{-1}^{1} d\mu e^{-\sigma_t d(r,\chi,\mu)}, \tag{3.64}$$

which reduces to

$$N = \frac{(\nu\sigma_f+\sigma_s)}{2}\int_0^b dr r\Phi(r)\int_0^{2\pi} d\chi\int_{-1}^{1} d\mu e^{-\sigma_t d(r,\chi,\mu)}. \tag{3.65}$$

Now $d(r,\chi,\mu)$ can be determined from two projections of Fig. 4. In Fig. 5, Fig. 4 is projected onto a plane perpendicular to the axis at z equals zero. In Fig. 6, Fig. 4 is projected onto a plane parallel to the cylindrical axis and containing $\underline{\Omega}$, the direction vector.

Obtain from Fig. 5 the following identities:

$$\beta = \chi - \Phi; \tag{3.66a}$$

$$\alpha = 90 + \Phi - \chi; \tag{3.66b}$$

$$b = r\cos(\Phi) + ZLD\sin(\alpha); \tag{3.66c}$$

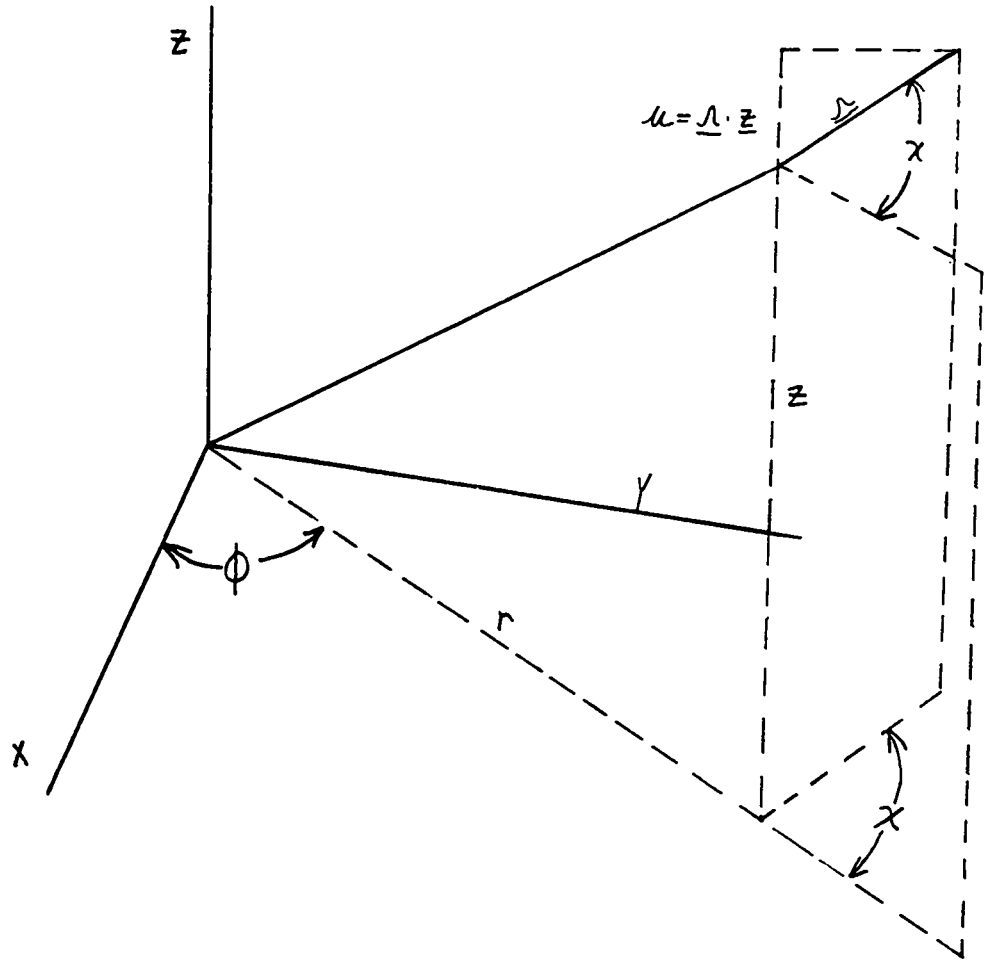


Figure 4.

One-dimensional cylindrical geometry.

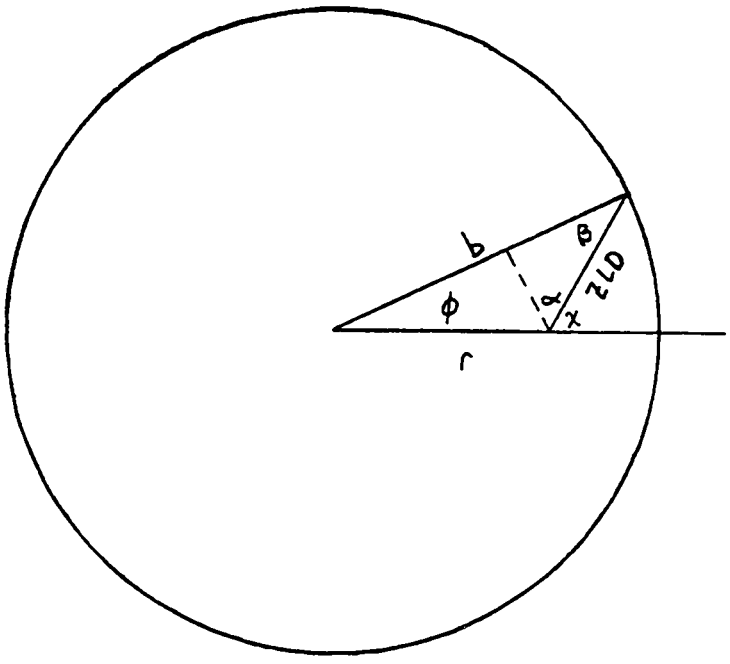


Figure 5.

A plane perpendicular to the cylinder axis.

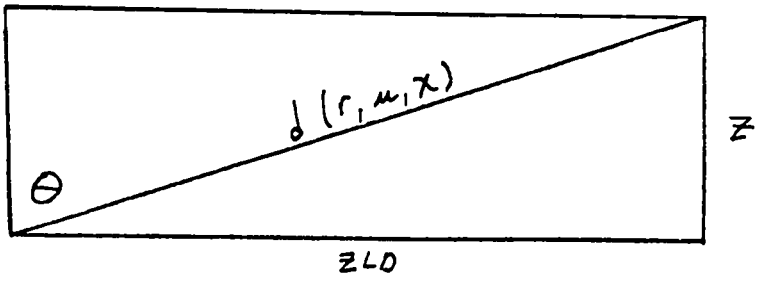


Figure 6.

A plane parallel to the cylinder axis.

$$ZLD \sin(\chi) = b \sin(\Phi). \tag{3.66d}$$

Begin with (3.66c) and use (3.66b) to remove α. Use (3.66d) to remove Φ, multiply through by b, divide through by $(b^2-ZLD^2\sin^2(\chi))^{\cdot 5}$, and square the result to get

$$ZLD^2+2r\cos(\chi)ZLD +r^2-b^2 = 0. \tag{3.67}$$

Solve (3.67) and note that ZLD is a distance and must, therefore, be positive:

$$ZLD = -r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{\cdot 5}. \tag{3.68}$$

Noting that $\mu=\cos\Theta$, and using Fig. 6, yields

$$d(r,\mu,\chi) = \frac{-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}} \quad . \tag{3.69}$$

With (3.69), (3.65) becomes

$$N = \frac{\nu\sigma_f+\sigma_s}{2}\int_0^b dr\, r\Phi(r)\int_0^{2\pi} d\chi\int_{-1}^{1} d\mu$$

$$e^{-\sigma_t\left[\frac{-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}}\right]}. \tag{3.70}$$

Now introduce the following definition;

$P_n(r)$ = the the probability that a neutron emitted isotropically at position r escapes the system with n additional scatterings inside, (3.71)

and note that isotropic emission in the geometry of Fig. 4 implies that

$\frac{d\mu}{2}\frac{d\chi}{2\pi}$ = the probability that a neutron emitted isotropically ends up within $d\mu$ of μ and within $d\chi$ of χ. (3.72)

Combining (3.69), (3.71), and (3.72) yields

$$P_0(r) = \int_{-1}^{1} \frac{d\mu}{2} \int_0^{2\pi} \frac{d\chi}{2\pi} e^{-\sigma_t \left[\frac{-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{.5}}{(1-\mu^2)^{.5}}\right]}, \quad (3.73)$$

or, upon using the symmetry in μ and χ,

$$P_0(r) = \frac{1}{\pi} \int_0^{1} d\mu \int_0^{\pi} d\chi \, e^{-\sigma_t \left[\frac{-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{.5}}{(1-\mu^2)^{.5}}\right]}. \quad (3.74)$$

Interchange the order of integration and make the transformation $\mu = \frac{\sinh(a)}{\cosh(a)}$ to yield

$$P_0(r) = \frac{1}{\pi} \int_0^{\pi} d\chi \int_0^{\infty} \frac{da}{\cosh^2 a}$$

$$e^{-\sigma_t [-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{.5}]\cosh(a)}. \quad (3.75)$$

The definition

$$K_{in}(x) = \int_0^{\infty} \frac{e^{-x\cosh(a)}}{\cosh^n(a)}\, da, \tag{3.76}$$

from Bickley and Naylor[40], is used upon (3.75) to obtain

$$P_0(r) = \frac{1}{\pi} \int_0^{\pi} d\chi K_{i2}[(-r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{.5})\sigma_t]. \tag{3.77}$$

This integral can be done by making a transformation from χ to Φ (see Fig. 5), which is a transformation from a local to a central coordinate system. From Fig. 5,

$$ZLD = -r\cos(\chi)+(b^2-r^2+r^2\cos^2(\chi))^{.5}, \tag{3.78}$$

and from the law of cosines,

$$ZLD = (b^2+r^2-2br\cos(\Phi))^{.5}. \tag{3.79}$$

From (3.66d) $d\chi$ is found to be

$$d\chi = d\Phi\, \frac{b^2-br\cos(\Phi)}{ZLD^2}. \tag{3.80}$$

With (3.78), (3.79), and (3.80), (3.77) becomes

$$P_0(r) = \frac{1}{\pi}\int_0^{\pi} d\Phi \frac{b^2-br\cos(\Phi)}{ZLD^2} K_{i2}(ZLD\sigma_t). \tag{3.81}$$

From Bickley and Naylor[40], $K_{i0}(x) = K_0(x)$, where $K_0(x)$ is a zero order modified Bessel function of the third kind, and

$$K_{in}(x) = \int_x^{\infty} dt K_{n-1}(t). \tag{3.82}$$

From (3.82),

$$K_{i2}(\sigma_t ZLD) = \int_{\sigma_t ZLD}^{\infty} dt \int_t^{\infty} K_0(y)dy. \tag{3.83}$$

Make the transformation y=tx to get

$$K_{i2}(\sigma_t ZLD) = \int_{\sigma_t ZLD}^{\infty} dt t \int_1^{\infty} dx K_0(tx). \tag{3.84}$$

Make the transformation $t=\sigma_t ZLDt'$ to get

$$K_{i2}(\sigma_t ZLD) = \int_1^{\infty} \sigma_t^2 ZLD^2 t'dt' \int_1^{\infty} dx K_0(\sigma_t ZLDt'x). \tag{3.85}$$

Drop the primes, $t'=t$, and interchange the order of integration:

$$K_{i2}(\sigma_t ZLD) = \sigma_t^2 ZLD^2 \int_1^\infty dx \int_1^\infty dtt K_0(\sigma_t ZLDtx). \tag{3.86}$$

Therefore, (3.81) becomes

$$P_0(r) = \frac{\sigma_t^2}{\pi} \int_0^\pi d\Phi(b^2 - br\cos(\Phi)) \int_1^\infty dx \int_1^\infty dtt K_0(\sigma_t ZLDtx), \tag{3.87}$$

or

$$P_0(r) = \frac{\sigma_t^2}{\pi} \int_1^\infty dx \int_1^\infty dtt \int_0^\pi d\Phi(b^2 - br\cos(\Phi))$$

$$K_0[\sigma_t tx(b^2 + r^2 - 2br\cos(\Phi))^{.5}] . \tag{3.88}$$

Using (B.20), (3.88) becomes

$$P_0(r) = \frac{\sigma_t^2}{\pi} \int_1^\infty dx \int_1^\infty dtt \int_0^\pi d\Phi(b^2 - br\cos(\Phi))$$

$$\left(\sum_{k=-\infty}^{\infty} K_k(\sigma_t txb) I_k(\sigma_t txr)\cos(k\Phi) \right), \tag{3.89}$$

which upon integration and application of L'Hospital's rule yields

$$P_0(r) = \frac{\sigma_t^2}{\pi} \int_1^\infty dx \int_1^\infty dtt [\pi b^2 K_0(\sigma_t txb) I_0(\sigma_t txr)$$

$$- \frac{\pi}{2} brK_{-1}(\sigma_t txb)I_{-1}(\sigma_t txr) - \frac{\pi}{2} brK_1(\sigma_t txb)I_1(\sigma_t txr)\Big]. \qquad (3.90)$$

Use (B.7) with (3.90), and let $y=tx\sigma_t$ to find

$$P_0(r) = \int_1^\infty \frac{dx}{x^2} \int_{x\sigma_t}^\infty dyy\left[b^2K_0(yb)I_0(yr) - brK_1(yb)I_1(yr)\right]. \qquad (3.91)$$

Integration by parts of the first term in the integrand of (3.91), so as to raise the order of the K_0 term to K_1, and application of the asymptotic series (B.6) will produce

$$P_0(r) = \sigma_t b\int_1^\infty \frac{dx}{x} K_1(xb\sigma_t)I_0(xr\sigma_t), \qquad (3.92)$$

which is far simpler than (3.73).

With (3.92) and (3.73), (3.70) becomes

$$N = 2\pi(\nu\sigma_f+\sigma_s)\int_0^b drr\Phi(r)P_0(r). \qquad (3.93)$$

Combine (3.5), (3.6), (3.93), cancel the 2π and normalize σ_t to unity to obtain

$$\gamma = c\int_0^b drr\Phi(r) \div$$

$$\left[\int_0^b drr\Phi(r)+c\int_0^b drr\Phi(r)P_0(r)\right] . \qquad (3.94)$$

The scalar fluxes to be used in (3.94) are conveniently taken as the variational scalar fluxes. Since the form of the V_2 scalar flux is the same as the form of the V_4 scalar flux, except that a_2 equals zero, the terms in (3.94) need be done only once. The production term in the numerator of (3.94) is

$$\text{prod.} = c\int_0^b drr\Phi(r) = c\int_0^b drr(1-a_1r^2-a_2r^4); \tag{3.95}$$

$$\text{prod.} = c(\frac{b^2}{2} - a_1\frac{b^4}{4} - a_2\frac{b^6}{6}). \tag{3.96}$$

Similarly, the collision term in the denominator of (3.94) is

$$\text{coll.} = (\frac{b^2}{2} - a_1\frac{b^4}{4} - a_2\frac{b^6}{6}). \tag{3.97}$$

The leakage term in the denominator of (3.94) is

$$\text{leak.} = cb\int_0^b dr(r-a_1r^3-a_2r^5)\int_1^\infty \frac{dx}{x} K_1(bx)I_0(xr). \tag{3.98}$$

To do this integral, interchange the order of integration, make the transformation rx=y, and integrate over y by parts. This will produce

$$\text{leak.} = bc\int_1^\infty dxK_1(bx)[I_1(bx)(\frac{b}{x^2} - \frac{a_1b^3}{x^2} - \frac{4a_1b}{x^4} - \frac{a_2b^5}{x^2}$$

$$- \frac{16a_2b^3}{x^4} - \frac{64a_2b}{x^6}) + I_0(bx)(\frac{2a_1b^2}{x^3} + \frac{4a_2b^4}{x^3} + \frac{32a_2b^2}{x^5})]. \quad (3.99)$$

The worst case in the numerical integration of (3.99) for large x, can be seen to be $K_1(bx)I_1(bx)\frac{b}{x^2}$, which asymptotically goes as $\frac{1}{2x^3}$ $[1+O(x^{-1})]$ for large x. The numerical integration of (3.99) will therefore have an error on the order of about 10^{-8} if the upper limit in (3.99) is 100,000 instead of infinity. This should be quite acceptable.

Now that all the terms in (3.94) are known for V_2 and V_4 scalar flux shapes, some calculations will be performed using the iterative technique outlined in section A. The results are presented in Table VIII. Note that the direct leakage approach yielded the exact same critical radii as the appropriate V_n technique. This is, of course, justification for considering the direct leakage operator exact. More comparison values will be found in Chapter Two, Table IV.

Table VIII

Critical Radii Estimates, in Units of the

Total Mean Free Path, for Infinite Cylinders as a Function of c

c	V_2	V_4	Direct leakage with V_2 scalar flux	Direct leakage with V_4 scalar flux	S_8^4	S_{16}^{11}	IT_4^{11}
1.6	1.0209	1.0209	1.0209	1.0209	1.0194	1.0231	1.0209
1.4	1.3971	1.3970	1.3971	1.3970	1.3950	1.3991	1.3970
1.2	2.2882	2.2873	2.2882	2.2873	2.2850	2.2891	2.2872
1.1	3.5819	3.5776	3.5819	3.5776	3.5757	3.5795	3.5774
1.05	5.4265	5.4119	5.4265	5.4119	5.4108	5.4140	5.4115
1.02	9.0960	9.0445	9.0960	9.0445	9.0454	9.0494	9.0446

F. Direct Leakage Operator and Diffusion Theory

A diffusion theory scalar flux initiation for the direct leakage operator is a good test of the operator's ability to calculate leakage accurately. Diffusion theory is easy to use and gives fairly good answers for large systems without strong sources or sinks. Traditionally, for one-group, one-region critical size problems, the diffusion theory criticality condition is that the materials buckling equals the geometrical buckling. The geometrical buckling, B_g^2, is determined from the solution of the scalar Helmholtz equation, (3.100), subject to conditions that the flux is positive and goes to zero at the extrapolated boundary:

$$\nabla^2\Phi(\underline{r}) + B_g^2\Phi(\underline{r}) = 0. \qquad (3.100)$$

The extrapolated boundaries are discussed in Appendix H. The materials buckling, B_m^2, is $\frac{\nu\sigma_f + \sigma_s - \sigma_t}{D}$ where $D = \frac{1}{3\sigma_t}$ for isotropic scattering. If σ_t is normalized to one, the materials buckling reduces to (3c-3). This procedure, which results from a neutron balance condition similar to that which led to (3.1), is used to calculate the diffusion theory critical sizes appearing in Tables IX, X, and XI. The asumption that $D = \frac{1}{3\sigma_t}$ leads to the most accurate results available from diffusion theory leakages. Readers interested in more detail should refer to Henry[38].

1. Slab Geometry

In a one-dimensional slab symmetric about x equals zero, the solution of (3.100) is $\cos(B_g x)$, subject to the condition that the flux vanish at a distance equal to the physical boundary plus the extrapolation distance. This yields

$$\Phi(x) = \cos(B_g x); \tag{3.101a}$$

$$B_g = \frac{\pi}{2b_{ex}}; \tag{3.101b}$$

$$b_{ex} = \text{b+extrapolation distance.} \tag{3.101c}$$

The production rate for use in the numerator of (3.6) is

$$\text{prod.} = c\int_{-b}^{b} dx\cos\left(\frac{\pi x}{2b_{ex}}\right); \tag{3.102}$$

$$\text{prod.} = \frac{4cb_{ex}}{\pi}\sin\left(\frac{\pi b}{2b_{ex}}\right). \tag{3.103}$$

The collision rate for use in the denominator of (3.6) is

$$\text{coll.} = \frac{4b_{ex}}{\pi}\sin\left(\frac{\pi b}{2b_{ex}}\right). \tag{3.104}$$

The leakage rate for use in the denominator of (3.6) is, after σ_t is normalized to unity,

$$N = \frac{\nu\sigma_f + \sigma_s}{2} \int_{-b}^{b} dx \cos\left(\frac{\pi x}{2b_{ex}}\right)[E_2(b-x) + E_2(b+x)], \tag{3.105}$$

or, rewriting (3.105) using (A.5), and noting symmetry,

$$N = (\nu\sigma_f + \sigma_s)\int_0^b dx \cos\left(\frac{\pi x}{2b_{ex}}\right) [e^{-(b-x)} + e^{-(b+x)}$$

$$-(b-x)E_1(b-x)-(b+x)E_1(b+x)] . \tag{3.106}$$

Using (3.103), (3.104), and (3.106) will produce from (3.6)

$$\gamma = \left[\frac{4cb_{ex}}{\pi} \sin\left(\frac{\pi b}{2b_{ex}}\right)\right] \div \left[\left(\frac{4b_{ex}}{\pi} \sin\left(\frac{\pi b}{2b_{ex}}\right)\right)\right.$$

$$+c\int_0^b dx \cos\left(\frac{\pi x}{2b_{ex}}\right)(e^{-(b-x)} + e^{-(b+x)}$$

$$\left.-(b-x)E_1(b-x)-(b+x)E_1(b+x)\right)] . \tag{3.107}$$

The integral in (3.107) is done numerically, and b is iterated until γ is unity. The results are tabulated in Table IX. Comparison of the results in Table IX reveals that the direct leakage operator moves diffusion theory into competition with such transport techniques as P_3-P_5 spherical harmonics expansions and S_4 discrete ordinates! The error from V_4 values varies from .92 percent to 1.75 percent.

2. Spherical Geometry

In a sphere, the solution of (3.100) is $\frac{1}{r}\sin(B_g r)$, subject to the condition that the flux vanishes at a distance equal to the physical boundary plus the extrapolation distance. This yields

$$\Phi(r) = \frac{1}{r}\sin(B_g r); \tag{3.108a}$$

$$B_g = \frac{\pi}{b_{ex}}; \tag{3.108b}$$

$$b_{ex} = b+\text{extrapolation distance}\ . \tag{3.108c}$$

The production rate for use in the numerator of (3.6) is

$$\text{prod.} = c\int_0^b dr 4\pi r^2 \frac{1}{r}\sin\left(\frac{\pi}{b_{ex}} r\right); \tag{3.109}$$

$$\text{prod.} = 4cb_{ex}\left[\frac{b_{ex}}{\pi}\sin\left(\frac{\pi}{b_{ex}}b\right)-b\cos\left(\frac{\pi}{b_{ex}}b\right)\right]. \tag{3.110}$$

The collision rate for use in the denominator of (3.6) is

$$\text{coll.} = 4b_{ex}\left[\frac{b_{ex}}{\pi}\sin\left(\frac{\pi}{b_{ex}}b\right) - b\cos\left(\frac{\pi}{b_{ex}}b\right)\right]. \tag{3.111}$$

The leakage rate for use in the denominator of (3.6) is, after σ_t is normalized to unity,

$$N = (\nu\sigma_f + \sigma_s)\int_0^b dr 4\pi r^2 \frac{1}{r}\sin\left(\frac{\pi}{b_{ex}}r\right)\frac{1}{4r}$$

$$\left[(b+r)E_2(b-r)-(b-r)E_2(b+r)- e^{-(b+r)} + e^{-(b-r)}\right], \tag{3.112}$$

or, rewriting (3.112) with (A.5) gives

$$N = (\nu\sigma_f + \sigma_s)\pi\int_0^b dr\sin\left(\frac{\pi}{b_{ex}}r\right)\left[(b^2-r^2)\left(E_1(b+r)-E_1(b-r)\right)\right.$$

$$\left.+(b+r+1)e^{-(b-r)} -(b-r+1)e^{-(b+r)}\right]. \qquad (3.113)$$

Using (3.110), (3.111), and (3.113) will produce from (3.6)

$$\gamma = 4cb_{ex}\left[\frac{b_{ex}}{\pi}\sin\left(\frac{\pi}{b_{ex}}b\right)-b\cos\left(\frac{\pi}{b_{ex}}b\right)\right] \div$$

$$\left[4b_{ex}\left(\frac{b_{ex}}{\pi}\sin\left(\frac{\pi}{b_{ex}}b\right)-b\cos\left(\frac{\pi}{b_{ex}}b\right)\right) + c\pi\int_0^b dr\sin\left(\frac{\pi}{b_{ex}}r\right)\right.$$

$$\left[(b^2-r^2)\left(E_1(b+r)-E_1(b-r)\right)+(b+r+1)e^{-(b-r)}\right.$$

$$\left.\left.-(b-r+1)e^{-(b+r)}\right]\right] . \qquad (3.114)$$

The integral in (3.114) is done numerically, and b is iterated until γ is unity. The results are tablulated in Table X. Comparison of the results in Table X indicates that the direct leakage operator makes diffusion theory comparable to P_3 spherical harmonics expansion and S_2-S_4 discrete ordinates! The error from V_4 calculations ranges from 1.27 percent to 1.64 percent for the direct leakage operator diffusion

theory calculation. In contrast, the Serber-Wilson error from V_4 ranges from -1.84 percent to -4.13 percent.

3. Cylindrical Geometry

In a cylinder, the solution of (3.100) is $J_0(B_g r)$, subject to the condition that the flux vanishes at a distance equal to the physical boundary plus the extrapolation distance. This yields

$$\Phi(r) = J_0(B_g r); \tag{3.115a}$$

$$B_g = \frac{2.405}{b_{ex}}; \tag{3.115b}$$

$$b_{ex} = b + \text{extrapolation distance}. \tag{3.115c}$$

The production rate for use in the numerator of (3.6) is

$$\text{prod.} = c\int_0^b dr 2\pi r J_0\left(\frac{2.405}{b_{ex}} r\right); \tag{3.116}$$

$$\text{prod} = \frac{2\pi b c b_{ex}}{2.405} J_1\left(\frac{2.405}{b_{ex}} b\right). \tag{3.117}$$

The collision rate for use in the denominator of (3.6) is

$$\text{coll.} = \frac{2\pi b b_{ex}}{2.405} J_1\left(\frac{2.405b}{b_{ex}}\right). \tag{3.118}$$

The leakage rate for use in the denominator of (3.6) is, after σ_t is normalized to unity,

$$N = (\nu\sigma_f + \sigma_s)\int_0^b dr 2\pi r J_0\left(\frac{2.405r}{b_{ex}}\right) b\int_1^\infty \frac{dx}{x} K_1(bx) I_0(rx), \tag{3.119}$$

which becomes after interchanging the order of integration

$$N = (\nu\sigma_f + \sigma_s)2\pi b\int_1^\infty \frac{dx}{x} K_1(bx)\int_0^b dr r J_0\left(\frac{2.405}{b_{ex}}r\right) I_0(rx). \tag{3.120}$$

Use (9.6.3) from Abramowitz and Stegun[41], which is

$$I_\nu(z) = e^{-.5\nu\pi i} J_\nu(e^{.5\pi i}z). \tag{3.121}$$

Then (3.121) reduces to

$$I_0(z) = J_0(iz). \tag{3.122}$$

Use (3.122), and make the transformation $\alpha = \frac{-2.405i}{b_{ex}}$, to reduce (3.120) to

$$N = 2\pi b(\nu\sigma_f + \sigma_s)\int_1^\infty \frac{dx}{x} K_1(bx)\int_0^b drrI_0(\alpha r)I_0(rx). \qquad (3.123)$$

This can be integrated through the use of (B.4) to

$$N = 2\pi b(\nu\sigma_f + \sigma_s)\int_1^\infty \frac{dx}{x} K_1(bx)\frac{1}{x^2-\alpha^2}$$

$$[-\alpha bI_0(bx)I_1(b\alpha)+bxI_0(b\alpha)I_1(bx)]. \qquad (3.124)$$

Note that $I_0(b\alpha)=J_0(\frac{2.405b}{b_{ex}})$, and $I_1(b\alpha)= -iJ_1(\frac{2.405b}{b_{ex}})$, to get

$$N = 2\pi b(\nu\sigma_f + \sigma_s)\int_1^\infty \frac{dxK_1(bx)}{x^3+x(\frac{2.405}{b_{ex}})^2}$$

$$[\frac{2.405b}{b_{ex}} J_1(\frac{2.405b}{b_{ex}})I_0(bx) + bxJ_0(\frac{2.405b}{b_{ex}})I_1(bx)] . \qquad (3.125)$$

Using (3.117), (3.118), and (3.125) will produce from (3.6)

$$\gamma = [\frac{2\pi bcb_{ex}}{2.405} J_1(\frac{2.405b}{b_{ex}})] \div [\frac{2\pi bb_{ex}}{2.405} J_1(\frac{2.405b}{b_{ex}})$$

$$+ 2\pi bc\int_1^{\infty} \frac{dxK_1(bx)}{x^3+x(\frac{2.405}{b_{ex}})^2} [\frac{2.405b}{b_{ex}} J_1(\frac{2.405b}{b_{ex}})I_0(bx)$$

$$+ bxJ_0(\frac{2.405b}{b_{ex}})I_1(bx)]] \quad . \qquad (3.126)$$

The integral in (3.126) is done numerically and b is iterated until γ is unity. The results are tabulated in Table XI. Comparison of the results in Table XI reveals that the direct leakage operator moves diffusion theory into competition with such transport techniques as S_2 discrete ordinates. This still represents a substantial improvement in that diffusion theory scalar fluxes are easier to obtain numerically than S_2 fluxes.

Table IX

Critical Half-Thickness Estimates, in Units of the Total Mean Free Path, for Infinite Slabs as a Function of c

c	V_4	Diffusion theory	Direct leakage operator with diffusion theory scalar flux	P_3^4	P_5^4	S_2^4	S_4^4
1.6	.5120	.7253	.5167	.5590	.5299	.6197	.5223
1.4	.7366	.9255	.7450	.7793	.7501	.8455	.7435
1.2	1.2894	1.4355	1.3084	1.3185	1.2985	1.3896	1.2949
1.1	2.1134	2.2219	2.1490	2.1354	2.1210	2.1984	2.1198
1.05	3.3005	3.3791	3.3586	3.3191	3.3073	3.3718	3.3078
1.02	5.6661	5.7162	5.7652	5.6828	5.6723	5.7262	5.6747

Table X

Critical Radii Estimates, in Units of the Total Mean Free Path, for Spheres as a Function of c

c	V_4	Diffusion theory	Direct leakage operator with diffusion theory scalar flux	$P_3{}^4$	SW^4	$S_2{}^4$	$S_4{}^4$
1.6	1.4761	1.8961	1.4948	1.5502	1.4152	1.4197	1.4613
1.4	1.9854	2.3595	2.0125	2.0394	1.9059	1.9198	1.9685
1.2	3.1723	3.4634	3.2202	3.2041	3.0570	3.0963	3.1530
1.1	4.8733	5.0897	4.9509	4.8953	4.7213	4.7960	4.8535
1.05	7.2784	7.4349	7.3970	7.2961	7.0930	7.2084	7.2591
1.02	12.0305	12.1290	12.2272	12.0450	11.8090	11.9780	12.0120

Table XI

Critical Radii Estimates, in Units of the Total Mean Free Path, for Infinite Cylinders as a Function of c

c	V_4	Diffusion theory	Direct leakage operator with diffusion theory scalar flux	P_3^4	P_5^4	S_2^4	S_4^4
1.6	1.0209	1.3471	1.0327	---	---	1.0647	1.0230
1.4	1.3970	1.6871	1.4152	---	---	1.4411	1.3958
1.2	2.2873	2.5124	2.3217	---	---	2.3297	2.2824
1.1	3.5776	3.7449	3.6355	---	---	3.6211	3.5725
1.05	5.4119	5.5330	5.5016	---	---	5.4654	5.4097
1.02	9.0445	9.1219	9.1939	---	---	9.1351	9.0495

IV. INDIRECT LEAKAGE OPERATOR

A. Introductory Remarks

As noted previously, the chief difference between the direct and indirect leakage operators is that the indirect leakage operator uses a fission source distribution for initiation and handles internally the transport of all neutrons which eventually leak out. The result of an indirect leakage calculation is a spatial leakage probability distribution for neutrons born in fission. It can be expected that the indirect leakage calculation will yield different critical size estimates than the V_n techniques even though the scalar fluxes are the same. This, of course, is due to the internal transport of neutrons and the numerical solution of the equation involved.

The kernel of the inhomogeneous singular Fredholm integral equation of the second kind, whose solution is the spatial leakage probability distribution, is shown to be identical to the kernel of the integral transport equation for each geometry. In order to emphasize that PC eigenvalues are all the result of a neutron balance condition, an eigenvalue different from (3.1) or (3.2) will be used. Take as the neutron balance condition, production rate minus leakage rate equals absorption rate. This leads to the eigenvalue defined by

$$A = \frac{\text{production rate - leakage rate}}{\text{absorption rate}} \tag{4.1}$$

In what follows, assume steady state, monoenergetic neutrons, isotropic fission, isotropic scattering, spatially constant cross sections, and no external sources. Also note that σ_t is normalized to unity, and that dimensions of length are in units of the total mean free path. The indirect leakage operator, as presented in this chapter, is the volume integral of the spatial leakage probability distribution and operates upon the spatial fission source distribution.

B. Indirect Leakage Operation in Slab Geometry

The following definitions and identities are repeated from Chapter Three:

$P_n^r(x)$ = the probability that a neutron emitted isotropically at position x escapes through the right hand side with n additional scatterings inside; (4.2)

$P_n^l(x)$ = the probability that a neutron emitted isotropically at position x escapes through the left hand side with n additional scatterings inside; (4.3)

$$P_0^r(x) = \int_0^1 \frac{d\mu}{2} e^{-\frac{(b-x)}{\mu}} = \frac{1}{2} E_2(b-x); \tag{4.4}$$

$$P_0^l(x) = \int_{-1}^0 \frac{d\mu}{2} e^{+\frac{(b+x)}{\mu}} = \frac{1}{2} E_2(b+x). \tag{4.5}$$

Now define:

$$P^r(x) = \sum_{n=0}^{\infty} P_n^r(x) = \text{the total probability that a neutron emitted isotropically at position x escapes through the right hand side;} \quad (4.6)$$

$$P^l(x) = \sum_{n=0}^{\infty} P_n^l(x) = \text{the total probability that a neutron emitted isotropically at position x escapes through the left hand side.} \quad (4.7)$$

Obviously $(P^l(x)+P^r(x))$ is the spatial leakage probability distribution for neutrons born in fission. From Chapter Three, $\frac{d\mu}{2}$ is the probability that a neutron emitted isotropically ends up within $d\mu$ of μ.

Note that

$$e^{\left(\frac{x-x'}{\mu}\right)} = \text{the probability that a neutron leaving x arrives at x', where } x' < x \text{ (i.e., } \mu<0), \quad (4.8)$$

$$e^{-\left(\frac{x'-x}{\mu}\right)} = \text{the probability that a neutron leaving x arrives at x', where } x' > x \text{ (i.e., } \mu>0), \quad (4.9)$$

and

$$\frac{dx'\sigma_s}{|\mu|} = \text{the probability that a neutron suffers a scattering collision in } dx' \text{ about } x'. \quad (4.10)$$

With (4.8), (4.9), (4.10), and (4.2), the following identity is obvious:

$$P_n^r(x) = \int_{-b}^{x} \int_{-1}^{0} \frac{d\mu}{2} e^{\left(\frac{x-x'}{\mu}\right)} \frac{dx'\sigma_s}{-\mu} P_{n-1}^r(x')$$

$$+ \int_{x}^{b} \int_{0}^{1} \frac{d\mu}{2} e^{-\left(\frac{x'-x}{\mu}\right)} \frac{dx'\sigma_s}{\mu} P_{n-1}^r(x'). \qquad (4.11)$$

The first integral term in (4.11) is the integral over neutrons commencing at positon x which initially head left, scatter at x', and then leak through the right hand side with (n-1) more interactions inside. The second integral term in (4.11) is the integral over neutrons commencing at position x which initially head right, scatter at x', and then leak through the right hand side with (n-1) more interactions inside. Replacing μ with $-\mu$ in the first integral term of (4.11), and collecting terms, results in

$$P_n^r(x) = \frac{\sigma_s}{2} \int_{-b}^{b} dx' \int_{0}^{1} \frac{d\mu}{\mu} e^{-\frac{|x-x'|}{\mu}} P_{n-1}^r(x'), \qquad (4.12)$$

or, after using (A.2),

$$P_n^r(x) = \frac{\sigma_s}{2} \int_{-b}^{b} dx' E_1(|x-x'|) P_{n-1}^r(x'). \qquad (4.13)$$

Equation (4.13) is a recursion relation for the $P_n^r(x)$ and, since $P_0^r(x)$ is already known, $P^r(x)$ can be constructed through the repeated application of (4.13). It will be more convenient to have an equation for $P^r(x)$, and this can be obtained by operating upon (4.13) with

$$P_0^r(x) + \sum_{n=1}^{\infty} \quad . \tag{4.14}$$

Interchange the order of integration-summation and use (4.6) to obtain

$$P^r(x) = P_0^r(x) + \frac{\sigma_s}{2} \int_{-b}^{b} dx' E_1(|x-x'|) P^r(x'). \tag{4.15}$$

The equation for $P^l(x)$ is similarly obtained:

$$P^l(x) = P_0^l(x) + \frac{\sigma_s}{2} \int_{-b}^{b} dx' E_1(|x-x'|) P^l(x'). \tag{4.16}$$

Note that the kernel in (4.15) and (4.16) is the same as in (2.1), the integral transport equation for slab geometry. Solution of (4.15) and (4.16) by analytic means has not proved possible, and, therefore, the following numerical scheme was employed. Note that, due to physical symmetry, $P^r(x)$ is equal to $P^l(-x)$, and that only one of (4.15) and (4.16) need be solved. Place N points on the interval from -b to b with a point on each boundary. Equally space the points such that each point is associated with an equal volume, namely Δx_n, except the boundary points, which are associated with half size volumes (i.e. $\Delta x_1 = \Delta x_N = \frac{1}{2}\Delta x_n$). Discretize (4.15) as follows:

$$P^r(x_i) = P_0^r(x_i) + \frac{\sigma_s}{2} \sum_{n=1}^{i-1} \Delta x_n P^r(x_n) E_1(|x_i - x_n|)$$

$$+ ST + \frac{\sigma_s}{2} \sum_{n=i+1}^{N} \Delta x_n P^r(x_n) E_1(|x_i - x_n|), \tag{4.17}$$

where ST in (4.17) corresponds to $x = x'$ in (4.15) and is the point where the singularity in the integrand occurs.

$$ST \simeq \int_{x_i - \frac{\Delta x_i}{2}}^{x_i + \frac{\Delta x_i}{2}} dx' \frac{\sigma_s}{2} P^r(x') E_1(|x_i - x'|). \tag{4.18}$$

If $P^r(x')$ is assumed smooth and slowly varying, and Δx_i is small, then

$$ST \simeq \frac{\sigma_s}{2} P^r(x_i) \int_{x_i - \frac{\Delta x_i}{2}}^{x_i + \frac{\Delta x_i}{2}} dx' E_1(|x_i - x'|). \tag{4.19}$$

Now use (A.2), interchange the order of integration, integrate over x', and integrate over μ, again using (A.2), to obtain

$$ST \simeq \sigma_s P^r(x_i)\left[1 - E_2\left(\frac{\Delta x_i}{2}\right)\right]. \tag{4.20}$$

Rewrite (4.20), using (A.3), the recursion formula for exponential integrals, as

$$ST \simeq \sigma_s P^r(x_i)\left[1 - e^{-\frac{\Delta x}{2}} + \frac{\Delta x}{2} E_1\left(\frac{\Delta x}{2}\right)\right]\Gamma. \tag{4.21}$$

In (4.21), Δx is a full sized volume even when i is a boundary point. The variable, Γ, takes on the value one half on the boundaries and one elsewhere. Then it follows that

$$P^r(x_i) = P_0^r(x_i) + \frac{\sigma_s}{2} \sum_{n=1}^{i-1} \Delta x_n P^r(x_n) E_1(|x_i - x_n|)$$

$$+ \frac{\sigma_s}{2} \sum_{n=i+1}^{N} \Delta x_n P^r(x_n) E_1(|x_i - x_n|) + \sigma_s P^r(x_i)$$

$$\left[1 - e^{-\frac{\Delta x}{2}} + \frac{\Delta x}{2} E_1\left(\frac{\Delta x}{2}\right)\right] \Gamma. \tag{4.22}$$

Solve (4.22) by guessing $P^r(x_i)$ as $P_0^r(x_i)$ and using (4.22) to generate a new $P^r(x_i)$. Use this new $P^r(x_i)$ in (4.22) to obtain another $P^r(x_i)$. Stop the iteration when the $P^r(x_i)$ have converged to some stable values. The leakage term for use in (4.1) is

$$\text{leak.} = \nu\sigma_f \int_{-b}^{b} dx \Phi(x) \left(P^l(x) + P^r(x)\right) \tag{4.23}$$

or

$$\text{leak.} = \nu\sigma_f \sum_{n=1}^{N} \Delta x_n \left(1 - a_1 x_n^2 - a_2 x_n^4\right) \left(P^r(x_n) + P^l(x_n)\right). \tag{4.24}$$

The production term for use in (4.1) is

$$\text{prod.} = \nu\sigma_f \left(2b - \frac{2}{3} a_1 b^3 - \frac{2}{5} a_1 b^5\right). \tag{4.25}$$

The absorption term for use in (4.1) is

$$\text{absorp.} = \sigma_a(2b - \frac{2}{3} a_1 b^3 - \frac{2}{5} a_1 b^5). \qquad (4.26)$$

Iterate (4.1) upon the size b until the eigenvalue A becomes unity. Note that the scalar fluxes used in (4.24), (4.25), and (4.26) are V_n scalar fluxes. To run the test problems in Table XII, σ_s was taken as 0.4 of the total cross section. A different value of σ_s would produce slightly different values of critical half-thickness (see, for example, Section C). Five hundred spatial points were used. Note that the critical half-thicknesses predicted through the use of the indirect leakage operator differ by no more than a tenth of a percent from the V_n calculations. More comparison values will be found in Chapter Three, Table VI, and Chapter Two, Table II.

Table XII

Critical Half-Thickness Estimates, in Units of the Total Mean Free Path, for Infinite Slabs as a Function of c

c	V_2	V_4	Indirect leakage with V_2 scalar flux	Indirect leakage with V_4 scalar flux	S_{16}[4]	Case's[10]	IT_4[11]
1.6	.5120	.5120	.5119	.5119	.5125	.5120	.5120
1.4	.7366	.7366	.7365	.7365	.7372	.7366	.7366
1.2	1.2894	1.2894	1.2891	1.2891	1.2902	1.2894	1.2894
1.1	2.1134	2.1134	2.1128	2.1127	2.1146	2.1133	2.1134
1.05	3.3010	3.3005	3.3010	3.2988	3.3023	3.3003	3.3004
1.02	5.6706	5.6661	5.6761	5.6610	5.6644	5.6655	5.6660

C. Indirect Leakage Operation in Spherical Geometry

Recall the following definitions and identities from Chapter Three:

$P_n(r)$ = the probability that a neutron emitted isotropically at position r escapes the system with n additional scatterings inside; (4.27)

$$P_0(r) = \frac{1}{4r}[(b+r)E_2(b-r)-(b-r)E_2(b+r)- e^{-(b+r)} + e^{-(b-r)}]. \quad (4.28)$$

Now define

$$P(r) = \sum_{n=0}^{\infty} P_n(r) = \text{the total probability that a neutron emitted isotropically at position r escapes.} \quad (4.29)$$

Obviously, P(r) is the spatial leakage probability distribution for neutrons born in fission. Also from Chapter Three, it is known that $\frac{d\mu}{2}$ is the probability that a neutron emitted isotropically ends up within $d\mu$ of μ. The distance from a spherical shell at r to another spherical shell at r′, r′>r, is

$$d_{r'>r} = -r\mu+(r'^2-r^2+r^2\mu^2)^{.5}. \quad (4.30)$$

This is (3.45) modified by replacing b with r′.

To complete the derivation of an equation for $P_n(r)$, the corresponding $d_{r'<r}$ will be needed. Fig. 7 is a drawing of a plane containing the point of emmission, the geometrical center of the sphere,

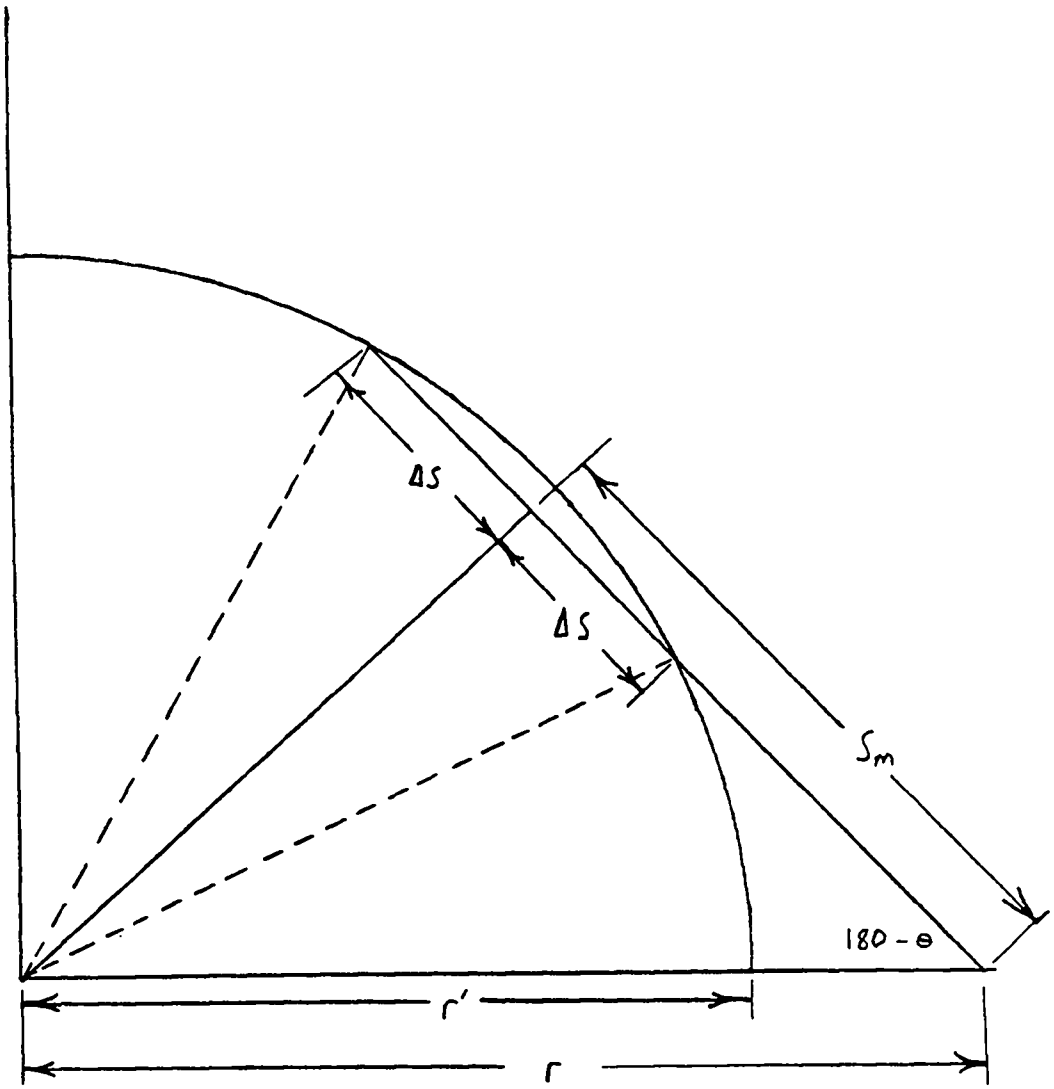


Figure 7.

Geometry for inward traveling neutron

in spherical systems.

and the direction vector along which the neutron is traveling. Note that for a neutron going from r to r′, r′<r, there are two places where the neutron can interact within dr′ of r′, namely, when entering and when leaving the spherical volume defined by r′. From Fig. 7, the following identities are obvious:

$$S_m = r\cos(180-\theta) = -r\mu, \tag{4.31}$$

$$\Delta S = [r'^2-(r\sin(180-\theta))^2]^{.5} = (r'^2-r^2+r^2\mu^2)^{.5}. \tag{4.32}$$

Therefore, the distance from r to r′, if r′<r, is

$$d_{r'<r} = -r\mu\pm(r'^2-r^2+r^2\mu^2)^{.5}. \tag{4.33}$$

It is possible for a neutron leaving r to hit a spherical shell at r′>r for all angles of emission. However, it can be seen from Fig. 7 that only angles greater than a certain limit can hit a spherical shell at r′<r, viz.

$$r\sin(180-\theta) < r', \tag{4.34}$$

which yields

$$\mu < -\left(\frac{r^2-r'^2}{r^2}\right)^{.5}. \tag{4.35}$$

The remaining detail needed to complete the derivation of $P_n(r)$ is the probability that a neutron reaching r′ scatters within dr′ of r′. The distance traveled along μ while traversing dr′ is multiplied by the probability per unit path length that the neutron scatters, namely,

$$\text{probability of scattering within dr′ of r′} = \frac{\sigma_s r'dr'}{(r'^2-r^2+r^2\mu^2)^{.5}} . \tag{4.36}$$

Now $P_n(r)$ can be written as

$$P_n(r) = \int_r^b \int_{-1}^{1} \frac{d\mu}{2} e^{[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}]} \frac{\sigma_s r'dr'}{(r'^2-r^2+r^2\mu^2)^{.5}} P_{n-1}(r')$$

$$+\int_0^r \int_{-1}^{-(\frac{r^2-r'^2}{r^2})^{.5}} \frac{d\mu}{2} e^{[r\mu+(r'^2-r^2+r^2\mu^2)^{.5}]}$$

$$\frac{\sigma_s r'dr'}{(r'^2-r^2+r^2\mu^2)^{.5}} P_{n-1}(r') + \int_0^r \int_{-1}^{-(\frac{r^2-r'^2}{r^2})^{.5}}$$

$$\frac{d\mu}{2} e^{[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}]} \frac{\sigma_s r'dr'}{(r'^2-r^2+r^2\mu^2)^{.5}} P_{n-1}(r') . \tag{4.37}$$

The first integral term in (4.37) is the integral over all possible occurences that the neutron leaves r within dμ of μ, travels to r′(r′>r) from r without incident, scatters within dr′ of r′, and then leaks out of the system with (n-1) more scatterings inside. The second integral term is identical except that it accounts for neutrons which interact upon reaching the spherical shell at r′<r. The third term is also

identical except that it accounts for neutrons which interact in the spherical shell at r'<r after having penetrated that shell once. Upon rearranging terms, (4.37) becomes

$$P_n(r) = \frac{\sigma_s}{2}\Big[\int_r^b dr' r' P_{n-1}(r') \int_{-1}^{1} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}}$$

$$e^{\left[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}\right]} + \int_0^r dr' r' P_{n-1}(r')$$

$$\int_{-1}^{-\left(\frac{r^2-r'^2}{r^2}\right)^{.5}} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}} e^{\left[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}\right]}$$

$$+ \int_0^r dr' r' P_{n-1}(r') \int_{-1}^{-\left(\frac{r^2-r'^2}{r^2}\right)^{.5}} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}}$$

$$e^{\left[r\mu+(r'^2-r^2+r^2\mu^2)^{.5}\right]}\Big] \quad . \tag{4.38}$$

To do the integration over μ in the first term of (4.38), make the transformation

$$a = (r'^2-r^2+r^2\mu^2)^{.5} - r\mu, \tag{4.39a}$$

where

$$\mu = \frac{(r'^2-r^2-a^2)}{2ar}, \tag{4.39b}$$

and

$$d\mu = -\frac{da}{2r}\left(\frac{r'^2-r^2+a^2}{a^2}\right), \tag{4.39c}$$

to get

$$\int_{-1}^{1} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}} e^{[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}]}$$

$$= \int_{(r'-r)}^{(r'+r)} \frac{da}{ra} e^{-a} = \frac{1}{r}[E_1(r'-r)-E_1(r+r')]. \tag{4.40}$$

The exponential integral functions in (4.40) come from application of (A.3). The same transformation, (4.39), is made in order to perform the integration over μ in the second term of (4.38). It is found that

$$\int_{-1}^{-\left(\frac{r^2-r'^2}{r^2}\right)^{.5}} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}} e^{[r\mu-(r'^2-r^2+r^2\mu^2)^{.5}]} =$$

$$\int_{(r^2-r'^2)^{.5}}^{(r+r')} \frac{da}{ra} e^{-a} = \frac{1}{r}[E_1((r^2-r'^2)^{.5}) - E_1(r+r')]. \tag{4.41}$$

In order to perform the integration over μ in the last term of (4.38), use the following transformation,

$$a = -r\mu-(r'^2-r^2+r^2\mu^2)^{.5}, \tag{4.42a}$$

where

$$\mu = \frac{(r'^2-r^2-a^2)}{2ar}, \tag{4.42b}$$

and

$$d\mu = -\frac{da}{2r}\left(\frac{r'^2-r^2+a^2}{a^2}\right), \tag{4.42c}$$

to get

$$-\int_{-1}^{-\left(\frac{r^2-r'^2}{r^2}\right)^{.5}} \frac{d\mu}{(r'^2-r^2+r^2\mu^2)^{.5}}\, e^{[r\mu+(r'^2-r^2+r^2\mu^2)^{.5}]}$$

$$\int_{(r-r')}^{(r^2-r'^2)^{.5}} \frac{da}{ra}\, e^{-a} = \frac{1}{r}[E_1(r-r') - E_1((r^2-r'^2)^{.5})]. \tag{4.43}$$

Substitute (4.40), (4.41), and (4.43) in (4.38), rearrange terms, and multiply through by r; then

$$rP_n(r) = \frac{\sigma_s}{2}\int_0^b dr'\,[E_1(|r-r'|)-E_1(r+r')]r'P_{n-1}(r'). \tag{4.44}$$

Equation (4.44) can be simplified if the following conventions are made: $r \geq 0$, r' in the interval of $-b$ to b, $P_n(-r') = P_n(r')$. Rewrite the second term in the integrand of (4.44) by making the transformation $r' = -r'$:

$$-\frac{\sigma_s}{2}\int_0^b dr'E_1(r+r')r'P_{n-1}(r')$$

$$= -\frac{\sigma_s}{2}\int_0^{-b} dr'E_1(r-r')r'P_{n-1}(-r')$$

$$= \frac{\sigma_s}{2} \int_{-b}^{0} dr' E_1(|r-r'|) r' P_{n-1}(r'). \tag{4.45}$$

Using (4.45), (4.44) becomes

$$rP_n(r) = \frac{\sigma_s}{2} \int_{-b}^{b} dr' E_1(|r-r'|) r' P_{n-1}(r'). \tag{4.46}$$

Operate on both sides of (4.46) with $rP_0(r) + \sum_{n=1}^{\infty}$, interchange the order of integration - summation, and use (4.29), to obtain

$$rP(r) = rP_0(r) + \frac{\sigma_s}{2} \int_{-b}^{b} dr' E_1(|r-r'|) r' P(r'). \tag{4.47}$$

Equation (4.47) is the same as (4.16) if P(x) is replaced by rP(r). For criticality estimates, a solution for rP(r) is just as useful as a solution for P(r), due to the $4\pi r^2 dr$ volume term. The discretized equation analogous to (4.22) is

$$r_i P(r_i) = r_i P_0(r_i) + \frac{\sigma_s}{2} \sum_{n=1}^{i-1} \Delta r_n E_1(|r_i - r_n|) r_n P(r_n)$$

$$+ \frac{\sigma_s}{2} \sum_{n=i+1}^{N} \Delta r_n E_1(|r_i - r_n|) r_n P(r_n)$$

$$+ \sigma_s r_i P(r_i) \left[1 - e^{-\frac{\Delta r}{2}} + \frac{\Delta r}{2} E_1\left(\frac{\Delta r}{2}\right)\right] \Gamma, \tag{4.48}$$

where the r_i are spaced equally by distance. The solution of (4.48) is identical to that of (4.22) except that: $r_i P(r_i) = -r_{N+1-i} P(r_{N+1-i})$ for i in the interval 1 to $(\frac{N-1}{2})$, N must be odd, and only the information stored in the interval $(\frac{N+1}{2})$ to N is relevant. These restrictions are a consequence of the assumptions made to reduce (4.44) to (4.46). The leakage rate for use in (4.1) is

$$\text{leak.} = \nu\sigma_f \sum_{i=\frac{N+1}{2}}^{N} 4\pi r_i \Delta r_i (1-a_1 r_i^2 - a_2 r_i^4) r_i P(r_i). \tag{4.49}$$

The production rate for use in (4.1) is

$$\text{prod.} = 4\pi\nu\sigma_f \left(\frac{b^3}{3} - a_1\frac{b^5}{5} - a_2\frac{b^7}{7}\right). \tag{4.50}$$

The absorption rate for use in (4.1) is

$$\text{absorp.} = 4\pi\sigma_a \left(\frac{b^3}{3} - a_1\frac{b^5}{5} - a_2\frac{b^7}{7}\right). \tag{4.51}$$

Iterate (4.1) upon the size b until the eigenvalue A becomes unity. Note that the scalar fluxes used in (4.49), (4.50), and (4.51), are V_n scalar fluxes. To run the test problems in Table XIII, σ_s was taken to be 0.2, 0.4, and 0.8 of the total cross section. The critical size, in the problems solved here, should depend only upon the value of c. There is, however, some dependence upon the ratio $\frac{\sigma_s}{\sigma_t}$ in the indirect

calculation. This is probably due to the fact that the size of the numerical mesh becomes less adequate as the scattering transport increases (i.e., as the size increases or the ratio of $\frac{\sigma^s}{\sigma_t}$ increases). Due to the internal transport, the critical radii predicted by the indirect leakage calculations differ from the V_n calculations. This difference is in no case greater than one half of a percent. One thousand and one spatial points were used, and this resulted in 501 useful data points. As the number of points was increased, the answers monotonically approach the V_n values. As the value of σ_s was increased, the answers decreased (errors increased). This indicates that more spatial resolution than 501 points is needed for large values of σ_s. More comparison values will be found in Chapter Three, Table VII, and Chapter Two, Table III.

Table XIII

Critical Radii Estimates, in Units of the

Total Mean Free Path, for Spheres as a Function of c

c	V_2	V_4	Indirect leakage, V_2 scalar flux, $\sigma_s = .4\sigma_t$	Indirect leakage, V_4 scalar flux, $\sigma_s = .2\sigma_t$	Indirect leakage, V_4 scalar flux, $\sigma_s = .4\sigma_t$	Indirect leakage, V_4 scalar flux, $\sigma_s = .8\sigma_t$	Case's[10]
1.6	1.4768	1.4761	1.4768	1.4760	1.4759	1.4746	1.4761
1.4	1.9870	1.9854	1.9873	1.9853	1.9850	1.9830	1.9853
1.2	3.1785	3.1723	3.1810	3.1719	3.1713	3.1675	3.1721
1.1	4.8934	4.8733	4.9039	4.8723	4.8709	4.8632	4.8727
1.05	7.3326	7.2784	7.3641	7.2759	7.2725	7.2570	7.2772
1.02	12.1835	12.0305	12.2778	12.0226	12.0118	11.9695	12.0275

D. Indirect Leakage Operation in Cylindrical Geometry

Recall the following definitions and identities from Chapter Three;

$P_n(r)$ = the probability that a neutron emitted isotropically at position r escapes the system with n additional scatterings inside; (4.52)

$$P_0(r) = b\int_1^\infty \frac{dx}{x} K_1(bx)I_0(rx). \tag{4.53}$$

Now define

$P(r) = \sum_{n=0}^{\infty} P_n(r)$ = the total probability that a neutron emitted isotropically at position r escapes. (4.54)

The quantity, P(r), obviously, is the spatial leakage probability distribution for neutrons born in fission. Also, from Chapter Three, it is known that $\frac{d\mu}{2}\frac{d\chi}{2\pi}$ is the probability that a neutron emitted isotropically ends up within $d\mu$ of μ and within $d\chi$ of χ. The distance from a cylindrical shell at r to another cylindrical shell at r′, r′>r, is

$$d_{r'>r} = \frac{-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{.5}}{(1-\mu^2)^{.5}} \quad . \tag{4.55}$$

This is (3.69) modified by replacing b with r′.

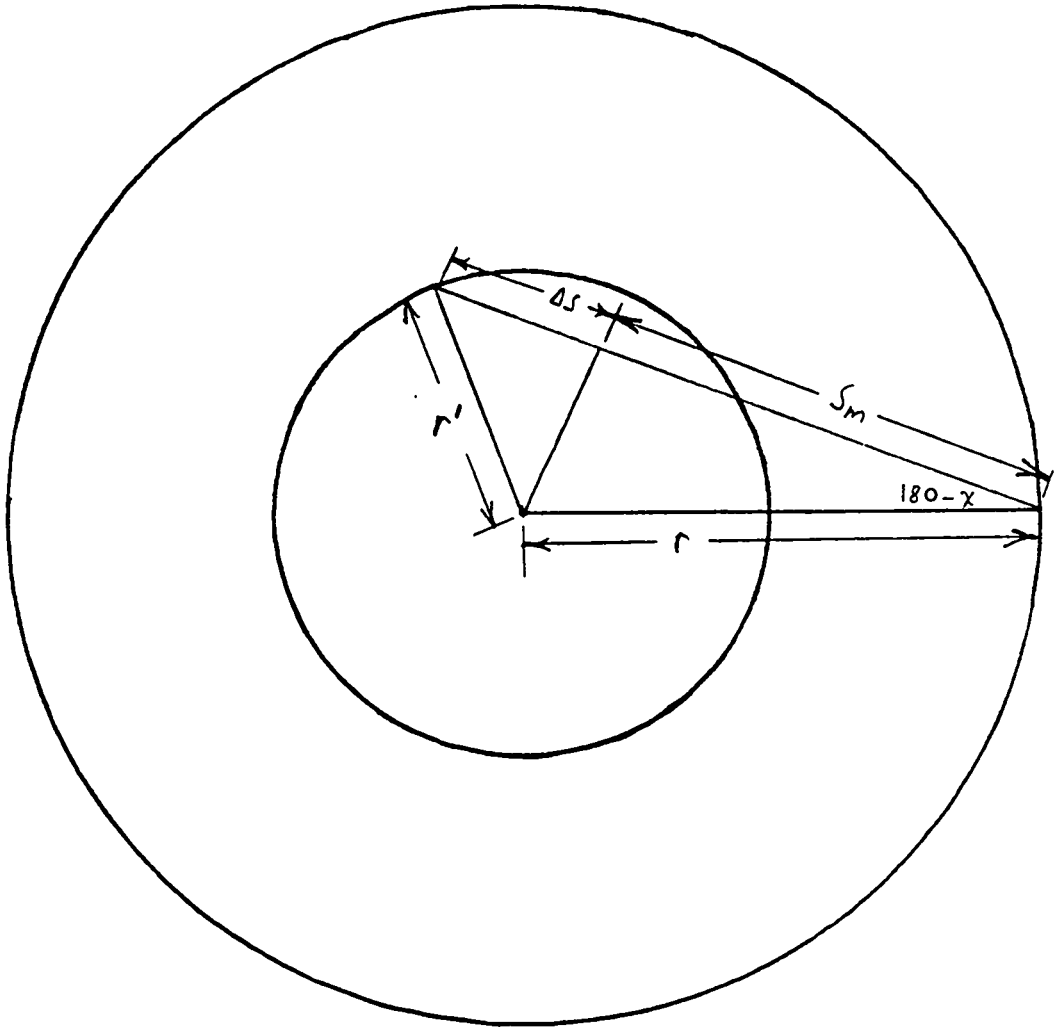


Figure 8.

Geometry for inward traveling neutrons

in cylindrical systems.

To complete the derivation of an equation for $P_n(r)$, the corresponding $d_{r'<r}$ will be required. Fig. 8 is a drawing projecting the geometry of a one-dimensional cylinder upon a plane perpendicular to the axis. As in spherical geometry, a neutron going from r to r′, r′<r, can interact within dr′ of r′ in two places, namely when entering and leaving the cylindrical volume defined by r′. From Fig. 8, the following identities are obvious:

$$S_m = r\cos(180-\chi) = -r\cos(\chi); \tag{4.56}$$

$$\Delta S = [r'^2-(r\sin(180-\chi))^2]^{.5} , \tag{4.57a}$$

$$\Delta S = (r'^2-r^2\sin^2(\chi))^{.5} . \tag{4.57b}$$

Therefore, the distance from r to r′, r′<r, is

$$d_{r'<r} = \frac{-r\cos(\chi)\pm(r'^2-r^2\sin^2(\chi))^{.5}}{(1-\mu^2)^{.5}} , \tag{4.58}$$

where the denominator in (4.58) accounts for the z coordinate. For all angles of emission, a neutron can hit a cylindrical shell of greater radius than that of the point of emission. However, from Fig. 8, it can be seen that only angles in χ within certain limits can hit a cylindrical shell at r′<r, viz.

$$r \sin(\pm(180-\chi)) < r', \tag{4.59}$$

which yields

$$\pi-\arcsin\left(\frac{r'}{r}\right) < \chi < \pi+\arcsin\left(\frac{r'}{r}\right). \tag{4.60}$$

For the derivation of $P_n(r)$, one more detail is needed, namely, the probability that a neutron reaching r′ scatters within dr′ of r′. The distance traveled along μ and χ while traversing dr′ is multiplied by the probability per unit path length that the neutron scatters, namely,

$$\text{probability of scattering within } dr' \text{ of } r' = \frac{\sigma_s r'dr'}{(r'^2-r^2\sin^2(\chi))^{\cdot 5}(1-\mu^2)^{\cdot 5}}. \tag{4.61}$$

With (4.52), (4.55), (4.58), (4.60), and (4.61),

$$P_n(r) = \int_r^b \int_{-1}^{1} \int_0^{2\pi} \frac{d\mu}{2}\frac{d\chi}{2\pi}$$

$$e^{\left[\frac{r\cos(\chi)-(r'^2-r^2\sin^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}}\right]} \frac{\sigma_s r'dr'P_{n-1}(r')}{(r'^2-r^2\sin^2(\chi))^{\cdot 5}(1-\mu^2)^{\cdot 5}}$$

$$+ \int_0^r \int_{-1}^{1} \int_{\pi-}^{\pi+} \frac{d\mu}{2}\frac{d\chi}{2\pi} e^{\left[\frac{r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}}\right]}$$

$$\frac{\sigma_s r'dr'P_{n-1}(r')}{(r'^2-r^2\sin^2(\chi))^{\cdot 5}(1-\mu^2)^{\cdot 5}} + \int_0^r \int_{-1}^{1} \int_{\pi-}^{\pi+} \frac{d\mu}{2}\frac{d\chi}{2\pi}$$

$$e^{\left[\frac{r\cos(\chi)-(r'^2-r^2\sin^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}}\right]}$$

$$\frac{\sigma_s r' dr' P_{n-1}(r')}{(r'^2 - r^2\sin^2(\chi))^{.5}(1-\mu^2)^{.5}}, \tag{4.62}$$

where $\pi+$ and $\pi-$ come from (4.60). The first integral term in (4.62) is the integral over all possible occurences that the neutron leaves r within $d\mu$ of μ, within $d\chi$ of χ, travels to r' ($r'>r$) from r without incident, scatters within dr' of r' and then leaks out of the system with (n-1) more scatterings inside. The second integral term is identical except that it accounts for neutrons which interact upon reaching the cylindrical shell at $r'<r$. The third term is also identical, except that it accounts for neutrons which interact in the cylindrical shell at $r'<r$ after having penetrated that shell once. Upon rearranging terms and using symmetry in μ and χ,

$$P_n(r) = \frac{\sigma_s}{\pi}\left[\int_r^b dr' r' P_{n-1}(r') \int_0^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{.5}}\right.$$

$$\int_0^1 \frac{d\mu}{(1-\mu^2)^{.5}} e^{\left[\frac{r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}}{(1-\mu^2)^{.5}}\right]}$$

$$\left. + \int_0^r dr' r' P_{n-1}(r') \int_{\pi-}^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{.5}}\right.$$

$$\left.\int_0^1 \frac{d\mu}{(1-\mu^2)^{.5}} e^{\left[\frac{r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}}{(1-\mu^2)^{.5}}\right]}\right]$$

$$+ \int_0^r dr' r' P_{n-1}(r') \int_{\pi-}^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{\cdot 5}}$$

$$\int_0^1 \frac{d\mu}{(1-\mu^2)^{\cdot 5}} e^{\left[\frac{r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{\cdot 5}}{(1-\mu^2)^{\cdot 5}}\right]}]. \qquad (4.63)$$

Make the substitution $\mu = \frac{\sinh(a)}{\cosh(a)}$ in (4.63), and note that all integrals over μ take the form

$$\int_0^\infty \frac{da}{\cosh(a)} e^{-B\cosh(a)}. \qquad (4.64)$$

From Bickley and Naylor[40],

$$\int_0^\infty \frac{da}{\cosh(a)} e^{-B\cosh(a)} = K_{i1}(B) = \int_B^\infty K_0(t)dt. \qquad (4.65)$$

Use (4.65) with (4.63), operate upon the result with $P_0(r) + \sum_{n=1}^{\infty}$, and use (4.54) to obtain

$$P(r) = P_0(r) + \frac{\sigma_s}{\pi} [\int_r^b dr' r' P(r') \int_0^\pi \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{\cdot 5}}$$

$$\int^\infty \frac{K_0(t)dt}{[-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{\cdot 5}]} + \int_0^r dr' r' P(r')$$

$$\int_{\pi-}^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{\cdot 5}} \int^\infty \frac{K_0(t)dt}{[-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{\cdot 5}]} +$$

$$\int_0^r dr' r' P(r') \int_{\pi-}^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{.5}}$$

$$\int^{\infty} \frac{K_0(t)dt}{[-r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}]} \Big]. \qquad (4.66)$$

The reduction of (4.66) will now proceed. Make the transformation

$$t = y[-r\cos(\chi) + (r'^2 - r^2\sin^2(\chi))^{.5}] \qquad (4.67)$$

in

$$INT1 = \int_0^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{.5}} \int^{\infty} \frac{K_0(t)dt}{[-r\cos(\chi) + (r'^2 - r^2\sin^2(\chi))^{.5}]} \qquad (4.68)$$

to obtain

$$INT1 = \int_0^{\pi} \frac{d\chi(-r\cos(\chi) + (r'^2 - r^2\sin^2(\chi))^{.5})}{(r'^2 - r^2\sin^2(\chi))^{.5}}$$

$$\int_1^{\infty} dy K_0[y(-r\cos(\chi) + (r'^2 + r^2\sin^2(\chi))^{.5})]. \qquad (4.69)$$

This integral can be done by making a transformation from χ to Φ (see Fig. 5), which is, of course, a transformation from a local to a central coordinate system. Use (3.78) through (3.80), with r′ replacing b, in (4.69) and reduce to

$$INT1 = \int_0^{\pi} d\Phi \int_1^{\infty} dy\ K_0(y(r'^2 + r^2 - 2rr'\cos(\Phi))^{.5}). \qquad (4.70)$$

Interchange the order of integration, note that r'>r, and compare to (C.12), where the integral has been done, to obtain

$$INT1 = \pi\int_1^{\infty} dy K_0(yr')I_0(yr). \tag{4.71}$$

Make the transformation (4.67) in

$$INT2 = \int_{\pi-}^{\pi} \frac{d\chi}{(r'^2-r^2\sin^2(\chi))^{.5}} \int_{[-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{.5}]}^{\infty} K_0(t)dt \tag{4.72}$$

to obtain

$$INT2 = \int_{\pi-}^{\pi+} \frac{d\chi[-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{.5}]}{(r'^2-r^2\sin^2(\chi))^{.5}}$$

$$\int_1^{\infty} dy\ K_0[y(-r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{.5})]. \tag{4.73}$$

Equation (4.73) can be integrated if a transformation to a central coordinate system is used (i.e., from χ to Φ). From Fig. 9,

$$ZLD = -r\cos(\chi)+(r'^2-r^2\sin^2(\chi))^{.5}, \tag{4.74}$$

$$ZLD = (r'^2+r^2-2rr'\cos(\Phi))^{.5}, \tag{4.75}$$

$$r'\sin\Phi = ZLD\sin(\chi), \tag{4.76}$$

and

$$r'\cos\Phi = ZLD\cos(\chi)+r. \tag{4.77}$$

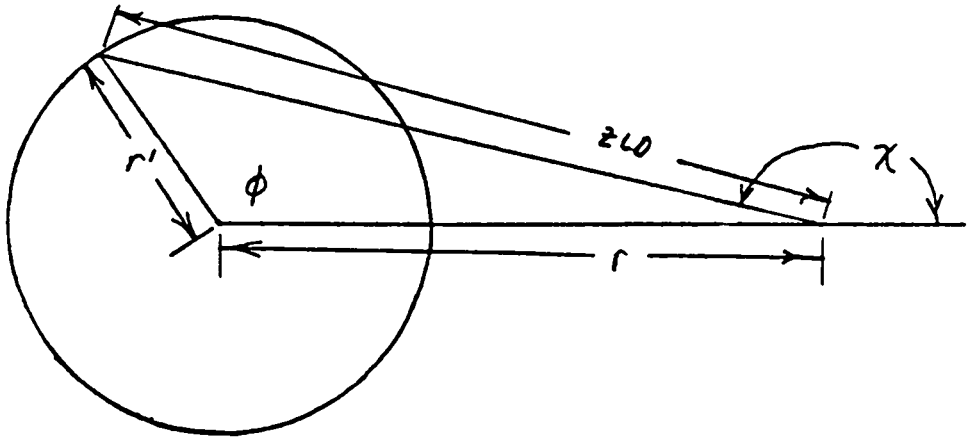


Figure 9.

Central vs local coordinates for use with INT2.

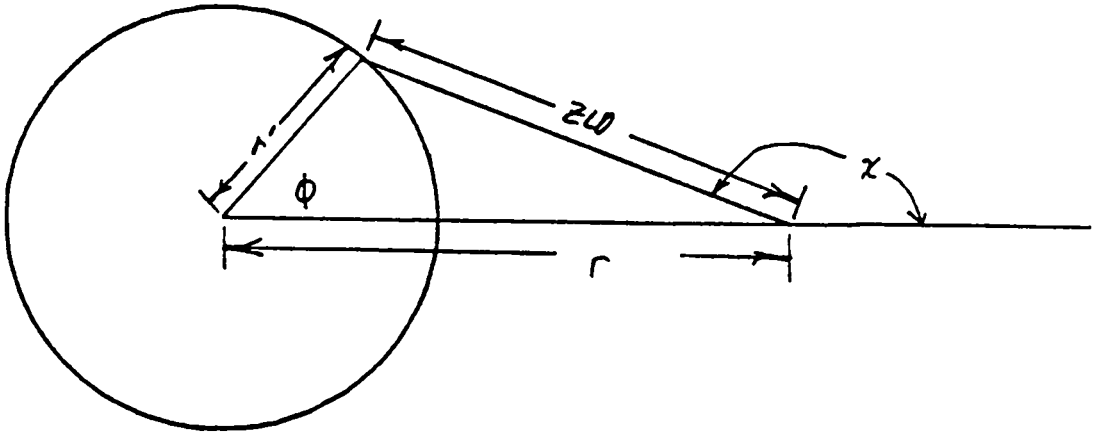


Figure 10.

Central vs local coordinates for use with INT3.

From (4.76) and (4.77), it is found that

$$d\chi = d\Phi \frac{(r'^2 - rr'\cos(\Phi))}{ZLD^2} . \tag{4.78}$$

Therefore, after transformation from χ to Φ and suitable reduction, (4.73) becomes

$$INT2 = \int_1^{\infty} dy \int_{\cos^{-1}(\frac{r'}{r})}^{\pi} d\Phi K_0[y(r'^2 + r^2 - 2rr'\cos(\Phi))^{.5}]. \tag{4.79}$$

Make the transformation

$$t = y[-r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}] . \tag{4.80}$$

in

$$INT3 = \int_{\pi-}^{\pi} \frac{d\chi}{(r'^2 - r^2\sin^2(\chi))^{.5}} \int_{[-r\cos(\chi)-(r'^2-r^2\sin^2(\chi))^{.5}]}^{\infty} K_0(t)dt \tag{4.81}$$

to obtain

$$INT3 = \int_{\pi-}^{\pi} \frac{d\chi[-r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}]}{(r'^2 - r^2\sin^2(\chi))^{.5}}$$

$$\int_1^{\infty} dy \; K_0[y(-r\cos(\chi) - (r'^2 - r^2\sin^2(\chi))^{.5}]. \tag{4.82}$$

Equation (4.82) can be integrated if a transformation to a central coordinate system is used. From Fig. 10,

$$ZLD = -r\cos(\chi)-(r'^2-r^2\sin^2(\chi))^{.5}, \tag{4.83}$$

$$ZLD = (r'^2+r^2-2rr'\cos\Phi), \tag{4.84}$$

$$r'\sin\Phi = ZLD\sin(\chi), \tag{4.85}$$

$$r'\cos\Phi = ZLD\cos(\chi)+r. \tag{4.86}$$

From (4.85) and (4.86), it is found that

$$d\chi = d\Phi \frac{(r'^2-rr'\cos(\Phi))}{ZLD^2}. \tag{4.87}$$

Therefore, after transformation from χ to Φ and suitable reduction, (4.73) becomes

$$INT3 = \int_1^{\infty} dy \int_0^{\cos^{-1}(\frac{r'}{r})} d\Phi K_0[y(r'^2+r^2-2rr'\cos(\Phi))^{.5}]. \tag{4.88}$$

It should be noted that in the steps leading to (4.88) a negative of a square root was used. This was required in order that (4.88) be positive and thus represent a contribution to the leakage probability as (4.88) physically should.

With (4.68), (4.71), (4.72), (4.79), (4.81) and (4.88), (4.66) becomes

$$P(r) = P_0(r)+\sigma_s \int_r^b dr'r'P(r') \int_1^{\infty} dy K_0(yr')I_0(yr)$$

$$+\frac{\sigma_s}{\pi}\int_0^r dr'r'P(r')\int_1^\infty dy\int_0^\pi d\Phi K_0[y(r'^2+r^2-2rr'\cos(\Phi))^{.5}]. \qquad (4.89)$$

The integral over Φ in (4.89), upon comparison with (C.12), and subject to the condition $r'<r$, is seen to be

$$\int_0^\pi d\Phi K_0[y(r'^2+r^2-2rr'\cos(\Phi))^{.5}] = \pi K_0(yr)I_0(yr'). \qquad (4.90)$$

Therefore, (4.89) reduces to

$$P(r) = P_0(r)+\sigma_s\int_0^b dr'r'P(r')K(r,r') \qquad (4.91)$$

where

$$K(r,r') = \int_1^\infty dy K_0(yr)I_0(yr'), \qquad r'<r; \qquad (4.92)$$

$$K(r,r') = \int_1^\infty dy K_0(yr')I_0(yr), \qquad r'>r. \qquad (4.93)$$

Equation (4.91) is an inhomogeneous Fredholm integral equation of the second kind with singular kernel. The kernel in (4.91) is the same as the kernel in the cylindrical integral transport equation. The solution of (4.91) proceeds numerically along the same lines as the solution of (4.22). Analogous to (4.22) is, with the r_i equidistant, r_1 in the center and r_N on boundary,

$$P(r_i) = P_0(r_i)+\sigma_s \sum_{n=1}^{n=i-1} \Delta r_n r_n P(r_n)K_{n<i}(r_i,r_n)$$

$$+\sigma_s \sum_{n=i+1}^{N} \Delta r_n r_n P(r_n) K_{n>i}(r_i, r_n)$$

$$+\sigma_s \Delta r_n r_n P(r_n) K_{n=i}(r_i, r_n), \tag{4.94}$$

where

$$K_{n<i}(r_i, r_n) = \int_1^\infty dy K_0(yr_i) I_0(yr_n), \tag{4.95}$$

$$K_{n>i}(r_i, r_n) = \int_1^\infty dy K_0(yr_n) I_0(yr_i), \tag{4.96}$$

and

$$K_{n=i}(r_i, r_n) = \int_1^\infty dy K_0(yr_i) I_0(yr_i). \tag{4.97}$$

The integrals in (4.95), (4.96), and (4.97) are done numerically. The leakage for use in (4.1) is

$$\text{leak.} = \nu\sigma_f \sum_{i=1}^{N} 2\pi r_i \Delta r_i (1 - a_1 r_i^2 - a_2 r_i^4) P(r_i). \tag{4.98}$$

The production for use in (4.1) is

$$\text{prod.} = 2\pi\nu\sigma_f \left(\frac{b^2}{2} - a_1 \frac{b^4}{4} - a_2 \frac{b^6}{6}\right). \tag{4.99}$$

The absorption for use in (4.1) is

$$\text{absorp.} = 2\pi\sigma_a \left(\frac{b^2}{2} - a_1 \frac{b^4}{4} - a_2 \frac{b^6}{6}\right). \tag{4.100}$$

Iterate (4.1) upon the size b until the eigenvalue A becomes unity. Note that the scalar fluxes used in (4.98), (4.99), and (4.100) are V_n scalar fluxes. To run the test problems in Table XIV, σ_s was taken to be 0.4 of the total cross section. A different value of σ_s would produce slightly different values of the critical radii. Five hundred spatial points were used. The error, from V_n fluxes, varies from 0.5% to 3.0%. Since the indirect leakage answers monotonically approached the V_n answers as the number of points was increased, it is assumed that the indirect leakage calculations showed the large error due to poor spatial resolution. This does not necessarily mean that the answers would coincide. In fact, due to internal transport in the indirect leakage approach, the author does not believe they would. The answers were not posted to greater precision due to the inordinately large running times associated with the 500 point codes, as the running time is proportional to the number of points squared.

Table XIV

Critical Radii Estimates, in Units of the

Total Mean Free Path, for Cylinders as a Function of c

c	V_2	V_4	Indirect leakage V_2 scalar flux	Indirect leakage V_4 scalar flux	S_8^4	S_{16}^{11}	IT_4^{11}
1.6	1.0209	1.0209	1.0266	1.0265	1.0194	1.0231	1.0209
1.4	1.3971	1.3970	1.4065	1.4062	1.3950	1.3991	1.3970
1.2	2.2882	2.2873	2.3097	2.3079	2.2850	2.2891	2.2872
1.1	3.5819	3.5776	3.6318	3.6227	3.5757	3.5795	3.5774
1.05	5.4265	5.4119	5.5422	5.5098	5.4108	5.4140	5.4115
1.02	9.0960	9.0445	9.4358	9.3173	9.0454	9.0494	9.0446

E. Components of the Indirect Leakage Operation in Spherical Geometry

An interesting aspect of the indirect leakage method of calculating criticality is the ability to calculate the spatial leakage probability distribution. Heretofore, this distribution has not been examined. Sections A, B, C, and D have been devoted to proving that the indirect leakage operation is a viable method of determining criticality. Now that the indirect leakage approach has been proven, it is desirable to examine the components of the indirect leakage operation. This investigation will be limited to spherical geometry since spheres, of the geometries so far considered, are the only finite geometries. Remember that $rP(r)$ is the continuous case, and $r_iP(r_i)$ is the discretized case. In what follows, $P(r_i)$ and its numerical Neumann series components, $P_n(r_i)$, will be determined and plotted. The number of terms in the Neumann series is determined by requiring 10^{-3} convergence of $\sum_n P_n(r_i)$ to $P(r_i)$ for all points r_i. The individual $P_n(r_i)$ is, of course, the spatial probability distribution that a neutron will scatter n times from its birth at r_i and then leak out of the sphere.

In Section C, the numerical solution for $r_iP(r_i)$ was achieved. Subsequent division of $r_iP(r_i)$ by r_i will yield the desired $P(r_i)$ except at r_i equal to zero. To determine $P(0)$, reformulate (4.47);

$$P(r) = P_0(r) + \frac{\sigma_s}{2r}\int_{-b}^{b} dr' E_1(|r-r'|)r'P(r'). \qquad (4.101)$$

Both terms on the right hand side of (4.101) are indeterminate at $r=0$.

Application of L'Hospital's rule to the inhomogeneous term in (4.101) yields

$$P_0(0) = e^{-b}. \tag{4.102}$$

Breaking the integration in (4.101) at $r'=r$ will remove the absolute value. Subsequently replace r' by $-r'$ in the integral over $r'<r$, use the fact that $P(r') = P(-r')$, and apply L'Hospital's rule to obtain

$$P(0) = e^{-b+\sigma_s \int_0^b dr' e^{-r'} P(r')}. \tag{4.103}$$

The numerical equation corresponding to (4.103) is

$$P(0) = e^{-b+\sigma_s \sum_{k=\frac{N+1}{2}}^{N} \Delta r_k e^{-r_k} P(r_k)}. \tag{4.104}$$

Since P(0) is the only unknown term in (4.104), P(0) is guessed and iterated until (4.104) is solved. $P_0(r_i)$ is already known for all r_i, and the subsequent $P_n(r_i)$ can be obtained from it through the use of the numerical analog of (4.46), namely,

$$r_i P(r_i) = \frac{\sigma_s}{2} \sum_{k=1}^{i-1} \Delta r_k E_1(|r_i - r_k|) r_k P_{n-1}(r_k)$$

$$+ \frac{\sigma_s}{2} \sum_{k=i+1}^{N} \Delta r_k E_1(|r_i - r_k|) r_k P_{n-1}(r_k)$$

$$+ \sigma_s r_i P_{n-1}(r_i)\left[1-e^{-\frac{\Delta r}{2}} + \frac{\Delta r}{2} E_1\left(\frac{\Delta r}{2}\right)\right]\Gamma. \qquad (4.105)$$

The $P_n(0)$ are determined from an equation similar to (4.104), namely,

$$P_n(0) = \sigma_s \sum_{k=\frac{N+1}{2}}^{N} \Delta r_k e^{-r_k} P_{n-1}(r_k). \qquad (4.106)$$

The $P_n(r_i)$ and $P(r_i)$ have been plotted for four cases in this thesis. The first case, Fig. 11 to Fig. 32, is for a large, highly scattering system of low neutron multiplicity. The critical radius is 11.9695, c=1.02, and $\sigma_s = .8\sigma_t$. The second case is identical except the critical radius is 12.0226, c=1.02, and $\sigma_s = .2\sigma_t$. The second case is plotted in Fig. 33 to Fig. 39. Fig. 40 through Fig. 46 cover the third case which has a critical radius of 1.4746, c=1.6, and $\sigma_s = .8\sigma_t$. The final case, in Fig. 47 through Fig. 50, has a critical radius of 1.4760, c=1.6, and $\sigma_s = .2\sigma_t$. Note that the 0th term is the direct leakage probability, and the 0th partial sum is just the 0th term. Since only scattering and absorption are considered here, one minus the total leakage probability is the absorption probability.

In Case One, namely the large, highly scattering system of low neutron multiplicity, it would be expected that a large number of the terms $P_n(r_i)$ would be required. However, no one foresaw that it would require as many as 45 terms to approximate the total leakage probability. The 0th and total leakage probability as in Fig. 11 are shaped as expected, namely, that a higher probability of leakage is predicted for each incremental step towards the boundary. Note that the

total leakage probability on the boundary is almost entirely made up of direct leakage. In Fig. 12 it can be seen that the first term of the leakage probability, the probability that a neutron emitted at a certain position scatters once and then leaks out, has changed shape from the zeroth term. The downward turn in the first term near the boundary is due to the fact that a large fraction of the neutrons born near the boundary leak out directly, and thus are not available for the "once scatter and out" scheme. This general shape-rising from the center, peaking and then dropping toward the boundary-is retained with diminishing magnitude and with the peak moving toward the center until about the thirty-fifth term. At this time the scattering transport is so large that only neutrons born near the center contribute significantly to the remaining terms, since neutrons born near the boundary are likely to have leaked out. It is interesting to note that neutrons born on the boundary have about a four in five chance of leaking out, and that neutrons born in the center only have about a one in five hundred chance of leaking out.

Case Two is essentially the same as Case One, except that the critical radii are slightly different, as σ_s in this case is only one quarter of the σ_s in Case One. The critical radii used are different, since they were obtained from Table XIII which contains indirect leakage critical radii estimates. Such estimates vary slightly with σ_s. The shapes and trends in Case Two are essentially the same as those in Case One, except that the total leakage probability curve is steeper. This is accounted for by the much higher absorption in Case Two. In other words, neutrons born near the boundary are still likely to leak out due

to their proximity to the boundary, but the neutrons born near the center are much more likely to be absorbed than in Case One. Such high absorption makes it possible to approximate the leakage with six terms (plus the inhomogeneous term) of the numerical Neumann series, because after a neutron scatters six times it doesn't affect the leakage. In Case Two, a neutron born near the boundary will leak about six times out of ten. A neutron born near the center stands only about 1 chance in 80,000 of leaking out.

Case Three is plotted in Fig. 40 through Fig. 46. This is a very small but highly scattering system. The large scattering cross section requires that ten terms in the numerical Neumann series be considered. The surprising element is that the leakage probability distribution is virtually flat. A neutron born anywhere in the sphere stands about a 65 percent chance of leaking out. The general shape trends discussed earlier apply here.

Case Four, in Fig. 47 through Fig. 50, stands in relation to Case Three as Case Two does to Case One. The system considered in Case Four has a slightly different critical radius and only one quarter the scattering cross section of Case Three. The general shape trends for the three terms in the numerical Neumann series follow the previous pattern. In Case Four, the total leakage probability follows the expected pattern, namely that a neutron born near the edge is more likely to leak out than one born near the center. The difference is slight, since a neutron born near the center is about 25 percent likely to leak out vs. 65 percent near the boundary.

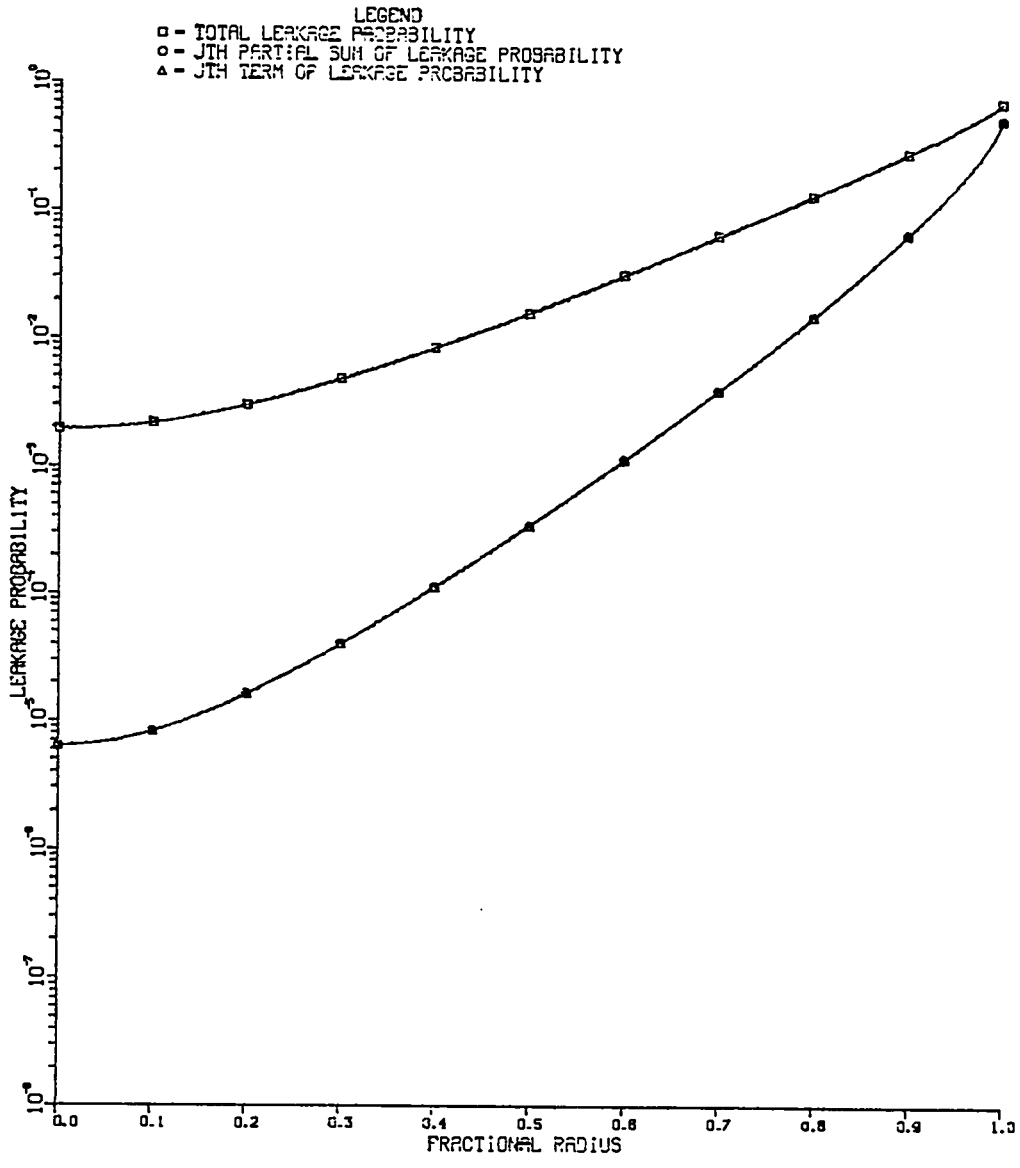


Figure 11.

Plot of the total leakage probability, 0th partial sum of the leakage probability, and 0th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

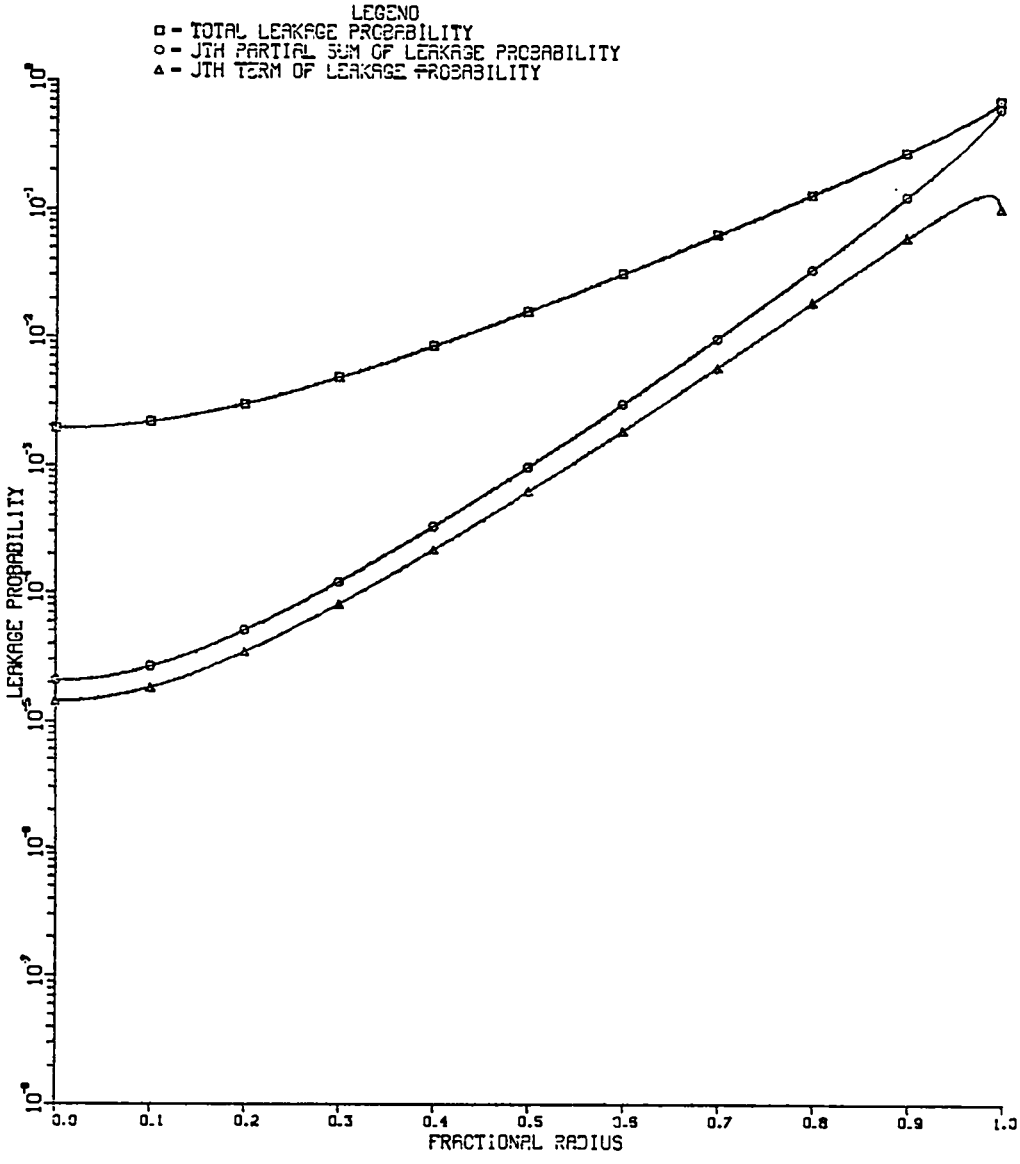


Figure 12.

Plot of the total leakage probability, 1st partial sum of the leakage probability, and 1st term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

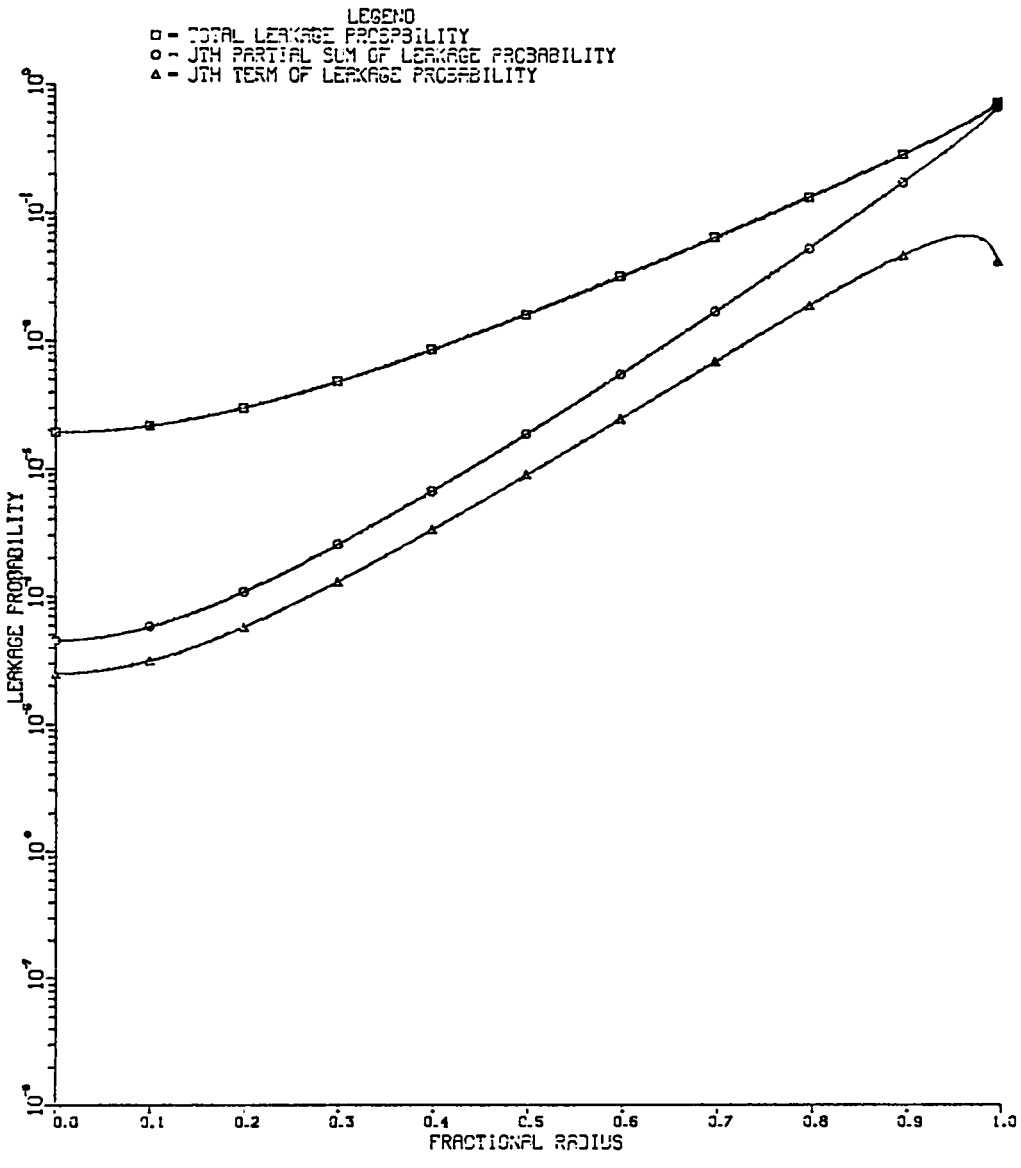


Figure 13.

Plot of the total leakage probability, 2nd partial sum of the leakage probability, and 2nd term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

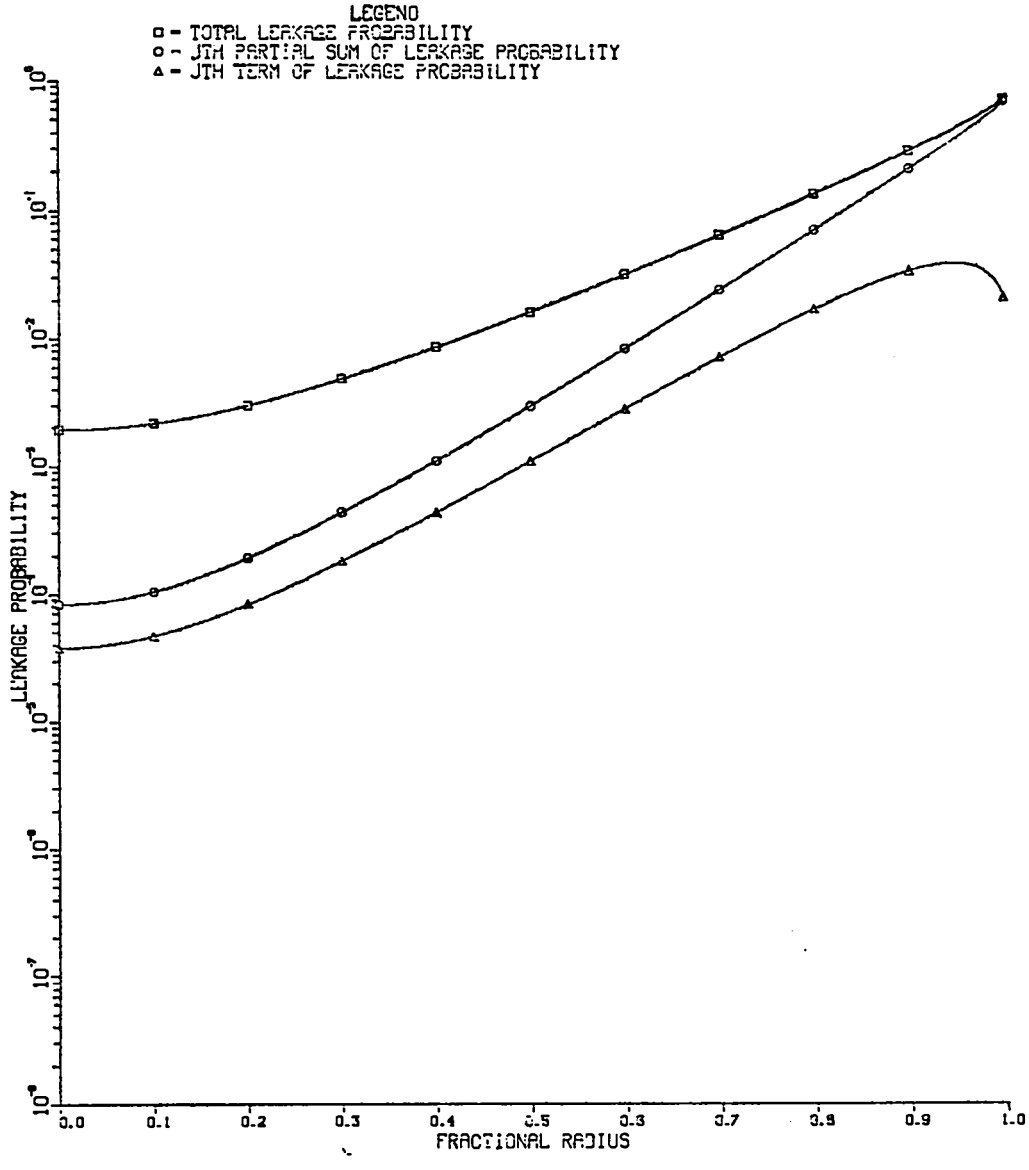


Figure 14.

Plot of the total leakage probability, 3rd partial sum of the leakage probability, and 3rd term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

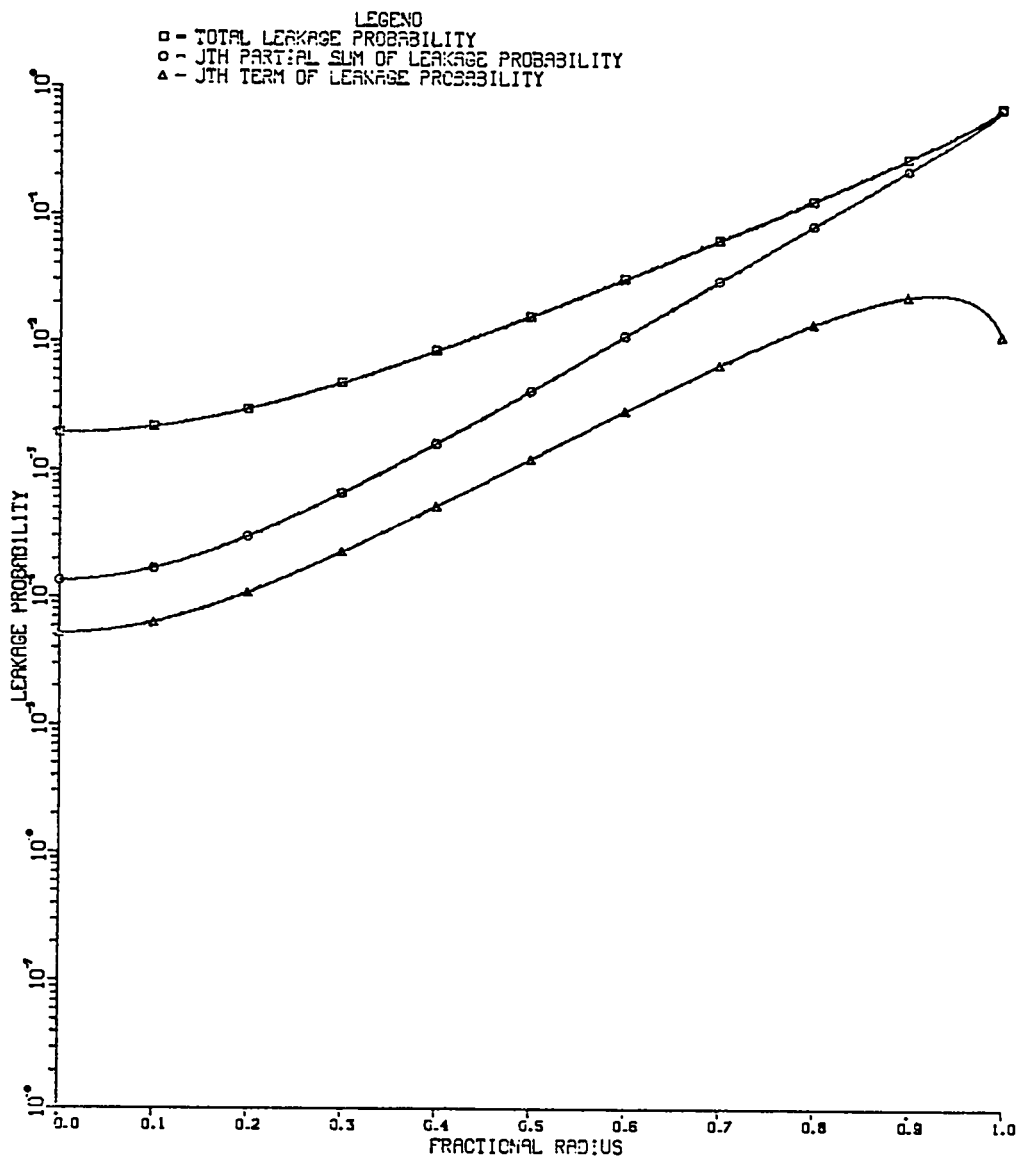


Figure 15.

Plot of the total leakage probability, 4th partial sum of the leakage probability, and 4th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

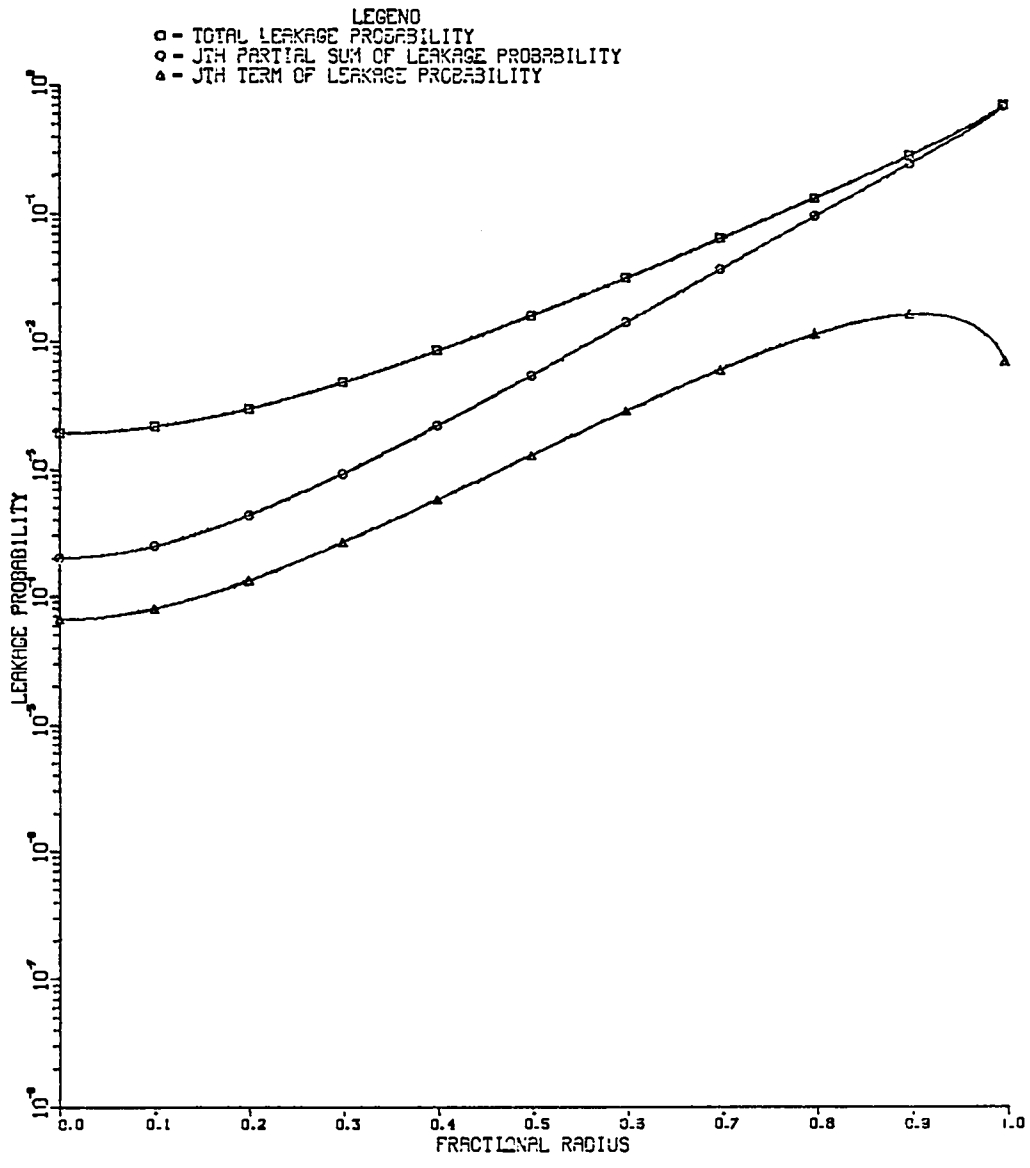


Figure 16.

Plot of the total leakage probability, 5th partial sum of the leakage probability, and 5th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

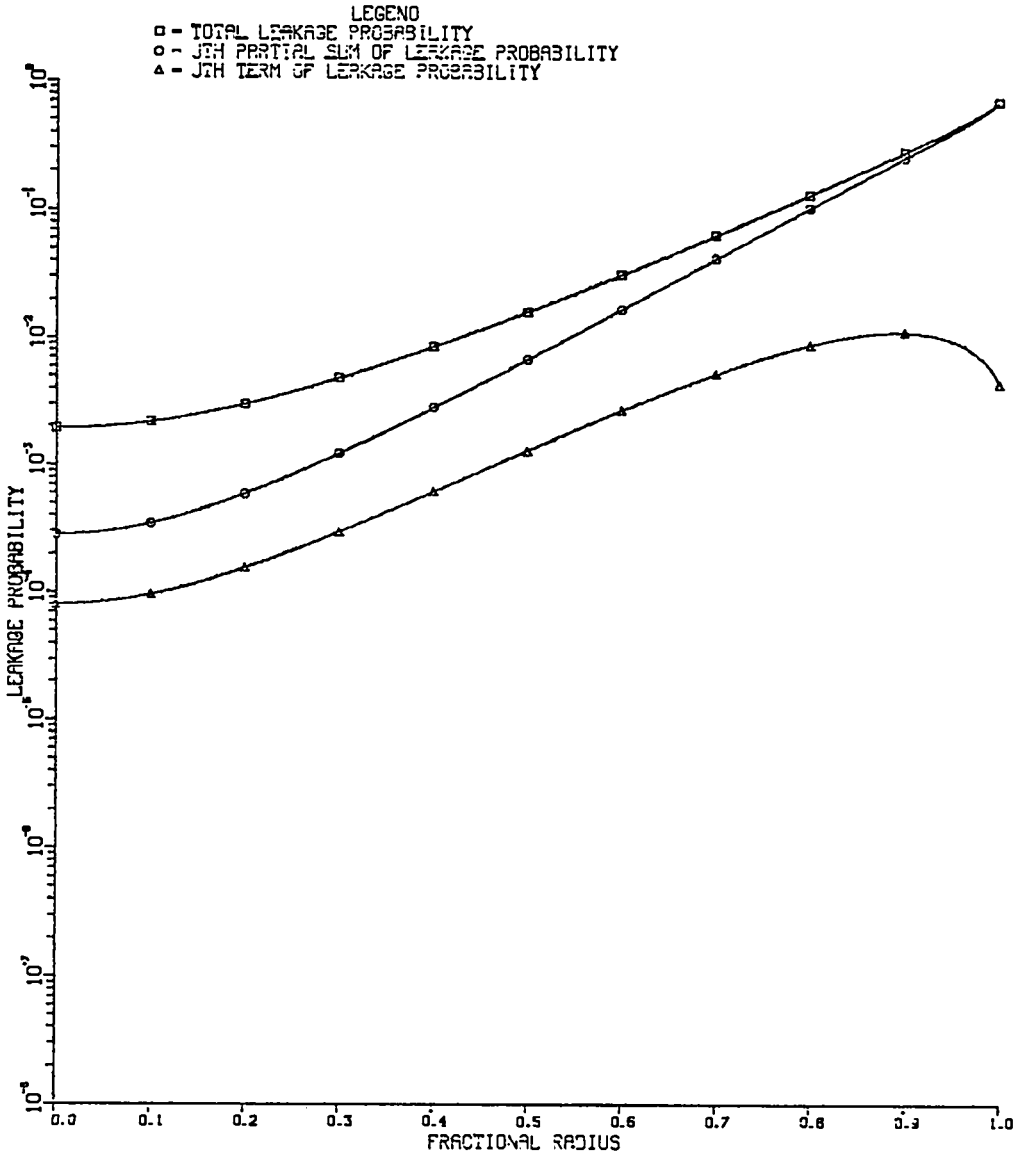


Figure 17.

Plot of the total leakage probability, 6th partial sum of the leakage probability, and 6th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

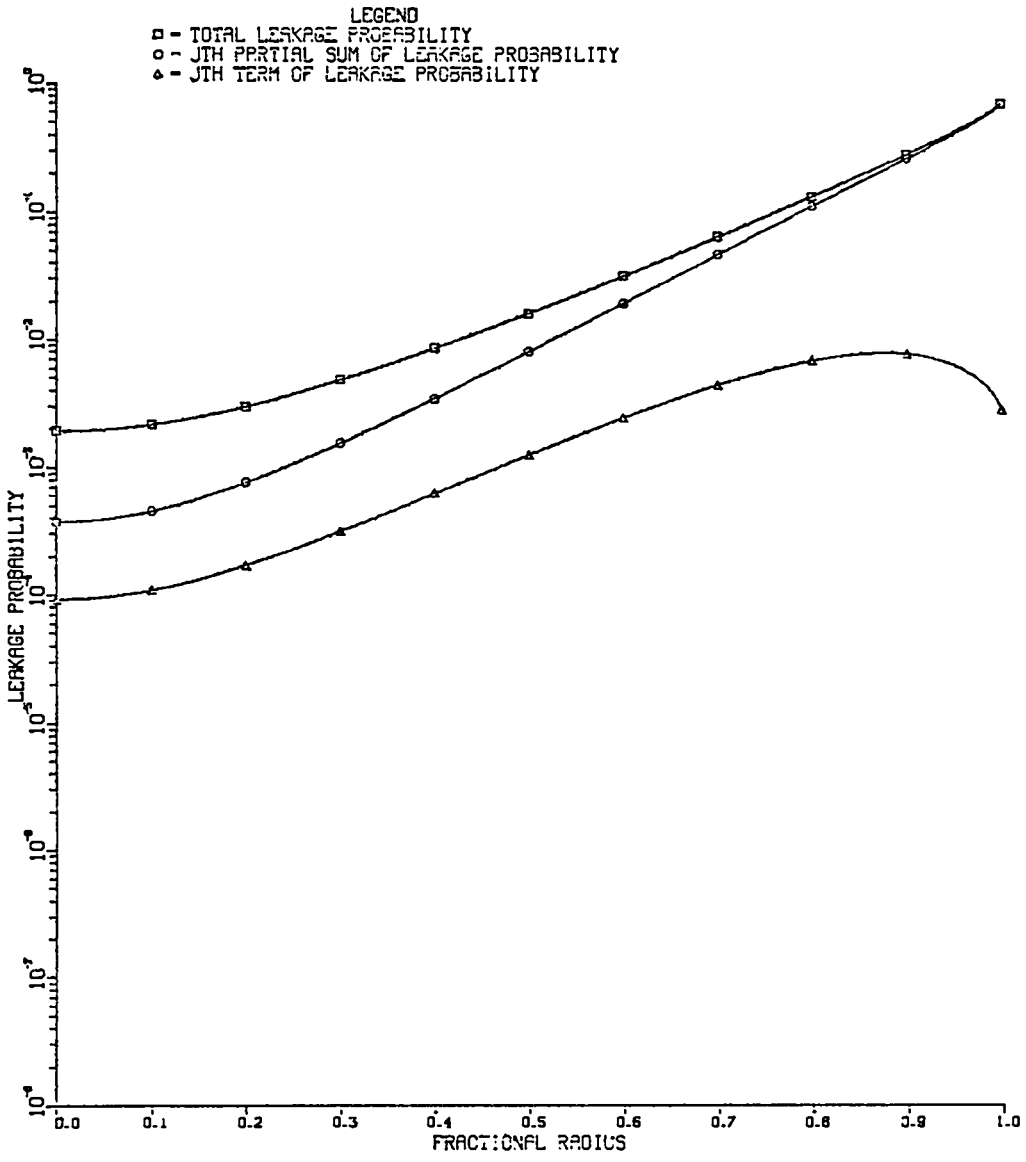


Figure 18.

Plot of the total leakage probability, 7th partial sum of the leakage probability, and 7th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

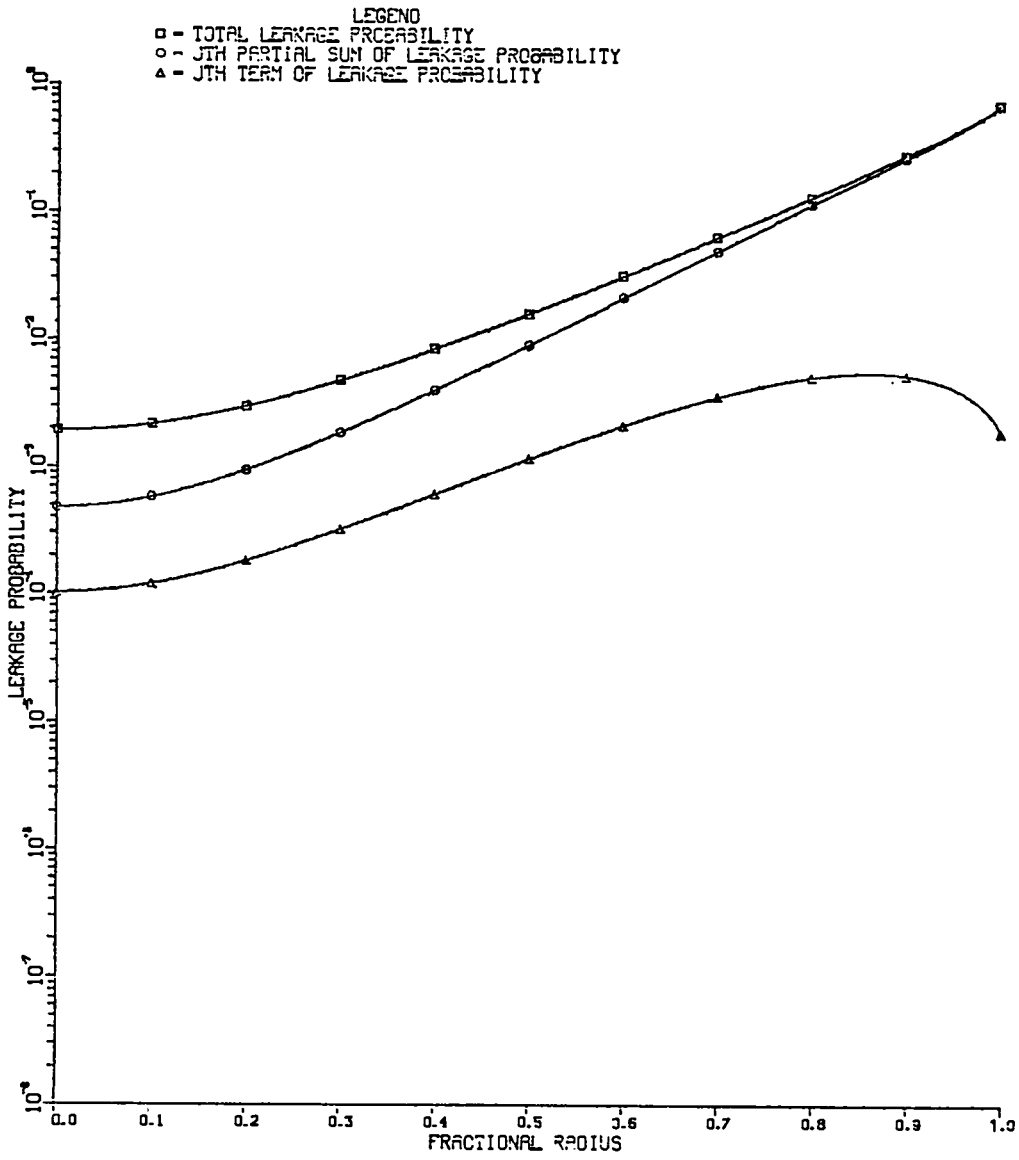


Figure 19.

Plot of the total leakage probability, 8th partial sum of the leakage probability, and 8th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

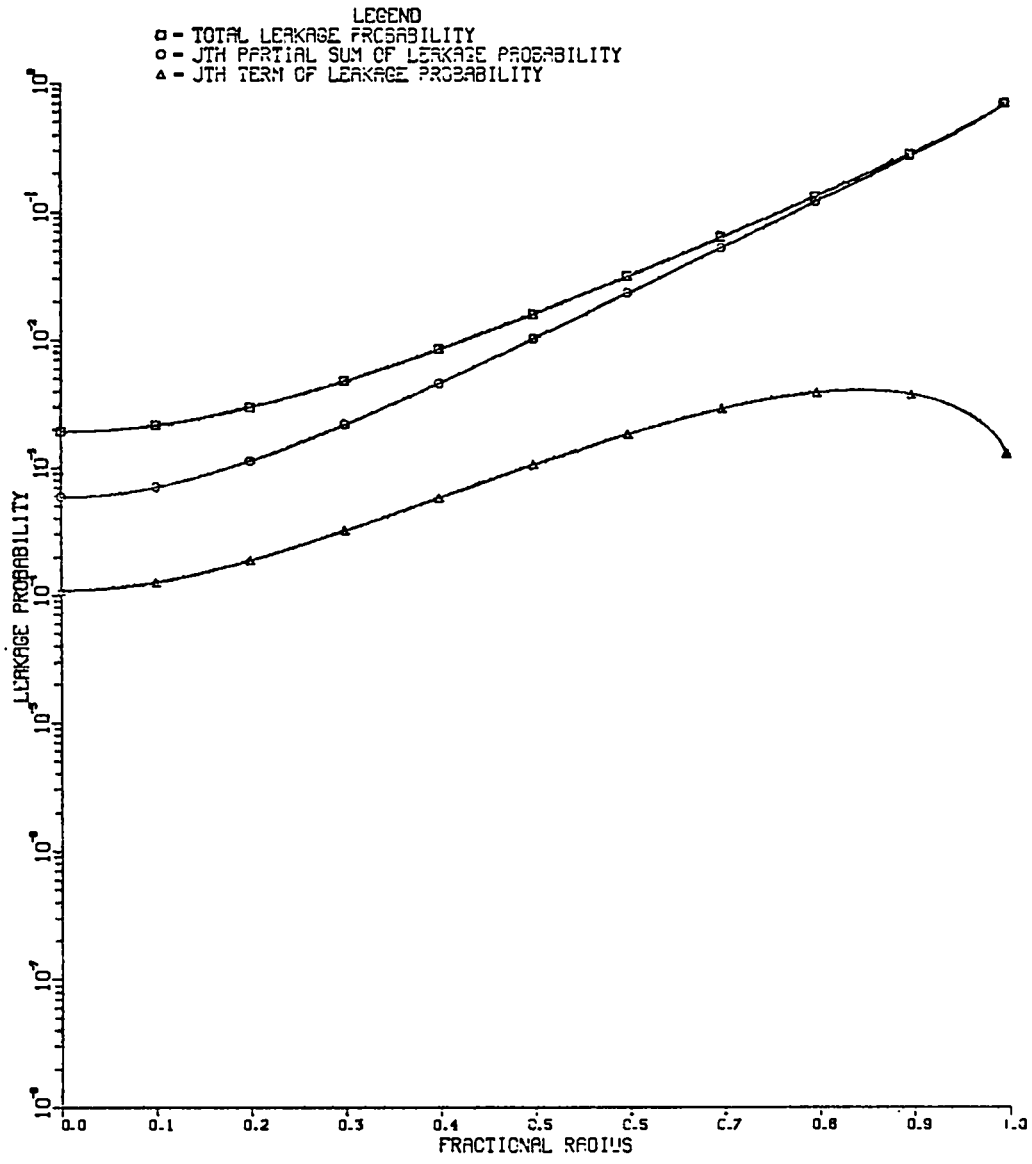


Figure 20.

Plot of the total leakage probability, 9th partial sum of the leakage probability, and 9th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

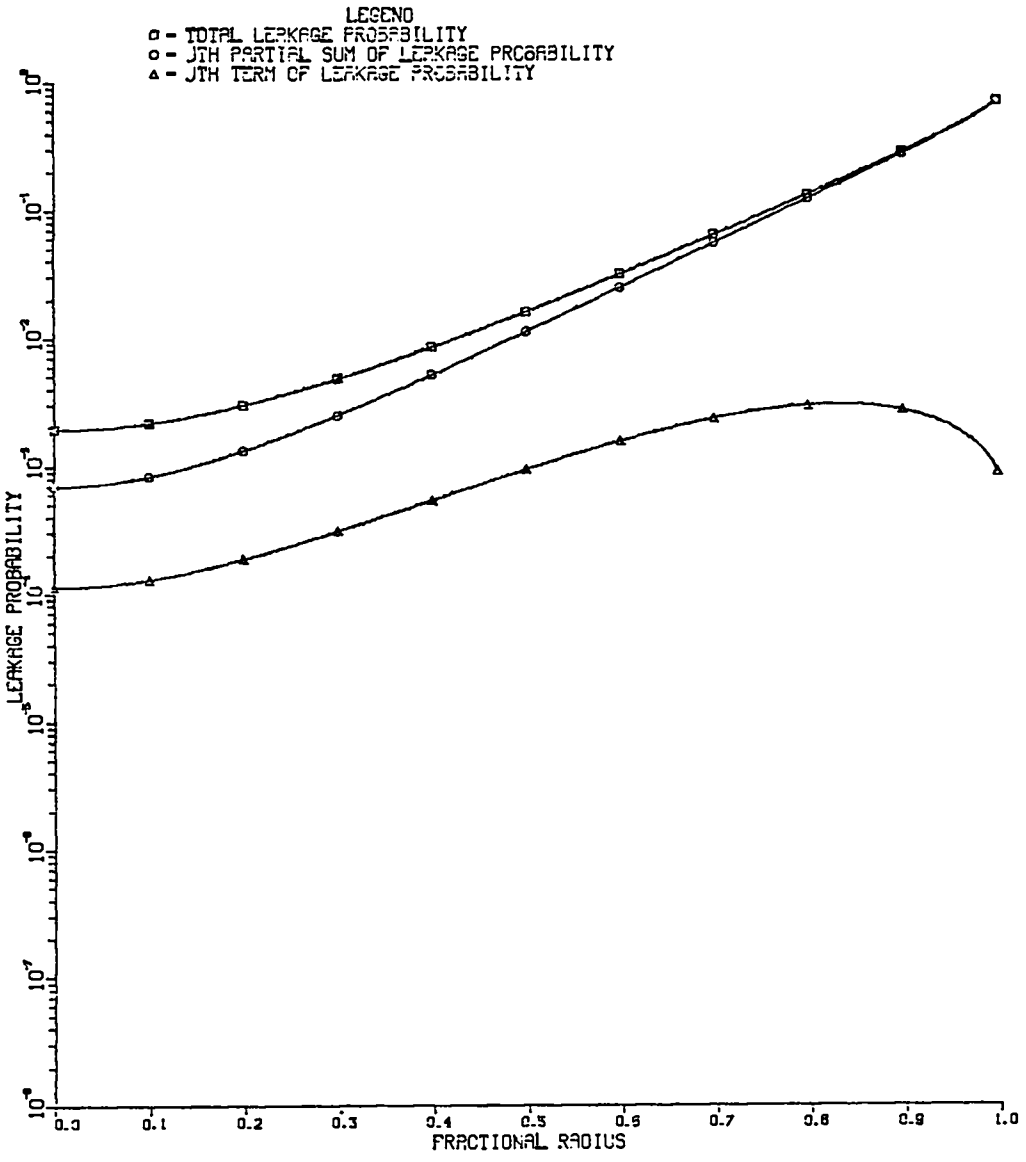


Figure 21.

Plot of the total leakage probability, 10th partial sum of the leakage probability, and 10th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

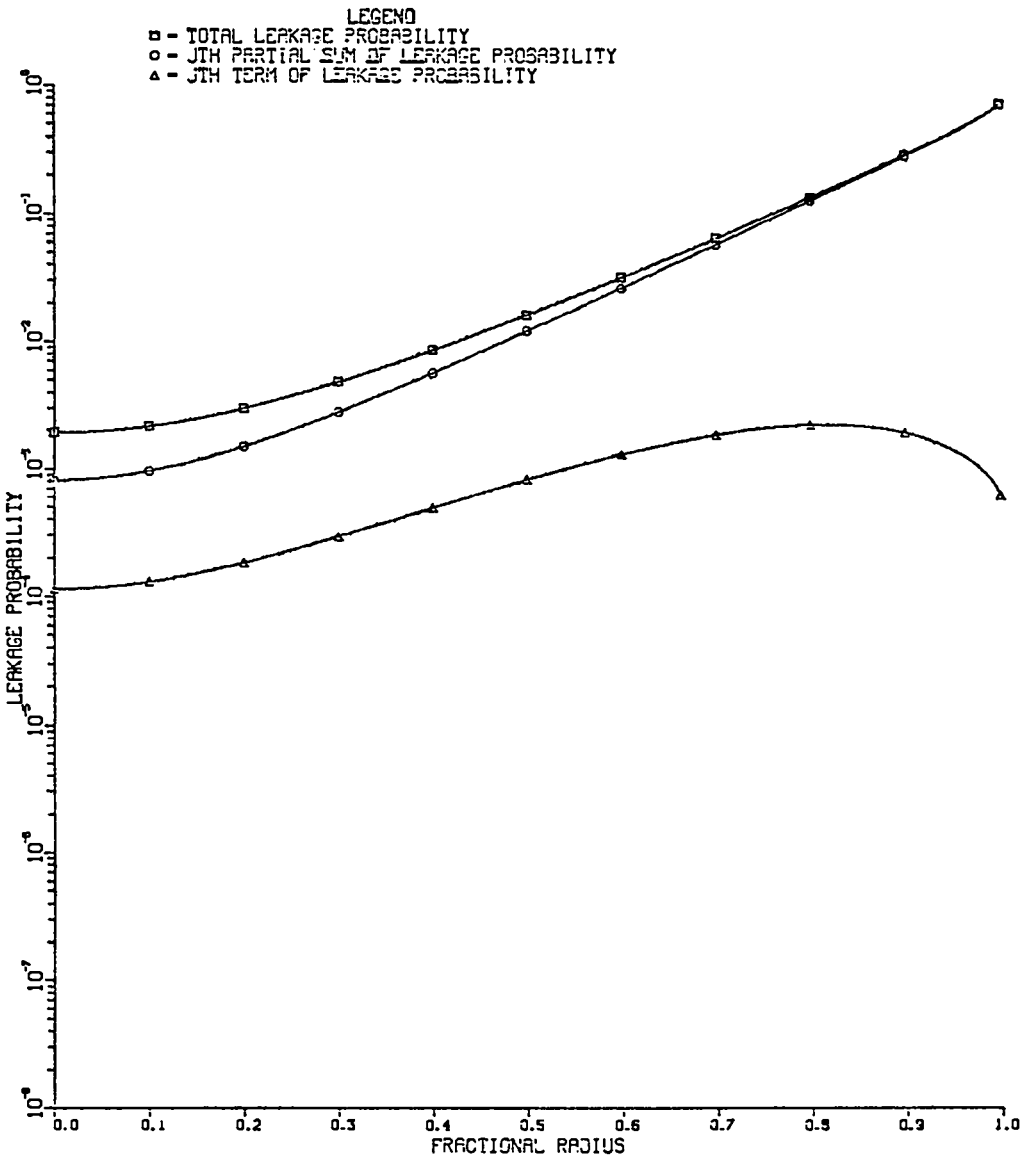


Figure 22.

Plot of the total leakage probability, 11th partial sum of the leakage probability, and 11th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

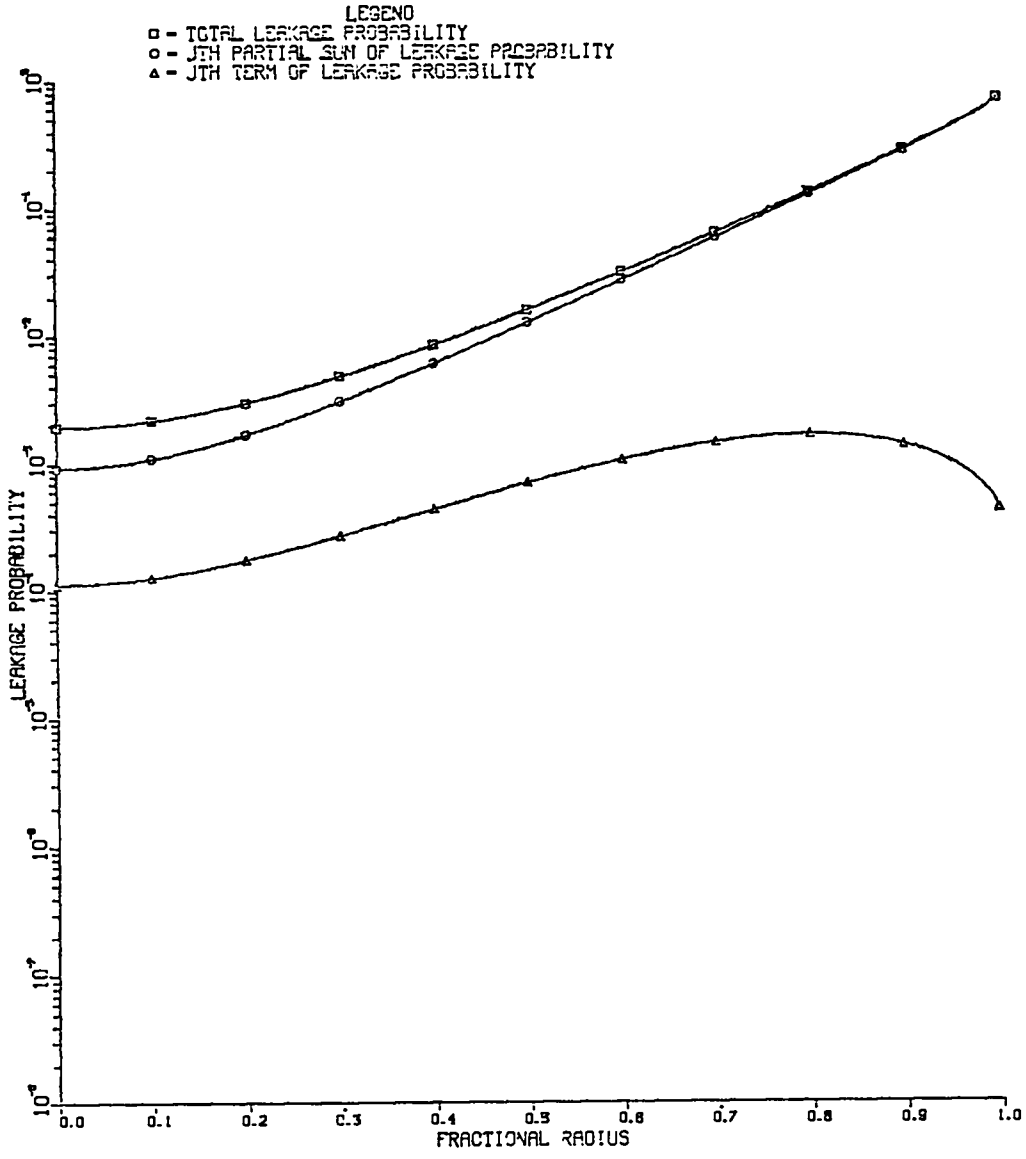


Figure 23.

Plot of the total leakage probability, 12th partial sum of the leakage probability, and 12th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

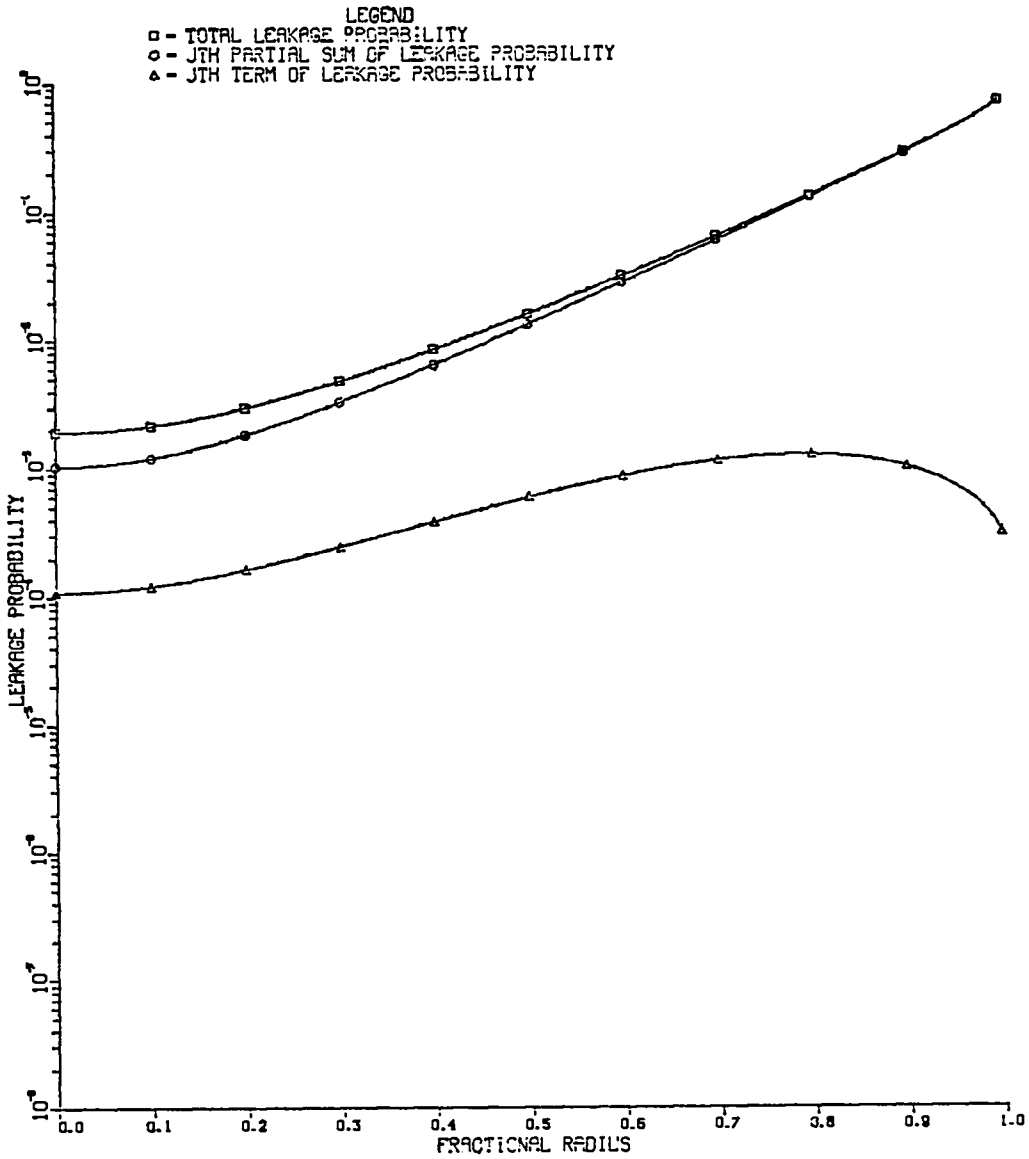


Figure 24.

Plot of the total leakage probability, 13th partial sum of the leakage probability, and 13th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

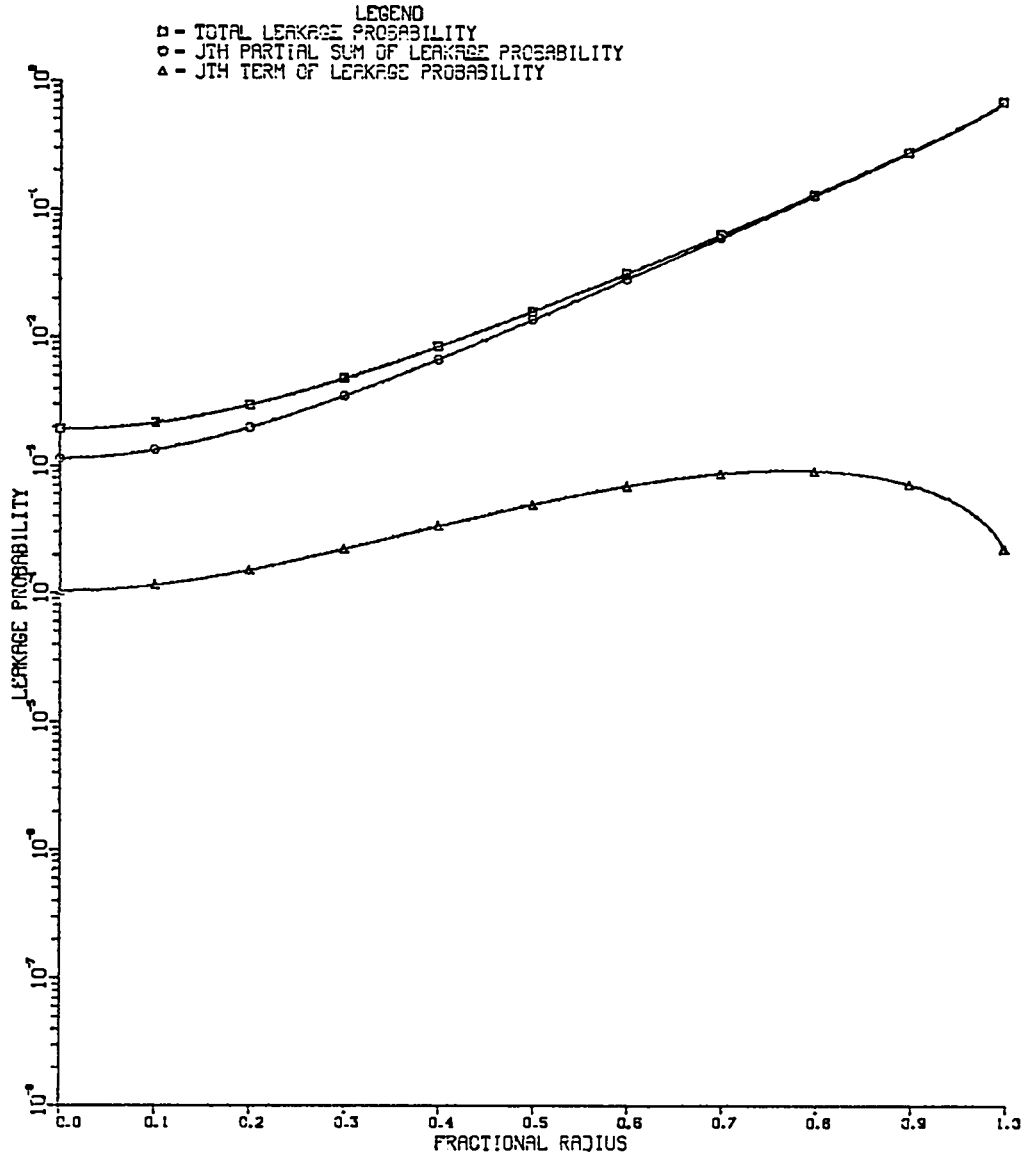


Figure 25.

Plot of the total leakage probability, 14th partial sum of the leakage probability, and 14th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

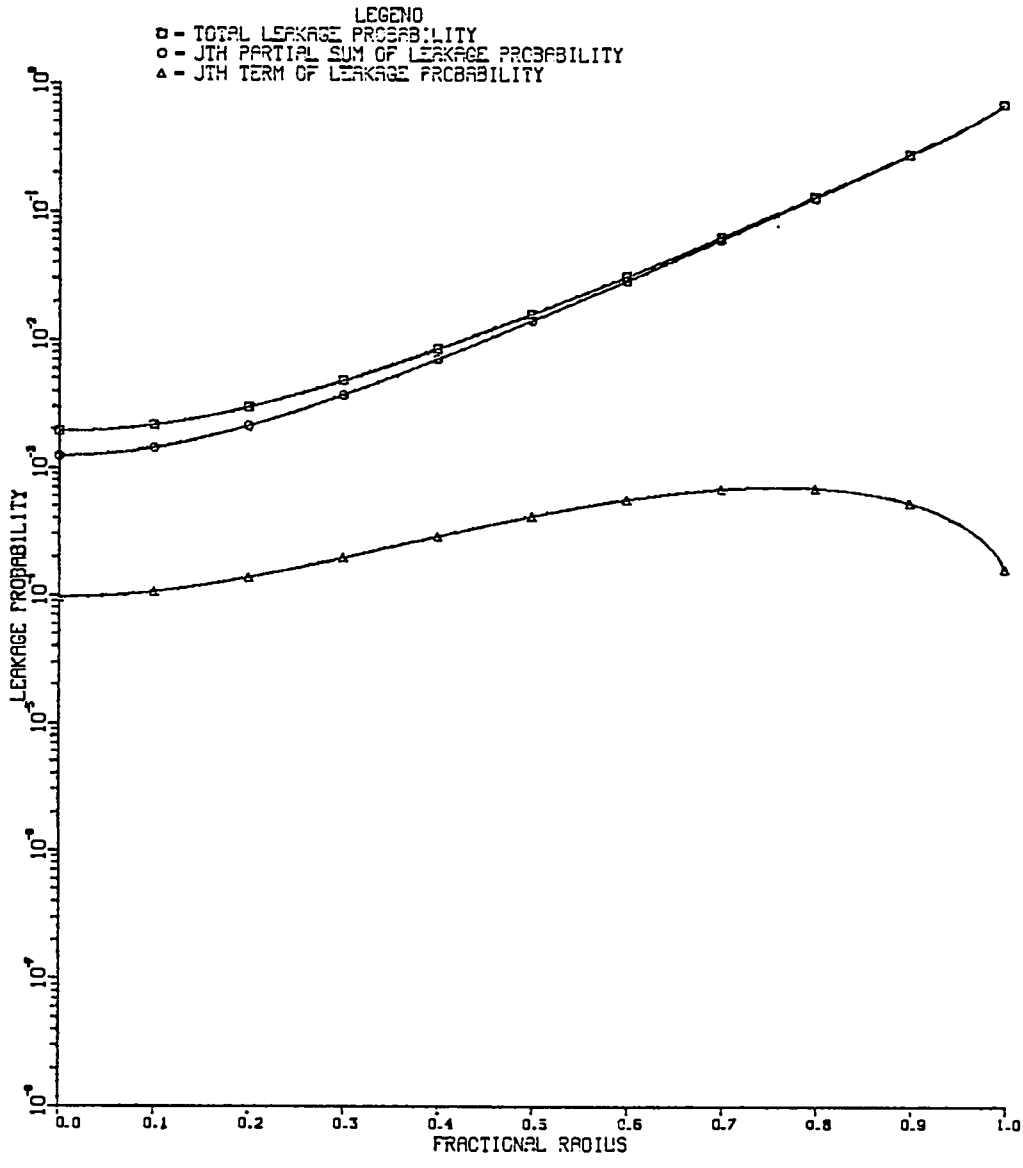


Figure 26.

Plot of the total leakage probability, 15th partial sum of the leakage probability, and 15th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

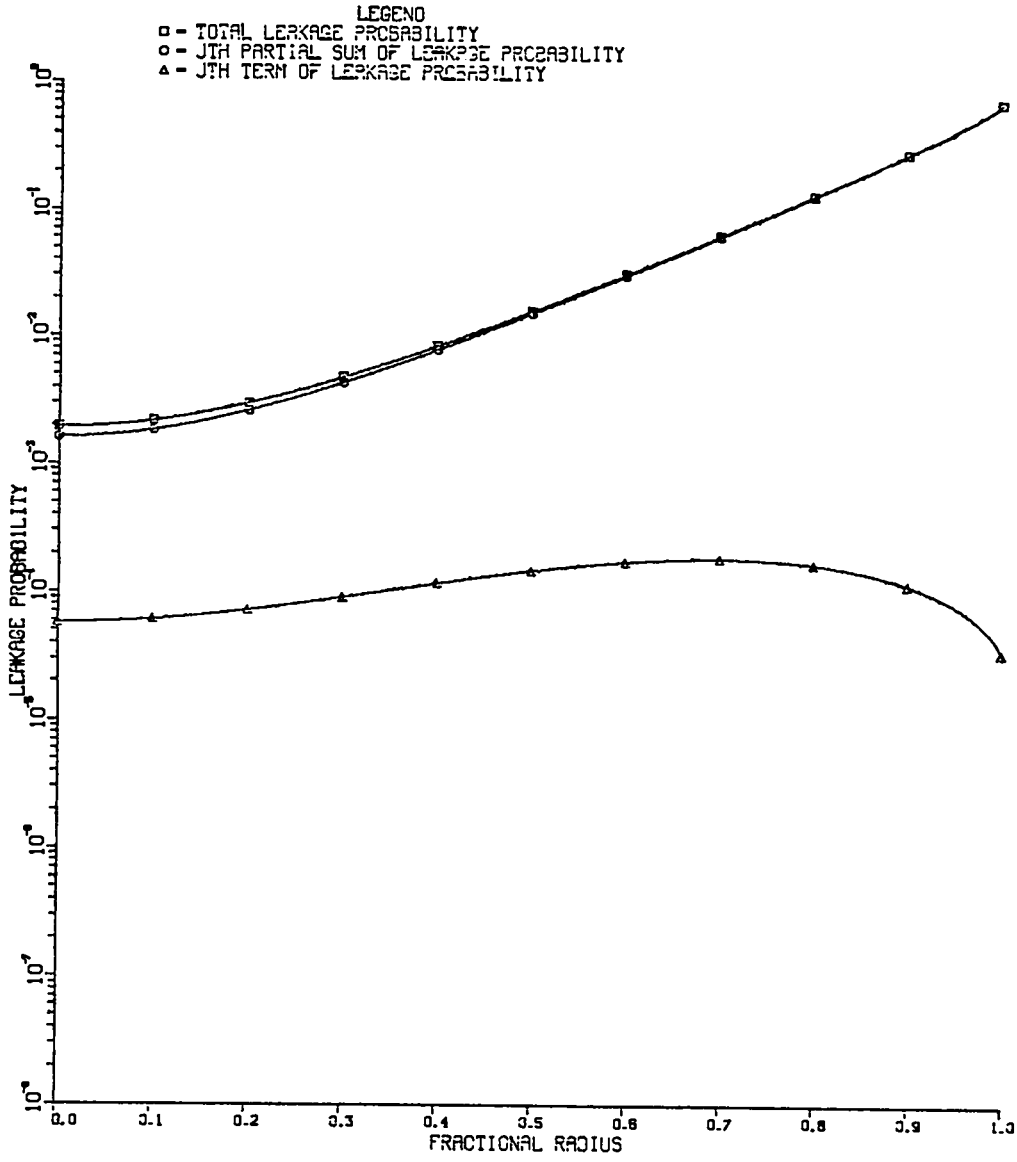


Figure 27.

Plot of the total leakage probability, 20th partial sum of the leakage probability, and 20th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

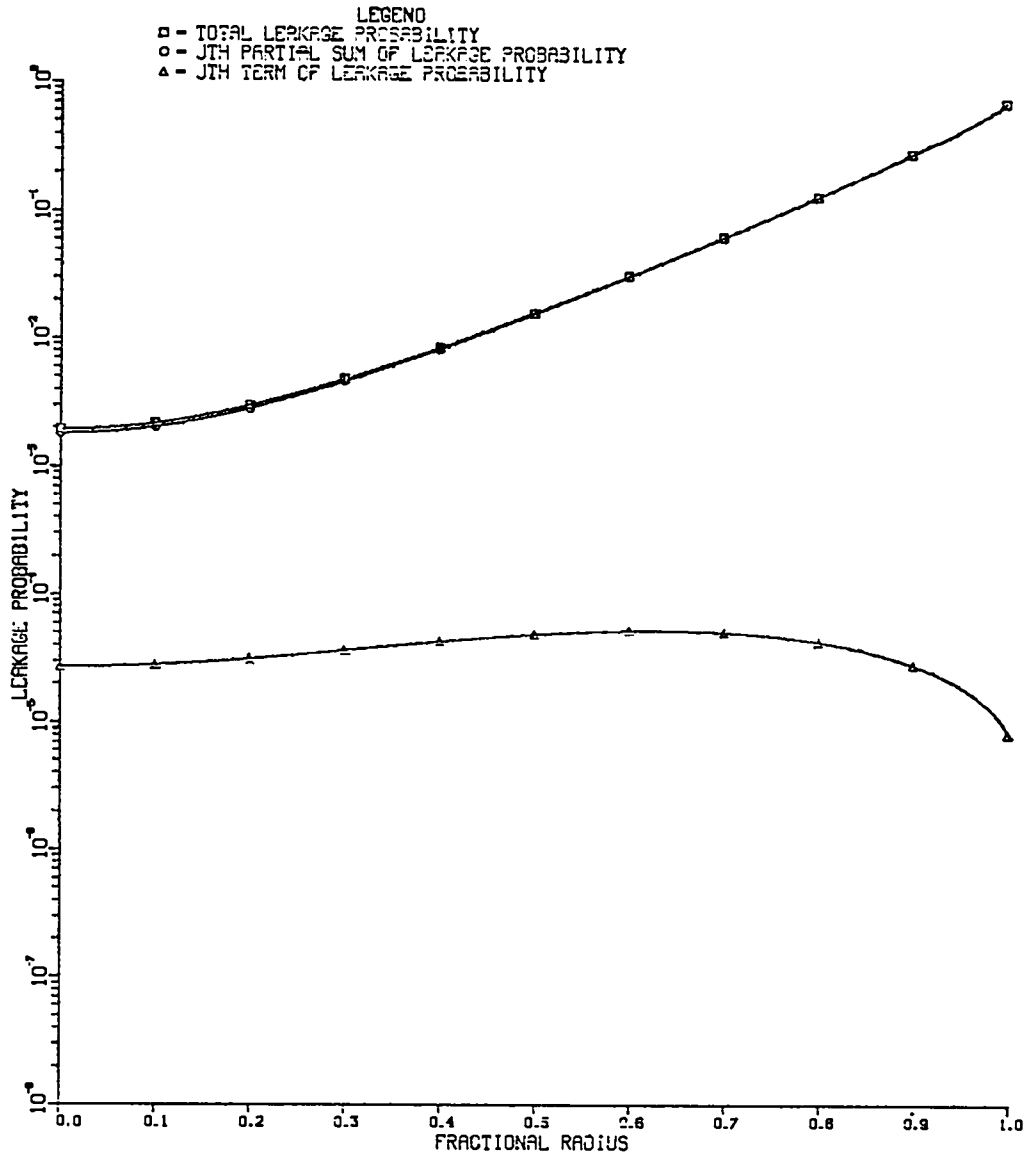


Figure 28.

Plot of the total leakage probability, 25th partial sum of the leakage probability, and 25th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

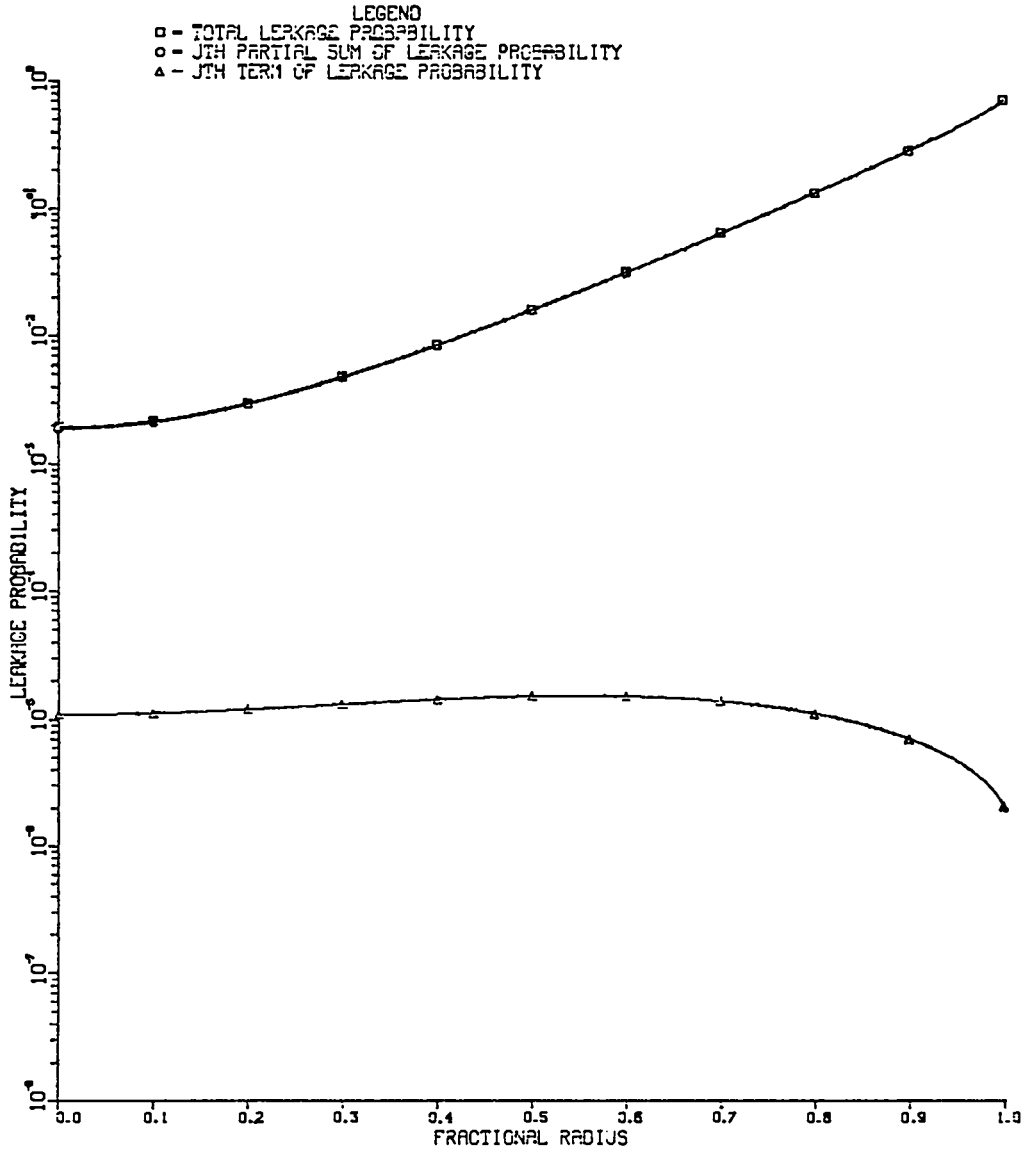


Figure 29.

Plot of the total leakage probability, 30th partial sum of the leakage probability, and 30th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

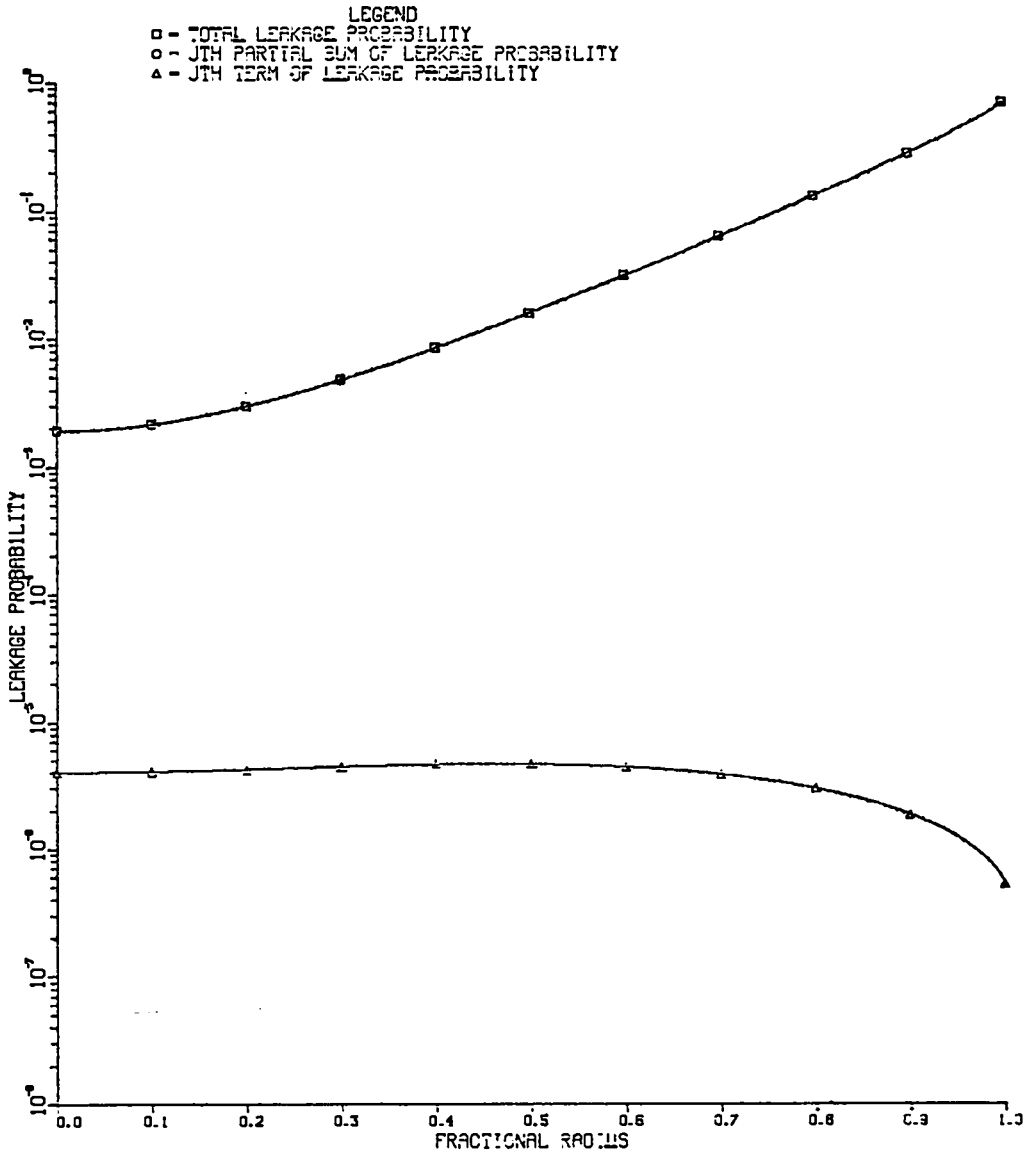


Figure 30.

Plot of the total leakage probability, 35th partial sum of the leakage probability, and 35th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

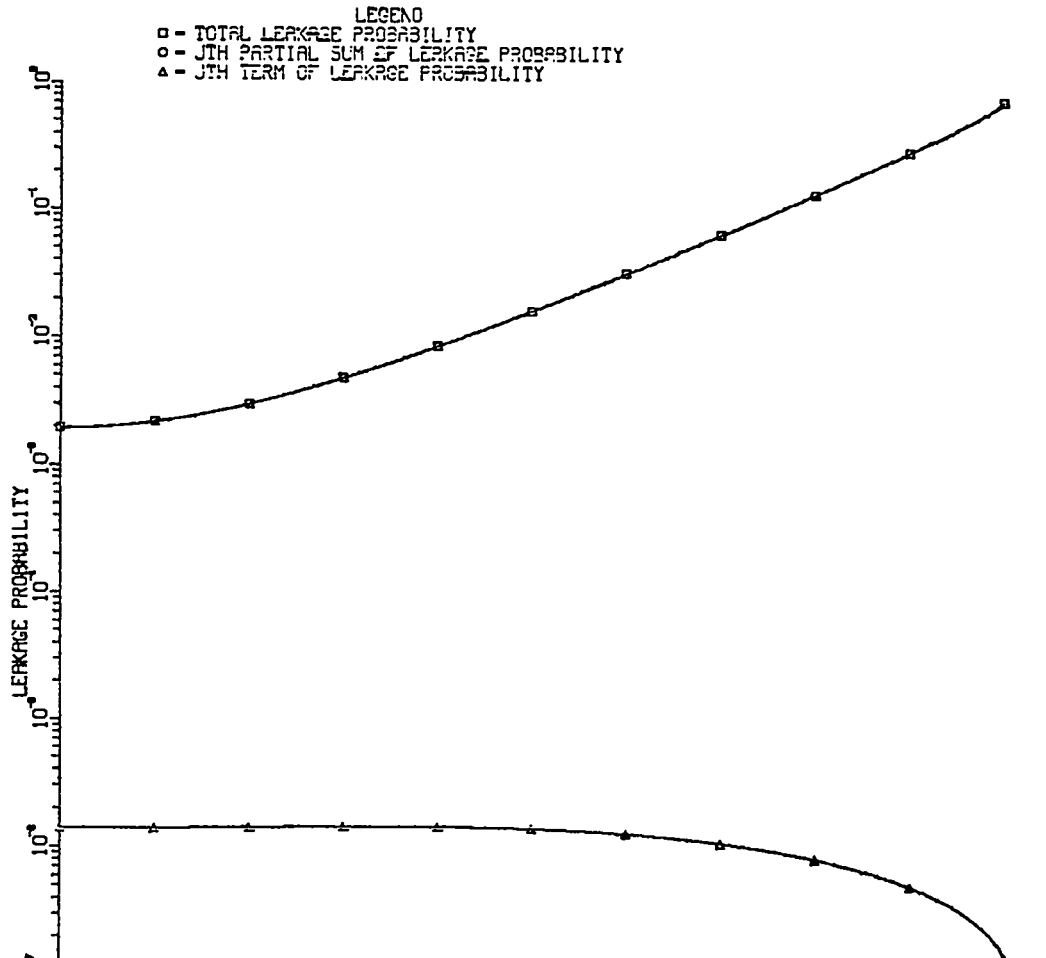


Figure 31.

Plot of the total leakage probability, 40th partial sum of the leakage probability, and 40th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

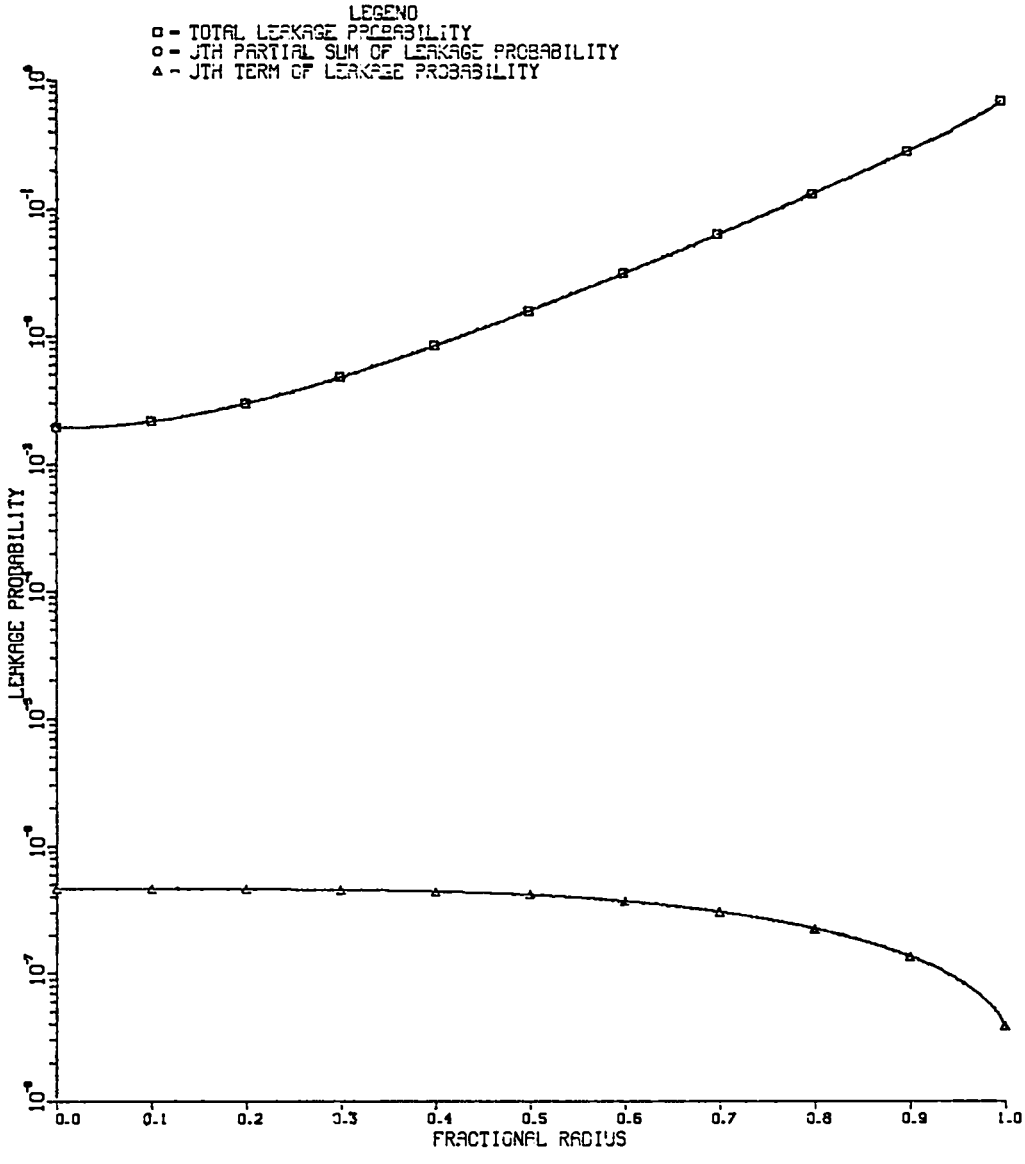


Figure 32.

Plot of the total leakage probability, 45th partial sum of the leakage probability, and 45th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .8\sigma_t$.

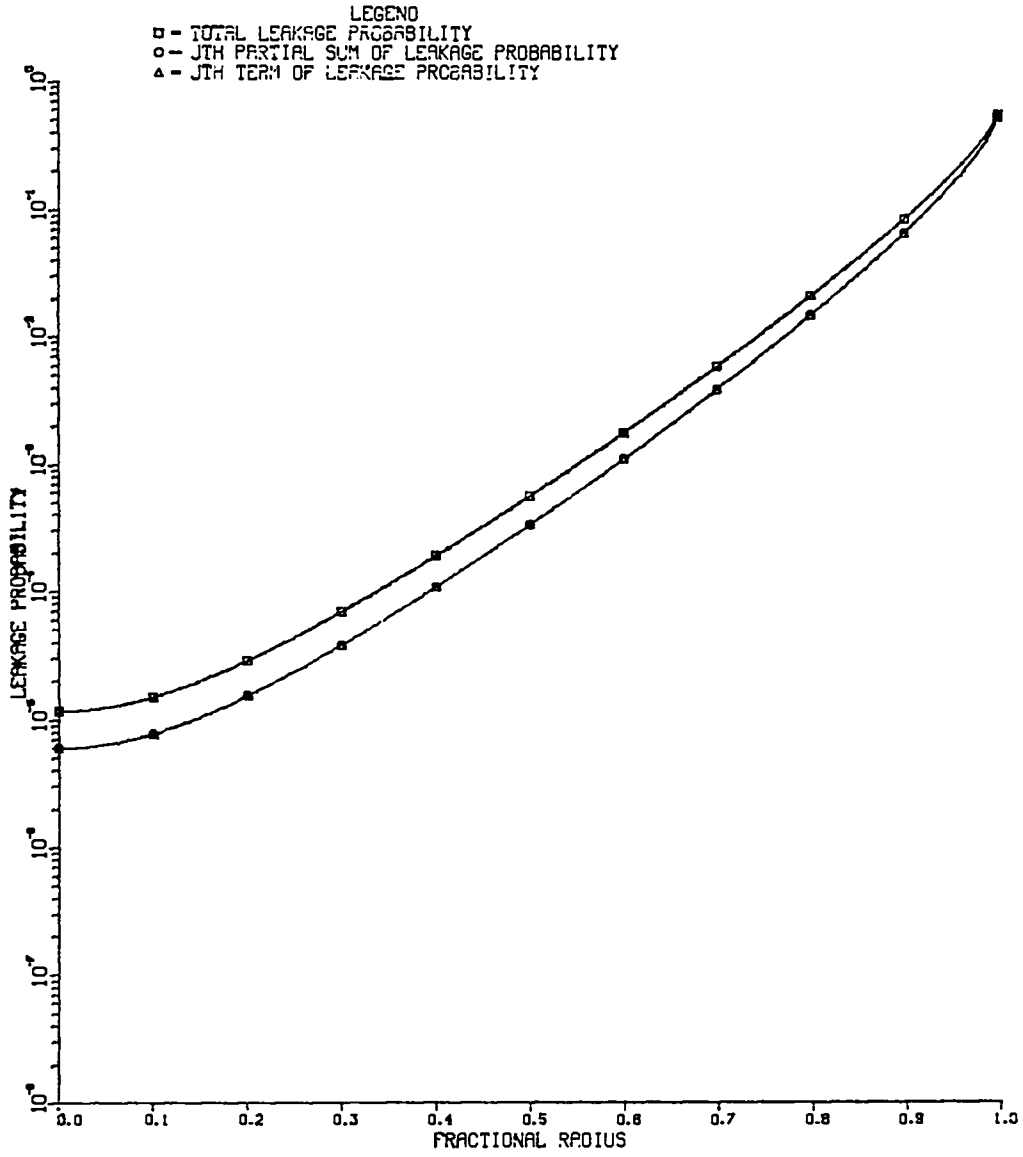


Figure 33.

Plot of the total leakage probability, 0th partial sum of the leakage probability, and 0th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

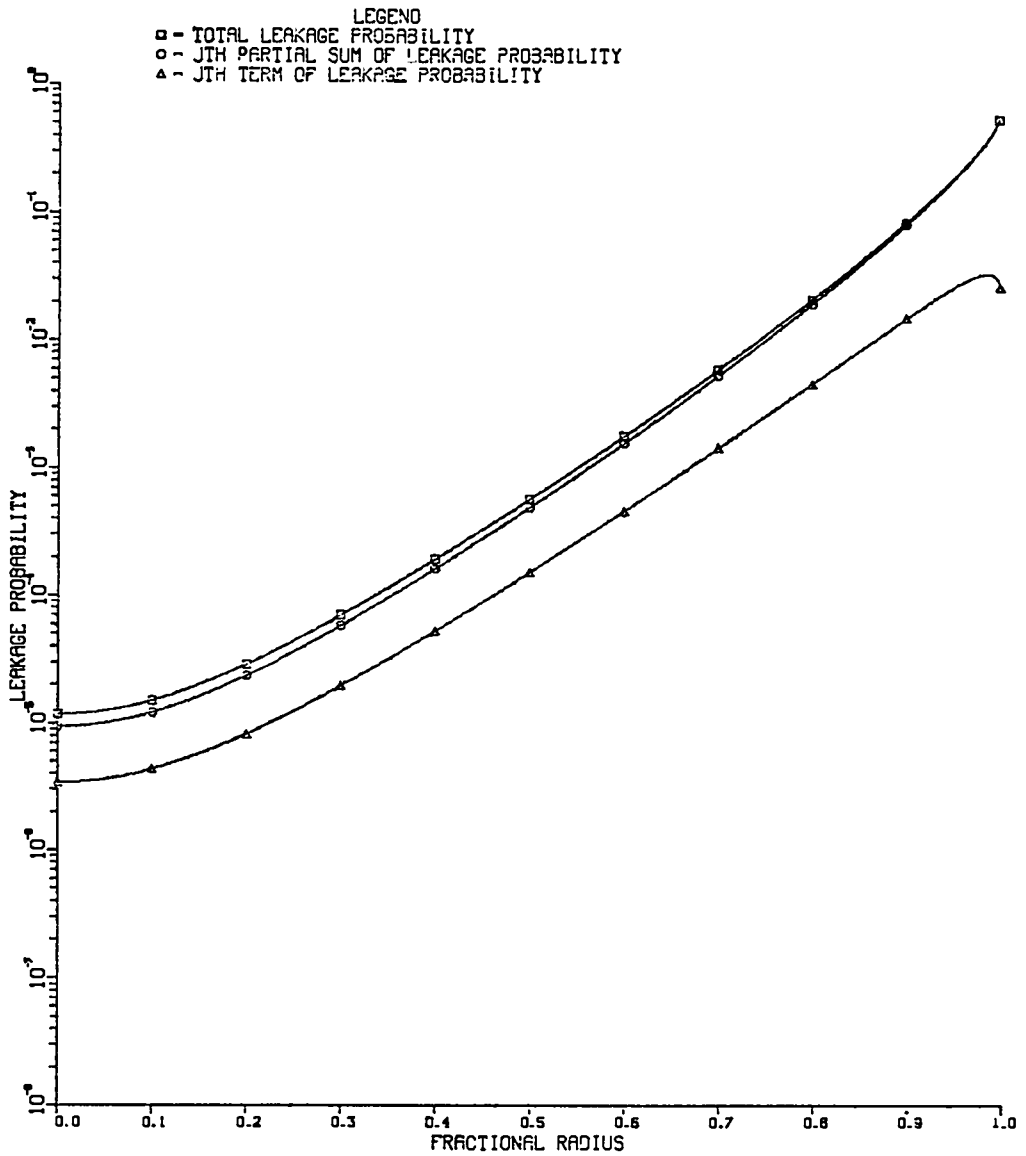


Figure 34.

Plot of the total leakage probability, 1st partial sum of the leakage probability, and 1st term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

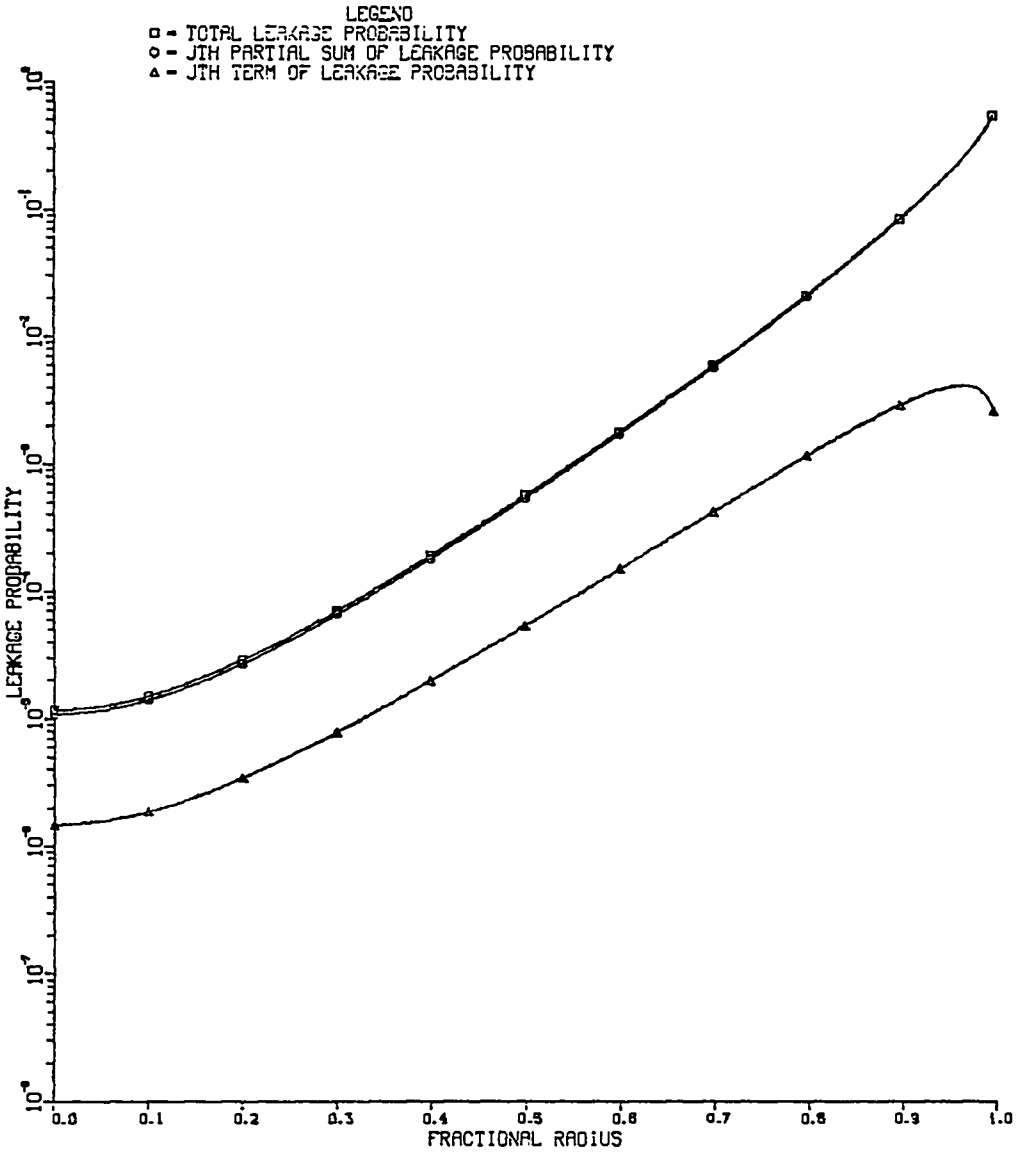


Figure 35.

Plot of the total leakage probability, 2nd partial sum of the leakage probability, and 2nd term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

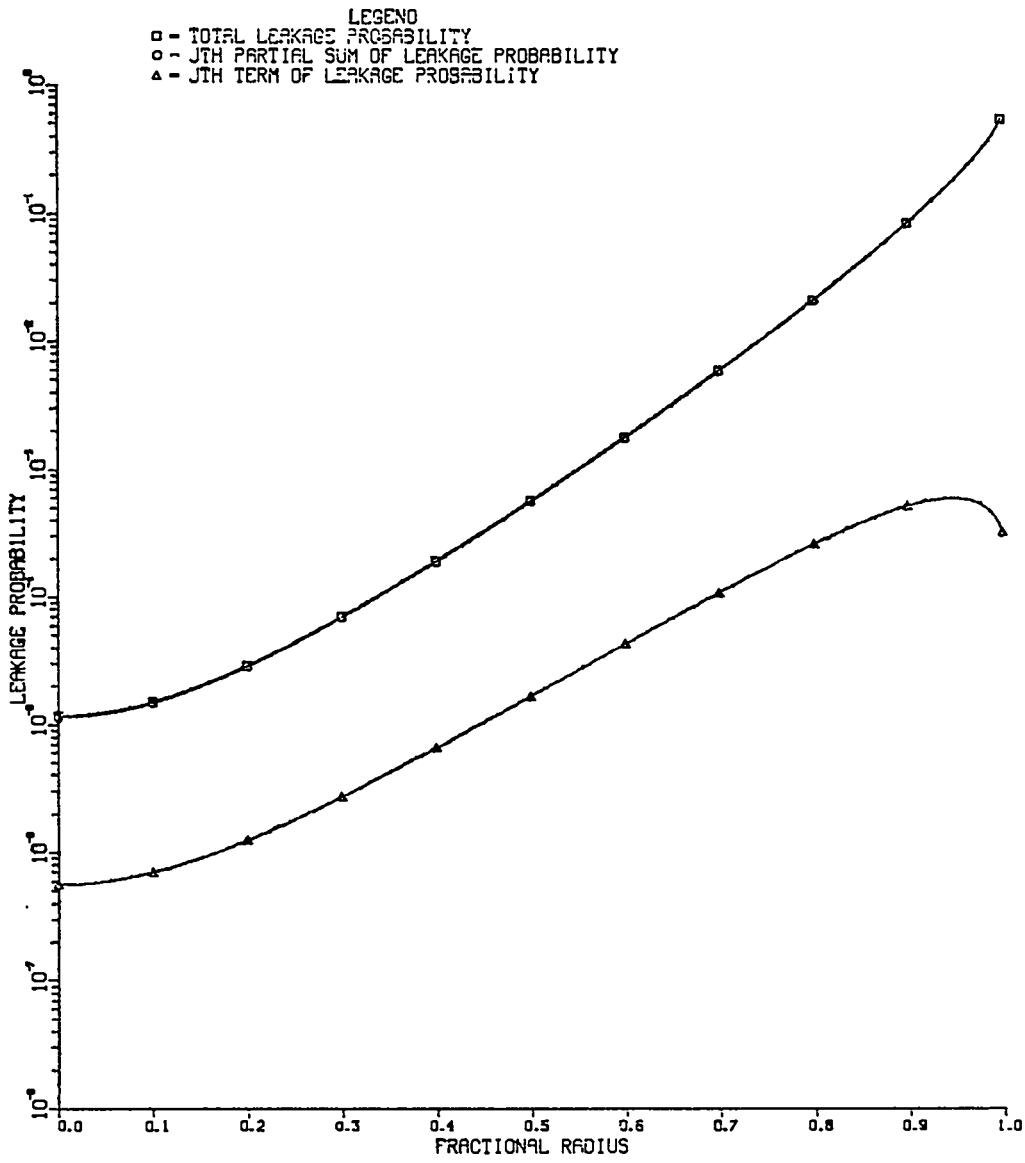


Figure 36.

Plot of the total leakage probability, 3rd partial sum of the leakage probability, and 3rd term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

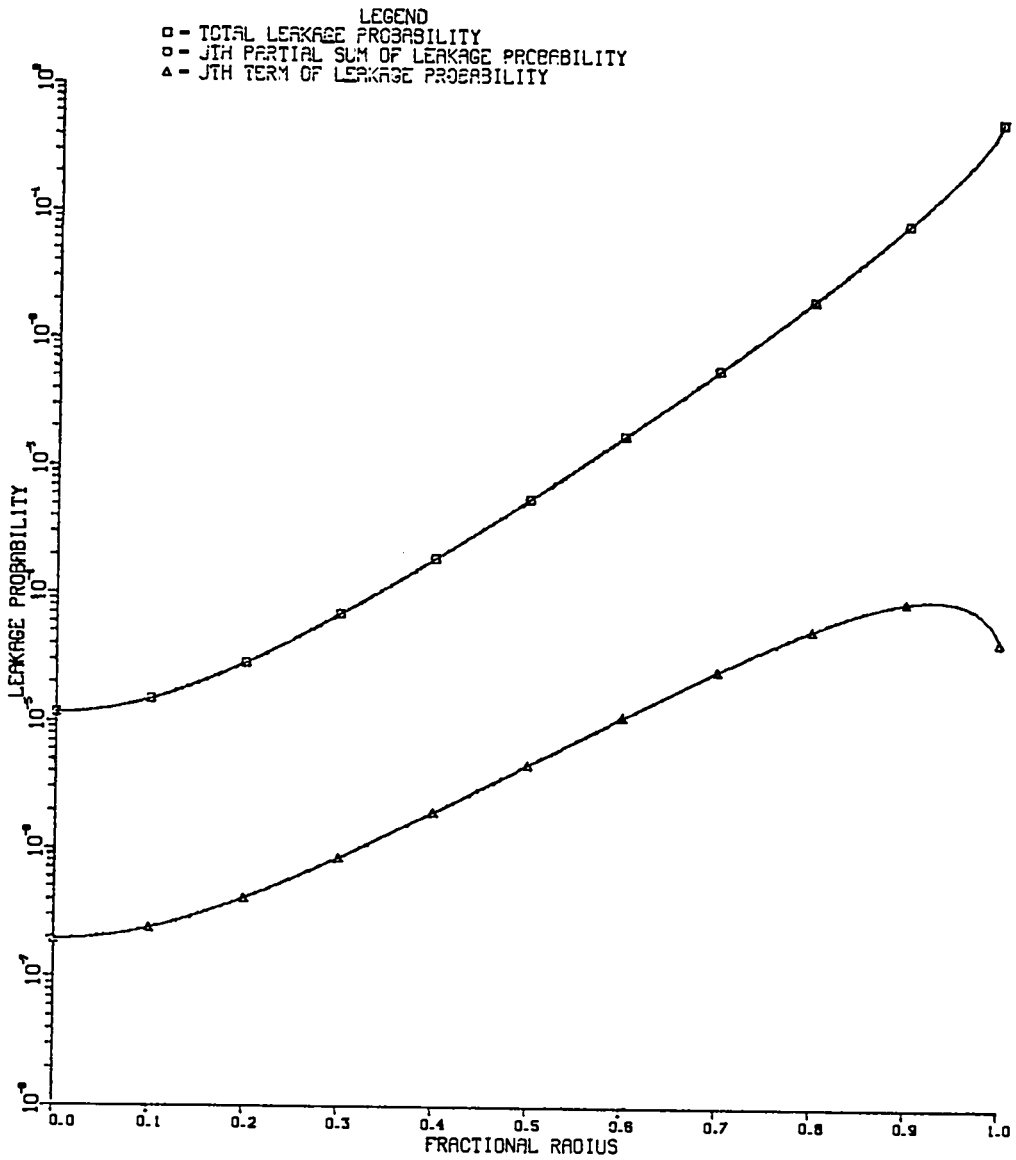


Figure 37.

Plot of the total leakage probability, 4th partial sum of the leakage probability, and 4th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

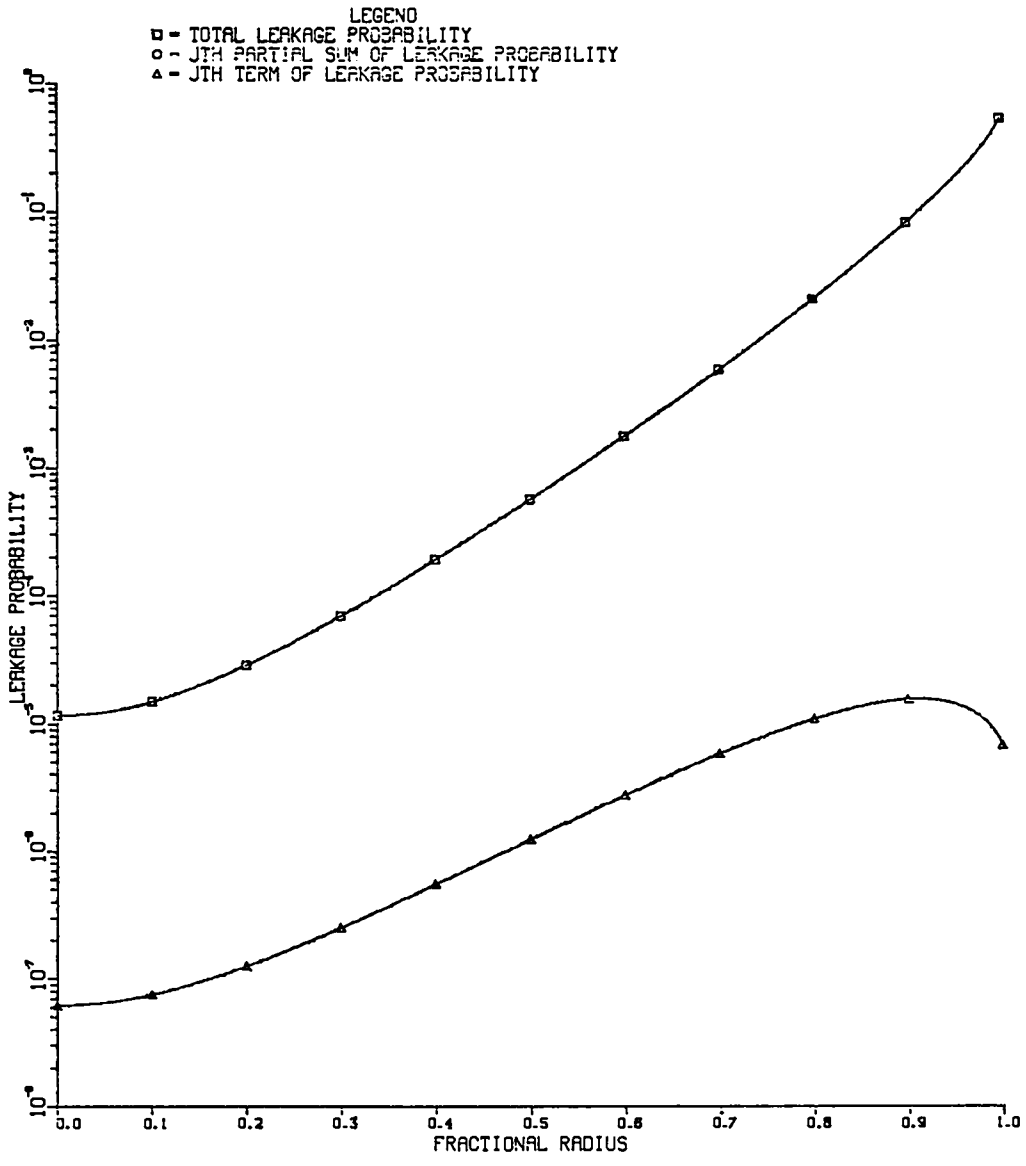


Figure 38.

Plot of the total leakage probability, 5th partial sum of the leakage probability, and 5th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

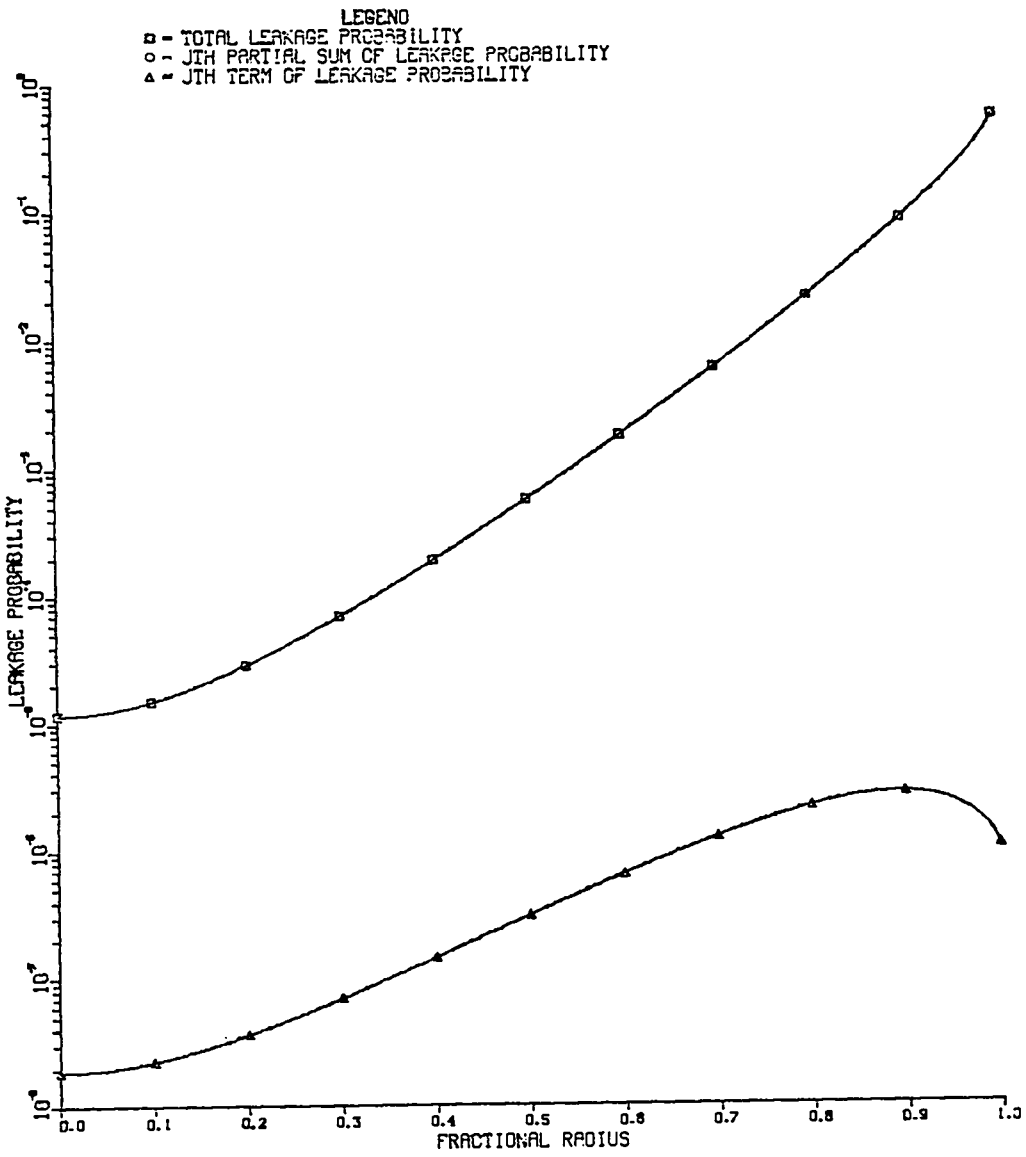


Figure 39.

Plot of the total leakage probability, 6th partial sum of the leakage probability, and 6th term of the leakage probability versus fractional radius for c=1.02, and $\sigma_s = .2\sigma_t$.

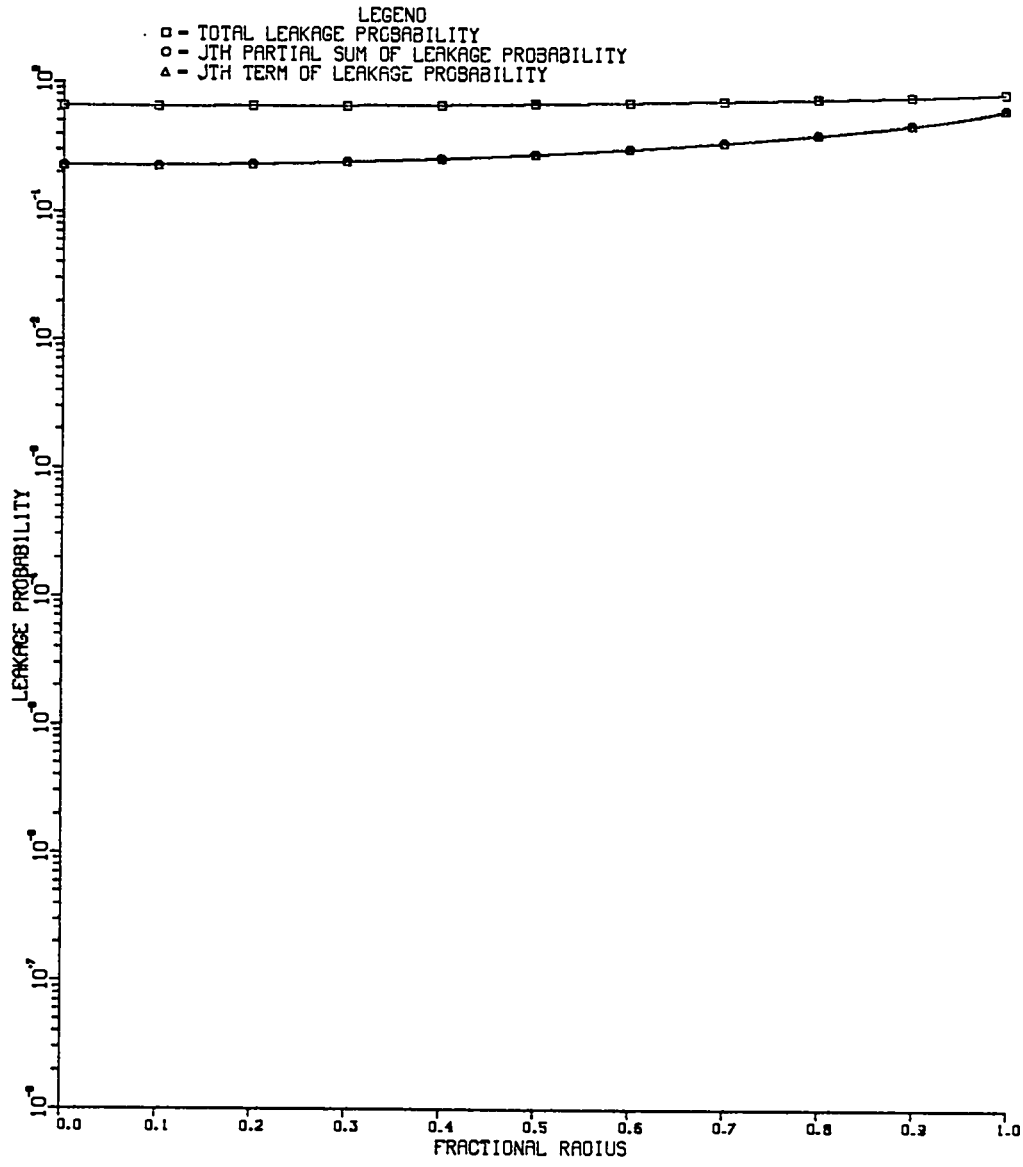


Figure 40.

Plot of the total leakage probability, 0th partial sum of the leakage probability, and 0th term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

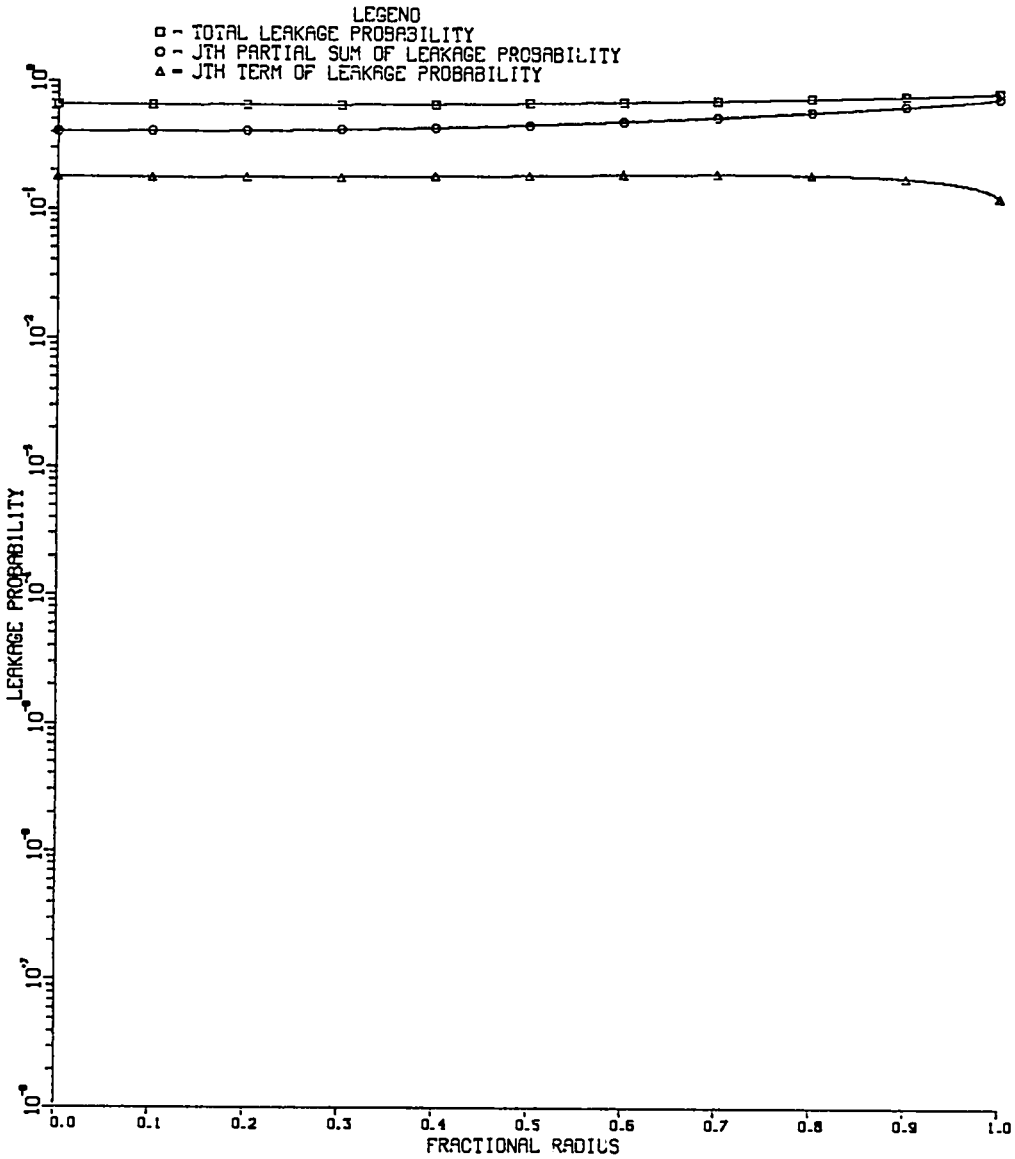


Figure 41.

Plot of the total leakage probability, 1st partial sum of the leakage probability, and 1st term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

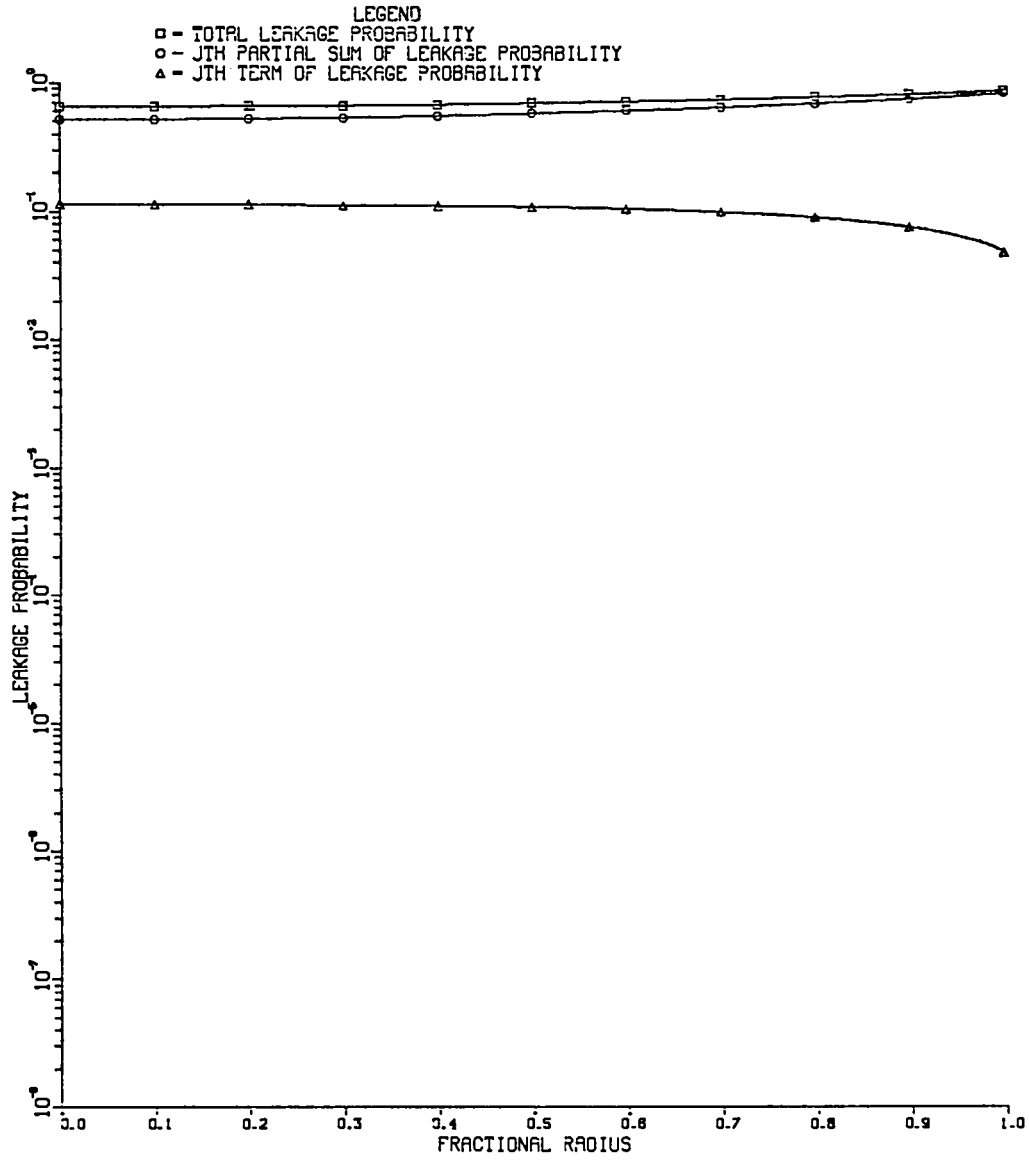


Figure 42.

Plot of the total leakage probability, 2nd partial sum of the leakage probability, and 2nd term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

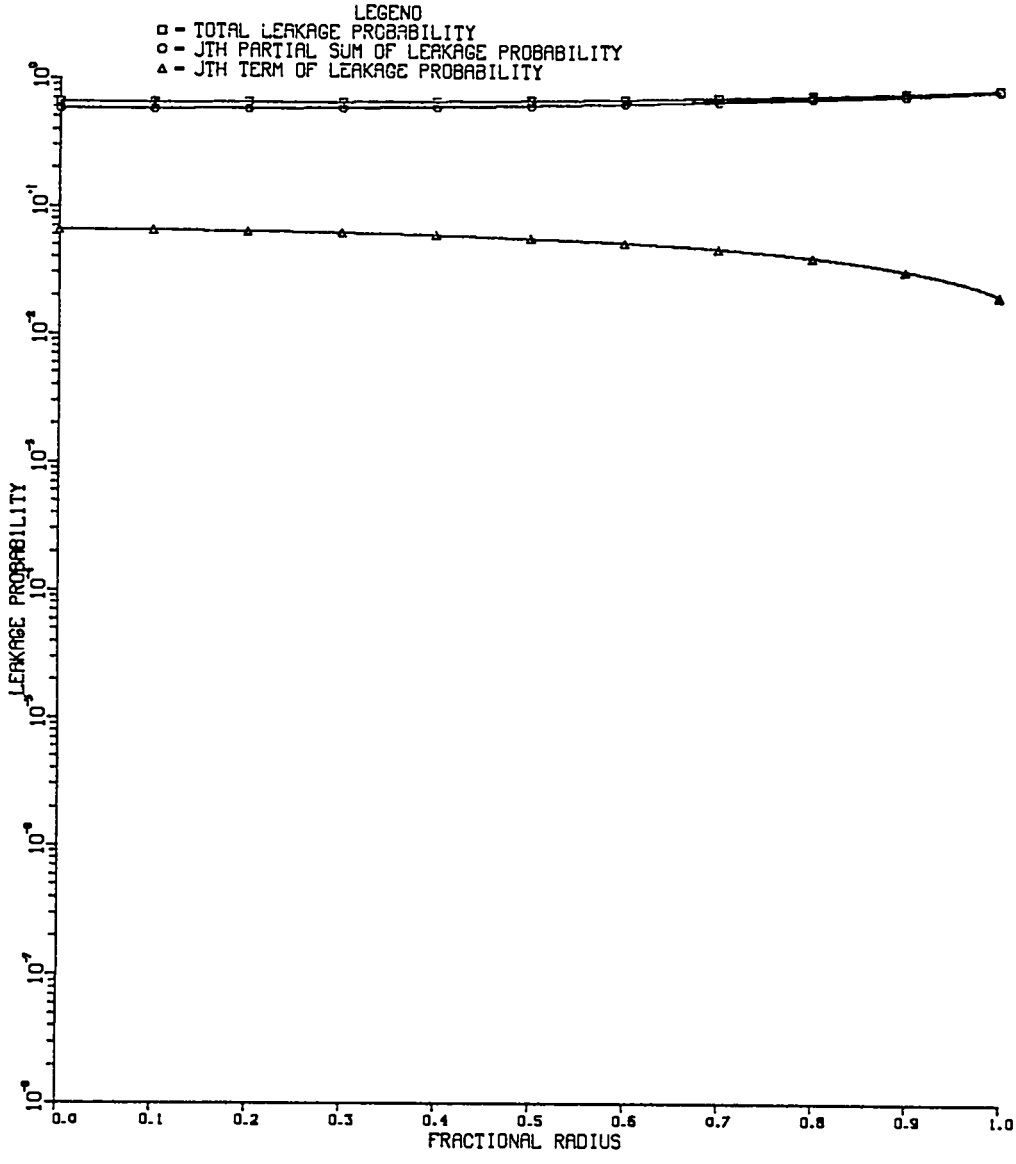


Figure 43.

Plot of the total leakage probability, 3rd partial sum of the leakage probability, and 3rd term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

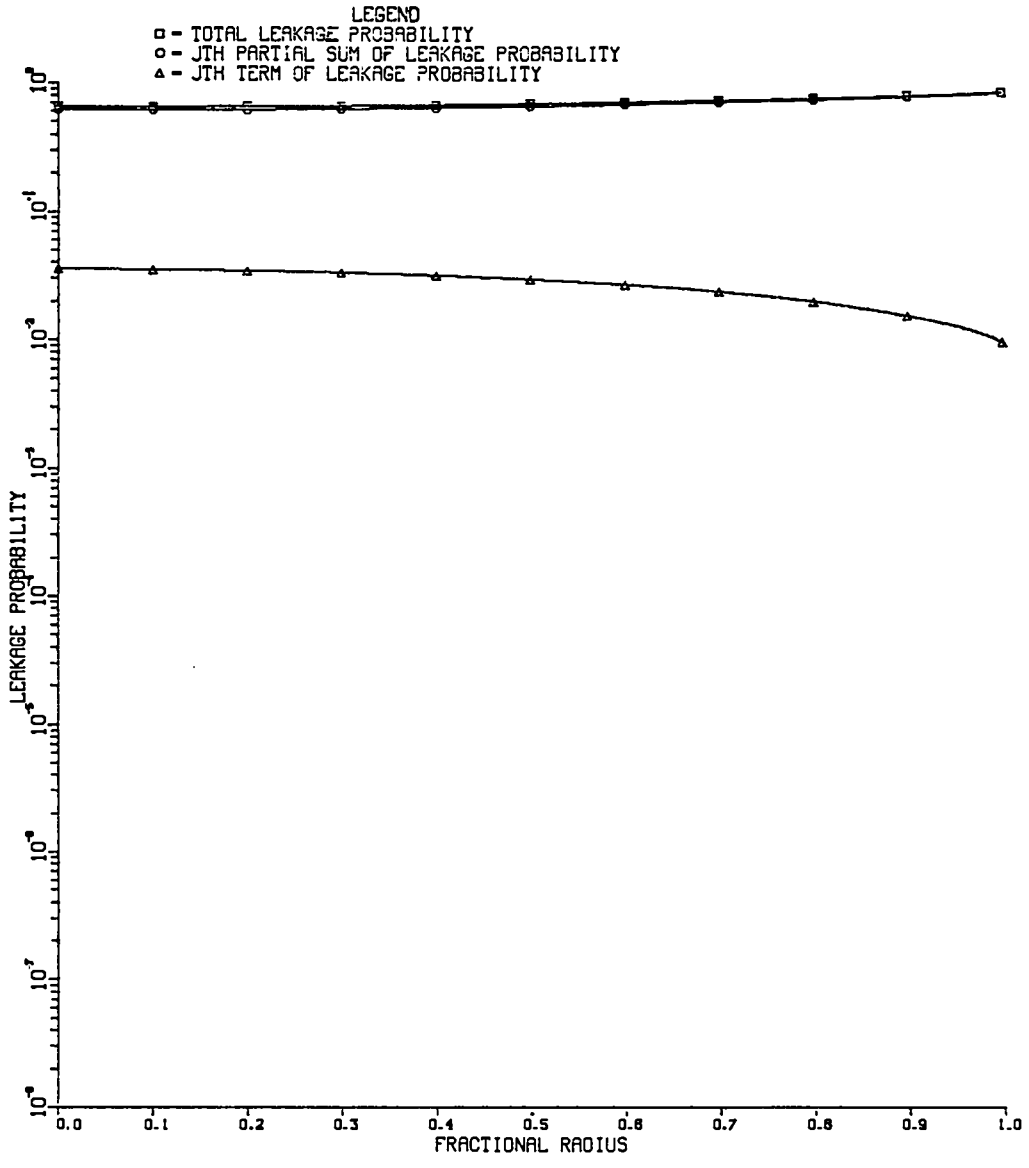


Figure 44.

Plot of the total leakage probability, 4th partial sum of the leakage probability, and 4th term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

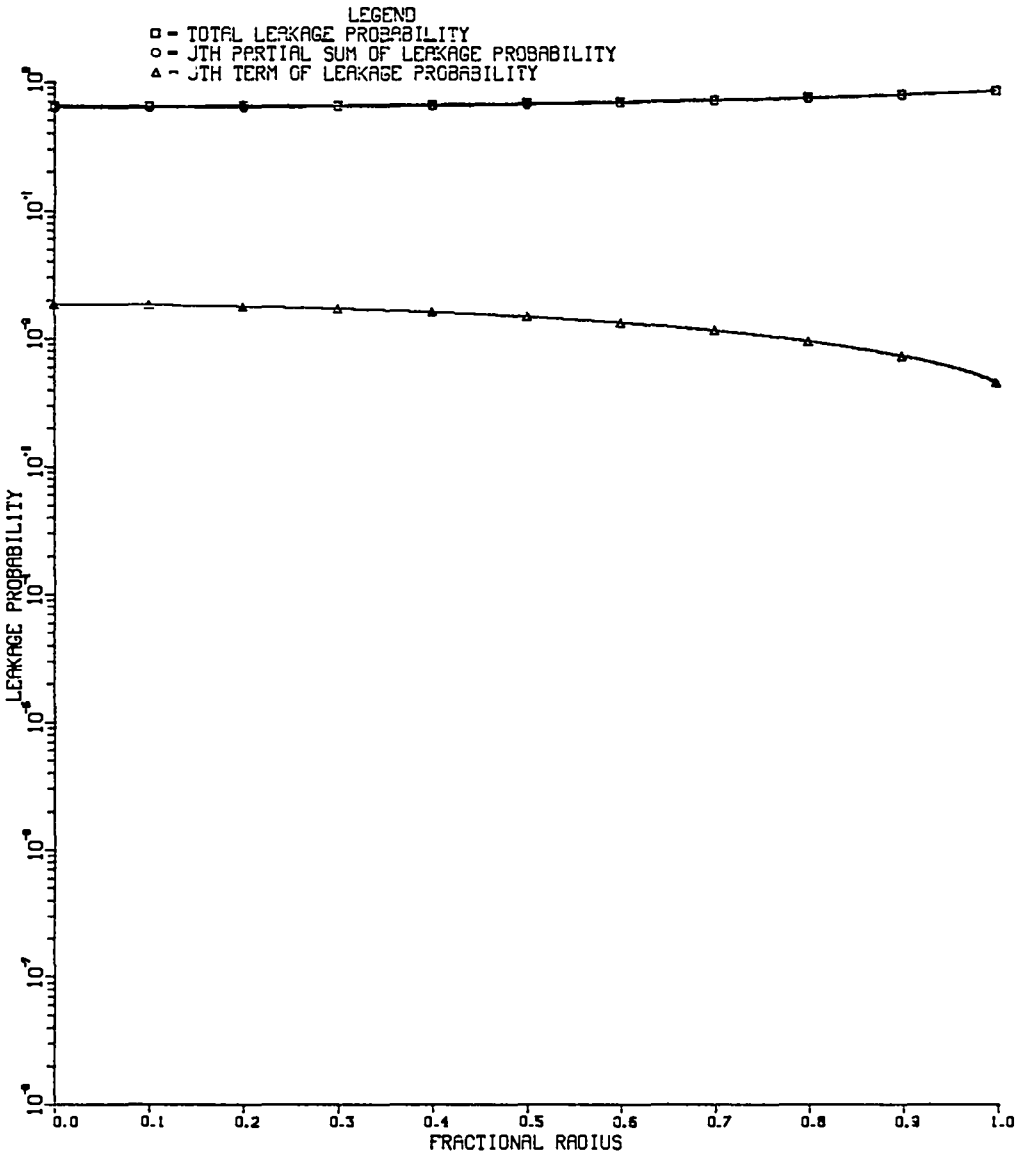


Figure 45.

Plot of the total leakage probability, 5th partial sum of the leakage probability, and 5th term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

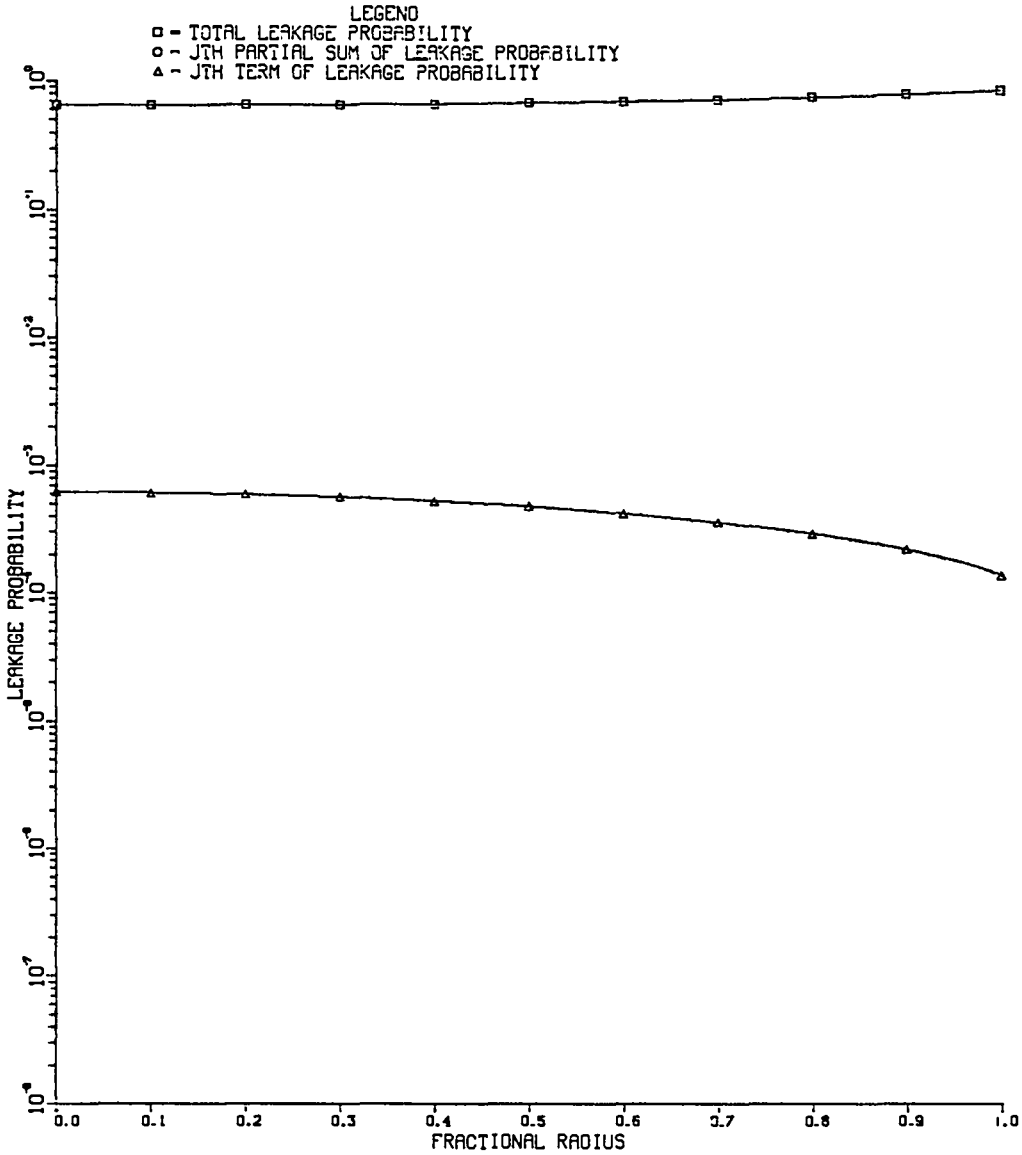


Figure 46.

Plot of the total leakage probability, 10th partial sum of the leakage probability, and 10th term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .8\sigma_t$.

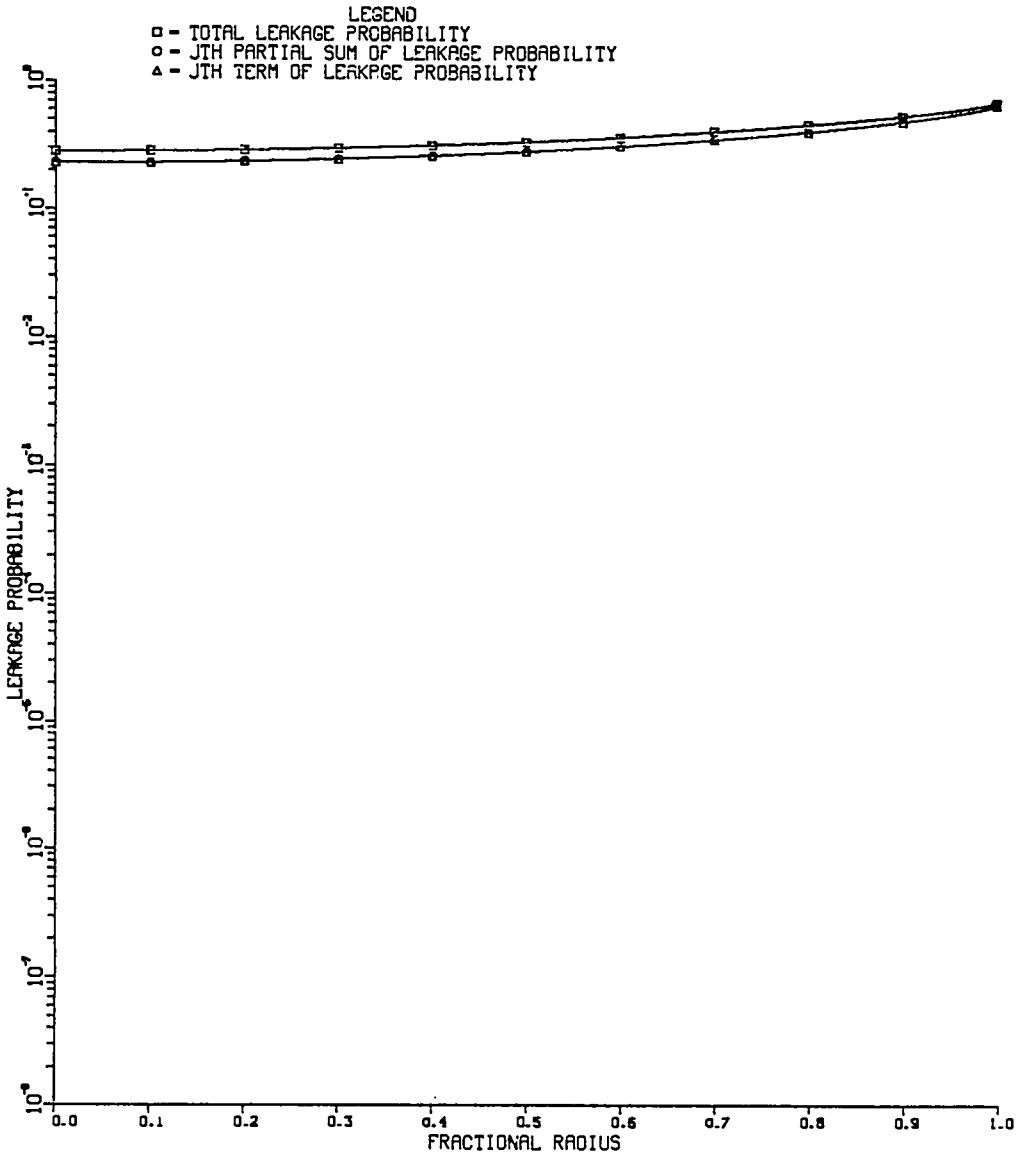


Figure 47.

Plot of the total leakage probability, 0th partial sum of the leakage probability, and 0th term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .2\sigma_t$.

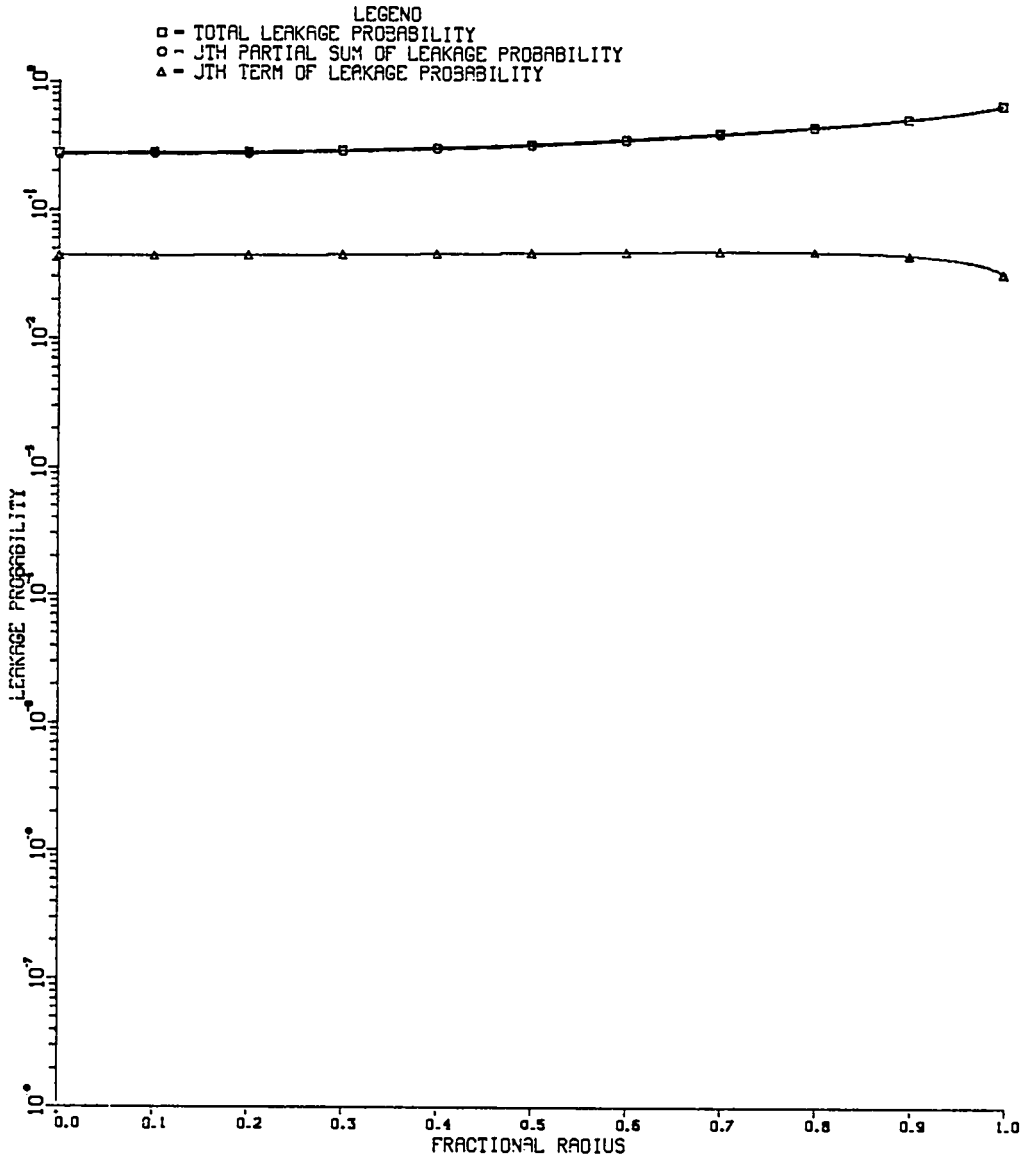


Figure 48.

Plot of the total leakage probability, 1st partial sum of the leakage probability, and 1st term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .2\sigma_t$.

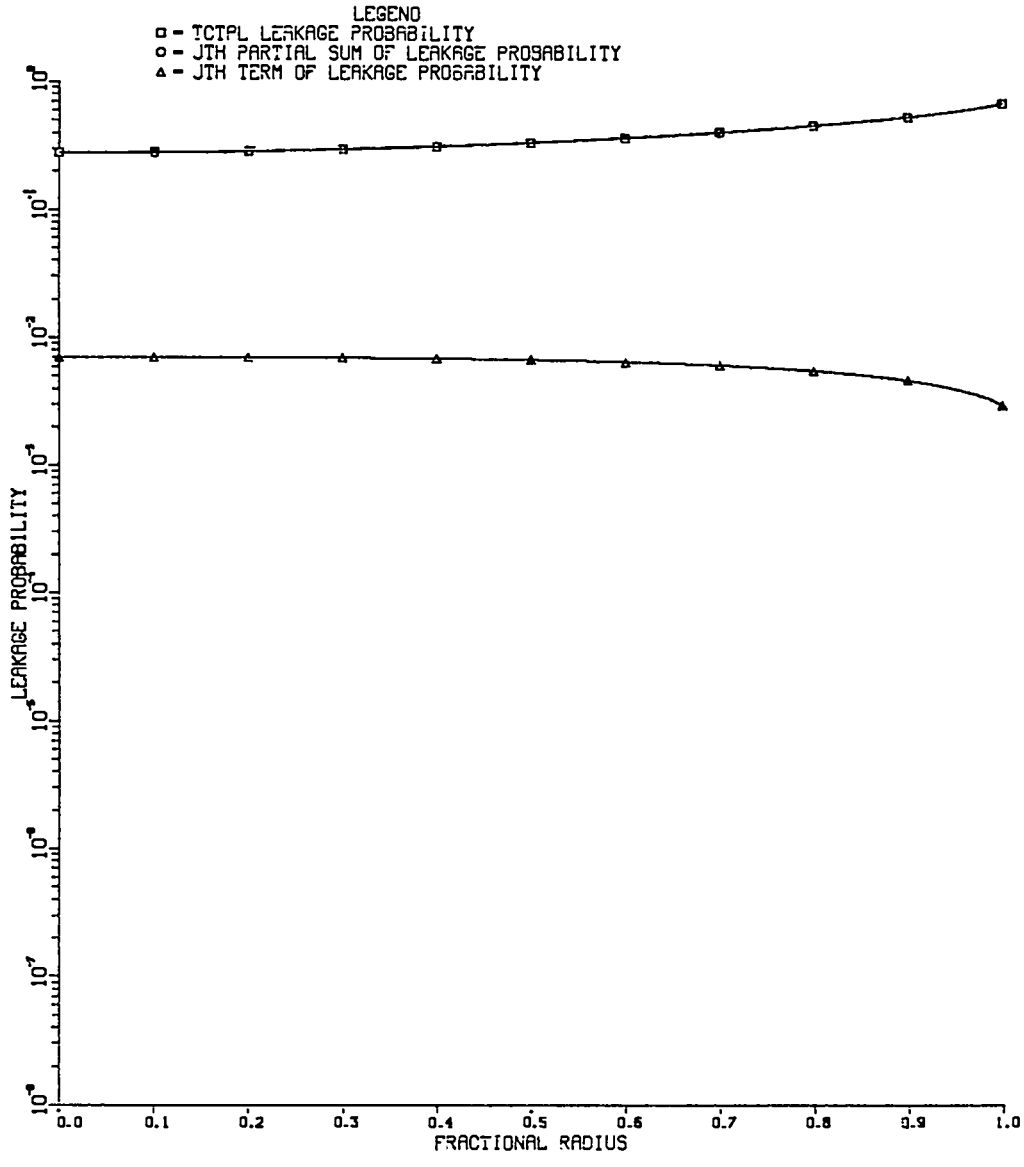


Figure 49.

Plot of the total leakage probability, 2nd partial sum of the leakage probability, and 2nd term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .2\sigma_t$.

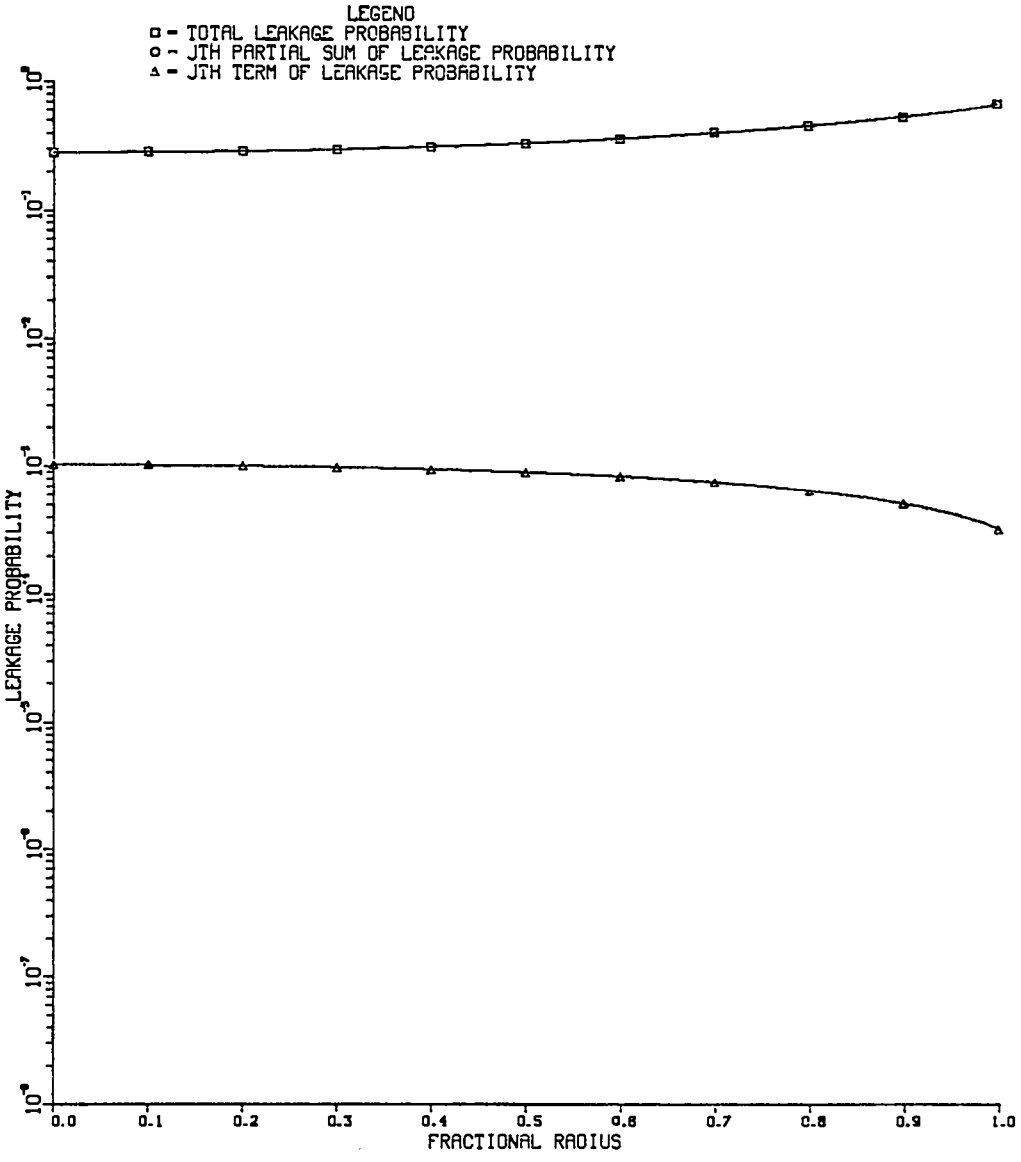


Figure 50.

Plot of the total leakage probability, 3rd partial sum of the leakage probability, and 3rd term of the leakage probability versus fractional radius for c=1.6, and $\sigma_s = .2\sigma_t$.

V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

A. Variational Techniques

Although extended in this thesis to cover V_4 slab and V_4 cylinder calculations and standarized in form, the variational technique will require additional work to exploit its precision. Currently, the V_4 calculation will yield a critical radius estimate with a minimum of four significant digits of precision. The V_2 calculation provides a minimum of two significant digits of precision. With these as a rough guide, the variational technique would require a V_8 calculation to match the eight significant digits of precision claimed by Kaper, Leaf, and Lindeman[10] in their "benchmark" calculation. Such an extension would be worthwhile for at least two reasons. First, an independent check would be provided for the work in Ref. 10, which is limited to slabs and spheres, and, second, very accurate benchmark critical radii estimates would be obtainable for cylindrical geometry. Currently, the V_4 cylindrical critical radii estimates presented herein equal or exceed the accuracy of any other results available.

In such an extension of the precision of the V_n calculations, further standardization would be of great utility. This standardization would complete the process begun in Chapter Two by deriving a general form for the coefficients of the parameters to appear in (2.8). Currently, each coefficient is individually determined through tedious and complex integration. Such a general term would enable the precision of the V_n technique to be extended to an arbitrary degree.

Also of use to the proposed extension of the V_n technique to higher order would be a more efficient algorithm for the solution of the algebraic equations resulting from the minimization of (2.8). The current technique, while usable, would result in large computational penalties. Although such penalties are acceptable in order to extend the accuracy of critical size estimates, an improved technique is needed.

B. Direct Leakage Operator

The direct leakage operator has been shown to provide an efficient method for calculating leakages. When used in a neutron balance condition with a diffusion theory scalar flux, the direct leakage operator resulted in a reduction of up to forty percent in the error in the critical radii estimate with respect to V_4 calculations. The next obvious step would be to incorporate the direct leakage operator into a neutron balance condition in an S_n code. Such a step would enable the quantification of improvements in criticality calculations. This, of course, would require the reformulation of the direct leakage operator in a discrete form rather than the current analytic form. Subsequent extensions would incorporate anisotropic scattering, energy dependence, multiregion spaces, and time dependence. These extensions, while time consuming, can be carried through.

An interesting extension is the inclusion of the direct leakage operator into the boundary conditions. The result of such an application of the direct leakage operator would, hopefully, be a reduction in the magnitude of the boundary perturbations caused by

inadequate satisfaction of boundary conditions. S_n techniques are inherently underspecified, and such an inclusion would not necessarily overspecify the system or replace boundary conditions. More likely, it would reduce the use of artificial relations such as the diamond difference scheme. There is, however, the possibility that the direct leakage operator could replace some or all of the boundary conditions. The work done at present does not justify the exclusion of any option.

C. Indirect Leakage Operation

Of most relevance to the extension of the indirect leakage calculation is a more efficient algorithm for the solution of the inhomogeneous singular Fredholm integral equation of the second kind. In any large production code, namely, a code sophisticated enough to handle several problems and initiated through the reading of an input deck, it is almost certain that the equation will be solved numerically. This implies a numerical kernel of large array size, i.e., 250,000 values. The input/output memory charges for such an array sustain large time penalties and are the chief reason for the cylindrical indirect leakage codes being considerably less efficient than the slab and sphere indirect leakage codes. Since, in general, inhomogeneous integral equations are easier to solve than their homogeneous counterparts, and, since the kernel for the indirect leakage equation is the same as the kernel for the integral transport equation, the first choice for a more efficient algorithm would seem to be in the S_n neutronics schemes[19]. Subsequent extensions would include anisotropic scattering, energy dependence, multiregion spaces, and time dependence. The extension to

anisotropic scattering could be of great significance to anisotropic integral transport theory.

D. Summary of Thesis

To recapitulate, the work in this thesis can be divided into three main sections. The first section standardized V_n computations, increased their scope to include V_4 slab and V_4 cylinder, and obtained precise critical radii estimates heretofore unavailible in cylindrical geometry. Also, a derivation of the cylindrical integral transport equation was presented which will complement the derivations present in the literature for slab and sphere. In the second section, a direct leakage operator was derived and proved exact. Subsequent application of that operator in eigenvalue problems resulted in substantial accuracy improvements for diffusion theory eigenvalues. In the third section, an indirect method of looking at the leakage was developed. The governing equation was shown to have the same kernel as the appropriate integral transport equation. The accuracy of the approach was proved, and detailed calculations of the spatial leakage probability in spherical geometry were performed. These calculations, unique to this thesis, showed surprisingly high orders of the scattering multiplicity need to be considered for accurate leakage calculations.

ACKNOWLEDGEMENTS

I wish to acknowledge the assistance of three people in the preparation of this manuscript: Professor R. A. Axford, my thesis advisor, who provided financial support as well as aid in the conceptual development; Dr. R. D. Johnson, who assisted greatly in the analytic development; and Georgia Pedicini, my wife, who did the typing.

I also wish to acknowledge the facilities at the Los Alamos National Laboratory which made this work possible.

REFERENCES

1. G. I. Bell and S. Glasstone, Nuclear Reactor Theory (Van Nostrand Reinold, New York, 1970).

2. M. H. L. Pryce, "Critical Conditions in Neutron Multiplication ", United Kingdom Ministry of Supply report, MSP-2A (1947).

3. B. Davison and J. B. Sykes, Neutron Transport Theory (Oxford University Press, London, 1957).

4. B. G. Carlson and G. I. Bell, "Solution of the Transport Equation by the Sn Method", Proceedings of the Second United Nations International Conference on the Peaceful Uses of Atomic Energy (1958), Vol. 16.

5. S. Frankel and S. Goldberg, "The Mathematical Development of the Endpoint Method", Los Alamos Scientific Laboratory report, LA-258 (1945).

6. K. M. Case and P. F. Zweifel, Linear Transport Theory (Addison-Wesley Publishing Company, Palo Alto, 1967).

7. A. M. Weinberg and E. P. Wigner, The Physical Theory of Neutron Chain Reactors (The University of Chicago Press, Chicago, 1958).

8. S. Frankel and E. Nelson, "Methods of Treatment of Displacement Integral Equations", Atomic Energy Commission report, AECD-3497 (1953).

9. G. J. Mitsis, "Transport Solutions to the One-Dimensional Critical Problem", Nuclear Science and Engineering, Vol. 17, 55 (1963).

10. H. G. Kaper, A. J. Lindeman, and G. K. Leaf, "Benchmark Values for the Slab and Sphere Criticality Problem in One-Group Neutron Transport Theory", Nuclear Science and Engineering, Vol. 54, 94 (1974).

11. H. Hembd, "The Integral Transform Method for Neutron Transport Problems", Nuclear Science and Engineering, Vol. 40, 224 (1970).

12. M. S. Milgram, "Analytic Method for the Numerical Solution of the Integral Transport Equation for a Homogeneous Cylinder", Nuclear Science and Engineering, Vol 68, 249 (1978).

13. R. Sanchez, "Transport Calculations in Cylindrical Geometry", Nuclear Science and Engineering, Vol. 70, 318 (1979).

14. E. B. Dahl and N. G. Sjöstrand, "Eigenvalue Spectrum of Multiplying Slabs and Spheres for Monoenergetic Neutrons with Anisotropic Scattering", Nuclear Science and Engineering, Vol. 69, 114 (1979).

15. R. Serber, "A Graphical Representation of Critical Masses and Multiplication Rates", Los Alamos Scientific Laboratory report, LA-234 (1945).

16. K. M. Case, "Multimedia Critical Mass Problems", Los Alamos Scientific Laboratory report, LA-247 (1945).

17. B. Carlson, "The Serber Wilson Method, Formulae and Computation Methods", Los Alamos Scientific Laboratory report, LA-756 (1949).

18. B. Carlson, "Neutron Diffusion-Spherical Harmonics Theory", Los Alamos Scientific Laboratory report, LA-571 (1946).

19. H. Greenspan, C. N. Kelber, and D. Okrent, Computing Methods in Reactor Physics (Gordon and Breach, New York, 1968).

20. C. Syros and P. Theocharopoulos, "Structural Polynomial Solution of the Transport Equation for Critical and Noncritical Slabs with Anisotropic Scattering", Annals of Nuclear Energy, Vol. 4, 495 (1977).

21. C. Syros and P. Theocharopoulos, "The Influence of Boundary Conditions on the Precision of the Eigenvalues of the Boltzmann Equation", Nuclear Science and Engineering, Vol. 73, 108 (1980).

22. R. T. Ackroyd, "A Finite Element Method for Neutron Transport-I. Some Theoretical Considerations", Annals of Nuclear Energy, Vol. 5, 75 (1978).

23. J. Galliara and M. M. R. Williams, "A Finite Element Method for Neutron Transport-II. Some Practical Considerations", Annals of Nuclear Energy, Vol. 6, 205 (1979).

24. R. T. Ackroyd and D. T. Grenfell, "A Finite Element Method for Neutron Transport-III. Two-Dimensional One-Group Test Problems", Annals of Nuclear Energy, Vol. 6, 563 (1979).

25. B. A. Splawski, A. K. Ziver, and J. Galliara, "Using Orthogonal Functions with a Finite Element Method for Approximating Even Parity Neutron Flux in Slab Geometry", Nuclear Science and Engineering, Vol. 77, 351 (1981).

26. B. G. Carlson, "Solution of the Transport Equation by Sn Approximations", Los Alamos Scientific Laboratory report, LA-1599 (1953).

27. T. J. Seed, W. F. Miller Jr., and F. W. Brinkley Jr., "TRIDENT:A Two-Dimensional Multigroup, Triangular Mesh Discrete Ordinates, Explicit Neutron Transport Code", Los Alamos Scientific Laboratory report, LA-6735M (1977).

28. K. M. Case, F. de Hoffman, and G. Placzek, Introduction to the Theory of Neutron Diffusion (Los Alamos Scientific Laboratory, Los Alamos, 1953).

29. P. A. M. Dirac, "Approximate Rate of Neutron Multiplication for a Solid of Arbitrary Shape and Uniform Density", United Kingdom Report, MS-D-5 (1943).

30. Y. A. Chao and A. S. Martinez, "On Approximations to the Neutron Escape Probability from an Absorbing Body", Nuclear Science and Engineering, Vol. 66, 254 (1978).

31. I. Lux and I. Vidovszky, "Approximation to Neutron Escape Probability for Slab and Cylinder", Nuclear Science and Engineering, Vol. 69, 442 (1979).

32. Y. A. Chao, "Reply to the Comments by Lux and Vidovszky on an Approximation to Neutron Escape Probability", Nuclear Science and Engineering, Vol. 69, 443 (1979).

33. H. P. Raghav, "Polynomial Expression for the Neutron Escape Probability from an Absorbing Body", Nuclear Science and Engineering, Vol. 73, 302 (1980).

34. Y. A. Chao, "Reply to 'Polynomial Expression for the Neutron Escape Probability from an Absorbing Body'", Nuclear Science and Engineering, Vol. 73, 304 (1980).

35. D. Kwiat, "An Improved Single Parameter Escape Probability Function", Nuclear Science and Engineering, Vol. 76, 255 (1980).

36. E. B. Dahl and N. G. Sjöstrand, "Reply to 'The Influence of Boundary Conditions on the Precision of the Eigenvalues of the Boltzmann Equation'", Nuclear Science and Engineering, Vol. 73, 109 (1980).

37. R. Courant and D. Hilbert, Methods of Mathematical Physics (Interscience Publishers Inc., New York, 1953).

38. A. F. Henry, Nuclear Reactor Analysis (The MIT Press, Cambridge, 1975).

39. Y. Ronen, D. Shvarts, and J. J. Wagschal, "A Comparison of Some Eigenvalues in Reactor Theory", Nuclear Science and Engineering, Vol. 60, 97 (1976).

40. W. G. Bickley and J. Naylor, "A Short Table of the Functions Kin(x), from n=1 to n=16", Philosophical Magazine, Vol. 7, 343 (1935).

41. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (National Bureau of Standards, U. S. Government Printing Office, 1964).

42. E. Erdélyi, W. Magnus, F. Oberhettinger, F. G. Tricomi, Higher Transcendental Functions Volume II (McGraw-Hill, New York, 1953).

43. G. B. Thomas, Calculus and Analytic Geometry Alternate Edition (Addison-Wesley, Reading, 1972).

44. W. Kaplan, Advanced Calculus Second Edition (Addison-Wesley, Reading, 1973).

APPENDIX A

THE EXPONENTIAL INTEGRAL FUNCTIONS

The exponential integral function is defined by Abramowitz and Stegun[41] as

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}\, dt, \quad (n=0,1,2,\ldots). \tag{A.1}$$

Making the substitution $t = \frac{1}{u}$ into (A.1) yields

$$E_n(x) = \int_0^1 u^{n-2}\, e^{\frac{-x}{u}}\, du, \quad (n=0,1,2,\ldots). \tag{A.2}$$

Making the substitution $u = \frac{x}{a}$ into (A.2) yields

$$E_n(x) = x^{n-1}\int_x^\infty a^{-n}\, e^{-a}\, da, \quad (n=0,1,2,\ldots). \tag{A.3}$$

For this thesis the most useful form is (A.2). The values of $E_n(x)$ used in this thesis were generated by using a system numerical quadrature routine to evaluate (A.2). A separate test program was set up to evaluate the accuracy of the quadrature routine against the values of $E_n(x)$ tabulated in Abramowitz and Stegun[41]. The quadrature routine was accurate to the eight significant digits tabulated for all orders and magnitudes of arguments. In addition, the system library contains $E_1(x)$ as a callable function.

The derivatives of the $E_n(x)$ can be deduced from (A.1) to be

$$\frac{d}{dx}E_n(x) = \int_1^\infty - \frac{te^{-xt}}{t^n}\,dt = -E_{n-1}(x), \quad (n=1,2,3,\ldots). \tag{A.4}$$

Also of use for the reduction of all higher order exponential integrals is the following recurrence relation from Abramowitz and Stegun[41]:

$$E_{n+1}(x) = \frac{1}{n}\left[e^{-x}-xE_n(x)\right], \quad (n=1,2,3,\ldots). \tag{A.5}$$

APPENDIX B

RELATIONS AMONG MODIFIED BESSEL FUNCTIONS

The following integral and derivative formulas among the modified Bessel functions of the first kind (I_n) and of the third kind (K_n, which are alternatively known as modified Bessel functions of the second kind) are from Erdélyi[42]:

$$\int x^{n+1} I_n(x)dx = x^{n+1} I_{n+1}(x), \tag{B.1a}$$

$$\int x^{n+1} K_n(x)dx = -x^{n+1} K_{n+1}(x); \tag{B.1b}$$

$$\left(\frac{1}{x}\frac{d}{dx}\right)^m [x^n I_n(x)] = x^{n-m} I_{n-m}(x), \tag{B.2a}$$

$$\left(\frac{1}{x}\frac{d}{dx}\right)^m [x^n K_n(x)] = (-1)^m x^{n-m} K_{n-m}(x). \tag{B.2b}$$

Also from Erdélyi[42] are the following recurrence relations among the modified Bessel functions of various orders with identical arguments:

$$I_{n-1}(x) - I_{n+1}(x) = 2nx^{-1} I_n(x), \tag{B.3a}$$

$$K_{n-1}(x) - K_{n+1}(x) = -2nx^{-1} K_n(x). \tag{B.3b}$$

In the following integral formula, v_ν and V_μ are any modified Bessel functions of the first or third kind. The primes denote partial differentiation with respect to the arguments[42]:

$$\int\left[(\beta^2-\alpha^2)x+\frac{(\mu^2-\nu^2)}{x}\right]v_\nu(\alpha x)V_\mu(\beta x)dx$$

$$= x\left[-\alpha V_\mu(\beta x)v'_\nu(\alpha x)+\beta v_\nu(\alpha x)V'_\mu(\beta x)\right]. \qquad \text{(B.4)}$$

Asymptotic series for large values of the arguments, which are crucial to portions of this thesis, are from Abramowitz and Stegun[41].

$$I_n(x) \simeq \frac{e^x}{(2\pi x)^{.5}}\left[1-\frac{4n^2-1}{8x}+O(x^{-2})\right], \qquad \text{(B.5a)}$$

$$K_m(z) \simeq \left(\frac{\pi}{2z}\right)^{.5} e^{-z}\left[1+\frac{4m^2-1}{8z}+O(z^{-2})\right]; \qquad \text{(B.5b)}$$

therefore,

$$I_n(x)K_m(z) \simeq \frac{e^{-(z-x)}}{2(zx)^{.5}}\left[1+\frac{4m^2-1}{8z}-\frac{4n^2-1}{8x}\right.$$

$$\left.-\frac{(16m^2n^2-4m^2-4n^2+1)}{64zx}+O(x^{-2},z^{-2})\right]. \qquad \text{(B.6)}$$

Also from Abramowitz and Stegun[41] are the following relations among the negative and positive integer order modified Bessel functions of the first and third kind:

$$I_{-n}(x) = I_n(x), \tag{B.7a}$$

$$K_{-n}(x) = K_n(x) \tag{B.7b}$$

Frequently in this thesis integrals of the following form will be required:

$$\int d\Theta K_0[y(r'^2+r^2-2rr'\cos(\Theta))^{.5}]. \tag{B.8}$$

The only way the author has found to evaluate (B.8) is to expand K_0 by modifying the Graf's addition theorem. The Graf's theorem in (B.9) is from Abramowitz and Stegun[41].

$$\zeta_\nu(w) \begin{matrix}\cos\\ \sin\end{matrix}(\nu x) = \sum_{k=-\infty}^{\infty} \zeta_{\nu+k}(u)J_k(v)\begin{matrix}\cos\\ \sin\end{matrix}(k\alpha) \tag{B.9a}$$

where

$$|ve^{\pm i\alpha}| < |u|, \tag{B.9b}$$

$$w = (u^2+v^2-2uv\cos(\alpha))^{.5}, \tag{B.9c}$$

$$u-v\cos(\alpha) = w\cos(\chi), \tag{B.9d}$$

and

$$v\sin(\alpha) = x\sin(\chi). \tag{B.9e}$$

The ζ_ν are Bessel functions of the first kind $(J_n(x))$, of the second kind $(Y_n(x)$ or Weber's functions$)$, or of the third kind $(H_n^{(1)}(x)$ and $H_n^{(2)}(x)$ or first and second Hankel functions$)$.

Rewrite (B.9) for a zero order first Hankel function with $\alpha=0$ as

$$H_0^{(1)}[(u^2+v^2-2uv\cos(\theta))^{.5}] = \sum_{k=-\infty}^{\infty} H_k^{(1)}(u)J_k(v)\cos(k\theta). \qquad (B.10)$$

Now substitute

$$u=yre^{.5i\pi}, \quad v=yr'e^{.5i\pi}, \quad r'<r, \qquad (B.11)$$

and

$$u=yr'e^{.5i\pi}, \quad v=yre^{.5i\pi}, \quad r'>r, \qquad (B.12)$$

into (B.10) to get

$$H_0^{(1)}[e^{.5i\pi}y(r'^2+r^2-2rr'\cos(\theta))^{.5}] = \sum_{k=-\infty}^{\infty} H_k^{(1)}(e^{.5i\pi}yr)$$

$$J_k(e^{.5i\pi}yr')\cos(k\theta) \quad \text{for } r'<r \qquad (B.13)$$

and

$$H_0^{(1)}[e^{.5i\pi}y(r'^2+r^2-2rr'\cos(\theta))^{.5}] = \sum_{k=-\infty}^{\infty} H_k^{(1)}(e^{.5i\pi}yr')$$

$$J_k(e^{.5i\pi}yr)\cos(k\theta) \quad \text{for } r'>r. \qquad (B.14)$$

Multiply both sides of (B.13) and (B.14) by

$$(\tfrac{1}{2}\pi i)(e^{.5ki\pi})(e^{-.5ki\pi}) \qquad (B.15)$$

to get

$$\frac{1}{2}\pi i H_0^{(1)}\left[e^{.5i\pi y(r'^2+r^2-2rr'\cos(\theta))^{.5}}\right]$$

$$= \sum_{k=-\infty}^{\infty} \frac{1}{2}\pi i e^{.5ki\pi} H_k^{(1)}\left(e^{.5i\pi yr}\right)$$

$$e^{-.5k\pi i} J_k\left(e^{.5i\pi yr'}\right)\cos(k\theta) \quad \text{for } r'<r \tag{B.16}$$

and

$$\frac{1}{2}\pi i H_0^{(1)}\left[e^{.5i\pi y(r'^2+r^2-2rr'\cos(\theta))^{.5}}\right]$$

$$= \sum_{k=-\infty}^{\infty} \frac{1}{2}\pi i e^{.5k\pi i} H_k^{(1)}\left(e^{.5i\pi yr'}\right)$$

$$e^{-.5k\pi i} J_k\left(e^{.5i\pi yr}\right)\cos(k\theta) \quad \text{for } r'>r. \tag{B.17}$$

Now use the following relations[41] between normal and modified Bessel functions,

$$I_k(z) = e^{-.5k\pi i} J_k\left(ze^{.5\pi i}\right) \tag{B.18}$$

and

$$K_k(z) = \frac{1}{2}\pi i e^{.5k\pi i} H_k^{(1)}\left(ze^{.5\pi i}\right), \tag{B.19}$$

to get from (B.16) and (B.17) to (B.20):

$$K_0[y(r'^2+r^2-2rr'\cos(\theta))^{.5}] = \sum_{k=-\infty}^{\infty} K_k(yr)I_k(yr')\cos(k\theta)$$

for $r'<r$, (B.20a)

$$K_0[y(r'^2+r^2-2rr'\cos(\theta))^{.5}] = \sum_{k=-\infty}^{\infty} K_k(yr')I_k(yr)\cos(k\theta)$$

for $r'>r$. (B.20b)

The expansion in (B.20) will be used to perform integrals of the form of (B.8) whenever they occur in this thesis.

APPENDIX C

CYLINDRICAL INTEGRAL TRANSPORT EQUATION

Start with the monoenergetic, steady state, general geometry integral Boltzmann transport equation with spatially constant cross sections, isotropic fission, isotropic scattering, no external sources, and the total macroscopic cross section normalized to unity[1], viz.,

$$\Phi(\underline{r}) = c\int_V \frac{e^{-R}}{4\pi R^2}\,\Phi(\underline{r}')dV'; \tag{C.1a}$$

$$R = |\underline{r} - \underline{r}'|, \tag{C.1b}$$

and insert the variables defined in Fig. C-1 and Fig. C-2, in which the geometry of a one-dimensional cylinder is projected upon two planes. One plane, Fig. C-1, is parallel to the cylinder axis and contains the endpoints of both the $\underline{r}$ and the $\underline{r}'$ position vectors. The other plane, Fig. C-2, is perpendicular to the cylinder axis at z equal to zero. Note the following relations;

$$R^2 = s^2+z^2, \tag{C.2}$$

$$s^2 = r^2+r'^2-2rr'\cos(\Theta), \tag{C.3}$$

$$dV' = r'd\Theta dr'dz. \tag{C.4}$$

Then, (C.1a) becomes

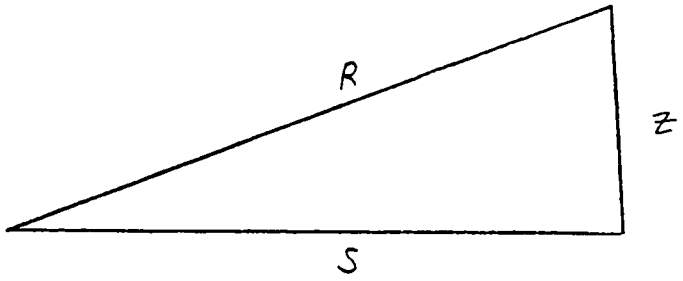


Figure C-1.

Projection of cylindrical coordinates

onto a plane parallel to the axis.

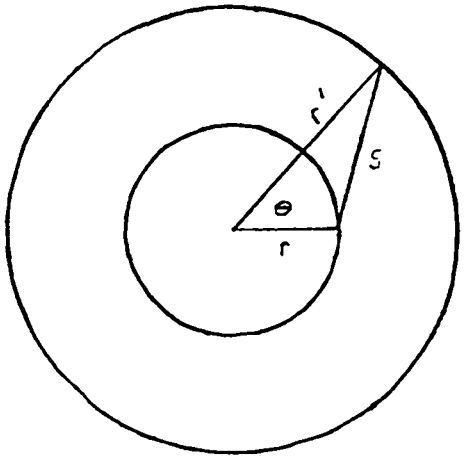


Figure C-2.

Projection of cylindrical coordinates

onto a plane perpendicular to the axis.

$$\Phi(r) = \int_0^b dr' r' \Phi(r') c \int_{-\infty}^{\infty} dz \int_0^{2\pi} d\theta \frac{e^{-(r^2+r'^2-2rr'\cos(\theta)+z^2)\cdot 5}}{4\pi(r^2+r'^2-2rr'\cos(\theta)+z^2)} . \quad (C.5)$$

Noting the symmetry in z and θ in (C.5), interchanging the order of integration, and using (C.3), yield

$$\Phi(r) = \frac{c}{\pi} \int_0^b dr' r' \Phi(r') \int_0^{\pi} d\theta \int_0^{\infty} dz \frac{e^{-s\left(1+\frac{z^2}{s^2}\right)\cdot 5}}{s^2\left(1+\frac{z^2}{s^2}\right)} . \quad (C.6)$$

With the transformation

$$z = s \sinh(a), \quad (C.7)$$

(C.6) becomes

$$\Phi(r) = \frac{c}{\pi} \int_0^b dr' r' \Phi(r') \int_0^{\pi} d\theta \int_0^{\infty} da \frac{1}{s} \frac{1}{\cosh(a)} e^{-s\cosh(a)} . \quad (C.8)$$

From Bickley and Naylor[40],

$$\int_0^{\infty} da \frac{1}{\cosh(a)} e^{-s\cosh(a)} = K_{i1}(s) = \int_s^{\infty} K_0(t) dt, \quad (C.9)$$

where $K_{i1}(x)$ is the first order Bickley function. Therefore, (C.8) becomes

$$\Phi(r) = \frac{c}{\pi}\int_0^b dr' r' \Phi(r') \int_0^\pi d\Theta \frac{1}{s} \int_s^\infty K_0(t)dt. \qquad (C.10)$$

Now let

$$t = ys, \qquad (C.11)$$

interchange the order of integration, and use (C.3) to obtain from (C.10)

$$\Phi(r) = \frac{c}{\pi}\int_0^b dr' r' \Phi(r') \int_1^\infty dy \int_0^\pi d\Theta K_0[y(r^2+r'^2-2rr'\cos(\Theta))^{.5}]. \qquad (C.12)$$

By using (B.20), the Graf's addition formula adapted to modified Bessel functions, (C.12) becomes

$$\Phi(r) = \frac{c}{\pi}\int_0^r dr' r' \Phi(r') \int_1^\infty dy \int_0^\pi d\Theta \sum_{k=-\infty}^{\infty} K_k(yr) I_k(yr') \cos(k\Theta)$$

$$+ \frac{c}{\pi}\int_r^b dr' r' \Phi(r') \int_1^\infty dy \int_0^\pi d\Theta \sum_{k=-\infty}^{\infty} K_k(yr') I_k(yr) \cos(k\Theta), \qquad (C.13)$$

which upon integration over Θ becomes

$$\Phi(r) = c\int_0^r dr' r' \Phi(r') \int_1^\infty dy K_0(yr) I_0(yr')$$

$$+ c\int_r^b dr' r' \Phi(r') \int_1^\infty dy K_0(yr') I_0(yr) \qquad (C.14)$$

or

$$\Phi(r) = c\int_0^b dr'r'\Phi(r')K(r,r'); \qquad \text{(C.15a)}$$

$$K(r,r') = \int_1^\infty dy K_0(yr)I_0(yr') \quad \text{for } r'<r, \qquad \text{(C.15b)}$$

$$K(r,r') = \int_1^\infty dy K_0(yr')I_0(yr) \quad \text{for } r'>r. \qquad \text{(C.15c)}$$

This is the Boltzmann integral transport equation for the scalar flux in infinite cylindrical geometry. From the starting equation (C.5) it can be seen that the kernel in (C.15) is real and symmetric.

APPENDIX D

VARIATIONAL IMPLEMENTATION IN SLAB GEOMETRY

The variational principle is

$$\frac{c}{2} = \frac{\int_{-b}^{b} dx \Phi^2(x)}{\int_{-b}^{b} dx \Phi(x) \int_{-b}^{b} dx' \Phi(x') E_1(|x-x'|)} \quad . \tag{D.1}$$

Trial functions of the form

$$\Phi(x) = 1 - a_1 x^2 - a_2 x^4, \tag{D.2}$$

where a_1 and a_2 are adjustable parameters, will be used to rewrite (D.1) as the generalized principle in (D.3):

$$\frac{c}{2} = \frac{TT1 a_1^2 + TT2 a_2^2 + TT3 a_1 a_2 + TT4 a_1 + TT5 a_2 + TT6}{BB1 a_1^2 + BB2 a_2^2 + BB3 a_1 a_2 + BB4 a_1 + BB5 a_2 + BB6} \quad . \tag{D.3}$$

The substitution of (D.2) into the numerator of (D.1) and subsequent integration yield the following definitions;

$$TT1 = \frac{2}{5} b^5, \tag{D.4a}$$

$$TT2 = \frac{2}{9} b^9, \tag{D.4b}$$

$$TT3 = \frac{4}{7} b^7, \tag{D.4c}$$

$$TT4 = -\frac{4}{3}b^3, \tag{D.4d}$$

$$TT5 = -\frac{4}{5}b^5, \tag{D.4e}$$

$$TT6 = 2b. \tag{D.4f}$$

Furthermore, after substitution of (D.2) into (D.1), the denominator of (D.1) can be rewritten as

$$\int_{-b}^{b} dx\Phi(x)\int_{-b}^{b} dx'\Phi(x')E_1(|x-x'|) = \int_{-b}^{b} dx\int_{-b}^{b} dx'E_1(|x-x'|)$$

$$-a_1\left[\int_{-b}^{b} dx\int_{-b}^{b} dx'x'^2E_1(|x-x'|) + \int_{-b}^{b} dxx^2\int_{-b}^{b} dx'E_1(|x-x'|)\right]$$

$$-a_2\left[\int_{-b}^{b} dx\int_{-b}^{b} dx'x'^4E_1(|x-x'|) + \int_{-b}^{b} dxx^4\int_{-b}^{b} dx'E_1(|x-x'|)\right]$$

$$+a_1a_2\left[\int_{-b}^{b} dxx^2\int_{-b}^{b} dx'x'^4E_1(|x-x'|)+\int_{-b}^{b} dxx^4\int_{-b}^{b} dx'x'^2E_1(|x-x'|)\right]$$

$$+a_1^2\left[\int_{-b}^{b} dxx^2\int_{-b}^{b} dx'x'^2E_1(|x-x'|)\right]$$

$$+a_2^2\left[\int_{-b}^{b} dxx^4\int_{-b}^{b} dx'x'^4E_1(|x-x'|)\right]. \tag{D.5}$$

Noting that $E_1(|x-x'|)$ is real and symmetric leads to the following definitions:

$$BB1 = \int_{-b}^{b} dxx^2\int_{-b}^{b} dx'x'^2E_1(|x-x'|), \tag{D.6a}$$

$$BB2 = \int_{-b}^{b} dx x^4 \int_{-b}^{b} dx' x'^4 E_1(|x-x'|), \tag{D.6b}$$

$$BB3 = 2\int_{-b}^{b} dx x^2 \int_{-b}^{b} dx' x'^4 E_1(|x-x'|), \tag{D.6c}$$

$$BB4 = -2\int_{-b}^{b} dx \int_{-b}^{b} dx' x'^2 E_1(|x-x'|), \tag{D.6d}$$

$$BB5 = -2\int_{-b}^{b} dx \int_{-b}^{b} dx' x'^4 E_1(|x-x'|), \tag{D.6e}$$

$$BB6 = \int_{-b}^{b} dx \int_{-b}^{b} dx' E_1(|x-x'|). \tag{D.6f}$$

In the integration of each of the definitions in (D.6) to closed form some intermediate integrals will prove convenient. Start with

$$\int_{-b}^{b} dx' E_1(|x-x'|),$$

and use (A.2), the definition of $E_n(x)$, to obtain

$$\int_{-b}^{b} dx' E_1(|x-x'|) = \int_{-b}^{b} dx' \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|x-x'|}{\mu}}. \tag{D.7}$$

Interchanging the order of integration yields

$$\int_{-b}^{b} dx' E_1(|x-x'|) = \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dx' e^{\frac{-|x-x'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \left[\int_{-b}^{x} dx' e^{\frac{-(x-x')}{\mu}} + \int_x^b dx' e^{\frac{-(x'-x)}{\mu}}\right], \tag{D.8}$$

therefore,

$$\int_{-b}^{b} dx' E_1(|x-x'|) = \int_0^1 d\mu \left[2 - e^{\frac{-(b+x)}{\mu}} - e^{\frac{-(b-x)}{\mu}}\right]. \qquad (D.9)$$

Similarly

$$\int_{-b}^{b} dx' x'^2 E_1(|x-x'|) = \int_{-b}^{b} dx' x'^2 \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|x-x'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dx' x'^2 e^{\frac{-|x-x'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \left[\int_{-b}^{x} dx' x'^2 e^{\frac{-(x-x')}{\mu}} + \int_x^b dx' x'^2 e^{\frac{-(x'-x)}{\mu}}\right], \qquad (D.10)$$

therefore,

$$\int_{-b}^{b} dx' x'^2 E_1(|x-x'|) = \int_0^1 d\mu \left[2x^2 + 4\mu^2\right.$$

$$\left. -(b^2+2\mu b+2\mu^2)\left(e^{\frac{-(b+x)}{\mu}} + e^{\frac{-(b-x)}{\mu}}\right)\right]. \qquad (D.11)$$

And again,

$$\int_{-b}^{b} dx' x'^4 E_1(|x-x'|) = \int_{-b}^{b} dx' x'^4 \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|x-x'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dx' x'^4 e^{\frac{-|x-x'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \left[\int_{-b}^{x} dx' x'^4 e^{\frac{-(x-x')}{\mu}} + \int_x^b dx' x'^4 e^{\frac{-(x'-x)}{\mu}}\right], \qquad (D.12)$$

therefore,

$$\int_{-b}^{b} dx' x'^4 E_1(|x-x'|) = \int_0^1 d\mu [2x^4+24\mu^2x^2+48\mu^4-(b^4+4\mu b^3$$

$$+12\mu^2b^2+24\mu^3b+24\mu^4)(e^{\frac{-(b+x)}{\mu}}+e^{\frac{-(b-x)}{\mu}})]. \qquad (D.13)$$

Start with the definition of BB1 in (D.6) and use (D.11) to obtain

$$BB1 = \int_{-b}^{b} dx x^2 \int_0^1 d\mu [2x^2+4\mu^2$$

$$-(b^2+2\mu b+2\mu^2)(e^{\frac{-(b+x)}{\mu}}+e^{\frac{-(b-x)}{\mu}})]. \qquad (D.14)$$

Interchange the order of integration and perform the integration over x to obtain

$$BB1 = \int_0^1 d\mu [\frac{4}{5} b^5+ \frac{8}{3} \mu^2b^3-2\mu b^4-8\mu^5$$

$$+e^{\frac{-2b}{\mu}}(2\mu b^4+8\mu^2b^3+16\mu^3b^2+16\mu^4b+8\mu^5)]. \qquad (D.15)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ:

$$BB1 = \frac{4}{5} b^5-b^4+ \frac{8}{9} b^3- \frac{4}{3} +2b^4E_3(2b)+8b^3E_4(2b)$$

$$+16b^2E_5(2b)+16bE_6(2b)+8E_7(2b). \qquad (D.16)$$

Use the definition of BB2 in (D.6) and (D.13) to obtain

$$BB2 = \int_{-b}^{b} dx x^4 \int_0^1 d\mu [2x^4+24\mu^2x^2+48\mu^4-(b^4+4\mu b^3+12\mu^2b^2$$

$$+24\mu^3b+24\mu^4)(e^{\frac{-(b+x)}{\mu}}+e^{\frac{-(b-x)}{\mu}})]. \qquad (D.17)$$

Interchange the order of integration and perform the integration over x to find

$$BB2 = \int_0^1 d\mu [\frac{4}{9} b^9+\frac{48}{7}\mu^2b^7+\frac{96}{5}\mu^4b^5-2\mu b^8-16\mu^3b^6$$

$$-1152\mu^9+e^{\frac{-2b}{\mu}}(1152\mu^9+2304\mu^8b+2304\mu^7b^2+1536\mu^6b^3$$

$$+768\mu^5b^4+288\mu^4b^5+80\mu^3b^6+16\mu^2b^7+2\mu b^8)]. \qquad (D.18)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integrations over μ; then it follows that

$$BB2 = \frac{4}{9} b^9-b^8+\frac{16}{7} b^7-4b^6+\frac{96}{25} b^5-115.2+2b^8E_3(2b)$$

$$+16b^7E_4(2b)+80b^6E_5(2b)+288b^5E_6(2b)+768b^4E_7(2b)$$

$$+1536b^3E_8(2b)+2304b^2E_9(2b)+2304bE_{10}(2b)+1152E_{11}(2b). \qquad (D.19)$$

Use the definition of BB3 in (D.6) and (D.13) to obtain

$$BB3 = 2\int_{-b}^{b} dx x^2 \int_0^1 d\mu [2x^4+24\mu^2x^2+48\mu^4-(b^4+4\mu b^3+12\mu^2b^2$$

$$+24\mu^3b+24\mu^4)(e^{\frac{-(b+x)}{\mu}}+e^{\frac{-(b-x)}{\mu}})]. \qquad (D.20)$$

Interchange the order of integration and perform the integration over x to find

$$BB3 = \int_0^1 d\mu [\frac{8}{7} b^7+\frac{96}{5}\mu^2b^5+64\mu^4b^3-4\mu b^6-8\mu^2b^5-24\mu^3b^4$$

$$-32\mu^4b^3-192\mu^7+e^{\frac{-2b}{\mu}}(4\mu b^6+24\mu^2b^5+88\mu^3b^4$$

$$+224\mu^4b^3+384\mu^5b^2+384\mu^6b+192\mu^7)]. \qquad (D.21)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integrations over μ to obtain

$$BB3 = \frac{8}{7} b^7-2b^6+\frac{56}{15} b^5-6b^4+\frac{32}{5} b^3-24+4b^6E_3(2b)$$

$$+24b^5E_4(2b)+88b^4E_5(2b)+224b^3E_6(2b)$$

$$+384b^2E_7(2b)+384bE_8(2b)+192E_9(2b) \qquad (D.22)$$

Use the definition of BB4 in (D.6) and (D.11) to obtain

$$BB4 = -2\int_{-b}^{b} dx \int_{0}^{1} d\mu \left[2x^2+4\mu^2-(b^2+2\mu b+2\mu^2) \right.$$

$$\left. \left(e^{\frac{-(b+x)}{\mu}} + e^{\frac{-(b-x)}{\mu}} \right) \right]. \qquad (D.23)$$

Interchange the order of integration and perform the integration over x to find

$$BB4 = -2\int_{0}^{1} d\mu \left[\frac{4}{3} b^3+8\mu^2 b-2\mu b^2-4\mu^2 b-4\mu^3 \right.$$

$$\left. + e^{\frac{-2b}{\mu}} (2\mu b^2+4\mu^2 b+4\mu^3) \right]. \qquad (D.24)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integrations over μ; then

$$BB4 = -\frac{8}{3} b^3+2b^2-\frac{8}{3} b+2-4b^2 E_3(2b)-8bE_4(2b)-8E_5(2b). \qquad (D.25)$$

Use the definition of BB5 in (D.6) and (D.13) to obtain

$$BB5 = -2\int_{-b}^{b} dx \int_{0}^{1} d\mu \left[2x^4+24\mu^2 x^2+48\mu^4-(b^4+4\mu b^3 \right.$$

$$\left. +12\mu^2 b^2+24\mu^3 b+24\mu^4) \left(e^{\frac{-(b+x)}{\mu}} + e^{\frac{-(b-x)}{\mu}} \right) \right]. \qquad (D.26)$$

Interchange the order of integration and perform the integration over x; then

$$BB5 = \int_0^1 d\mu\Big[-\frac{8}{5}b^5-\frac{96}{3}\mu^2b^3-192\mu^4b+4\mu b^4+16\mu^2b^3+48\mu^3b^2$$

$$+96\mu^4b+96\mu^5-e^{\frac{-2b}{\mu}}\left(4\mu b^4+16\mu^2b^3+48\mu^3b^2+96\mu^4b+96\mu^5\right)\Big]. \qquad (D.27)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integrations over μ to find

$$BB5 = -\frac{8}{5}b^5+2b^4-\frac{16}{3}b^3+12b^2-\frac{96}{5}b+16-4b^4E_3(2b)$$

$$-16b^3E_4(2b)-48b^2E_5(2b)-96bE_6(2b)-96E_7(2b). \qquad (D.28)$$

Use the definition of BB6 in (D.6) and (D.9) to obtain

$$BB6 = \int_{-b}^{b} dx \int_0^1 d\mu\Big[2-e^{\frac{-(b+x)}{\mu}}-e^{\frac{-(b-x)}{\mu}}\Big]. \qquad (D.29)$$

Interchange the order of integration and perform the integration over x; then

$$BB6 = \int_0^1 d\mu\left(4b-2\mu+2\mu e^{\frac{-2b}{\mu}}\right). \qquad (D.30)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integrations over μ to find

$$BB6 = 4b-1+2E_3(2b). \tag{D.31}$$

All the coefficients appearing in (D.3), the generalized principle, are now known. To minimize this principle see Appendix G. To obtain a quadratic trial function estimate instead of the quartic in (D.2), simply let a_2 in (D.3) go to zero. To minimize the resulting principle see Appendix G.

APPENDIX E

VARIATIONAL IMPLEMENTATION IN SPHERICAL GEOMETRY

The variational principle is

$$\frac{c}{2} = \frac{\int_{-b}^{b} dr r^2 \Phi^2(r)}{\int_{-b}^{b} dr r \Phi(r) \int_{-b}^{b} dr' r' \Phi(r') E_1(|r-r'|)} . \tag{E.1}$$

Trial functions of the form

$$\Phi(r) = 1 - a_1 r^2 - a_2 r^4, \tag{E.2}$$

where a_1 and a_2 are adjustable parameters, when used to rewrite (E.1), produce the generalized principle

$$\frac{c}{2} = \frac{TT1 a_1^2 + TT2 a_2^2 + TT3 a_1 a_2 + TT4 a_1 + TT5 a_2 + TT6}{BB1 a_1^2 + BB2 a_2^2 + BB3 a_1 a_2 + BB4 a_1 + BB5 a_2 + BB6} . \tag{E.3}$$

The substitution of (E.2) into the numerator of (E.1) and subsequent integration yield the following definitions;

$$TT1 = \frac{2}{7} b^7, \tag{E.4a}$$

$$TT2 = \frac{2}{11} b^{11}, \tag{E.4b}$$

$$TT3 = \frac{4}{9} b^9, \tag{E.4c}$$

$$TT4 = -\frac{4}{5} b^5, \tag{E.4d}$$

$$TT5 = -\frac{4}{7} b^7, \tag{E.4e}$$

$$TT6 = \frac{2}{3} b^3. \tag{E.4f}$$

Furthermore, after substitution of (E.2) into (E.1), the denominator of (E.1) can be rewritten as

$$\int_{-b}^{b} drr\Phi(r)\int_{-b}^{b} dr'r'\Phi(r')E_1(|r-r'|) = \int_{-b}^{b} drr\int_{-b}^{b} dr'r'E_1(|r-r'|)$$

$$-a_1\left[\int_{-b}^{b} drr^3\int_{-b}^{b} dr'r'E_1(|r-r'|)+\int_{-b}^{b} drr\int_{-b}^{b} dr'r'^3E_1(|r-r'|)\right]$$

$$-a_2\left[\int_{-b}^{b} drr\int_{-b}^{b} dr'r'^5E_1(|r-r'|)+\int_{-b}^{b} drr^5\int_{-b}^{b} dr'r'E_1(|r-r'|)\right]$$

$$+a_1a_2\left[\int_{-b}^{b} drr^3\int_{-b}^{b} dr'r'^5E_1(|r-r'|)+\int_{-b}^{b} drr^5\int_{-b}^{b} dr'r'^3E_1(|r-r'|)\right]$$

$$+a_1^2\left[\int_{-b}^{b} drr^3\int_{-b}^{b} dr'r'^3E_1(|r-r'|)\right]$$

$$+a_2^2\left[\int_{-b}^{b} drr^5\int_{-b}^{b} dr'r'^5E_1(|r-r'|)\right]. \tag{E.5}$$

Noting that $E_1(|r-r'|)$ is real and symmetric leads to the following definitions:

$$BB1 = \int_{-b}^{b} drr^3\int_{-b}^{b} dr'r'^3E_1(|r-r'|), \tag{E.6a}$$

$$BB2 = \int_{-b}^{b} dr r^5 \int_{-b}^{b} dr' r'^5 E_1(|r-r'|), \tag{E.6b}$$

$$BB3 = 2\int_{-b}^{b} dr r^3 \int_{-b}^{b} dr' r'^5 E_1(|r-r'|), \tag{E.6c}$$

$$BB4 = -2\int_{-b}^{b} dr r \int_{-b}^{b} dr' r'^3 E_1(|r-r'|), \tag{E.6d}$$

$$BB5 = -2\int_{-b}^{b} dr r \int_{-b}^{b} dr' r'^5 E_1(|r-r'|), \tag{E.6e}$$

$$BB6 = \int_{-b}^{b} dr r \int_{-b}^{b} dr' r' E_1(|r-r'|). \tag{E.6f}$$

In the integration of each of the definitions in (E.6) to closed form, some of the intermediate integrals will prove convenient. Start with

$$\int_{-b}^{b} dr' r' E_1(|r-r'|),$$

and use (A.2), the definition of $E_n(x)$, to obtain

$$\int_{-b}^{b} dr' r' E_1(|r-r'|) = \int_{-b}^{b} dr' r' \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|r-r'|}{\mu}}. \tag{E.7}$$

Interchanging the order of integration yields

$$\int_{-b}^{b} dr' r' E_1(|r-r'|) = \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dr' r' e^{\frac{-|r-r'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu}\left[\int_{-b}^{r} dr' r' e^{\frac{-(r-r')}{\mu}} + \int_r^b dr' r' e^{\frac{-(r'-r)}{\mu}}\right], \qquad \text{(E.8)}$$

therefore,

$$\int_{-b}^{b} dr' r' E_1(|r-r'|) = \int_0^1 d\mu\left[2r+(b+\mu)\left(e^{\frac{-(b+r)}{\mu}} - e^{\frac{-(b-r)}{\mu}}\right)\right]. \qquad \text{(E.9)}$$

Similarly,

$$\int_{-b}^{b} dr' r'^3 E_1(|r-r'|) = \int_{-b}^{b} dr' r'^3 \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|r-r'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dr' r'^3 e^{\frac{-|r-r'|}{\mu}}$$

$$\int_0^1 \frac{d\mu}{\mu}\left[\int_{-b}^{r} dr' r'^3 e^{\frac{-(r-r')}{\mu}} + \int_r^b dr' r'^3 e^{\frac{-(r'-r)}{\mu}}\right], \qquad \text{(E.10)}$$

therefore,

$$\int_{-b}^{b} dr' r'^3 E_1(|r-r'|) = \int_0^1 d\mu\left[2r^3+12\mu^2 r+(b^3+3\mu b^2\right.$$

$$\left.+6\mu^2 b+6\mu^3)\left(e^{\frac{-(b+r)}{\mu}} + e^{\frac{-(b-r)}{\mu}}\right)\right]. \qquad \text{(E.11)}$$

And again,

$$\int_{-b}^{b} dr' r'^5 E_1(|r-r'|) = \int_{-b}^{b} dr' r'^5 \int_0^1 \frac{d\mu}{\mu} e^{\frac{-|r-r'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \int_{-b}^{b} dr' r'^5 e^{\frac{-|r-r'|}{\mu}}$$

$$= \int_0^1 \frac{d\mu}{\mu} \left[\int_{-b}^{r} dr' r'^5 e^{\frac{-(r-r')}{\mu}} + \int_r^b dr' r'^5 e^{\frac{-(r'-r)}{\mu}} \right], \qquad \text{(E.12)}$$

therefore,

$$\int_{-b}^{b} dr' r'^5 E_1(|r-r'|) = \int_0^1 d\mu \left[2r^5 + 40\mu^2 r^3 + 240\mu^4 r \right.$$

$$+ \left(b^5 + 5\mu b^4 + 20\mu^2 b^3 + 60\mu^3 b^2 + 120\mu^4 b \right.$$

$$\left. \left. + 120\mu^5 \right) \left(e^{\frac{-(b+r)}{\mu}} + e^{\frac{-(b-r)}{\mu}} \right) \right]. \qquad \text{(E.13)}$$

Start with the definition of BB1 in (E.6) and use (E.11) to obtain

$$BB1 = \int_{-b}^{b} dr r^3 \int_0^1 d\mu \left[2r^3 + 12\mu^2 r + \left(b^3 + 3\mu b^2 \right.\right.$$

$$\left. \left. + 6\mu^2 b + 6\mu^3 \right) \left(e^{\frac{-(b+r)}{\mu}} + e^{\frac{-(b-r)}{\mu}} \right) \right]. \qquad \text{(E.14)}$$

Interchange the order of integration and perform the integration over r to find

$$BB1 = \int_0^1 d\mu \left[\frac{4}{7} b^7 + \frac{24}{5} \mu^2 b^5 + 72\mu^7 - 6\mu^3 b^4 - 2\mu b^6 \right.$$

$$- e^{\frac{-2b}{\mu}} \left(72\mu^7 + 144\mu^6 b + 144\mu^5 b^2 + 96\mu^4 b^3 + 42\mu^3 b^4 \right.$$

$$\left. \left. + 12\mu^2 b^5 + 2\mu b^6 \right) \right]. \qquad \text{(E.15)}$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB1 = \frac{4}{7} b^7 - b^6 + \frac{8}{5} b^5 - 1.5b^4 + 9 - 72E_9(2b) - 144bE_8(2b) - 144b^2E_7(2b)$$

$$-96b^3E_6(2b) - 42b^4E_5(2b) - 12b^5E_4(2b) - 2b^6E_3(2b). \qquad (E.16)$$

Use the definition of BB2 in (E.6) and (E.13) to obtain

$$BB2 = \int_{-b}^{b} dr r^5 \int_0^1 d\mu [2r^5 + 40\mu^2 r^3 + 240\mu^4 r + (b^5 + 5\mu b^4$$

$$+20\mu^2 b^3 + 60\mu^3 b^2 + 120\mu^4 b + 120\mu^5)(e^{\frac{-(b+r)}{\mu}} + e^{\frac{-(b-r)}{\mu}})]. \qquad (E.17)$$

Interchange the order of integration and perform the integration over r to find

$$BB2 = \int_0^1 d\mu [\frac{4}{11} b^{11} - 2\mu b^{10} + \frac{80}{9} \mu^2 b^9 - 30\mu^3 b^8 + \frac{480}{7} \mu^4 b^7 - 80\mu^5 b^6$$

$$+28,800\mu^{11} - e^{\frac{-2b}{\mu}}(28,800\mu^{11} + 57,600\mu^{10}b + 57,600\mu^9 b^2$$

$$+38,400\mu^8 b^3 + 19,200\mu^7 b^4 + 7680\mu^6 b^5 + 2480\mu^5 b^6$$

$$+640\mu^4 b^7 + 130\mu^3 b^8 + 20\mu^2 b^9 + 2\mu b^{10})]. \qquad (E.18)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB2 = \frac{4}{11} b^{11} - b^{10} + \frac{80}{27} b^9 - \frac{15}{2} b^8 + \frac{96}{7} b^7 - \frac{40}{3} b^6 + 2400$$

$$-28{,}800E_{13}(2b) - 57{,}600bE_{12}(2b) - 57{,}600b^2E_{11}(2b)$$

$$-38{,}400b^3E_{10}(2b) - 19{,}200b^4E_9(2b) - 7680b^5E_8(2b) - 2480b^6E_7(2b)$$

$$-640b^7E_6(2b) - 130b^8E_5(2b) - 20b^9E_4(2b) - 2b^{10}E_3(2b). \qquad (E.19)$$

Use the definition of BB3 in (E.6) and (E.13) to obtain

$$BB3 = 2\int_{-b}^{b} dr r^3 \int_0^1 d\mu [2r^5 + 40\mu^2 r^3 + 240\mu^4 r + (b^5 + 5\mu b^4$$

$$+20\mu^2 b^3 + 60\mu^3 b^2 + 120\mu^4 b + 120\mu^5)(e^{\frac{-(b+r)}{\mu}} + e^{\frac{-(b-r)}{\mu}})]. \qquad (E.20)$$

Interchange the order of integration and perform the integration over r to find

$$BB3 = \int_0^1 d\mu [\frac{8}{9} b^9 - 4\mu b^8 + \frac{104}{7} \mu^2 b^7 - 44\mu^3 b^6 + 96\mu^4 b^5 - 120\mu^5 b^4$$

$$+2880\mu^9 - e^{\frac{-2b}{\mu}}(2880\mu^9 + 5760\mu^8 b + 5760\mu^7 b^2$$

$$+3840\mu^6b^3+1800\mu^5b^4+624\mu^4b^5+164\mu^3b^6+32\mu^2b^7$$

$$+4\mu b^8)]. \qquad \text{(E.21)}$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB3 = \frac{8}{9}\,b^9-2b^8+\frac{104}{21}\,b^7-11b^6+\frac{96}{5}\,b^5-20b^4+288-2880E_{11}(2b)$$

$$-5760bE_{10}(2b)-5760b^2E_9(2b)-3840b^3E_8(2b)-1800b^4E_7(2b)$$

$$-624b^5E_6(2b)-164b^6E_5(2b)-32b^7E_4(2b)-4b^8E_3(2b). \qquad \text{(E.22)}$$

Use the definition of BB4 in (E.6) and (E.11) to obtain

$$BB4 = -2\int_{-b}^{b} dr\, r\int_0^1 d\mu\,[2r^3+12\mu^2r+(b^3+3\mu b^2+6\mu^2b$$

$$+6\mu^3)(e^{\frac{-(b+r)}{\mu}}+e^{\frac{-(b-r)}{\mu}})]. \qquad \text{(E.23)}$$

Interchange the order of integration and perform the integration over r to find

$$BB4 = \int_0^1 d\mu\,[-\frac{8}{5}\,b^5+4\mu b^4-8\mu^2b^3+12\mu^3b^2-24\mu^5$$

$$+e^{\frac{-2b}{\mu}}(24\mu^5+48\mu^4b+36\mu^3b^2+16\mu^2b^3+4\mu b^4)]. \qquad (E.24)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB4 = -\frac{8}{5}b^5+2b^4-\frac{8}{3}b^3+3b^2-4+24E_7(2b)+48bE_6(2b)$$

$$+36b^2E_5(2b)+16b^3E_4(2b)+4b^4E_3(2b). \qquad (E.25)$$

Use the definition of BB5 in (E.6) and (E.13) to obtain

$$BB5 = -2\int_{-b}^{b} dr\, r\int_0^1 d\mu[2r^5+40\mu^2r^3+240\mu^4r+(b^5+5\mu b^4$$

$$+20\mu^2b^3+60\mu^3b^2+120\mu^4b+120\mu^5)(e^{\frac{-(b+r)}{\mu}}+e^{\frac{-(b-r)}{\mu}})]. \qquad (E.26)$$

Interchange the order of integration and perform the integration over r to find

$$BB5 = \int_0^1 d\mu[-\frac{8}{7}b^7+4\mu b^6-16\mu^2b^5+60\mu^3b^4-160\mu^4b^3$$

$$+240\mu^5b^2-480\mu^7+e^{\frac{-2b}{\mu}}(480\mu^7+960\mu^6b+720\mu^5b^2$$

$$+320\mu^4b^3+100\mu^3b^4+24\mu^2b^5+4\mu b^6)]. \qquad (E.27)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB5 = -\frac{8}{7}b^7+2b^6-\frac{16}{3}b^5+15b^4-32b^3+40b^2-60+480E_9(2b)$$

$$+960bE_8(2b)+720b^2E_7(2b)+320b^3E_6(2b)$$

$$+100b^4E_5(2b)+24b^5E_4(2b)+4b^6E_3(2b). \qquad (E.28)$$

Use the definition of BB6 in (E.6) and (E.9) to obtain

$$BB6 = \int_{-b}^{b} dr\, r \int_0^1 d\mu \left[2r+(b+\mu)\left(e^{\frac{-(b+r)}{\mu}}-e^{\frac{-(b-r)}{\mu}}\right)\right]. \qquad (E.29)$$

Interchange the order of integration and perform the integration over r to find

$$BB6 = \int_0^1 d\mu \left[\frac{4}{3}b^3-2\mu b^2+2\mu^3-e^{\frac{-2b}{\mu}}\left(2\mu b^2+4\mu^2 b+2\mu^3\right)\right]. \qquad (E.30)$$

Now use (A.2), the definition of $E_n(x)$, to perform the integration over μ; then

$$BB6 = \frac{4}{3}b^3-b^2+\frac{1}{2}-2b^2E_3(2b)-4bE_4(2b)-2E_5(2b). \qquad (E.31)$$

All the coefficients appearing in (E.3), the generalized principle, are now known. To minimize this principle see Appendix G. To obtain a quadratic trial function estimate instead of the quartic in (E.2), simply let a_2 in (E.3) go to zero. To minimize the resulting principle see Appendix G.

APPENDIX F

VARIATIONAL IMPLEMENTATION IN CYLINDRICAL GEOMETRY

The variational principle is

$$c = \frac{\int_0^b drr\Phi^2(r)}{\int_0^b drr\Phi(r)\int_0^b dr'r'\Phi(r')K(r,r')}, \qquad (F.1)$$

$K(r,r')$ real and symmetric.

Trial functions of the form

$$\Phi(r) = 1-a_1r^2-a_2r^4, \qquad (F.2)$$

where a_1 and a_2 are adjustable parameters, when used to rewrite (F.1), produce the generalized principle

$$c = \frac{TT1a_1^2+TT2a_2^2+TT3a_1a_2+TT4a_1+TT5a_2+TT6}{BB1a_1^2+BB2a_2^2+BB3a_1a_2+BB4a_1+BB5a_2+BB6}. \qquad (F.3)$$

The substitution of (F.2) into the numerator of (F.1) and subsequent integration yield the following definitions;

$$TT1 = \frac{1}{6}b^6, \qquad (F.4a)$$

$$TT2 = \frac{1}{10}b^{10}, \qquad (F.4b)$$

$$TT3 = \frac{1}{4}b^8, \qquad (F.4c)$$

$$TT4 = -\frac{1}{2}b^4, \tag{F.4d}$$

$$TT5 = -\frac{1}{3}b^6, \tag{F.4e}$$

$$TT6 = \frac{1}{2}b^2. \tag{F.4f}$$

Furthermore, after substitution of (F.2) into (F.1), the denominator of (F.1) can be rewritten as

$$\int_0^b drr\Phi(r)\int_0^b dr'r'\Phi(r')K(r,r') = \int_0^b drr\int_0^b dr'r'K(r,r')$$

$$-a_1[\int_0^b drr\int_0^b dr'r'^3K(r,r')+\int_0^b drr^3\int_0^b dr'r'K(r,r')]$$

$$-a_2[\int_0^b drr\int_0^b dr'r'^5K(r,r')+\int_0^b drr^5\int_0^b dr'r'K(r,r')]$$

$$+a_1a_2[\int_0^b drr^3\int_0^b dr'r'^5K(r,r')+\int_0^b drr^5\int_0^b dr'r'^3K(r,r')]$$

$$+a_1^2[\int_0^b drr^3\int_0^b dr'r'^3K(r,r')]$$

$$+a_2^2[\int_0^b drr^5\int_0^b dr'r'^5K(r,r')]. \tag{F.5}$$

Noting that K(r,r') is real and symmetric leads to the following definitions;

$$BB1 = \int_0^b drr^3\int_0^b dr'r'^3K(r,r'), \tag{F.6a}$$

$$BB2 = \int_0^b drr^5 \int_0^b dr' r'^5 K(r,r'), \tag{F.6b}$$

$$BB3 = 2\int_0^b drr^3 \int_0^b dr' r'^5 K(r,r'), \tag{F.6c}$$

$$BB4 = -2\int_0^b drr \int_0^b dr' r'^3 K(r,r'), \tag{F.6d}$$

$$BB5 = -2\int_0^b drr \int_0^b dr' r'^5 K(r,r'), \tag{F.6e}$$

and

$$BB6 = \int_0^b drr \int_0^b dr' r' K(r,r'), \tag{F.6f}$$

where

$$K(r,r') = \int_1^\infty dy K_0(yr) I_0(yr') \quad \text{for } r'<r, \tag{F.7a}$$

and

$$K(r,r') = \int_1^\infty dy K_0(yr') I_0(yr) \quad \text{for } r'>r. \tag{F.7b}$$

In the integration of each of the definitions in (F.6) to a form usable for numerical evaluation, some of the intermediate integrals will prove convenient. Start with

$$\int_0^b dr' r' K(r,r') = \int_0^r dr' r' \int_1^\infty dy K_0(yr) I_0(yr')$$

$$+\int_r^b dr' r' \int_1^\infty dy K_0(yr') I_0(yr), \tag{F.8}$$

interchange the order of integration, and make the following transformation,

$$x' = r'y, \tag{F.9a}$$

$$x = ry. \tag{F.9b}$$

This yields

$$\int_0^b dr'r'K(r,r') = \int_1^\infty \frac{dy}{y^2}$$

$$\left[K_0(x)\int_0^x dx'x'I_0(x')+I_0(x)\int_x^{by} dx'x'K_0(x')\right], \tag{F.10}$$

which can be integrated, by using (B.1), to

$$\int_0^b dr'r'K(r,r') = \int_1^\infty \frac{dy}{y^2}$$

$$\left[xK_0(x)I_1(x)+xK_1(x)I_0(x)-byK_1(by)I_0(x)\right]. \tag{F.11}$$

Similarly,

$$\int_0^b dr'r'^3K(r,r') = \int_0^r dr'r'^3\int_1^\infty dyK_0(yr)I_0(yr')$$

$$+\int_r^b dr'r'^3\int_1^\infty dyK_0(yr')I_0(yr). \tag{F.12}$$

Interchange the order of integration and make the transformation in (F.9) to obtain

$$\int_0^b dr' r'^3 K(r,r') = \int_1^\infty \frac{dy}{y^4}$$

$$\left[K_0(x)\int_0^x dx' x'^3 I_0(x') + I_0(x)\int_x^{by} dx' x'^3 K_0(x')\right]. \qquad \text{(F.13)}$$

This can be integrated, by parts, through the use of (B.1) and (B.2), to

$$\int_0^b dr' r'^3 K(r,r') = \int_1^\infty \frac{dy}{y^4} \left[x^3 K_0(x) I_1(x) - 2x^2 K_0(x) I_2(x) + x^3 K_1(x) I_0(x)\right.$$

$$\left. + 2x^2 K_2(x) I_0(x) - I_0(x)\left((by)^3 K_1(by) + 2(by)^2 K_2(by)\right)\right]. \qquad \text{(F.14)}$$

Similarly,

$$\int_0^b dr' r'^5 K(r,r') = \int_0^r dr' r'^5 \int_1^\infty dy K_0(yr) I_0(yr')$$

$$+ \int_r^b dr' r'^5 \int_1^\infty dy K_0(yr') I_0(yr). \qquad \text{(F.15)}$$

Interchange the order of integration and make the transformation in (F.9) to obtain

$$\int_0^b dr' r'^5 K(r,r') = \int_1^\infty \frac{dy}{y^6}$$

$$\left[K_0(x)\int_0^x dx' x'^5 I_0(x') + I_0(x)\int_x^{by} dx' x'^5 K_0(x')\right]. \qquad \text{(F.16)}$$

This can be integrated, by parts, through the use of (B.1) and (B.2), to

$$\int_0^b dr' r'^5 K(r,r') = \int_1^\infty \frac{dy}{y^6}[x^5 K_0(x) I_1(x) - 4x^4 K_0(x) I_2(x) + 8x^3 K_0(x) I_3(x)$$

$$+ x^5 I_0(x) K_1(x) + 4x^4 I_0(x) K_2(x) + 8x^3 I_0(x) K_3(x)$$

$$- I_0(x)\left((by)^5 K_1(by) + 4(by)^4 K_2(by) + 8(by)^3 K_3(by)\right)]. \qquad (F.17)$$

Start with the definition of BB1 in (F.6), use (F.14) and (F.9), and interchange the order of integration to obtain

$$BB1 = \int_1^\infty \frac{dy}{y^8} \int_0^{by} dx[x^6 K_0(x) I_1(x) - 2x^5 K_0(x) I_2(x) + x^6 K_1(x) I_0(x)$$

$$+ 2x^5 K_2(x) I_0(x) - x^3 I_0(x)\left((by)^3 K_1(by) + 2(by)^2 K_2(by)\right)]. \qquad (F.18)$$

This can be integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of recursion relations (B.3), to

$$BB1 = \int_1^\infty \frac{dy}{y^8} \int_0^{by} dx[x^6 I_1(x) K_0(x) + x^6 K_0(x) I_3(x)]. \qquad (F.19)$$

Now make the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB1 = \int_0^b dr r^6 \int_r^\infty \frac{dx}{x} \left[I_1(x)K_0(x)+K_0(x)I_3(x) \right]. \qquad (F.20)$$

The above inner integrations can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, by use of the recursion relations (B.3), to

$$BB1 = \int_0^b dr r^6 [\frac{10}{9} - I_1(r)K_1(r)(\frac{10}{9} r + \frac{8}{9} \frac{1}{r}) - K_0(r)I_0(r)(\frac{10}{9} r + \frac{4}{3} \frac{1}{r})$$

$$+ I_1(r)K_0(r)(\frac{14}{9} + \frac{8}{3} \frac{1}{r^2}) + \frac{4}{9} K_1(r)I_0(r)], \qquad (F.21)$$

which becomes

$$BB1 = \frac{10}{63} b^7 - \int_0^b dr [I_1(r)K_1(r)(\frac{10}{9} r^7 + \frac{8}{9} r^5) + K_0(r)I_0(r)(\frac{10}{9} r^7$$

$$+ \frac{4}{3} r^5) - I_1(r)K_0(r)(\frac{14}{9} r^6 + \frac{8}{3} r^4) - \frac{4}{9} r^6 K_1(r)I_0(r)]. \qquad (F.22)$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

Start with the definition of BB2 in (F.6), use (F.17) and (F.9), and interchange the order of integration to obtain

$$BB2 = \int_1^\infty \frac{dy}{y^{12}} \int_0^{by} dx [x^{10}K_0(x)I_1(x) - 4x^9 K_0(x)I_2(x) + 8x^8 K_0(x)I_3(x)$$

$$+x^{10}I_0(x)K_1(x)+4x^9I_0(x)K_2(x)+8x^8I_0(x)K_3(x)$$

$$-x^5I_0(x)\left[(by)^5K_1(by)+4(by)^4K_2(by)+8(by)^3K_3(by)\right)]. \qquad (F.23)$$

This can be integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of recursion relations (B.3), to

$$BB2 = \int_1^{\infty} \frac{dy}{y^{12}} \int_0^{by} dx$$

$$[x^{10}K_0(x)I_3(x)+ \frac{2}{3} x^{10}K_0(x)I_1(x)+ \frac{1}{3} x^{10}K_0(x)I_5(x)]. \qquad (F.24)$$

Now make the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB2 = \int_0^b drr^{10}\int_r^{\infty} \frac{dx}{x} [\frac{2}{3} K_0(x)I_1(x)+K_0(x)I_3(x)+ \frac{1}{3} K_0(x)I_5(x)]. \qquad (F.25)$$

The above inner integrations can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, by the use of the recursion relations (B.3), to

$$BB2 = \int_0^b drr^{10}[\frac{178}{225} -K_1(r)I_1(r)(\frac{178}{225} r+ \frac{416}{225r} + \frac{128}{25r^3}) -K_0(r)I_0(r)$$

$$(\frac{178}{225} r+ \frac{184}{75r} + \frac{320}{25r^3}) +K_0(r)I_1(r)($$

$$\frac{314}{225} + \frac{608}{75r^2} + \frac{640}{25r^4})+$$

$$K_1(r)I_0(r)(\frac{136}{225} + \frac{64}{25r^2})], \qquad (F.26)$$

which becomes

$$BB2 = \frac{178}{2475}\, b^{11} - \int_0^b dr[K_1(r)I_1(r)(\frac{178}{225}\, r^{11} + \frac{416}{225}\, r^9 + \frac{128}{25}\, r^7)$$

$$+K_0(r)I_0(r)(\frac{178}{225}\, r^{11} + \frac{184}{75}\, r^9 + \frac{320}{25}\, r^7) - K_0(r)I_1(r)$$

$$(\frac{314}{225}\, r^{10} + \frac{608}{75}\, r^8 + \frac{640}{25}\, r^6) - K_1(r)I_0(r)$$

$$(\frac{136}{225}\, r^{10} + \frac{64}{25}\, r^8)]. \qquad (F.27)$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

Start with the definition of BB3 in (F.6), use (F.17) and (F.9), and interchange the order of integration to obtain

$$BB3 = 2\int_1^\infty \frac{dy}{y^{10}} \int_0^{by} dx[x^8K_0(x)I_1(x) - 4x^7K_0(x)I_2(x)$$

$$+8x^6K_0(x)I_3(x)+x^8I_0(x)K_1(x)+4x^7I_0(x)K_2(x)+8x^6I_0(x)K_3(x)$$

$$-x^3I_0(x)\left[(by)^5K_1(by)+4(by)^4K_2(by)+8(by)^3K_3(by)\right]\Big]. \qquad (F.28)$$

This can integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of recursion relations (B.3), to

$$BB3 = 2\int_1^\infty \frac{dy}{y^{10}} \int_0^{by} dx$$

$$\left[\frac{5}{6}\, x^8K_0(x)I_1(x)+x^8K_0(x)I_3(x)+ \frac{1}{6}\, x^8K_0(x)I_5(x)\right]. \qquad (F.29)$$

Now make the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB3 = \int_0^b drr^8\int_r^\infty \frac{dx}{x} \left[\frac{5}{3}\, K_0(x)I_1(x)+2K_0(x)I_3(x)+ \frac{1}{3}\, K_0(x)I_5(x)\right]. \qquad (F.30)$$

The above inner integrations can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, by the use of the recursion relations (B.3), to

$$BB3 = \int_0^b drr^8\left[\frac{428}{225} -I_1(r)K_1(r)\left(\frac{428}{225}\, r+ \frac{616}{225r} + \frac{384}{75r^3}\right)\right.$$

$$-I_0(r)K_0(r)\left(\frac{428}{225}\, r+ \frac{284}{75r} + \frac{320}{25r^3}\right)+ K_0(r)I_1(r)$$

$$\left(\frac{664}{225}+\frac{2424}{225r^2}+\frac{640}{25r^4}\right)+I_0(r)K_1(r)\left(\frac{236}{225}+\frac{64}{25r^2}\right)\Big], \qquad \text{(F.31)}$$

which becomes

$$BB3 = \frac{428}{2025}\,b^9-\int_0^b dr\Big[I_1(r)K_1(r)\left(\frac{428}{225}\,r^9+\frac{616}{225}\,r^7+\frac{384}{75}\,r^5\right)$$

$$+I_0(r)K_0(r)\left(\frac{428}{225}\,r^9+\frac{284}{75}\,r^7+\frac{320}{25}\,r^5\right)-K_0(r)I_1(r)$$

$$\left(\frac{664}{225}\,r^8+\frac{2424}{225}\,r^6+\frac{640}{25}\,r^4\right)-I_0(r)K_1(r)\left(\frac{236}{225}\,r^8+\frac{64}{25}\,r^6\right)\Big]. \qquad \text{(F.32)}$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

Start with the definition of BB4 in (F.6), use (F.14) and (F.9), and interchange the order of integration to obtain

$$BB4 = -2\int_1^\infty \frac{dy}{y^6}\int_0^{by} dx\Big[x^4K_0(x)I_1(x)-2x^3K_0(x)I_2(x)+x^4K_1(x)I_0(x)$$

$$+2x^3K_2(x)I_0(x)-xI_0(x)\left((by)^3K_1(by)+2(by)^2K_2(by)\right)\Big]. \qquad \text{(F.33)}$$

This can be integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of the recursion relations (B.3), to

$$BB4 = -\int_1^\infty \frac{dy}{y^6}\int_0^{by} dx\left(3x^4K_0(x)I_1(x)+x^4K_0(x)I_3(x)\right). \qquad \text{(F.34)}$$

Now take the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB4 = -\int_0^b dr r^4 \int_r^\infty \frac{dx}{x} \left(3K_0(x)I_1(x)+K_0(x)I_3(x)\right). \tag{F.35}$$

The above inner integrations can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, by the use of the recursion relations (B.3), to

$$BB4 = \int_0^b dr r^4 \left[-\frac{28}{9} + I_1(r)K_1(r)\left(\frac{28}{9} r + \frac{8}{9} r^{-1}\right) + K_0(r)I_0(r) \right.$$

$$\left. \left(\frac{28}{9} r + \frac{4}{3} r^{-1}\right) - I_1(r)K_0(r)\left(\frac{32}{9} + \frac{8}{3r^2}\right) - \frac{4}{9} K_1(r)I_0(r) \right], \tag{F.36}$$

which becomes

$$BB4 = -\frac{28}{45} b^5 + \int_0^b dr \left[I_1(r)K_1(r)\left(\frac{28}{9} r^5 + \frac{8}{9} r^3\right) + K_0(r)I_0(r)\left(\frac{28}{9} r^5 \right.\right.$$

$$\left.\left. + \frac{4}{3} r^3\right) - I_1(r)K_0(r)\left(\frac{32}{9} r^4 + \frac{8}{3} r^2\right) - \frac{4}{9} r^4 K_1(r)I_0(r) \right]. \tag{F.37}$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

Start with the definition of BB5 in (F.6), use (F.17) and (F.9), and interchange the order of integration to get

$$BB5 = -2\int_1^{\infty} \frac{dy}{y^8} \int_0^{by} dx\left[x^6K_0(x)I_1(x)-4x^5K_0(x)I_2(x)+8x^4K_0(x)I_3(x)\right.$$

$$+x^6I_0(x)K_1(x)+4x^5I_0(x)K_2(x)+8x^4I_0(x)K_3(x)-xI_0(x)$$

$$\left.\left((by)^5K_1(by)+4(by)^4K_2(by)+8(by)^3K_3(by)\right)\right]. \qquad (F.38)$$

This can be integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of the recursion relations (B.3), to

$$BB5 = -2\int_1^{\infty} \frac{dy}{y^8} \int_0^{by} dx$$

$$\left[\frac{4}{3}\, x^6I_1(x)K_0(x)+ \frac{1}{2}\, x^6K_0(x)I_3(x)+ \frac{1}{6}\, x^6K_0(x)I_5(x)\right]. \qquad (F.39)$$

Now take the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB5 = -2\int_0^{b} drr^6\int_r^{\infty} \frac{dx}{x}$$

$$\left[\frac{4}{3}\, I_1(x)K_0(x)+ \frac{1}{2}\, K_0(x)I_3(x)+ \frac{1}{6}\, K_0(x)I_5(x)\right]. \qquad (F.40)$$

The above inner integrations can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, by the use of the recursion relations (B.3), to

$$BB5 = -2\int_0^b drr^6\left[\frac{314}{225} - I_1(r)K_1(r)\left(\frac{314}{225}\,r + \frac{208}{225r} + \frac{384}{150r^3}\right)\right.$$

$$-I_0(r)K_0(r)\left(\frac{314}{225}\,r + \frac{92}{75r} + \frac{32}{5r^3}\right) + K_0(r)I_1(r)\left(\frac{382}{225} + \frac{304}{75r^2}\right.$$

$$\left.\left. + \frac{64}{5r^4}\right) + I_0(r)K_1(r)\left(\frac{68}{225} + \frac{32}{25r^2}\right)\right], \qquad \text{(F.41)}$$

which becomes

$$BB5 = -\frac{628}{1575}\,b^7 + 2\int_0^b dr\left[I_1(r)K_1(r)\left(\frac{314}{225}\,r^7 + \frac{208}{225}\,r^5 + \frac{384}{150}\,r^3\right)\right.$$

$$+I_0(r)K_0(r)\left(\frac{314}{225}\,r^7 + \frac{92}{75}\,r^5 + \frac{32}{5}\,r^3\right) - K_0(r)I_1(r)\left(\frac{382}{225}\,r^6\right.$$

$$\left.\left. + \frac{304}{75}\,r^4 + \frac{64}{5}\,r^2\right) - I_0(r)K_1(r)\left(\frac{68}{225}\,r^6 + \frac{32}{25}\,r^4\right)\right]. \qquad \text{(F.42)}$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

Start with the definition of BB6 in (F.6), use (F.11) and (F.9), and interchange the order of integration to get

$$BB6 = \int_1^{\infty} \frac{dy}{y^4} \int_0^{by} dx$$

$$[x^2K_0(x)I_1(x)+x^2K_1(x)I_0(x)-xI_0(x)byK_1(by)]. \qquad (F.43)$$

This can be integrated, by parts, through the use of (B.1) and (B.2), and reduced, through the use of the recursion relations (B.3), to

$$BB6 = 2\int_1^{\infty} \frac{dy}{y^4} \int_0^{by} dx x^2K_0(x)I_1(x). \qquad (F.44)$$

Now take the inverse transformation of (F.9), interchange the order of integration, and use (F.9) to obtain

$$BB6 = 2\int_0^b dr r^2 \int_r^{\infty} \frac{dx}{x} K_0(x)I_1(x). \qquad (F.45)$$

The above inner integration can be done through the use of (B.4) and the asymptotic series (B.6). The resulting equation is reduced, through the use of the recursion relations (B.3), to

$$BB6 = 2\int_0^b dr[r^2-r^3I_1(r)K_1(r)-r^3K_0(r)I_0(r)+r^2K_0(r)I_1(r)], \qquad (F.46)$$

which becomes

$$BB6 = \frac{2}{3} b^3-2\int_0^b dr[r^3I_1(r)K_1(r)+r^3K_0(r)I_0(r)-r^2K_0(r)I_1(r)]. \qquad (F.47)$$

This equation has a well-behaved integrand, since $K_0(r)$ has a logarithmic singularity as r approaches zero, and $K_n(r)$ goes as r^{-n} as r approaches zero, and is, therefore, easy to evaluate numerically.

All the coefficients appearing in (F.3), the generalized principle, are now known. To minimize the principle see Appendix G. To obtain a quadratic trial function estimate instead of the quartic in (F.2), simply let a_2 in (F.3) go to zero. To minimize the resulting principle see Appendix G.

APPENDIX G

MINIMIZING THE GENERALIZED VARIATIONAL PRINCIPLE

A. The Quadratic Trial Function

For the quadratic trial function case, the trial flux has the form of (G.1), and the variational principle takes the form of (G.2);

$$\Phi(r) = 1-a_1 r^2, \tag{G.1}$$

$$\gamma = \frac{TT1a_1^2+TT4a_1+TT6}{BB1a_1^2+BB4a_1+BB6}. \tag{G.2}$$

The relation, (G.2), must be minimized with respect to the parameter a_1 in order to determine a_1 and, thereby, estimate the eigenvalue γ. The following conditions are necessary for the minimization of (G.2)[43];

$$\frac{\partial}{\partial a_1}(\gamma) = 0, \tag{G.3a}$$

$$\frac{\partial^2}{\partial a_1^2}(\gamma) > 0. \tag{G.3b}$$

The first of (G.3) leads to

$$cc1a_1^2+cc2a_1+cc3 = 0, \tag{G.4}$$

where

$$cc1 = TT1BB4-TT4BB1, \tag{G.5a}$$

$$cc2 = 2TT1BB6-2TT6BB1, \tag{G.5b}$$

and

$$cc3 = TT4BB6-TT6BB4. \tag{G.5c}$$

The solution to (G.4) is

$$a_1 = \frac{-cc2 \pm (cc2^2 - 4cc1cc3)^{.5}}{2cc1} . \tag{G.6}$$

To choose the correct a_1, physical conditions are used. These conditions must be satisfied and have always resulted in the elimination of one of the choices for a_1. The first condition is that the flux must have its maximum at the geometrical center. This requires

$$\frac{\partial}{\partial r} \Phi(r)|_{r=0} = 0, \tag{G.7a}$$

and

$$\frac{\partial^2}{\partial r^2} \Phi(r)|_{r=0} < 0. \tag{G.7b}$$

The first of (G.7) is trivially satisfied, and the second of (G.7) leads to

$$a_1 > 0. \tag{G.8}$$

The second physical condition is that the flux be everywhere nonnegative. From (G.1) and (G.8) this leads to

$$1-a_1 b^2 \geq 0 \tag{G.9a}$$

or

$$a_1 \leq \frac{1}{b^2} . \tag{G.9b}$$

With (G.6), (G.8), and (G.9), a physical a_1 has been selected which extremizes the principle (G.2). Now this a_1 must satisfy the second of (G.3) to qualify as a flux estimate. The second of (G.3) leads to

$$\frac{2TT1}{bott} - \frac{2(2TT1a_1+TT4)(2BB1a_1+BB4)}{bott^2} - \frac{(TT1a_1^2+TT4a_1+TT6)(2BB1)}{bott^2}$$

$$+ \frac{2(TT1a_1^2+TT4a_1+TT6)(2BB1a_1+BB4)^2}{bott^3} > 0 \tag{G.10}$$

where $bott = BB1a_1^2+BB4a_1+BB6$.

The value of a_1 which satisfies (G.6), (G.8), (G.9) and (G.10) is the value which, in conjunction with (G.1), gives a flux estimate for the system. In conjunction with (G.2), a_1 gives an eigenvalue estimate for the just critical system of size b.

B. The Quartic Trial Function

For the quartic trial function case the trial flux has the form of (G.11), and the variational principle takes the form of (G.12);

$$\Phi(r) = 1-a_1r^2-a_2r^4, \tag{G.11}$$

$$\gamma = \frac{TT1a_1^2+TT2a_2^2+TT3a_1a_2+TT4a_1+TT5a_2+TT6}{BB1a_1^2+BB2a_2^2+BB3a_1a_2+BB4a_1+BB5a_2+BB6}\,. \tag{G.12}$$

The principle, (G.12), must be minimized with respect to the parameters a_1 and a_2 in order to determine a_1 and a_2 and, thereby, estimate the eigenvalue γ. The following conditions are necessary for the minimization of (G.12)[44]:

$$\frac{\partial}{\partial a_1}(\gamma) = 0, \tag{G.13a}$$

$$\frac{\partial}{\partial a_2}(\gamma) = 0; \tag{G.13b}$$

$$\left(\frac{\partial^2\gamma}{\partial a_1\partial a_2}\right)^2 - \left(\frac{\partial^2\gamma}{\partial a_1^2}\right)\left(\frac{\partial^2\gamma}{\partial a_2^2}\right) < 0; \tag{G.14}$$

$$\frac{\partial^2\gamma}{\partial a_2^2} + \frac{\partial^2\gamma}{\partial a_1^2} > 0. \tag{G.15}$$

The equations (G.13) lead to

$$a_1^2(c_1+c_2a_2)+a_1(c_3+c_4a_2+c_5a_2^2)+(c_6a_2+c_7a_2^2+c_8a_2^3+c_9) = 0, \qquad (G.16a)$$

and

$$a_2^2(c_{17}+c_{15}a_1)+a_2(c_{12}a_1^2+c_{14}a_1+c_{16})$$
$$+(c_{10}a_1^3+c_{11}a_1^2+c_{13}a_1+c_{18}) = 0, \qquad (G.16b)$$

where

$$c_1 = TT1BB4-TT4BB1; \qquad (G.17a)$$

$$c_2 = TT1BB3-TT3BB1; \qquad (G.17b)$$

$$c_3 = 2TT1BB6-2TT6BB1; \qquad (G.17c)$$

$$c_4 = 2TT1BB5-2TT5BB1; \qquad (G.17d)$$

$$c_5 = 2TT1BB2-2TT2BB1; \qquad (G.17e)$$

$$c_6 = TT3BB6+TT4BB5-TT6BB3-TT5BB4; \qquad (G.17f)$$

$$c_7 = TT3BB5+TT4BB2-TT5BB3-TT2BB4; \qquad (G.17g)$$

$$c_8 = TT3BB2-TT2BB3; \qquad (G.17h)$$

$$c_9 = TT4BB6-TT6BB4; \qquad (G.17i)$$

$$c_{10} = TT3BB1-TT1BB3; \qquad (G.17j)$$

$$c_{11} = TT3BB4+TT5BB1-TT4BB3-TT1BB5; \qquad (G.17k)$$

$$c_{12} = 2TT2BB1-2TT1BB2; \qquad (G.17l)$$

$$c_{13} = TT3BB6+TT5BB4-TT6BB3-TT4BB5; \qquad (G.17m)$$

$$c_{14} = 2TT2BB4-2TT4BB2; \qquad (G.17n)$$

$$c_{15} = TT2BB3-TT3BB2; \qquad (G.17o)$$

$$c_{16} = 2TT2BB6-2TT6BB2; \qquad (G.17p)$$

$$c_{17} = TT2BB5-TT5BB2; \tag{G.17q}$$

$$c_{18} = TT5BB6-TT6BB5. \tag{G.17r}$$

The physical conditions which apply to the solution of (G.16) are, for a_1, nonnegative flux and maximum flux at r=0. These conditions lead to, from (G.11),

$$a_1 < \frac{1}{b^2} - a_2 b^2, \tag{G.18a}$$

and

$$a_1 > 0. \tag{G.18b}$$

The physical conditions which apply to the solution of (G.16) are, for a_2, nonnegative flux and first derivative of flux always less than zero. The first condition is obvious; the second stems from the fact that, in simple systems, the leakage depletes the neutron population near the surface more than the population near the center. An analogy is to compare the neutron flux with the temperature profile which occurs in a heat source region surrounded by an infinite heat sink region and in which the sources are linearly proportional to the temperature. These conditions lead to, from (G.11),

$$a_2 < \frac{1}{b^4} - \frac{a_1}{b^2}, \tag{G.19a}$$

and

$$a_2 > -\frac{a_1}{2b^2} \,. \qquad \text{(G.19b)}$$

The solution algorithm for (G.16) is to set a_2 to zero and calculate the coefficients for the equation quadratic in a_1. Solve the equation quadratic in a_1 subject to (G.18). Now use this value of a_1 to calculate the coefficients for the equation quadratic in a_2. Solve the equation quadratic in a_2 subject to (G.19). With this value of a_2 return to the equation quadratic in a_1. This iteration is carried out until a_1 and a_2 are converged to some criterion. This then gives a set of a_1 and a_2 which extremizes (G.12) and abides by the physical conditions (G.18) and (G.19). It must now be shown that this pair of values, a_1 and a_2, minimize (G.12). Let

$$VPT = TT1a_1^2+TT2a_2^2+TT3a_1a_2+TT4a_1+TT5a_2+TT6, \qquad \text{(G.20)}$$

and

$$VPB = BB1a_1^2+BB2a_2^2+BB3a_1a_2+BB4a_1+BB5a_2+BB6. \qquad \text{(G.21)}$$

Then,

$$\frac{\partial^2\gamma}{\partial a_1^2} = \frac{(2TT1)}{VPB} - \frac{2(2TT1a_1+TT3a_2+TT4)(2BB1a_1+BB3a_2+BB4)}{VPB^2}$$

$$- \frac{(2BB1)VPT}{VPB^2} + \frac{2(2BB1a_1+BB3a_2+BB4)^2VPT}{VPB^3}; \qquad \text{(G.22)}$$

$$\frac{\partial^2\gamma}{\partial a_1 \partial a_2} = \frac{TT3}{VPB} - \frac{(2TT1a_1+TT3a_2+TT4)(2BB2a_2+BB3a_1+BB5)}{VPB^2}$$

$$- \frac{BB3VPT}{VPB^2} - \frac{(2BB1a_1+BB3a_2+BB4)(2TT2a_2+TT3a_1+TT5)}{VPB^2}$$

$$+ \frac{2(2BB1a_1+BB3a_2+BB4)(2BB2a_2+BB3a_1+BB5)}{VPB^3} VPT; \qquad (G.23)$$

and

$$\frac{\partial^2\gamma}{\partial a_2^2} = \frac{(2TT2)}{VPB} - \frac{2(2TT2a_2+TT3a_1+TT5)(2BB2a_2+BB3a_1+BB5)}{VPB^2}$$

$$- \frac{(2BB2)VPT}{VPB^2} + \frac{2(2BB2a_2+BB3a_1+BB5)^2VPT}{VPB^3}. \qquad (G.24)$$

The pair, a_1 and a_2, determined above are substituted into (G.22), (G.23), and (G.24), which are in turn substituted into (G.14) and (G.15). The values of a_1 and a_2 determined above satisfy (G.14) and (G.15). The values of a_1 and a_2 which minimize (G.12) subject to the physical conditions (G.18) and (G.19) will yield an eigenvalue estimate for the system of size b when used with (G.12), and a flux estimate for the system when used with (G.11).

APPENDIX H

EXTRAPOLATION DISTANCES

In the diffusion theory scalar fluxes appearing in Chapter Three, the boundary conditions applied are the so-called extrapolation distance boundary conditions. These conditions require that the scalar flux vanish at a specified distance, the extrapolation distance, beyond the physical boundary. The extrapolation distance boundary conditions are a mathematical approximation to the physical boundary conditions of no incoming neutrons[38]. Ordinarily, due to diffusion theory being restricted to c approximately equal to one, only one extrapolation distance is required. However, in this thesis, a wide range of the neutron multiplicities are used, and, therefore, more extrapolation distances are required. The extrapolation distances usually used come from the Milne problem[1,7] and are strictly correct only for semi-infinite slabs. Davison[3] suggests that it is not unreasonable to use the Milne problem extrapolation distances for all geometries and sizes. From Table 2.5 of Bell and Glasstone[1] it can be seen that the product cx_0, where x_0 is the extrapolation distance, is smooth and slowly varying. Linear interpolation from this table, rather than from a table of extrapolation distances, is easy and yields the values in Table H-1.

TABLE H-1

Extrapolation Distances for Diffusion Theory Scalar Fluxes

c	extrapolation distance in units of the mean free path
1.02	.6965
1.05	.6767
1.10	.6460
1.20	.5924
1.40	.5084
1.60	.4455

APPENDIX I

CODE LISTINGS AND COMMENTS

In essence, this appendix is just a repository for the codes used in this thesis. An index to these codes appears as column four in Table I-1. Code names were chosen according to a pattern which follows. The leading t in the first twenty-one code names merely denotes that the code is a thesis version. The next two alphanumeric characters denote the type of scalar flux used in the code (dt=diffusion theory, v2=V_2, v4=V_4). The fourth and fifth characters denote the geometry (sb=slab, sp=spherical, cl=cylindrical), and the sixth character denotes the type of critical size search carried on (1=variational, 2=direct leakage operator, 3=indirect leakage operator). tv4eig, the twenty-second code listed in Table I-1, is a modification of tv4sb1 which does not iterate on the size but prints an eigenvalue estimate for a specified input size. tv4lop is a modification of tv4sp3 which not only calculates the total leakage probability distribution but also calculates each term of the Neumann series which comprises the total leakage probability. tv4lop does not iterate on size and, therefore, must be supplied with the parameters which yield a critical system. xplot takes the output of tv4lop and plots it using the "disspla" plotting package.

The coding is in standard Fortran suitable for use on the CDC-7600 computer, with the exception of tv2cl3, tv4cl3, tv4lop, and xplot. The excepted codes were written for the Cray-1 computer because of the array sizes or do-loop counters used. There are several special callable functions available at the LANL computer facilities which were used in

the following codes. Included in these functions are the first order exponential integral function (available as el(x)), and the zeroth and first order modified Bessel functions of the first and third kind (i.e., $K_0(x)$ is available as besk0(x)). The modified Bessel functions are also available exponentially scaled (i.e., $e^x K_0(x)$ is available as besk0e(x)). A system quadrature routine called "quad", as in (H.1), is used frequently in this thesis' coding. This routine,

```
LHS = quad(func,a,b,re,mxeval,kount),                    (H.1)
```

will integrate the function func, which is included as an external function statement, from a to b subject to relative error re. This routine has been extensively tested and is accurate so long as the function being integrated is bounded.

Table I-1 contains a list of the codes and their running times per cycle, where a cycle is defined as one K_{eff} calculation. Times were included for both the smallest and the largest critical radii used. The difference in running time between different radii for the V_4 scalar flux codes is due primarily to the time required in iteration to get the a_1 and a_2 coefficients. The method three codes, indicated by the last digit of code names, will also require longer times to converge to the spatial leakage probability distribution for the larger systems. This is due primarily to the fact that there is more scattering transport before leakage in the larger systems. Times listed for all but two of the codes are total times including compiler, CPU, in/out, and memory, and are for the CDC-7600 computer. The exceptions to this are the times

for the tv2c13 and tv4c13 codes which are Cray-1 total times. The Cray-1 computer runs at approximately twice the speed of the CDC-7600 computer.

TABLE I-1

Code Execution Times and Index

code	execution time smallest radii in seconds	execution time largest radii in seconds	pages listed
tv2sb1	.018	.018	232
tv4sb1	.038	2.390	234
tv2sb2	.018	.018	237
tv4sb2	.038	2.391	239
tdtsb2	.022	.026	242
tv2sb3	44.129	101.963	243
tv4sb3	44.447	159.191	246
tv2sp1	.022	.024	250
tv4sp1	.481	94.915	252
tv2sp2	.022	.024	255
tv4sp2	.468	92.868	257
tdtsp2	.029	.029	260
tv2sp3	95.931	211.087	261
tv4sp3	96.355	307.597	264
tv2c11	.039	.071	268
tv4c11	.191	22.197	270

Table I-1

continued

code	execution time smallest radii in seconds	execution time largest radii in seconds	pages listed
tv2c12	.069	.077	273
tv4c12	.204	22.582	275
tdtc12	.059	.077	279
tv2c13	869.606	1430.628	280
tv4c13	1184.173	1490.040	283
tv4eig	---	---	288
tv4lop	---	---	291
xplot	---	---	294

```
      program tv2sb1(kout,tty,tape5=kout,tape6=tty)
      real keff
      common rn,b,re,mxeval,vp
      call xsetf(0)
      bbb=.0001
      b=.5120
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expl
      common rn,b,re,mxeval,vp
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      tt1=.4*b5
      tt4=-4./3.*b3
      tt6=2.*b
      rn=3.
      e32b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+16.*b2*e52b+16.*b*e62b+8.*e72b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb6=4.*b-1.+2.*e32b
      ac1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      ac3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6))
```

```
 bott=bb1*a1*a1+bb4*a1+bb5
 x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
1+tt6)/bott**3.
 if(x2ndder.lt.0.0) call exit(2)
 if(x2ndder.eq.0.0) call exit(2)
 return
 end


 function expi(u)
 common rn,b,re,mxeval,vp
 expi=u**(rn-2.)*exp(-2.*b/u)
 return
 end
```

```
      program tv4sb1(kout,tty,tape5=kout,tape6=tty)
      real keff
      common rn,b,re,mxeval,vp
      call xsetf(0)
      bbb=.0001
      b=.5120
      c=1.8
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common rn,b,re,mxeval,vp
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      tt1=.4*b5
      tt2=2./9.*b9
      tt3=4./7.*b7
      tt4=-4./3.*b3
      tt5=-.8*b5
      tt6=2.*b
      rn=3.
      e32b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+16.*b2*e52b+16.*b*e62b+8.*e72b
      bb2=4./9.*b9-b8+16./7.*b7-4.*b6+96./25.*b5-115.2
     1+2.*b8*e32b+16.*b7*e42b+80.*b6*e52b+288.*b5*e62b+768.*b4*e72b
```

```
     1+1536.*b3*e82b+2304.*b2*e92b+2304.*b*e102b+1152.*e112b
      bb3=8./7.*b7-2.*b6+56./15.*b5-6.*b4+32./5.*b3-24.+4.*b6*e32b
     1+24.*b5*e42b+88.*b4*e52b+224.*b3*e62b+384.*b2*e72b+384.
     1*b*e82b+192.*e92b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb5=-8./5.*b5+2.*b4-16./3.*b3+12.*b2-96./5.*b+16.
     1-4.*b4*e32b-16.*b3*e42b-48.*b2*e52b-96.*b*e62b-96.*e72b
      bb6=4.*b-1.+2.*e32b
      c1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c18+c14*a1+c12*a1*a1
      cc6=c16+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      fixck1=1./b2-a2*b2
      if(a11.gt.fixck1) a1=a12
      if(a12.gt.fixck1) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      fixck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
      if(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.fixck2) a2=a22
      if(a22.gt.fixck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
      if(a1.gt.fixck1) call exit(2)
      if(a1.lt.0.0) call exit(2)
      if(a2.gt.fixck2) call exit(2)
      if(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
```

```
 x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
1*vpt/vpb3
 x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
 x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
 detr=x2npa12*x2npa12-x2npa11*x2npa22
 If((x2npa22+x2npa11).lt.0.0) call exit(2)
 If((x2npa22+x2npa11).eq.0.0) call exit(2)
 If(detr.gt.0.0) call exit(2)
 If(detr.eq.0.0) call exit(2)
 return
 end


 function expi(u)
 common rn,b,re,mxeval,vp
 expi=u**(rn-2.)*exp(-2.*b/u)
 return
 end
```

```
      program tv2sb2(kout,tty,tape5=kout,tape6=tty)
      real keff
      common b2,b3,e32b,e42b,e52b,rn,b,re,mxeval,a1
      call xsetf(0)
      bbb=.0001
      b=.5120
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=2.*(b-a1/3.*b3)
      xprod=c*coll
      xleak=.5*c*(1.-a1+4./3.*a1*b-a1*b2
     1+(2.*a1*b2-2.)*e32b+4.*a1*b*e42b+4.*a1*e52b)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expl
      common b2,b3,e32b,e42b,e52b,rn,b,re,mxeval,a1
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      tt1=.4*b5
      tt4=-4./3.*b3
      tt6=2.*b
      rn=3.
      e32b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+18.*b2*e52b+15.*b*e62b+8.*e72b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb6=4.*b-1.+2.*e32b
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
```

```
 if(a12.lt.0.0) a1=a11
 if(a1.gt.flxck) call exit(2)
 if(a1.lt.0.0) call exit(2)
 vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6))
 bott=bb1*a1*a1+bb4*a1+bb6
 x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
1+tt6)/bott**3.
 if(x2ndder.lt.0.0) call exit(2)
 if(x2ndder.eq.0.0) call exit(2)
 return
 end


 function expi(u)
 common b2,b3,e32b,e42b,e52b,rn,b,re,mxeval,a1
 expi=u**(rn-2.)*exp(-2.*b/u)
 return
 end
```

```
      progrom tv4sb2(kout.tty.tape5=kout.tape6=tty)
      real keff
      common /one/ rn,b,re.mxeval,a1,a2
      common /two/ b2.b3.b4,b5,e32b,e42b.e52b,e62b.e72b
      call xsetf(0)
      bbb=.0001
      b=.5120
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=2.*(b-a1/3.*b3-a2/5.*b5)
      xprod=c*coll
      xleak=.5*c*(1.-a1+4./3.*a1*b-a1*b2
     1+(2.*a1*b2-2.)*e32b+4.*a1*b*e42b+4.*a1*e52b)
      xleak=xleak-.5*c*a2*(b4-8./3.*b3+6.*b2
     1-48./5.*b+8.-2.*b4*e32b-8.*b3*e42b-24.*b2*e52b
     1-48.*b*e62b-48.*e72b)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common /one/ rn,b,re.mxeval,a1,a2
      common /two/ b2.b3,b4,b5,e32b,e42b.e52b,e62b,e72b
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b8*b
      b8=b7*b
      b9=b8*b
      tt1=.4*b5
      tt2=2./9.*b9
      tt3=4./7.*b7
      tt4=-4./3.*b3
      tt5=-.8*b5
      tt6=2.*b
      rn=3.
      e32b=quad(expi..0000000000001.1.,re.mxeval.kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval.kount)
      rn=5.
      e52b=quad(expl,.0000000000001.1..re.mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval.kount)
      rn=7.
      e72b=quad(expl..0000000000001.1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi..0000000000001.1..re.mxeval,kount)
      rn=9.
```

```
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+16.*b2*e52b+16.*b*e62b+8.*e72b
      bb2=4./9.*b9-b8+16./7.*b7-4.*b6+96./25.*b5-115.2
     1+2.*b8*e32b+18.*b7*e42b+80.*b6*e52b+288.*b5*e62b+768.*b4*e72b
     1+1538.*b3*e82b+2304.*b2*e92b+2304.*b*e102b+1152.*e112b
      bb3=8./7.*b7-2.*b6+56./15.*b5-6.*b4+32./5.*b3-24.+4.*b6*e32b
     1+24.*b5*e42b+88.*b4*e52b+224.*b3*e62b+384.*b2*e72b+384.
     1*b*e82b+192.*e92b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb5=-8./5.*b5+2.*b4-16./3.*b3+12.*b2-96./5.*b+16.
     1-4.*b4*e32b-16.*b3*e42b-48.*b2*e52b-96.*b*e62b-96.*e72b
      bb6=4.*b-1.+2.*e32b
      c1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxak1=1./b2-a2*b2
      if(a11.gt.flxck1) a1=a12
      if(a12.gt.flxck1) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
      if(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.flxck2) a2=a22
      if(a22.gt.flxck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
```

```
      if(a1.gt.flxck1) call exit(2)
      if(a1.lt.0.0) call exit(2)
      if(a2.gt.flxck2) call exit(2)
      if(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      if((x2npa22+x2npa11).lt.0.0) call exit(2)
      if((x2npa22+x2npa11).eq.0.0) call exit(2)
      if(detr.gt.0.0) call exit(2)
      if(detr.eq.0.0) call exit(2)
      return
      end


      function expl(u)
      common /one/ rn,b,re,mxeval,a1,a2
      common /two/ b2,b3,b4,b5,e32b,e42b,e52b,e62b,e72b
      expl=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      progrom tdtsb2(kout,tty,tape5=kout,tape6=tty)
      external fleak
      real keff
      common xpi,b,re,mxeval,bex
      call xsetf(0)
      bbb=.00001
      b=.5166
      c=1.60
      kk=0
      jj=0
      xpi=3.1415926535898
      mxeval=950
      re=.000000001
100   continue
      bex=b+.4455
      coll=4./xpi*bex*sin(xpi*b/(2.*bex))
      xprod=c*coll
      xul=b-.0000000000001
      xleak=c*quad(fleak,0.0,xul,re,mxeval,kount)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.) b=b+bbb
      if(keff.lt.1.) jj=1
      if(keff.gt.1.) b=b-bbb
      if(keff.gt.1.) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      function fleak(x)
      common xpi,b,re,mxeval,bex
      fleak=cos(xpi*x/(2*bex))*(exp(-(b-x))+exp(-(b+x))
     1-(b-x)*e1(b-x)-(b+x)*e1(b+x))
      return
      end
```

```
      program tv2sb3(kout.tty.tape5=kout.tape6=tty)
      dimension xterm1(5000)
      dimension xir(5000)
      dimension xil(5000)
      real keff
      common b3.rn.b.re.mxeval.a1
      call xsetf(0)
      jdim=500
      bbb=.0001
      b=.5120
      c=1.6
      sgt=1.0
      sgs=.4
      sga=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=2.*(b-a1*b3/3.)
      xprod=sgfu*coll
      xabs=sga*coll
      delxx=2.*b/float(jdim-1)
      do 400 i=1,jdim
      if(i.eq.jdim) xterm1(i)=.5
      if(i.eq.jdim) go to 350
      xx=float(i-1)*delxx-b
      xterm1(i)=.5*(exp(-(b-xx))-(b-xx)*e1(b-xx))
350   continue
      xir(i)=xterm1(i)
400   continue
500   continue
      do 1000 i=1,jdim
      aa=0.
      bb=0.
      iu=i-1
      if(iu.eq.0) go to 700
      do 600 j=1,iu
      delx=delxx
      if(j.eq.1) delx=.5*delxx
      xxx=float(i-j)*delxx
      aa=aa+xir(j)*e1(xxx)*delx
600   continue
700   continue
      ii=i+1
      if(ii.eq.jdim+1) go to 900
      do 800 j=ii,jdim
      delx=delxx
      if(j.eq.jdim) delx=.5*delxx
      xxx=float(j-i)*delxx
      bb=bb+xir(j)*e1(xxx)*delx
800   continue
900   continue
      yy=1.0
      if(i.eq.1) yy=.5
      if(i.eq.jdim) yy=.5
      eps=.5*delxx
      st=sgs*xir(i)*(1.-exp(-eps)+eps*e1(eps))*yy
      xil(i)=(aa+bb)*sgs*.5+st+xterm1(i)
1000  continue
      iac=0
      do 1100 i=1,jdim
```

```
      if(abs((xil(i)-xir(i))/xir(i)).ge..00001) icc=1
      xir(i)=xil(i)
1100  continue
      if(icc.eq.1) go to 500
      do 1200 i=1,jdim
      xil(i)=xir(jdim+1-i)
1200  continue
      xleak=0.
      do 1300 i=1,jdim
      delx=delxx
      if(i.eq.1) delx=.5*delxx
      if(i.eq.jdim) delx=.5*delxx
      xx=float(i-1)*delxx-b
      xleak=egfu*(1.-a1*xx*xx)*(xil(i)+xir(1))*delx+xleak
1300  continue
      keff=(xprod-xleak)/xabs
      write(5,1500) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
1500  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expl
      common b3,rn,b,re,mxeval,a1
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      tt1=.4*b5
      tt4=-4./3.*b3
      tt6=2.*b
      rn=3.
      e32b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+18.*b2*e52b+16.*b*e62b+8.*e72b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb6=4.*b-1.+2.*e32b
      ec1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
```

```
 if(a1.lt.0.0) call exit(2)
 vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6))
 batt=bb1*a1*a1+bb4*a1+bb6
 x2ndder=2.*tt1/batt-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
1/batt**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/batt**2.
1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
1+tt6)/batt**3.
 If(x2ndder.lt.0.0) call exit(2)
 If(x2ndder.eq.0.0) call exit(2)
 return
 end


 function expi(u)
 common b3,rn,b.re.mxeval,a1
 expi=u**(rn-2.)*exp(-2.*b/u)
 return
 end
```

```
      program tv4sb3(kout,tty,tape5=kout,tape6=tty)
      dimension xterm1(5000)
      dimension xir(5000)
      dimension xil(5000)
      real keff
      common b3,b5,rn,b,re,mxeval,a1,a2
      call xsetf(0)
      jdim=500
      bbb=.0001
      b=.5120
      c=1.8
      sgt=1.0
      sgs=.4
      sga=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=2.*(b-a1*b3/3.-a2*b5/5.)
      xprod=sgfu*coll
      xabs=sga*coll
      delxx=2.*b/float(jdim-1)
      do 400 i=1,jdim
      if(i.eq.jdim) xterm1(i)=.5
      if(i.eq.jdim) go to 350
      xx=float(i-1)*delxx-b
      xterm1(i)=.5*(exp(-(b-xx))-(b-xx)*e1(b-xx))
350   continue
      xir(i)=xterm1(i)
400   continue
500   continue
      do 1000 i=1,jdim
      aa=0.
      bb=0.
      iu=i-1
      if(iu.eq.0) go to 700
      do 600 j=1,iu
      delx=delxx
      if(j.eq.1) delx=.5*delxx
      xxx=float(i-j)*delxx
      aa=aa+xir(j)*e1(xxx)*delx
600   continue
700   continue
      ii=i+1
      if(ii.eq.jdim+1) go to 900
      do 800 j=ii,jdim
      delx=delxx
      if(j.eq.jdim) delx=.5*delxx
      xxx=float(j-i)*delxx
      bb=bb+xir(j)*e1(xxx)*delx
800   continue
900   continue
      yy=1.0
      if(i.eq.1) yy=.5
      if(i.eq.jdim) yy=.5
      eps=.5*delxx
      st=sgs*xir(i)*(1.-exp(-eps)+eps*e1(eps))*yy
      xil(i)=(aa+bb)*sgs*.5+st+xterm1(i)
1000  continue
      icc=0
      do 1100 i=1,jdim
```

```
      if(abs((xil(i)-xir(i))/xir(i)).ge..00001) icc=1
      xir(i)=xil(i)
1100  continue
      if(icc.eq.1) go to 500
      do 1200 i=1,jdim
      xil(i)=xir(jdim+1-i)
1200  continue
      xleak=0.
      do 1300 l=1,jdim
      delx=delxx
      if(l.eq.1) delx=.5*delxx
      if(l.eq.jdim) delx=.5*delxx
      xx=float(i-1)*delxx-b
      xleak=sgfu*(1.-a1*xx*xx-a2*xx*xx*xx*xx)*(xil(i)+xir(i))
     1*delx+xleak
1300  continue
      keff=(xprod-xleak)/xabs
      write(5,1500) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
1500  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common b3,b5,rn,b,re,mxeval,a1,a2
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      tt1=.4*b5
      tt2=2./9.*b9
      tt3=4./7.*b7
      tt4=-4./3.*b3
      tt5=-.8*b5
      tt6=2.*b
      rn=3.
      e32b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
```

```
     1+16.*b2*e52b+16.*b*e62b+8.*e72b
      bb2=4./9.*b9-b8+16./7.*b7-4.*b6+96./25.*b5-115.2
     1+2.*b8*e32b+16.*b7*e42b+80.*b6*e52b+288.*b5*e62b+768.*b4*e72b
     1+1536.*b3*e82b+2304.*b2*e92b+2304.*b*e102b+1152.*e112b
      bb3=8./7.*b7-2.*b6+56./15.*b5-6.*b4+32./5.*b3-24.+4.*b6*e32b
     1+24.*b5*e42b+88.*b4*e52b+224.*b3*e62b+384.*b2*e72b+384.
     1*b*e82b+192.*e92b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb5=-8./5.*b5+2.*b4-16./3.*b3+12.*b2-96./5.*b+16.
     1-4.*b4*e32b-16.*b3*e42b-48.*b2*e52b-96.*b*e62b-96.*e72b
      bb6=4.*b-1.+2.*e32b
      a1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      a3=2.*tt1*bb6-2.*bb1*tt6
      a4=2.*tt1*bb5-2.*bb1*tt5
      a5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      a17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      If(a11.gt.flxck1) a1=a12
      If(a12.gt.flxck1) a1=a11
      If(a11.lt.0.0) a1=a12
      If(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxak2=1./b4-a1/b2
      If(a21.lt.-a1/(2.*b2)) a2=a22
      If(a22.lt.-a1/(2.*b2)) a2=a21
      If(a21.gt.flxck2) a2=a22
      if(a22.gt.flxck2) a2=a21
      If(abs((test1-a1)/a1).gt..0000001) go to 1000
      If(abs((test2-a2)/a2).gt..0000001) go to 1000
      If(a1.gt.flxck1) call exit(2)
      If(a1.lt.0.0) call exit(2)
      If(a2.gt.flxck2) call exit(2)
      If(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
```

```
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      if((x2npa22+x2npa11).lt.0.0) call exit(2)
      if((x2npa22+x2npa11).eq.0.0) call exit(2)
      if(detr.gt.0.0) call exit(2)
      if(detr.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common b3,b5,rn,b,re,mxeval,a1,a2
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv2spl(kout,tty,tape5=kout,tape6=tty)
      real keff
      common rn,b,re,mxeval,vp
      call xsetf(0)
      bbb=.0001
      b=1.476
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      externcl expi
      common rn,b,re,mxeval,vp
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(exp1,.0000000000001,1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      bb1=4./7.*b7-b6+8./5.*b5-1.5*b4+9.-72.*e92b-144.*b*e82b
     1-144.*b2*e72b-98.*b3*e62b-42.*b4*e52b-12.*b5*e42b-2.*b6*e32b
      bb4=-8./5.*b5+2.*b4-8./3.*b3+3.*b2-4.+24.*e72b+48.*b*e62b+36.
     1*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb6=4./3.*b3-b2+.5-2.*b2*e32b-4.*b*e42b-2.*e52b
      tt1=2./7.*b7
      tt4=-.8*b5
      tt6=2./3.*b3
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
```

```
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      If(a1.gt.flxck) call exit(2)
      If(a1.lt.0.0) call exit(2)
      vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6))
      bott=bb1*a1*a1+bb4*a1+bb6
      x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
     1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
     1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
     1+tt6)/bott**3.
      if(x2ndder.lt.0.0) call exit(2)
      If(x2ndder.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common rn,b,re,mxeval,vp
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv4sp1(kout,tty,tape5=kout,tape6=tty)
      real keff
      common rn,b,re,mxeval,vp
      call xsetf(0)
      bbb=.0001
      b=1.4761
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expl
      common rn,b,re,mxeval,vp
      rn=13.
      e132b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=12.
      e122b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      b10=b9*b
      b11=b10*b
      b12=b11*b
      b13=b12*b
      bb1=4./7.*b7-b6+8./5.*b5-1.5*b4+9.-72.*e92b
     1-144.*b*e82b-144.*b2*e72b-96.*b3*e62b-42.*b4*e52b-12.*b5*e42b
```

```
     1-2.*b6*e32b
      bb2=4./11.*b11-b10+80./27.*b9-15./2.*b8+96./7.*b7-
     140./3.*b6+2400.-28800.*e132b-57600.*b*e122b-
     157600.*b2*e112b-38400.*b3*e102b-19200.*b4*e92b-7680.*
     1b5*e82b-2480.*b6*e72b-640.*b7*e62b-130.*b8*e52b-20.
     1*b9*e42b-2.*b10*e32b
      bb3=8./9.*b9-2.*b8+104./21.*b7-11.*b6+96./5.*b5-20.*b4
     1+288.-2880.*e112b-5760.*b*e102b-5760.*b2*e92b-3840.
     1*b3*e82b-1800.*b4*e72b-624.*b5*e62b-164.*b6*e52b
     1-32.*b7*e42b-4.*b8*e32b
      bb4=-8./5.*b5+2.*b4-8./3.*b3+3.*b2-4.+24.*e72b
     1+48.*b*e62b+36.*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb5=-8./7.*b7+2.*b6-16./3.*b5+15.*b4-32.*b3+40.*b2
     1-60.+480.*e92b+960.*b*e82b+720.*b2*e72b+320.*b3*e62b
     1+100.*b4*e52b+24.*b5*e42b+4.*b6*e32b
      bb6=4./3.*b3-b2+.5-2.*e52b-4.*b*e42b-2.*b2*e32b
      tt1=2./7.*b7
      tt2=2./11.*b11
      tt3=4./9.*b9
      tt4=-.6*b5
      tt5=-4./7.*b7
      tt6=2./3.*b3
      c1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      ac1=a1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      ac3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      ac4=c17+c15*a1
      ac5=c16+c14*a1+c12*a1*a1
      ac6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      fixck1=1./b2-a2*b2
      if(a11.gt.fixck1) a1=a12
      if(a12.gt.fixck1) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      fixck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
```

```
      if(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.flxck2) a2=a22
      if(a22.gt.flxck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
      if(a1.gt.flxck1) call exit(2)
      if(a1.lt.0.0) call exit(2)
      if(a2.gt.flxck2) call exit(2)
      if(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      if((x2npa22+x2npa11).lt.0.0) call exit(2)
      if((x2npa22+x2npa11).eq.0.0) call exit(2)
      if(detr.gt.0.0) call exit(2)
      if(detr.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common rn,b,re,mxeval,vp
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv2sp2(kout,tty,tape5=kout,tape6=tty)
      real keff
      common /one/ rn,b,re,mxeval,a1
      common /two/ b2,b3,b4,b5,e42b,e52b,e62b,e72b
      call xsetf(0)
      bbb=.0001
      b=1.478
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=b3/3.-a1*b5/5.
      xprod=c*coll
      xxll=.25*(b2-.5-b*e42b-2.*e52b+(1.+b)*exp(-2.*b))
     1-.25*a1*(b4-4./3.*b3+1.5*b2-2.-b3*e42b-6.*b2*e52b-18.*b*e62b
     1-24.*e72b+exp(-2.*b)*(b3+3.*b2+6.*b+6.))
      xleak=c*xxll
      keff=xprod/(coll+xleak)
      write(6,200) b,xleak,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common /one/ rn,b,re,mxeval,a1
      common /two/ b2,b3,b4,b5,e42b,e52b,e62b,e72b
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      bb1=4./7.*b7-b6+8./5.*b5-1.5*b4+9.-72.*e92b-144.*b*e82b
     1-144.*b2*e72b-96.*b3*e62b-42.*b4*e52b-12.*b5*e42b-2.*b6*e32b
      bb4=-8./5.*b5+2.*b4-8./3.*b3+3.*b2-4.+24.*e72b+48.*b*e62b+36.
     1*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb8=4./3.*b3-b2+.5-2.*b2*e32b-4.*b*e42b-2.*e52b
      tt1=2./7.*b7
      tt4=-.8*b5
      tt8=2./3.*b3
```

```
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb5))
      bott=bb1*a1*a1+bb4*a1+bb8
      x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
     1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
     1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
     1+tt6)/bott**3.
      if(x2ndder.lt.0.0) call exit(2)
      if(x2ndder.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common /one/ rn,b,re,mxeval,a1
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv4sp2(kout,tty,tape5=kout,tape6=tty)
      real keff
      common /one/ rn,b,re,mxeval,a1,a2
      common /two/ b2,b3,b4,b5,b6,b7,e42b,e52b,e62b,e72b,e82b,e92b
      call xsetf(0)
      bbb=.0001
      b=1.478
      c=1.6
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=b3/3.-a1*b5/5.-a2*b7/7.
      xprod=c*coll
      xxll=.25*(b2-.5-b*e42b-2.*e52b+(1.+b)*exp(-2.*b))
     1-.25*a1*(b*-4./3.*b3+1.5*b2-2.-b3*e42b-6.*b2*e52b-18.*b*e62b
     1-24.*e72b+exp(-2.*b)*(b3+3.*b2+6.*b+6.))-.25*a2*(b6-6./3.*b5
     1+7.5*b4-16.*b3+20.*b2-30.-b5*e42b-10.*b4*e52b-60.*b3*e62b
     1-240.*b2*e72b-600.*b*e82b-720.*e92b+exp(-2.*b)*(b5+5.*b4+20.
     1*b3+60.*b2+120.*b+120.))
      xleak=c*xxll
      keff=xprod/(coll+xleak)
      write(6,200) b,xleak,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common /one/ rn,b,re,mxeval,a1,a2
      common /two/ b2,b3,b4,b5,b6,b7,e42b,e52b,e62b,e72b,e82b,e92b
      rn=13.
      e132b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=12.
      e122b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
```

```
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      b10=b9*b
      b11=b10*b
      b12=b11*b
      b13=b12*b
      bb1=4./7.*b7-b6+8./5.*b5-1.5*b4+9.-72.*e92b
     1-144.*b*e82b-144.*b2*e72b-96.*b3*e82b-42.*b4*e52b-12.*b5*e42b
     1-2.*b6*e32b
      bb2=4./11.*b11-b10+80./27.*b9-15./2.*b8+96./7.*b7-
     140./3.*b6+2400.-28800.*e132b-57600.*b*e122b-
     157600.*b2*e112b-38400.*b3*e102b-19200.*b4*e92b-7680.*
     1b5*e82b-2480.*b6*e72b-840.*b7*e82b-130.*b8*e52b-20.
     1*b9*e42b-2.*b10*e32b
      bb3=8./9.*b9-2.*b8+104./21.*b7-11.*b6+96./5.*b5-20.*b4
     1+288.-2880.*e112b-5760.*b*e102b-5760.*b2*e92b-3840.
     1*b3*e82b-1800.*b4*e72b-624.*b5*e62b-164.*b6*e52b
     1-32.*b7*e42b-4.*b8*e32b
      bb4=-8./5.*b5+2.*b4-8./3.*b3+3.*b2-4.+24.*e72b
     1+48.*b*e62b+36.*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb5=-8./7.*b7+2.*b6-16./3.*b5+15.*b4-32.*b3+40.*b2
     1-60.+480.*e92b+960.*b*e82b+720.*b2*e72b+320.*b3*e62b
     1+100.*b4*e52b+24.*b5*e42b+4.*b6*e32b
      bb6=4./3.*b3-b2+.5-2.*e52b-4.*b*e42b-2.*b2*e32b
      tt1=2./7.*b7
      tt2=2./11.*b11
      tt3=4./9.*b9
      tt4=-.8*b5
      tt5=-4./7.*b7
      tt6=2./3.*b3
      a1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      a12=2.*tt2*bb1-2.*bb2*tt1
      a13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      a14=2.*tt2*bb4-2.*bb2*tt4
      a15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      a18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      ac1=c1+c2*a2
      ac2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      ac4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      ac6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
```

```
a12=(-deta1-cc2)/(2.*cc1)
flxck1=1./b2-a2*b2
if(a11.gt.flxck1) a1=a12
if(a12.gt.flxck1) a1=a11
If(a11.lt.0.0) a1=a12
If(a12.lt.0.0) a1=a11
deta2=sqrt(cc5*cc5-4.*cc4*cc6)
a21=(deta2-cc5)/(2.*cc4)
a22=(-deta2-cc5)/(2.*cc4)
flxck2=1./b4-a1/b2
If(a21.lt.-a1/(2.*a2)) a2=a22
If(a22.lt.-a1/(2.*b2)) a2=a21
If(a21.gt.flxck2) a2=a22
If(a22.gt.flxck2) a2=a21
if(abs((test1-a1)/a1).gt..0000001) go to 1000
If(abs((test2-a2)/a2).gt..0000001) go to 1000
If(a1.gt.flxck1) call exit(2)
if(a1.lt.0.0) call exit(2)
if(a2.gt.flxck2) call exit(2)
If(a2.lt.-a1/(2.*b2)) call exit(2)
vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
vpb2=vpb*vpb
vpb3=vpb2*vpb
vp=2.*vpt/vpb
x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
1*vpt/vpb3
x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
detr=x2npa12*x2npa12-x2npa11*x2npa22
If((x2npa22+x2npa11).lt.0.0) call exit(2)
If((x2npa22+x2npa11).eq.0.0) call exit(2)
if(detr.gt.0.0) call exit(2)
If(detr.eq.0.0) call exit(2)
return
end


function expl(u)
common /one/ rn,b,re.mxeval,a1,a2
expl=u**(rn-2.)*exp(-2.*b/u)
return
end
```

```
      program tdtsp2(kout.tty,tape5=kout.tape6=tty)
      external fleak
      real keff
      common xpi,b,re,mxeval,bex
      call xsetf(0)
      bbb=.00001
      b=1.4948
      c=1.80
      kk=0
      jj=0
      xpi=3.14159265358898
      mxeval=950
      re=.000000001
100   continue
      bex=b+.4455
      coll=bex/xpi*(bex/xpi*sin(xpi/bex*b)-b*cos(xpi/bex*b))
      xprod=c*coll
      xul=b-.0000000000001
      xleak=.25*c*quad(fleak,0.0,xul,re,mxeval,kount)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.) b=b+bbb
      if(keff.lt.1.) jj=1
      if(keff.gt.1.) b=b-bbb
      if(keff.gt.1.) kk=1
      if(jj+kk.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      function fleak(r)
      common xpi,b,re,mxeval,bex
      fleak=sin(xpi/bex*r)*((b*b-r*r)*(e1(b+r)-e1(b-r))+(b+r+1)
     1*exp(-(b-r))-(b-r+1)*exp(-(b+r)))
      return
      end
```

```
      program tv2sp3(kout,tty,tape5=kout,tape6=tty)
      dimension xterm1(5000)
      dimension rxi(5000)
      dimension test(5000)
      real keff
      common b3,b5,rn,b,re,mxeval,a1
      call xsetf(0)
      jdim=1001
      bbb=.0001
      b=1.4768
      c=1.8
      sgt=1.0
      sgs=.4
      sga=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=b3/3.-a1*b5/5.
      xprod=sgfu*coll
      xabs=sga*coll
      delrr=2.*b/float(jdim-1)
      kdim=(jdim+1)/2
      iuxi=kdim-1
      do 400 i=kdim,jdim
      if(i.eq.jdim) xterm1(i)=.25*(2.*b+1.-exp(-2.*b))
      if(i.eq.jdim) go to 350
      rr=float(i-1)*delrr-b
      xterm1(i)=.25*((b*b-rr*rr)*(e1(b+rr)-e1(b-rr))+(b+rr+1.)*
     1exp(-(b-rr))-(b-rr+1.)*exp(-(b+rr)))
350   continue
      rxi(i)=xterm1(i)
400   continue
500   continue
      do 550 i=1,iuxi
      rxi(i)=-rxi(jdim+1-i)
550   continue
      do 1000 i=kdim,jdim
      aa=0.
      bb=0.
      iu=i-1
      do 600 j=1,iu
      delr=delrr
      if(j.eq.1) delr=.5*delrr
      rrx=float(i-j)*delrr
      aa=aa+rxi(j)*e1(rrx)*delr
600   continue
      ii=i+1
      if(ii.eq.jdim+1) go to 900
      do 800 j=ii,jdim
      delr=delrr
      if(j.eq.jdim) delr=.5*delrr
      rrx=float(j-i)*delrr
      bb=bb+rxi(j)*e1(rrx)*delr
800   continue
900   continue
      yy=1.0
      if(i.eq.jdim) yy=.5
      eps=.5*delrr
      et=sge*rxi(i)*(1.-exp(-eps)+eps*e1(eps))*yy
      test(i)=(aa+bb)*sgs*.5+st+xterm1(i)
```

```
1000  continue
      icc=0
      rxi(kdim)=0.0
      kkdim=kdim+1
      do 1100 i=kkdim,jdim
      if(abs((test(i)-rxi(i))/rxi(i)).ge..00001) icc=1
      rxi(i)=test(i)
1100  continue
      if(icc.eq.1) go to 500
      xleak=0.
      do 1300 i=kdim,jdim
      delr=delrr
      if(i.eq.jdim) delr=.5*delrr
      rr=float(i-1)*delrr-b
      xleak=sgfu*(1.-a1*rr*rr)*rxi(i)*rr*delr+xleak
1300  continue
      keff=(xprod-xleak)/xabs
      write(5,1500) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
1500  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common b3,b5,rn,b,re,mxeval,a1
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001.1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001.1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi..0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(expi,.0000000000001.1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      bb1=4./7.*b7-b6+6./5.*b5-1.5*b4+9.-72.*e92b-144.*b*e82b
     1-144.*b2*e72b-96.*b3*e62b-42.*b4*e52b-12.*b5*e42b-2.*b6*e32b
      bb4=-6./5.*b5+2.*b4-6./3.*b3+3.*b2-4.+24.*e72b+48.*b*e62b+36.
     1*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb6=4./3.*b3-b2+.5-2.*b2*e32b-4.*b*e42b-2.*e52b
      tt1=2./7.*b7
      tt4=-.8*b5
      tt6=2./3.*b3
      ac1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      ac3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
```

```
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=2.*((tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6))
      bott=bb1*a1*a1+bb4*a1+bb6
      x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
     1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
     1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
     1+tt6)/bott**3.
      if(x2ndder.lt.0.0) call exit(2)
      if(x2ndder.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common b3,b5,rn,b,re,mxeval,a1
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv4sp3(kout,tty,tape5=kout,tape6=tty)
      dimension xterm1(5000)
      dimension rxi(5000)
      dimension test(5000)
      real keff
      common b3,b5,b7,rn,b,re,mxeval,a1,a2
      call xsetf(0)
      jdim=1001
      bbb=.0001
      b=1.4759
      c=1.6
      sgt=1.0
      sge=.4
      sgs=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      coll=b3/3.-a1*b5/5.-a2*b7/7.
      xprod=sgfu*coll
      xabs=sge*coll
      delrr=2.*b/float(jdim-1)
      kdim=(jdim+1)/2
      iuxl=kdim-1
      do 400 i=kdim,jdim
      if(i.eq.jdim) xterm1(i)=.25*(2.*b+1.-exp(-2.*b))
      if(i.eq.jdim) go to 350
      rr=float(i-1)*delrr-b
      xterm1(i)=.25*((b*b-rr*rr)*(e1(b+rr)-e1(b-rr))+(b+rr+1.)*
     1exp(-(b-rr))-(b-rr+1.)*exp(-(b+rr)))
350   continue
      rxi(i)=xterm1(i)
400   continue
500   continue
      do 550 i=1,iuxl
      rxi(i)=-rxi(jdim+1-i)
550   continue
      do 1000 i=kdim,jdim
      aa=0.
      bb=0.
      iu=i-1
      do 800 j=1,iu
      delr=delrr
      if(j.eq.1) delr=.5*delrr
      rrx=float(i-j)*delrr
      aa=aa+rxi(j)*e1(rrx)*delr
800   continue
      ii=i+1
      if(ii.eq.jdim+1) go to 900
      do 800 j=ii,jdim
      delr=delrr
      if(j.eq.jdim) delr=.5*delrr
      rrx=float(j-i)*delrr
      bb=bb+rxi(j)*e1(rrx)*delr
800   continue
900   continue
      yy=1.0
      if(i.eq.jdim) yy=.5
      epe=.5*delrr
      et=sge*rxi(i)*(1.-exp(-epe)+eps*e1(eps))*yy
      test(i)=(aa+bb)*sge*.5+st+xterm1(i)
```

```
1000  continue
      icc=0
      rxi(kdim)=0.0
      kkdim=kdim+1
      do 1100 i=kkdim,jdim
      if(abs((test(i)-rxi(i))/rxi(i)).ge..00001) icc=1
      rxi(i)=test(i)
1100  continue
      if(icc.eq.1) go to 500
      xleak=0.
      do 1300 i=kdim,jdim
      delr=delrr
      if(i.eq.jdim) delr=.5*delrr
      rr=float(i-1)*delrr-b
      xleak=sgfu*(1.-a1*rr*rr-a2*rr*rr*rr*rr)*rxi(i)
     1*rr*delr+xleak
1300  continue
      keff=(xprod-xleak)/xabs
      write(5,1500) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(jj+kk.lt.2) go to 100
1500  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expi
      common b3,b5,b7,rn,b,re,mxeval,a1,a2
      rn=13.
      e132b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=12.
      e122b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      rn=3.
      e32b=quad(expi,.0000000000001,1.,re,mxeval,kount)
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      b10=b9*b
      b11=b10*b
```

```
      b12=b11*b
      b13=b12*b
      bb1=4./7.*b7-b6+8./5.*b5-1.5*b4+9.-72.*e92b
     1-144.*b*e82b-144.*b2*e72b-96.*b3*e62b-42.*b4*e52b-12.*b5*e42b
     1-2.*b6*e32b
      bb2=4./11.*b11-b10+80./27.*b9-15./2.*b8+96./7.*b7-
     140./3.*b6+2400.-28800.*e132b-57600.*b*e122b-
     157600.*b2*e112b-38400.*b3*e102b-19200.*b4*e92b-7680.*
     1b5*e82b-2480.*b6*e72b-640.*b7*e62b-130.*b8*e52b-20.
     1*b9*e42b-2.*b10*e32b
      bb3=8./9.*b9-2.*b8+104./21.*b7-11.*b6+96./5.*b5-20.*b4
     1+288.-2880.*e112b-5760.*b*e102b-5760.*b2*e92b-3840.
     1*b3*e82b-1800.*b4*e72b-624.*b5*e62b-164.*b6*e52b
     1-32.*b7*e42b-4.*b8*e32b
      bb4=-8./5.*b5+2.*b4-8./3.*b3+3.*b2-4.+24.*e72b
     1+48.*b*e82b+36.*b2*e52b+16.*b3*e42b+4.*b4*e32b
      bb5=-8./7.*b7+2.*b6-16./3.*b5+15.*b4-32.*b3+40.*b2
     1-60.+480.*e92b+960.*b*e82b+720.*b2*e72b+320.*b3*e62b
     1+100.*b4*e52b+24.*b5*e42b+4.*b6*e32b
      bb6=4./3.*b3-b2+.5-2.*e52b-4.*b*e42b-2.*b2*e32b
      tt1=2./7.*b7
      tt2=2./11.*b11
      tt3=4./9.*b9
      tt4=-.8*b5
      tt5=-4./7.*b7
      tt6=2./3.*b3
      a1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      a9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt5-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      aa1=c1+c2*a2
      aa2=c3+c4*a2+c5*a2*a2
      aa3=c9+c8*a2+c7*a2*a2+c6*a2*a2*a2
      cc4=c17+c15*a1
      cc8=c16+c14*a1+a12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      If(a11.gt.flxck1) a1=a12
      If(a12.gt.flxck1) a1=a11
      If(a11.lt.0.0) a1=a12
      If(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc8)
```

```
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      fixck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
      if(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.fixck2) a2=a22
      if(a22.gt.fixck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
      if(a1.gt.fixck1) call exit(2)
      if(a1.lt.0.0) call exit(2)
      if(a2.gt.fixck2) call exit(2)
      if(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      if((x2npa22+x2npa11).lt.0.0) call exit(2)
      if((x2npa22+x2npa11).eq.0.0) call exit(2)
      if(detr.gt.0.0) call exit(2)
      if(detr.eq.0.0) call exit(2)
      return
      end


      function expi(u)
      common b3,b5,b7,rn,b,re,mxeval,a1,a2
      expi=u**(rn-2.)*exp(-2.*b/u)
      return
      end
```

```
      program tv2c11(kout,tty,tape5=kout,tape6=tty)
      real keff
      common b,re,mxeval,vp
      call xsetf(0)
      bbb=.0001
      b=1.0209
      c=1.8
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb4
      external fbb6
      common b,re,mxeval,vp
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb4=quad(fbb4,.0000000000001,b,re,mxeval,kount)
      bb4=-28./45.*b**5.+xbb4
      xbb6=quad(fbb6,.0000000000001,b,re,mxeval,kount)
      bb6=2./3.*b**3.-2.*xbb6
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      tt1=b6/6.
      tt4=-.5*b4
      tt6=b2/2.
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=(tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6)
      bott=bb1*a1*a1+bb4*a1+bb6
      x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
     1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
     1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
     1+tt6)/bott**3.
      if(x2ndder.lt.0.0) call exit(2)
      if(x2ndder.eq.0.0) call exit(2)
```

```
 return
 end


 function fbb1(r)
 common b,re,mxeval,vp
 fbb1=besi1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besi1(r)*besk0(r)
1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**8.*besk1(r)*besi0(r)
 return
 end


 function fbb4(r)
 common b,re,mxeval,vp
 fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
1-4./9.*r**4.*besk1(r)*besi0(r)
 return
 end


 function fbb6(r)
 common b,re,mxeval,vp
 fbb6=r**3.*(besi1(r)*besk1(r)+besk0(r)*besi0(r))-r*r
1*besk0(r)*besi1(r)
 return
 end
```

```
      program tv4c11(kout,tty,tape5=kout,tape6=tty)
      real keff
      common b,re,mxeval,vp
      call xsetf(0)
      bbb=.000001
      b=1.020859
      c=1.8
      kk=0
      jj=0
      mxeval=950
      re=.000000001
100   continue
      call flux
      keff=c-vp
      write(6,200) b,bbb,keff
      if(keff.lt.0.0) b=b+bbb
      if(keff.lt.0.0) jj=1
      if(keff.gt.0.0) b=b-bbb
      if(keff.gt.0.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb2
      external fbb3
      external fbb4
      external fbb5
      external fbb6
      common b,re,mxeval,vp
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb2=quad(fbb2,.0000000000001,b,re,mxeval,kount)
      bb2=178./2475.*b**11.-xbb2
      xbb3=quad(fbb3,.0000000000001,b,re,mxeval,kount)
      bb3=428./2025.*b**9.-xbb3
      xbb4=quad(fbb4,.0000000000001,b,re,mxeval,kount)
      bb4=-28./45.*b**5.+xbb4
      xbb5=quad(fbb5,.0000000000001,b,re,mxeval,kount)
      bb5=-828./1575.*b**7.+2.*xbb5
      xbb6=quad(fbb6,.0000000000001,b,re,mxeval,kount)
      bb6=2./3.*b**3.-2.*xbb8
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      b8=b6*b2
      b10=b8*b2
      tt1=b8/6.
      tt2=.1*b10
      tt3=.25*b8
      tt4=-.5*b4
      tt5=-1./3.*b6
      tt6=b2/2.
      c1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt5-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
```

```
      c9=tt4*bb5-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      a17=tt2*bb5-bb2*tt5
      a18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      ce1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      If(a11.gt.flxck1) a1=a12
      If(a12.gt.flxck1) a1=a11
      If(a11.lt.0.0) a1=a12
      If(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
      If(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.flxck2) a2=a22
      if(a22.gt.flxck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
      If(a1.gt.flxck1) call exit(2)
      If(a1.lt.0.0) call exit(2)
      If(a2.gt.flxck2) call exit(2)
      If(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      If((x2npa22+x2npa11).lt.0.0) call exit(2)
      If((x2npa22+x2npa11).eq.0.0) call exit(2)
      If(detr.gt.0.0) call exit(2)
      If(detr.eq.0.0) call exit(2)
```

```
      return
      end


      function fbb1(r)
      common b,re,mxeval,vp
      fbb1=besi1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
     1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besl1(r)*besk0(r)
     1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**6.*besk1(r)*besl0(r)
      return
      end


      function fbb2(r)
      common b,re,mxeval,vp
      fbb2=besk1(r)*besi1(r)*(178./225.*r**11.+418./225.*r**9.
     1+128./25.*r**7.)+besk0(r)*besi0(r)*(176./225.*r**11.+184./75.
     1*r**9.+320./25.*r**7.)-besk0(r)*besl1(r)*(314./225.
     1*r**10.+608./75.*r**8.+640./25.*r**6.)-besk1(r)*besi0(r)*
     1(136./225.*r**10.+64./25.*r**8.)
      return
      end


      function fbb3(r)
      common b,re,mxeval,vp
      fbb3=besk1(r)*besl1(r)*(428./225.*r**9.+616./225.*r**7.
     1+384./75.*r**5.)+besi0(r)*besk0(r)*(428./225.*r**9.+284./75.
     1*r**7.+320./25.*r**5.)-besk0(r)*besl1(r)*(664./225.*r**8.+
     12424./225.*r**6.+640./25.*r**4.)-besi0(r)*besk1(r)*(236./225.
     1*r**8.+64./25.*r**6.)
      return
      end


      function fbb4(r)
      common b,re,mxeval,vp
      fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
     1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
     1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
     1-4./9.*r**4.*besk1(r)*besl0(r)
      return
      end


      function fbb5(r)
      common b,re,mxeval,vp
      fbb5=besl1(r)*besk1(r)*(314./225.*r**7.+208./225.*r**5.+384.
     1/150.*r**3.)+besk0(r)*besl0(r)*(314./225.*r**7.+92./75.
     1*r**5.+32./5.*r**3.)-besk0(r)*besi1(r)*(382./225.*r**6.
     1+304./75.*r**4.+84./5.*r**2.)-besi0(r)*besk1(r)*(68./225.
     1*r**8.+32./25.*r**4.)
      return
      end


      function fbb6(r)
      common b,re,mxeval,vp
      fbb6=r**3.*(besl1(r)*besk1(r)+besk0(r)*besl0(r))-r*r
     1*besk0(r)*besi1(r)
      return
      end
```

```
      program tv2c12(kout,tty,tape5=kout,tape6=tty)
      external fleak
      real keff
      common b,re,mxeval,al
      call xsetf(0)
      bbb=.000001
      b=1.020884
      c=1.80
      kk=0
      jj=0
      mxeval=950
      re=.000000001
      xpi=3.1415926535898
100   continue
      call flux
      coll=2.*xpi*(.5*b*b-.25*a1*b**4.)
      xprod=c*coll
      xleak=2.*c*xpi*b*quad(fleak,1.0,100000.,re,mxeval,kount)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb4
      external fbb8
      common b,re,mxeval,a1
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb4=quad(fbb4,.0000000000001,b,re,mxeval,kount)
      bb4=-28./45.*b**5.+xbb4
      xbb6=quad(fbb6,.0000000000001,b,re,mxeval,kount)
      bb6=2./3.*b**3.-2.*xbb6
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      tt1=b6/6.
      tt4=-.5*b4
      tt6=b2/2.
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb8-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt8
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      fixck=1./b2
      a1=-500.
      if(a11.gt.fixck) a1=a12
      if(a12.gt.fixck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.fixck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=(tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6)
      bott=bb1*a1*a1+bb4*a1+bb6
      x2nddar=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
```

```
1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
1+tt6)/bott**3.
 if(x2ndder.lt.0.0) call exit(2)
 if(x2ndder.eq.0.0) call exit(2)
 return
 end


 function fbb1(r)
 common b,re,mxeval,a1
 fbb1=besl1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besi1(r)*besk0(r)
1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**6.*besk1(r)*besi0(r)
 return
 end


 function fbb4(r)
 common b,re,mxeval,a1
 fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
1-4./9.*r**4.*besk1(r)*besi0(r)
 return
 end


 function fbb6(r)
 common b,re,mxeval,a1
 fbb6=r**3.*(besl1(r)*besk1(r)+besk0(r)*besl0(r))-r*r
1*besk0(r)*besl1(r)
 return
 end


 function fleak(x)
 common b,re,mxeval,a1
 bbx=b*x
 fleak=besk1e(bbx)*(besl1e(bbx)*(b/(x*x)-a1*b**3./(x*x)
1-4.*a1*b/(x**4.))+besi0e(bbx)*(2.*a1*b*b/(x**3.)))
 return
 end
```

```
      program tv4c12(kout.tty.tape5=kout.tape6=tty)
      external fleak
      real keff
      common b,re,mxeval,a1,a2
      call xsetf(0)
      bbb=.000001
      b=1.020859
      c=1.80
      kk=0
      jj=0
      mxeval=950
      re=.000000001
      xpi=3.1415926535898
100   continue
      call flux
      coll=2.*xpi*(.5*b*b-.25*a1*b**4.-1./8.*a2*b**8.)
      xprod=c*coll
      xleak=2.*c*xpi*b*quad(fleak,1.0.100000.,re.mxeval.kount)
      keff=xprod/(coll+xleak)
      write(6,200) b.bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb2
      external fbb3
      external fbb4
      external fbb5
      external fbb6
      common b,re,mxeval,a1,a2
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb2=quad(fbb2,.0000000000001,b,re,mxeval,kount)
      bb2=178./2475.*b**11.-xbb2
      xbb3=quad(fbb3,.0000000000001,b,re.mxeval,kount)
      bb3=428./2025.*b**9.-xbb3
      xbb4=quad(fbb4..0000000000001,b.re,mxeval.kount)
      bb4=-28./45.*b**5.+xbb4
      xbb5=quad(fbb5,.0000000000001.b.re,mxeval,kount)
      bb5=-628./1575.*b**7.+2.*xbb5
      xbb6=quad(fbb6..0000000000001,b,re.mxeval.kount)
      bb6=2./3.*b**3.-2.*xbb6
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      b8=b6*b2
      b10=b8*b2
      tt1=b6/6.
      tt2=.1*b10
      tt3=.25*b8
      tt4=-.5*b4
      tt5=-1./3.*b6
      tt6=b2/2.
      a1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt8
```

```
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      a12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      a14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      a16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      g2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      If(a11.gt.flxck1) a1=a12
      If(a12.gt.flxck1) a1=a11
      If(a11.lt.0.0) a1=a12
      If(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxck2=1./b4-a1/b2
      If(a21.lt.-a1/(2.*b2)) a2=a22
      If(a22.lt.-a1/(2.*b2)) a2=a21
      If(a21.gt.flxck2) a2=a22
      If(a22.gt.flxck2) a2=a21
      If(abs((test1-a1)/a1).gt..0000001) go to 1000
      If(abs((test2-a2)/a2).gt..0000001) go to 1000
      If(a1.gt.flxck1) call exit(2)
      If(a1.lt.0.0) call exit(2)
      If(a2.gt.flxck2) call exit(2)
      If(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
```

```
      detr=x2npa12*x2npa12-x2npa11*x2npa22
      if((x2npa22+x2npa11).lt.0.0) call exit(2)
      if((x2npa22+x2npa11).eq.0.0) call exit(2)
      if(detr.gt.0.0) call exit(2)
      If(detr.eq.0.0) call exit(2)
      return
      end


      function fbb1(r)
      common b,re,mxeval,a1,a2
      fbb1=besi1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
     1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besi1(r)*besk0(r)
     1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**6.*besk1(r)*besi0(r)
      return
      end


      function fbb2(r)
      common b,re,mxeval,a1,a2
      fbb2=besk1(r)*besi1(r)*(178./225.*r**11.+416./225.*r**9.
     1+128./25.*r**7.)+besk0(r)*besi0(r)*(178./225.*r**11.+184./75.
     1*r**9.+320./25.*r**7.)-besk0(r)*besi1(r)*(314./225.
     1*r**10.+608./75.*r**8.+640./25.*r**6.)-besk1(r)*besi0(r)*
     1(136./225.*r**10.+64./25.*r**8.)
      return
      end


      function fbb3(r)
      common b,re,mxeval,a1,a2
      fbb3=besk1(r)*besi1(r)*(428./225.*r**9.+616./225.*r**7.
     1+384./75.*r**5.)+besi0(r)*besk0(r)*(428./225.*r**9.+284./75.
     1*r**7.+320./25.*r**5.)-besk0(r)*besi1(r)*(664./225.*r**8.+
     12424./225.*r**6.+640./25.*r**4.)-besi0(r)*besk1(r)*(236./225.
     1*r**8.+64./25.*r**6.)
      return
      end


      function fbb4(r)
      common b,re,mxeval,a1,a2
      fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
     1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
     1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
     1-4./9.*r**4.*besk1(r)*besi0(r)
      return
      end


      function fbb5(r)
      common b,re,mxeval,a1,a2
      fbb5=besi1(r)*besk1(r)*(314./225.*r**7.+208./225.*r**5.+384.
     1/150.*r**3.)+besk0(r)*besi0(r)*(314./225.*r**7.+92./75.
     1*r**5.+32./5.*r**3.)-besk0(r)*besi1(r)*(382./225.*r**6.
     1+304./75.*r**4.+64./5.*r**2.)-besi0(r)*besk1(r)*(68./225.
     1*r**6.+32./25.*r**4.)
      return
      end


      function fbb8(r)
      common b,re,mxeval,a1,a2
      fbb8=r**3.*(besi1(r)*besk1(r)+besk0(r)*besi0(r))-r*r
```

```
1*besk0(r)*besi1(r)
 return
 end


 function fleak(x)
 common b,re,mxeval,a1,a2
 bbx=b*x
 fleak=besk1e(bbx)*(besi1e(bbx)*(b/(x*x)-a1*b**3./(x*x)
1-4.*a1*b/(x**4.)-a2*b**5./(x*x)-16.*a2*b**3./(x**4.)
1-84.*a2*b/(x**6.))+besi0e(bbx)*(2.*a1*b**2./(x**3.)+
14.*a2*b**4./(x**3.)+32.*a2*b**2./(x**5.)))
 return
 end
```

```
      program tdtc12(kout,tty,tape5=kout,tape6=tty)
      external fleak
      real keff
      common xpi,b,re,mxeval,bex
      call xsetf(0)
      bbb=.00001
      b=1.0327
      c=1.80
      kk=0
      jj=0
      mxeval=950
      re=.000000001
      xpi=3.1415926535898
100   continue
      bex=b+.4455
      coll=bex*2.*xpi*b*besj1(2.405*b/bex)/2.405
      xprod=c*coll
      xleak=2.*xpi*b*c*quad(fleak,1.0,100000.,re,mxeval,kount)
      keff=xprod/(coll+xleak)
      write(6,200) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
200   format(1x,3f20.10)
      call exit(2)
      end


      function fleak(x)
      common xpi,b,re,mxeval,bex
      ck1=2.405*b/bex
      ck2=2.405/bex
      fleak=besk1e(b*x)/(x**3.+x*ck2**2.)*(ck1*besj1(ck1)
     1*besi0e(b*x)+b*x*besj0(ck1)*besi1e(b*x))
      return
      end
```

```
      program tv2c13(kout.tty,tape5=kout.tape6=tty)
      external term1
      external xtr1
      external xtr2
      external xtr3
      dimension x1(500)
      dimension xterm1(500)
      dimension xterm2(500)
      dimension test(500)
      dimension xtr(500.500)
      real keff
      common b.re.mxeval.rr,rp.a1
      call xsetf(0)
      jdim=500
      bbb=.0001
      b=1.0267
      c=1.6
      sgt=1.0
      sgs=0.4
      sga=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      xpi=3.1415926535898
100   continue
      re=.000000001
      call flux
      re=.0001
      coll=2.*xpi*(b*b*.5-a1*b**4.*.25)
      xprod=sgfu*coll
      xabs=sga*coll
      delrr=b/float(jdim-1)
      do 200 i=1,jdim
      rr=float(i-1)*delrr
      xterm1(i)=b*quad(term1.1.0.2500.,re.mxeval,kount)
      x1(i)=xterm1(i)
200   continue
      do 800 i=1,jdim
      rr=float(i-1)*delrr
      ixll=i+1
      if(ixll.eq.jdim+1) go to 400
      do 300 n=ixll.jdim
      rp=float(n-1)*delrr
      xtr(I.n)=qnc7(xtr1.1.0,2500..re.mxeval,kount)
300   continue
400   continue
      if(I.eq.1) xtr(I.I)=0.0
      if(i.eq.1) go to 500
      xtr(i,i)=qnc7(xtr2.1.0,1000000.,re.mxeval,kount)
500   continue
      ixul=I-1
      if(ixul.eq.0) go to 700
      do 600 n=1,ixul
      rp=float(n-1)*delrr
      xtr(I,n)=qnc7(xtr3,1.0,2500.,re,mxeval,kount)
600   continue
700   continue
800   continue
900   continue
      do 1400 i=1,jdim
      xterm2(I)=0.0
      do 1000 n=1.jdim
      rp=float(n-1)*delrr
```

```
      delr=delrr
      if(n.eq.1) delr=.5*delrr
      if(n.eq.jdim) delr=.5*delrr
      xterm2(i)=xterm2(i)+delr*rp*xi(n)*xtr(i,n)
1000  continue
1400  continue
      icc=0
      do 1500 i=1,jdim
      test(i)=xterm1(i)+sgs*xterm2(i)
      if(abs((test(i)-xl(i))/xl(i)).gt..00001) icc=1
      xl(i)=test(i)
1500  continue
      if(icc.eq.1) go to 900
      xleak=0.0
      do 1800 i=1,jdim
      rr=float(i-1)*delrr
      delr=delrr
      if(i.eq.jdim) delr=.5*delrr
      xleak=xleak+sgfu*(1.-a1*rr*rr)*2.*xpi*rr*delr*xi(i)
1600  continue
      keff=(xprod-xleak)/xabs
      write(6,1700) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
1700  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb4
      external fbb6
      common b,re,mxeval,rr,rp,a1
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb4=quad(fbb4,.0000000000001,b,re,mxeval,kount)
      bb4=-28./45.*b**5.+xbb4
      xbb6=quad(fbb6,.0000000000001,b,re,mxeval kount)
      bb6=2./3.*b**3.-2.*xbb6
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      tt1=b6/6.
      tt4=-.5*b4
      tt6=b2/2.
      cc1=tt1*bb4-bb1*tt4
      cc2=2.*tt1*bb6-2.*bb1*tt6
      cc3=tt4*bb6-bb4*tt6
      det=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(det-cc2)/(2.*cc1)
      a12=(-det-cc2)/(2.*cc1)
      flxck=1./b2
      a1=-500.
      if(a11.gt.flxck) a1=a12
      if(a12.gt.flxck) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      if(a1.gt.flxck) call exit(2)
      if(a1.lt.0.0) call exit(2)
      vp=(tt1*a1*a1+tt4*a1+tt6)/(bb1*a1*a1+bb4*a1+bb6)
```

```
 bott=bb1*a1*a1+bb4*a1+bb6
 x2ndder=2.*tt1/bott-2.*(2.*bb1*a1+bb4)*(2.*tt1*a1+tt4)
1/bott**2.-2.*bb1*(tt1*a1*a1+tt4*a1+tt6)/bott**2.
1+2.*(2.*bb1*a1+bb4)*(2.*bb1*a1+bb4)*(tt1*a1*a1+tt4*a1
1+tt5)/bott**3.
 if(x2ndder.lt.0.0) call exit(2)
 if(x2ndder.eq.0.0) call exit(2)
 return
 end


 funotion fbb1(r)
 common b,re,mxeval,rr,rp,a1
 fbb1=besi1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besi1(r)*besk0(r)
1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**8.*besk1(r)*besi0(r)
 return
 end


 function fbb4(r)
 common b,re,mxeval,rr,rp,a1
 fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
1-4./9.*r**4.*besk1(r)*besi0(r)
 return
 end


 function fbb8(r)
 common b,re,mxeval,rr,rp,a1
 fbb8=r**3.*(besi1(r)*besk1(r)+besk0(r)*besi0(r))-r*r
1*besk0(r)*besi1(r)
 return
 end


 function term1(x)
 common b,re,mxeval,rr,rp,a1
 term1=besk1e(b*x)*besi0e(rr*x)*exp(-(b-rr)*x)/x
 return
 end


 function xtr1(y)
 common b,re,mxeval,rr,rp,a1
 xtr1=besk0e(y*rp)*besi0e(y*rr)*exp(-(rp-rr)*y)
 return
 end


 function xtr2(y)
 common b,re,mxeval,rr,rp,a1
 xtr2=besk0e(y*rr)*besi0e(y*rr)
 return
 end


 function xtr3(y)
 common b,re,mxeval,rr,rp,a1
 xtr3=besk0e(y*rr)*besi0e(y*rp)*exp(-(rr-rp)*y)
 return
 end
```

```
      program tv4c13(kout,tty,tape5=kout,tape6=tty)
      external term1
      external xtr1
      external xtr2
      external xtr3
      dimension xi(500)
      dimension xterm1(500)
      dimension xterm2(500)
      dimension test(500)
      dimension xtr(500,500)
      real keff
      common b,re,mxeval,rr,rp,a1,a2
      call xsetf(0)
      jdim=500
      bbb=.0001
      b=1.0287
      c=1.6
      sgt=1.0
      sgs=0.4
      sga=sgt-sgs
      sgfu=c*sgt-sgs
      kk=0
      jj=0
      mxeval=950
      xpi=3.1415926535898
100   continue
      re=.000000001
      call flux
      re=.0001
      coll=2.*xpi*(b*b*.5-a1*b**4.*.25-a2*b**6./6.)
      xprod=sgfu*coll
      xabs=sga*coll
      delrr=b/float(jdim-1)
      do 200 i=1,jdim
      rr=float(i-1)*delrr
      xterm1(i)=b*quad(term1,1.0,2500.,re,mxeval,kount)
      xi(i)=xterm1(i)
200   continue
      do 800 i=1,jdim
      rr=float(i-1)*delrr
      ixii=i+1
      if(ixii.eq.jdim+1) go to 400
      do 300 n=ixii,jdim
      rp=float(n-1)*delrr
      xtr(i,n)=qnc7(xtr1,1.0,2500.,re,mxeval,kount)
300   continue
400   continue
      if(i.eq.1) xtr(i,i)=0.0
      if(i.eq.1) go to 500
      xtr(i,i)=qnc7(xtr2,1.0,1000000.,re,mxeval,kount)
500   continue
      ixui=i-1
      if(ixui.eq.0) go to 700
      do 600 n=1,ixui
      rp=float(n-1)*delrr
      xtr(i,n)=qnc7(xtr3,1.0,2500.,re,mxeval,kount)
600   continue
700   continue
800   continue
900   continue
      do 1400 i=1,jdim
      xterm2(i)=0.0
      do 1000 n=1,jdim
      rp=float(n-1)*delrr
```

```
      delr=delrr
      if(n.eq.1) delr=.5*delrr
      if(n.eq.jdim) delr=.5*delrr
      xterm2(i)=xterm2(i)+delr*rp*xi(n)*xtr(i,n)
1000  continue
1400  continue
      icc=0
      do 1500 i=1,jdim
      test(i)=xterm1(i)+sgs*xterm2(i)
      if(abs((test(i)-xl(i))/xi(i)).gt..00001) icc=1
      xl(i)=test(i)
1500  continue
      if(icc.eq.1) go to 900
      xleak=0.0
      do 1600 i=1,jdim
      rr=float(i-1)*delrr
      delr=delrr
      if(i.eq.jdim) delr=.5*delrr
      xleak=xleak+sgfu*(1.-a1*rr**2.-a2*rr**4.)*2.*xpi*rr*delr*xi(i)
1600  continue
      keff=(xprod-xleak)/xabs
      write(6,1700) b,bbb,keff
      if(keff.lt.1.0) b=b+bbb
      if(keff.lt.1.0) jj=1
      if(keff.gt.1.0) b=b-bbb
      if(keff.gt.1.0) kk=1
      if(kk+jj.lt.2) go to 100
1700  format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external fbb1
      external fbb2
      external fbb3
      external fbb4
      external fbb5
      external fbb6
      common b,re,mxeval,rr,rp,a1,a2
      xbb1=quad(fbb1,.0000000000001,b,re,mxeval,kount)
      bb1=10./63.*b**7.-xbb1
      xbb2=quad(fbb2,.0000000000001,b,re,mxeval,kount)
      bb2=178./2475.*b**11.-xbb2
      xbb3=quad(fbb3,.0000000000001,b,re,mxeval,kount)
      bb3=428./2025.*b**9.-xbb3
      xbb4=quad(fbb4,.0000000000001,b,re,mxeval,kount)
      bb4=-28./45.*b**5.+xbb4
      xbb5=quad(fbb5,.0000000000001,b,re,mxeval,kount)
      bb5=-528./1575.*b**7.+2.*xbb5
      xbb6=quad(fbb6,.0000000000001,b,re,mxeval,kount)
      bb6=2./3.*b**3.-2.*xbb6
      b2=b*b
      b4=b2*b2
      b6=b4*b2
      b8=b6*b2
      b10=b8*b2
      tt1=b6/6.
      tt2=.1*b10
      tt3=.25*b8
      tt4=-.5*b4
      tt5=-1./3.*b6
      tt6=b2/2.
      a1=tt1*bb4-bb1*tt4
```

```
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      if(a11.gt.flxck1) a1=a12
      if(a12.gt.flxck1) a1=a11
      if(a11.lt.0.0) a1=a12
      if(a12.lt.0.0) a1=a11
      deta2=sqrt(cc5*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxck2=1./b4-a1/b2
      if(a21.lt.-a1/(2.*b2)) a2=a22
      if(a22.lt.-a1/(2.*b2)) a2=a21
      if(a21.gt.flxck2) a2=a22
      if(a22.gt.flxck2) a2=a21
      if(abs((test1-a1)/a1).gt..0000001) go to 1000
      if(abs((test2-a2)/a2).gt..0000001) go to 1000
      if(a1.gt.flxck1) call exit(2)
      if(a1.lt.0.0) call exit(2)
      if(a2.gt.flxck2) call exit(2)
      if(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
```

```
1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
 detr=x2npa12*x2npa12-x2npa11*x2npa22
 if((x2npa22+x2npa11).lt.0.0) call exit(2)
 if((x2npa22+x2npa11).eq.0.0) call exit(2)
 if(detr.gt.0.0) call exit(2)
 if(detr.eq.0.0) call exit(2)
 return
 end


 function fbb1(r)
 common b,re,mxeval,rr,rp,a1,a2
 fbb1=besi1(r)*besk1(r)*(10./9.*r**7.+8./9.*r**5.)+besk0(r)
1*besi0(r)*(10./9.*r**7.+4./3.*r**5.)-besi1(r)*besk0(r)
1*(14./9.*r**6.+8./3.*r**4.)-4./9.*r**6.*besk1(r)*besi0(r)
 return
 end


 function fbb2(r)
 common b,re,mxeval,rr,rp,a1,a2
 fbb2=besk1(r)*besi1(r)*(178./225.*r**11.+416./225.*r**9.
1+128./25.*r**7.)+besk0(r)*besi0(r)*(178./225.*r**11.+184./75.
1*r**9.+320./25.*r**7.)-besk0(r)*besi1(r)*(314./225.
1*r**10.+608./75.*r**8.+640./25.*r**8.)-besk1(r)*besi0(r)*
1(136./225.*r**10.+64./25.*r**8.)
 return
 end


 function fbb3(r)
 common b,re,mxeval,rr,rp,a1,a2
 fbb3=besk1(r)*besi1(r)*(428./225.*r**9.+616./225.*r**7.
1+384./75.*r**5.)+besi0(r)*besk0(r)*(428./225.*r**9.+264./75.
1*r**7.+320./25.*r**5.)-besk0(r)*besi1(r)*(664./225.*r**8.+
12424./225.*r**6.+640./25.*r**4.)-besi0(r)*besk1(r)*(236./225.
1*r**8.+64./25.*r**6.)
 return
 end


 function fbb4(r)
 common b,re,mxeval,rr,rp,a1,a2
 fbb4=(28./9.*r**5.+8./9.*r**3.)*besi1(r)*besk1(r)
1+(28./9.*r**5.+4./3.*r**3.)*besk0(r)*besi0(r)-
1(32./9.*r**4.+8./3.*r**2.)*besi1(r)*besk0(r)
1-4./9.*r**4.*besk1(r)*besi0(r)
 return
 end


 function fbb5(r)
 common b,re,mxeval,rr,rp,a1,a2
 fbb5=besi1(r)*besk1(r)*(314./225.*r**7.+208./225.*r**5.+384.
1/150.*r**3.)+besk0(r)*besi0(r)*(314./225.*r**7.+92./75.
1*r**5.+32./5.*r**3.)-besk0(r)*besi1(r)*(382./225.*r**6.
1+304./75.*r**4.+64./5.*r**2.)-besi0(r)*besk1(r)*(68./225.
1*r**6.+32./25.*r**4.)
 return
 end


 function fbb6(r)
```

```
      common b,re,mxeval,rr,rp,a1,a2
      fbb6=r**3.*(besi1(r)*besk1(r)+besk0(r)*besi0(r))-r*r
     1*besk0(r)*besi1(r)
      return
      end


      function term1(x)
      common b,re,mxeval,rr,rp,a1,a2
      term1=besk1e(b*x)*besi0e(rr*x)*exp(-(b-rr)*x)/x
      return
      end


      function xtr1(y)
      common b,re,mxeval,rr,rp,a1,a2
      xtr1=besk0e(y*rp)*besi0e(y*rr)*exp(-(rp-rr)*y)
      return
      end


      function xtr2(y)
      common b,re,mxeval,rr,rp,a1,a2
      xtr2=besk0e(y*rr)*besi0e(y*rr)
      return
      end


      function xtr3(y)
      common b,re,mxeval,rr,rp,a1,a2
      xtr3=besk0e(y*rr)*besi0e(y*rp)*exp(-(rr-rp)*y)
      return
      end
```

```
      program tv4eig(kout,tty,tape5=kout,tape6=tty)
      common rn,b,re,mxeval,vp
      call xsetf(0)
      b=.5
      mxeval=950
      re=.000000001
      call flux
      write(6,200) b,vp
200   format(1x,3f20.10)
      call exit(2)
      end


      subroutine flux
      external expl
      common rn,b,re,mxeval,vp
      b2=b*b
      b3=b2*b
      b4=b3*b
      b5=b4*b
      b6=b5*b
      b7=b6*b
      b8=b7*b
      b9=b8*b
      tt1=.4*b5
      tt2=2./9.*b9
      tt3=4./7.*b7
      tt4=-4./3.*b3
      tt5=-.8*b5
      tt6=2.*b
      rn=3.
      e32b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=4.
      e42b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=5.
      e52b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=6.
      e62b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=7.
      e72b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=8.
      e82b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=9.
      e92b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=10.
      e102b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      rn=11.
      e112b=quad(expl,.0000000000001,1.,re,mxeval,kount)
      bb1=.8*b5-b4+8./9.*b3-4./3.+2.*b4*e32b+8.*b3*e42b
     1+16.*b2*e52b+16.*b*e62b+8.*e72b
      bb2=4./9.*b9-b8+16./7.*b7-4.*b6+96./25.*b5-115.2
     1+2.*b8*e32b+16.*b7*e42b+80.*b6*e52b+288.*b5*e62b+768.*b4*e72b
     1+1536.*b3*e82b+2304.*b2*e92b+2304.*b*e102b+1152.*e112b
      bb3=8./7.*b7-2.*b6+56./15.*b5-6.*b4+32./5.*b3-24.+4.*b6*e32b
     1+24.*b5*e42b+88.*b4*e52b+224.*b3*e62b+384.*b2*e72b+384.
     1*b*e82b+192.*e92b
      bb4=-8./3.*b3+2.*b2-8./3.*b+2.-4.*b2*e32b-8.*b*e42b-8.*e52b
      bb5=-8./5.*b5+2.*b4-16./3.*b3+12.*b2-96./5.*b+16.
     1-4.*b4*e32b-16.*b3*e42b-48.*b2*e52b-96.*b*e62b-96.*e72b
      bb6=4.*b-1.+2.*e32b
      c1=tt1*bb4-bb1*tt4
      c2=tt1*bb3-bb1*tt3
      c3=2.*tt1*bb6-2.*bb1*tt6
      c4=2.*tt1*bb5-2.*bb1*tt5
```

```
      c5=2.*tt1*bb2-2.*bb1*tt2
      c6=tt3*bb6+tt4*bb5-bb3*tt6-bb4*tt5
      c7=tt3*bb5+tt4*bb2-bb3*tt5-bb4*tt2
      c8=tt3*bb2-bb3*tt2
      c9=tt4*bb6-bb4*tt6
      c10=tt3*bb1-bb3*tt1
      c11=tt3*bb4+tt5*bb1-bb3*tt4-tt1*bb5
      c12=2.*tt2*bb1-2.*bb2*tt1
      c13=tt3*bb6+tt5*bb4-bb3*tt6-tt4*bb5
      c14=2.*tt2*bb4-2.*bb2*tt4
      c15=tt2*bb3-bb2*tt3
      c16=2.*tt2*bb6-2.*bb2*tt6
      c17=tt2*bb5-bb2*tt5
      c18=tt5*bb6-tt6*bb5
      a2=0.0
      a1=-500.
1000  continue
      test1=a1
      test2=a2
      cc1=c1+c2*a2
      cc2=c3+c4*a2+c5*a2*a2
      cc3=c9+c6*a2+c7*a2*a2+c8*a2*a2*a2
      cc4=c17+c15*a1
      cc5=c16+c14*a1+c12*a1*a1
      cc6=c18+c13*a1+c11*a1*a1+c10*a1*a1*a1
      deta1=sqrt(cc2*cc2-4.*cc1*cc3)
      a11=(deta1-cc2)/(2.*cc1)
      a12=(-deta1-cc2)/(2.*cc1)
      flxck1=1./b2-a2*b2
      If(a11.gt.flxck1) a1=a12
      If(a12.gt.flxck1) a1=a11
      If(a11.lt.0.0) a1=a12
      If(a12.lt.0.0) a1=a11
      deta2=sqrt(cc6*cc5-4.*cc4*cc6)
      a21=(deta2-cc5)/(2.*cc4)
      a22=(-deta2-cc5)/(2.*cc4)
      flxck2=1./b4-a1/b2
      If(a21.lt.-a1/(2.*b2)) a2=a22
      if(a22.lt.-a1/(2.*b2)) a2=a21
      If(a21.gt.flxck2) a2=a22
      If(a22.gt.flxck2) a2=a21
      If(abs((test1-a1)/a1).gt..0000001) go to 1000
      If(abs((test2-a2)/a2).gt..0000001) go to 1000
      If(a1.gt.flxck1) call exit(2)
      If(a1.lt.0.0) call exit(2)
      if(a2.gt.flxck2) call exit(2)
      If(a2.lt.-a1/(2.*b2)) call exit(2)
      vpt=tt1*a1*a1+tt2*a2*a2+tt3*a1*a2+tt4*a1+tt5*a2+tt6
      vpb=bb1*a1*a1+bb2*a2*a2+bb3*a1*a2+bb4*a1+bb5*a2+bb6
      vpb2=vpb*vpb
      vpb3=vpb2*vpb
      vp=2.*vpt/vpb
      x2npa11=2.*tt1/vpb-2.*(2.*tt1*a1+tt3*a2+tt4)*
     1(2.*bb1*a1+bb3*a2+bb4)/vpb2-2.*bb1*vpt/vpb2+2.*
     1(2.*bb1*a1+bb3*a2+bb4)*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt/vpb3
      x2npa12=tt3/vpb-(2.*tt1*a1+tt3*a2+tt4)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-bb3*vpt/vpb2-(2.*bb1*a1+bb3*a2+bb4)*(
     12.*tt2*a2+tt3*a1+tt5)/vpb2+2.*(2.*bb1*a1+bb3*a2+bb4)
     1*vpt*(2.*bb2*a2+bb3*a1+bb5)/vpb3
      x2npa22=2.*tt2/vpb-2.*(2.*tt2*a2+tt3*a1+tt5)*(2.*bb2*a2
     1+bb3*a1+bb5)/vpb2-2.*bb2*vpt/vpb2+2.*(2.*bb2*a2
     1+bb3*a1+bb5)*(2.*bb2*a2+bb3*a1+bb5)*vpt/vpb3
      detr=x2npa12*x2npa12-x2npa11*x2npa22
```

```
if((x2npa22+x2npa11).lt.0.0) call exit(2)
if((x2npa22+x2npa11).eq.0.0) call exit(2)
if(detr.gt.0.0) call exit(2)
if(detr.eq.0.0) call exit(2)
return
end


function expi(u)
common rn,b,re.mxeval.vp
expi=u**(rn-2.)*exp(-2.*b/u)
return
end
```

```
      program tv4lop(kout1,tty,tape5=kout1,tape6=tty)
      dimension xterm1(2001)
      dimension rxic(2001)
      dimension rxil(0:50,2001)
      dimension test(2001)
      dimension xic(2001)
      dimension xil(0:50,2001)
      call xsetf(0)
      llact=50
      jdim=1001
      b=11.9895
      sge=.8
      delrr=2.*b/float(jdim-1)
      kdim=(jdim+1)/2
      iuxl=kdim-1
      do 300 i=kdim,jdim
      if(i.eq.jdim) xterm1(i)=.25*(2.*b+1.-exp(-2.*b))
      if(i.eq.jdim) go to 200
      rr=float(i-1)*delrr-b
      xterm1(i)=.25*((b*b-rr*rr)*(e1(b+rr)-e1(b-rr))+(b+rr+1.)*
     1exp(-(b-rr))-(b-rr+1.)*exp(-(b+rr)))
200   continue
      rxic(1)=xterm1(1)
300   continue
400   continue
      do 500 i=1,iuxl
      rxic(i)=-rxic(jdim+1-i)
500   continue
      do 900 i=kdim,jdim
      aa=0.
      bb=0.
      iu=i-1
      do 600 j=1,iu
      delr=delrr
      if(j.eq.1) delr=.5*delrr
      rrx=float(i-j)*delrr
      aa=aa+rxic(j)*e1(rrx)*delr
600   continue
      ii=i+1
      if(ii.eq.jdim+1) go to 800
      do 700 j=ii,jdim
      delr=delrr
      if(j.eq.jdim) delr=.5*delrr
      rrx=float(j-i)*delrr
      bb=bb+rxic(j)*e1(rrx)*delr
700   continue
800   continue
      yy=1.0
      if(i.eq.jdim) yy=.5
      eps=.5*delrr
      st=sgs*rxic(i)*(1.-exp(-eps)+eps*e1(eps))*yy
      test(i)=(aa+bb)*sge*.5+st+xterm1(i)
900   continue
      icc=0
      rxic(kdim)=0.0
      kkdim=kdim+1
      do 1000 i=kkdim,jdim
      if(abs((test(i)-rxic(i))/rxic(i)).ge..00001) icc=1
      rxic(i)=test(i)
      rr=float(i-1)*delrr-b
      xic(i)=rxic(i)/rr
1000  continue
      if(icc.eq.1) go to 400
      xic(kdim)=0.0
```

```
1010  continue
      icc=0
      sum=0.0
      do 1020 i=kdim,jdim
      delr=delrr
      if(i.eq.kdim) delr=.5*delrr
      if(i.eq.jdim) delr=.5*delrr
      rp=float(i-1)*delrr-b
      sum=sum+delr*exp(-rp)*xic(i)
1020  continue
      test(kdim)=exp(-b)+egs*sum
      if(abe((test(kdim)-xic(kdim))/test(kdim)).ge..00001) icc=1
      xic(kdim)=test(kdim)
      if(icc.eq.1) go to 1010
      do 1100 i=kkdim,jdim
      rxii(0,i)=xterm1(i)
      rr=float(i-1)*delrr-b
      xii(0,i)=rxii(0.i)/rr
      xii(iisct,i)=xii(0,i)
1100  continue
      rxii(0.kdim)=0.0
      xii(0,kdim)=exp(-b)
      xii(iisct,kdim)=xii(0,kdim)
      io=0
1150  continue
      io=io+1
      if(io.eq.iisct) call exit(2)
      iol=io-1
      do 1200 i=1,iux1
      rxii(iol,i)=-rxii(iol,jdim+1-i)
1200  continue
      do 1600 i=kdim,jdim
      aa=0.
      bb=0.
      iu=i-1
      do 1300 j=1,iu
      delr=delrr
      if(j.eq.1) delr=.5*delrr
      rrx=float(i-j)*delrr
      aa=aa+rxii(iol,j)*e1(rrx)*delr
1300  continue
      ii=i+1
      if(ii.eq.jdim+1) go to 1500
      do 1400 j=ii,jdim
      delr=delrr
      if(j.eq.jdim) delr=.5*delrr
      rrx=float(j-i)*delrr
      bb=bb+rxii(iol,j)*e1(rrx)*delr
1400  continue
1500  continue
      yy=1.0
      if(i.eq.jdim) yy=.5
      eps=.5*delrr
      et=sgs*rxii(iol,i)*(1.-exp(-eps)+eps*e1(eps))*yy
      rxii(io,i)=(aa+bb)*sgs*.5+st
      rr=float(i-1)*delrr-b
      if(i.eq.kdim) go to 1550
      xii(io,i)=rxii(io,i)/rr
      xii(iisct,i)=xii(iisct,i)+xii(io,i)
1550  continue
1800  continue
      xii(io,kdim)=0.0
      do 1820 i=kdim,jdim
      delr=delrr
```

```
      if(i.eq.kdim) delr=.5*delrr
      if(i.eq.jdim) delr=.5*delrr
      rp=float(i-1)*delrr-b
      xil(io,kdim)=xil(io,kdim)+sgs*delr*exp(-rp)*xil(iol,i)
1620  continue
      xll(iisct,kdim)=xil(iisct,kdim)+xil(io,kdim)
      do 1700 i=kdim,jdim
      if(xil(iisct,i).gt.xic(i)) call exit(2)
      if(abe((xil(iisct,i)-xic(i))/xic(i)).gt..001) go to 1150
1700  continue
      write(5,1900) b,kdim,io
      write(5,1800) (xic(i),i=kdim,jdim)
      write(5,1800) (xll(iisct,i),i=kdim,jdim)
      do 1750 j=0,io
      write(5,1800) (xil(j,i),i=kdim,jdim)
1750  continue
1800  format(1x,3e20.10)
1900  format(1x,f20.10,2i5)
      call exit(2)
      end
```

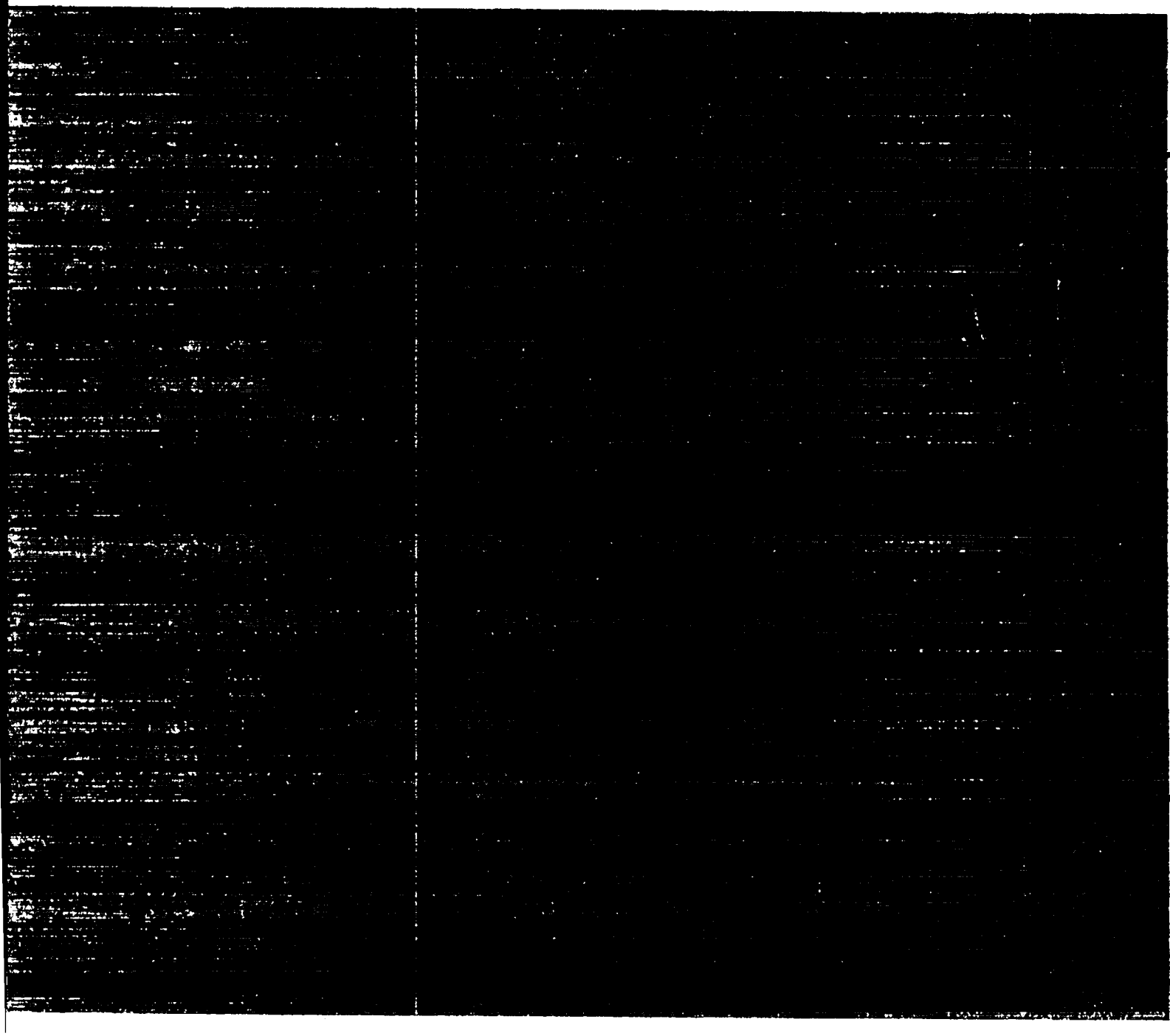
```
      program xplot(kout1,tty,output,tape5=kout1,tape6=tty,
     1tape7=output)
      dimension xic(1001)
      dimension sum(-1:50,1001)
      dimension xii(0:50,1001)
      dimension rpr(1001)
      dimension dilpj(1001)
      dimension dilps(1001)
      dimension ipkray(50)
      read(5,1100) b,kdim,isct
      read(5,1000) (xic(i),i=1,kdim)
      read(5,1000) (xii(50,i),i=1,kdim)
      do 100 j=0,isct
      read (5,1000) (xii(j,i),i=1,kdim)
  100 continue
      delrr=1./float(kdim-1)
      do 200 i=1,kdim
      rpr(i)=float(i-1)*delrr
  200 continue
      do 400 j=-1,isct
      do 300 i=1,kdim
      sum(j,i)=0.0
  300 continue
  400 continue
      do 600 j=0,isct
      do 500 i=1,kdim
      sum(j,i)=sum(j-1,i)+xii(j,i)
  500 continue
  600 continue
      call gplot(1hu,12hdiagplot,12)
      ni=ilnest(ipkray,50,40)
      call lines("total leakage probability$",ipkray,1)
      call lines("jth partial sum of leakage probability$",ipkray,2)
      call lines("jth term of leakage probability$",ipkray,3)
      do 800 j=0,isct
      do 700 i=1,kdim
      dilpj(i)=xii(j,i)
      dilps(i)=sum(j,i)
  700 continue
      call bgnpl(j)
      call physor(1.0,1.0)
      call page(12.0,15.0)
      call title(1hx,0,17hfractional radius,17,19hleakage probability
     1,19,10.0,11.0)
      xstp=.1
      ycycle=11./8.
      imark=(kdim-1)/10
      call ylog(0.0,xstp,.00000001,ycycle)
      call curve(rpr,xic,kdim,imark)
      call curve(rpr,dilps,kdim,imark)
      call curve(rpr,dilpj,kdim,imark)
      call legend(ipkray,3,1.0,11.0)
      call endpl(j)
  800 continue
 1000 format(1x,3e20.10)
 1100 format(1x,f20.10,2i5)
      call gdone
      call exit(2)
      end
```


☆ U.S. GOVERNMENT PRINTING OFFICE:1982-0-576-020/6

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Microfiche $4.50 (A01)

Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code
001-025	$ 5.00	A02	151-175	$11.00	A08	301-325	$17.00	A14	451-475	$23.00	A20
026-050	6.00	A03	176-200	12.00	A09	326-350	18.00	A15	476-500	24.00	A21
051-075	7.00	A04	201-225	13.00	A10	351-375	19.00	A16	501-525	25.00	A22
076-100	8.00	A05	226-250	14.00	A11	376-400	20.00	A17	526-550	26.00	A23
101-125	9.00	A06	251-275	15.00	A12	401-425	21.00	A18	551-575	27.00	A24
126-150	10.00	A07	276-300	16.00	A13	426-450	22.00	A19	576-600	28.00	A25
									601-up		A99

Add $1.00 for each additional 25-page increment or portion thereof from 601 pages up.



Los Alamos