LA-12888-T
Thesis

UC-000
Issued: February 1995

A Transport Based One-Dimensional Perturbation Code for Reactivity Calculations in Metal Systems

Tracy Reneé Wenz

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545

ACKNOWLEDGMENTS

I would like to take this opportunity to thank Tom McLaughlin and Doug O'Dell of the Nuclear Criticality Safety group (ESH-6) at Los Alamos National Laboratory for their support and guidance on this thesis work. I would particularly like to thank my thesis advisor, Robert Busch, for the support and guidance he provided through out this process and for reading all the earlier versions of this manuscript. I would also like to express my appreciation to my committee members, Norman Roderick; Anil Prinja; and Doug O'Dell, for their comments and suggestions in the final stages of this work. And my sincere thanks goes to Diane Gwinn and Karen Hayes for helping me with all the paper work that I always seemed to end up doing at the last minute.

DISCLAIMER

DISCLAIMER

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A TRANSPORT BASED ONE-DIMENSIONAL PERTURBATION CODE FOR REACTIVITY CALCULATIONS IN METAL SYSTEMS

by

Tracy Renee' Wenz

ABSTRACT

A one-dimensional reactivity calculation code is developed using first order perturbation theory. The reactivity equation is based on the multi-group transport equation using the discrete ordinates method for angular dependence. In addition to the first order perturbation approximations, the reactivity code uses only the isotropic scattering data, but cross section libraries with higher order scattering data can still be used with this code. The reactivity code obtains all the flux, cross section, and geometry data from the standard interface files created by ONEDANT, a discrete ordinates transport code.

Comparisons between calculated and experimental reactivities were done with the central reactivity worth data for Lady Godiva, a bare uranium metal assembly. Good agreement is found for isotopes that do not violate the assumptions in the first order approximation. In general for cases where there are large discrepancies, the discretized cross section data is not accurately representing certain resonance regions that coincide with dominant flux groups in the Godiva assembly. Comparing reactivities calculated with first order perturbation theory and a straight $\Delta k/k$ calculation shows agreement within 10% indicating the perturbation of the calculated fluxes is small enough for first order perturbation theory to be applicable in the modeled system. Computation time comparisons between reactivities calculated with first order perturbation theory and straight $\Delta k/k$ calculations indicate considerable time can be saved performing a calculation with a perturbation code particularly as the complexity of the modeled problems increase.

CHAPTER 1

INTRODUCTION

Perturbation theory, when applied to nuclear reactor theory, provides a useful means to calculate the change in the multiplication factor, or reactivity, of a system. This system change can be in the form of a variation in geometry, material (e.g. addition, removal, or replacement), temperature (e.g. Doppler broadening), or other parameters affecting the neutron population of the system. Depending upon the order of the approximations used in deriving the reactivity equations from perturbation theory (Bell and Glasstone, 1970; Carmichael, 1970; Hansen and Maier, 1960; McDaniel, 1993; Weinberg and Wigner, 1958), small or large system changes can be addressed. In this thesis, the system changes will be limited to small perturbations which are described by first order perturbation theory.

First order perturbation theory was more widely used in the past when computers were much slower at performing calculations and the availability of machines to perform high speed computations was limited or nonexistent (Hansen and Maier, 1960; Ussachoff, 1955; Weinberg and Wigner, 1958). By placing the constraint that the perturbation was small so that the perturbed flux did not differ much from the unperturbed flux over the entire system, the reactivity change could be calculated using only the unperturbed forward and adjoint neutron fluxes. Using this method, many reactivity calculations could be performed based on only two transport code calculations for a given system configuration. This reduction in the number of calculations and the amount of total computation time was a major advantage of perturbation theory.

However, even with the advent of high speed serial computers (e.g. CRAY YMP computers and SUN SPARC workstations) and the development of more efficient transport algorithms (ONEDANT (O'Dell et al., 1989) and TWODANT (Alcouffe et al., 1992)), perturbation codes continue to be developed and used (Ahn et al., 1993; Carmichael, 1970; Dean, 1988; George and LaBauve, 1988; Hopkins, 1971). One reason for this continued

interest is that there are still time and resource savings (Hopkins, 1971) obtainable with these codes, particularly as the complexity of the modeled problems increases. In general, adjoint calculations which form the basis of perturbation theory are particularly well suited for parametric studies or any analysis (Ahn et al., 1993) where there is the potential for performing many repetitive calculations. Examples of these calculations include, but are not limited to, reactivity calculations using first order perturbation theory where only the forward and adjoint unperturbed flux need be calculated and radiation dose calculations with a fixed source where a single adjoint calculation results in the detector response at various locations in the volume (Bell and Glasstone, 1970).

Existing computer codes for calculating reactivity worths using perturbation theory are reported sparingly in literature. One of the earlier reported uses of perturbation theory was by Hansen and Maier (1953). Here a first order perturbation equation for reactivity worth calculations was derived based on the Boltzman transport equation using a P_3 transport approximation (spherical harmonics expansion) and three neutron energy groups. This work was later reported (Hansen and Maier, 1960) using an S_n code and 16 energy group cross sections.

Perturbation codes for more general use as production codes were also developed. Hardie and Little (1969) developed a reactivity code based on the two-dimensional multigroup diffusion equation. First order perturbation codes based on the Boltzman form of the transport equation for one (Carmichael, 1970) and two (Hopkins, 1971) dimensions were also developed at Los Alamos.

More recent developments in first order perturbation theory for reactivity calculations include the code by Dean (1988) which is based on the Boltzman transport equation discretized using spherical harmonics. The diffusion theory based reactivity code by Hardie and Little (1969) was upgraded by George and LaBauve (1988) to accept standard interface files (O'Dell, 1977) created by more current transport codes like TWODANT (Alcouffe et al., 1992).

In this work, a first order perturbation code based on the Boltzman transport equation using the discrete ordinates method is developed. This project was undertaken because such a code does not appear to be available that will readily accept flux data using the standard interface file format (O'Dell, 1977). Flux data obtained from ONEDANT (O'Dell et al., 1989) are used to calculate the reactivity worths for uranium metal systems. ONEDANT solves the one-dimensional steady state Boltzman equation using a multigroup energy structure and the method of discrete ordinates to handle the angular dependence. The binary data files generated by ONEDANT use the standard interface file format.

The remaining chapters of this thesis discuss the development and use of this perturbation code utilizing ONEDANT binary files. Chapter 2 discusses the derivation of the reactivity equations using first order perturbation theory. Chapters 3 and 4 discuss the reactivity computer code and present results calculated using the code, and Chapter 5 provides the conclusions from this work.

CHAPTER 2

DEVELOPMENT OF PERTURBATION EQUATIONS

This chapter discusses the derivation of the reactivity equation using first order perturbation theory. Specifically, the forward and adjoint transport equations are discussed along with the method used for calculating the adjoint operators. Next, the reactivity equation is derived in detail and compared with other formulations found in literature.

2.1 Neutron Transport Equation

The linear time-independent Boltzman transport equation is the starting point for the derivation of the first order perturbation equations. The general form of the Boltzman equation (Duderstadt and Hamilton, 1976; O'Dell et al., 1989) is presented in Equation 2.1 where Ψ is the angular flux.

$$\Omega \bullet \nabla \Psi(\mathbf{r},E,\Omega) + \Sigma_t(\mathbf{r},E)\Psi(\mathbf{r},E,\Omega) = \iint \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\Psi(\mathbf{r},E',\Omega')dE'd\Omega' + \frac{\chi(E)}{4\pi}\iint \nu(E')\Sigma_f(\mathbf{r},E')\Psi(\mathbf{r},E',\Omega')dE'd\Omega' + Q(\mathbf{r},E,\Omega) \qquad (2.1)$$

The first two terms on the left hand side represent loss mechanisms for neutrons. The divergence term is the neutron loss due to leakage at the surface of the geometry, and the total macroscopic cross section term determines the neutron loss from all neutron interactions inside the geometry. The right hand side of Equation 2.1 contains the terms that contribute neutrons to the system. The first integral term represents neutrons that are scattered into a different energy group; thus the scattered neutrons act as a source for the new energy group E. The second integral term accounts for neutrons born into energy group E from fission assuming an isotropic distribution. Finally, the last term, $Q(\mathbf{r},E,\Omega)$,

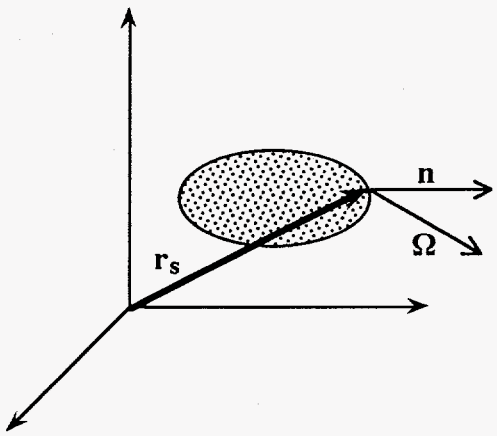

Figure 1. Location of the neutron direction vector, **Ω**, and the surface unit vector, **n**, relative to a surface where the boundary conditions are defined.

includes the effect of neutrons present from external sources. External sources are neutron sources that are independent of the neutron flux, Ψ.

The boundary condition used for this equation is that for a nonreentrant surface. This condition implies that once a neutron leaks from the geometry it will not be scattered back into the geometry (i.e. a vacuum boundary condition). The nonreentrant boundary condition (Duderstadt and Hamilton, 1976) is typically written as

$$\Psi(\mathbf{r}_s,E,\Omega) = 0, \quad \text{if } \Omega\bullet\mathbf{n}_s < 0 \tag{2.2}$$

As shown in Figure 1, $\mathbf{r}_s$ is some location on the surface of the geometry, $\mathbf{n}_s$ is a unit vector on the surface of the geometry pointing away from the surface, and Ω is a unit vector representing the direction the neutrons are traveling. If $\Omega\bullet\mathbf{n}_s$ is less than zero, neutrons are entering the surface; similarly, if $\Omega\bullet\mathbf{n}_s$ is greater than zero, neutrons are exiting the surface.

We will also assume that there are no external neutron sources. The implication of this assumption is we are dealing with a critical system. As a result, the external source term in Equation 2.1 is set equal to zero. Equation 2.3 is the general form of the transport equation with its associated boundary conditions that will be used to derive the excess reactivity equation using first order perturbation theory.

$$\Omega\bullet\nabla\Psi(\mathbf{r},E,\Omega) + \Sigma_t(\mathbf{r},E)\Psi(\mathbf{r},E,\Omega) =$$
$$\iint \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\Psi(\mathbf{r},E',\Omega')dE'd\Omega' +$$
$$\frac{\chi(E)}{4\pi}\frac{1}{k}\iint \nu\Sigma_f(\mathbf{r},E')\Psi(\mathbf{r},E',\Omega')dE'd\Omega'$$

$$\text{with } \Psi(\mathbf{r}_s,E,\Omega) = 0 \quad \text{if } \Omega\bullet\mathbf{n}_s < 0 \tag{2.3}$$

Note that Equation 2.3 has an additional factor of 1/k in front of the fission term. This factor is the eigenvalue for the equation and enables a nontrivial solution to be found (McDaniel, 1993). The inverse of the eigenvalue, k, is the effective multiplication factor for the system and indicates whether the system is critical (k=1), subcritical (k<1), or supercritical (k>1).

The general procedure for deriving the reactivity equation (Bell and Glasstone, 1970; Carmichael, 1970; Duderstadt and Hamilton, 1976; Hansen and Maier, 1960; Hopkins, 1971; Weinberg and Wigner, 1958) is to use the forward transport equation (Equation 2.3) with perturbed quantities substituted in for the angular flux, cross sections, and eigenvalue. Then multiply this equation by the adjoint flux. Next, the adjoint transport equation is multiplied by the perturbed forward flux and subtracted from the previous equation. At this point, certain terms may be neglected depending on the order of the approximation used in the calculation. The equation is then integrated over all phase space (position, dr; energy, dE; and direction, $d\Omega$). Finally, the equation is solved for the change in the eigenvalue and thus the reactivity. This procedure is detailed in the remainder of this chapter.

The first step in obtaining the reactivity equation is to deal with the energy and angular direction integrations on the scattering and fission terms in Equation 2.3. To remove the energy dependence, the multigroup approximation (Duderstadt and Hamilton, 1976; McDaniel, 1993; O'Dell et al., 1989; O'Dell and Alcouffe, 1987) is used. This approximation divides the continuous energy range of the neutrons into a finite number, NOG (number of groups), of energy groups. Integrating Equation 2.3 over each of the finite energy groups (see Equation 2.4) results in a system of NOG coupled equations as illustrated in Equation 2.6a.

$$\int_{\Delta E_g} \Omega \bullet \nabla \Psi(\mathbf{r},E,\Omega)\, dE + \int_{\Delta E_g} \Sigma_t(\mathbf{r},E)\Psi(\mathbf{r},E,\Omega)\, dE =$$

$$\int_{\Delta E_g} \iint \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\Psi(\mathbf{r},E',\Omega')dE'd\Omega'\, dE +$$

$$\int_{\Delta E_g} \frac{\chi(E)}{4\pi} \frac{1}{k} \iint \nu\Sigma_f(\mathbf{r},E')\Psi(\mathbf{r},E',\Omega')dE'd\Omega'\, dE \qquad (2.4)$$

By definition, the angular neutron flux for energy group g is:

$$\Psi_g(\mathbf{r},\Omega) = \int_{\Delta E_g} \Psi(\mathbf{r},E,\Omega)\, dE \qquad (2.5)$$

Upon substituting Equation 2.5 into Equation 2.4, the multigroup neutron transport equation becomes:

$$\Omega \bullet \nabla \Psi_g(\mathbf{r},\Omega) + \Sigma_t^g(\mathbf{r})\Psi_g(\mathbf{r},\Omega) =$$

$$\int \sum_{g'}^{NOG} \Sigma_s^{g'\rightarrow g}(\mathbf{r},\Omega'\rightarrow\Omega)\Psi_{g'}(\mathbf{r},\Omega')\, d\Omega' +$$

$$\frac{\chi^g}{4\pi}\frac{1}{k}\int \sum_{g'}^{NOG} \nu\Sigma_f^{g'}(\mathbf{r})\Psi_{g'}(\mathbf{r},\Omega')\, d\Omega' \qquad g = 1, \ldots, NOG \qquad (2.6a)$$

The group averaged parameters presented in Equation 2.6a for the total, scattering, and fission cross sections and the χ distribution are defined as follows:

$$\Sigma_t^g(\mathbf{r}) = \frac{\int_{\Delta E_g} \Sigma_t(\mathbf{r},E)\Psi(\mathbf{r},E,\Omega)\, dE}{\int_{\Delta E_g} \Psi(\mathbf{r},E,\Omega)\, dE} \qquad (2.6b)$$

$$\Sigma_s^{g'\to g}(\mathbf{r},\Omega'\to\Omega) = \frac{\int_{\Delta E_g} dE \int_{\Delta E_{g'}} \Sigma_s(\mathbf{r},E'\to E,\Omega'\to\Omega)\Psi(\mathbf{r},E',\Omega')\, dE'}{\int_{\Delta E_{g'}} \Psi(\mathbf{r},E',\Omega')\, dE} \tag{2.6c}$$

$$\nu\Sigma_f^{g'}(\mathbf{r}) = \frac{\int_{\Delta E_{g'}} \nu\Sigma_f(\mathbf{r},E')\Psi(\mathbf{r},E',\Omega')\, dE'}{\int_{\Delta E_{g'}} \Psi(\mathbf{r},E',\Omega')\, dE} \tag{2.6d}$$

$$\chi^g = \int_{\Delta E_g} \chi(E)\, dE \tag{2.6e}$$

The angular integrals in Equation 2.6a are addressed by first looking at the factors multiplying the angular fluxes in these two terms. Inherent in the fission term is the assumption that the fission process is isotropic, thus the factor of $1/4\pi$, so the factors multiplying the angular flux in the fission term can be pulled outside the integral.

Second, only isotropic scattering is being considered in this derivation. This approximation is acceptable because the cross section library (Bell et al., 1963; Hansen and Roach, 1969) used in the reactivity calculations contains only isotropic (P_0) scattering data and the total cross sections have had transport corrections applied to compensate for the anisotropic component in the scattering. The spherical harmonics expansion of the angular scattering term (O'Dell et al., 1989) is given by:

$$\sigma_s^{g'\to g}(\mathbf{r},\Omega'\bullet\Omega) = \sum_{L=0}^{ISCT} \left(\frac{2L+1}{4\pi}\right) \sigma_{sL}^{g'\to g}(\mathbf{r})\, P_L(\Omega'\bullet\Omega) \tag{2.7a}$$

For the case where only isotropic scattering (L = 0 and P_0 = 1) occurs, Equation 2.7a reduces to the angular independent form:

$$\sigma_s^{g'\to g}(\mathbf{r},\Omega'\bullet\Omega) = \frac{\sigma_{s0}^{g'\to g}(\mathbf{r})}{4\pi} \tag{2.7b}$$

Substituting Equation 2.7b into Equation 2.6a results in two angular independent factors that multiply the angular flux in the scattering term; these terms can also be pulled outside of the integral.

In general, the integral over angle in Equation 2.6a can be calculated numerically (see Appendix A for details) as:

$$\int \Psi_g(\mathbf{r},\Omega)\, d\Omega \approx \sum_{m=1}^{MM} \Delta A\, \Psi_{g.m}(\mathbf{r}) \tag{2.8}$$

The transport code (Alcouffe et al., 1992; O'Dell et al., 1989) used in this work to calculate the unperturbed fluxes utilizes the method of discrete ordinates to handle the angular variables (Ω), so this method is used to handle the numerical integrations over angle in Equation 2.6a. With the discrete ordinates method, the continuum in which a neutron can travel is divided into MM discrete directions (Ω_m); the resulting discretized angular flux (O'Dell and Alcouffe, 1987) is defined as:

$$\Psi_{g.m}(\mathbf{r}) \equiv \frac{1}{w_m} \int_{\Omega_m} \Psi_g(\mathbf{r},\Omega)\, d\Omega \tag{2.9}$$

where the quadrature weight, w_m, represents the discrete area on a unit sphere corresponding to the direction Ω_m which is normalized by a factor of $1/4\pi$ such that:

$$\sum_{m=1}^{MM} w_m = 1 \tag{2.10}$$

From this method, the scalar flux for group g, $\phi_g(\mathbf{r})$, is defined as:

$$\phi_g(\mathbf{r}) \equiv \sum_{m=1}^{MM} w_m \Psi_{g.m}(\mathbf{r}) \approx \int \Psi_g(\mathbf{r},\Omega)\, d\Omega \tag{2.11}$$

and approximately equals the integral of the flux over all angles. In the limit of an infinitely small surface area, ΔA, the integral is exact. Thus by adding a factor of $1/4\pi$ and being selective on the choice for the finite areas (O'Dell and Alcouffe, 1987), the general numerical integration approximation is converted to a discrete ordinates approximation.

Substituting the integral approximation from Equations 2.8 and 2.11 in the two integral terms of Equation 2.6a results in:

$$\Omega \bullet \nabla \Psi_g(\mathbf{r},\Omega) + \Sigma_t^g(\mathbf{r}) \Psi_g(\mathbf{r},\Omega) =$$

$$\sum_{g'}^{NOG} \Sigma_s^{g' \to g}(\mathbf{r})\, \phi_{g'}(\mathbf{r}) + \chi^g \frac{1}{k} \sum_{g'}^{NOG} \nu \Sigma_f^{g'}(\mathbf{r})\, \phi_{g'}(\mathbf{r})$$

$$g = 1, \ldots, NOG \tag{2.12}$$

Note that the two $1/4\pi$ terms in the fission and scattering terms are no longer present in Equation 2.12. These factors were absorbed into the numerical integration approximation for the discrete ordinates method described in Equations 2.9 and 2.10.

2.2 Adjoint Transport Equation

As stated earlier, the adjoint transport equation will be used in the derivation of the reactivity equation; henceforth, the theory of adjoints and their implications in neutron transport theory is discussed.

From the theory of linear algebra (Friedman, 1956), every bounded linear operator L has a corresponding adjoint operator L^*. The adjoint operator can be found by calculating the inner (or scalar) product of two functions. This inner product is defined as:

$$(\Phi, \varphi) = \int \Phi(p)\varphi(p)dp \tag{2.13}$$

where dp represents all phase space. The adjoint operator must satisfy the following condition.

$$(\Phi, L\varphi) = (\varphi, L^*\Phi) \tag{2.14}$$

In the above equation, Φ is referred to as the adjoint flux that receives the adjoint operation, and φ is the forward flux operated on by the linear operator L.

Specific operators that occur in the neutron transport equation in Equation 2.3 include:

$$\Omega\bullet\nabla,\ \Sigma_t(\mathbf{r},E)\bullet,\ 1/k\bullet, \int\int \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)dE'd\Omega',$$

$$\text{and } \chi(E)\int\int \nu(E')\Sigma_f(\mathbf{r},E')dE'd\Omega'.$$

First, the adjoints of operators that do not result in the production of neutrons (i.e. fission) or in energy or angular transfer (i.e. scattering) will be calculated. Such operators include some sort of removal process, such as σ_t or σ_r, and system wide constants.

Constant operators

For the operator $L\varphi = \Sigma_t(\mathbf{r},E)\varphi(\mathbf{r},E,\Omega)$, its adjoint is found as follows:

$$\begin{aligned}(\Phi, L\varphi) &= \int\int\int \Phi(\mathbf{r},E,\Omega)\{\Sigma_t(\mathbf{r},E)\varphi(\mathbf{r},E,\Omega)\}d\mathbf{r}\ dE\ d\Omega \\ &= \int\int\int \varphi(\mathbf{r},E,\Omega)\{\Sigma_t(\mathbf{r},E)\Phi(\mathbf{r},E,\Omega)\}d\mathbf{r}\ dE\ d\Omega = (\varphi, L^*\Phi)\end{aligned} \tag{2.15}$$

Since L operates on φ as a scalar multiplier, the commutative (AB = BA) and associative (A[BC] = [AB]C) laws of scalar multiplication are used to rearrange L so that it operates on Φ. In this case, $L = \Sigma_t(\mathbf{r},E)\bullet = L^*$, so this operator is self-adjoint. In general, all constant operators are self-adjoint. This being the case, the term 1/k is also self-adjoint since it is a system wide constant.

Production operators

This section discusses production operators which include those that produce neutrons by either creating them (fission) or redistributing them (scattering). These operators also fall into the category of constant operators because they act on the flux as scalar multipliers. An additional conversion step is required so that the variable notation on the adjoint flux conforms to the notation on the forward flux (Henry, 1975; Lewins, 1965).

The calculation of the adjoint scattering operator:

$$L\varphi = \int\int \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\varphi(\mathbf{r},E',\Omega')\,dE'd\Omega' \qquad (2.16)$$

is shown in Equation 2.17 (Henry, 1975; Lewins, 1965). A change in variables and in the order of integration is performed in the last two lines, respectively, of this equation so that the adjoint flux is a function of the same variables as the forward flux. With changing variables in an integral, there is also a corresponding change in the limits of integration to reflect the new variables (Purcell and Varberg, 1984). This issue will not be addressed here because the information needed to determine the adjoint operator is the final form of the integrand after the forward operator is moved from the forward flux to the adjoint flux.

$$
\begin{aligned}
(\Phi, L\varphi) &= \int d\mathbf{r} \int d\Omega \int dE\ \Phi(\mathbf{r},E,\Omega) \int d\Omega' \int dE'\ \Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\varphi(\mathbf{r},E',\Omega') \\
&= \int d\mathbf{r} \int d\Omega \int dE \int d\Omega' \int dE'\ \varphi(\mathbf{r},E',\Omega')\{\Sigma_s(\mathbf{r},E'\rightarrow E,\Omega'\rightarrow\Omega)\Phi(\mathbf{r},E,\Omega)\} \\
&= \int d\mathbf{r} \int d\Omega' \int dE' \int d\Omega \int dE\ \varphi(\mathbf{r},E,\Omega)\{\Sigma_s(\mathbf{r},E\rightarrow E',\Omega\rightarrow\Omega')\Phi(\mathbf{r},E',\Omega')\} \\
&= \int d\mathbf{r} \int d\Omega \int dE\ \varphi(\mathbf{r},E,\Omega) \int d\Omega' \int dE'\ \Sigma_s(\mathbf{r},E\rightarrow E',\Omega\rightarrow\Omega')\Phi(\mathbf{r},E',\Omega')\} \\
&= (\varphi, L^*\Phi) \qquad (2.17)
\end{aligned}
$$

where

$$L^*\Phi = \int\int \Sigma_s(\mathbf{r},E\rightarrow E',\Omega\rightarrow\Omega')\Phi(\mathbf{r},E',\Omega')\ dE'd\Omega'$$

The adjoint of the fission operator, $L\varphi = \chi(E) \int\int \nu(E')\Sigma_f(\mathbf{r},E')\varphi(\mathbf{r},E',\Omega')dE'd\Omega'$, is (Henry, 1975; Lewins, 1965):

$$
\begin{aligned}
(\Phi, L\varphi) &= \int d\mathbf{r} \int d\Omega \int dE\ \Phi(\mathbf{r},E,\Omega)\ \chi(E) \int d\Omega' \int dE'\ \nu(E')\Sigma_f(\mathbf{r},E')\varphi(\mathbf{r},E',\Omega') \\
&= \int d\mathbf{r} \int d\Omega \int dE \int d\Omega' \int dE'\ \varphi(\mathbf{r},E',\Omega')\{\chi(E)\nu(E')\Sigma_f(\mathbf{r},E')\Phi(\mathbf{r},E,\Omega)\} \\
&= \int d\mathbf{r} \int d\Omega' \int dE' \int d\Omega \int dE\ \varphi(\mathbf{r},E,\Omega)\{\chi(E')\nu(E)\Sigma_f(\mathbf{r},E)\Phi(\mathbf{r},E',\Omega')\} \\
&= \int d\mathbf{r} \int d\Omega \int dE\ \varphi(\mathbf{r},E,\Omega)\ \nu(E)\Sigma_f(\mathbf{r},E) \int d\Omega' \int dE'\ \chi(E')\Phi(\mathbf{r},E',\Omega') \\
&= (\varphi, L^*\Phi) \qquad (2.18)
\end{aligned}
$$

where

$$L^*\Phi = \nu(E)\Sigma_f(\mathbf{r},E) \int\int \chi(E')\Phi(\mathbf{r},E',\Omega')dE'd\Omega'$$

The adjoint operators for the production terms fall out more directly when the general perturbed reactivity equation is defined and an arbitrary weighting function is introduced into the equation. This weighting function, which turns out to be the adjoint function, is

set so that certain higher order terms drop out when a first order approximation is assumed (Henry, 1975; McDaniel, 1993).

Leakage operators

The adjoint for the leakage operator, $L = \Omega \bullet \nabla$, can be calculated without specifying a specific geometry or number of dimensions; consequently, the generality of the derivation is maintained. Several mathematical tools will be useful in performing this derivation. The first is the use of Gauss' Divergence Theorem (Bell and Glasstone, 1970; Purcell and Varberg, 1984; Tuma, 1987) in Equation 2.19. This theorem converts a volume integral into a surface integral and will be useful when defining the adjoint boundary condition.

$$\int_V d^3r \, \nabla \bullet \mathbf{A} = \int_S dS \, \mathbf{n} \bullet \mathbf{A} \tag{2.19}$$

Here, $\mathbf{n}$ is a unit vector normal to the surface of S, pointing outward from S (similar to $\mathbf{n}_S$ in Figure 1). Last, variations of the following vector identity (Tuma, 1987) will come in handy where α is a scalar quantity and $\mathbf{A}$ is a vector quantity:

$$\nabla \bullet (\alpha \mathbf{A}) = \alpha \nabla \bullet \mathbf{A} + \mathbf{A} \bullet \nabla \alpha \tag{2.20}$$

The steps for determining the adjoint of the leakage operator are detailed below:

$$(\Phi, L\varphi) = \int \int \int \Phi(\mathbf{r},E,\Omega)\{\Omega \bullet \nabla \varphi(\mathbf{r},E,\Omega)\} d\mathbf{r} \, dE \, d\Omega \tag{2.21a}$$

The omega vector can be put in the argument of the ∇ operator because Ω is not a function of the variables on which ∇ operates (i.e. $\nabla \bullet (\Omega\varphi) = \Omega \bullet \nabla\varphi + \varphi \nabla \bullet \Omega = \Omega \bullet \nabla\varphi$ since the derivative of a constant, Ω, is zero).

$$(\Phi, L\varphi) = \int \int \int \Phi(\mathbf{r},E,\Omega) \nabla \bullet (\Omega \varphi(\mathbf{r},E,\Omega)) d\mathbf{r} \, dE \, d\Omega \tag{2.21b}$$

Using the identity in Equation 2.20 again, the function $\Phi(\mathbf{r},E,\Omega)$ can be placed in the argument of the ∇ operator with the functional arguments dropped to simplify the notation:

$$(\Phi, L\varphi) = \int\int\int \nabla\bullet(\Phi\Omega\varphi)dr\ dE\ d\Omega - \int\int\int \varphi\Omega\bullet\nabla(\Phi)dr\ dE\ d\Omega \qquad (2.21c)$$

In order to obtain the adjoint operator, the condition in Equation 2.14 must be satisfied. For this to occur, the first term in the above equation must go to zero; the second term is the adjoint operator. By applying Gauss' Divergence Theorem, the volume integral in the first integral of Equation 2.21c is converted into a surface integral as shown in Equation 2.21d. Applying the forward boundary condition and defining an appropriate adjoint boundary condition will result in this term having a value of zero.

$$(\Phi, L\varphi) = \iiint \mathbf{n}\bullet\Omega\Phi\varphi\ dSdEd\Omega - \iiint \varphi\Omega\bullet\nabla(\Phi)dr\ dE\ d\Omega \qquad (2.21d)$$

The first integral is split into two components so that the directions in which the boundary conditions are defined are explicitly implied in the equation:

$$(\Phi, L\varphi) = \iint_{\mathbf{n}\bullet\Omega<0}\int \mathbf{n}\bullet\Omega\Phi\varphi\ dSdEd\Omega + \iint_{\mathbf{n}\bullet\Omega>0}\int \mathbf{n}\bullet\Omega\Phi\varphi\ dSdEd\Omega - \iiint \varphi\Omega\bullet\nabla(\Phi)dr\ dE\ d\Omega \qquad (2.21e)$$

The first integral equals zero because the forward boundary condition states that $\varphi(\mathbf{r}_S,E,\Omega) = 0$ for $\mathbf{n}\bullet\Omega < 0$ and the integral is evaluated over the surface which the boundary condition is defined. In order for the second integral to have a value of zero, the adjoint boundary condition can be defined such that $\Phi(\mathbf{r}_S,E,\Omega) = 0$ for $\mathbf{n}\bullet\Omega > 0$. Applying both sets of boundary conditions removes the extra term in Equation 2.21d so that the condition in Equation 2.14 is satisfied which results in the definition of the adjoint leakage operator, $L^* = -\Omega\bullet\nabla$. The forward and adjoint leakage operators differ by only a minus sign for all geometries and dimensions.

Upon substituting all the adjoint operators derived above into the adjoint equation, results in the final form of the adjoint equation that will be used in the next section along with its corresponding adjoint boundary conditions.

$$-\Omega \bullet \nabla \Psi_g^*(\mathbf{r},\Omega) + \Sigma_t^g(\mathbf{r}) \Psi_g^*(\mathbf{r},\Omega) =$$

$$\sum_{g'}^{NOG} \Sigma_s^{g \to g'}(\mathbf{r})\, \phi_{g'}^*(\mathbf{r}) + \chi^{g'} \frac{1}{k} \sum_{g'}^{NOG} \nu \Sigma_f^g(\mathbf{r})\, \phi_{g'}^*(\mathbf{r})$$

$$\text{where } \Psi_g^*(\mathbf{r}_s,\Omega) = 0 \quad \text{if } \Omega \bullet \mathbf{n}_s > 0 \quad \text{and} \quad g = 1, \ldots, NOG \qquad (2.22)$$

In Equation 2.22, the terms Ψ_g^* and ϕ_g^* represent the adjoint angular and adjoint scalar fluxes. In reactor theory, these also signify the neutron importance (Bell and Glasstone, 1970) which indicates how a given neutron will affect future generations of neutrons. For instance, if a region of a core has a high neutron importance relative to another area, then a neutron in that area will be more likely to increase the neutron population (for instance by causing additional fissions) than one in the other area. One would expect the neutron importance to be low at the surface of an unreflected reactor because it is most likely a neutron at the surface will escape the reactor and not cause additional fissions. On the other hand, a thermal neutron at the center of the core would be expected to have a high neutron importance because (1) it is very unlikely it will leak from the reactor and (2) for the case of a thermal reactor, the fission probability is much higher at thermal energies. Neutron importance functions can be measured and calculated for non-fissioning materials as well. One possible use for this data would be for determining the best location to place a control rod in a thermal reactor.

2.3 Perturbation Equation

With the forward and adjoint transport equations defined, we are ready to introduce a perturbation to the critical system. This system change can be in the form of a geometry effect (i.e. $\Omega\bullet\nabla$) or a material replacement of some sort (i.e. Σ_s, $\nu\Sigma_f$, Σ_t, or χ). The perturbation will also affect the neutron fluxes and the eigenvalue of the critical system. These changes can be represented as follows:

$$\begin{aligned}
\Sigma^{\bullet}_{t.g} &= \Sigma^{g}_{t} + \delta\Sigma^{g}_{t} \\
\Sigma^{\bullet}_{s.g'\to g} &= \Sigma^{g'\to g}_{s} + \delta\Sigma^{g'\to g}_{s} \\
\nu\Sigma^{\bullet}_{f.g} &= \nu\Sigma^{g}_{f} + \delta\nu\Sigma^{g}_{f} \\
\chi^{\bullet}_{g} &= \chi^{g} + \delta\chi^{g} \\
\frac{1}{k^{\bullet}} &= \frac{1}{k+\delta k} \\
\Psi^{\bullet}_{g} &= \Psi_{g} + \delta\Psi_{g} \\
\phi^{\bullet}_{g} &= \phi_{g} + \delta\phi_{g}
\end{aligned} \tag{2.23}$$

where the dotted terms on the left hand side of Equation 2.23 (note the slight notation change) represent the perturbed values and the δ terms on the right hand side represent the amount a particular parameter is changed due to a change in the critical system parameter. Substituting the quantities in Equation 2.23 into Equation 2.12 results in the forward perturbed transport equation as shown below.

$$\Omega\bullet\nabla\Psi^{\bullet}_{g}(\mathbf{r},\Omega) + \Sigma^{\bullet}_{t.g}(\mathbf{r})\Psi^{\bullet}_{g}(\mathbf{r},\Omega) =$$

$$\sum_{g'}^{NOG} \Sigma^{\bullet}_{s.g'\to g}(\mathbf{r})\,\phi^{\bullet}_{g'}(\mathbf{r}) + \frac{1}{k^{\bullet}}\sum_{g'}^{NOG}\chi^{g}\nu\Sigma^{\bullet}_{f.g'}(\mathbf{r})\,\phi^{\bullet}_{g'}(\mathbf{r})$$

$$g = 1, \ldots, NOG \tag{2.24}$$

Note that in Equation 2.24 the unperturbed chi factor, χ, is used. An explicit perturbation on χ is not included here because it is not a typical perturbation parameter (Bell and Glasstone, 1970; Engle et al., 1954; Hansen and Maier, 1960; Peterson, 1953), and it is not possible to use more than one value in a flux calculation (Alcouffe et al., 1992; O'Dell et al., 1989).

With the adjoint (Equation 2.22) and perturbed (Equation 2.24) transport equations defined, it is now possible to proceed with the derivation of the reactivity equation. The general procedure for doing this is as follows (Bell and Glasstone, 1970; Carmichael, 1970; Hansen and Maier, 1960; Hopkins, 1971; Weinberg and Wigner, 1958). First, the adjoint equation is multiplied by the forward perturbed angular flux. Similarly, the perturbed equation is multiplied through by the adjoint angular flux. These two equations are then subtracted from one another at which point the resulting equation is integrated over all phase space (d**r**, dE, and dΩ). Finally, the equation is solved for the change in the eigenvalue which gives the reactivity. The remainder of this section provides some of the details that lead to the final reactivity equation.

First, Equation 2.22 (the adjoint transport equation) is multiplied by the perturbed angular flux, $\Psi^{\bullet}_{g}(\mathbf{r},\Omega)$, and Equation 2.24 (the perturbed transport equation) is multiplied by the adjoint angular flux, $\Psi^{*}_{g}(\mathbf{r},\Omega)$. Equation 2.25 shows the result after these two equations are subtracted from one another.

$$\Psi^{*}_{g}\,\Omega\bullet\nabla\Psi^{\bullet}_{g} + \Psi^{\bullet}_{g}\,\Omega\bullet\nabla\Psi^{*}_{g} + \Sigma^{\bullet}_{t.g}\Psi^{\bullet}_{g}\Psi^{*}_{g} - \Sigma^{g}_{t}\,\Psi^{\bullet}_{g}\Psi^{*}_{g} =$$

$$\sum_{g'}^{NOG} \Sigma^{\bullet}_{s.g'\rightarrow g}\,\phi^{\bullet}_{g'}\,\Psi^{*}_{g} - \sum_{g'}^{NOG} \Sigma_{s}^{g\rightarrow g'}\,\phi^{*}_{g'}\,\Psi^{\bullet}_{g} +$$

$$\frac{1}{k^{\bullet}} \sum_{g'}^{NOG} \chi^{g}\,\nu\Sigma^{\bullet}_{f.g'}\,\phi^{\bullet}_{g'}\,\Psi^{*}_{g} - \frac{1}{k} \sum_{g'}^{NOG} \chi^{g'}\,\nu\Sigma^{g}_{f}\,\phi^{*}_{g'}\,\Psi^{\bullet}_{g}$$

$$g = 1, \ldots, NOG \qquad (2.25)$$

Integrating Equation 2.25 will be delayed a few steps to simplify the notation. At this point, the primary simplifying assumption is introduced; the perturbation effect on the system is small so the perturbed fluxes are not appreciably different from the original unperturbed fluxes (see Equation 2.26). With this small perturbation constraint, the unperturbed fluxes can be substituted into Equation 2.25 in place of the perturbed fluxes with little error (Ussachoff, 1955).

$$\Psi_g^{\bullet} \approx \Psi_g \tag{2.26a}$$

$$\phi_g^{\bullet} \approx \phi_g \tag{2.26b}$$

The assumptions in Equation 2.26 are the source of the name first order perturbation theory. If the perturbed flux was represented in the form of a Taylor expansion, the error term would be first order in $\Delta\Psi$ (Appendix B illustrates this for another case).

The next simplification is to expand the perturbed eigenvalue using a Taylor expansion, the result is shown in Equation 2.27 (see Appendix B for the details). This expansion is valid for small changes in 1/k, but this requirement is all ready satisfied when using the flux approximation in Equation 2.26.

$$\frac{1}{k^{\bullet}} = \frac{1}{k} - \frac{\Delta k}{k^2} \tag{2.27}$$

Substituting Equations 2.26 and 2.27 into Equation 2.25 results in Equation 2.28 after integrating this equation over all phase space. Note that the energy dependence has already been accounted for by applying the multigroup energy approximation earlier, so integrals over energy are not required.

$$\int\int(\Psi_g^* \, \Omega\bullet\nabla\Psi_g + \Psi_g \, \Omega\bullet\nabla\Psi_g^*) \, drd\Omega + \int\int(\Sigma_{t.g}^{\bullet} \, \Psi_g\Psi_g^* - \Sigma_t^g \, \Psi_g\Psi_g^*) \, drd\Omega =$$

$$\int\int\int\left(\sum_{g'}^{NOG} \Sigma_{s.g'\to g}^{\bullet} \, \phi_{g'} \, \Psi_g^* - \sum_{g'}^{NOG} \Sigma_s^{g\to g'} \, \phi_{g'}^* \Psi_g\right) drd\Omega +$$

$$\int\int\int\left(\frac{1}{k}\sum_{g'}^{NOG} \chi^g \, \nu\Sigma_{f.g'}^{\bullet} \, \phi_{g'}\Psi_g^* - \frac{\Delta k}{k^2}\sum_{g'}^{NOG} \chi^g \, \nu\Sigma_{f.g'}^{\bullet} \, \phi_{g'}\Psi_g^*\right) drd\Omega -$$

$$\int\int\int\frac{1}{k}\sum_{g'}^{NOG} \chi^{g'} \, \nu\Sigma_f^g \, \phi_{g'}^*\Psi_g^{\bullet} \, drd\Omega$$

$$g = 1, \ldots, NOG \qquad (2.28)$$

In simplifying this equation, the first integral term on the left-hand side will be addressed first. This first term contains all the leakage operators and is eliminated by application of the forward (Equation 2.2) and adjoint (Equation 2.22) boundary conditions. Using Gauss' Divergence Theorem from Equation 2.19, the volume integrals that are operating on the leakage terms are converted into surface integrals over which the boundary conditions are defined. The result is shown in Equation 2.29.

$$\int\int(\Psi_g^* \, \Omega\bullet\nabla\Psi_g + \Psi_g \, \Omega\bullet\nabla\Psi_g^*) \, drd\Omega = \int\int\Omega\bullet\nabla\Psi_g\Psi_g^* \, drd\Omega =$$

$$\int\int\nabla\bullet(\Omega\Psi_g\Psi_g^*) \, drd\Omega = \int\int\mathbf{n}\bullet\Omega\Psi_g\Psi_g^* \, dSd\Omega =$$

$$\int\int_{\mathbf{n}\bullet\Omega<0}\int\mathbf{n}\bullet\Omega\Psi_g\Psi_g^* \, dSd\Omega + \int\int_{\mathbf{n}\bullet\Omega>0}\int\mathbf{n}\bullet\Omega\Psi_g\Psi_g^* \, dSd\Omega = 0 \qquad (2.29)$$

The integral in Equation 2.29 goes to zero because of the forward and adjoint boundary conditions: $\Psi_g(\mathbf{r}_s,\Omega) = 0$ if $\Omega\bullet\mathbf{n}_s < 0$ and $\Psi_g^*(\mathbf{r}_s,\Omega) = 0$ if $\Omega\bullet\mathbf{n}_s > 0$. The surface over which the integration in Equation 2.29 is performed is the same surface on which the

boundary conditions are specified, and since the forward and adjoint angular fluxes are zero on the surface, the resulting integral is also zero.

Now is a good time to perform the angular integrals on the remaining four integral terms in Equation 2.28. Note that the only factors dependent on angle in Equation 2.28 are the angular fluxes, so the angular integral can be pulled inside the summations on the scattering and fission terms such that the integrand contains only the angular flux. Recalling the approximation in Equation 2.11 where the integral of the angular flux roughly equals the sum of the angular flux times its appropriate weighting factor, and that this term by definition is the scalar flux, removes the angular dependence from the scatter and fission terms. The numerical integration of the total cross section integral is essentially the same except there is no way to explicitly remove the angular flux terms as was done in Equation 2.11. For this case, the numerical integration looks like:

$$\int \Psi_g(\mathbf{r},\Omega)\Psi^*_g(\mathbf{r},\Omega)\, d\Omega \approx \sum_{m=1}^{MM} w_m \Psi_{g.m}(\mathbf{r})\, \Psi^*_{g.m}(\mathbf{r}) \tag{2.30}$$

Equation 2.28 now looks like:

$$\int \sum_{m=1}^{MM} (\Sigma^{\bullet}_{t.g}\, w_m \Psi_{g.m}\Psi^*_{g.m} - \Sigma^g_t\, w_m \Psi_{g.m}\Psi^*_{g.m})\, d\mathbf{r} =$$

$$\int\left(\sum_{g'}^{NOG} \Sigma^{\bullet}_{s.g'\to g}\, \phi_{g'}\, \phi^*_g - \sum_{g'}^{NOG} \Sigma_s^{g\to g'}\, \phi^*_{g'}\, \phi_g\right) d\mathbf{r} +$$

$$\int\left(\frac{1}{k}\sum_{g'}^{NOG} \chi^g\, \nu\Sigma^{\bullet}_{f.g'}\, \phi_{g'}\, \phi^*_g - \frac{\Delta k}{k^2}\sum_{g'}^{NOG} \chi^g\, \nu\Sigma^{\bullet}_{f.g'}\, \phi_{g'}\, \phi^*_g\right) d\mathbf{r} -$$

$$\int\frac{1}{k}\sum_{g'}^{NOG} \chi^{g'}\, \nu\Sigma^g_f\, \phi^*_{g'}\, \phi_g\, d\mathbf{r}$$

$$g = 1, \ldots, NOG \tag{2.31}$$

The total cross section term, the scattering cross section term, and part of the fission term can be simplified by using the definitions in Equation 2.23. Even though some of the indices do not match in the fission and scattering terms, when the series are written out, the terms reduce to those listed in Equation 2.32.

$$\int \sum_{m=1}^{MM} (\delta\Sigma_t^g \, w_m \Psi_{g.m} \, \Psi^*_{g.m}) \, dr = \int\left(\sum_{g'}^{NOG} \delta\Sigma_s^{g\rightarrow g'} \, \phi^*_{g'} \, \phi_g\right) dr -$$

$$\int\left(\frac{\Delta k}{k^2} \sum_{g'}^{NOG} \chi^g \, \nu\Sigma^{\bullet}_{f.g'} \, \phi_{g'} \, \phi^*_g\right) dr + \int\left(\frac{1}{k} \sum_{g'}^{NOG} \delta(\chi^{g'} \, \nu\Sigma_f^g)\phi^*_{g'} \, \phi_g\right) dr$$

$$g = 1, \ldots, NOG \qquad (2.32)$$

Only two steps remain to derive the reactivity equation. The first step is to integrate each term over the entire volume of the system. This process is accomplished using the approximation in Equation 2.8. The only values that are not dependent upon position in Equation 2.32 are the factors involving k, the multiplication factor. The remaining values are position dependent, so they will end up with position dependent indices. In this case, the weighting factor will be the fine mesh volume at each calculation point which is determined by ONEDANT. The general numerical integration result over **dr** is shown in Equation 2.33 where IT is the total number of fine mesh spaces, V is the volume of each fine mesh space, and σ_x is some arbitrary position dependent group constant.

$$\int \sigma_x(\mathbf{r}) \, w_m \Psi_{g.m}(\mathbf{r}) \, \Psi^*_{g.m}(\mathbf{r}) \, dr \approx \sum_{l=1}^{IT} V_l \, \sigma_{x.l} \, w_m \Psi_{g.m.l} \, \Psi^*_{g.m.l} \qquad (2.33)$$

The last step is to solve the equation for $\Delta k/k^2$ which is the reactivity, ρ. This result is shown in Equation 2.34 where there are three terms in the numerator and one term in the denominator of this equation. Equation 2.34 has units of fraction of reactivity; dividing by the effective delayed neutron fraction (β_{eff}) will convert the units to dollars of reactivity.

$$\rho = \frac{\Delta k}{k^2}$$

$$= \frac{\frac{1}{k}\sum_{l=1}^{IT}\sum_{g=1}^{NOG}\sum_{g'}^{NOG} v_l \delta(\chi^{g'} \nu\Sigma_f^{g.l})\phi^*_{g'.l}\,\phi_{g.l} + \sum_{l=1}^{IT}\sum_{g=1}^{NOG}\sum_{g'=1}^{NOG} v_l \delta\Sigma_s^{g\rightarrow g'.l}\,\phi^*_{g'.l}\,\phi_{g.l}}{\sum_{l=1}^{IT}\sum_{g=1}^{NOG}\sum_{g'}^{NOG} v_l\,\chi^g\,\nu\Sigma^{\bullet}_{f.g'.l}\,\phi_{g'.l}\,\phi^*_{g.l}}$$

$$- \frac{\sum_{l=1}^{IT}\sum_{g=1}^{NOG}\sum_{m=1}^{MM} v_l\, w_m\, \delta\Sigma_t^{g.l}\,\Psi_{g.m.l}\,\Psi^*_{g.m.l}}{\sum_{l=1}^{IT}\sum_{g=1}^{NOG}\sum_{g'}^{NOG} v_l\,\chi^g\,\nu\Sigma^{\bullet}_{f.g'.l}\,\phi_{g'.l}\,\phi^*_{g.l}} \qquad (2.34)$$

2.4 Comparison with Other Formulations

In this section, Equation 2.34 is compared with other reactivity equations to confirm the validity of the derivation in the previous section. In particular, first order perturbation theory calculations by Bell and Glasstone (1970), Hansen and Maier (1960), and Carmichael (1970) are reviewed.

Bell and Glasstone define the first order reactivity equation as follows:

$$\frac{\Delta k}{k^*} \approx \frac{-\int \Delta\sigma\ \phi^\dagger\ \phi\ dV\ d\Omega\ dE\ +\ \int \Delta[\sigma f]\ \phi^\dagger\ \phi\ dV\ d\Omega'\ dE'\ d\Omega\ dE}{\frac{1}{4\pi}\int \nu\sigma_f\ \phi^\dagger\ \phi\ dV\ d\Omega'\ dE'\ d\Omega\ dE} \qquad (2.35)$$

where $\sigma f = \frac{\nu\sigma_f}{4\pi} + \sigma_x f_x$ $\qquad$ $\sigma_x \equiv$ scatter cross sections

$\sigma \equiv$ total cross section $\qquad$ $\phi^\dagger \equiv$ adjoint flux

$k^* \equiv$ perturbed eigenvalue

Equation 2.35 matches Equation 2.34 except for two points: (1) the denominator in Equation 2.35 contains unperturbed cross sections whereas Equation 2.34 has perturbed cross sections and (2) the perturbed eigenvalue is present in Equation 2.35 whereas the unperturbed eigenvalue is used in Equation 2.34. The perturbed eigenvalue in Equation 2.35 arises from the use of a different expansion technique for the perturbed eigenvalue (assuming k = 1), in particular (Bell and Glasstone, 1970):

$$\frac{1}{k^*} = \frac{1}{k^*} - \frac{1}{k} + \frac{1}{k} = \frac{1}{k} + \frac{-\Delta k}{kk^*} \qquad (2.36)$$

However, the Taylor expansion in Equation 2.27 was used to obtain Equation 2.34. For small changes in k (less than 10%), the two approximations (Equations 2.27 and 2.36) are within 1% of each other.

Based on the equations used by Bell and Glasstone to derive the reactivity equation (Equation 2.35), the perturbed value for the fission cross section should be used in the denominator. Since it is not included in the final form, the assumption was most likely made that, because the perturbation has such a small effect on the system, performing the integral over the entire volume with the perturbed cross sections will not differ much from the integral calculated using the unperturbed cross sections. This is a reasonable assumption given that experimental reactivity measurements are typically based on adding small quantities of material to a void rather than replacing fuel with the perturbation material (Engle et al., 1954; Peterson, 1953). Also, the requirement that the perturbation has a small effect on the flux distribution (see Equation 2.26) minimizes the effect regardless of whether void or fuel is displaced.

Next, the reactivity derivation presented by Hansen and Maier (1960) is compared. The general form of their reactivity equation is:

$$\Delta k = 1 - \frac{1}{k^p} = \tag{2.37}$$

$$\frac{\int \Psi^p \Delta[\chi \nu \sigma_f + \sigma]\, \phi^\dagger \, dr\, d\Omega'\, dE'\, d\Omega\, dE \; - \; \int \Psi^p\, \Delta[\sigma_t]\, \phi^\dagger \, dr\, d\Omega\, dE}{\int \Psi^p\, [\chi \nu \sigma_f]^p\, \phi^\dagger \, dr\, d\Omega'\, dE'\, d\Omega\, dE}$$

where $\sigma \equiv$ scatter cross section $\quad \phi^\dagger \equiv$ adjoint flux

$k^p \equiv$ perturbed eigenvalue $\quad \Psi^p \equiv$ perturbed forward flux

Note that in Equation 2.37, the perturbed cross sections are present in the denominator. After applying the assumptions for first order perturbation theory, Equation 2.37 takes the final form (Hansen and Maier, 1960):

$$\Delta k \approx \tag{2.38}$$

$$\frac{\int \phi\, \Delta[\chi \nu \sigma_f + \sigma]\, \phi^\dagger \, dr\, d\Omega'\, dE'\, d\Omega\, dE \; - \; \int \phi\, \Delta[\sigma_t]\, \phi^\dagger \, dr\, d\Omega\, dE}{\int \phi\, [\chi \nu \sigma_f]\, \phi^\dagger \, dr\, d\Omega'\, dE'\, d\Omega\, dE}$$

Equation 2.38 is identical in form to Equation 2.35, thus the previous discussion also applies here.

Last, the derivation by Carmichael (1970) is presented. Their final form of the reactivity equation is already discretized and has been simplified to correspond with the derivation in Equation 2.34 (i.e. no delayed neutrons and a single value for χ) as follows:

$$\rho = \frac{k' - k}{k} \approx I_f + I_s - I_t \tag{2.39}$$

where

$$I_f = \frac{1}{F}\sum_{\gamma}\sum_{g}\sum_{g'} \frac{1}{k} v_\gamma \chi^{\gamma}_{g'} \Delta(\nu\sigma_f)_{g,\gamma}\phi_{g,\gamma}\phi^*_{g',\gamma}$$

$$I_s = \frac{1}{F}\sum_{\gamma}\sum_{g}\sum_{g'} v_\gamma \Delta(\sigma_s)_{g',g,\gamma}\phi_{g',\gamma}\phi^*_{g,\gamma}$$

$$I_t = \frac{1}{F}\sum_{\gamma}\sum_{g}\sum_{n} v_\gamma w_n \Delta(\sigma_t)_{g,\gamma}\Psi_{g,\gamma,n}\Psi^*_{g,\gamma,n}$$

$$F = \sum_{\gamma}\sum_{g}\sum_{g'} \frac{1}{k} v_\gamma \chi^{\gamma}_{g'} (\nu\sigma_f)_{g,\gamma}\phi_{g,\gamma}\phi^*_{g',\gamma}$$

In the above equation, γ is the isotope index, g and g' are energy indices, n is the angular quadrature index, * indicates an adjoint parameter, ϕ is the scalar flux, and Ψ is the angular flux. Again, unperturbed cross sections are used in the denominator as explained before. There also appears to be an extra factor of 1/k in the denominator (the F term). However, if both sides of Equation 2.39 are multiplied by a factor of 1/k, the 1/k factor cancels in the denominator and the left-hand side looks like $\Delta k/k^2$ which is the same form found in Equation 2.34 when the Taylor expansion approach is used to remove the perturbed eigenvalue from the equation.

Based on these three comparisons, it appears the formulation in Section 2.3 is consistent with the other references.

CHAPTER 3

DISCUSSION OF PERTURBATION CODE

This chapter discusses the incorporation of the reactivity equation derived in the last chapter into a computer code that uses the flux, cross section, and geometry data generated from ONEDANT. The computer code operation and limitations are discussed, and a sample problem is reviewed.

3.1 Overview of Computer Code

A computer code named REACT was written using FORTRAN (Etter, 1987) to calculate the reactivity using Equation 2.34. The necessary cross section, geometry, and neutron flux data for this program are supplied from the binary files generated from running ONEDANT. The specific perturbation data (e.g. perturbation location, replacement material, and delayed neutron fraction) are entered from the keyboard. The reactivity code gathers all the data, calculates the reactivity, and prints the results to the screen with the units of fraction and dollars of reactivity.

The reactivity code can be run on any computer system that has a FORTRAN compiler, but to avoid problems with converting the binary files to different machine formats, it is recommended that it be run on the same machine as was ONEDANT. The arrays used in the reactivity program are not dynamically dimensioned, so there is the possibility that the machine may not have sufficient memory to create the executable file. However, this has not been a problem on a SUN SPARC station IPX.

Performing a reactivity calculation experimentally requires two measurements: (1) of the unperturbed critical system and (2) of the perturbed critical system. The unperturbed system consists of the assembly without the replacement sample in place. The system is usually setup so there is an empty space in the location where the sample will go. In cases where the samples are in containers, an empty container is placed where the sample will go

for the unperturbed measurement. Once the critical condition has been achieved in the assembly for the unperturbed case, the replacement material is placed in the assembly for the perturbed measurement. The assembly is again brought to critical with the sample in the assembly. The change in control rod position between the perturbed and unperturbed measurements is a measure of the reactivity worth of the sample. A positive reactivity indicates the system was made supercritical by adding the sample. A negative reactivity indicates the system went subcritical with the addition of the sample. The reactivity measurement method describe here is just one way to do a reactivity calculation (Henry, 1975) and is the one used in the Lady Godiva measurements (Engle et al., 1954; Engle et al., 1960; Hansen and Maier, 1960) that will be modeled shortly.

The experimental reactivity calculation can be duplicated numerically using perturbation theory. With first order perturbation theory, only the unperturbed system needs to be modeled, but two calculations need to be performed: one for the unperturbed forward flux and one for the unperturbed adjoint flux as indicated by Equation 2.34. In most cases, the modeled system does not exactly match the actual experimental system because of geometrical limitations (i.e. spherical, cylindrical, or Cartesian) of the transport code used to calculate the neutron fluxes in the assembly. The next two sections discuss how the ONEDANT input file is setup and run for the flux calculations and how the reactivity code works. Finally, Section 3.4 goes step-by-step through a sample problem with the REACT code.

3.2 Running ONEDANT code

As stated previously, ONEDANT solves the one dimensional time independent transport equation using the multigroup energy approximation (Duderstadt and Hamilton, 1976; O'Dell et al., 1989; O'Dell and Alcouffe, 1987) and the discrete ordnance approximation for the angular dependence (O'Dell et al., 1989; O'Dell and Alcouffe, 1987). Position dependence is handled by using finite differencing techniques, and several

methods are used to accelerate the convergence of the fluxes and eigenvalues (O'Dell et al., 1989). The input file to ONEDANT supplies the geometry, material compositions, and calculational parameters necessary to solve the problem. Appropriate cross section data must also be supplied either in the input file or in a separate file.

As ONEDANT solves the transport equation, many binary data files (O'Dell et al., 1989; O'Dell, 1977) are generated. These files contain the various code specifications, angular and scalar fluxes, macroscopic cross sections, etc. that will be used by the reactivity code. Table I lists the particular binary files that are needed by the reactivity code. The first column contains the name of the binary file, the second column gives a brief description of the file, and the last column indicates whether the file was created from a forward run (F) or an adjoint run (A) of ONEDANT. The data in column 3 will be discussed in detail later. Detailed file formats for all of the binary files are listed in either Appendix A of the ONEDANT manual (O'Dell et al., 1989) or in the report discussing standard interface files (O'Dell, 1977).

Table I. ONEDANT binary files used by reactivity code.

File Name	Contents	Run (F/A)
AAFLUX	adjoint angular fluxes	A
ATFLUX	adjoint scalar fluxes	A
GEODST	geometry data for problem: coarse mesh intervals, material assignments to mesh intervals, etc.	F
MACRXS	material macroscopic cross sections	F
RAFLUX	forward angular fluxes	F
RTFLUX	forward scalar fluxes	F
SNCONS	angular quadrature constants	F

To use the REACT code, it is necessary to run the ONEDANT code in both the forward and adjoint calculational modes. Figure 2 illustrates a sample input file for ONEDANT. Additional information on how to setup this file is available in the ONEDANT manual (O'Dell et al., 1989).

```
     1    0     0
 Lady Godiva Model, unperturbed case
/
/ *** block i  ***
     igeom=sph ngroup=16 isn=16 niso=118 mt=3
     nzone=3 im=2 it=47
       t
/
/ *** block ii (geometry) ***
     xmesh=0.0,0.7269,8.760
     xints=7 40
     zones=0 1
           t
/
/ *** block iii (cross sections) ***
     lib=bxslib
      t
/
/ *** block iv (mixing) ***
     matls=fuel "235-yr" 0.044879
                "u238y"  0.003018;
           s1 "235-yr"   0.047912;
           s2 "u238y"    0.047306;
     assign=matls;
                  t
/
/ *** block v (solver) ***
     ith=0 ievt=1 isct=0 ibr=0 raflux=1
     epsi=0.00001 epso=0.000001
              t
```

Figure 2. Sample input file for ONEDANT: modeling the unperturbed Lady Godiva assembly.

In Figure 2 the sample problem is a ONEDANT model of the unperturbed Lady Godiva critical assembly. Lady Godiva is a sphere of uranium metal with a radius of roughly 8.76 cm. A one inch diameter glory hole runs through the center of the assembly where replacement samples, 1/2" x 1/2" right circular cylinders, are placed and packed around uranium metal inserts. The sample itself is placed inside a hollowed insert. Figure 3 is a

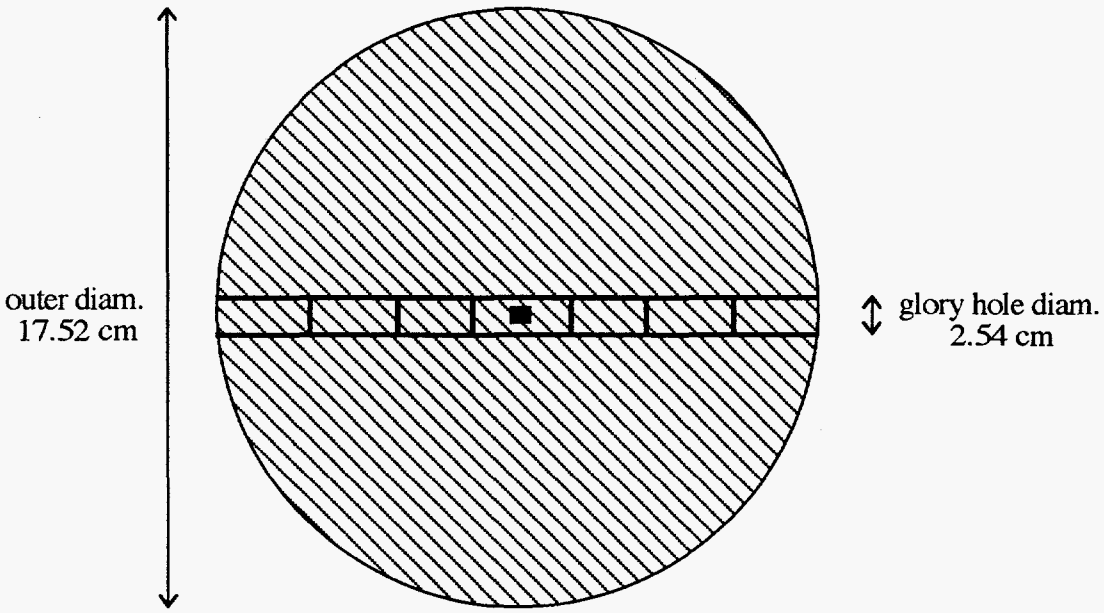

Figure 3. Simplified cross sectional drawing of the Lady Godiva critical assembly.

cross sectional drawing of the Godiva assembly illustrating some of these features. An effective density is used in the number density calculations to account for the presence of various voids in the assembly which are not modeled in ONEDANT. In the one-dimensional model listed in Figure 2, the cylindrical samples are modeled as spheres with the same volume and the glory hole is assumed to be completely filled with uranium metal inserts as was the case in the actual experiments.

The XMESH statement in Block II lists the boundaries of the replacement sample (0.7269 cm) and the critical assembly (8.760 cm). The ZONES statement indicates that the first coarse mesh region is composed of a void, representing the location of the replacement sample for the unperturbed case, and the second coarse mesh region is fuel.

Block IV is where the material compositions are supplied to ONEDANT. Both the unperturbed materials, those comprising the assembly: fuel, shielding, etc., and the replacement materials are defined here. The MATLS statement creates the material macroscopic cross sections from the isotopic microscopic cross sections supplied in Block III. The material cross sections are assigned zone numbers with the ASSIGN statement. These zone numbers are used by the ZONES statement in Block II to assign material cross sections to each of the coarse mesh intervals defined in the XMESH statement. The replacement materials will not be used in the flux calculation because they are not included in the ZONES statement in Block II, but they will be supplied to the MACRXS file because they are included in the ASSIGN statement of Block IV. Based on the ASSIGN=MATLS statement used in the sample problem:

$$\begin{aligned} &\text{Material 1} \equiv \text{Zone 1} \equiv \text{"fuel"} \\ &\text{Material 2} \equiv \text{Zone 2} \equiv \text{"s1"} \\ &\text{Material 3} \equiv \text{Zone 3} \equiv \text{"s2"} \end{aligned} \qquad (3.1)$$

NOTE: the ASSIGN=MATLS statement must be used when making the materials. Any mixing performed in the ASSIGN statement is not directly saved; only the mixing in

the MATLS statement is retained explicitly in the MACRXS file. REACT is not setup to handle remixing materials after the fact; therefore, all the material mixing must be done in the MATLS statement; consequently, the user is relied on to perform this calculation.

There is no real limit on the number of materials that can be specified in the MATLS array provided there is enough computer memory to store all the data. The same is true for the REACT code. However, if more samples need to be calculated which were not included in the original MACRXS file, another file can be created from ONEDANT without running the entire flux calculation again. An example of an input file used in this situation is shown in Figure 4.

```
     1    0     0
 Lady Godiva Model, unperturbed case
/
/ *** block i  ***
     igeom=sph ngroup=16 isn=16 niso=118 mt=3
     nzone=3 im=2 it=47 nosolv=1
       t
/
/ *** block ii (geometry) ***
     xmesh=0.0,0.7269,8.760
     xints=7 40
     zones=0 1
           t
/
/ *** block iii (cross sections) ***
     lib=bxslib
      t
/
/ *** block iv (mixing) ***
     matls=fuel "235-yr" 0.044879
                "u238y"  0.003018;
           s1 "b"        0.14151;
           s2 "ni"       0.09130;
     assign=matls;
                 t
```

Figure 4. Sample input file for ONEDANT for obtaining additional macroscopic cross sections without rerunning the entire code.

Note the differences between the input file in Figure 2 from the one in Figure 4:

(1) The statement NOSOLV=1 was added to Block I. This statement suppresses the running of the solver module.

(2) The replacement samples in Block IV are for different materials. The unperturbed materials, in this case the "fuel", should always be included. Only the replacement materials (i.e. "s1", "s2", ...) should be changed, added, or deleted.

(3) Block V has been deleted from the input file. This last step in not necessary because the NOSOLV option prevents the solver module from executing.

After running ONEDANT with the abbreviated input file, two files, the MACRXS file and the GEODST file, need to replace the previous file versions. Both of these files contain information relevant to the number and type of materials that were created in Block IV.

To obtain the forward and adjoint fluxes, the code has to be run twice. Once with the ITH parameter set equal to zero (forward calculation) and once with ITH set equal to 1 (adjoint calculation). Referring back to the last column in Table I, the files that are used for the reactivity code from the forward run are those with the letter F in the third column. Using the MACRXS and SNCONS files from the forward run eliminates several conversion steps relating to energy and angle order. Specifically, the energy order is reversed and the angular directions are reflected in the files from the adjoint run (O'Dell et al., 1989; O'Dell, 1977). It does not make a difference which run the GEODST file is pulled from, so the forward file was chosen and tested. Similarly, the files in Table I with an A in the last column should be taken from the adjoint run.

Any multigroup cross section library with any scattering order can be used with ONEDANT to calculate the fluxes. In the reactivity calculation, REACT only uses the P_0 scatter cross sections; however, REACT can handle cross section libraries with higher orders of scattering; this information is simply not stored after it has been read in from the MACRXS file. The only concern in handling higher orders of scattering is that the SCAT array in REACT is dimensioned large enough to hold all the scattering data initially. There

is indeed a check to make sure of this fact and appropriate error messages are listed if the check fails. However, when running ONEDANT to obtain the forward and adjoint fluxes, use the highest order of scattering (ISCT) that results in the most reasonable eigenvalue. Using a lower scattering order in the flux calculation does not reduce the number of scattering cross sections stored in the MACRXS file; all the available scattering cross sections are included. Calculations illustrating the effect of scattering order on the calculated reactivity are included in the results section.

A final note is presented on how ONEDANT handles changes to the χ parameter that can be made on the input file. The ONEDANT manual indicates there are two locations in the input file where the value of χ can be changed. One location is in Block III with the CHIVEC statement and the other is in Block V with the CHI statement. When the cross section file is in the form of a BXSLIB, a XSLIBB, or a MACBCD file, using the CHIVEC statement in Block III does not change the χ values that are used in the flux calculation or that are written to the MACRXS file. The χ values stored in the cross section file are still used. If the CHI statement in Block V is used to change the χ values, the new values input in Block V are used in the flux calculation; however, these new values are not stored in the MACRXS file. The value stored in MACRXS is the χ that was in the input cross section file originally. This can be a problem if the χ's stored on the cross section file are all zeros.

To get around this problem, edit the cross section file directly, before running ONEDANT, to update the χ values. If the cross section file is a binary file, do the following to create an ASCII file that can be edited. Run a ONEDANT input file with NOSOLV = 1 in Block I (as shown in Figure 4) and add WRITMXS = XSLIBB to Block III. This step creates an ASCII version of the BXSLIB file. Make the necessary changes to the χ data which are located at the top of the XSLIBB file. Now, run the ONEDANT input file without the NOSOLV and WRITMXS statements and change the LIB statement

in Block III to LIB = XSLIBB. Now the new χ data will be used in the flux calculation and written to the MACRXS file where it can be accessed by the reactivity code.

3.3 Operation of Reactivity Code

Once all the binary files are gathered and placed in a directory where they can be accessed, the REACT code can be executed. The only information that the user will have to supply is (1) the number of fine mesh locations undergoing a perturbation, (2) the number of each fine mesh interval with its corresponding replacement material number, and (3) the effective delayed neutron fraction for the system. The mesh interval and replacement material numbers are entered as single line, space delineated entry of the form: "FM# MAT#". The order in which the meshes are entered is not important. If the delayed neutron fraction is not known, simply enter a zero; the reactivity in units of dollars will not be calculated. Recall that the material numbers are based on the order in which the materials are listed in the MATLS statement (see the relations in Equation 3.1). Once all the data is entered, the calculation will proceed and the results will be printed on the screen. Finally, the option to do another calculation with the existing data files is presented. An affirmative answer prompts the user for the next set of perturbation data while a negative answer ends the program (see Figure 5).

Figure 5 is a calculational flow diagram for REACT. REACT, indicated by the dotted box in Figure 5, is the driver code that calls the subroutines, indicated by the rounded rectangles in Figure 5, to do their assigned tasks. The order shown in Figure 5 is the order in which each of the subroutines is executed. Appendix C contains descriptions for each of the subroutines shown in Figure 5. Note that in case there is an error in reading one of the files, all the binary files are read before the user is prompted to enter the perturbation information from the terminal.

Several checks have been implemented in the code to minimize erroneous computations. For instance, when the flux files are read, the number of dimensions is

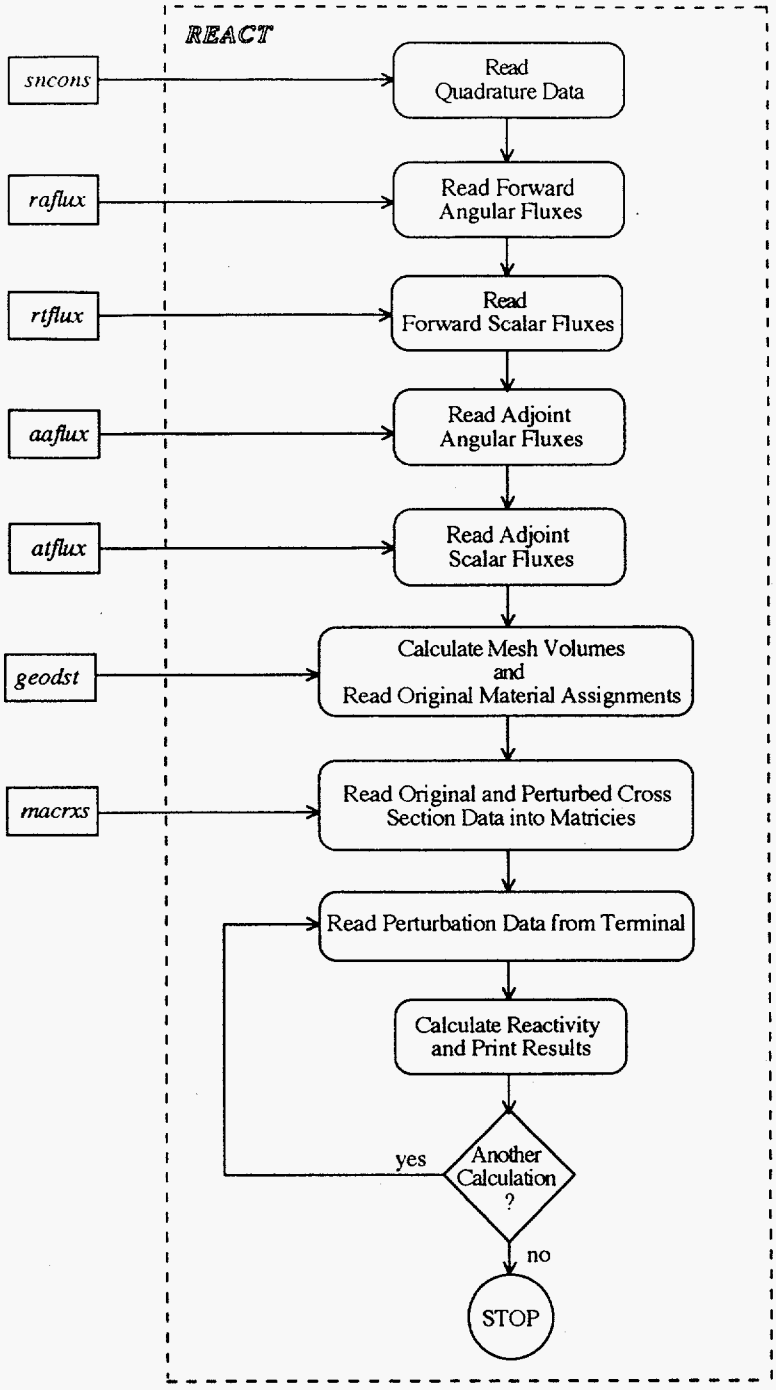

Figure 5. Flow diagram of reactivity code REACT. The rectangular boxes are input files, the rounded rectangular boxes represent subroutines, and the outer dashed box represents the main program REACT.

verified not to be greater than one. This was added to ensure that files from TWODANT (Alcouffe et al., 1992), the two dimensional version of ONEDANT, were not used inadvertently. This is a possible concern because some of the file names used by TWODANT are the same as those used by ONEDANT.

When data is stored in arrays, the number of data points stored in the array is compared with the maximum dimensions of the array to assure that the arrays are not undersized which could result in the loss of data. The maximum dimension (LIM) is set at 100 for all the arrays with the exception of the SCAT array located in the subroutine that reads the cross section data. This array is dimensioned at 1000 (NSLIM) because SCAT is a one dimensional array that holds all the scatter cross sections (P_0 and up) for all the materials of a given energy group when they are initially read from the binary file MACRXS. This can be quite a large number. Variables that are checked include: the number of energy groups (NOG), the number of fine mesh points (IT), the number of fine mesh boundaries (IT+1), the number of angular directions (MM), and the number of scattering terms. Whenever one of these checks fails, an error message listing the subroutine in which the error occurred and the type of error is printed on the terminal. The program then terminates so that the problem can be fixed.

Another check is performed on the replacement material entered from the keyboard. If the entered material number is greater than the available material numbers, an error message is printed on the screen and the operator is asked to reenter both the mesh number and the material number again. A warning is also printed if the perturbation material and the original material are the same for a given fine mesh space. This occurrence does not cause a fatal error, but it may be an indication that data was not entered properly. The resulting calculated reactivity will include this data variation, so the result may be incorrect. Simply type "y" when asked to do another calculation and reenter the perturbation data again. This check is performed during the reactivity calculation, so nothing can be changed until the calculation is finished.

3.4 Sample Problem Using Reactivity Code

In this section a specific calculation is stepped through from beginning to end. The task is to calculate the central reactivity worths in the Lady Godiva assembly for a pure metal sample of ^{235}U and a pure metal sample of ^{238}U. Lady Godiva (Engle et al., 1954; Engle et al., 1960; Hansen and Paxton, 1969; Peterson, 1953; Wenz and Busch, 1994) is essentially a spherical uranium metal assembly which has a critical mass of 52.65 kg, an effective density of 18.7 g/cm^3, and a ^{235}U enrichment of 93.7 wt%. The radius of a sphere with these mass and density values is 8.76 cm. Reactivity worth measurements were performed in Lady Godiva (Engle et al., 1954; Engle et al., 1960; Peterson, 1953) by placing samples in the center of the core. Specifically, the glory hole was filled with uranium metal plugs. The plug at the center was hollow so that a cylindrical 1/2" by 1/2" sample could be placed inside it. For the unperturbed case, the plug at the center was empty; the plug was then filled with a given material for the perturbed measurement. Experimentally, the change in the control rod positions between the unperturbed and the perturbed systems was used to infer the reactivity of the sample based on previous control rod calibration measurements (Engle et al., 1954).

The input file for this calculation, Figure 2, has been discussed previously. Based on this input file, there are seven fine mesh regions (see XINTS statement in Block II) established in the first coarse mesh region that define the replacement sample. Since the material assigned to this region is a void (ZONE 0), this model is for the unperturbed system, as it should be. The order of the materials in the MATLS statement in Block IV corresponds to the material numbers that will be used when REACT is run (i.e. material number 1 is "fuel", material 2 is "s1", and material 3 is "s2").

The ONEDANT code is ready to be run in the forward and adjoint modes to obtain the necessary binary files. After running the first ONEDANT calculation, move the appropriate files (based on Table I) to another directory so that they will not be overwritten

when ONEDANT is run for the second time. Once all the files are generated, place them all together where REACT can access them.

Now the REACT code can be executed. Appendix D contains the listing of the interactive session between the user and the program. The plain text shows the questions asked by the code, and the responses from the user are in bold text. In this example, the executable file is called "xpert." First the number of perturbed fine mesh spaces is entered:

>7

Next, the fine mesh number followed by the replacement material number are entered:

>1 2 <ret>

>2 2 <ret>

⇓

>7 2 <ret>

Last, the effective delayed neutron fraction is entered:

>0.0065

The reactivity fraction and the reactivity in units of dollars are then printed to the screen. When asked to do another calculation, the response is yes, so the code requests new perturbation data again. The entries are the same as before except material 2 is replaced with material 3. The perturbation information is the only data read in again. Upon completion of the calculation, the code execution is ended. Appendix E summarizes the steps necessary to do a reactivity calculation with the REACT code as illustrated by the previous example. Last, Appendix F contains listings of all the source code including the subroutines and the driver program.

CHAPTER 4

NUMERICAL RESULTS

This chapter summarizes the results from calculating reactivities using the REACT computer code and compares them with experimental values. The effect scattering order has on the reactivity calculation is illustrated, and a brief study of code execution times is provided.

4.1 Calculations Using Reactivity Code

Central reactivity worths for a number of different materials have been determined experimentally in Lady Godiva (Engle et al., 1954; Engle et al., 1960; Peterson, 1953). These are compared with values calculated numerically using the reactivity equation derived in Section 2.3 and implemented in a FORTRAN computer code called REACT. Two cross section libraries are used in these calculations: the Hansen-Roach library (Bell et al., 1963; Hansen and Roach, 1969) and the MENDF library (Little, 1987). The Hansen-Roach library is a 16 energy group library containing P_0 cross sections for all its isotopes plus P_1 cross sections for hydrogen and deuterium. Multiple cross sections exist for each of the fissionable isotopes in the library to account for different amounts of resonance self-shielding through the calculation of a potential scattering cross section (Busch and O'Dell, 1991). The MENDF library is a 30 energy group library with a maximum scattering order of P_4. The resonance integrals for the fissionable isotopes are calculated for an infinitely dilute system (the opposite of a metal system). However, this library is suitable for calculations with the Lady Godiva assembly because the energies of the dominant fluxes in Godiva are well above the resonance energies in ^{235}U and ^{238}U ($E_{res} < 100$ keV, Wenz and Busch, 1994).

Figure 6 is a plot comparing the energy group boundaries for the two cross section libraries for groups with energies greater than 0.1 MeV. The MENDF library has roughly

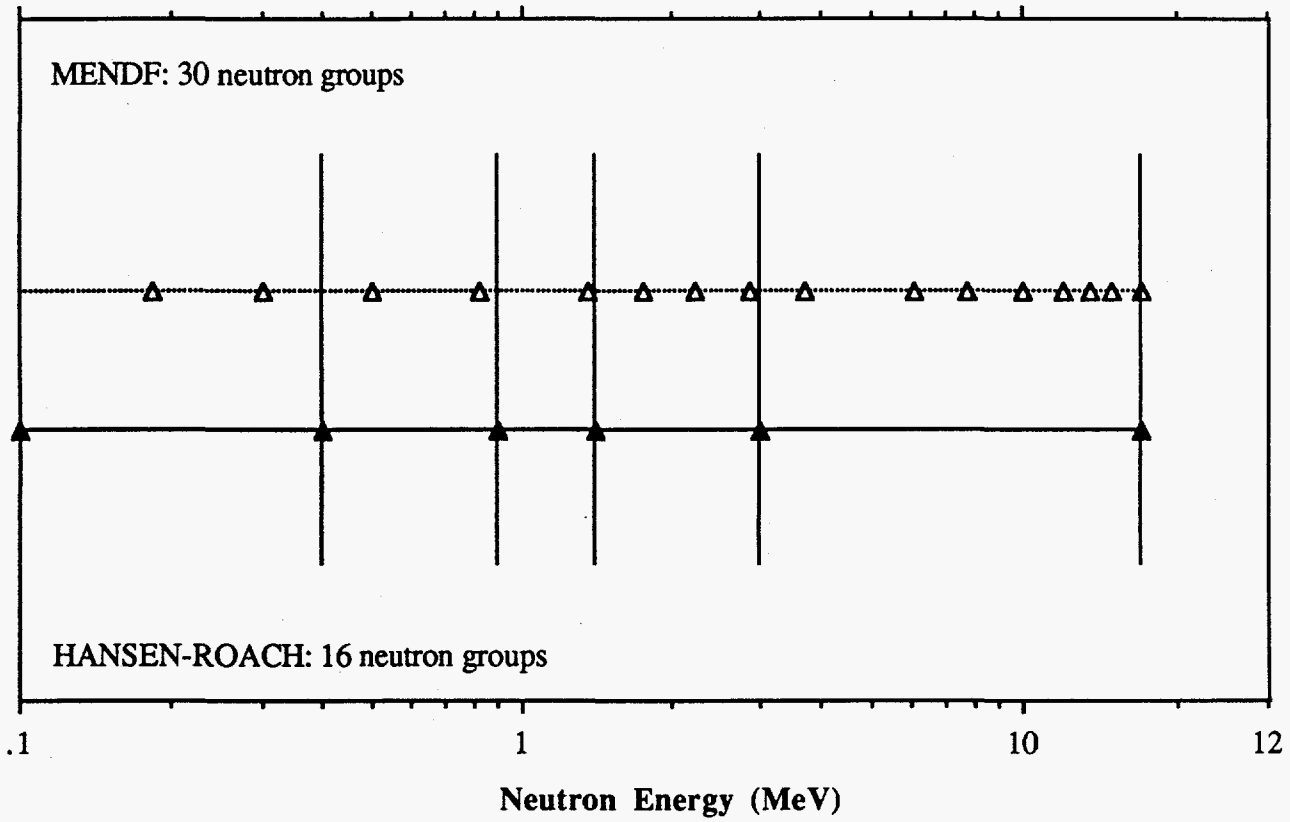

Figure 6. Comparison of energy group structures for the Hansen-Roach (lower data) and the MENDF (upper data) cross section libraries.

three times as many energy groups as the Hansen-Roach library in this energy range. This increased energy definition may account for some of the trends in the upcoming tables.

Column two in Table II shows the results of reactivity calculations using the REACT code with the Hansen-Roach library. The third column of Table II lists the experimentally determined reactivities along with the experimental uncertainty for the cases where that information was available. The last column lists the ratio of the calculated reactivity to the experimental reactivity.

Table II. Central reactivity worths (cents/mole) calculated with the Hansen-Roach library.

Material	REACT (Sn = 16)	Experiment (Ref. 9, 10, 16)	REACT / EXP
H	55.5	47.8	1.16
D	21.7	17.8	1.22
Be	10.0	7.3 ± 1.1	1.37
B	-6.4	-6.9 ± 0.3	0.93
C	4.0	2.4 ± 0.3	1.67
Al	1.2	0.5 ± 0.3	2.40
Fe	0.18	-0.2 ± 0.3	-0.90
Co	0.58	-0.6 ± 0.3	-0.97
Ni	-4.4	-4.4 ± 0.3	1.00
Cu	-0.09	-1.8 ± 0.3	0.05
Cu-OR	-1.4	-1.8 ± 0.3	0.78
Th	1.5	-1.4 ± 0.3	-1.07
^{235}U	137	149	0.92
^{238}U	21.6	24.3	0.89
^{239}Pu	260	285	0.91
^{240}Pu	167	170 ± 17	0.98

Most of the calculated reactivities in Table II are well within a factor of two of the experimental values with the exception of iron, cobalt, and thorium which are also off by the sign of the reactivity change. The reactivity worths for the low atomic mass isotopes are overestimated compared to measured values. This trend may be a result of the isotropic scattering treatment for these forward scattering isotopes. Beryllium has the added effect of (n,2n) reactions that may skew the calculation towards more positive reactivity values, and carbon has some slight resonance activity in the scattering cross section at energies greater than 1 MeV that may account for the high calculated value as well.

Thorium-232 is an example of an isotope with a fast fission threshold cross section. This cross section increases by several orders of magnitude across the four highest energy groups of the Hansen-Roach library, so the limited number of energy groups in this range may be overemphasizing the fission cross sections in some of the energy groups resulting in a more positive reactivity effect compared to the experimental value. However, the experimental reactivity for thorium is found to be negative contrary to what the cross section data suggests and the spontaneous fission and (α,n) rates suggest. Calculations in this thesis and other reports (George and LaBauve, 1988) all indicate a positive reactivity effect as well. The thorium sample is quite small; experimentally, the change in reactivity is on the order of 0.001 dollars using a change in control rod position measurement technique. This measurement method may not be precise enough to measure the small change in reactivity and may indicate the experimental uncertainty is larger than what is reported.

Several of the cross sections on the Hansen-Roach library are marked as having unknown origins indicating they are not directly traceable to either of the original reports (Bell et al., 1963; Hansen and Roach, 1969) documenting the data. Consequently, these cross sections should be used with this fact in mind. The copper cross section used to calculate the first copper reactivity in Table II is the only one that falls into this category. A new version of the Hansen-Roach library (O'Dell, 1994) has been released which contains

data for 167 isotopes. The added cross section data comes from Oak Ridge in a Hansen-Roach format. Performing the same copper calculation with the Oak Ridge copper data resulted in much better agreement with experiment (see Table II, "Cu-OR"). An average calculated reactivity to experiment ratio of 0.78 was found as compared to 0.05. So, inconsistently derived cross section data may also be affecting the reactivity result for this particular isotope.

In general, isotopes with calculated reactivities differing markedly from measured reactivities have cross section issues that are responsible for the discrepancy. The presence of resonances in the dominant flux energy groups (e.g. Al, Fe, Co) or cross section values that are changing by orders of magnitude in a single energy group (i.e. Th) which are not well represented in a finite energy group structure are two examples of such issues. It is interesting to note that the calculations performed by George and LaBauve (1988) using a diffusion based perturbation code with an 80 energy group cross section library also resulted in large discrepancies between calculated and experimental reactivities for iron, aluminum, and thorium, just to name a few. Their calculated reactivity for thorium also had a sign error, and it was the only isotope that had such an error.

Table III contains the same information as Table II for reactivity calculations made with the MENDF library. With this library, the reactivity worths for the low atomic mass isotopes are underestimated compared to the measured values (e.g. H, D, C, and Al). In Table III, the only case in which the calculated sign does not agree with the experimental sign is for thorium-232. The calculated-to-experiment reactivity ratio tends to be closer to 1.0 and fewer sign errors occur for the MENDF library when compared with results from the Hansen-Roach library. Much of this improved precision can be attributed to the greater energy definition in the 30 group library as illustrated in Figure 6.

Table III. Central reactivity worths (cents/mole) calculated with the MENDF library.

Material	REACT (Sn = 16)	Experiment (Ref. 9, 10, 16)	REACT / EXP
H	37.3	47.8	0.78
D	16.1	17.8	0.90
Be	7.9	7.3 ± 1.1	1.08
B	-7.4	-6.9 ± 0.3	1.07
C	1.8	2.4 ± 0.3	0.75
Al	0.06	0.5 ± 0.3	0.12
Fe	-0.66	-0.2 ± 0.3	3.30
Co	-0.55	-0.6 ± 0.3	0.92
Ni	-4.1	-4.4 ± 0.3	0.93
Cu	-1.3	-1.8 ± 0.3	0.72
Th	1.6	-1.4 ± 0.3	-1.14
^{235}U	137	149	0.92
^{238}U	23.1	24.3	0.95
^{239}Pu	265	285	0.93
^{240}Pu	167	170 ± 17	0.98

4.2 Effect of Scattering Order on Reactivity Calculations

Two sets of calculations were run to illustrate the effect cross section scattering order has on the calculated reactivities. The scattering order affects the calculated reactivity through the fluxes and the eigenvalue calculated from ONEDANT since only P_0 scattering data is used in the reactivity calculations. This stipulation also implies a transport corrected total cross section is not used in the reactivity calculation even though one is used in the flux calculation as indicated below.

Flux data were obtained from ONEDANT using the MENDF library for cases where ISCT = 0 and ISCT = 3 with TRCOR = DIAG (a transport correction to the total cross section; O'Dell et al., 1989). Running ONEDANT with ISCT = 3 and TRCOR = DIAG results in essentially the same eigenvalue as if ISCT = 4 were used except less computation time is required. Table IV lists the reactivities calculated with the two flux cases plus the ratio of the two calculations. The P_0 calculations (ISCT = 0) underestimate the reactivity particularly for isotopes where scattering is a significant interaction.

Table IV. Central reactivity worths (cents/mole) calculated with the MENDF library as a function of scattering order used in flux calculation.

Material	ISCT = 0	ISCT = 3	$\frac{\text{ISCT = 3}}{\text{ISCT = 0}}$
H	25.3	36.1	1.43
D	9.3	15.6	1.68
C	0.27	1.7	6.30
^{235}U	126	135	1.07
^{238}U	15.2	22.5	1.48
^{239}Pu	253	263	1.04

Including the effect of anisotropic scattering in the flux calculations has the effect of decreasing the scattering cross section. The spherical harmonics expansion of the scattering term in Equation 2.7a results in Legendre polynomials, $P_L(\Omega' \bullet \Omega)$, that produce negative values for quadrature angles less than 0.5 (which happens for quadrature orders greater than two). The negative polynomials result in a reduction in the scattering cross section as defined by Equation 2.7a. Comparison of the forward and adjoint fluxes calculated with isotropic (ISCT = 0) and anisotropic (ISCT = 3, TRCOR = DIAG) treatments of the scattering term reveal, in general, that the isotropic fluxes are larger in magnitude than the anisotropic fluxes. Also, the effective multiplication factor is larger for

the isotropic case than it is for the anisotropic case. Going from an isotropic to an anisotropic model indicates the neutrons are not being redistributed (moderated) as effectively to energy groups that are more likely to cause more production (fission).

This decrease in moderator effectiveness for anisotropic scattering is expected. The kinematics of an elastic scattering collision between a neutron and a target atom indicates two trends (Duderstadt and Hamilton, 1976): (1) more neutron energy is lost per collision as the atomic mass of the target nucleus decreases and (2) the smaller (more forward) the scattering angle between the scattered neutron and the target nucleus, the less energy that will be transferred from the neutron to the target atom (in other words, a backscatter event causes the greatest energy loss in the neutron). It is primarily the second trend that is affecting the flux distributions for the two scattering models. Isotropic scattering implies there is an equal probability for a neutron to scatter in all directions. Anisotropic scattering says that the scattering is biased in a particular direction. For neutrons in the laboratory system, the bias is in the forward direction; as a result, the effectiveness of a target atom to moderate a neutron is reduced. This is evident in the decreased neutron fluxes and effective multiplication factor for the anisotropic calculation.

Table IV indicates the reactivity for a given material increases for the anisotropic scattering case which is the opposite effect based on the above discussion. Now, we need to look at the parameters that are affected in the reactivity calculation. In Equation 2.34, the cross sections and the volumes remain unchanged, but the fluxes and the eigenvalue are reduced. The reduced flux population increases the effect the denominator has on the reactivity because dividing by a smaller number is equivalent to multiplying by a larger number. The addition of a given material to a system with a smaller neutron population (in this case caused by the anisotropic scattering model) will have an overall larger effect on the system as indicated in Table IV.

4.3 Run Time Comparisons

Several flux calculations were performed with ONEDANT and TWODANT to determine code execution times for some simple and more complex geometries. The results from these tallies help illustrate the usefulness of perturbation codes for reducing computer calculation time. These calculations were run on two machines: (1) a SUN SPARC station IPX, a common machine used for computations at Los Alamos, and (2) a VAX 6000-320, a machine used for computing at the University of New Mexico. Both the Hansen-Roach library and the MENDF library with multiple scatter orders were used in these calculations when possible.

Table V lists the execution times from ONEDANT for computing the fluxes in a one-dimensional sphere of uranium metal. From these calculations, one-dimensional problems require minimal calculational time even for higher order scatter problems, so application of a perturbation code may not be as time effective except, possibly, for problems with large mesh structures.

Table V. Execution times for one-dimensional flux calculations with ONEDANT for a uranium metal sphere.

Calculation Description	Execution Time (sec.) SUN	VAX
40 mesh, 16 energy groups, ISCT = 0	3.7	20.0
40 mesh, 30 energy groups, ISCT = 0	7.6	68.9
40 mesh, 30 energy groups, ISCT = 3	12.8	117

Table VI lists the flux computation times from TWODANT for a two-dimensional uranium metal cylinder on the two computers. It is really with the more complicated problems that the utility of perturbation codes becomes evident. Execution times are now up to an hour for a 40 x 40 mesh calculation with 30 energy groups and P_3 scattering order. For the case where multiple hour runs are required, a perturbation code could

reduce the computing time significantly because only two transport calculations would be required and the execution time of the perturbation code is minuscule in comparison to the transport code run times. Data are not available for the VAX with the 30 group, 40 x 40 mesh problems because of memory limitations, but scaling the known VAX data in Table VI by the ratio of the SUN to VAX execution times gives approximate times on the order of three to ten hours. Here, use of a perturbation code could save days of computation time.

Table VI. Execution times for two-dimensional flux calculations with TWODANT for a uranium metal cylinder.

Calculation Description	Execution Time (min.) SUN	VAX
40x40 mesh, 16 energy groups, ISCT = 0	10.6	74.8
20x20 mesh, 30 energy groups, ISCT = 0	6.3	60.7
40x40 mesh, 30 energy groups, ISCT = 0	21.7	N/A
40x40 mesh, 30 energy groups, ISCT = 3	66.2	N/A

As an example, to calculate 10 reactivities for a given system two transport calculations, one forward and one adjoint for the unperturbed case, would be required for a reactivity calculation using perturbation theory. Using a 40 x 40 mesh with 30 energy groups and isotropic scattering, the execution time on the SUN would be ~45 minutes and ~6 hours on the VAX. Now, if this calculation were done using the $\Delta k/k$ equation, 11 transport calculations, 10 perturbed and one unperturbed, would be required taking ~4 hours on the SUN and ~33 hours on the VAX. This example clearly shows the reduction in execution times possible with perturbation codes. Another factor to consider is the times quoted in Tables V and VI represent the amount of CPU time required for the calculation, not the amount of elapsed time which could be more. If each of the ten reactivity calculations is for a different system then the same number of transport calculations is required for both methods. Based on time considerations, perturbation theory may not be the preferred

calculation method but it definitely would be if all the calculations were being done on the same system.

Finally, a comparison of reactivities calculated using perturbation theory and those calculated using a straight Δk/k calculation where both perturbed and unperturbed effective multiplication factors are used is shown in Table VII for selected isotopes. The second column is the reactivity calculated with perturbation theory using the REACT code, and the third column is the reactivity calculated with the formula Δk/k. In these calculations, the same angular quadrature, meshing, and cross section files were used. The results show there is close agreement between the two calculational methods indicating the small flux perturbation assumption is valid for these isotopes.

Table VII. Comparison of reactivity calculations (cents/mole) using perturbation theory and the Δk/k equation.

Material	REACT	Δk/k	$\frac{\text{REACT}}{\Delta k/k}$
Be	10.0	10.1	0.99
B	-6.4	-6.8	0.94
C	4.0	4.1	0.98
Al	1.2	1.2	1.00
Fe	0.18	0.19	0.95
Co	0.58	0.60	0.97
Ni	-4.4	-4.3	1.02
^{235}U	137	147	0.93
^{238}U	21.6	20.4	1.06

CHAPTER 5
CONCLUSIONS

The reactivity calculations using the REACT computer code based on the equation derived in Chapter 2 agree well with experimental values. Of the two cross section libraries used in these calculations, the MENDF library which has the greater energy definition in the dominant flux groups for the modeled Lady Godiva assembly resulted in reactivities that agreed the best with experiment. Materials that still exhibited large differences between calculation and experiment tend to have cross section data that vary greatly over small energy ranges. These energy ranges are much smaller than the width of the energy group structures in either of the cross section libraries and, consequently, are difficult to model accurately. The resonances in iron and the fast fission threshold in thorium are just two examples of such data. This limited energy resolution is a problem in any code that uses discretized cross section data.

The overall agreement between reactivities calculated using perturbation theory and those calculated using the $\Delta k/k$ formulation is on the order of 10% for materials that do not violate the small perturbation assumptions. Section 4.3 shows how the use of perturbation codes can greatly reduce the amount of time spent performing calculations. A simple two-dimensional 40x40 mesh problem using 30 energy groups and three orders of scattering takes over an hour to run on a SUN SPARC station IPX. Reactivity calculations for 10 materials could take days to run using a $\Delta k/k$ approach to the calculation, where as using a perturbation code could reduce the time to a single afternoon. Even if all the materials will not satisfy the assumptions of the reactivity code, any portion that do could save considerable computation time, particularly as the complexity of the modeled problem increases.

Another feature of reactivity codes in general is their ability to provide a more direct means to determine the effect of slight variations in cross section data on a system. Since

the replacement material is not being homogenized with other components, any change in the calculated reactivity is due directly to the change in the cross section data of the replacement material. Thus, uncertainty analysis in cross section data is well suited for this calculational method along with the fact that many calculations can be run within a short period of time.

One issue that has not been addressed in detail is the range in which perturbation calculations are valid. There is no simple formula to indicate whether a calculation is valid or not. In fact, most papers and text books that deal with perturbation theory are rather vague on this subject. Most authors say that if the resulting reactivity is large, then the calculation should be considered suspect because the small flux perturbation assumption may no longer be valid. McDaniel (1993) placed a value of one dollar of reactivity (in either direction) as being large. This is a reasonable upper limit because a reactor is prompt critical at one dollar above delayed critical. If the reactivity level remains below prompt critical, the neutron population will grow slowly such that it can be controlled well within the time scale of human response times implying a small perturbation has been made to the system. The possibility of maintaining a steady state (or quasi-steady state) flux distribution is much more likely for this situation. If the system exceeds prompt critical the neutron population will grow without bound on a much shorter time scale, and the assumptions used in first order perturbation theory will, no doubt, be violated.

Such a loose criterion for determining situations in which first order perturbation theory is valid puts a great deal of responsibility on the user of this (or any) reactivity code to make sure the results of the calculation are reasonable. Of course, a perturbed flux calculation could be performed and compared with the unperturbed case to verify the flux perturbation is small, but this would somewhat defeat the purpose of using a perturbation code in the first place. However, by simply looking at the system being modeled (flux distributions) and knowing the behavior of the cross sections of the perturbation materials, the user should be able to estimate whether a particular calculation will violate the

assumptions of the code. For instance, placing a small boron sample in a fast reactor will probably not have a large effect on the local flux distribution around the sample. However, if this same measurement were performed in a thermal reactor, like the AGN-201, the change in the local flux distribution about the sample would probably be large enough to invalidate a reactivity calculation done with a first order perturbation code because of the large thermal absorption cross section boron possesses.

One final note is the application of this reactivity code is not strictly limited to unreflected metal systems, the case for which the code was tested. This code could be readily used for reflected metal systems and even thermal systems provided that the boundary conditions for which the reactivity equation was derived are satisfied (i.e. vacuum boundary condition). Reflected systems are readily applicable to this code because the continuity boundary conditions at material interfaces are inherent in the finite differencing scheme used to numerically calculate the fluxes (O'Dell and Alcouffe, 1987; O'Dell et al., 1989). The only major limitation would be the use of isotropic scattering cross sections in the reactivity calculations.

LIST OF REFERENCES

1. J. G. Ahn, N. Z. Cho, J. E. Kuh, "Generation of Spatial Weighting Functions for Ex-Core Detectors by Adjoint Transport Calculation," Nuclear Technology, 103, p.114-121, 1993.

2. R. E. Alcouffe, F. W. Brinkley, D. R. Marr, R. D. O'Dell, "User's Guide for TWODANT: A Code Package for Two-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport," LA-10049-M Rev. 1.8, Los Alamos National Laboratory, April 1992.

3. G. I. Bell, J. J. Devaney, G. E. Hansen, C. B. Mills, W. H. Roach, "Los Alamos Group-Averaged Cross Sections," LAMS-2941, Los Alamos Scientific Laboratory, September 1963.

4. G. I. Bell and S. Glasstone, Nuclear Reactor Theory, Van Nostrand Reinhold, New York, 1970.

5. R. D. Busch and R. D. O'Dell, "Validity of the Hansen-Roach Cross Sections in Low-Enriched Uranium Systems," 1, Paper IV:5, Proceedings of the International Conference on Nuclear Criticality Safety, September 9-13, 1991, Oxford, UK.

6. B. M. Carmichael, "DAC1, a One-Dimensional Sn Perturbation Code," LA-4342, April 1970.

7. V. A. Dean, The Application of Perturbation Theory to a Small Epithermal Reactor, Ph. D Thesis, University of New Mexico, 1988.

8. J. J. Duderstadt and L. J. Hamilton, Nuclear Reactor Analysis, Ch. 5 and Ch. 7, John Wiley, New York, 1976.

9. L. B. Engle, G. E. Hansen, H. C. Paxton, "Material Replacement Measurements in Topsy and Godiva Assemblies," LA-1708, Los Alamos Scientific Laboratory, July 1954.

10. L. B. Engle, G. E. Hansen, H. C. Paxton, "Reactivity Contributions of Various Materials in Topsy, Godiva, and Jezebel," Nuclear Science & Engineering, 8, p. 543-569, 1960.

11. D. M. Etter, Structured Fortran77 for Engineers and Scientists, 2nd Ed., Benjamin-Cummings, Menlo Park, CA, 1987.

12. J. H. Ferziger, Numerical Methods for Engineering Application, Ch. 3, John Wiley & Sons, New York, 1981.

13. B. Friedman, Principles and Techniques of Applied Mathematics, John Wiley and Sons, New York, 1956.

14. D. C. George and R. J. LaBauve, "PERTV - A Standard File Version of the PERT-V Code," LA-11206-MS, Los Alamos National Laboratory, February 1988.

15. G. E. Hansen and C. Maier, "Material Replacement Experiments: Theory and Measurements for the Lady Godiva Assembly," LA-1525, Los Alamos Scientific Laboratory, April 1953.

16. G. E. Hansen and C. Maier, "Perturbation Theory of Reactivity Coefficients for Fast-Neutron Critical Systems," Nuclear Science & Engineering, 8, p. 532-542, 1960.

17. G. E. Hansen and H. C. Paxton, "Reevaluated Critical Specifications of Some Los Alamos Fast-Neutron Systems," LA-4208, Los Alamos Scientific Laboratory, September 1969.

18. G. E. Hansen and W. H. Roach, "Six- and Sixteen-Group Cross Sections for Fast and Intermediate Critical Assemblies," LAMS-2543, Los Alamos Scientific Laboratory, December 1969.

19. R. W. Hardie and W. W. Little, "PERT-V, A Two-dimensional Perturbation Code for Fast Reactor Analysis," BNWL-1162, Pacific Northwest Laboratories, September 1969.

20. A. F. Henry, Nuclear-Reactor Analysis, Ch. 7 and 8, MIT Press, Cambridge, Massachusetts, 1975.

21. G. C. Hopkins, "DAC2, a Two-Dimensional Sn Perturbation Code," LA-4703, July 1971.

22. J. Lewins, Importance: The Adjoint Function, Pergamon Press, New York, 1965.

23. R. C. Little, "Replacement of Public Multi-group Libraries," Los Alamos National Laboratory internal memorandum, X-6:RCL-87-642, December 17, 1987. (MENDF reference)

24. P. McDaniel, ChNE 511 Class Notes on Perturbation Theory, Spring 1993.

25. R. D. O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes, Version IV," LA-6941-MS, Los Alamos Scientific Laboratory, September 1977.

26. R. D. O'Dell and R. E. Alcouffe, "Transport Calculations for Nuclear Analyses: Theory and Guidelines for Effective Use of Transport Codes," LA-10983-MS, Los Alamos National Laboratory, September 1987.

27. R. D. O'Dell , "Memo on 167-Isotope Hansen-Roach Library and Other Things," internal memorandum, January 1994.

28. R. D. O'Dell, F. W. Brinkley, Jr., D. R. Marr, R. E. Alcouffe, "Revised User's Manual for ONEDANT: A Code Package for One-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport," LA-9184-M Rev. 1, Los Alamos National Laboratory, December 1989.

29. R. E. Peterson, "Lady Godiva: An Unreflected Uranium-235 Critical Assembly," LA-1614, Los Alamos Scientific Laboratory, September 1953.

30. E. J. Purcell and D. Varberg, Calculus with Analytic Geometry, 4th Ed., Prentice-Hall, Englewood Cliffs, NJ, 1984.

31. J. J. Tuma, Engineering Mathematics Handbook, 3rd Ed., McGraw-Hill, New York, 1987.

32. L. N. Ussachoff, "Equation for the Importance of Neutrons, Reactor Kinetics and the Theory of Perturbation," Proc. of the International Conference on the Peaceful Uses of Atomic Energy, Geneva, August 8-20 1955, 5, p. 503-510, 1956.

33. T. R. Wenz and R. D. Busch, "Modeling of Central Reactivity Worth Measurements in Lady Godiva," Nuclear Technology, 105, p. 31-36, January 1994.

34. A. M. Weinberg and E. P. Wigner, The Physical Theory of Neutron Chain Reactors, Ch. XVI, University of Chicago Press, Chicago, 1958.

APPENDIX A

Notes on Numerical Integration

By taking the integral of an analytic function over a specified domain, the area underwhich the function is defined is calculated. This area can be calculated numerically using a Riemann sum (Purcell and Varberg, 1984) as given by Equation A.1.

$$R_p = \sum_{i=1}^{n} f(x_{i,ave})\,\Delta x_i \approx \int_a^b f(x)\,dx \qquad (A.1)$$

The Riemann sum simply divides the area under the function f(x) in the domain [a,b] into a finite number of rectangles so that the area can be calculated for each rectangle and summed. In Equation A.1, Δx_i is the width of the rectangle at i and $f(x_{i,ave})$ is the value of the function evaluated at the center of the rectangle. All of these quantities are illustrated in Figure 7. As the width of the rectangle is decreased, the numerical calculation approaches the analytic value.

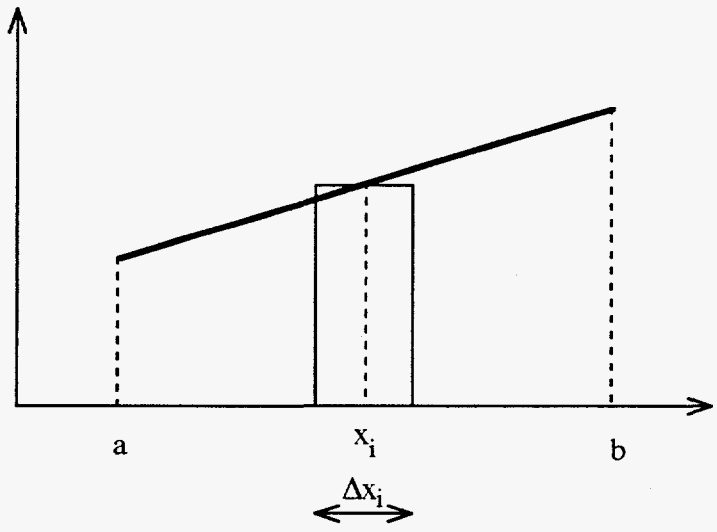

Figure 7. Area representation for numerical integration.

APPENDIX B

Taylor Series Expansion of Perturbed Eigenvalue

The general procedure for determining the Taylor series expansion of a function is discussed here with specific application to the function $1/k^*$ which is used in Equation 2.27 of Chapter 2. The general expression for determining the Taylor series expansion (Purcell and Varberg, 1984; Tuma, 1987) of a function about some point a is:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^n(a)}{n!}(x-a)^n \qquad \text{(B.1)}$$

If n is allowed to go to infinity, the right-hand side exactly equals the left-hand side. The above equation can also be written as f(x) expanded about some small change Δx (Ferziger, 1981; Tuma, 1987). This is the more common form used in deriving finite difference formulas for numerical differentiation and is the form that will be used here.

$$f(x+\Delta x) \approx f(x) + \Delta x\, f'(x) + \frac{\Delta x^2}{2!} f''(x) + \ldots + \frac{\Delta x^n}{n!} f^n(x) \qquad \text{(B.2)}$$

Expanding x about some small Δx as applied to the 1/x function when using the first order approximation of the Taylor series expansion gives:

$$f(x) = \frac{1}{x} \qquad f'(x) = \frac{-1}{x^2}$$

$$f(x+\Delta x) \approx f(x) + \Delta x\, f'(x) \approx \frac{1}{x} - \frac{\Delta x}{x^2} \qquad \text{(B.3)}$$

Changing the above to the k notation yields

$$f(k+\Delta k) \approx f(k) + \Delta k\, f'(k) \qquad \text{(B.4)}$$

where $f(k+\Delta k) = \frac{1}{k^*}$ $\qquad f(k) = \frac{1}{k}$ $\qquad f'(k) = \frac{-1}{k^2}$

Substituting the values into Equation B.4 leads to the following first order approximation which is the form listed in Equation 2.27 of Chapter 2.

$$\frac{1}{k^*} = \frac{1}{k} - \frac{\Delta k}{k^2} \qquad \text{(B.5)}$$

When using the Taylor expansion, the point that is being expanded about is k. In this analysis, k represents the effective multiplication factor for the critical unperturbed reactor, so k is equal to 1, the critical condition. Because of the small change in flux assumption (Equation 2.26), there is a corresponding small change in k. k is therefore bounded roughly between [0.9,1.1] and will not approach zero (where a singularity in the function 1/x exists).

APPENDIX C

Description of Computer Program and Subroutines

The following is a list of the subroutines in the reactivity program REACT along with a description of the functions they perform and the variables that are supplied to the main program. The order in which the subroutines are presented here is in the same order that they are listed in Figure 5.

REACT — This is the main (driver) program that executes all the subroutines that contribute to the reactivity calculation. It also controls whether another calculation is performed (as indicated in Figure 5).

QUAD — This subroutine reads the angular quadrature data from the SNCONS file. It supplies the quadrature weights (w_m), the number of dimensions, and the error flag to the main program.

FAFLX1 — This subroutine reads the 1-D forward angular fluxes located at the mesh point boundaries from the RAFLUX file. The mesh centered angular fluxes, the effective multiplication factor, and the error flag are sent to the main program.

FSFLX1 — This subroutine reads the 1-D mesh centered forward angular fluxes from the RTFLUX file and sends them and the error flag to the main program.

AAFLX1 — This subroutine reads the 1-D adjoint angular fluxes at the mesh boundaries from the AAFLUX file. The mesh centered angular fluxes and the error flag are sent to the main program.

ASFLX1 — This subroutine reads the 1-D mesh centered adjoint angular fluxes from the ATFLUX file and sends them and the error flag to the main program.

GEOM — This subroutine reads in all the coarse mesh geometry data and the material assignment data from the GEODST file. From this information, the fine mesh volumes are calculated in Cartesian, cylindrical, or spherical geometries. The original (unperturbed) material assignments are also determined by fine mesh interval. Last, the fine mesh volumes (V_I), the original material assignments, the number of fine mesh spaces (IT), and the error flag are sent to the main program.

XSECT — This subroutine reads in the macroscopic cross section data from the MACRXS file. This data is based on the ASSIGN=MATLS statement from the ONEDANT input file and includes both the original (unperturbed) and replacement (perturbed) data. Only P0 scattering data is retained. The scattering, total, and fission cross sections, the number of energy groups (NOG), and the error flag are sent to the main program.

PERTDAT — This subroutine reads in the data entered from the keyboard regarding the number of fine mesh spaces undergoing a perturbation, the locations of the fine mesh spaces along with the number of the corresponding replacement material, and the effective delayed neutron fraction. All of this data is supplied to the main program.

CALCRHO — This subroutine calculates the reactivity based on all the information supplied from the previous subroutines and prints the results to the screen.

APPENDIX D

Output from Sample Run Discussed in Chapter 3

The following text is a listing of the input and output generated from the running of the reactivity program REACT. The bold text is the data that was entered by the user from the terminal. Typing the first line starts the execution of the code. The data supplied to the program was based on the ONEDANT input file shown in Figure 2. The seven mesh points undergoing a perturbation are the total number of fine meshes that make up the first coarse mesh interval in the input and correspond to the locations of the replacement sample. Material 2 is the ^{235}U ("235-yr") sample, and material 3 is the ^{238}U ("u238y") sample, per Figure 2.

critical<194>**xpert**

^-*^

Enter the number of fine mesh spaces that are undergoing
a perturbation.

7

Now enter the cell number for each fine mesh that will
be perturbed followed by the replacement material number
(Press return after each entry). eg. "12 3"

1 2
2 2
3 2
4 2
5 2
6 2
7 2

Enter the effective delayed neutron fraction, B-eff,
for the system. (Enter zero if it is unknown.)

0.0065

*** that's all the input ***

****

The reactivity due to the perturbation is:

0.11389E-02 (reactivity fraction)

0.17522E+00 (reactivity in dollars)

Beff = 0.006500

****

Calculate another reactivity using the same flux and cross section data? (y/n)
y

^-*^

Enter the number of fine mesh spaces that are undergoing
a perturbation.

7

Now enter the cell number for each fine mesh that will
be perturbed followed by the replacement material number
(Press return after each entry). eg. "12 3"

1 3
2 3
3 3
4 3
5 3
6 3
7 3

Enter the effective delayed neutron fraction, B-eff,
for the system. (Enter zero if it is unknown.)

0.0065

*** that's all the input ***

****

The reactivity due to the perturbation is:

0.17782E-03 (reactivity fraction)

0.27356E-01 (reactivity in dollars)

Beff = 0.006500

****

Calculate another reactivity using the same flux and cross section data? (y/n)
n

critical<195>

APPENDIX E

Checklist for Running REACT Code

1. Model the unperturbed assembly with ONEDANT and create an input file.

   Select the geometry (Cartesian, spherical, or cylindrical) that will best model the assembly. Make any simplifying assumptions to the model that are necessary to accommodate the chosen coordinate system.

   Define the unperturbed materials (those materials that make the structure of the assembly: fuel, reflectors, etc.) in the MATLS statement in Block IV of the input file. These materials may or may not change from one perturbation calculation to the next and should always be included in the input file.

   Define the replacement materials (these correspond to the replacement sample materials) in the MATLS statement in Block IV. These materials may change for each set of reactivity calculations run.

   Use the ASSIGN = MATLS form of the material assignment statement in Block IV.

2. Select an appropriate cross section library for the calculations.

3. Run ONEDANT in the forward calculational mode (ITH = 0 in Block V).
   Store the following binary files from this run in another directory so they will not be over written:

   GEODST
   MACRXS
   RAFLUX
   RTFLUX
   SNCONS

4. Run ONEDANT in the adjoint calculational mode (ITH = 1 in Block V).
   Store the following binary files from this run in the same directory as the other binary files:

   AAFLUX
   ATFLUX

5. Setup symbolic links or do whatever is required so the REACT code can access the binary files.

6. Run the REACT code by typing the name of the executable file. Be prepared to enter the following data:

   - Total number of fine mesh spaces undergoing a perturbation
   - The particular fine mesh number of a mesh undergoing a perturbation and the material number of the replacement material being put in this location.
   - The effective delayed neutron fraction. If this value is not known, simply enter a zero and the unit conversion will not be performed.

7. If more replacement materials need to be run with the existing flux data, just add the statement NOSOLV=1 to Block I and make the necessary additions, deletions, or changes to the replacement materials in Block IV. Leave the unperturbed materials as they are.

   Now rerun the input file in the forward calculation mode only (ITH=0 in Block V). Replace the old MACRXS and GEODST binary files with the ones just created.

   Run the REACT code with the new replacement material data.

APPENDIX F

Listings of REACT Subroutines and Driver Program

```
      subroutine aaflx1(afadji,iflag)
c
c     This program reads the binary file AAFLUX produced from
c     ONEDANT in order to obtain the adjoint angular fluxes.
c
c     ***Note: the fluxes are read in such that position 1 in the flux
c        array corresponds to group 1.
c
c     File format obtained from LA-6941-MS, "Standard Interface Files
c     and Procedures for Reactor Physics Codes, Version IV", RDO.
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
        real afadji(lim,lim,lim)
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=11,file='aaflux',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   read 'file identification'
c-----------------------------------------------------------------------
c
        read(11) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read 'specifications'
c-----------------------------------------------------------------------
c
        read(11) ndim,ngroup,ninti,nintj,nintk,ndir,effk,adum
        nbdryi = ninti + 1
        nbdryj = nintj + 1
        nbdryk = nintk + 1
        if (ndim.ge.3) then
          write(*,500)
           iflag = 999
           go to 53
        endif
c
c-----------------------------------------------------------------------
c   read angular fluxes at first dimension boundaries.  Note that the
c   energy groups and angular directions have to be read in reverse
c   order (as indicated in LA-6941-MS) for them to correspond with the
c   group and directions for the forward calculation and with
c   the adjoint values printed out in the ODNOUT.DAT file.
c-----------------------------------------------------------------------
c
        do 50 l=ngroup,1,-1
          read(11) ((afadji(l,m,i), m=ndir,1,-1), i=1,nbdryi)
 50     continue
c
c     (the remaining data in the file is not needed.)
c     calculate angular fluxes at the center of the mesh
c
        do 52 l=1,ngroup
           do 52 m=1,ndir
              do 52 i=1,ninti
                 afadji(l,m,i) = (afadji(l,m,i) +
     &                            afadji(l,m,i+1))/2.0
 52     continue
c
c-----------------------------------------------------------------------
c
 500    format(//,
     &          'ADJFLX1: **** error reading raflux/aaflux file ****',
     &           8x,'the number of dimensions is greater than 1',//)
c-----------------------------------------------------------------------
c
 53     close(unit=11)
        return
        end
```

```
      subroutine asflx1(fadj,iflag)
c
c     This program reads the binary file ATFLUX produced from
c     ONEDANT in order to obtain the adjoint scalar fluxes.
c
c     ***Note: the fluxes are read in such that position 1 in the flux
c        array corresponds to group 1.
c
c     File format obtained from LA-6941-MS, "Standard Interface Files
c     and Procedures for Reactor Physics Codes, Version IV", RDO.
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
        double precision fadj(lim,lim)
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=11,file='atflux',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   read 'file identification'
c-----------------------------------------------------------------------
c
        read(11) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read 'specifications'
c-----------------------------------------------------------------------
c
        read(11) ndim,ngroup,ninti,nintj,nintk,iter,effk,adum,nblok
        if (ndim.ge.2) then
          write(*,500)
          iflag = 999
          go to 53
        endif
c
c-----------------------------------------------------------------------
c   Note that the fluxes are read in reverse order (as indicated in
c   LA-6941-MS) in the energy variable so that the fluxes for group 1
c   are in the first position which corresponds with the ordering
c   the adjoint values are printed in the ONDOUT.DAT file.
c-----------------------------------------------------------------------
c
        do 50 m=1,nblok
          jl = (m-1)*((ngroup-1)/nblok + 1) + 1
          jup = m*((ngroup-1)/nblok + 1)
          ju = min0(ngroup,jup)
          read(11) ((fadj(j,i), i=1,ninti), j=ju,jl,-1)
 50     continue
c
c-----------------------------------------------------------------------
c
 500    format(//,
     &         'ASFLX1: **** error reading atflux file ****',
     &         8x,'the number of dimensions is greater than 1',//)
c-----------------------------------------------------------------------
c
 53     close(unit=11)
        return
        end
```

```
      subroutine calcrho(pertmat,beff,vol,origmat,wm,afadj,
     &            afreg,effk,fmat,mmat,npert,ndir,ninti,nintj,
     &            ngroup,sfreg,sfadj,qmat)
c
c       Note: * multiple sample perturbations can be run provided
c             the same flux, geometry and cross section data are
c             used (i.e. no new data can be read in).
c
c
c       ?????????????????????? Definitions ??????????????????????
c          direta - direction cosine in the eta direction
c          dirmu - direction cosine in the mu direction
c         dirwgt - quadrature weight
c          ndim - number of dimensions
c          ndir - number of quadrature directions
c       ????????????????????????????????????????????????????????????
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100, pi=3.141592654)
        integer origmat(lim), pertmat(lim)
        double precision fmat(lim,lim,lim), mmat(lim,lim,lim),
     &       rho1, xnum, denom, dfmat(lim,lim,lim), qmat(lim,lim),
     &       dmmat(lim,lim,lim), rho2, dqmat(lim,lim),
     &       sfreg(lim,lim), sfadj(lim,lim)
        real wm(lim), beff, effk, afreg(lim,lim,lim),
     &       afadj(lim,lim,lim),vol(lim)
c
c-----------------------------------------------------------------------
c   initialize arrays to zero
c-----------------------------------------------------------------------
c
        do 70 i=1,lim
        do 70 j=1,lim
           dqmat(i,j) = 0.0
        do 70 k=1,lim
           dfmat(i,j,k) = 0.0
           dmmat(i,j,k) = 0.0
 70     continue
c
c-----------------------------------------------------------------------
c   calculate the delta-M and delta-F values, the change in cross
c   section due to the perturbation
c-----------------------------------------------------------------------
c
        do 72 i=1,ninti*nintj
           if (pertmat(i).ne.-1) then
              if (origmat(i).eq.pertmat(i)) write(*,720)i
              do 74 j=1,ngroup
              do 74 k=1,ngroup
                 if (pertmat(i).eq.0) then
                    dmmat(j,k,i) = -mmat(j,k,origmat(i))
                    dfmat(j,k,i) = -fmat(j,k,origmat(i))
                    dqmat(j,i) = -qmat(j,origmat(i))
                 elseif (origmat(i).eq.0) then
                    dmmat(j,k,i) = mmat(j,k,pertmat(i))
                    dfmat(j,k,i) = fmat(j,k,pertmat(i))
                    dqmat(j,i) = qmat(j,pertmat(i))
                 else
                    dmmat(j,k,i) = mmat(j,k,pertmat(i)) -
     &                             mmat(j,k,origmat(i))
                    dfmat(j,k,i) = fmat(j,k,pertmat(i)) -
     &                             fmat(j,k,origmat(i))
                    dqmat(j,i) = qmat(j,pertmat(i)) -
     &                           qmat(j,origmat(i))
                 endif
 74           continue
           endif
 72     continue
c
c-----------------------------------------------------------------------
c   calculate the numerator and the denominator for the reactivity
c   equation
c-----------------------------------------------------------------------
c
        xnum = 0.0
        denom = 0.0
        do 76 l=1,ninti*nintj
        do 76 i=1,ngroup
        do 77 j=1,ngroup
c   sum scalar flux terms
           xnum = xnum +
     &          (vol(l)*(dmmat(i,j,l)*sfreg(j,l)*sfadj(i,l)))
     &        + (vol(l)/effk)*(dfmat(i,j,l)*sfreg(j,l)*sfadj(i,l))
c
           denom = denom +
     &             vol(l)*(fmat(j,i,origmat(l)) +
     &              dfmat(j,i,l))*sfreg(i,l)*sfadj(j,l)
 77     continue
        do 78 k=1,ndir
c   sum angular flux terms
           xnum = xnum -
     &          vol(l)*dqmat(i,l)*afreg(i,k,l)*afadj(i,k,l)*wm(k)
 78     continue
 76     continue
c
c-----------------------------------------------------------------------
c   calculate rho, units of fraction reactivity
c-----------------------------------------------------------------------
c
        rho1 = xnum/denom
c
c-----------------------------------------------------------------------
c   calculate rho, units of dollars
c-----------------------------------------------------------------------
c
        if (beff.ne.0.0) rho2 = xnum/denom/beff
c
c-----------------------------------------------------------------------
c   print results out to screen
c-----------------------------------------------------------------------
c
        write(*,700)
        write(*,705)
        if (beff.eq.0.0) then
           write(*,710)rho1
        else
           write(*,715)rho1,rho2,beff
        endif
        write(*,705)
        write(*,700)
c
c-----------------------------------------------------------------------
c
 700    format(///)
 705    format(/,44('*'))
 710    format(/,'The reactivity due to the perturbation is:',//,
     &         5x,e12.5,3x,'(reactivity fraction)',//)
 715    format(/,'The reactivity due to the perturbation is:',//,
```

```
     &            5x,e12.5,3x,'(reactivity fraction)',//,
     &             5x,e12.5,3x,'(reactivity in dollars)',//,
     &             20x,'Beff = ',f8.6)
 720       format(/,'***Warning: the PERTMAT equals ORIGMAT at fine',
     &           ' mesh ',i4)
c
c-----------------------------------------------------------------------
        return
        end
```

```
      subroutine faflx1(afregi,effk,iflag)
c
c     This program reads the binary files RAFLUX produced from
c     ONEDANT in order to obtain the 1-D forward angular fluxes.
c
c     File format obtained from LA-6941-MS, "Standard Interface Files
c     and Procedures for Reactor Physics Codes, Version IV", RDO.
c
c     ?????????????????????? Definitions ??????????????????????
c        afregi - array that holds boundary angular fluxes
c         ndim - number of dimensions
c         ndir - number of quadrature directions
c     ????????????????????????????????????????????????????????
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
        real afregi(lim,lim,lim), effk
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=12,file='raflux',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   read file identification
c-----------------------------------------------------------------------
c
        read(12) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read specifications
c-----------------------------------------------------------------------
c
        read(12) ndim,ngroup,ninti,nintj,nintk,ndir,effk,power
        nbdryi = ninti + 1
        if (ndim.ge.2) then
          write(*,900)
          iflag = 999
          go to 91
        endif
c
c-----------------------------------------------------------------------
c   check to make sure arrays are dimensioned large enough to handle
c   "ngroup" energy groups
c-----------------------------------------------------------------------
c
        if (ngroup.gt.lim) then
          write(*,910)ngroup,lim
          iflag = 999
          go to 91
        endif
c
c-----------------------------------------------------------------------
c   check to make sure arrays are dimensioned large enough to handle
c   "nbdryi" mesh points
c-----------------------------------------------------------------------
c
        if (nbdryi.gt.lim) then
          write(*,920)nbdryi,lim
          iflag = 999
          go to 91
        endif
c
c-----------------------------------------------------------------------
c   read angular fluxes at first dimension boundaries
c-----------------------------------------------------------------------
c
        do 90 l=1,ngroup
          read(12) ((afregi(l,m,i), m=1,ndir), i=1,nbdryi)
 90     continue
c
c   (the remaining data in the file is not needed)
c   calculate angular fluxes at mesh center
c
        do 92 l=1,ngroup
          do 92 m=1,ndir
            do 92 i=1,ninti
              afregi(l,m,i) = (afregi(l,m,i) + afregi(l,m,i+1))/2.0
 92     continue
c
c-----------------------------------------------------------------------
c
 900    format(//,'FORFLX1:*** error reading raflux/aaflux file ***',
     &          8x,'the number of dimensions is greater than 1',//)
 910    format(//,'FORFLX1:**** arrays undersized for number of ',
     &         'energy groups',//,14x,
     &         'NGROUP = ',i5,4x,'LIM = ',i5,//)
 920    format(//,'FORFLX1:**** arrays undersized for number of ',
     &         'mesh points',//,14x,
     &         'Number of boundry points = ',i5,4x,'LIM = ',i5,//)
c-----------------------------------------------------------------------
c
 91     close(unit=12)
        return
        end
```

```
      subroutine fsflx1(freg,iflag)
c
c     This program reads the binary files RTFLUX produced from
c     ONEDANT in order to obtain the 1-D forward scalar fluxes.
c
c     File format obtained from LA-6941-MS, "Standard Interface Files
c     and Procedures for Reactor Physics Codes, Version IV", RDO.
c
c     ?????????????????? Definitions ??????????????????
c       freg - array that holds fine mesh scalar fluxes
c       ndim - number of dimensions
c       ndir - number of quadrature directions
c     ????????????????????????????????????????????????????
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
       real effk
       double precision freg(lim,lim)
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=12,file='rtflux',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   read 'file identification'
c-----------------------------------------------------------------------
c
        read(12) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read 'specifications'
c-----------------------------------------------------------------------
c
        read(12) ndim,ngroup,ninti,nintj,nintk,iter,effk,power,nblok
        if (ndim.gt.2) then
          write(*,900)
           iflag = 999
           go to 91
        endif
c
c-----------------------------------------------------------------------
c   check to make sure arrays are dimensioned large enough to handle
c   "ngroup" energy groups
c-----------------------------------------------------------------------
c
        if (ngroup.gt.lim) then
           write(*,910)ngroup,lim
           iflag = 999
           go to 91
        endif
c
c-----------------------------------------------------------------------
c   read 'one dimensional regular total flux'
c-----------------------------------------------------------------------
c
        do 90 m=1,nblok
           jl = (m-1)*((ngroup-1)/nblok +1) + 1
           jup = m*((ngroup-1)/nblok + 1)
           ju = min0(ngroup,jup)
          read(12) ((freg(j,i), i=1,ninti), j=jl,ju)
 90     continue
c
c-----------------------------------------------------------------------
c
 900    format(//,'FSFLX1:*** error reading rtflux file ***',
     &          8x,'the number of dimensions is greater than 1',//)
 910    format(//,'FSFLX1:**** arrays undersized for number of ',
     &          'energy groups',//,14x,
     &          'NGROUP = ',i5,4x,'LIM = ',i5,//)
c-----------------------------------------------------------------------
c
 91     close(unit=12)
        return
        end
```

```
      subroutine geom(vol,origmat,ninti,nintj,iflag)
c
c       This program reads the binary file GEODST produced from either
c       ONEDANT or TWODANT in order to calculate cell volumes.  The cell
c       volumes are based on the fine mesh intervals.  The zone numbers
c       for each region are read in and converted to zone numbers for
c       each fine mesh.  Since the ASSIGN=MATLS statement is used in the
c       code, the zone numbers relate directly to material numbers and
c       thus the original material assignments.
c
c       File format obtained from LA-6941-MS, "Standard Interface Files
c       and Procedures for Reactor Physics Codes, Version IV", RDO.
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100,pi=3.141592654)
        integer ngop(lim),ifints(lim),jfints(lim),
     &          nznr(lim),origmat(lim),mr(lim,lim)
        double precision xmesh(lim),ymesh(lim)
        real junk(lim),vol(lim)
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=11,file='geodst',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c  read file identification
c-----------------------------------------------------------------------
c
        read(11) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c  read specifications
c-----------------------------------------------------------------------
c
        read(11)igom,nzone,nreg,nzcl,ncinti, ncintj,ncintk,ninti,
     &          nintj,nintk,imbl,imb2,jmbl,jmb2,kmbl,kmb2,nbs,nbcs,
     &          nibcs,nzwbb,ntriag,nrass,nihpt,(ngop(i),i=1,4)
        ncbndi = ncinti + 1
        ncbndj = ncintj + 1
c
c-----------------------------------------------------------------------
c  read one-dimensional coarse mesh interval boundaries and
c  calculate fine mesh volumes
c-----------------------------------------------------------------------
c
        if ((igom.gt.0).and.(igom.le.3)) then
           read(11)(xmesh(i), i=1,ncbndi), (ifints(i), i=1,ncinti)
           n = 0
           do 10 i=1,ncinti
              dx = (xmesh(i+1) - xmesh(i))/ifints(i)
              do 12 it=1,ifints(i)
                 n = n + 1
                 if (igom.eq.1) vol(n) = dx
                 if (igom.eq.2) vol(n) = pi*((it*dx + xmesh(i))**2
     &              - ((it-1)*dx + xmesh(i))**2)
                 if (igom.eq.3) vol(n) = ((it*dx + xmesh(i))**3
     &              - ((it-1)*dx + xmesh(i))**3)*pi*4.0/3.0
  12          continue
  10       continue
        endif
c
c-----------------------------------------------------------------------
c  read two-dimensional coarse mesh interval boundaries and
c  calculate fine mesh volumes
c-----------------------------------------------------------------------
c
        if ((igom.ge.6).and.(igom.le.11)) then
           read(11) (xmesh(i), i=1,ncbndi), (ymesh(j), j=1,ncbndj),
     &              (ifints(i),i=1,ncinti), (jfints(j),j=1,ncintj)
           n = 0
           do 20 j=1,ncintj
              dy = (ymesh(j+1) - ymesh(j))/jfints(j)
              do 20 jt=1,jfints(j)
                 do 20 i=1,ncinti
                    dx = (xmesh(i+1) - xmesh(i))/ifints(i)
                    do 20 it=1,ifints(i)
                       n = n + 1
                       if (igom.eq.6) vol(n) = dx*dy
                       if (igom.eq.7) vol(n) = pi*dy*((it*dx +
     &                    xmesh(i))**2 - ((it-1)*dx + xmesh(i))**2)
                       if ((igom.gt.7).and.(igom.le.11)) then
                          write(*,200)igom
                          iflag = 999
                          go to 29
                       endif
  20          continue
        endif
c
c-----------------------------------------------------------------------
c  three-dimensional coarse mesh interval boundaries (skipped)
c-----------------------------------------------------------------------
c
        if (igom.ge.12) then
           write(*,200)igom
           iflag = 999
           go to 29
        endif
c
c-----------------------------------------------------------------------
c  read geometry data (volr,bsq,bndc,bnci,nzhbb,nzc,nznr).  some data,
c  which is not needed, is read into dummy varialbes to minimize array
c  requirements.
c-----------------------------------------------------------------------
c
        if ((igom.gt.0).or.(nbs.gt.0)) then
           read(11)(junk(n),n=1,nreg),(junk(n),n=1,nbs),
     &             (junk(n),n=1,nbcs), (junk(n),n=1,nibcs),
     &             (ngop(n),n=1,nzwbb), (ngop(n),n=1,nzone),
     &             (nznr(n),n=1,nreg)
        endif
c
c-----------------------------------------------------------------------
c       read region assignments to coarse mesh intervals
c-----------------------------------------------------------------------
c
        if((igom.gt.0).and.(nrass.eq.0)) then
          read(11) ((mr(i,j),i=1,ncinti), j=1,ncintj)
          m = 0
          if (ncintj.eq.1) then
            do 22 i=1,ncinti
            do 22 ii=1,ifints(i)
               m = m + 1
               origmat(m) = nznr(mr(i,ncintj))
  22        continue
          else
            do 23 j=1,ncintj
            do 23 jj=1,jfints(j)
            do 23 i=1,ncinti
```

```
               do 23 ii=1,ifints(i)
                  m = m + 1
                  origmat(m) = nznr(mr(i,j))
  23           continue
             endif
          endif
c
c-----------------------------------------------------------------------
c        read region assignments to fine mesh intervals
c-----------------------------------------------------------------------
c
          if ((igom.gt.0).and.(nrass.eq.1)) then
             read(11) ((mr(i,j),i=1,ninti),j=1,nintj)
             m = 0
             do 24 j=1,nintj
             do 24 i=1,ninti
                m = m + 1
                origmat(m) = nznr(mr(i,j))
  24         continue
          endif
c
c-----------------------------------------------------------------------
c
  200     format(//,'GEOM:**** not setup to handle specified',
     &          ' geometry: IGOM = ',i2,' ***',//,
     &          '*** code terminated ***',/)
c-----------------------------------------------------------------------
c
  29      close(unit=11)
          return
          end
```

```
      subroutine pertdat(nmat,pertmat,beff,npert)
c
c     This subroutine reads which mesh spaces will be perturbed and
c     the number of the replacement material based on the order in the
c     "ASSIGN=" section of the ONEDANT/TWODANT input.
c
c     ?????????????????? Subroutine Definitions ??????????????????
c     matnum - replacement material number for fine mesh "nmesh"
c      nmat - total number of materials
c     nmesh - number of perturbed fine mesh interval
c     ??????????????????????????????????????????????????????????????????
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
        integer pertmat(lim)
c
c-----------------------------------------------------------------------
c   initialize array to no perturbation status,ie. -1
c-----------------------------------------------------------------------
c
        do 60 k=1,lim
           pertmat(k) = -1
 60     continue
c
c-----------------------------------------------------------------------
c   read in perturbation information from the terminal
c-----------------------------------------------------------------------
c
        write(*,600)
        write(*,605)
        read(*,*)npert
        write(*,610)
        do 62 k=1,npert
 65        read(*,*)nmesh,matnum
           if (matnum.gt.nmat) then
              write(*,640)matnum
              go to 65
           endif
           pertmat(nmesh) = matnum
 62     continue
        write(*,620)
        read(*,*)beff
        write(*,630)
c
c-----------------------------------------------------------------------
c
 600    format(/////,24('^-','*'),'^')
 605    format(/,'Enter the number of fine mesh spaces that are',
     &         ' undergoing',/,'  a perturbation.',/)
 610    format(/,'Now enter the cell number for each fine mesh',
     &         ' that will',/,'  be perturbed followed by the',
     &         ' replacement material number',/,
     &         '  (Press return after each entry). eg. "12 3"',/)
 620    format(/,'Enter the effective delayed neutron fraction,',
     &          ' B-eff,',/,
     &         '  for the system. (Enter zero if it is unknown.)',/)
 630    format(//,8x,'*** that''s all the input ***',//)
 640    format(5x,'*** material ',i3,' is not a valid material ***',
     &         /,9x,'Re-enter the pair.')
c-----------------------------------------------------------------------
c
        return
        end
```

```
      subroutine quad(dirwgt,ndim,ndir,iflag)
c
c     This program reads the binary file SNCONS produced from either
c     ONEDANT or TWODANT in order to obtain the quadrature weights.
c
c     File format obtained from LA-6941-MS, "Standard Interface Files
c     and Procedures for Reactor Physics Codes, Version IV", RDO.
c
c     ?????????????????????? Definitions ??????????????????????
c        direta - direction cosine in the eta direction
c        dirmu - direction cosine in the mu direction
c        dirwgt - quadrature weight
c        ndim - number of dimensions
c        ndir - number of quadrature directions
c     ????????????????????????????????????????????????????????????
c
        implicit integer (i,j,k,l,m,n)
        parameter (lim=100)
        real dirwgt(lim), dirmu(lim)
        character*6 hname, huse(2)
c
c-----------------------------------------------------------------------
        open(unit=11,file='sncons',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   read file identification
c-----------------------------------------------------------------------
c
        read(11) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read specifications
c-----------------------------------------------------------------------
c
        read(11) ndim,ndir,idum,idum
        if (ndim.gt.2) then
           write(*,350)
           iflag = 999
           go to 33
        endif
c
c-----------------------------------------------------------------------
c   check to make sure arrays are dimensioned large enough to handle
c   "ndir" directions
c-----------------------------------------------------------------------
c
        if (ndir.gt.lim) then
           write(*,360)ndir,lim
           iflag = 999
           go to 33
        endif
c
c-----------------------------------------------------------------------
c   read one-dimensional Sn constants
c-----------------------------------------------------------------------
c
        if (ndim.eq.1) then
           read(11) (dirwgt(i), i=1,ndir), (dirmu(i), i=1,ndir)
        endif
c
c-----------------------------------------------------------------------
c   read multi-dimensional Sn constants: dirwgt(i), dirmu(i),
c   direta(i).  Note dummy variables used for variables that are
c   not needed in the code.
c-----------------------------------------------------------------------
c
        if (ndim.ge.2) then
           read(11) (dirwgt(i), i=1,ndir),(dirmu(i),i=1,ndir),
     &              (dirmu(i), i=1,ndir)
        endif
c
c-----------------------------------------------------------------------
c
 350    format(//,'QUAD:**** error reading raflux/aaflux file ****',
     &        8x,'the number of dimensions is greater than 2',//)
 360    format(//,'QUAD:**** arrays undersized for number of ',
     &        'angular directions',//,5x,'NDIR = ',i5,
     &        4x,'LIM = ',i5,//)
c
c-----------------------------------------------------------------------
c
 33     close(unit=11)
      return
      end
```

```
      program react
c
c     Driver program that calculates the reactivity for a given system
c     using first order perturbation theory.
c
c     ?????????????????????? Subroutine Definitions ??????????????????????
c     aaflx1 - sub that reads in 1D adjoint angular fluxes
c     aaflx2 - sub that reads in 2D adjoint angular fluxes
c     asflx1 - sub that reads in 1D adjoint scalar fluxes
c     asflx2 - sub that reads in 2D adjoint scalar fluxes
c     calcrho - sub that calculates the reactivity and prints out
c        the results.
c     faflx1 - sub that reads in 1D forward angular fluxes
c     faflx2 - sub that reads in 2D forward angular fluxes
c     fsflx1 - sub that reads in 1D forward scalar fluxes
c     fsflx2 - sub that reads in 2D forward scalar fluxes
c     geom - sub that calculates fine mesh volumes and reads in
c       original material assignments
c     pertdat - sub that reads in perturbation information from terminal
c     quad - sub that reads in quadrature weights
c     xsect - sub that reads in cross section data and puts it in the
c        form of the F and M matricies
c     ?????????????????????? Variable Definitions ??????????????????????
c     afadj - array holding the adjoint angular flux
c     afreg - array holding the forward angular flux
c     beff - delayed neutron fraction for the system
c     effk - keff from forward, unperturbed calculation
c     fmat - matrix storing fission cross section data
c     iflag - variable indicating pgm needs to be terminated because
c       arrays are underdimensioned or some other logistic error.
c     mmat - matrix storing scattering cross section data
c     nmat - total number of materials specified in problem
c     qmat - matrix storing total cross section data
c     ndim - number of geometric dimensions
c     ndir - number of angular dimensions (related to n in Sn)
c     ngroup - number of neutron energy groups used in calculation
c     ninti - number of fine mesh intervals in x direction
c     nintj - number of fine mesh intervals in y direction
c     npert - number of fine mesh intervals with perturbations
c     origmat - array storing original material assignments to each
c       fine mesh interval.
c     pertmat - array storing perturbation material assignments for
c       each fine mesh interval.
c     qmat - matrix storing total cross section data
c     sfadj - array holding the adjoint scalar flux
c     sfreg - array holding the forward scalar flux
c     vol - array storing fine mesh volumes
c     wm - array storing qudrature weight data
c     ??????????????????????????????????????????????????????????????????????
c
      implicit integer (i,j,k,l,m,n)
      parameter (lim=100)
      integer origmat(lim), pertmat(lim)
      double precision fmat(lim,lim,lim), mmat(lim,lim,lim),
     &     qmat(lim,lim),sfadj(lim,lim),sfreg(lim,lim)
      real wm(lim),beff,effk,afadj(lim,lim,lim),afreg(lim,lim,lim),
     &     vol(lim)
      character*8 again
c
c-----------------------------------------------------------------------
      iflag = 0
c
c-----------------------------------------------------------------------
c   read in quadrature weights
c-----------------------------------------------------------------------
c
      call quad(wm,ndim,ndir,iflag)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in forward angular fluxes
c-----------------------------------------------------------------------
c
      if (ndim.eq.1) call faflx1(afreg,effk,iflag)
      if (ndim.eq.2) call faflx2(afreg,effk,iflag)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in forward scalar fluxes
c-----------------------------------------------------------------------
c
      if (ndim.eq.1) call fsflx1(sfreg,iflag)
c     if (ndim.eq.2) call fsflx2(sfreg,iflag)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in adjoint angular fluxes
c-----------------------------------------------------------------------
c
      if (ndim.eq.1) call aaflx1(afadj,iflag)
      if (ndim.eq.2) call aaflx2(afadj)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in adjoint scalar fluxes
c-----------------------------------------------------------------------
c
      if (ndim.eq.1) call asflx1(sfadj,iflag)
c     if (ndim.eq.2) call asflx2(sfadj)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   calculate fine mesh cell volumes and read original
c   material assignments
c-----------------------------------------------------------------------
c
      call geom(vol,origmat,ninti,nintj,iflag)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in cross section data
c-----------------------------------------------------------------------
c
      call xsect(fmat,mmat,qmat,ngroup,nmat,iflag)
      if (iflag.eq.999) go to 999
c
c-----------------------------------------------------------------------
c   read in the perturbation data
c-----------------------------------------------------------------------
c
 888  call pertdat(nmat,pertmat,beff,npert)
c
c-----------------------------------------------------------------------
c   calculate the system reactivity
c-----------------------------------------------------------------------
c
      call calcrho(pertmat,beff,vol,origmat,wm,afadj,afreg,effk,
     &             fmat,mmat,npert,ndir,ninti,nintj,ngroup,sfreg,
```

```
     &                sfadj,qmat)
        write(*,905)
        read(*,*)again
        if ((again.eq.'Y').or.(again.eq.'y')) go to 888
        write(*,915)
c
c-----------------------------------------------------------------------
c
 905    format('Calculate another reactivity using the same flux and',
     &         ' cross section data? (y/n)')
 915    format(////)
c-----------------------------------------------------------------------
c
 999    stop
        end
```

```
      subroutine xsect(fmat,mmat,qmat,ngroup,nmat,iflag)
c
c       This program reads the macroscopic cross section data for each
c      material specified in the MATLS= statement from the binary file
c       MACRXS. The data is then put into the appropriate form for the
c       F and M matricies which are then sent to the main program.
c
c       Note: * the total cross section in position j=3 is used
c             * higher order scattering cross sections are read in but
c               only the P0 (ISCT=0) data is used.
c             * currently, this routine cannot handle data with
upscatter;
c               however, it can be easily modified to do so.
c             * removal cross section is not used because sig-t needs
angular
c               flux while sig-s is multiplied by the scalar flux
c
c       File format obtained from LA-9184-M, Rev. "Revised User's Manual
c      for ONEDANT...", RDO, et al.
c
c       ?????????????????????????? Definitions ??????????????????????????
c       fmat - matrix holding fission cross section data (chi*nu*sigf)
c      mmat - matrix holding scattering cross section data
c      qmat - matrix holding total cross section data
c       nslim - max array size for "scat" array
c       ????????????????????????????????????????????????????????????????????
c
      implicit integer (i,j,k,l,m,n)
      parameter (lim=100,nslim=1000)
      integer ngpb(lim,lim),ifsg(lim,lim)
      double precision c(lim,lim,lim), scat(nslim), vel(lim),
     &    mmat(lim,lim,lim), fmat(lim,lim,lim), qmat(lim,lim)
      character*8 hname, huse(2), hmat(lim)
c
c-----------------------------------------------------------------------
      open(unit=11,file='macrxs',form='unformatted',status='old')
c
c-----------------------------------------------------------------------
c   initialize M matrix
c-----------------------------------------------------------------------
c
      do 20 i=1,lim
      do 20 j=1,lim
      do 20 k=1,lim
 20   mmat(i,j,k) = 0.0
c
c-----------------------------------------------------------------------
c   read file identification
c-----------------------------------------------------------------------
c
      read(11) hname,(huse(i),i=1,2),ivers
c
c-----------------------------------------------------------------------
c   read file control
c-----------------------------------------------------------------------
c
      read(11) ngroup,nmat,nord,ned,idpf,lng,maxup,maxdn,nprin,i21p1
      nedt = ned + nprin
c
c-----------------------------------------------------------------------
c   read file data: hmat(i), hed(j), vel(n), emax(n).  Note: dummy
c   variables used for those variables not used in code to conserve
c   memory required for arrays.
c-----------------------------------------------------------------------
c
      read(11) (hmat(i),i=1,nmat),(hmat(j),j=1,nedt),
     &         (vel(n),n=1,ngroup),(vel(n),n=1,ngroup),emin
c
c-----------------------------------------------------------------------
c   read cross section data
c-----------------------------------------------------------------------
c
      do 22 k=1,ngroup
         read(11) ((c(i,j,k),i=1,nmat), j=1,nedt)
         if (nord.gt.0) then
            read(11) ((ngpb(l,j),l=1,nord),j=1,nmat),
     &               ((ifsg(l,j),l=1,nord),j=1,nmat)
c
            ntab = 0
            do 24 i=1,nmat
            do 24 j=1,nord
 24         ntab = ntab + ngpb(j,i)
            if (ntab.gt.nslim) then
               write(*,200)ntab,nslim,k
               iflag = 999
               go to 21
            endif
            read(11) (scat(j),j=1,ntab)
c
c   extract P0 cross section (m=1) and put in the proper form of the
c   M matrix for use in the reactivity calculation
c
            ncnt = 0
            do 26 l=1,nmat
               do 27 m=1,nord
               do 27 n=1,ngpb(m,l)
                  ncnt = ncnt + 1
                  if (m.eq.1) mmat(k,k+1-n,l) = scat(ncnt)
 27            continue
c
c   store total cross section in N matrix
c   (Note: the leakage term integrates out so only total x-s on diag)
c
               qmat(k,l) = c(1,3,k)
 26         continue
         endif
 22   continue
c
c-----------------------------------------------------------------------
c   create the fission matrix F for use in the reactivity calculation
c-----------------------------------------------------------------------
c
      do 28 i=1,nmat
      do 28 j=1,ngroup
      do 28 k=1,ngroup
         fmat(j,k,i) = c(i,1,j)*c(i,2,k)
 28   continue
c
c-----------------------------------------------------------------------
c
200   format(//,'XSECT:**** arrays undersized for number of ',
     &       'scattering terms',//,14x,'NTAB = ',i5,' at group ',i3,
     &       4x,'LIM = ',i5,//)
c-----------------------------------------------------------------------
c
 21   close(unit=11)
      return
      end
```