NUMERICAL METHODS FOR DIFFUSION THEORY CRITICALITY

© M. Ragheb
4/11/2008

1. INTRODUCTION

The diffusion equation can be solved analytically for simple geometries and material compositions such as homogeneous reactors. More complex reactor designs involve multiple regions of different compositions such as core regions with fissile materials and reflector regions with scattering materials. This suggests the discretization of the diffusion equation in the spatial variable at specified mesh points and the use of computers to solve the ensuing linear system of equations for the magnitude of the flux at 4each mesh point, as well as the corresponding eigen-value which is the effective multiplication factor.

2. THE ONE GROUP DIFFUSION EQUATION

Multi-group diffusion theory problems involve a calculation in the spatial variable for each group of neutrons. To study the treatment of the spatial variable, we thus concentrate on the treatment of the one-group diffusion equation:

$$-\nabla .[D(r)\nabla \phi(\bar{r})] + \Sigma_a(\bar{r})\phi(\bar{r}) = \frac{\nu_f}{k}\Sigma_f(\bar{r})\phi(\bar{r}) \tag{1}$$

where we explicitly account for the spatial dependence of the diffusion coefficient D, the macroscopic absorption cross section $\Sigma_a(\bar{r})$, the macroscopic fission cross section $\Sigma_f(\bar{r})$, and the neutron flux $\phi(\bar{r})$.

The factor k is here the eigen-value belonging to the fundamental eigen-vector $\phi(\bar{r})$. If k > 1, the system is supercritical, since the neutron production must be reduced for the neutron balance to be valid. If k = 1, then the losses of neutrons are equal to the production rate and the system is critical. If k < 1, the system is subcritical. Thus k can be identified as the effective multiplication factor of the multiplying system.

In addition to Eqn. 1, we must satisfy the boundary condition that the neutron flux vanishes at the extrapolated boundary of the system:

$$\phi(R+d) = \phi(R+0.71\lambda_{tr}) = \phi(R_e) = 0 \tag{2}$$

where R is the reactor's core radius in spherical geometry, and λ_{tr} is the transport mean free path.

3. ANALYTICAL EXPRESSION FOR THE MULTIPLICATION FACTOR

In spherical geometry Eqn. 1 becomes:

$$-\frac{1}{r^2}\frac{d}{dr}[r^2 D(r)\frac{d}{dr}\phi(r)] + \Sigma_a(r)\phi(r) = \frac{\nu_f}{k}\Sigma_f(r)\phi(r) \qquad (3)$$

For a homogeneous reactor,

D(r) = D =constant,
Σ_a , Σ_f = constant,

and Eqn. 3 simplifies to:

$$\frac{d^2\phi(r)}{dr^2} + \frac{2}{r}\frac{d\phi(r)}{dr}] + \frac{1}{D}[\frac{\nu_f}{k}\Sigma_f - \Sigma_a]\phi(r) = 0 \qquad (4)$$

If we let:

$$\phi(r) = \frac{u(r)}{r} \qquad (5)$$

thus:
$$\frac{d\phi(r)}{dr} = \frac{1}{r}\frac{du(r)}{dr} - \frac{u(r)}{r^2}$$

and:
$$\frac{d^2\phi(r)}{dr^2} = -\frac{1}{r^2}\frac{du(r)}{dr} + \frac{1}{r}\frac{d^2u(r)}{dr^2} - \frac{1}{r^2}\frac{du(r)}{dr} + 2\frac{u(r)}{r^3}$$

Substituting in Eqn. 4, we get:

$$-2\frac{1}{r^2}\frac{du}{dr} + \frac{1}{r}\frac{d^2r}{dr^2} + 2\frac{u}{r^3} + 2\frac{1}{r^2}\frac{du}{dr} - 2\frac{u}{r^3} + \frac{1}{D}[\frac{\nu_f\Sigma_f}{k} - \Sigma_a]\frac{u}{r} = 0$$

Upon cancellation of terms, we get:

$$\frac{d^2u}{dr^2} + \frac{1}{D}\left[\frac{\nu_f\Sigma_f}{k} - \Sigma_a\right]u = 0 \qquad (6)$$

Equation 6 has a solution:

$$u = A\cos(Br) + C\sin(Br) \qquad (7)$$

From Eqn.5, the neutron flux is thus given by:

$$\phi = \frac{A\cos(Br)}{r} + \frac{C\sin(Br)}{r} \tag{8}$$

where the material buckling is given by:

$$B_m^2 = \frac{1}{D}\left[\frac{\nu_f \Sigma_f}{k} - \Sigma_a\right] \tag{9}$$

Since the neutron flux must remain finite at the center of the reactor's core at r = 0, this implies that the constant A = 0, and:

$$\phi = \frac{C\sin(Br)}{r} \tag{10}$$

At the extrapolated core boundary, Eqn. 2 applies, and we get:

$$\phi(R_e) = \frac{C\sin(BR_e)}{R_e} = 0 \tag{11}$$

where $R_e = R + 0.71\lambda_{tr}$

is the extrapolated radius.

This implies that:

$$BR_e = n\pi, n = 1,2,3,... \tag{12}$$

For the fundamental mode eigenvalue, n = 1, and the geometrical buckling is given by:

$$B_g^2 = (\frac{\pi}{R_e})^2 \tag{13}$$

The criticality condition is obtained by equating the geometrical buckling to the material buckling, to the buckling in general:

$$B_g^2 = B_m^2 = B^2$$

Thus using Eqns. 9 and 13 we get:

$$\frac{1}{D}\left[\frac{\nu_f \Sigma_f}{k} - \Sigma_a\right] = (\frac{\pi}{R_e})^2$$

from which we the criticality equation in spherical geometry as:

$$k = \frac{\nu_f \Sigma_f}{\Sigma_a + D(\frac{\pi}{R_e})^2} \tag{14}$$

Another form for the multiplication factor k can be obtained from Eqn. 1 if we integrate over the reactor's volume:

$$k = \frac{\int_V \nu_f \Sigma_f(\bar{r})\phi(\bar{r})dV}{-\int_V \nabla.[D\nabla\phi(\bar{r})]dV + \int_V \Sigma_a(\bar{r})\phi(\bar{r})dV} \tag{15}$$

which is the ratio of the neutron production rate from fissions to the neutron loss rates due to leakage and to absorptions.

It was possible to obtain an exact analytical solution for $\phi(\bar{r})$ in the case of a homogeneous reactor. If the system is not homogeneous when D(r) is not a constant; which is the situation encountered in practice, a discretization of the spatial variable $\bar{r}$ becomes necessary.

4. DISCRETIZATION OF THE DIFFUSION EQUATION

We consider Eqn. 3 for discretization, and we use the "Box Integration" method to discretize it. The spatial coordinate is subdivided into mesh points as shown in Fig. 1 for spherical coordinates.

Around each spatial mesh point (j), one can apply neutron conservation, by constructing a box and integrating the diffusion equation over it. This establishes a relationship between each mesh point and the adjacent ones. We choose the j-th point to be lying on the interfaces between two regions.

The spacing between the mesh points does not have to be uniform and is given by:

$$\Delta r_j = r_{j+1} - r_j \tag{16}$$

Integrating Eq 3 from $r_{j-1/2}$ to $r_{j+1/2}$ yields:

$$
\begin{aligned}
& -\int_{r_{j-1/2}}^{r_{j+1/2}} \frac{1}{r^2}\frac{d}{dr}[r^2 D(r)\frac{d\phi(r)}{dr}]4\pi r^2 dr \\
& + \int_{r_{j-1/2}}^{r_{j+1/2}} \Sigma_a(r)\phi(r)4\pi r^2 dr = \frac{\nu_f}{k}\int_{r_{j-1/2}}^{r_{j+1/2}} \Sigma_f(r)\phi(r).4\pi r^2 dr
\end{aligned}
\qquad (17)
$$

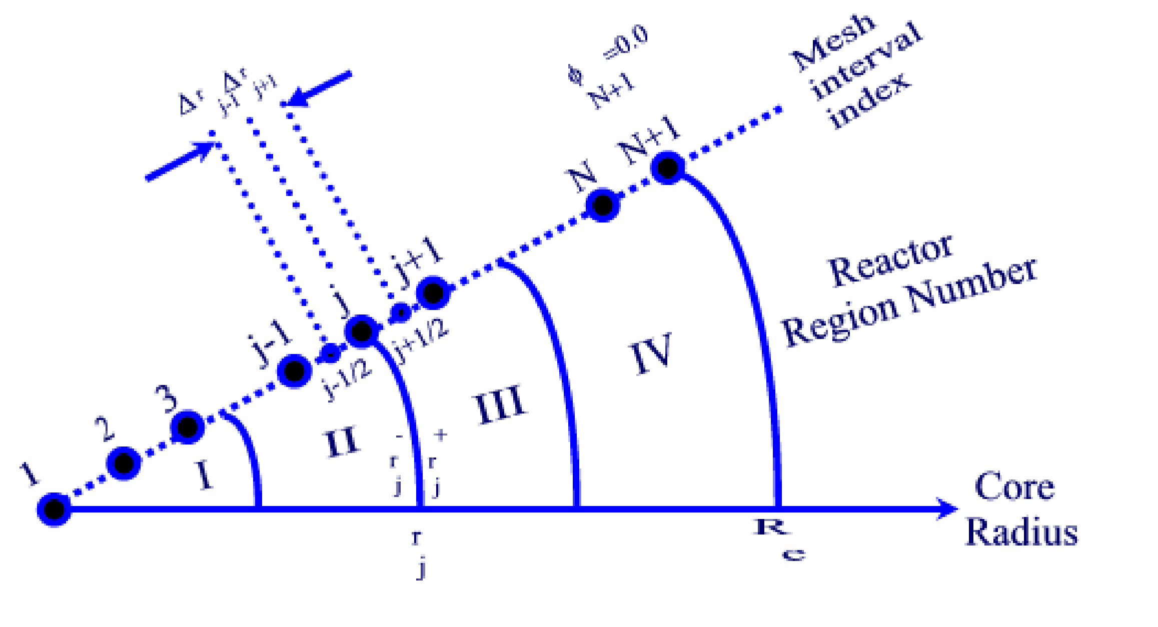

Fig. 1 Coordinate system for the difference discretization of the diffusion equation in spherical geometry.

Integrating the leakage term we get:

$$
\begin{aligned}
& -[r^2 D(r)\frac{d\phi(r)}{dr}]_{r_{j+1/2}} + [r^2 D(r)\frac{d\phi(r)}{dr}]_{r_{j-1/2}} \\
& + \int_{r_{j-1/2}}^{r_{j+1/2}} \Sigma_a(r)\phi(r)r^2 dr = \frac{\nu_f}{k}\int_{r_{j-1/2}}^{r_{j+1/2}} \Sigma_f(r)\phi(r)r^2 dr
\end{aligned}
\qquad (18)
$$

Since the material properties can vary, particularly around interfaces, we expand the flux in the two integrals in a Taylor's series in which we truncate after the second term:

$$I_1 = \int_{r_{j-1/2}}^{r_{j+1/2}} \Sigma_a(r)\phi(r)r^2 dr = \int_{r_{j-1/2}}^{r_j^-} \Sigma_a(r)\phi(r)r^2 dr + \int_{r_j^+}^{r_{j+1/2}} \Sigma_a(r)\phi(r)r^2 dr$$

$$= \int_{r_{j-1/2}}^{r_j^-} (r_j^+)^2 \Sigma_a(r_j^-)[\phi(r_j^-) - (r_j^- - r_{j-1/2})\frac{d\phi(r_j^-)}{dr_j} + ...]dr + \qquad (19)$$

$$+ \int_{r_j^+}^{r_{j+1/2}} (r_j^+)^2 \Sigma_a(r_j^+)[\phi(r_j^+) + (r_{j+1/2} - r_j^+)\frac{d\phi(r_j^+)}{dr_j} + ...]dr$$

If we make the approximations:

$$(r_j^-)^2 \cong (r_j^+)^2 \cong (r_j)^2$$

$$\int_{r_{j-1/2}}^{r_j^-} (r_j^- - r_{j-1/2})\frac{d\phi(r_j^-)}{dr_j} dr \cong \int_{r_j^+}^{r_{j+1/2}} (r_{j+1/2} - r_j^+)\frac{d\phi(r_j^+)}{dr_j} dr$$

the integral is approximately equal to:

$$\begin{aligned} I_1 &\cong (r_j)^2 \Sigma_a(r_j^-)\phi(r_j^-)(r_j^- - r_{j-1/2}) + (r_j)^2 \Sigma_a(r_j^+)\phi(r_j^+)(r_{j+1/2} - r_j^+) \\ &\cong (r_j)^2 \Sigma_a(r_j^-)\phi(r_j)(r_j - r_{j-1/2}) + (r_j)^2 \Sigma_a(r_j^+)\phi(r_j)(r_{j+1/2} - r_j) \qquad (20) \\ &\cong (r_j)^2 \Sigma_a(r_j^-)\phi(r_j)\frac{(r_j - r_{j-1})}{2} + (r_j)^2 \Sigma_a(r_j^+)\phi(r_j)\frac{(r_{j+1} - r_j)}{2} \end{aligned}$$

where we used approximation:

$$\phi(r_j^-) \cong \phi(r_j^+) \cong \phi(r_j)$$

In the case where:

$$\Sigma_a(r_j^-) \cong \Sigma_a(r_j^+),$$

which applies to all the points <u>except</u> at the interfaces, we get:

$$\Sigma_a(r_j^-) \cong \Sigma_a(r_j^+)$$

and:

$$I_1 \cong r_j{}^2 \Sigma_a(r_j)\phi(r_j)\frac{r_{j+1}-r_{j-1}}{2}$$
$$\cong r_j{}^2 \Sigma_a(r_j)\phi(r_j)\frac{\Delta r_j + \Delta r_{j-1}}{2} \tag{21}$$

The same relationship holds for the source integral:

$$I_2 \cong \frac{\nu_f}{k}\Sigma_f(r_j)\phi(r_j)(r_j)^2.\frac{\Delta r_j + \Delta r_{j-1}}{2} \tag{22}$$

Substituting in Eq. 18 we can write:

$$[r^2 D(r)\frac{d\phi(r)}{dr}]_{r_{j-1/2}} - [r^2 D(r)\frac{d\phi(r)}{dr}]_{r_{j+1/2}} + r_j^2\Sigma_a(r_j)\phi(r_j)\frac{\Delta r_j + \Delta r_{j-1}}{2}$$
$$= \frac{\nu_f}{k}\Sigma_f(r_j)\phi(r_j)(r_j)^2.\frac{\Delta r_j + \Delta r_{j-1}}{2} \tag{23}$$

We can now approximate the derivatives by forward first order difference equations:

$$\left[\frac{d\phi(r)}{dr}\right]_{r_{j-1/2}} \cong \frac{\phi(r_j)-\phi(r_{j-1})}{\Delta r_{j-1}} \tag{24}$$

$$\left[\frac{d\phi(r)}{dr}\right]_{r_{j+1/2}} \cong \frac{\phi(r_{j+1})-\phi(r_j)}{\Delta r_{j-1}} \tag{25}$$

Inserting Eqns. 24 and 25 into Eqn. 21, we get:

$$r_{j-1/2}^2 D(r_{j-1/2})\frac{\phi(r_j)}{\Delta r_{j-1}} - r_{j-1/2}^2 D(r_{j-1/2})\frac{\phi(r_{j-1})}{\Delta r_{j-1}} +$$
$$r_{j+1/2}^2 D(r_{j+1/2})\frac{\phi(r_j)}{\Delta r_j} - r_{j+1/2}^2 D(r_{j+1/2})\frac{\phi(r_{j+1})}{\Delta r_j} +$$
$$r_j^2\Sigma_a(r_j)\phi(r_j)\frac{\Delta r_j + \Delta r_{j-1}}{2}$$
$$= \frac{\nu_f}{k}\Sigma_f(r_j)\phi(r_j)(r_j)^2.\frac{\Delta r_j + \Delta r_{j-1}}{2} \tag{26}$$

At an interface the last two terms are replaced by their equivalents from Eq.20 as:

$$
\begin{aligned}
&r_{j-1/2}^2 D(r_{j-1/2})\frac{\phi(r_j)}{\Delta r_{j-1}} - r_{j-1/2}^2 D(r_{j-1/2})\frac{\phi(r_{j-1})}{\Delta r_{j-1}} + \\
&r_{j+1/2}^2 D(r_{j+1/2})\frac{\phi(r_j)}{\Delta r_j} - r_{j+1/2}^2 D(r_{j+1/2})\frac{\phi(r_{j+1})}{\Delta r_j} + \\
&r_j^2 \Sigma_a(r_j^-)\phi(r_j)\frac{\Delta r_{j-1}}{2} + r_j^2 \Sigma_a(r_j^+)\phi(r_j)\frac{\Delta r_j}{2} \\
&= \frac{\nu_f}{k}\Sigma_f(r_j^-)\phi(r_j)(r_j)^2\frac{\Delta r_{j-1}}{2} + \frac{\nu_f}{k}\Sigma_f(r_j^+)\phi(r_j)(r_j)^2\frac{\Delta r_j}{2}
\end{aligned}
\tag{27}
$$

Thus we have two finite difference equations relating the mesh points, one applying for interface points, and one for non-interface points.

Combining the terms in ϕ_{j-1}, ϕ_j and ϕ_{j+1} we can now write the finite difference equations for the mesh points:

$$
-a_j\phi_{j+1} + b_j\phi_j - c_j\phi_{j-1} = \frac{F_j}{k}\phi_j \qquad , j = 1,2,\ldots,N+1 \tag{28}
$$

where we have defined:

$$\phi_j \equiv \phi(r_j)$$

$$a_j \equiv \frac{r_{j+1/2}D(r_{j+1/2})}{\Delta r_j}$$

$$b_j \equiv \frac{r_{j-1/2}^2 D(r_{j-1/2})}{\Delta r_j} + \frac{r_{j+1/2}^2 D(r_{j+1/2})}{\Delta r_j} + \frac{r_j^2}{2}[\Sigma_a(r_j^-)\Delta r_{j-1} + \Sigma_a(r_j^+)\Delta r_j]$$

$$c_j \equiv \frac{r_{j-1/2}^2 D(r_{j-1/2})}{\Delta r_{j-1}}$$

$$F_j \equiv \frac{\nu_f r_j^2}{2k}[\Sigma_f(r_j^-)\Delta r_{j-1} + \Sigma_f(r_j^+)\Delta r_j]$$

If we expand Eqns. 28, and set $\phi_{N+1} = 0.0$ as the boundary condition, we get the tri-diagonal linear system:

$$
\begin{array}{cccccccc}
+b_1\phi_1 & -a_1\phi_2 & & & & & = & \dfrac{F_1}{k}\phi_1 \\
-c_2\phi_1 & +b_2\phi_2 & -a_2\phi_3 & & & & = & \dfrac{F_2}{k}\phi_2 \\
 & -c_3\phi_2 & +b_3\phi_3 & -a_3\phi_4 & & & = & \dfrac{F_3}{k}\phi_3 \\
 & & . & . & . & & = & . \\
 & & & . & . & . & = & . \\
 & & & & . & . & = & . \\
 & & & & & -c_N\phi_{N-1} \quad +b_N\phi_N & = & \dfrac{F_N}{k}\phi_N
\end{array}
\tag{29}
$$

In matrix form this equation can be written as:

$$\underline{\underline{A}}\,\underline{\phi} = \frac{\underline{\underline{F}}}{k}\underline{\phi} \tag{30}$$

where:

$$
\underline{\underline{A}} = \begin{pmatrix}
+b_1 & -a_1 & & & \bigcirc \\
-c_2 & +b_2 & -a_2 & & \\
 & -c_3 & +b_3 & -a_3 & \\
 & & \ddots & \ddots & \ddots \\
\bigcirc & & & -c_N & +b_N
\end{pmatrix}
$$

$$
\underline{\underline{F}} = \begin{pmatrix}
F_1 & & & \bigcirc \\
 & F_2 & & \\
 & & F_3 & \\
 & \bigcirc & & \ddots \\
 & & & & F_N
\end{pmatrix}
$$

and: $\underline{\phi} = (\phi_1 \quad \phi_2 \quad \phi_3 \cdots\cdots \phi_N)$

The method used above is called a "three-point difference method" since the flux equations at three neighboring points *j*-1, *j* and *j*+1 are coupled to each other.

5. TREATMENT OF DIFFERENT GEOMETRIES

The previous derivation is formulated for spherical geometry. For other geometries, the leakage term is written in general as:

$$\nabla.(D\nabla\phi) = \frac{1}{r^{\gamma}}[\frac{d}{dr}(r^{\gamma}D\frac{d\phi}{dr})] \tag{31}$$

where:

$\gamma = 0$, for cartesian coordinates
$\gamma = 1$, for cylindrical coordinates
$\gamma = 2$, for spherical coordinates

The volume elements in each case are also given by:

$$dV_{\gamma} = 2^{\gamma}\pi^{(1-\delta_{\gamma 0})}r^{\gamma}dr \tag{32}$$

where:

$$\begin{aligned}\delta_{\gamma 0} &= 0, \quad \forall\ \gamma \neq 0 \\ &= 1, \quad \forall\ \gamma = 0,\end{aligned}$$

is the Kronecker delta.

The 2π or 4π factors do not appear in the finite difference equations. The volume element is a true volume element only in spherical coordinates. In cylindrical coordinates it is a surface element, and it is a line element in slab geometry.

6. THE BUCKLING CORRECTION FOR MULTIMENSIONAL SYSTEMS

The buckling correction allows accounting for three-dimensional effects in a one-dimensional numerical calculation, by recognizing that the buckling in a given dimension represents neutron leakage and can be replaced by an effective absorption rate term.

In three dimensions and cartesian coordinates, the diffusion equation:

$$\frac{\partial}{\partial x}(D\frac{\partial\phi}{\partial x}) + \frac{\partial}{\partial y}(D\frac{\partial\phi}{\partial y}) + \frac{\partial}{\partial z}(D\frac{\partial\phi}{\partial z}) - \Sigma_a\phi + \frac{\nu_f\Sigma_f}{k}\phi = 0 \tag{33}$$

If we solve the equation in the x-direction only but wish to account for the neutron leakage in the y and z directions, assuming homogeneity in those directions, we can do this by writing:

$$\frac{d^2\phi}{dy} + B_y^2\phi = \frac{d^2\phi}{dz^2} + B_z^{2\phi} = 0 \tag{34}$$

Thus Eqn. 33 can be written as:

$$\frac{d}{dx}\left[D\frac{d\phi(x)}{dx}\right] - \left[(B_y^2 + B_z^2)D + \Sigma_a\right]\phi(x) + \frac{\nu_f\Sigma_f}{k}\phi(x) = 0$$

or:

$$\frac{d}{dx}\left[D\frac{d\phi(x)}{dx}\right] - \Sigma_a^{'}\phi(x) + \frac{\nu_f\Sigma_f}{k}\phi(x) = 0 \tag{35}$$

where a modified macroscopic cross section term in cartesian coordinates is used as:

$$\Sigma_a^{'} = \left[(B_y^2 + B_z^2)D + \Sigma_a\right] \tag{36}$$

In cylindrical coordinates, it becomes:

$$\Sigma_a^{'} = \left[B_z^2 D + \Sigma_a\right] \tag{37}$$

Thus one can compensate for the leakages that are not included in a one-dimensional calculation by assuming homogeneity in the untreated directions.

7. CRITICALITY OF BARE UNREFLECTED REACTOR CORES

We consider a bare unreflected spherical reactor core with the following parameters:

Radius	R = 50.0 [cm]
Diffusion coefficient	D = 1.0 [cm]
Macroscopic absorption cross section:	Σ_a = 0.75 [cm^{-1}]
Product of macroscopic fission cross section and neutron yield per fission:	$\nu\Sigma_f$ = 0.78 [neutron.cm^{-1}]

The diffusion area of the reactor can be calculated as:

$$L^2 = \frac{D}{\Sigma_a} = \frac{1}{0.75} = 1.333[cm^2]$$

The geometric buckling is:

$$B_b^2 = \left(\frac{\pi}{R_e}\right)^2 \approx \left(\frac{\pi}{R}\right)^2 = \left(\frac{\pi}{50}\right)^2 = 3.948x10^{-3}[cm^{-2}]$$

where we are ignoring the extrapolated distance.

We rewrite Eqn. 14 as:

$$k = \frac{\nu_f \Sigma_f}{\Sigma_a + D(\frac{\pi}{R_e})^2} = \frac{\frac{\nu_f \Sigma_f}{\Sigma_a}}{1 + \frac{D}{\Sigma_a}(\frac{\pi}{R_e})^2} = \frac{k_\infty}{1 + L^2 B^2} \tag{38}$$

We can then calculate the infinite medium multiplication factor from Eqn. 38 as:

$$k_\infty = \frac{\nu \Sigma_f}{\Sigma_a} = \frac{0.78}{0.75} = 1.04$$

and consequently an exact analytical solution for the effective multiplication factor can be calculated as:

$$k_{eff} = \frac{k_\infty}{1 + L^2 B_g^2} = \frac{1.04}{1 + 1.333x3.948x10^{-3}} = 1.03455$$

If the numerical code is used to calculate this value of the effective multiplication factor, the following input data file for a single "1" region reactor, a spherical geometry "2", and a convergence factor of "0.1e-5", and the values of the radius (R = 50.0 [cm]), diffusion coefficient (D = 1.0 [cm]), macroscopic absorption cross section ($\sum_a$ = 0.75 [cm^{-1}]), and the product of the average number of neutron per fission and macroscopic fission cross section (ν$\sum_f$ = 0.78 [neutron.cm^{-1}]), and zero geometric buckling correction values, can be used as input values:

```
 1 2+0.1e-05
   50.0
    1.0
    0.75
    0.78
    0.000     0.000     0.000
```

The numerical value for the effective multiplication value is remarkably similar to the earlier obtained analytical value as:

$$(k_{eff})_{numerical} = 1.0345$$

Obviously such a value of the effective multiplication factor implies a supercritical reactor. To obtain the value of the critical radius, one uses the criticality code in performing a criticality search. By varying the value of the radius as shown in Fig. 2, and inverting the graph at the effective multiplication value of unity one can obtain from the graph a critical radius of:

$$R_{critical} = 17.77\,[cm]$$

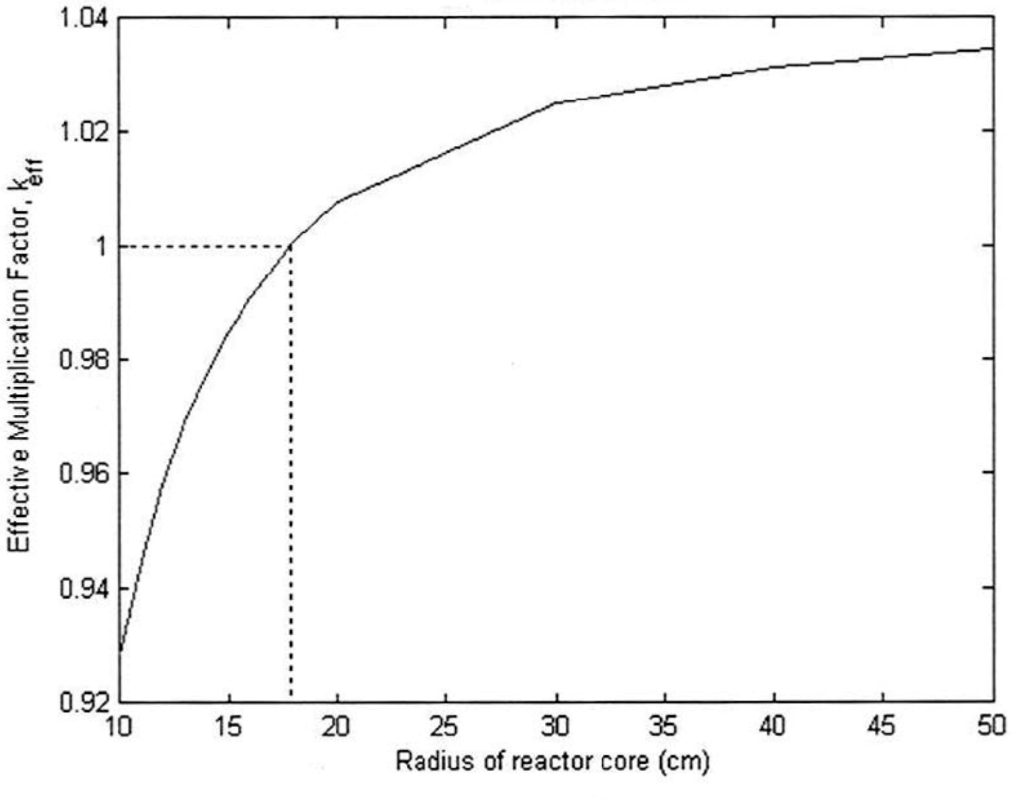

Fig. 2: Effective multiplication value k_{eff} as a function of the core radius R.

For this critical radius, the pointwise numerical values of the normalized flux can be plotted and are shown in Fig. 3.

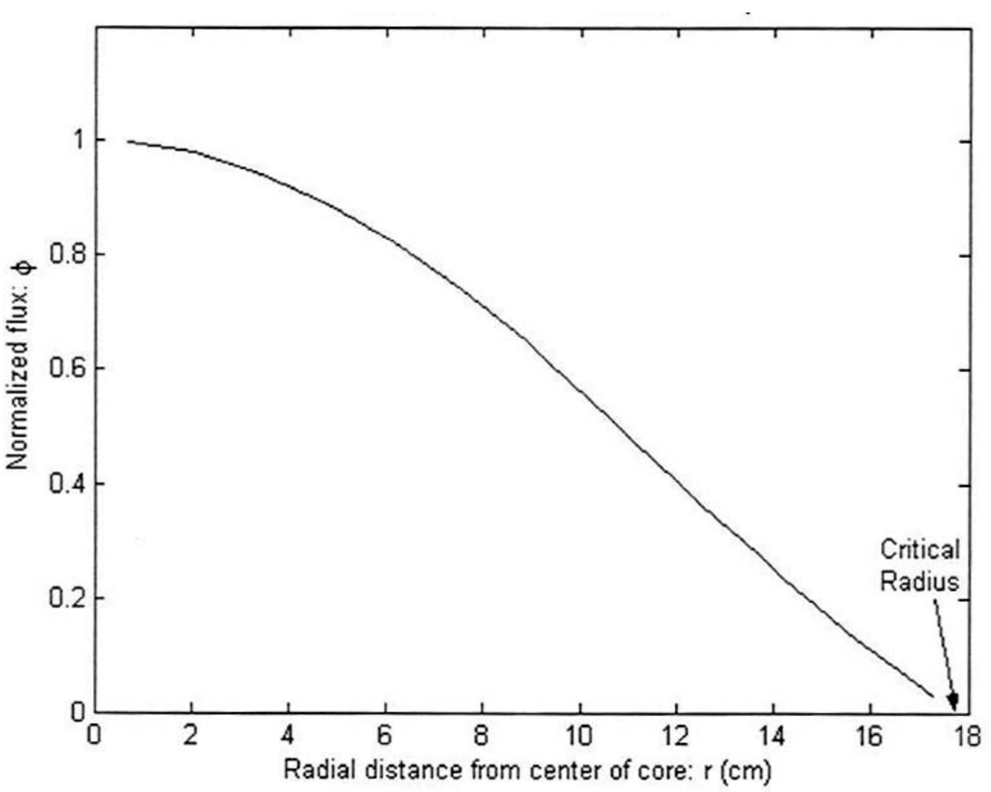

Fig. 3: Normalized flux distribution for the critical core.

8. CRITICALITY OF REFLECTED CORES

The advantage of a numerical approach to criticality becomes apparent when one deals with geometries and configurations that do not easily lend themselves to the derivation of exact analytical solutions like in the previous case. To demonstrate the usefulness of the numerical approach, we consider here the criticality of a reflected core. In this situation, the reactor geometry consists of two regions. The core region contains a fissile material and is a multiplying medium in nature, and the outer region is a scattering material.

This type of problem contains two degrees of freedom. By varying both the reflector thickness and the inner core radius, one in fact obtains a three dimensional surface of the effective multiplication factor as a function of the core radius and the reflector thickness. The critical configuration is obtained as the intersection of this surface with the plane representing an effective multiplication factor equal to unity. This intersection can be projected on the plane of the core radius and reflector thickness leading to a curve describing the combination of core radius and reflector thickness that would make the system critical.

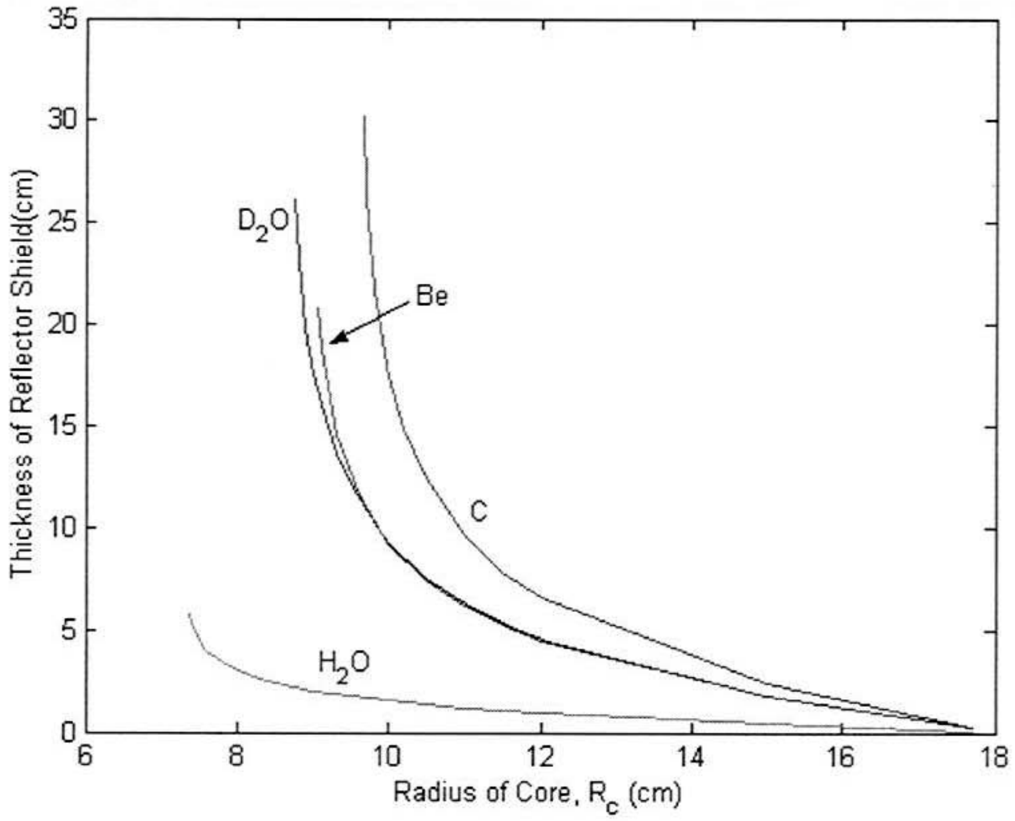

Fig. 4:.Critical core radii and reflector thicknesses for different reflector materials.

If the numerical code is used to calculate this value of the effective multiplication factor, the following input data file for a two region "2" region reactor, a spherical geometry "2", and a convergence factor of 0.1e-5, and the values of the radius (R = 8.0 [cm]), a reflector thickness of (11 - 8 = 3 cms), diffusion coefficient in the core (D = 1.0 [cm]), and in the reflector (D = 0.164 [cm]), macroscopic absorption cross section in the core ($\sum_a = 0.75$ [cm^{-1}]), and in the reflector ($\sum_a = 0.022$ [cm^{-1}]) and the product of the average number of neutron per fission and macroscopic fission cross section in the core ($\nu\sum_f = 0.78$ [neutron.cm^{-1}]), and in the reflector ($\nu\sum_f = 0.00$ [neutron.cm^{-1}]) and zero geometric buckling correction values can be used:

```
 2 2+0.1e-05
    8.000   11.000
    1.000    0.164
    0.750    0.022
    0.780    0.000
    0.000    0.000    0.000
```

Using the numerical approach one can investigate for instance the effects of different reflectors on the criticality of a given core composition. In Fig. 4, the critical core radius and the associated reflector thicknesses using different moderators is shown. It can be noticed that the reflector leads to a smaller critical radius in all the choices of reflector materials, than the unreflected bare core.

If light water is chosen as the reflector material, Fig. 5 shows a choice of a critical core radius at 8 cms, and an associated 3 cms of a water reflector. This is much smaller than the critical radius of the bare core at 17.7 cms. Figure 5 shows the normalized core and reflector flux distribution for light water as a reflector.

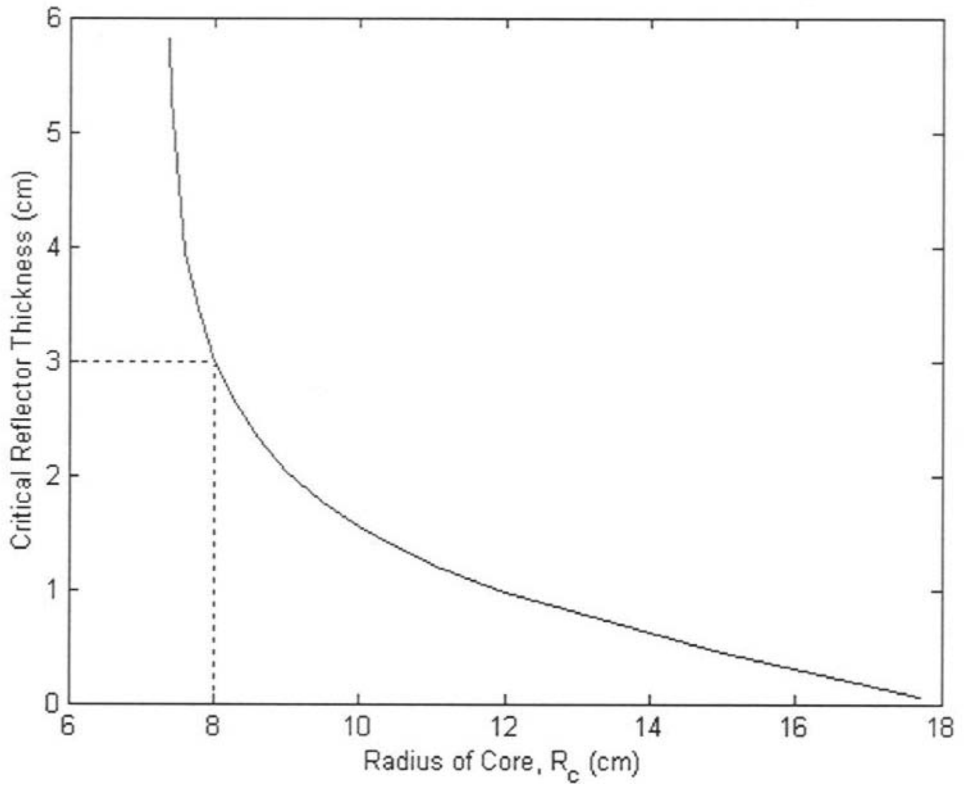

Fig. 5: A choice of a critical core radius and an associated reflector thickness for light water as a reflecting material.

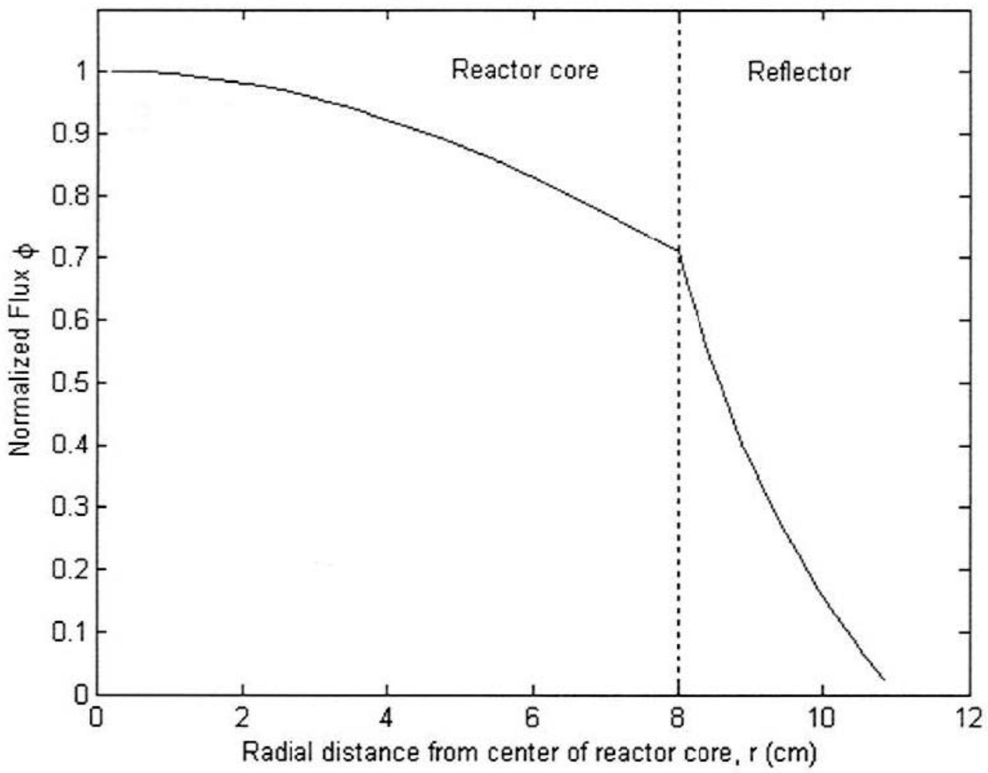

Fig. 6: Normalized core and reflector flux distribution for light water as a reflector.

EXERCISES

1. Calculate analytically the effective multiplication factor for a one region one dimensional spherical reactor core with the following parameters, then compare the results to those calculated numerically with the criticality code listed in the Appendices.

Radius	R = 50.0 cm
Diffusion coefficient	D = 1.0 cm
Macroscopic absorption cross section:	$\Sigma_a = 0.75$ cm^{-1}
Product of macroscopic fission cross section and neutron yield per fission:	$\nu\Sigma_f = 0.78$ neutron.cm^{-1}

2. Consider a spherical reactor core with the following parameters:
R = 35 [cms]
D = 1.0 [cm]
$\Sigma_a = 0.75$ [cm^{-1}]
$\nu\Sigma_f = 0.78$ [neutron.cm^{-1}]
$\varepsilon = p = f \approx 1$
a) Use the one-group diffusion theory to calculate the infinite medium multiplication factor, the diffusion area, the geometric buckling, and the effective multiplication factor.
b) Calculate analytically the radius that will make this reactor just critical.

c) Now use the numerical criticality code to check the analytical result for the effective multiplication factor, and compare it to the analytical result given in part1 above. Plot the normalized flux distribution.
d) Perform a "Criticality Search" by varying the radius of the reactor, plot the value of the effective multiplication factor against the radius, and deduce the critical radius, or the radius that will make the effective multiplication factor equal to unity. Check the numerical result against the analytical result in part 2 above.
e) Now, surround your reactor with a reflector of your choice (e.g. H_2O, D_2O, Be, C, U^{238}, etc.). Treat the problem as a two-region reactor. By varying both the thickness of the reflector and the radius of the core, optimize the design by determining a choice of appropriate core radius and reflector thickness. Justify your choice of the optimal configuration. Plot your results and discuss your observations and findings. Plot the normalized flux in your optimized final configuration.

REFERENCES

1. M. Ragheb, "Lecture Notes on Fission Reactors Design Theory," FSL-33, University of Illinois, 1982.
2. J. R. Lamarsh, "Introduction to Nuclear Engineering," Addison-Wesley Publishing Company, 1983.

APPENDIX

CRITICALITY CODE LISTING

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!          Multiregion One-dimensional One-group Diffusion Theory Criticality Code
!          Evaluation of Effective Multiplication Factor or Eigenvalue and Normalized Neutron Flux
!          Enhanced version of the ODOG procedure using the Power Iteration Method
!          ANSI Fortran-90 or 95 procedure. Unix or Windows operating System.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!          Dr. M. Ragheb
!          Department of Nuclear, Plasma and Radiological Engineering
!          University of Illinois at Urbana-Champaign
!          216 Talbot Laboratory, 104 S. Wright St., Urbana, Illinois 61801, USA.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
           program criticality
!
!          Version 4.5, 2008
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
           real lamda1,lamda2
!          lamda1  =        first eigenvalue iteration
!          lamda2  =        second eigenvalue iteration
!          Maximum number of regions in dimension statement = 5
           dimension rr(5),delx(5),n(5),r(100),rp12(100),rm12(100),delr(100)
!          rr               =        distances delimiting regions, measured from plane of symmetry [cm]
!          delx    =        size of interval in each region, chosen as equal to or less than the neutron
!                                    mean free path in the region [cm]
!          n                =        number of intervals in each region
!          r                =        r(j) = distance of each point from the origin
!          rp12    =        r(j + 1/2)
```

```
!		rm12	=		r(j - 1/2)
!		delr	=		r(j+1)-r(j)
		dimension d(5),dp12(100),dm12(100),sigma(5),sigp(100),sigm(100)
!		d			=		diffusion coefficient in each region [cm]
!		dp12	=		d(j + 1/2)
!		dm12	=		d(j - 1/2)
!		sigma	=		macroscopic absorption cross section in each region [cm-1]
!		sigp	=		sigma(j+)
!		sigm	=		sigma(j-)
		dimension ms(5),f(5),fp(100),fm(100),a(100),b(100),c(100),w(100),biga(100,100)
!		ms			=		number of intervals up to and including each region
!		f			=		nu*sigmaf, product of average number of neutrons per fission event and
!							macroscopic fission cross section [neutrons*cm-1]
!		fp			=		f(j+)
!		fm			=		f(j-)
!		a			=		matrix super diagonal
!		b			=		matrix diagonal
!		c			=		matrix sub-diagonal
!		w			=		source term of the diffusion equation in matrix form
!		biga	=		diffusion operator matrix
		dimension phi1(100),phi2(100),s1(100),s2(100),psi(100)
!		phi1	=		first flux iteration
!		phi2	=		second flux iteration
!		s1			=		first source term iteration
!		s2			=		second source term iteration
!		psi			=		source term
!
!		Sample input file 'incrit1', for a single region bare unreflected spherical reactor core
! 1 2+0.1e-05
!   50.0
!    1.0
!    0.75
!    0.78
!    0.000    0.000    0.000
!		Sample input file 'incrit2', for a two region spherical reactor core with outer reflector
! 2 2+0.1e-05
!    7.000   11.000
!    1.0      0.164
!    0.750    0.022
!    0.780    0.00
!    0.000    0.000    0.000
!		Sample input file 'incrit3', for a three region spherical reactor core with inner and outer reflectors
! 3 2+0.1e-05
!    2.000    8.000    12.000
!    0.164    1.0      0.164
!    0.022    0.750    0.022
!    0.00     0.780    0.00
!    0.000    0.000    0.000
!
		character*1 tab
		tab=char(9)
!		Open the input and output files
!		Input file is:		incrit
!		Output file is:		outcrit
!		Plotting file is plot
		open(unit=10,file='incrit3',status='old')
		open(unit=11,file='outcrit')
		open(unit=12,file='plot')
!   Write program information
		write (11,23)
		write (*,23)
```

```
23          format('Multiregion One-dimensional One-group Diffusion Theory Criticality',/,&
            &'Effective Multiplication factor or Eigenvalue and neutron flux evaluated',/,&
            &'Enhanced version of the ODOG procedure using the Power Iteration Method',/,&
            &'Fortran-90 or 95 procedure',/,&
            &'Unix or Windows operating system',/&
   &'Dr. Magdi Ragheb',/,&
            &'University of Illinois at Urbana-Champaign',/,&
            &'216 Talbot laboratory, 104 S. Wright St., Urbana, Illinois 61801, USA.',/)
!           Read problem data from file incrit
!           Write problem output on file outcrit
!           Write plot file on file plot for input to plotting routine, e. g. Microsoft Excel
            read(10,1)m,nn,eps
1           format(2I2,e8.1)
!           m               =       number of regions
!           nn              =       geometry index,    nn=0 cartesian geometry
!                                                      nn=1 cylindrical geometry
!                                                      nn=2 spherical geometry
!           eps             =       convergence parameter for the eigenvalue iteration
            write (11,91)
            write (*,91)
91          format('Number of regions',2x,'Geometry Index: 0=cartesian 1=cylindrical 2=spherical',&
  &2x,'Convergence parameter')
            write (11,1) m,nn,eps
            write (*,1) m,nn,eps
            read(10,2)(rr(i),i=1,m)
2           format(8f10.3)
            write(11,3)
            write(*,3)
3           format('Regions boundaries')
            write(11,2)(rr(i),i=1,m)
            write(*,2)(rr(i),i=1,m)
            read(10,2)(d(i),i=1,m)
            write(11,4)
            write(*,4)
4           format('Regions diffusion coefficients')
            write(11,2)(d(i),i=1,m)
            write(*,2)(d(i),i=1,m)
            read(10,2)(sigma(i),i=1,m)
            write(11,5)
            write(*,5)
5           format('Regions macroscopic absorption cross section')
            write(11,2)(sigma(i),i=1,m)
            write(*,2)(sigma(i),i=1,m)
            read(10,2)(f(i),i=1,m)
            write(11,6)
            write(*,6)
6           format('Regions nu*macroscopic fission cross section product')
            write(11,2)(f(i),i=1,m)
            write(*,2)(f(i),i=1,m)
            read(10,93)bcz,bpy,bpz
93          format(3f10.3)
            write(11,92)
            write(*,92)
92          format('Buckling corrections')
            write(11,93) bcz,bpy,bpz
            write(*,*) bcz,bpy,bpz
!           Buckling corrections allow for three dimensional effects.
!           The buckling corrections can be assigned zero values.
!           bcz             =       cylindrical geometry buckling axial correction
!           bpy             =       cartesian geometry buckling correction in y direction
!           bpz             =       cartesian geometry buckling correction in z direction
```

```
!
          if(nn.eq.2)write(11,7)
          if(nn.eq.2)write(*,7)
7         format('Spherical Geometry')
          if(nn.eq.1)write(11,8)
          if(nn.eq.1)write(*,8)
8         format('Cylindrical Geometry')
          if(nn.eq.0)write(11,9)
          if(nn.eq.0)write(*,9)
9         format('Cartesian Geometry')
          if(nn.eq.2) go to 222
          if(nn.eq.0) go to 111
!         Buckling correction in the axial direction for a finite height cylinder
          do i=1,m
                    sigma(i)=sigma(i)+d(i)*bcz
          end do
          go to 222
!         Buckling correction in the y and z dimensions in cartesian geometry
111       do i=1,m
                    sigma(i)=sigma(i)+d(i)*(bpy+bpz)
          end do
!         Computation of the number of intervals in each region
222       n(1)=rr(1)*sigma(1)+1
          n1=n(1)
          if(n1.le.10) n(1)=20
          if(m.gt.1) go to 333
          m=2
          rr(2)=rr(1)
          rr(1)=rr(1)/2.0
          n(1)=rr(1)*sigma(1)+1
          d(2)=d(1)
          f(2)=f(1)
          sigma(2)=sigma(1)
333       do i=2,m
                    n(i)=(rr(i)-rr(i-1))*sigma(i)
                    ni=n(i)
                    if(ni.le.10) n(i)=20
          end do
          write (11,12)
          write(*,12)
12        format('Number of intervals in each region')
          write(11,13)(n(i),i=1,m)
          write(*,13) (n(i),i=1,m)
13        format(5i10)
!         Computation of the size of intervals in each region
          delx(1)=rr(1)/(n(1)-0.5)
          do i=2,m
                    delx(i)=(rr(i)-rr(i-1))/n(i)
          end do
          write(11,14)
          write(*,14)
14        format('Interval sizes in each region')
          write(11,2)(delx(i),i=1,m)
          write(*,2)(delx(i),i=1,m)
!         Initialization of first mesh point variables
          r(1)=delx(1)/2.0
          rp12(1)=r(1)+delx(1)/2.0
          rm12(1)=0.0
          delr(1)=delx(1)
          delrm=delx(1)/2.0
          dp12(1)=d(1)
```

```
              dm12(1)=d(1)
              sigp(1)=sigma(1)
              sigm(1)=sigma(1)
              fp(1)=+f(1)
              fm(1)=+f(1)
              a(1)=(rp12(1)**nn)*dp12(1)/delr(1)
              c(1)=0.0
              b(1)=a(1)+c(1)+r(1)**nn*(sigp(1)*delr(1)+sigm(1)*delrm)
              w(1)=r(1)**nn*(delr(1)*fp(1)+delrm*fm(1))
              ms(1)=n(1)
              do i=2,m
                      ms(i)=ms(i-1)+n(i)
              end do
              nt=ms(m)
!             Computation of mesh parameters in first region
              n1=n(1)
              do 555 i=2,n1
                      r(i)=r(1)+(i-1)*delx(1)
                      if(i.eq.n1) go to 444
                      rp12(i)=r(i)+delx(1)/2.0
                      rm12(i)=r(i)-delx(1)/2.0
                      fp(i)=f(1)
                      fm(i)=f(1)
                      sigp(i)=sigma(1)
                      sigm(i)=sigma(1)
                      dp12(i)=d(1)
                      dm12(i)=d(1)
                      go to 555
444                   rp12(i)=r(i)+delx(2)/2.0
                      rm12(i)=r(i)-delx(1)/2.0
                      fp(i)=f(2)
                      fm(i)=f(1)
                      sigp(i)=sigma(2)
                      sigm(i)=sigma(1)
                      dp12(i)=d(2)
                      dm12(i)=d(1)
555           continue
!             Computation of mesh parameters in other regions
              delx(m+1)=0.0
              f(m+1)=0.0
              sigma(m+1)=0.0
              d(m+1)=0.0
              do 666 i=2,m
                      msi=ms(i)
                      msm=ms(i-1)
              do 666 k=msm,msi
                      r(k)=r(msm)+(k-msm)*delx(i)
                      if(k.eq.msm) go to 777
                      if(k.eq.msi) go to 888
                      rp12(k)=r(k)+delx(i)/2.0
                      rm12(k)=r(k)-delx(i)/2.0
                      fp(k)=f(i)
                      fm(k)=f(i)
                      sigp(k)=sigma(i)
                      sigm(k)=sigma(i)
                      dp12(k)=d(i)
                      dm12(k)=d(i)
                      go to 666
777                   rp12(k)=r(k)+delx(i)/2.0
                      rm12(k)=r(k)-delx(i)/2.0
                      fp(k)=f(i)
```

```
                        fm(k)=f(i-1)
                        sigp(k)=sigma(i)
                        sigm(k)=sigma(i-1)
                        dp12(k)=d(i)
                        dm12(k)=d(i-1)
                        go to 666
888                     rp12(k)=r(k)+delx(i+1)/2.0
                        rm12(k)=r(k)-delx(i)/2.0
                        fp(k)=f(i+1)
                        fm(k)=f(i)
                        sigp(k)=sigma(i+1)
                        sigm(k)=sigma(i)
                        dp12(k)=d(i+1)
                        dm12(k)=d(i)
666         continue
!           Computation of sub-diagonal, diagonal, super-diagonal, and source terms
            l=nt-1
            do i=2,l
                        delr(i)=r(i+1)-r(i)
                        a(i)=rp12(i)**nn*dp12(i)/delr(i)
                        c(i)=rm12(i)**nn*dm12(i)/delr(i-1)
                        b(i)=a(i)+c(i)+r(i)**nn*(sigp(i)*delr(i)+sigm(i)*delr(i-1))/2.0
                        w(i)=r(i)**nn*(fp(i)*delr(i)+fm(i)*delr(i-1))/2.0
            end do
            write(11,15)
            write(*,15)
15          format(10x,'Sub-diagonal',10x,'Diagonal',10x,'Super-Diagonal',5x,'Source Term')
            do i=1,l
                        a(i)=-a(i)
                        c(i)=-c(i)
                        write(11,16)c(i),b(i),a(i),w(i)
                        write(*,16)c(i),b(i),a(i),w(i)
            end do
16          format(4(10x,E10.3))
!           Formation of matrix of coefficients A
            do  i=1,l
                        do j=1,l
                                    if(i.eq.j) go to 116
                                    if(i.eq.j+1) go to 117
                                    if(i.eq.j-1) go to 118
                                    biga(i,j)=0.0
                                    go to 119
116        biga(i,j)=b(i)
                                    go to 119
117                                 biga(i,j)=c(i)
                                    go to 119
118        biga(i,j)=a(i)
119                     continue
      end do
            end do
!           Coefficients matrix is written as needed by uncommenting the following statements
!           write(*,*) biga
!           do i=1,l
!                       write(11,17)(biga(i,j),j=1,l)
!                       write(*,17)(biga(i,j),j=1,l)
!17                     format(5E12.4)
!           end do
121         iff=1
            c(1)=0.0
            a(l)=0.0
            lamda1=1.0
```

```
        do i=1,l
                phi1(i)=1.0
        end do
        kk=1
123     continue
        do i=1,l
                s1(i)=w(i)*phi1(i)
                psi(i)=s1(i)/lamda1
        end do
!       Solve tridiagonal linear system of equations
        call tridag(iff,l,c,b,a,psi,phi2)
        phim=0.0
        do i=1,l
                if(phim.lt.phi2(i)) phim=phi2(i)
        end do
        do i=1,l
                phi1(i)=phi2(i)/phim
        end do
        do i=1,l
                s2(i)=w(i)*phi2(i)
        end do
        sumt=0.0
        sumb=0.0
        do i=1,l
                sumt=sumt+s2(i)*s2(i)
                sumb=sumb+s2(i)*psi(i)
        end do
        lamda2=sumt/sumb
!       Calculate relative error for use as convergence parameter
        rat=abs((lamda1-lamda2)/lamda1)
        if(rat.lt.eps) go to 122
        lamda1=lamda2
        kk=kk+1
        go to 123
122     continue
        write(11,18)kk
        write(*,18)kk
18      format('The number of outer iterations is:',I5)
        write(11,19)
        write(*,19)
19      format(3x,'Distance',1x,'Normalized Flux')
        do i=1,l
                write(11,20)r(i),tab,phi1(i)
                write(*,20)r(i),tab,phi1(i)
        end do
        do i=1,l
                write(12,20) r(i),TAB,phi1(i)
        end do
20      format(f10.3,a1,f10.3)
        write(11,21)
        write(*,21)
21      format('Eigenvalue, Effective multiplication factor, k(eff).')
        write(11,22)lamda2
        write(*,22)lamda2
22      format(f12.4)
        end
!
        subroutine tridag(ifirst,ilast,a,b,c,d,v)
!       This procedure solves a system of simultaneous linear equations with a tridiagonal
!   coefficient matrix.
!       a       = array of sub-diagonal coefficients
!       b           = array of diagonal coefficients
!       c           = array of super-diagonal coefficients
!       d           = source vector
!       v(if) ... v(l)          = final solution vector
!       ifirst  = first equation number
!       ilast       = last equation number
!       beta, gamma = intermediate arrays
        dimension a(100),b(100),c(100),d(100),v(100),beta(201),gamma(201)
!       Generate intermediate arrays beta an gamma
        beta(ifirst)=b(ifirst)
        gamma(ifirst)=d(ifirst)/beta(ifirst)
        ifirstp1=ifirst+1
        do i=ifirstp1,ilast
                    beta(i)=b(i)-(a(i)*(c(i-1)/beta(i-1)))
                    gamma(i)=(d(i)-a(i)*gamma(i-1))/beta(i)
        end do
!       Computation of final solution vector
        v(ilast)=gamma(ilast)
        last=ilast-ifirst
        do k=1,last
                    i=ilast-k
                    v(i)=gamma(i)-(c(i)*(v(i+1)/beta(i)))
        end do
        return
        end
```