TWO GROUP NEUTRON FLUX ANALYSIS

Greene, T; Phung, Q; Taylor, Z.

October 04, 2017

NE 571, Reactor Theory and Design I

Fall 2017

Prof. G. Maldonado

Table of Contents

Introduction and Background

Consider a two-dimensional finite cylinder of radius (R) and height (H) with a core composed of a homogenously distributed multiplying medium of fissile material surrounded by a water reflector and vacuum as shown in **Figure 1** below. Assume that the greatest concentration of fissile material, therefore the greatest neutron flux, is present at the points $r = 0$ and $h = 0$ and gradually decreases to zero at the points $r = \pm R$ and $z = \pm H/2$.

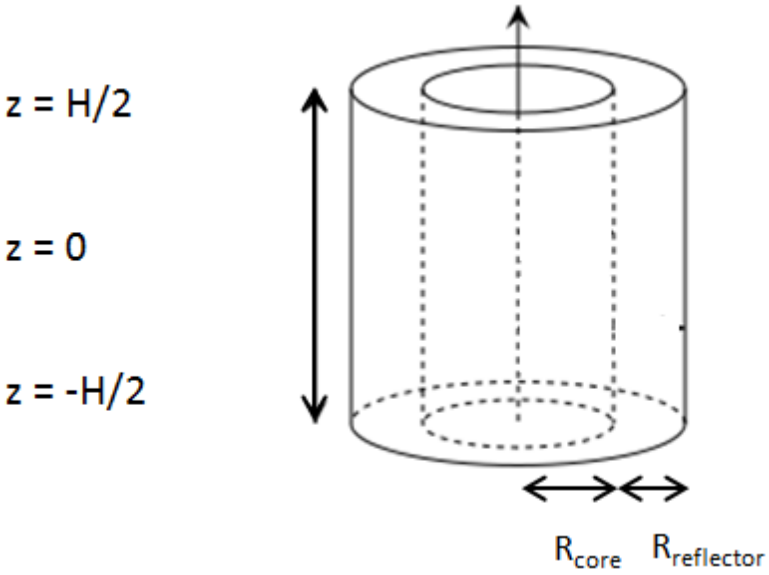


Figure 1: A Two-dimensional finite cylinder of varying geometry consisting of a cylindrical reactor core surrounded by a water reflector.

The purpose of this project was to employ numerical techniques and develop a program to solve for the neutron flux distribution and neutron multiplication factor (k) using two-group diffusion theory at various nodal points in the cylinder when given variable radii (R_{Core} and $R_{Reflector}$), heights, and nodal values for the radius and height, N and M respectively. The program will also be used to estimate the time necessary to solve the problem for a low, medium, and high number of nodes. Additionally, the program will also be used to calculate the critical dimensions of the homogenous core, using the following assumption: $R = H/2$, and the maximum "reflector savings" achievable for a critical core.

Methodology

The methodology of this project involves the derivation and generation of a code capable of providing a numerical solution to the two-group diffusion equation when given certain boundary and initial conditions, calculating the critical dimensions, and the reflector savings of the cylindrical core given the properties of a typical LWR. A complete, but brief, description of the numerical solution is provided below. The numerical solutions were then analyzed by plotting and examining the effective multiplication factor, k_{eff}.

Numerical Solution

To solve the two-group neutron diffusion equation numerically for a homogeneously distributed multiplying medium finite cylinder with a reflector, a steady-state assumption was applied to the model:

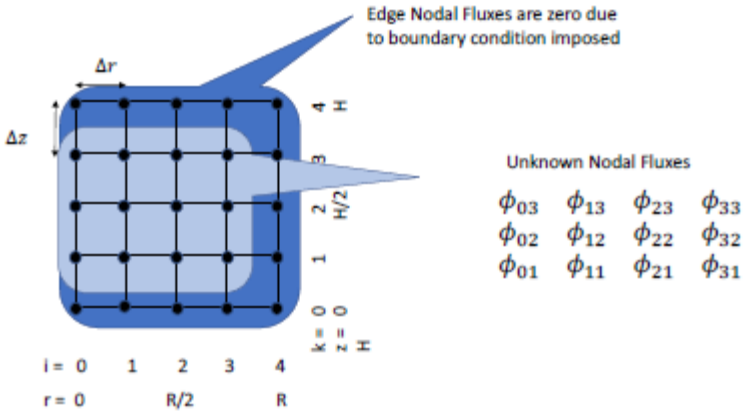
$$-\nabla \cdot D_1 \nabla \phi_1 + \Sigma_{R1}\phi_1 = \frac{1}{k}\left[\nu\Sigma_{f1}\phi_1 + \nu\Sigma_{f2}\phi_2\right] \qquad \textbf{Fast: Group 1} \qquad (1)$$

$$-\nabla \cdot D_2 \nabla \phi_2 + \Sigma_{a2}\phi_2 = \Sigma_{s12}\phi_1 \qquad \textbf{Thermal: Group 2} \qquad (2)$$

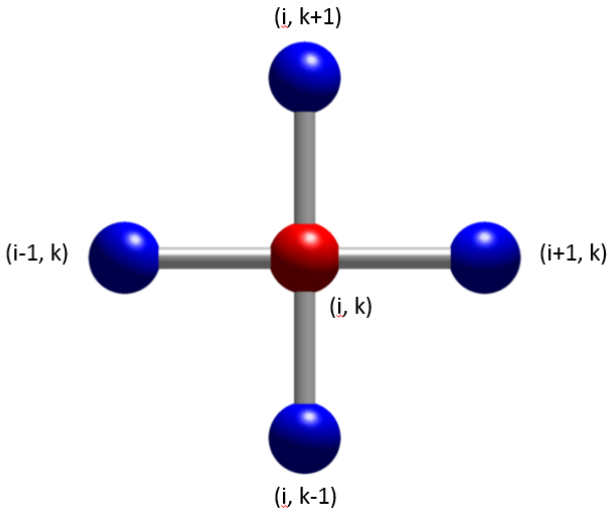
where the laplacian ∇^2 was determined to be:

$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r} r \frac{\partial}{\partial r} + \frac{\partial^2}{\partial z^2}. \qquad (3)$$

The discretization method of choice, the finite-difference equation, was applied to the cylinder by dividing the radial and axial directions into nodes which were then used for developing the numerical solution. For the purposes of developing the numerical solution, a five-point stencil, nodal discretization was used, see **Figure 2** below.



(a)



(b)

Figure 2: (a**)** Nodal discretization for two-dimensional finite cylinder; **(**b**)** Five-point stencil for node (*i*, *k*)

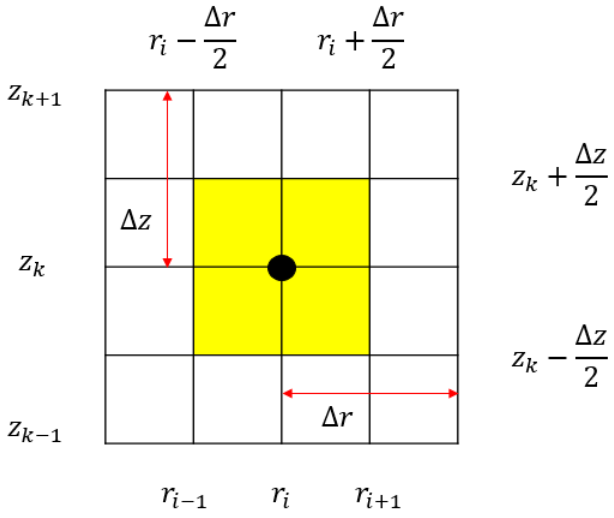


Figure 3: Discretization for node (*i*, *k*)

For an arbitrary internal node (*i*, *k*) in either the core or reflector, equations 1 and 2 are integrated around the volume of the node, see **Figure 3** for node (*i*, *k*). The distance between two nodes in the radial and axial directions are labeled Δr and Δz respectively, with the volume around node (*i*, *k*) defined by the distances $r_i \pm \frac{\Delta r}{2}$ and $z_k \pm \frac{\Delta z}{2}$. Multiplying both equations by *r* for simplification yields the following for the integrated region around an interior node (*i*, *k*).

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r}{2}}^{r_i+\frac{\Delta r}{2}} \left(-D_1 \frac{\partial}{\partial r} r \frac{\partial \phi_1}{\partial r} - Dr \frac{\partial^2 \phi_1}{\partial z^2} + \Sigma_{R1} \phi_1 r\right) dr =$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r}{2}}^{r_i+\frac{\Delta r}{2}} \left(\frac{1}{k}\left[\nu_1 \Sigma_{f1} \phi_1 + \nu_2 \Sigma_{f2} \phi_2\right] r\right) dr, \quad (4)$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r}{2}}^{r_i+\frac{\Delta r}{2}} \left(-D_2 \frac{\partial}{\partial r} r \frac{\partial \phi_2}{\partial r} - Dr \frac{\partial^2 \phi_2}{\partial z^2} + \Sigma_a \phi_2 r\right) dr =$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r}{2}}^{r_i+\frac{\Delta r}{2}} (\Sigma_{s12} \phi_1 r) dr, \quad (5)$$

Where *D* is the diffusion coefficient; Σ_R is the macroscopic removal cross section; ϕ is the neutron flux (cm^{-2} s^{-1}); *k* is the neutron multiplication factor; ν is the average number of neutrons released per fission; Σ_f is the macroscopic fission cross section; and Σ_{s12} is the macroscopic scattering cross section from energy group 1 to 2. The subscripts 1 and 2 denote the fast (1) and thermal (2) energy regimes respectively. Since no initial or boundary conditions are needed to discretize an internal node, equations 4 and 5 are developed in the exact same method as the internal nodes from project 1.

Fast:

$$-D_1\left(r_i + \frac{\Delta r}{2}\right)\Delta z \frac{\left(\phi_{i+1,k}{}^1 - \phi_{i,k}{}^1\right)}{\Delta r} + D_1\left(r_i - \frac{\Delta r}{2}\right)\Delta z \frac{\left(\phi_{i,k}{}^1 - \phi_{i-1,k}{}^1\right)}{\Delta r} - D_1(r_i \Delta \mathrm{r}) \frac{\left(\phi_{i,k+1}{}^1 - \phi_{i,k}{}^1\right)}{\Delta z} +$$

$$D_1(r_i \Delta \mathrm{r}) \frac{\left(\phi_{i,k}{}^1 - \phi_{i,k-1}{}^1\right)}{\Delta z} + \Sigma_{R1} \phi_{i,k}{}^1 \Delta z (r_i \Delta \mathrm{r}) = \frac{1}{k}\left[\nu_1 \Sigma_{f1} \phi_{i,k}{}^1 + \nu_2 \Sigma_{f2} \phi_{i,k}{}^2\right] \Delta z (r_i \Delta \mathrm{r}) \quad (6)$$

Thermal:

$$-D_2\left(r_i+\frac{\Delta r}{2}\right)\Delta z\frac{\left(\phi_{i+1,k}{}^2-\phi_{i,k}{}^2\right)}{\Delta r}+D_2\left(r_i-\frac{\Delta r}{2}\right)\Delta z\frac{\left(\phi_{i,k}{}^2-\phi_{i-1,k}{}^2\right)}{\Delta r}-D_2(r_i\Delta\mathrm{r})\frac{\left(\phi_{i,k+1}{}^2-\phi_{i,k}{}^2\right)}{\Delta z}+$$

$$D_2(r_i\Delta\mathrm{r})\frac{\left(\phi_{i,k}{}^2-\phi_{i,k-1}{}^2\right)}{\Delta z}+\Sigma_a\phi_{i,k}{}^2\Delta z(r_i\Delta\mathrm{r})=\Sigma_{s12}\phi_{i,k}{}^1\Delta z(r_i\Delta\mathrm{r}) \quad (7)$$

For this project, a full half-cylinder was assumed with "zero flux" boundary conditions imposed at all outermost nodes i.e. at *R* and $\pm$ H/2, ϕ = 0. The left-most node from the half-cylinder model will be treated as the central node and does not depend on the volume around an entire node or have a dependence on r_i. To account for the flux at r = 0 for this central position, it is necessary to enforce the integral of diffusion equation to be satisfied over the volume of the node (i, k), where the nodal region is defined by 0 to $\frac{\Delta r}{2}$ and $z_k-\frac{\Delta z}{2}$ to $z_k+\frac{\Delta z}{2}$. The equation, after multiplying through by r for simplification, at the central nodes (i, k) is thus:

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_0^{r_i+\frac{\Delta r}{2}}\left(-D_1\frac{\partial}{\partial r}r\frac{\partial\phi_1}{\partial r}-D_1r\frac{\partial^2\phi_1}{\partial z^2}+\Sigma_{R1}\phi_1r\right)dr=$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_0^{r_i+\frac{\Delta r}{2}}\left(\frac{1}{k}\left[\nu_1\Sigma_{f1}\phi_1+\nu_2\Sigma_{f2}\phi_2\right]r\right)dr, \quad (8)$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_0^{r_i+\frac{\Delta r}{2}}\left(-D_2\frac{\partial}{\partial r}r\frac{\partial\phi_2}{\partial r}-D_2r\frac{\partial^2\phi_2}{\partial z^2}+\Sigma_a\phi_2r\right)dr=$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_0^{r_i+\frac{\Delta r}{2}}(\Sigma_{s12}\phi_1r)dr, \quad (9)$$

Again, since no initial or boundary conditions are needed to discretize a central node, equations 8 and 9 are developed in the exact same method as the central nodes from project 1 with the final discretized equations being:

Fast:

$$-D_1\frac{\Delta r}{2}\Delta z^2\left({\phi_{i,k}}^1-{\phi_{0,k}}^1\right)-D_1\left({\phi_{0,k+1}}^1-2{\phi_{0,k}}^1+{\phi_{0,k-1}}^1\right)\frac{\Delta r^3}{8}+\Sigma_{R1}{\phi_{0,k}}^1\Delta z^2\frac{\Delta r^3}{8}$$

$$=\frac{1}{k}\left[\nu_1\Sigma_{f1}{\phi_{0,k}}^1+\nu_2\Sigma_{f2}{\phi_{0,k}}^2\right]\Delta z^2\frac{\Delta r^3}{8} \quad (10)$$

Thermal:

$$-D_2\frac{\Delta r}{2}\Delta z^2\left({\phi_{i,k}}^2-{\phi_{0,k}}^2\right)-D_2\left({\phi_{0,k+1}}^2-2{\phi_{0,k}}^2+{\phi_{0,k-1}}^2\right)\frac{\Delta r^3}{8}+\Sigma_a{\phi_{0,k}}^2\Delta z^2\frac{\Delta r^3}{8}$$

$$=\Sigma_{s12}{\phi_{i,k}}^1\Delta z^2\frac{\Delta r^3}{8} \quad (11)$$

For this project, because the fissile core was surrounded by a water reflector, a mesh point node was needed between the two materials. This mesh node is defined as the point at which two regions, nodes, meet and are subjected to the following conditions:

$$-D_m\frac{d\phi}{dx_m}\Big|_{x=x_i}=-D_n\frac{d\phi}{dx_n}\Big|_{x=x_i}\ and\ J_m|_{x_i}=J_n|_{x_i} \quad (12)$$

The subscripts *m* and *n* refer to the two different types of materials, in this case the fissile material and the water. The initial conditions simply state that the diffusion coefficients and flux currents are equal at the mesh point. The simplified Equations 13 and 14 below provide the two-group equations for the mesh point between the two regions:

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_{r_i-\frac{\Delta r_m}{2}}^{r_i}\left(-{D_1}^m\frac{\partial}{\partial r}r\frac{\partial\phi_1}{\partial r}-{D_1}^m r\frac{\partial^2\phi_1}{\partial z^2}+{\Sigma_{R1}}^m\phi_1 r\right)dr$$

$$+\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_{r_i}^{r_i+\frac{\Delta r_n}{2}}\left(-{D_1}^n\frac{\partial}{\partial r}r\frac{\partial\phi_1}{\partial r}-{D_1}^n r\frac{\partial^2\phi_1}{\partial z^2}+{\Sigma_{R1}}^n\phi_1 r\right)dr\ =$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_{r_i-\frac{\Delta r_m}{2}}^{r_i}\left(\frac{1}{k}\left[{\nu_1}^m{\Sigma_{f1}}^m\phi_1+{\nu_2}^m{\Sigma_{f2}}^m\phi_2\right]r\right)dr+,$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}}dz\int_{r_i}^{r_i+\frac{\Delta r_n}{2}}\left(\frac{1}{k}\left[{\nu_1}^n{\Sigma_{f1}}^n\phi_1+{\nu_2}^n{\Sigma_{f2}}^n\phi_2\right]r\right)dr \quad (13)$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r_m}{2}}^{r_i} \left(-{D_1}^m \frac{\partial}{\partial r} r \frac{\partial \phi_1}{\partial r} - {D_1}^m r \frac{\partial^2 \phi_1}{\partial z^2} + {\Sigma_a}^m \phi_1 r\right) dr$$

$$+ \int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i}^{r_i+\frac{\Delta r_n}{2}} \left(-{D_1}^n \frac{\partial}{\partial r} r \frac{\partial \phi_1}{\partial r} - {D_1}^n r \frac{\partial^2 \phi_1}{\partial z^2} + {\Sigma_a}^n \phi_1 r\right) dr \; =$$

$$\int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i-\frac{\Delta r_m}{2}}^{r_i} (\Sigma_{s12} \phi_1 r) dr + \int_{z_k-\frac{\Delta z}{2}}^{z_k+\frac{\Delta z}{2}} dz \int_{r_i}^{r_i+\frac{\Delta r_n}{2}} (\Sigma_{s12} \phi_1 r) dr \qquad (14)$$

After applying the initial conditions (equation 12) needed to discretize a mesh point node, equations 13 and 14 are developed in the exact same method as the interior nodes from project 1.

Fast:

$${D_1}^m \left(r_i - \frac{\Delta r_m}{2}\right) \Delta z \frac{\left({\phi_{i,k}}^1 - {\phi_{i-1,k}}^1\right)}{\Delta r_m} - {D_1}^m \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) \Delta z \frac{\left({\phi_{i,k+1}}^1 - {\phi_{i,k}}^1\right)}{\Delta z} + {D_1}^m \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) \frac{\left({\phi_{i,k}}^1 - {\phi_{i,k-1}}^1\right)}{\Delta z} + {\Sigma_{R1}}^m {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) - {D_1}^n \left(r_i + \frac{\Delta r_n}{2}\right) \Delta z \frac{\left({\phi_{i+1,k}}^1 - {\phi_{i,k}}^1\right)}{\Delta r_n} - {D_1}^n \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) \Delta z \frac{\left({\phi_{i,k+1}}^1 - {\phi_{i,k}}^1\right)}{\Delta z} + {D_1}^n \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) \frac{\left({\phi_{i,k}}^1 - {\phi_{i,k-1}}^1\right)}{\Delta z} + {\Sigma_{R1}}^n {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) = \frac{1}{k} {\nu_1}^m {\Sigma_{f1}}^m {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) + \frac{1}{k} {\nu_2}^m {\Sigma_{f2}}^m {\phi_{i,k}}^2 \Delta z \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) + \frac{1}{k} {\nu_1}^n {\Sigma_{f1}}^n {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) + \frac{1}{k} {\nu_2}^n {\Sigma_{f2}}^n {\phi_{i,k}}^2 \Delta z \left(\frac{r_i \Delta r_n}{2} - \frac{{\Delta r_n}^2}{8}\right) \qquad (15)$$

Thermal:

$${D_1}^m \left(r_i - \frac{\Delta r_m}{2}\right) \Delta z \frac{\left({\phi_{i,k}}^1 - {\phi_{i-1,k}}^1\right)}{\Delta r_m} - {D_1}^m \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) \Delta z \frac{\left({\phi_{i,k+1}}^1 - {\phi_{i,k}}^1\right)}{\Delta z} + {D_1}^m \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) \frac{\left({\phi_{i,k}}^1 - {\phi_{i,k-1}}^1\right)}{\Delta z} + {\Sigma_a}^m {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) - {D_1}^n \left(r_i + \frac{\Delta r_n}{2}\right) \Delta z \frac{\left({\phi_{i+1,k}}^1 - {\phi_{i,k}}^1\right)}{\Delta r_n} - {D_1}^n \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) \Delta z \frac{\left({\phi_{i,k+1}}^1 - {\phi_{i,k}}^1\right)}{\Delta z} + {D_1}^n \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) \frac{\left({\phi_{i,k}}^1 - {\phi_{i,k-1}}^1\right)}{\Delta z} + {\Sigma_a}^n {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right) = {\Sigma_{s12}}^m {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_m}{2} - \frac{{\Delta r_m}^2}{8}\right) + {\Sigma_{s12}}^n {\phi_{i,k}}^1 \Delta z \left(\frac{r_i \Delta r_n}{2} + \frac{{\Delta r_n}^2}{8}\right)$$

(16)

The numerical solution then to the nodal model of the discretized cylinder can be found by creating matrices of finite difference solutions for varying number of nodes using equations 6, 7, 10, 11, 15, and 16, and the exterior flux conditions for $r = +R$ and $z = \pm H/2$. The solution of the flux as a function of r and z within the cylinder can be found by employing the five-point stencil for each node and the following equations:

$$\bar{\bar{A}}_1\bar{\phi}_1 = \frac{1}{k}\left[\nu\bar{\bar{\Sigma}}_{f1}\bar{\phi}_1 + \nu\bar{\bar{\Sigma}}_{f2}\bar{\phi}_2\right] \qquad \textbf{Fast} \tag{17}$$

$$\bar{\bar{A}}_2\bar{\phi}_2 = \bar{\bar{\Sigma}}_{s12}\bar{\phi}_1 \qquad \textbf{Thermal} \tag{18}$$

where $\bar{\bar{\boldsymbol{A}}}_{\boldsymbol{1}}$ and $\bar{\bar{\boldsymbol{A}}}_{\boldsymbol{2}}$ are matrices of equations for the fast and thermal fluxes at each node, $\boldsymbol{\nu\bar{\bar{\Sigma}}_{f1}}$ and $\boldsymbol{\nu\bar{\bar{\Sigma}}_{f2}}$ are fast and thermal fission source matrices, $\bar{\bar{\boldsymbol{\Sigma}}}_{\boldsymbol{s12}}$ a matrix of scattering energies, and $\vec{\phi}_1$ and $\vec{\phi}_2$are vectors of fast and thermal flux solutions to be solved for. Since the solutions for the fluxes are non-linear, it becomes necessary to solve for both fluxes (ϕ_1 and ϕ_2) and the multiplication factor (k) iteratively through the power iteration method.

To begin to solve equation 17, an initial guess for k_0 and S_0 were implemented into equation 17 to solve for ϕ_1^1. ϕ_1^1 was then placed into equation 18 to solve for ϕ_2^1, where k_1 and S_1 could then be determined. The new k and S values were then placed back into equation 3 to solve for ϕ_1^2 and subsequently ϕ_2^2 to find k_2 and S_2. This iteration process was continued until S converged to a value of 0.0001 % of the previous iteration of S.

To solve equation 33 iteratively by following the coordinates of **Figure 2**, the following matrices were used:

Fast Matrix:

$$\begin{bmatrix} c_{11} & c_{21} & & c_{12} & & & & & \\ m_{11} & m_{21} & m_{31} & & a_{22} & & & & \\ & a_{21} & a_{31} & & & a_{32} & & & \\ c_{11} & & & c_{12} & c_{22} & & c_{13} & & \\ & a_{21} & & m_{12} & m_{22} & m_{32} & & a_{23} & \\ & & a_{31} & & a_{22} & a_{32} & & & a_{33} \\ & & & c_{12} & & & c_{13} & a_{23} & \\ & & & & a_{22} & & m_{13} & m_{23} & m_{33} \\ & & & & & a_{32} & & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} {\phi_{11}}^1 \\ {\phi_{21}}^1 \\ {\phi_{31}}^1 \\ {\phi_{12}}^1 \\ {\phi_{22}}^1 \\ {\phi_{32}}^1 \\ {\phi_{13}}^1 \\ {\phi_{23}}^1 \\ {\phi_{33}}^1 \end{bmatrix} = \begin{bmatrix} S(\phi_{11}) \\ S(\phi_{21}) \\ S(\phi_{31}) \\ S(\phi_{12}) \\ S(\phi_{22}) \\ S(\phi_{32}) \\ S(\phi_{13}) \\ S(\phi_{23}) \\ S(\phi_{33}) \end{bmatrix}$$

Left-hand side (19)

$$\begin{bmatrix} S(\phi_{11}) \\ S(\phi_{21}) \\ S(\phi_{31}) \\ S(\phi_{12}) \\ S(\phi_{22}) \\ S(\phi_{32}) \\ S(\phi_{13}) \\ S(\phi_{23}) \\ S(\phi_{33}) \end{bmatrix} = \frac{1}{k} \begin{bmatrix} d_{11} & & & & & & & & \\ & m_{21} & & & & & & & \\ & & b_{31} & & & & & & \\ & & & d_{12} & & & & & \\ & & & & m_{22} & & & & \\ & & & & & b_{32} & & & \\ & & & & & & d_{13} & & \\ & & & & & & & m_{23} & \\ & & & & & & & & b_{33} \end{bmatrix} \begin{bmatrix} {\phi_{11}}^1 \\ {\phi_{21}}^1 \\ {\phi_{31}}^1 \\ {\phi_{12}}^1 \\ {\phi_{22}}^1 \\ {\phi_{32}}^1 \\ {\phi_{13}}^1 \\ {\phi_{23}}^1 \\ {\phi_{33}}^1 \end{bmatrix}$$

$$+ \frac{1}{k} \begin{bmatrix} d_{11} & & & & & & & & \\ & m_{21} & & & & & & & \\ & & b_{31} & & & & & & \\ & & & d_{12} & & & & & \\ & & & & m_{22} & & & & \\ & & & & & b_{32} & & & \\ & & & & & & d_{13} & & \\ & & & & & & & m_{23} & \\ & & & & & & & & b_{33} \end{bmatrix} \begin{bmatrix} {\phi_{11}}^2 \\ {\phi_{21}}^2 \\ {\phi_{31}}^2 \\ {\phi_{12}}^2 \\ {\phi_{22}}^2 \\ {\phi_{32}}^2 \\ {\phi_{13}}^2 \\ {\phi_{23}}^2 \\ {\phi_{33}}^2 \end{bmatrix}$$

Right-hand side (20)

Thermal Matrix:

$$\begin{bmatrix} c_{11} & c_{21} & & c_{12} & & & & & \\ m_{11} & m_{21} & m_{31} & & a_{22} & & & & \\ & a_{21} & a_{31} & & & a_{32} & & & \\ c_{11} & & & c_{12} & c_{22} & & c_{13} & & \\ & a_{21} & & m_{12} & m_{22} & m_{32} & & a_{23} & \\ & & a_{31} & & a_{22} & a_{32} & & & a_{33} \\ & & & c_{12} & & & c_{13} & a_{23} & \\ & & & & a_{22} & & m_{13} & m_{23} & m_{33} \\ & & & & & a_{32} & & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} {\phi_{11}}^2 \\ {\phi_{21}}^2 \\ {\phi_{31}}^2 \\ {\phi_{12}}^2 \\ {\phi_{22}}^2 \\ {\phi_{32}}^2 \\ {\phi_{13}}^2 \\ {\phi_{23}}^2 \\ {\phi_{33}}^2 \end{bmatrix} = \begin{bmatrix} S(\phi_{11}) \\ S(\phi_{21}) \\ S(\phi_{31}) \\ S(\phi_{12}) \\ S(\phi_{22}) \\ S(\phi_{32}) \\ S(\phi_{13}) \\ S(\phi_{23}) \\ S(\phi_{33}) \end{bmatrix}$$

Left-hand side (21)

$$\begin{bmatrix} S(\phi_{11}) \\ S(\phi_{21}) \\ S(\phi_{31}) \\ S(\phi_{12}) \\ S(\phi_{22}) \\ S(\phi_{32}) \\ S(\phi_{13}) \\ S(\phi_{23}) \\ S(\phi_{33}) \end{bmatrix} = \begin{bmatrix} d_{11} & & & & & & & & \\ & m_{21} & & & & & & & \\ & & b_{31} & & & & & & \\ & & & d_{12} & & & & & \\ & & & & m_{22} & & & & \\ & & & & & b_{32} & & & \\ & & & & & & d_{13} & & \\ & & & & & & & m_{23} & \\ & & & & & & & & b_{33} \end{bmatrix} \begin{bmatrix} {\phi_{11}}^1 \\ {\phi_{21}}^1 \\ {\phi_{31}}^1 \\ {\phi_{12}}^1 \\ {\phi_{22}}^1 \\ {\phi_{32}}^1 \\ {\phi_{13}}^1 \\ {\phi_{23}}^1 \\ {\phi_{33}}^1 \end{bmatrix}$$

Right-hand side (22)

The indices *a* and *b* refer to the internal node discretization equations 6 and 7, *c* and *d* refer to the central node discretization equations 10 and 11, while *m* refers to the mesh point discretization equations 15 and 16 . Equations 19 through 22 are then used in conjunction with the power iteration method to solve for the fast and thermal fluxes and multiplication factor for any number of nodes within a finite cylinder geometry of multiplying medium.

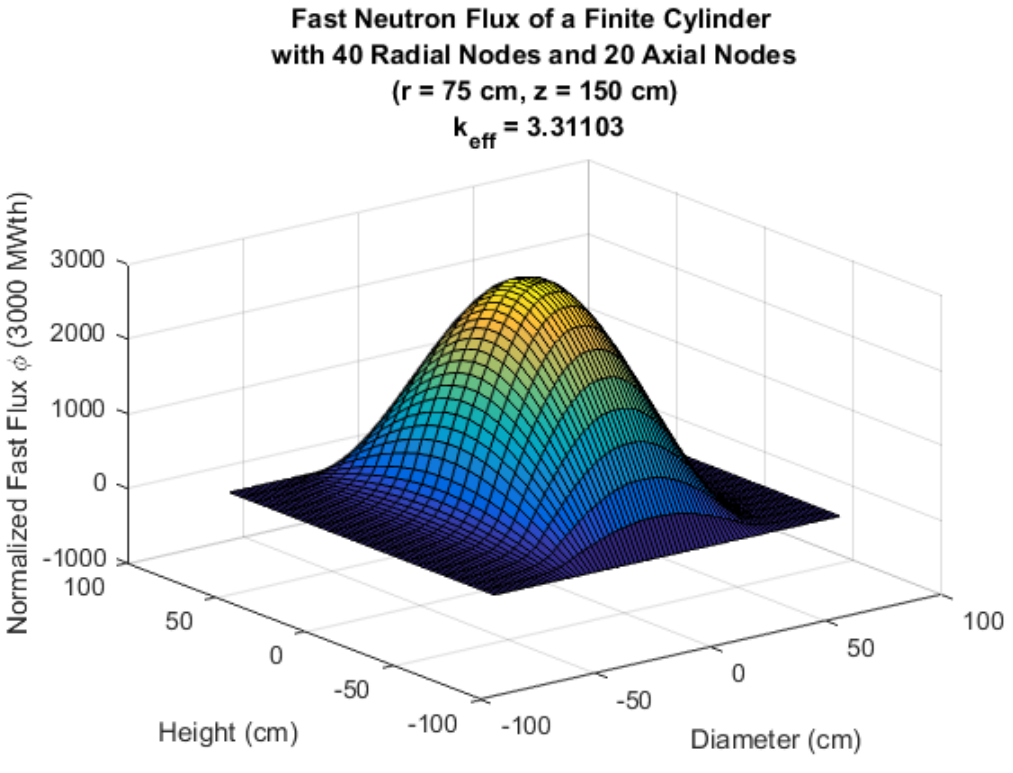
The program codes will be tested using low, medium, and high number of nodes (*N*, *M*) so as to provide estimates on the calculation time required to solve each case scenario. Once a discretization scheme (i.e., the number of nodes needed for a converged solution), the program

code will then be used to calculate the critical dimensions of the core with no reflector, as well as the "reflector savings" from a critical core configuration. In order to calculate the critical core dimensions, both the core and reflector regions will be given the same material properties to establish a solid core with no reflector. The program code will then introduce an iteration process in which the radius and height of the cylinder will be incrementally increased by a specified distance until k_{eff} converges to 1. The result will then provide the critical dimensions of the core given the material properties provided. Following this calculation, the critical core dimensions will then be used as a starting point to search for the reflector length to create "reflector savings" around the core. What this means is that the length of the reflector will be increased until the value of k_{eff} does not change by a specified epsilon value. The program will again perform incremental increases until the value of k_{eff} converges.

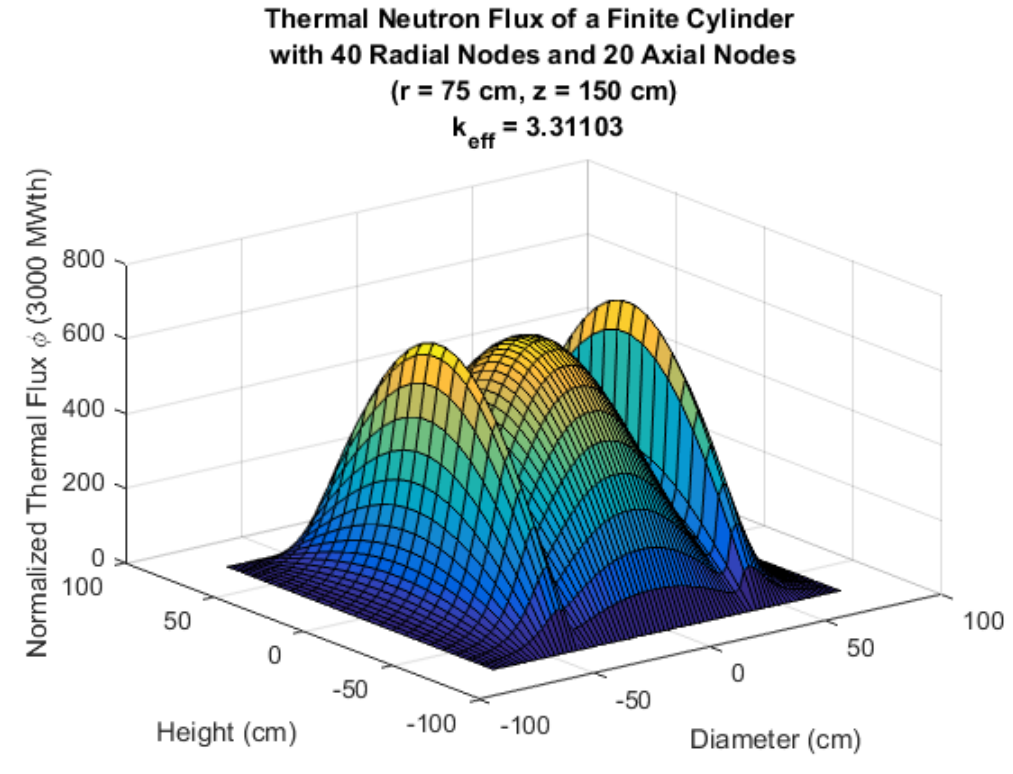
Results and Discussion

The purpose of the project was to use numerical methods to develop a program to solve for the neutron flux distribution, critical core dimension, reflector savings, and multiplication factor (k) using two-group diffusion theory given a variety of differing conditions from a typical LWR. As the plots in **Appendix A** of various nodal sizes (low, medium, and high) reveals, as the number of nodes increases k_{eff} increases as the core is more finely discretized. **Figure 4** presents the 20-nodal solution for the fast and thermal calculated fluxes for a given core dimensions. Both plots present the characteristic shapes expected for fast and thermal neutron fluxes. **Figure 5** presents flux profiles of a critical core with equal core and reflector dimensions (9.34 cm each) and a height (37.36 cm) that is twice the length of the complete core radius. All fluxes used throughout the project have been normalized to a core power of 3000 MWth. Between the plots of 10 and 20 nodes, there is a 1.200% difference in k_{eff} values, and between 20

and 50 nodes, there is only a 0.568% difference. For the purposes of the rest of the project 20 nodes were used to effectively present all relevant data results. As for the time estimates for the program to solve each nodal solution, again 20 nodes maximized the efficiency of the program needing only 1.505719 seconds to complete the solution. The 10-nodal solution was noted to be 0.079106 seconds and the 50-nodal solution at 171.670133 seconds. Once the number of nodes increased beyond 20 the efficiency of the program began to slow and only a minimal change in k_{eff} was observed.

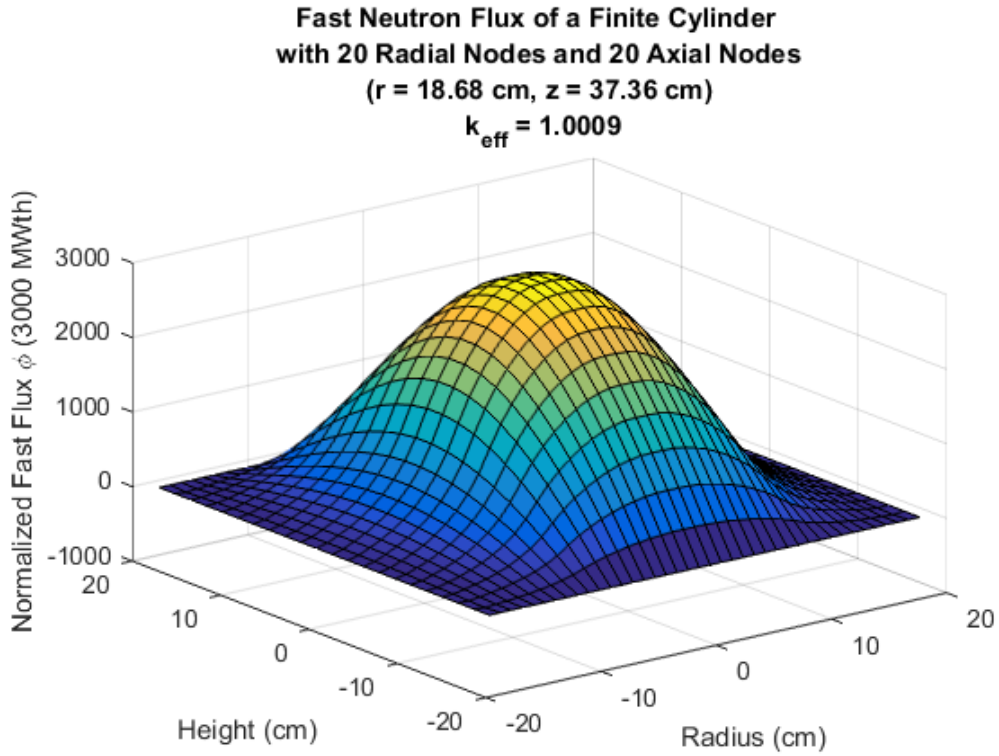


(a)

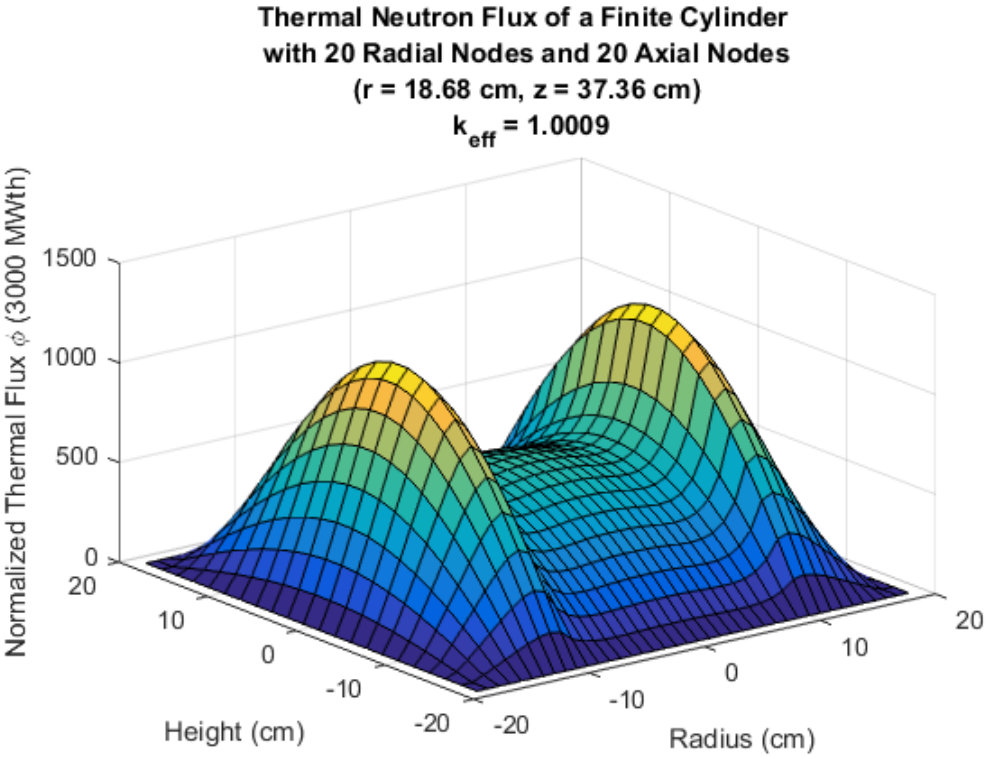


(b)

Figure 4: Fast (a) and thermal (b) fluxes for 20-node solution.



(a)

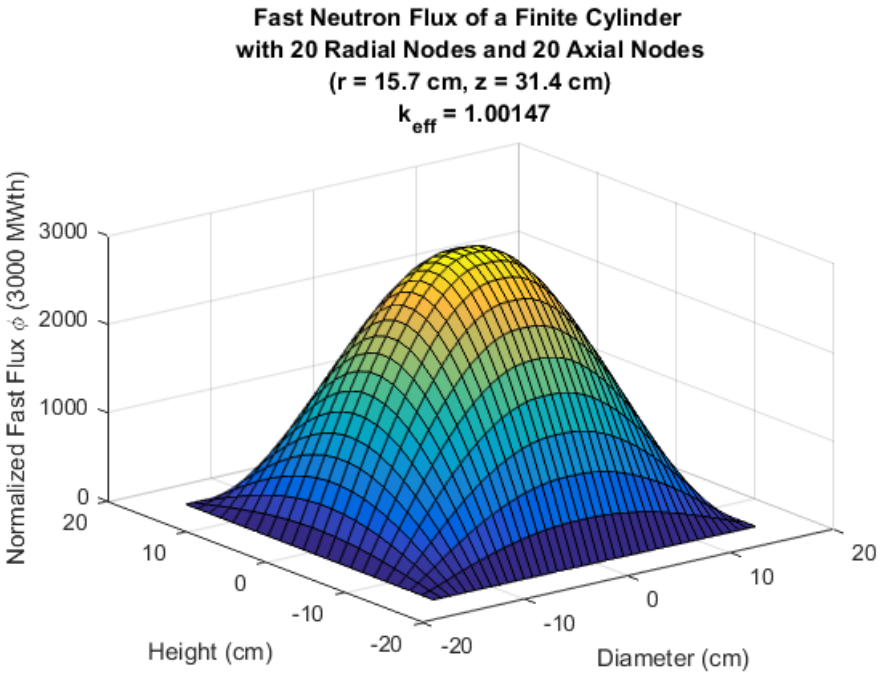


(b)

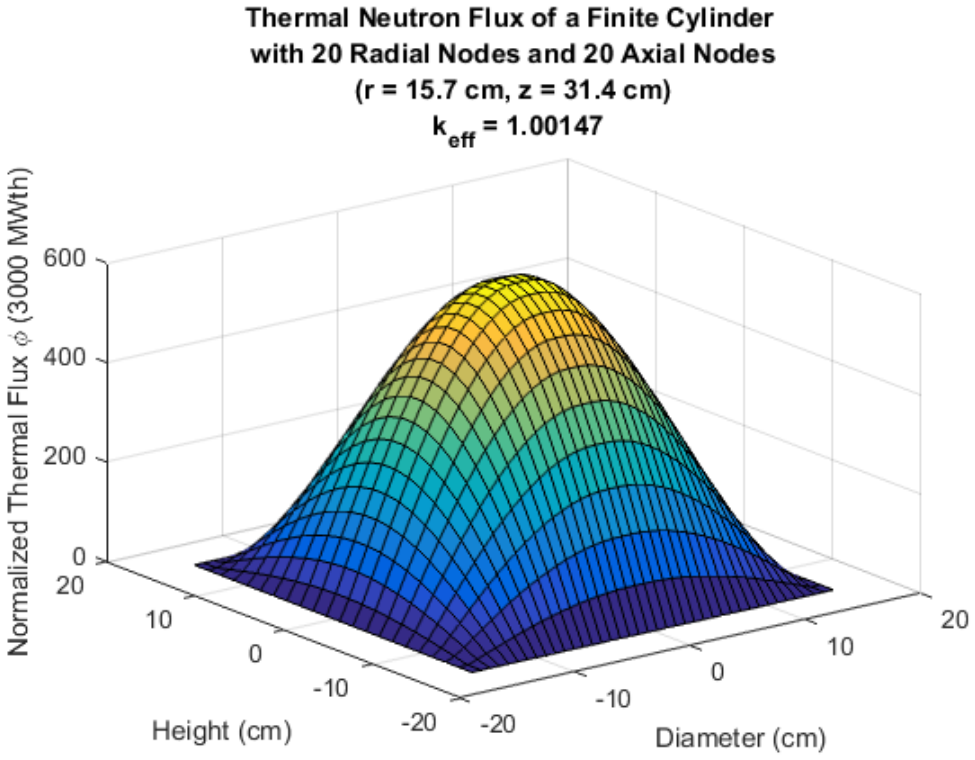
Figure 5: Fast (a) and thermal (b) fluxes for 20-node critical core.

In order to calculate the critical dimensions of a homogeneous core with no reflector and assuming that $R = H/2$, an iterative loop was created to increase the radius and height of the core by an amount of 0.01 cm until k_{eff} was approximately 1. As mentioned above, a 20-node calculation was employed given its speed and convergence efficiency. As **Figure 6** reveals the critical dimensions of the core were found to have a radius of 15.7 cm, a height of 31.4 cm, and $k_{eff} = 1.00147$. Both fast and thermal fluxes maintain the same shape as thermal neutrons are not being reflected back into the core as would be expected with a reflector. While this configuration of the core is slightly supercritical, it does show how the program successfully increased the radius and height until the dimensions of an approximately critical core was found. The program calculation time was observed to be 33.152657 seconds.

Once the critical dimensions of the core were established, the next step the calculation of a reflector thickness that yielded a maximum possible "reflector savings" for a critical core. For each iteration, only the width of the reflector would increase by a value of 0.1 cm until k_{eff} between successive iterations did not change by 0.0001. **Figure 7** reveals that an 87.3 cm thick reflector has to be added to the core until k_{eff} no longer changes by the desired epsilon value. Both the characteristic fast and thermal flux profiles can be clearly observed. **Figure 8** presents a plot of k_{eff} as a function of increase in reflector width. Normally, this plot would be inverted, exponentially increasing to an asymptotic k value; however, given some problems with the code, the plot exponentially decreases to an asymptotic k value. Most likely there is a problem with either the flux or k values not converging properly in the code. However, given this setup, the code did successfully increase the reflector width until k_{eff} reached a desired convergence epsilon. The program calculation time was observed to be 254.058338 seconds.

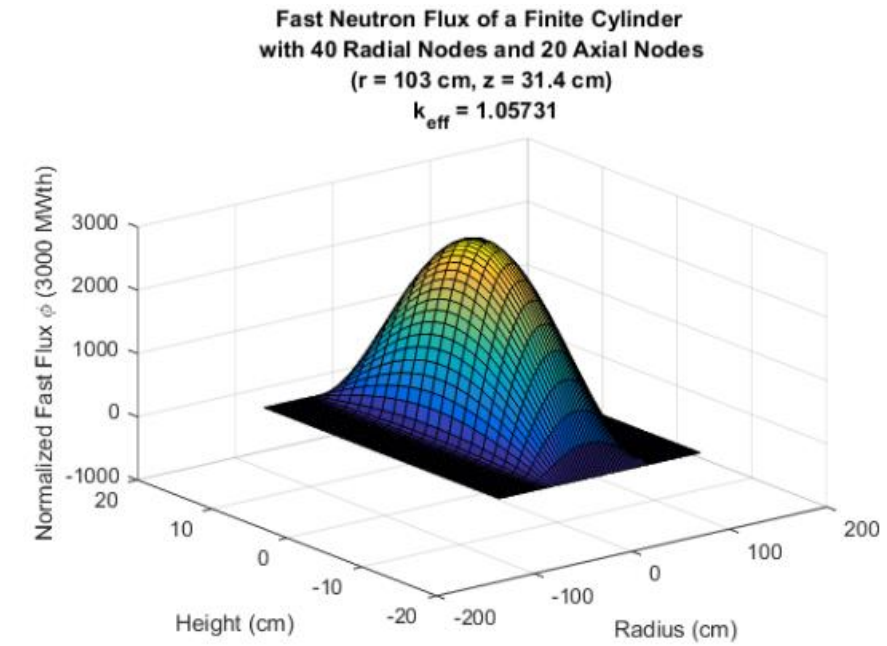


(a)

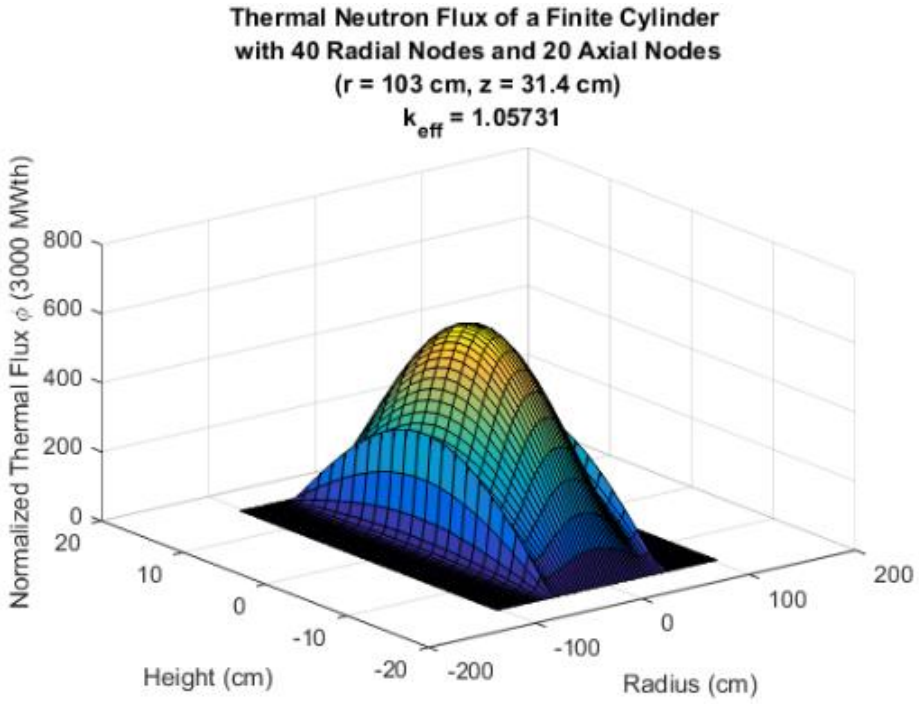


(b)

Figure 6: Fast (a) and thermal (b) fluxes for 20-node critical core solution



(a)



(b)

Figure 7: Fast (a) and thermal (b) fluxes for a critical core with increased reflector width

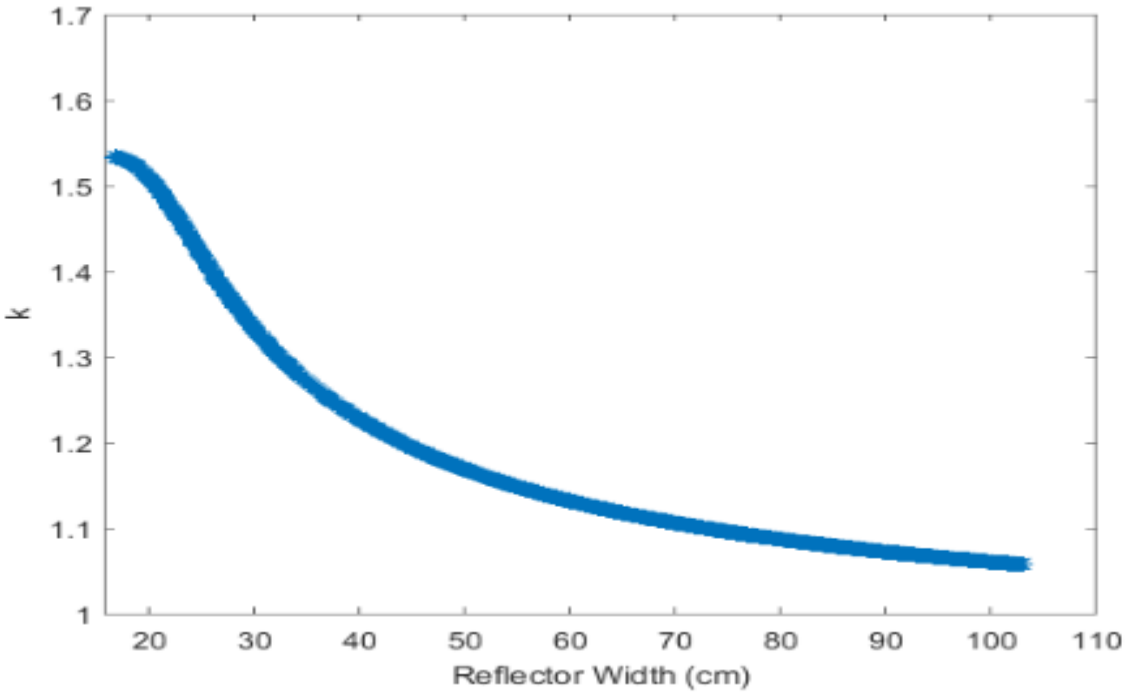
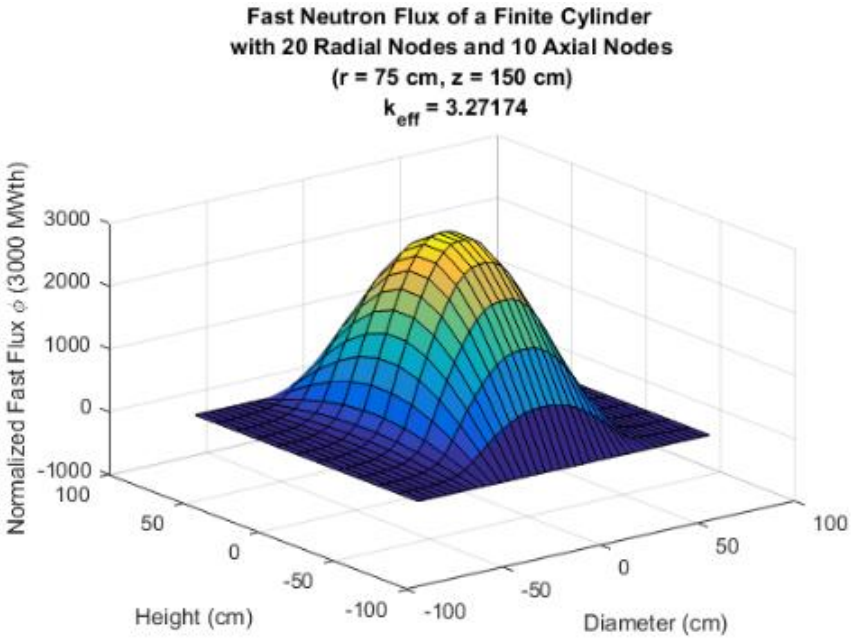
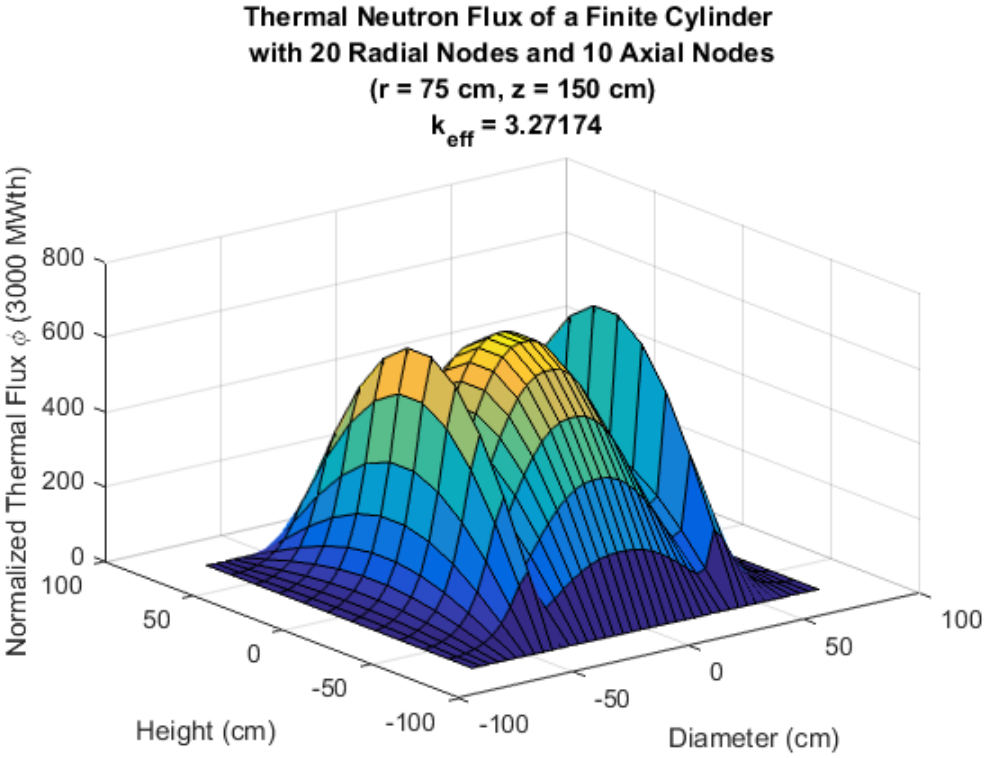


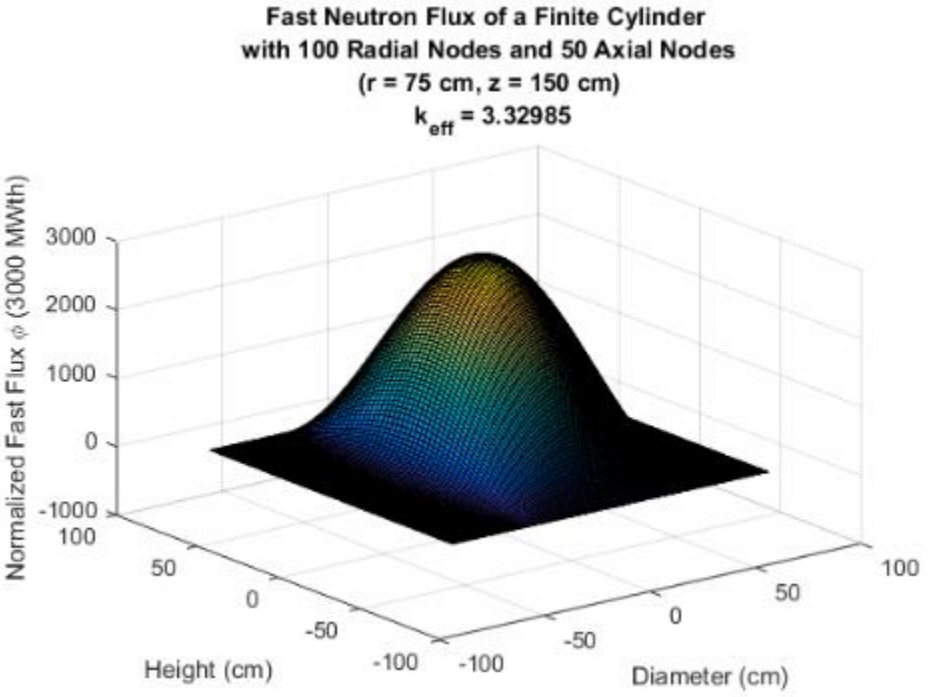
Figure 8: Plot of *k* versus reflector width

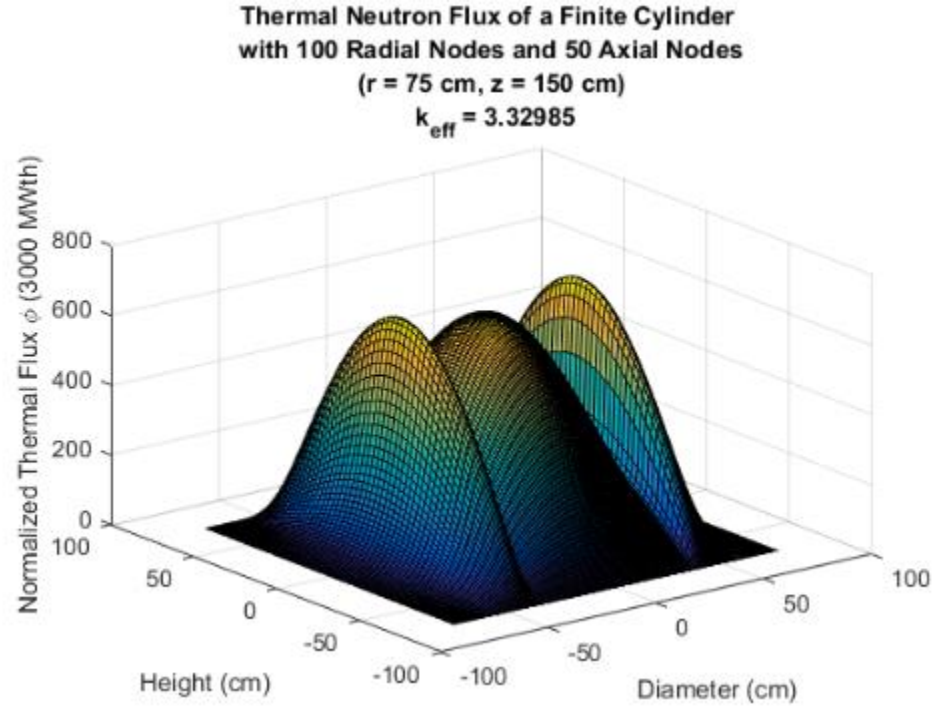
Appendix A

Plots of Numerical Solutions at various input parameters:






Fast Neutron Flux of a Finite Cylinder
with 100 Radial Nodes and 50 Axial Nodes
(r = 75 cm, z = 150 cm)
k_eff = 3.32985
Normalized Fast Flux ϕ (3000 MWth)
3000
2000
1000
0
-1000
100
50
0
-50
-100
Height (cm)
-100
-50
0
50
100
Diameter (cm)


Thermal Neutron Flux of a Finite Cylinder
with 100 Radial Nodes and 50 Axial Nodes
(r = 75 cm, z = 150 cm)
k_eff = 3.32985
Normalized Thermal Flux ϕ (3000 MWth)
800
600
400
200
0
100
50
0
-50
-100
Height (cm)
-100
-50
0
50
100
Diameter (cm)

Appendix B

Program Code

```
% Project 2 - NE 571

% Project code contains programming for iteration loops to solve
for the
% critical core dimensions and reflector savings.  All other
calculations
% are completed with the commented out sections.

% Numerical Solution

clearvars; close all; clc
%%
%  Core System Properties [Fast Thermal]
vSigf = [0.008476 0.18514];
D = [1.2627 0.3543];
SigR = 0.012627;
SigA = 0.1210;
SigS = 0.02619;

% Reflector System Properties
% vSigfw = [0.008476 0.18514];% - These commented out properties
are for the critical search
% Dw = [1.2627 0.3543];
% SigRw = 0.012627;
% SigAw = 0.1210;
% SigSw = 0.02619;
vSigfw = [0 0];
Dw = [1.13 0.16];
SigRw = 0.0494;
SigAw = 0.0197;
SigSw = 0.0494;
%%
% Numerical Solution
C = input('Input the number of nodes in the radial direction for
the core:');
W = input('Input the number of nodes in the radial direction for
the reflector:')-1;
N = C + W;
M = input('Input the number of nodes in the axial direction:');
Rc = input('Input the radius width of the core (cm):');
Rw = input('Input the radius width of the reflector (cm):');
R = Rc + Rw;
Z = input('Input the height of the cylinder (cm):');
```

```
%%
tic
% Critical core search
% k = 0;
% eps = 0.0001;
% count = 0;
%%
% % Reflector savings search
% oldk2 = 0;
% kdiff = 1;
%%

% While loops for critical core and reflector savings search

% while k < 1 % Critical core search

% % Critical Core Search
% Rc = Rc + 0.01;
% Rw = Rw + 0.01;
% R = Rc + Rw;
% Z = 2.*R;

% while kdiff > eps % Reflector savings search

% count = count + 1;

% % Reflector Savings
% Rw = Rw + 0.1;
% R = Rc + Rw;

Af = zeros(N.*M); % A matrices
Ath = zeros(N.*M);
Bf1 = zeros(N.*M); % B matices
Bf2 = zeros(N.*M);
Bth = zeros(N.*M);

drc = Rc./C; % delta r-core
drw = Rw./W; % delta r-reflector
drc2 = drc.*drc;
drw2 = drw.*drw;
ric = 2.*drc;
riw = 2.*drw;
dz = Z./M; % delta z
dz2 = dz.*dz;

% Defining A fast matrix: Central diagonal
```

```
for i = 1:N:N*M
    for j = 1:N
        if j == 1
            Af(i+(j-1), i+(j-1)) = SigR(1).*((drc.^3)./8).*dz2 +
2.*D(1).*((drc.^3)./8) + D(1).*(drc./2).*dz2;
        elseif j < C
            Af(i+(j-1), i+(j-1)) = SigR(1).*(ric.*(j-
1)).*drc2.*dz2 + 2.*D(1).*(ric.*(j-1)).*drc2 +
2.*D(1).*(ric.*(j-1)).*dz2;
        elseif j == C
            Af(i+(j-1), i+(j-1)) = (D(1)./drc).*((ric.*(j-1))-
(drc./2)).*dz + 2.*(D(1)./dz).*((((ric.*(j-1)).*drc)./2)-
(drc2./8)) ...
                + SigR(1).*dz.*((((ric.*(j-1)).*drc)./2)-
(drc2./8)) + (Dw(1)./drw).*(riw.*(j-1)+(drw./2)).*dz ...
                + 2.*((Dw(1)./dz).*(((riw.*(j-
1).*drw)./2)+(drw2./8))) + SigRw(1).*dz.*(((riw.*(j-
1).*drw)./2)+(drw2./8));
        elseif j > C
            Af(i+(j-1), i+(j-1)) = SigRw(1).*(riw.*(j-
1)).*drw2.*dz2 + 2.*Dw(1).*(riw.*(j-1)).*drw2 +
2.*Dw(1).*(riw.*(j-1)).*dz2;
        end
    end
end

for i = 1:N:N*M % Upper and lower central diagonals
    for j = 1:N
        if j == 1
            Af(i+(j-1), i+((j-1)+1)) = -D(1).*((drc./2).*dz2);
        elseif j < C
            Af(i+(j-1), i+((j-1)+1)) = -D(1).*((ric.*(j-
1))+(drc./2)).*dz2;
            Af(i+(j-1), i+(j-1)-1) = -D(1).*((ric.*(j-1))-
(drc./2)).*dz2;
        elseif j == C
            Af(i+(j-1), i+((j-1)+1)) = (-Dw(1)./drw).*(riw.*(j-
1)+(drw./2)).*dz;
            Af(i+(j-1), i+(j-1)-1) = (-D(1)./drc).*((ric.*(j-
1))-(drc./2)).*dz;
        elseif j > C  && j < N
            Af(i+(j-1), i+((j-1)+1)) = -Dw(1).*((riw.*(j-
1))+(drw./2)).*dz2;
            Af(i+(j-1), i+(j-1)-1) = -Dw(1).*((riw.*(j-1))-
(drw./2)).*dz2;
        elseif j == N
```

```
            Af(i+(j-1), i+(j-1)-1) = -Dw(1).*((riw.*(j-1))-
(drw./2)).*dz2;
        end
    end
end

for i = 1:N:(N*M)-N % Outside upper diagonal
    for j = 1:N
        if j == 1
            Af(i+(j-1), i+((j-1)+N)) = -D(1).*((drc.^3)./8);
        elseif j < C
            Af(i+(j-1), i+((j-1)+N)) = -D(1).*((ric.*(j-
1)).*drc2);
        elseif j == C
            Af(i+(j-1), i+((j-1)+N)) = (-D(1)./dz).*((((ric.*(j-
1)).*drc)./2)-(drc2./8)) - (Dw(1)./dz).*(((riw.*(j-
1).*drw)./2)+(drw2./8));
        elseif j > C  && j < N
            Af(i+(j-1), i+((j-1)+N)) = -Dw(1).*((riw.*(j-
1)).*drw2);
            m = m+1;
        elseif j == N
            Af(i+(j-1), i+((j-1)+N)) = -Dw(1).*((riw.*(j-
1)).*drw2);
        end
    end
end

for i = 1:N:(N*M)-N % Outside lower diagonal
    for j = 1:N
        if j == 1
            Af(i+((j-1)+N), i+(j-1)) = -D(1).*((drc.^3)./8);
        elseif j < C
            Af(i+((j-1)+N), i+(j-1)) = -D(1).*(ric.*(j-
1)).*drc2;
        elseif j == C
            Af(i+((j-1)+N), i+(j-1)) = (-D(1)./dz).*((((ric.*(j-
1)).*drc)./2)-(drc2./8)) - (Dw(1)./dz).*(((riw.*(j-
1).*drw)./2)+(drw2./8));
        elseif j > C  && j < N
            Af(i+((j-1)+N), i+(j-1)) = -Dw(1).*(riw.*(j-
1)).*drw2;
        elseif j == N
            Af(i+((j-1)+N), i+(j-1)) = -Dw(1).*(riw.*(j-
1)).*drw2;
        end
    end
```

```
end

% Defining A thermal matrix
for i = 1:N:N*M % Central diagonal
    for j = 1:N
        if j == 1
            Ath(i+(j-1), i+(j-1)) = SigA.*((drc.^3)./8).*dz2 +
2.*D(2).*((drc.^3)./8) + D(2).*(drc./2).*dz2;
        elseif j < C
            Ath(i+(j-1), i+(j-1)) = SigA.*(ric.*(j-
1)).*drc2.*dz2 + 2.*D(2).*(ric.*(j-1)).*drc2 +
2.*D(2).*(ric.*(j-1)).*dz2;
        elseif j == C
            Ath(i+(j-1), i+(j-1)) = (D(2)./drc).*((ric.*(j-1))-
(drc./2)).*dz + 2.*(D(2)./dz).*((((ric.*(j-1)).*drc)./2)-
(drc2./8)) ...
                + SigA.*dz.*((((ric.*(j-1)).*drc)./2)-(drc2./8))
+ (Dw(2)./drw).*(riw.*(j-1)+(drw./2)).*dz ...
                + 2.*((Dw(2)./dz).*(((riw.*(j-
1).*drw)./2)+(drw2./8))) + SigAw.*dz.*(((riw.*(j-
1).*drw)./2)+(drw2./8));
        elseif j > C
            Ath(i+(j-1), i+(j-1)) = SigAw.*(riw.*(j-
1)).*drw2.*dz2 + 2.*Dw(2).*(riw.*(j-1)).*drw2 +
2.*Dw(2).*(riw.*(j-1)).*dz2;
        end
    end
end

for i = 1:N:N*M % Upper and lower central diagonal
    for j = 1:N
        if j == 1
            Ath(i+(j-1), i+((j-1)+1)) = -D(2).*((drc./2).*dz2);
        elseif j < C
            Ath(i+(j-1), i+((j-1)+1)) = -D(2).*((ric.*(j-
1))+(drc2./2)).*dz2;
            Ath(i+(j-1), i+(j-1)-1) = -D(2).*((ric.*(j-1))-
(drc2./2)).*dz2;
        elseif j == C
            Ath(i+(j-1), i+((j-1)+1)) = (-Dw(2)./drw).*(riw.*(j-
1)+(drw./2)).*dz;
            Ath(i+(j-1), i+(j-1)-1) = (-D(2)./drc).*((ric.*(j-
1))-(drc./2)).*dz;
        elseif j > C  && j < N
            Ath(i+(j-1), i+((j-1)+1)) = -Dw(2).*((riw.*(j-
1))+(drw2./2)).*dz2;
```

```
            Ath(i+(j-1), i+(j-1)-1) = -Dw(2).*((riw.*(j-1))-
(drw2./2)).*dz2;
        elseif j == N
            Ath(i+(j-1), i+(j-1)-1) = -Dw(2).*((riw.*(j-1))-
(drw2./2)).*dz2;
        end
    end
end

for i = 1:N:(N*M)-N % Outside upper diagonal
    for j = 1:N
        if j == 1
            Ath(i+(j-1), i+((j-1)+N)) = -D(2).*((drc.^3)./8);
        elseif j < C
            Ath(i+(j-1), i+((j-1)+N)) = -D(2).*((ric.*(j-
1)).*drc2);
        elseif j == C
            Ath(i+(j-1), i+((j-1)+N)) = (-
D(2)./dz).*((((ric.*(j-1)).*drc)./2)-(drc2./8)) -
(Dw(2)./dz).*(((riw.*(j-1).*drw)./2)+(drw2./8));
        elseif j > C  && j < N
            Ath(i+(j-1), i+((j-1)+N)) = -Dw(2).*((riw.*(j-
1)).*drw2);
        elseif j == N
            Ath(i+(j-1), i+((j-1)+N)) = -Dw(2).*((riw.*(j-
1)).*drw2);
        end
    end
end

for i = 1:N:(N*M)-N % Outside lower diagonal
    for j = 1:N
        if j == 1
            Ath(i+((j-1)+N), i+(j-1)) = -D(2).*((drc.^3)./8);
        elseif j < C
            Ath(i+((j-1)+N), i+(j-1)) = -D(2).*(ric.*(j-
1)).*drc2;
        elseif j == C
            Ath(i+((j-1)+N), i+(j-1)) = (-
D(2)./dz).*((((ric.*(j-1)).*drc)./2)-(drc2./8)) -
(Dw(2)./dz).*(((riw.*(j-1).*drw)./2)+(drw2./8));
        elseif j > C  && j < N
            Ath(i+((j-1)+N), i+(j-1)) = -Dw(2).*(riw.*(j-
1)).*drw2;
        elseif j == N
            Ath(i+((j-1)+N), i+(j-1)) = -Dw(2).*(riw.*(j-
1)).*drw2;
```

```
        end
    end
end

% Defining B fast 1 matrix
for i = 1:N:N*M
    for j = 1:N
        if j == 1
            Bf1(i+(j-1), i+(j-1)) =
vSigf(1).*((drc.^3)./8).*dz2;
        elseif j < C
            Bf1(i+(j-1), i+(j-1)) = ((ric.*(j-
1)).*drc2.*dz2).*vSigf(1);
        elseif j == C
            Bf1(i+(j-1), i+(j-1)) = (vSigf(1).*dz.*(((((ric.*(j-
1)).*drc)./2)-(drc2./8)) + vSigfw(1).*dz.*((((riw.*(j-
1).*drw)./2)+(drw2./8)))));
        elseif j > C
            Bf1(i+(j-1), i+(j-1)) = (riw.*(j-
1)).*drw2.*dz2.*vSigfw(1);
        end
    end
end

% Defining B fast 2 matrix
for i = 1:N:N*M
    for j = 1:N
        if j == 1
            Bf2(i+(j-1), i+(j-1)) =
(((drc.^3)./8).*dz2).*vSigf(2);
        elseif j < C
            Bf2(i+(j-1), i+(j-1)) = ((ric.*(j-
1)).*drc2.*dz2).*vSigf(2);
        elseif j == C
            Bf2(i+(j-1), i+(j-1)) = (vSigf(2).*dz.*((((ric.*(j-
1)).*drc)./2)-(drc2./8))) + vSigfw(2).*dz.*((((riw.*(j-
1).*drw)./2)+(drw2./8)));
        elseif j > C
            Bf2(i+(j-1), i+(j-1)) = (riw.*(j-
1)).*drw2.*dz2.*vSigfw(2);
        end
    end
end

% Defining B thermal matrix
for i = 1:N:N*M
    for j = 1:N
```

```
        if j == 1
            Bth(i+(j-1), i+(j-1)) = SigS.*dz2.*((drc.^3)./8);
        elseif j < C
            Bth(i+(j-1), i+(j-1)) = SigS.*(ric.*(j-
1)).*drc2.*dz2;
        elseif j == C
            Bth(i+(j-1), i+(j-1)) = SigS.*dz.*((((ric.*(j-
1)).*drc)./2)-(drc2./8)) + SigSw.*dz.*(((riw.*(j-1).*drw)./2) +
(drw2./8));
        elseif j > C
            Bth(i+(j-1), i+(j-1)) = SigSw.*(riw.*(j-
1)).*drw2.*dz2;
        end
    end
end
%%
% Initial guess for Flux, k-eff, and Source
phi1 = ones(N*M,1);
phi2 = ones(N*M,1);

k = 1; % Initial k guess
S = (1./k).*((Bf1*phi1) + (Bf2*phi2)); % Initial fast-source
guess
phi1diff = 1.0; % Initial flux difference
kdiff = 1.0; % Initial flux difference

% Iterations to solve for flux, k-eff, and S
while(phi1diff > 0.00001) % || kdiff > 0.00001)
    oldk = k;
    oldphi1 = phi1;
    oldS = S;

    phi1 = Af\S;
    phi2 = Ath\(Bth*phi1);
    S = (1./oldk).*((Bf1*phi1) + (Bf2*phi2));
    k = oldk.*(sum(S)./sum(oldS));

    % Calculating the flux difference between successive
interations
    phi1diff = 0;
    for i = 1:N*M
        phi1diff = phi1diff + (phi1(i) - oldphi1(i)).^2;
    end
    phi1diff = sqrt(phi1diff);

%     % Calculating the k difference between successive
interations (k - convergence)
```

```
%     kdiff = 0;
%     for i = 1:N*M
%         kdiff = kdiff + (k - oldk).^2;
%     end
%     kdiff = sqrt(kdiff);
end

% % Reflector savings
% kdiff = abs(k - oldk2);
% oldk2 = k;
%
% kplot(count) = [k];
% Rplot(count) = [R];

% end
%%
% Normalize fluxes (integral of flux set to 3000 MWth)
phi1max = max(phi1);
phi1min = min(phi1);
phi2max = max(phi2);
phi2min = min(phi2);
a = 0;
b = 3000;
phi1norm = a + (phi1-phi2min).*((b-a)./(phi1max-phi2min));
phi2norm = a + (phi2-phi2min).*((b-a)./(phi1max-phi2min));

phi1norm = vec2mat(phi1norm,N);
phi1normprofile = zeros(M+2,N+1);
phi1normprofile((2:M+1),(1:N)) = 1;
Coordinate = find(phi1normprofile == 1);
phi1normprofile(Coordinate) = phi1norm;
mirrorphi1norm = fliplr(phi1normprofile(:,1:end));
totalphi1 = horzcat(mirrorphi1norm,phi1normprofile);

phi2norm = vec2mat(phi2norm,N);
phi2normprofile = zeros(M+2,N+1);
phi2normprofile((2:M+1),(1:N)) = 1;
Coordinate2 = find(phi2normprofile == 1);
phi2normprofile(Coordinate2) = phi2norm;
mirrorphi2norm = fliplr(phi2normprofile(:,1:end));
totalphi2 = horzcat(mirrorphi2norm,phi2normprofile);

toc

%%
% Plot of reflector savings
plot(Rplot,kplot,'*')
```

```
xlabel('Reflector Width (cm)')
ylabel('k')
xlim([Rc 110])
ylim([1 1.7])
set(gcf,'InvertHardcopy','on');
print('Project2_NE571_Reflector_Savings_20_nodes_k_eff','-
dpng','-r100');
%%
% Plotting for fast neutron flux
Ri = linspace(-R,R,2.*(N+1));
Zk = linspace(-Z./2,Z./2,M+2);
surf(Ri,Zk,totalphi1);
zlabel('Normalized Fast Flux \phi (3000 MWth)')
xlabel('Radius (cm)')
ylabel('Height (cm)')
t1 = sprintf('Fast Neutron Flux of a Finite Cylinder \n with %d
Radial Nodes and %d Axial Nodes \n (r = %0.5g cm, z = %0.5g cm)
\n k_{eff} = %0.6g',N+1,M,R,Z,k);
title(t1)
set(gcf,'InvertHardcopy','on');
print('Project2_NE571_TestCode','-dpng','-r100');

%%
% Plotting for thermal neutron flux
surf(Ri,Zk,totalphi2);
zlabel('Normalized Thermal Flux \phi (3000 MWth)')
xlabel('Radius (cm)')
ylabel('Height (cm)')
t2 = sprintf('Thermal Neutron Flux of a Finite Cylinder \n with
%d Radial Nodes and %d Axial Nodes \n (r = %0.5g cm, z = %0.5g
cm) \n k_{eff} = %0.6g',N+1,M,R,Z,k);
title(t2)
set(gcf,'InvertHardcopy','on');
print('Project2_NE571_2_TestCode','-dpng','-r100');
```