CHAPTER

20

Two-Group *k*-Eigenvalue Problems

OUTLINE

You, precious birds: your nests, your houses are in the trees, in the bushes. Multiply there, scatter there, in the branches of trees, the branches of bushes

–Popul Vuh, *as translated by Dennis Tedlock*

CHAPTER POINTS

- Two-group eigenvalue problems form a larger system of equations to apply inverse power iteration.
- The structure of two-group problems for thermal nuclear reactor systems allows inverse power iteration to be applied by solving two systems the size of a one-group problem at each iteration.

20.1 TWO-GROUP CRITICALITY PROBLEMS

To this point we have only solved single-group problems. Next we will extend our capabilities to two-group criticality problems. In this case we have two coupled diffusion equations of the form

Computational Nuclear Engineering and Radiological Science Using Python
DOI: 10.1016/B978-0-12-812253-2.00022-4

$$-\nabla \cdot D_1(r)\nabla\phi_1(r) + \Sigma_{t1}(r)\phi_1(r) = \Sigma_{s1\to 1}\phi_1(r) + \Sigma_{s2\to 1}\phi_2(r) + \frac{\chi_1}{k}\left[\nu\Sigma_{f1}(r)\phi_1(r) + \nu\Sigma_{f2}(r)\phi_2(r)\right], \quad (20.1a)$$

$$-\nabla \cdot D_2(r)\nabla\phi_2(r) + \Sigma_{t2}(r)\phi_2(r) = \Sigma_{s1\to 2}\phi_1(r) + \Sigma_{s2\to 2}\phi_2(r) + \frac{\chi_2}{k}\left[\nu\Sigma_{f1}(r)\phi_1(r) + \nu\Sigma_{f2}(r)\phi_2(r)\right], \quad (20.1b)$$

with a vacuum, reflecting, or albedo boundary condition at the outer surface

$$\mathcal{A}_g\phi_g(r) + \mathcal{B}_g\frac{d\phi_g}{dr} = 0 \qquad \text{for } r = R, \quad g = 1, 2,$$

and a reflecting boundary condition at $r = 0$

$$\frac{d}{dr}\phi_g(r) = 0 \qquad \text{for } r = 0, \quad g = 1, 2.$$

Note that the outer surface boundary conditions can be different for the different groups. In these equations ϕ_1 is the "fast" scalar flux and ϕ_2 is the thermal scalar flux. The fraction of fission neutrons born in group g is denoted by χ_g, Σ_{tg} is the total macroscopic cross-section in group g, $\nu\Sigma_{fg}$ is the product of the mean number of fission neutrons times the macroscopic fission cross-section for group g, and $\Sigma_{sg'\to g}$ is the macroscopic cross-section for scattering from group g' to group g.

Two-group problems are usually set up so that the following simplifications can be made

- There is no upscattering: $\Sigma_{s2\to 1} = 0$;
- All fission neutrons are born fast: $\chi_2 = 0$;
- We define the removal cross-section for group 1: $\Sigma_{r1} = \Sigma_{t1} - \Sigma_{s1\to 1} = \Sigma_{a1} + \Sigma_{s1\to 2}$;
- We define the removal cross-section for group 2: $\Sigma_{r2} = \Sigma_{t2} - \Sigma_{s2\to 2} = \Sigma_{a2}$.

Upon making these simplifications system (20.1) becomes

$$-\nabla \cdot D_1(r)\nabla\phi_1(r) + \Sigma_{r1}(r)\phi_1(r) = \frac{1}{k}\left[\nu\Sigma_{f1}(r)\phi_1(r) + \nu\Sigma_{f2}(r)\phi_2(r)\right],$$

$$-\nabla \cdot D_2(r)\nabla\phi_2(r) + \Sigma_{r2}(r)\phi_2(r) = \Sigma_{s1\to 2}\phi_1(r).$$

These equations can be discretized use the same procedure we used for one-group equations. The only difference is that now we will have coupling between each equation in the form of the fission terms and the downscattering.

If we apply our cell-centered discretization from the previous two lectures to the two-group equations, we get for the fast-group equations

$$-\frac{1}{V_i}\left[D_{1,i+1/2}S_{i+1/2}\frac{\phi_{1,i+1} - \phi_{1,i}}{\Delta r} - D_{1,i-1/2}S_{i-1/2}\frac{\phi_{1,i} - \phi_{1,i-1}}{\Delta r}\right] + \Sigma_{r1,i}\phi_{1,i} = \frac{1}{k}\left[\nu\Sigma_{f1,i}\phi_{1,i} + \nu\Sigma_{f2,i}\phi_{2,i}\right], \qquad i = 0, \ldots, I-1,$$

and

$$\left(\frac{\mathcal{A}_1}{2} - \frac{\mathcal{B}_1}{\Delta r}\right)\phi_{1,I-1} + \left(\frac{\mathcal{A}_1}{2} + \frac{\mathcal{B}_1}{\Delta r}\right)\phi_{1,I} = 0.$$

The thermal group equations are

$$-\frac{1}{V_i}\left[D_{2,i+1/2}S_{i+1/2}\frac{\phi_{2,i+1}-\phi_{2,i}}{\Delta r} - D_{2,i-1/2}S_{i-1/2}\frac{\phi_{2,i}-\phi_{2,i-1}}{\Delta r}\right] + \Sigma_{\mathrm{r}2,i}\phi_{2,i}$$
$$= \Sigma_{\mathrm{s}1\to 2,i}\phi_{1,i}, \qquad i = 0,\ldots,I-1,$$

and

$$\left(\frac{\mathcal{A}_2}{2} - \frac{\mathcal{B}_2}{\Delta r}\right)\phi_{2,I-1} + \left(\frac{\mathcal{A}_2}{2} + \frac{\mathcal{B}_2}{\Delta r}\right)\phi_{2,I} = 0.$$

These equations define our eigenvalue problem. Next, we discuss the particulars of this problem and how it can be solved.

20.2 GENERALIZED EIGENVALUE PROBLEM

We can write the $2(I+1)$ equations as a generalized eigenvalue problem of the form

$$\mathbf{A}\boldsymbol{\Phi} = \frac{1}{k}\mathbf{B}\boldsymbol{\Phi},$$

as before. The difference is that in this case we will write the system in a bit of a different form. First we define the solution vector $\boldsymbol{\Phi}$ as

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_{1,1} \\ \phi_{1,2} \\ \vdots \\ \phi_{1,I+1} \\ \phi_{2,1} \\ \phi_{2,2} \\ \vdots \\ \phi_{2,I+1} \end{pmatrix}.$$

The matrix $\mathbf{A}$ is what we call a block-matrix. That is a matrix that we define in terms of other, smaller matrices. In particular,

$$\mathbf{A} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{0} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix}.$$

The matrix $\mathbf{A}$ is a $2(I+1)\times 2(I+1)$ matrix with the $\mathbf{M}_{ij}$ each being $(I+1)\times(I+1)$ matrices. The $\mathbf{M}_{11}$ matrix is the left-hand side of the fast-group equation and $\mathbf{M}_{22}$ is the left-hand side

of the thermal flux equation. We can write out the non-zero entries of these matrices explicitly as

$$(\mathbf{M}_{gg})_{ii} = \begin{cases} \frac{2}{V_i \Delta r}\left[D_{g,i+1/2}S_{i+1/2} - D_{g,i-1/2}S_{i-1/2}\right] + \Sigma_{\mathrm{r}g,i} & i = 0 \ldots I-1 \\ \left(\frac{\mathcal{A}_g}{2} + \frac{\mathcal{B}_g}{\Delta r}\right) & i = I \end{cases},$$

$$(\mathbf{M}_{gg})_{i,i+1} = \left\{ -\frac{D_{g,i+1/2}S_{i+1/2}}{V_i \Delta r} \quad i = 0 \ldots I-1 \right.,$$

$$(\mathbf{M}_{gg})_{i,i-1} = \begin{cases} -\frac{D_{g,i-1/2}S_{i-1/2}}{V_i \Delta r} & i = 1 \ldots I-1 \\ \left(\frac{\mathcal{A}_g}{2} - \frac{\mathcal{B}_g}{\Delta r}\right) & i = I \end{cases}.$$

The matrix $\mathbf{M}_{21}$ is a diagonal matrix for the downscattering terms:

$$(\mathbf{M}_{21})_{ii} = \begin{cases} -\Sigma_{\mathrm{s}1\to 2,i} & i = 0 \ldots I-1 \\ 0 & i = I \end{cases}.$$

With these definitions the larger matrix $\mathbf{A}$ is defined. Now to define the right-hand side matrix $\mathbf{B}$. This matrix is written in block form as

$$\mathbf{B} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

The fission matrices, $\mathbf{P}_{1g}$ are $(I+1) \times (I+1)$ diagonal matrices of the form:

$$(\mathbf{P}_{1g})_{ii} = \nu \Sigma_{\mathrm{f}g,i}.$$

With these definitions we have now completely specified the generalized eigenvalue problem

$$\mathbf{A}\boldsymbol{\Phi} = \frac{1}{k}\mathbf{B}\boldsymbol{\Phi}.$$

20.3 INVERSE POWER METHOD FOR THE TWO GROUP PROBLEM

This eigenvalue problem has a particular structure that we can take advantage of. To use the inverse power iteration, recall that we have to solve systems of equations of the form

$$\mathbf{A}\mathbf{x}_{i+1} = \mathbf{B}\mathbf{x}_i.$$

In our case we can solve the matrix using block-forward substitution. What this means is that since our matrix $\mathbf{A}$ is of the form

$$\mathbf{A} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{0} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix},$$

we can solve the generic system

$$\begin{pmatrix} \mathbf{M}_{11} & \mathbf{0} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix},$$

as

$$\mathbf{y}_1 = \mathbf{M}_{11}^{-1}\mathbf{z}_1,$$

and

$$\mathbf{y}_2 = \mathbf{M}_{22}^{-1}\left(\mathbf{z}_2 - \mathbf{M}_{21}\mathbf{y}_1\right).$$

Therefore, instead of solving a large system involving $\mathbf{A}$, we solve

$$\mathbf{M}_{11}\mathbf{y}_1 = \mathbf{z}_1,$$

and

$$\mathbf{M}_{22}\mathbf{y}_2 = \left(\mathbf{z}_2 - \mathbf{M}_{21}\mathbf{y}_1\right).$$

That is, we solve two smaller systems. In particular we solve two, 1-group steady state diffusion equations in each iteration. Therefore, we can use our 1-group steady-state diffusion code from before, by modifying how it is called and what are the sources.

The benefit of solving a smaller system twice can be seen when we look at the scaling for the time to solution for LU factorization. As previously mentioned, LU factorization scales as the number of equations, n, to the third power: $O(n^3)$. This means that doubling the number of equations increases the time to solution by a factor of $2^3 = 8$. Solving the smaller system twice takes twice as long. Therefore, we save a factor of 4 in time to solution by solving two smaller systems. Also, the memory required is smaller because we do not form the matrix $\mathbf{A}$ and the $(I+1)^2$ zeros in the upper right block.

20.3.1 Inverse Power Iteration Function

Before looking at two-group diffusion problems, we will first show how to compute the eigenvalue of a block matrix system like the system above.

We now take our simple algorithm above and translate it into Python. This function will use the LU factorization functions discussed previously.

```
In [1]: def inversePowerBlock(M11, M21, M22, P11, P12,epsilon=1.0e-6,LOUD=False):
            """Solve the generalized eigenvalue problem
            (M11  0 ) (phi_1) = 1 (P11 P12) using inverse power iteration
            (M21 M22) (phi_2)    (0    0 )
            Inputs
            Mij: An LHS matrix (must be invertible)
            P1j: A fission matrix
            epsilon: tolerance on eigenvalue
            Outputs:
            l: the smallest eigenvalue of the problem
            x1: the associated eigenvector for the first block
            x2: the associated eigenvector for the second block
```

```
    """
    N,M = M11.shape
    assert(N==M)
    #generate initial guess
    x1 = np.random.random((N))
    x2 = np.random.random((N))
    l_old = np.linalg.norm(np.concatenate((x1,x2)))
    x1 = x1/l_old
    x2 = x2/l_old
    converged = 0
    #compute LU factorization of M11
    row_order11 = LU_factor(M11,LOUD=False)
    #compute LU factorization of M22
    row_order22 = LU_factor(M22,LOUD=False)
    iteration = 1;
    while not(converged):
        #solve for b1
        b1 = LU_solve(M11,np.dot(P11,x1) + np.dot(P12,x2),row_order11)
        #solve for b2
        b2 = LU_solve(M22,np.dot(-M21,b1),row_order22)
        #eigenvalue estimate is norm of combined vectors
        l = np.linalg.norm(np.concatenate((b1,b2)))
        x1 = b1/l
        x2 = b2/l
        converged = (np.fabs(l-l_old) < epsilon)
        l_old = l
        if (LOUD):
            print("Iteration:",iteration,"\tMagnitude of l =",1.0/l)
        iteration += 1
    return 1.0/l, x1, x2
```

To test this method, we can solve a very simple eigenproblem:

$$\begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0.1 \end{pmatrix} \mathbf{x} = l \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}.$$

The smallest eigenvalue is $\frac{1}{22} \approx 0.0454545\ldots$, and we will solve this using our inverse power iteration method.

```
In [2]: #define A
        M11 = np.identity(2)
        M11[0,0] = 10.0
        M11[1,1] = 0.5
        M22 = np.identity(2)
        M22[1,1] = 0.1
        M21 = -np.identity(2)
        #define P
        P11 = np.identity(2)
        P12 = np.identity(2)
        l, x1, x2  = inversePowerBlock(M11,M21,M22,P11,P12,epsilon=1.0e-8,LOUD=True)
```

```
Iteration: 1    Magnitude of l = 0.0887392840225
Iteration: 2    Magnitude of l = 0.0454591935894
Iteration: 3    Magnitude of l = 0.0454544458387
Iteration: 4    Magnitude of l = 0.0454545454546
Iteration: 5    Magnitude of l = 0.0454545454545
```

Now that our test passed, we will use this function to solve for the eigenvalue of a 1-D reactor.

20.4 SOLVING 1-D, TWO-GROUP DIFFUSION EIGENVALUE PROBLEMS

We will now modify our code from the previous lecture to deal with two-group eigenvalue problems. Now we will need to define more matrices and call our InversePowerBlock function.

```
In [3]: def TwoGroupEigenvalue(R,I,D1,D2,Sig_r1,Sig_r2,
                                    nu_Sigf1, nu_Sigf2,Sig_s12,
                                    BC1,BC2,
                                    geometry,epsilon = 1.0e-8):
            """Solve a neutron diffusion eigenvalue problem in a 1-D geometry
            using cell-averaged unknowns
            Args:
                R: size of domain
                I: number of cells
                Dg: name of function that returns diffusion coefficient
                    for a given r
                Sig_rg: name of function that returns Sigma_rg for a given r
                nuSig_fg: name of function that returns nu Sigma_fg for a given r
                Sig_s12: name of function that returns Sigma_s12 for a given r
                BC1: Boundary Value of fast phi at r=R in form [A,B]
                BC2: Boundary Value of thermal phi at r=R in form [A,B]
                geometry: shape of problem
                        0 for slab
                        1 for cylindrical
                        2 for spherical

            Returns:
                k: the multiplication factor of the system
                phi_fast:  the fast flux fundamental mode with norm 1
                phi_thermal:  the thermal flux fundamental mode with norm 1
                centers: position at cell centers

            """
            #create the grid
            Delta_r, centers, edges = create_grid(R,I)
            M11 = np.zeros((I+1,I+1))
            M21 = np.zeros((I+1,I+1))
            M22 = np.zeros((I+1,I+1))
            P11 = np.zeros((I+1,I+1))
```

```
P12 = np.zeros((I+1,I+1))
#define surface areas and volumes
assert( (geometry==0) or (geometry == 1) or (geometry == 2))
if (geometry == 0):
    #in slab it's 1 everywhere except at the left edge
    S = 0.0*edges+1
    S[0] = 0.0 #to enforce Refl BC
    #in slab its dr
    V = 0.0*centers + Delta_r
elif (geometry == 1):
    #in cylinder it is 2 pi r
    S = 2.0*np.pi*edges
    #in cylinder its pi (r^2 - r^2)
    V = np.pi*( edges[1:(I+1)]**2
                - edges[0:I]**2 )
elif (geometry == 2):
    #in sphere it is 4 pi r^2
    S = 4.0*np.pi*edges**2
    #in sphere its 4/3 pi (r^3 - r^3)
    V = 4.0/3.0*np.pi*( edges[1:(I+1)]**3
                - edges[0:I]**3 )

#Set up BC at R
M11[I,I] = (BC1[0]*0.5 + BC1[1]/Delta_r)
M11[I,I-1] = (BC1[0]*0.5 - BC1[1]/Delta_r)
M22[I,I] = (BC2[0]*0.5 + BC2[1]/Delta_r)
M22[I,I-1] = (BC2[0]*0.5 - BC2[1]/Delta_r)

#fill in rest of matrix
for i in range(I):
    r = centers[i]
    M11[i,i] = (0.5/(Delta_r * V[i])*((D1(r)+D1(r+Delta_r))*S[i+1]) +
               Sig_r1(r))
    M22[i,i] = (0.5/(Delta_r * V[i])*((D2(r)+D2(r+Delta_r))*S[i+1]) +
               Sig_r2(r))
    M21[i,i] = -Sig_s12(r)
    P11[i,i] = nu_Sigf1(r)
    P12[i,i] = nu_Sigf2(r)
    if (i>0):
        M11[i,i-1] = -0.5*(D1(r)+D1(r-Delta_r))/(Delta_r * V[i])*S[i]
        M11[i,i] += 0.5/(Delta_r * V[i])*((D1(r)+D1(r-Delta_r))*S[i])
        M22[i,i-1] = -0.5*(D2(r)+D2(r-Delta_r))/(Delta_r * V[i])*S[i]
        M22[i,i] += 0.5/(Delta_r * V[i])*((D2(r)+D2(r-Delta_r))*S[i])
    M11[i,i+1] = -0.5*(D1(r)+D1(r+Delta_r))/(Delta_r * V[i])*S[i+1]
    M22[i,i+1] = -0.5*(D2(r)+D2(r+Delta_r))/(Delta_r * V[i])*S[i+1]

#find eigenvalue
l,phi1,phi2 = inversePowerBlock(M11,M21,M22,P11,P12,epsilon)
k = 1.0/l
#remove last element of phi because it is outside the domain
phi1 = phi1[0:I]
phi2 = phi2[0:I]
return k, phi1, phi2, centers
```

To test this code we will solve a 1-group eigenvalue problem and pretend it has two groups. We will use the same case that we used for the fundamental mode for a homogeneous 1-group, 1-D reactor. We will set up the problem with $D_1 = D_2 = 3.850204978408833$ cm, $\nu\Sigma_{f,1} = \nu\Sigma_{f,2} = 0.1570$ cm^{-1}, and $\Sigma_{a,1} = \Sigma_{a,2} = 0.1532$ cm^{-1}. If we set $\Sigma_{s1\to2} = 0$, there will be no coupling from group 1 to group 2, therefore the scalar flux in group 2 will be zero everywhere and group 1 will be the same as that from the one group problem. Recall that for 1-group in spherical geometry the critical size can be found from

$$\frac{\nu\Sigma_f - \Sigma_a}{D} = \left(\frac{\pi}{R}\right)^2,$$

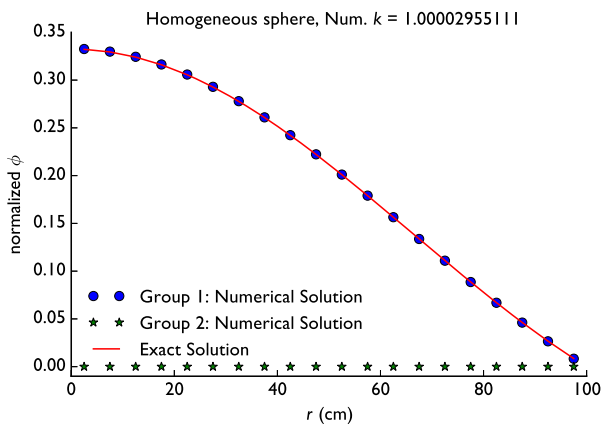
which leads to

$$R^2 = \frac{\pi^2 D}{\nu\Sigma_f - \Sigma_a}.$$

For this system the critical size is 100 cm.

We will run our eigenvalue solver on this problem. We should see that the eigenvalue converges to 1 as the number of mesh points is refined, also the eigenvector should be the fundamental mode.

```
In [4]: nuSigmaf_func = lambda r: 0.1570
        D_func = lambda r: 3.850204978408833
        Sigmaa_func = lambda r: 0.1532
        Sigmas_func = lambda r: 0.0
        R = 100
        I = 20
        #solution in spherical geometry with 100 cells
        k, phi_f,phi_t,centers = TwoGroupEigenvalue(R,I,D_func,D_func,
                                                    Sigmaa_func,Sigmaa_func,
                                                    nuSigmaf_func,nuSigmaf_func,
                                                    Sigmas_func,[1,0,0],[1,0,0],
                                                    2, epsilon=1.0e-10)
```



The flux shape looks correct, but the eigenvalue is slightly off as we saw last chapter; refining the mesh will improve the eigenvalue estimate. The thermal-flux for this problem is zero everywhere as expected.

Another way to check a two group problem is to solve an infinite medium problem. The formula for k_∞ is

$$k_\infty = \frac{\nu\Sigma_{\mathrm{f1}} + \frac{\Sigma_{\mathrm{s1}\to 2}}{\Sigma_{\mathrm{r2}}}\nu\Sigma_{\mathrm{f2}}}{\Sigma_{\mathrm{r1}}}.$$

Also, the ratio of the scalar fluxes will be

$$\frac{\phi_1}{\phi_2} = \frac{\Sigma_{\mathrm{r2}}}{\Sigma_{\mathrm{s1}\to 2}}.$$

For our test we will use the following values for the cross-sections (all values in cm^{-1})

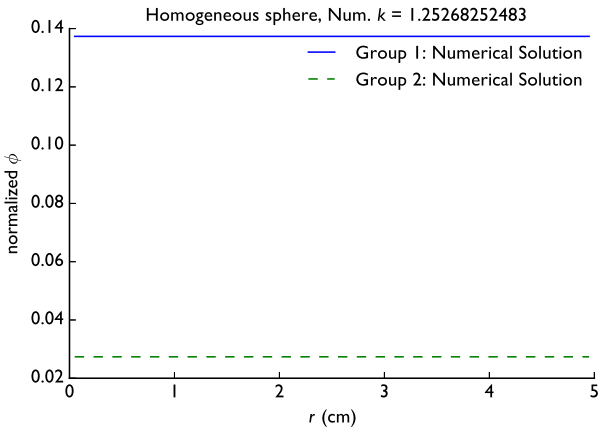
- $\nu\Sigma_{\mathrm{f1}} = 0.0085$
- $\Sigma_{\mathrm{s1}\to 2} = 0.0241$
- $\Sigma_{\mathrm{a1}} = 0.0121$
- $\Sigma_{\mathrm{r1}} = \Sigma_{\mathrm{s1}\to 2} + \Sigma_{\mathrm{a1}} = 0.0362$
- $\nu\Sigma_{\mathrm{f2}} = 0.185$
- $\Sigma_{\mathrm{r2}} = \Sigma_{\mathrm{a2}} = 0.121$

Using these values we get that $k_\infty = 1.25268$ and

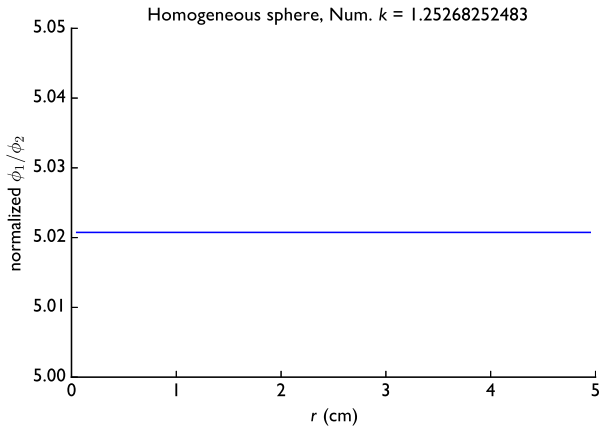
$$\frac{\phi_1}{\phi_2} = 5.021.$$

we will test our code using this solution. We will set $D_1 = D_2 = 0.1$ cm and we expect that with a reflecting boundary condition at the outer surface that $k_\infty \to 1.25268$ and the scalar flux ratio goes to 5.021.

```
In [5]: nuSigmaf1_func = lambda r: 0.0085
        nuSigmaf2_func = lambda r: 0.185
        D_func = lambda r: 0.1
        Sigmar1_func = lambda r: 0.0362
        Sigmar2_func = lambda r: 0.121
        Sigmas12_func = lambda r: 0.0241
        R = 5
        I = 50
        k, phi_f,phi_t,centers = TwoGroupEigenvalue(R,I,D_func,D_func,
                                                    Sigmar1_func,Sigmar2_func,
                                                    nuSigmaf1_func,nuSigmaf2_func,
                                                    Sigmas12_func,[0,1,0],[0,1,0],
                                                    2, epsilon=1.0e-8)
```



The eigenvalue is the value we expect, to within the iterative tolerance. We can also check the ratios of the scalar fluxes. The ratio should be 5.021.



This is indeed what we see in the solution.

20.5 TWO-GROUP REFLECTED REACTOR

Bare homogeneous reactor calculations are not where the usefulness of a numerical method is really needed as these problems can be solved by hand. A more complicated problem that we can use our new tool for is to analyze the increase in k_{eff} by surround the reactor with a reflector. That is, a material that can scatter neutrons back into the reactor. We consider a spherical reactor that has a reflector around it. As an example we set, for the reactor

- $D_1 = D_2 = 1$ cm
- $\nu\Sigma_{\text{f1}} = 0.00085\ \text{cm}^{-1}$

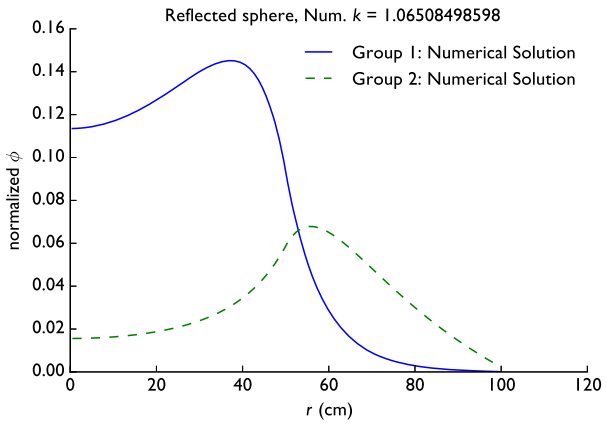
- $\Sigma_{s1\to 2} = 0.001\ \text{cm}^{-1}$
- $\Sigma_{a1} = 0.009\ \text{cm}^{-1}$
- $\Sigma_{r1} = \Sigma_{s1\to 2} + \Sigma_{a1} = 0.01\ \text{cm}^{-1}$
- $\nu\Sigma_{f2} = 0.057\ \text{cm}^{-1}$
- $\Sigma_{r2} = \Sigma_{a2} = 0.05\ \text{cm}^{-1}$,

and for the reflector

- $D_1 = D_2 = 1$ cm
- $\nu\Sigma_{f1} = 0.0\ \text{cm}^{-1}$
- $\Sigma_{s1\to 2} = 0.009\ \text{cm}^{-1}$
- $\Sigma_{a1} = 0.001\ \text{cm}^{-1}$
- $\Sigma_{r1} = \Sigma_{s1\to 2} + \Sigma_{a1} = 0.01\ \text{cm}^{-1}$
- $\nu\Sigma_{f2} = 0.0\ \text{cm}^{-1}$
- $\Sigma_{r2} = \Sigma_{a2} = 0.00049\ \text{cm}^{-1}$.

The code to set up and solve this problem follows.

```
In [6]: R_reac = 50.0
        nuSigmaf1_func = lambda r: 0.00085*(r<=R_reac)  + 0.0
        nuSigmaf2_func = lambda r: 0.057*(r<=R_reac)  + 0.0
        D_func = lambda r: 1.0
        Sigmar1_func = lambda r: 0.01
        Sigmar2_func = lambda r: 0.01*(r<=R_reac)  + 0.00049*(r>R_reac)
        Sigmas12_func = lambda r: 0.001*(r<=R_reac)  + 0.009*(r>R_reac)
        R = 100
        I = 100
        k, phi_f,phi_t,centers = TwoGroupEigenvalue(R,I,D_func,D_func,
                                                    Sigmar1_func,Sigmar2_func,
                                                    nuSigmaf1_func,nuSigmaf2_func,
                                                    Sigmas12_func,
                                                    [0.25,0.5*D_func(R)],
                                                    [0.25,0.5*D_func(R)],
                                                    2, epsilon=1.0e-6)
```

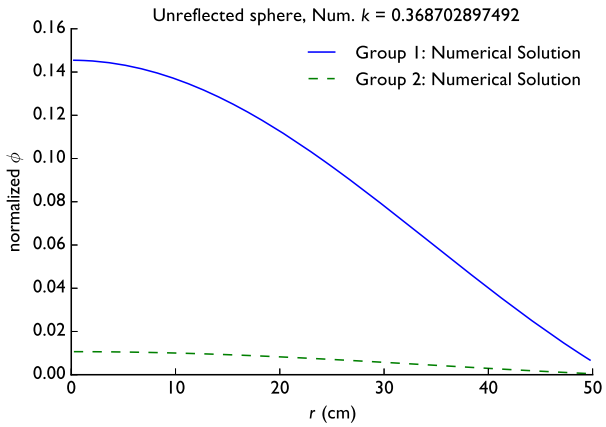


The scalar fluxes in this problem have several noticeable features:

- The thermal scalar flux has a peak in the reflector. This is from fast neutrons leaking out of the reactor and then slowing down in the reflector.
- The fast scalar flux has a peak toward the edge of the reactor. This peak is caused by thermal neutrons leaking back into the reactor from the reflector. These neutrons then cause fission in the reactor, producing fast fissions.

We can compare these results to an unreflected reactor of the same size.

```
In [7]:  R_reac = 500.0
         R = 50
         k, phi_f,phi_t,centers = TwoGroupEigenvalue(R,I,D_func,D_func,
                                                     Sigmar1_func,Sigmar2_func,
                                                     nuSigmaf1_func,nuSigmaf2_func,
                                                     Sigmas12_func,
                                                     [0.25,0.5*D_func(R)],
                                                     [0.25,0.5*D_func(R)],
                                                     2, epsilon=1.0e-6)
```



The eigenvalue decreased dramatically. This is due to the fact that the number of thermal neutrons in the system has been severely depressed, and the fission cross-section is higher for thermal neutrons. The implications of this phenomenon are explored in an exercise.

CODA

In this chapter we extended our reactor analysis capabilities to include two-group eigenvalue problems. We were able to apply this to a reflected reactor problem and observe some important reactor physics phenomenon. This marks the end of our foray into numerical solutions to the diffusion model of neutron transport. There is much more we could do: more groups, more dimensions, etc.

Rather than going deeper into diffusion models, we will pivot now to solving the neutron transport equation without making the diffusion approximation. We are going to investigate

a tool that is crucial in many nuclear engineering and radiological health applications: Monte Carlo calculations.

PROBLEMS

Programming Projects

1. Effective Albedo for Reflected Two-Group Reactor

The reflected two-group example discussed in the text above had the thermal flux peak inside the reflector. Replace this reflector with an albedo boundary condition at $r = 50$ cm. You can define a different value of α for the thermal and fast boundary condition. Using numerical experimentation, determine values for these alphas that result in a match for the eigenvalue of the reflected reactor, has the thermal flux have a maximum at $r = 50$ cm, and has the fast flux peak near the edge of the reactor. **Hint**: Think about what it means physically to have a reflector if a majority of the neutrons that leak out of the fuel are fast, and a majority of the neutrons that return are thermal.

Discuss your findings and compare the arrived at eigenvalue with the infinite medium eigenvalue of the reactor material. Does this explain why a vein of natural uranium in a mine (something very large) is subcritical, but natural uranium surrounded by heavy water (as in CANDU reactor) or graphite (as in the Chicago Pile) can be made critical?

2. 2-Group Heterogeneous Reactor Multiplication Factor

Consider the following 1-D cylindrical core consisting of 10 fuel regions + 1 reflector region (each region is of width 20 cm, total domain size is $R = 200$ cm).

$$\mathrm{U_{rodded}} \quad \mathrm{M} \quad \mathrm{U} \quad \mathrm{M} \quad \mathrm{U} \quad \mathrm{M} \quad \mathrm{U} \quad \mathrm{M} \quad \mathrm{U} \quad \mathrm{R}$$

In this table, R= reflector, U = UO_2, M = MOX, U_{rodded} = UO_2 + absorber. In this reactor we assume that our groups are set up so that only down scattering can be considered and all fission neutrons are born fast.

Write a Python code to solve this problem. Plot the solution for $\phi(r)$ for each group and comment on the behavior. Also give the value of the multiplication factor, k_{eff}. For this reactor give the ratio of the peak fission rate density to the average fission rate density over the fuel regions (i.e., not including the reflector). This ratio is called the power peaking factor.

Then, consider the same reactor where the control rods are removed and the "U_{rodded}" region becomes a "U" region. Discuss the change in the eigenvalue and the shape of the flux. The data you will need is below:

Material	D_1	D_2	Σ_{r1}	Σ_{r2}	$\nu\Sigma_{f1}$	$\nu\Sigma_{f2}$	$\Sigma_{s1\to2}$
U	1.2	0.4	0.029653979	0.093079585	0.004567474	0.114186862	0.020432526
M	1.2	0.4	0.029653979	0.23633218	0.006851211	0.351903125	0.015865052
R	1.2	0.2	0.051	0.04	0	0	0.05
U_{Rodded}	1.2	0.4	0.029820069	0.098477509	0.004567474	0.114186862	0.02032872