NPRE 555 Computational Project-2
Spring 2018

Numerical Solution of the Neutron Transport Equation Using the P_1 & P_3 Approximation for 1-D Infinite Slab

Due Date: 04/16/2018

Name: Muhammad Ahmed Abdelghany

- **Instructor:** Prof. Katy Huff

Table of Contents

I. Abstract

In this report, a discussion is introduced of how to numerically find the neutron scalar flux distribution inside an infinite bare slab with a finite thickness using the P_1 and P_3 approximations. The report starts with deriving the necessary equations used to develop a MATLAB code to solve this problem making use of Mark's boundary conditions one time and another time making use of Marshak's boundary conditions. The algorithm used, and the obtained results are discussed are also discussed in details and a comparison between the two approximation methods, and the use of each boundary condition type, is then introduced at the end.

The problem that we are solving is an infinite bare slab of moderator which has thickness $L = 5\ cm$, and contains a uniformly distributed source emitting $Q = 100\ \frac{n}{cm^3 s}$ which have ($\Sigma t = 1.5\ cm$, $\Sigma_S = 0.9\ cm$).

II. Deriving the Equations Needed for Developing the Code

P_1 and P_3 are the first and third order approximations of the P_N method. The essence of the P_N approximation method is to expand the angular flux and cross-sections in terms of the spherical harmonics. So, lets start with deriving the equations needed to develop a numerical code for the two approximations and show how we can describe the boundary conditions of both Mark's boundary conditions and Marshak's boundary conditions for the vacuum boundaries of the slab in the context of each approximation.

1. The P_1 Approximation

$$for\ (n = 0) \quad \therefore \frac{d\varphi_1(x)}{dx} + \Sigma_o(x)\ \varphi_o(x) = Q_o(x)$$

$$for\ (n = 1) \quad \therefore \frac{d\varphi_o(x)}{dx} + 3\Sigma_1(x)\ \varphi_1(x) = 3Q_1(x)$$

Since

$$\varphi_n(x) = \int_{\mu=-1}^{1} \psi(x,\mu)\, P_n(x)\ dx$$

$$\therefore\ \varphi_o(x) = \int_{\mu=-1}^{1}(1)\ \psi(x,\mu)\ d\mu = \varphi(x) \qquad \textit{(Scalar flux)}$$

$$\therefore\ \varphi_1(x) = \int_{\mu=-1}^{1} \mu\ \psi(x,\mu)\ d\mu = |\vec{J}(x)| \qquad \textit{(Current)}$$

$$\therefore \frac{dJ}{dx} + \Sigma_o(x)\ \varphi(x) = Q_o(x) \qquad (1)$$

$$\therefore \frac{d\varphi(x)}{dx} + 3\Sigma_1(x)\ \ J(x) = 3Q_1(x) \qquad (2)$$

let:

Δ: *is uniform in all the spatial mesh.*

$\Sigma's$ *are equal for all the meshes* $(uniform\ medium)$

- ***Source Terms***

$$Q_n(x) = 2\pi \int_{\mu=-1}^{1} Q(x,\mu)\, P_n(\mu)\ d\mu$$

$$Q_o(x) = 2\pi\ \frac{S}{4\pi} \times 2 = S \quad (uniform\ source\ strength)$$

$$Q_1(x) = 2\pi\ \frac{S}{4\pi} = S \int_{\mu=-1}^{1} \mu\ \ d\,\mu = 0$$

- ***Cross − sections terms***

$$\Sigma_n(x) = \Sigma_t(x) - \Sigma_{s_n}(x)$$

$$\Sigma_n(x) = \Sigma_t(x) - 2\pi \int_{\mu=-1}^{1} \Sigma_s(x,\mu_o)\, P_n(\mu_o)\ \ d\mu_o$$

$$\Sigma_o(x) = \Sigma_t(x) - 2\pi\ \frac{\Sigma_s}{4\pi}(2) = \Sigma_t(x) - \Sigma_s(x)$$

$$\therefore \Sigma_1(x) = \Sigma_t(x) - \pi\Sigma_s(0) = \Sigma_t$$

Integrate over (x) ***from*** $\left(x_{k-\frac{1}{2}} \longrightarrow x_{k+\frac{1}{2}}\right)$

Integrate eqnuation (2) ***in a way that produces expression to*** $J_{k+\frac{1}{2}}$ & $J_{k-\frac{1}{2}}$

$So,\ 1^{st}:\ x_k \longrightarrow x_{k+1}$, $2^{nd}:\ x_{k-1} \longrightarrow x_k$

$$\therefore -\frac{1}{3\Delta\Sigma_t}\varphi_{k+1} + \left(\Sigma_o\Delta + \frac{2}{3\Delta\Sigma_t}\right)\varphi_k - \frac{1}{3\Delta\Sigma_t}\varphi_{k-1} = Q_o\Delta = S\Delta$$

$$\therefore a_k\varphi_{k-1} + b_k\varphi_k + c_k\varphi_{k+1} = S_k \qquad (Repeat\ this\ eqn\ for\ (k = 1,..,k-1)\ \ in\ \varphi_o, \varphi_1, .., \varphi_k$$

where:

$$a_k = \frac{1}{3\Delta\Sigma_t} \qquad , b_k = (\Sigma_t - \Sigma_s)\Delta + \frac{2}{3\Delta\Sigma_t} \qquad , c_k = a_k \qquad , S_k = (S) * \Delta$$

then the above set of equations can be written in the matrix form as the following

$$\bar{\varphi} = \begin{pmatrix} \varphi_o \\ \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_n \end{pmatrix} \quad , \quad \bar{S} = \begin{pmatrix} S_o \\ S_1 \\ S_2 \\ \vdots \\ S_k \end{pmatrix}$$

$$\therefore \bar{\bar{A}}\bar{\varphi} = \bar{S} \quad , \quad \therefore \bar{\varphi} = \bar{\bar{A}}^{-1}\bar{S} \tag{3}$$

1.1. Mark' s BCs *(flux is zero at the boundary)*

$$\therefore \varphi(x = 0) = \varphi(x = L) = 0$$

$$\therefore \varphi_o = 0 \quad , \varphi_k = 0 \tag{4}$$

1.2. Marshak' s BCs *(incoming current from the vacuum is zero)*

$$(\mathbf{1}) \quad \boldsymbol{J^+(0) = 0} \quad , \quad \therefore \int_{\mu=0}^{1} \psi(0,\mu)\, P_1(\mu)\ d\mu = 0$$

$$\therefore J^+(0) = \int_{\mu=0}^{1} \mu \left(\frac{1}{2}\varphi_o(0) + \frac{3}{2}\mu\, \varphi_1(0)\right) d\mu = \frac{1}{4}\, \varphi_o(0) + \frac{1}{2}\, \varphi_1(0) = 0$$

$$\therefore \ \varphi_o(0) = \ -2\ \varphi_1(0)$$

$$\therefore \ \varphi_o(0) = -2\, J(0) \quad , \quad \therefore \ \varphi_{k=0} = -2\, J_{k=0}$$

let:

$$J_k = -\frac{1}{3\Delta\Sigma_t}(\varphi_{k+1} - \ \varphi_k)$$

$$\therefore \left(\mathbf{1} + \frac{\mathbf{2}}{\mathbf{3\Delta\Sigma_t}}\right) \boldsymbol{\varphi_o} - \left(\frac{\mathbf{2}}{\mathbf{3\Delta\Sigma_t}}\right) \boldsymbol{\varphi_1} = \mathbf{0} \tag{5}$$

$$(\mathbf{2}) \quad \boldsymbol{J^-(x = L) = 0} \quad , \quad \therefore \int_{\mu=-1}^{0} \psi(L,\mu)\, P_1(\mu)\ d\mu = 0$$

$$\therefore \int_{\mu=-1}^{0} \mu \left(\tfrac{1}{2}\varphi_o(L) + \tfrac{3}{2}\mu\, \varphi_1(L)\, d\mu\right) = -\tfrac{1}{4}\varphi_o(L) + \tfrac{1}{2}\varphi_1(L) \tag{4}$$

$$\therefore -\left(\frac{\mathbf{2}}{\mathbf{3\Delta\Sigma_t}}\right) \boldsymbol{\varphi_{k-1}} + \left(\mathbf{1} + \frac{\mathbf{2}}{\mathbf{3\Delta\Sigma_t}}\right) \boldsymbol{\varphi_k} = \mathbf{0} \tag{6}$$

2. The P_3 Approximation

$assuming \quad F_1(x) = \varphi_2(x) \quad , \quad F_o(x) = \varphi_o(x) + 2\varphi_2(x) \qquad \therefore \ \varphi_o(x) = F_o(x) - 2F_1(x)$

$$\varphi_o(x) = \int_{\mu=-1}^{1} (1)\, \psi(x,\mu)\, d\mu = \varphi(x) \quad (Scalar\ flux)$$

$$\Sigma_o(x) = \Sigma_t - \Sigma_s \quad , \Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_t \quad , \quad Q_o(x) = Q\ (Source\ strength)$$

$$\therefore -\frac{d}{dx}\left(\frac{1}{3\Sigma_1}\frac{d}{dx}F_o(x)\right) + \Sigma_o\, F_o(x) = 2\Sigma_o\, F_1 + Q_o(x) \qquad (1)$$

$$\therefore -\frac{d}{dx}\left(\frac{9}{35\Sigma_3}\frac{d}{dx}F_1(x)\right) + \left(\frac{4}{5}\,\Sigma_o + \Sigma_2\right)F_1(x) = \frac{2}{5}\Sigma_o\, F_o(x) - \frac{2}{5}Q_o(x) \qquad (2)$$

$from\ eqn\ (1)$:

$$\therefore \boldsymbol{F_{o_{k+1}}{}^{(n+1)} = \frac{1}{2}\left[\left(F_o{}^n(k+2) + F_o{}^n(k)\right) - A(\Sigma_o F_o{}^n(k) - 2\Sigma_o F_1{}^n(k) - Q)\right]} \qquad (3)$$

$$A = 3\Sigma_t\Delta^2 \ , \qquad (n: index\ for\ the\ iteration)$$

$from\ eqn\ (2)$:

$$\therefore \boldsymbol{F_1{}^{(n+1)}(k+1) = \frac{1}{2}\left[\left(F_1{}^n(k+2) + F_1{}^n(k)\right) - B\left(4\Sigma_o + 5\Sigma_t\right)F_1{}^n(k) - 2\left(\Sigma_o F_o{}^{n+1}(k) + Q\right)\right]}$$

$$B = \frac{7\Sigma_t\Delta^2}{9}$$

$$\therefore \boldsymbol{\varphi_k{}^{(n+1)} = F_{o_k}{}^{(n+1)} - 2F_{1_k}{}^{(n+1)}} \qquad \boldsymbol{\left(flux\ at\ the\ point\ x_k\right)}$$

2.1. Mark' s BCs *(flux is zero at the boundary)*

$$\varphi(x=0) = \varphi(x=L) = 0$$

$$\therefore \varphi_{k=1} = 0 \quad , \quad \therefore F_{o_{(k=1)}} - 2F_{1_{(k=1)}} = 0 \quad , \quad \therefore F_{o_{(k=1)}}{}^{(n+1)} = 2F_{1_{(k=1)}}{}^{(n+1)} \qquad (5)$$

$$\therefore \varphi_{k=k}{}^m = 0 \quad , \quad \therefore F_{o_k} - 2F_{1_k} = 0 \quad , \quad \therefore F_{o_k}{}^{(n+1)} = 2F_{1_k}{}^{(n+1)} \qquad (6)$$

2.2. Marshak' s BCs *(incoming current from the vacuum is zero)*

$$F_o(x) = \varphi_o(x) + 2\varphi_2(x)$$

$$\varphi_2(x) = F_1(x)$$

$\varphi_o(x) = F_o(x) - 2F_1(x)$

$(\mathbf{1})\ \boldsymbol{J^+(x=0)}$ $\quad (like\ before)$

$$\therefore J^+(x=0) = \int_{\mu=0}^{1} \psi(0,\mu)\ P_1(\mu)\ d\mu = 0$$

$$\therefore \psi(x,\mu) = \frac{2n+1}{2}\sum_{n=0}^{3} \varphi_n(x)\quad P_n(\mu)$$

$$= \frac{1}{2}(1)\ \varphi_o(x) + \frac{3}{2}(\mu)\varphi_1(x) + \frac{5}{2}(\frac{1}{2}(3\mu^2-1)\ \varphi_2(x) + \frac{7}{2}(\frac{1}{2}(5\mu^3-3\mu)\ \varphi_3(x)$$

$$\therefore J^+(0) = \int_{\mu=0}^{1} \mu\,(\frac{1}{2}\ \varphi_o(0) + \frac{3}{2}\mu\ \varphi_1(0) + \frac{5}{4}(3\mu^2-1)\ \varphi_2(0) + \frac{7}{4}(5\mu^2-3\mu)\ \varphi_3(0)d\mu$$

$$\therefore \boldsymbol{J^+(0) = \frac{1}{4}\varphi_o(0) + \frac{1}{2}\varphi_1(0) + \frac{5}{16}\varphi_2(0) = 0} \qquad \textbf{(A)}$$

Similarly:

$$J^-(L) = \int_{\mu=-1}^{0} \psi(0,\mu)\ P_1(\mu)\ d\mu = 0$$

$$\therefore \boldsymbol{J^-(L) = -\frac{1}{4}\varphi_o(L) + \frac{1}{2}\varphi_1(L) - \frac{5}{16}\varphi_2(L) = 0} \qquad \textbf{(B)}$$

Subst. into $(\boldsymbol{A})$ & $(\boldsymbol{B})$ ***using***:

$\varphi_o(0) = \varphi_{o_{k=1}} = F_{o_{k=1}} - 2F_{1_{k=1}}$ $\quad , \varphi_2(0) = \varphi_{2_{k=1}} = F_{1_{k=1}}$

$\varphi_o(L) = \varphi_{o_k} = F_{o_k} - 2F_{1_k}$ $\quad , \varphi_2(L) = \varphi_{2_k} = F_{1_k}$

$$\varphi_1(0) = \varphi_{1_{k=1}} = \frac{1}{3\Delta\Sigma_t}(F_o(k=1) - F_o(k=2))$$

$$\varphi_1(L) = \varphi_{1_k} = \frac{1}{3\Delta\Sigma_t}(F_o(k-1) - F_o(k))$$

Substitute all of these **8** ***quantities in eqn*** $(\boldsymbol{A})$ & $(\boldsymbol{B})$,
we will have two equations for the BCs interms of $\boldsymbol{F_o}$ & $\boldsymbol{F_1}$

$From\ (A)\ we\ can\ get:\ F_o(k=1)$

$From\ (B)\ we\ can\ get:\ F_o(k)$

Rewriting eqn $(\boldsymbol{A})$ ***after mulitplying it by*** $'*\mathbf{4}'$

$$\therefore \varphi_o(0) + 2\varphi_1(0) + \frac{5}{4}\varphi_2(0) = 0$$

let

$$\boldsymbol{C = \frac{2}{3\Delta\Sigma_t}} \quad , \therefore \boldsymbol{{F_o}^{n+1}(k=1) = \frac{1}{(1+C)}\left(\frac{3}{4}{F_1}^{n+1}(k=1) + C * {F_o}^{n+1}(k=2)\right)} \qquad \textbf{BC. 1}$$

Rewriting eqn(B) after mulitplying it by $\boldsymbol{' * 4'}$

$$\therefore \varphi_o(L) - 2\varphi_1(L) + \frac{5}{4}\varphi_2(L) = 0$$

$$\therefore \boldsymbol{{F_o}^{n+1}(k) = \frac{1}{(1+C)}\left(\frac{3}{4}{F_1}^{n+1}(k) + C * {F_o}^{n+1}(k-1)\right)} \qquad \textbf{BC. 2}$$

III. The Algorithm and Results

In this part, we will discuss how the developed codes work, and then we will introduce the results of each code, followed by a comparison of the results obtained from all of them. In this project four codes are developed; two for the P_1 approximation and two for the P_3 approximation. For each approximation method there is a code for implementing Mark's boundary conditions, and the other one for implementing Marshak's boundary conditions. The codes are written using MATLAB and can be found in Appendices A-D.

The idea behind the first code of P_1, is to solve the flux matrix $\bar{\varphi}$ for all the mesh points assuming Mark's boundary conditions at the vacuum boundaries, where the scalar flux is assumed to vanish there, as discussed in section I.1.1. of this report. The code starts with specifying the width of the slab L and the values of the cross-sections, then the number of mesh points k which can be chosen according to the required accuracy. The code then builds the S and A matrices and in the first step it takes care of the Mark's BCs by specifying the 1^{st} and k^{th} rows of the A Matrix. Next, it finds the inverse of the A matrix and calculate the flux matrix (phi). At the end, a plot of flux (phi) vs the width of the slab (x) is generated. The code is included in Appendix A, and the neutron scalar flux distribution for $k = 101$ mesh points, is shown in Figure-1.

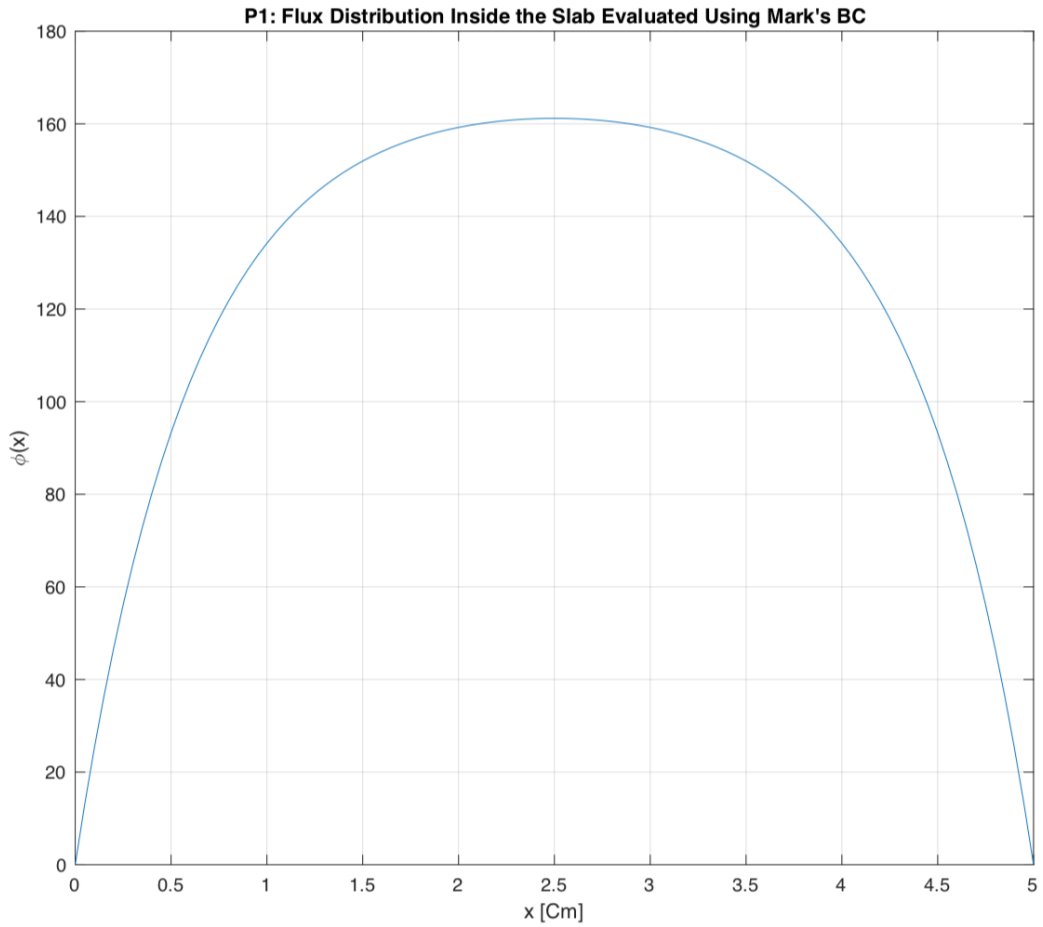

Figure 1: P1 Flux Distribution Using Mark's BC

The second P_1 code assumes the Marshak's boundary conditions, which assumes that the incoming neutron current is zero at the vacuum boundaries, hence the only difference between it and the previous code is that, while building the S and A matrices, the first step takes care of the Marshak's BCs by specifying the 1st and kth rows of the A Matrix, as discussed in section II.1.2. The code can be found in Appendix B and the neutron scalar flux distribution for $k = 101$ mesh points, is shown in Figure-2.

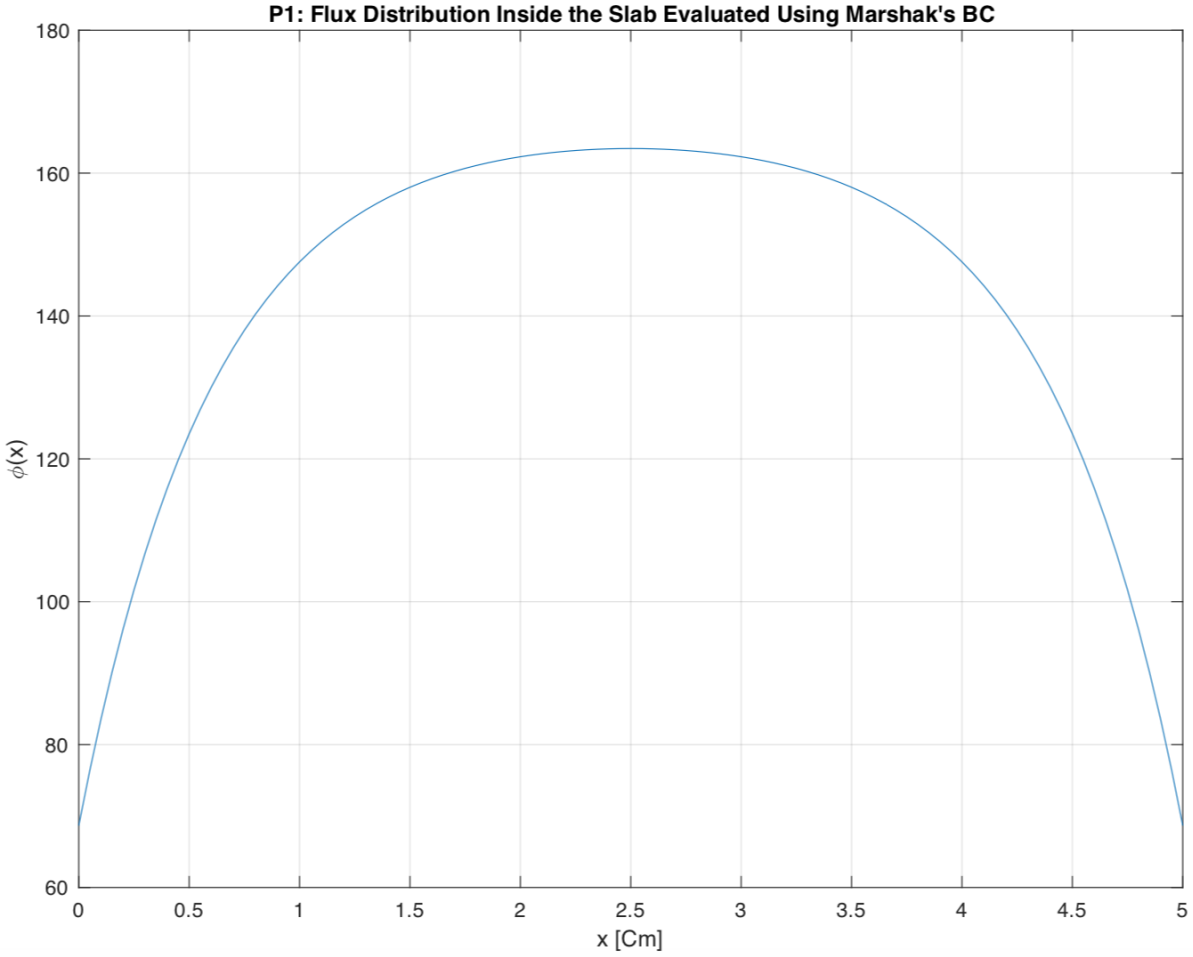

Figure 2: P1 Flux Distribution Using Marshak's BC

On the other hand, the first P_3 code is implementing the iterative method to find a convergent value of both F_1 and F_0 at each mesh point in the slab, which are related to the scalar flux, as discussed in section II.2 of this report. In this first P_3 code, Mark's boundary conditions is assumed for the vacuum boundaries, where the scalar flux is assumed to vanish. As before, the code starts with specifying the width of the slab L, the values cross-sections, and the number of mesh points k. The code is developed to implement an iterative scheme for F_1 and F_0, and to find the scalar flux after matching a certain conversion criterion (mx_dif). The code starts with specifying an arbitrary initial guess for F_1 and F_0 corresponding to the 1st iteration, then it calculates the values of F_1 and F_0 at each mesh point for the specific iteration. At each iteration, it assumes the initial values of F_1 of the two boundary points that are not possible to be calculated from the F_1 equations, to be equal to the corresponding values of the

nearest neighboring points. Next, for each iteration, the code implements Mark's BCs for the two boundary points for F_0 in terms of F_1. Then, it calculates the scalar flux (phi) for each mesh point and calculates the difference in (phi) between each two-successive iteration for each mesh point and compares the maximum of this difference with the pre-specified convergence criterion (mx_dif). After the convergence criterion is satisfied, the code displays the number of iterations n required for convergence, and then generate a plot of the flux (phi) vs the width of the slab (x). The code can be found in Appendix C and the neutron scalar flux distribution for $k = 101$ mesh points, is shown in Figure-3.

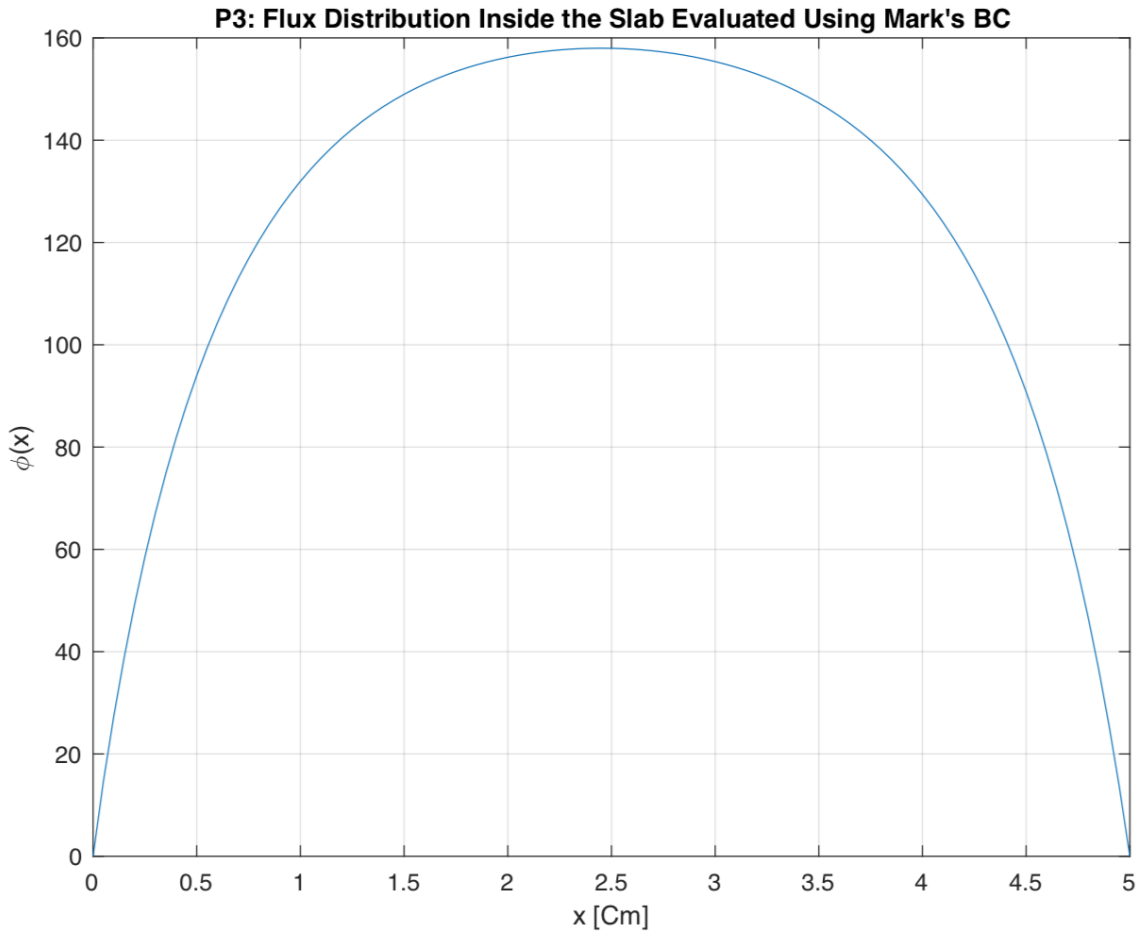

Figure 3: P3 Flux Distribution Using Mark's BC

The second P_3 code assumes the Marshak's boundary conditions, which assumes that the incoming neutron current is zero at the vacuum boundaries, hence the only difference between it and the previous P_3 code is that, it implements the equations required of applying the Marshak's boundary conditions during each iteration, for the two boundary points, for F_0 in terms of F_1, as discussed in section II.2.2. At the end, the code displays the number of iterations n required to satisfy the conversion criterion and then generate a plot of the flux (phi) vs the width of the slab (x). The code can be found in

Appendix D and the neutron scalar flux distribution for $k = 101$ mesh points, is found to be as shown in Figure-4.

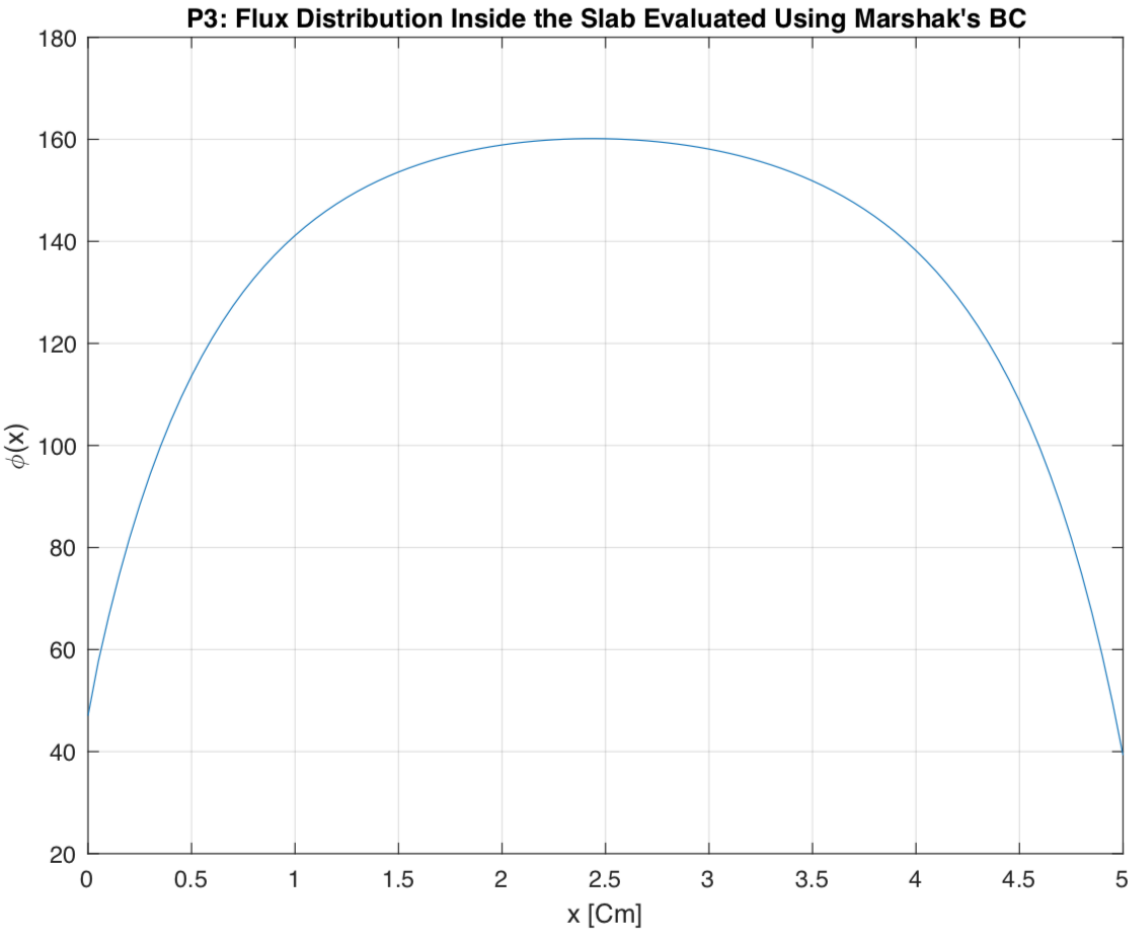

Figure 4: P3 Flux Distribution Using Marshak's BC

IV. Comparison & Conclusion

If we compared the results found by these four codes, we can get an idea of the difference between the P_1 approximation and P_3 approximation. Also, for each method we can see the difference expected by the physics of the problem for using Mark's boundary conditions and Marshak's boundary conditions. In Figure-5, the left figure shows the results for implementing the P_1 method using Mark's boundary conditions vs Marskak's boundary conditions. Where in the right figure, we can see the results for implementing the P_3 method using Mark's boundary conditions vs Marskak's boundary conditions. In addition, in Figure-6, we can examine the difference between using P_1 method vs using P_3 method while implementing Mark's boundary conditions. While in Figure-7, we can examine the difference between using P1 method vs using P3 method while implementing Marskak's boundary conditions.

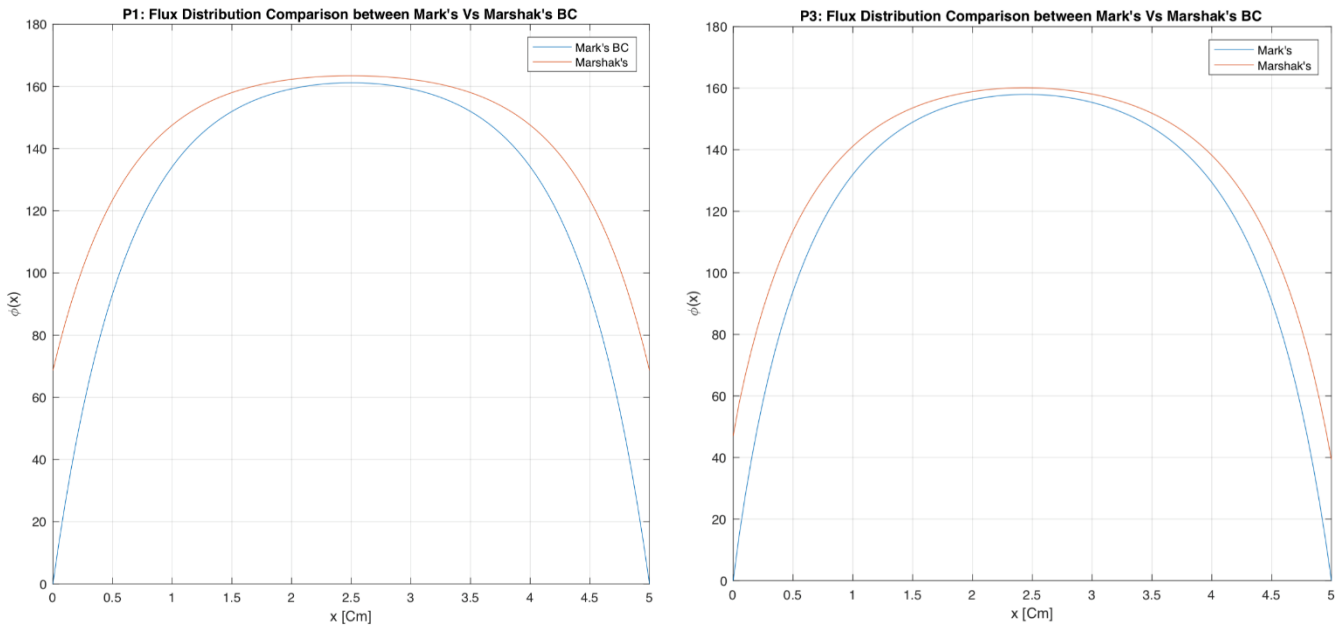

Figure 5 Mark's Vs Marshak's BC for both P1 and P3

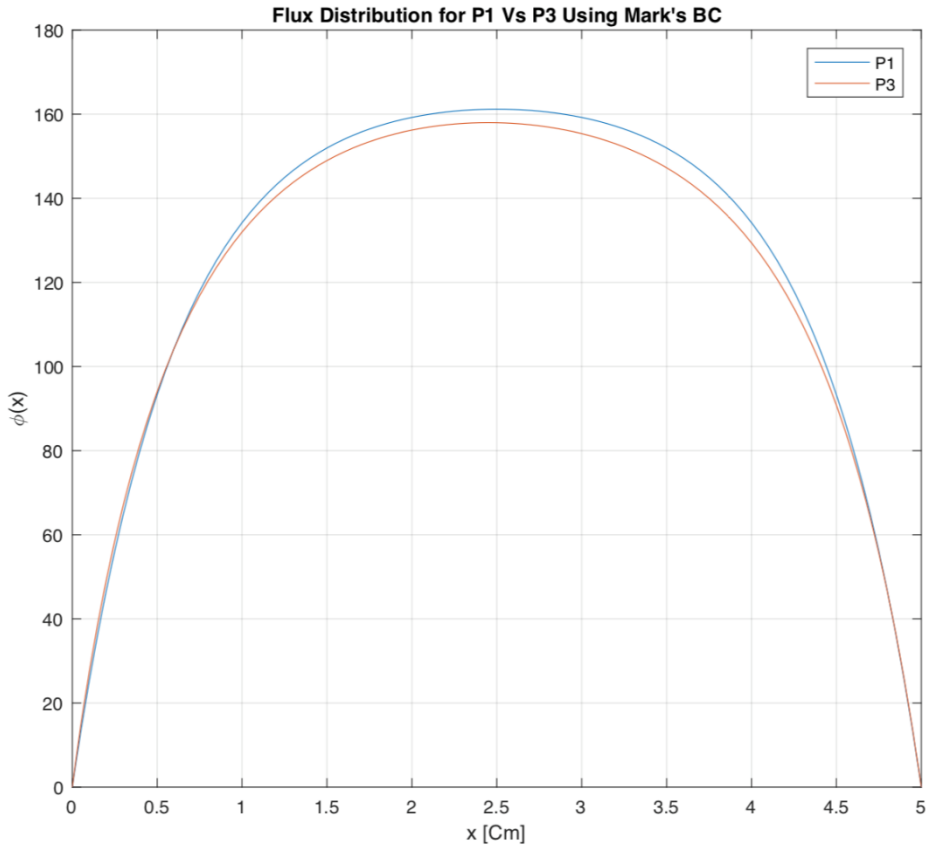

Figure 6: P1 Vs P3 Using Mark's BC

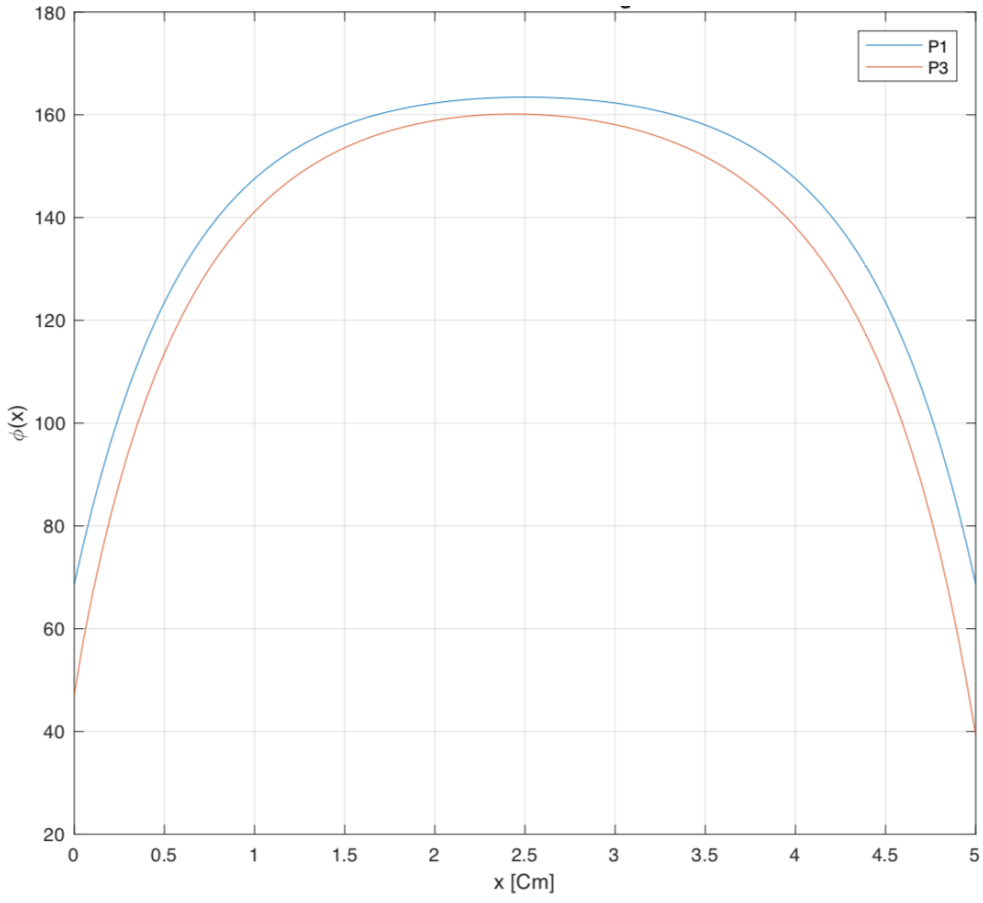

Figure 7: P1 Vs P3 Using Marshak's BC

In conclusion, there is a slight difference between using the P_1 approximation and the P_3 approximation. Marshak's boundary conditions looks more accurate to capture the real physics of the problem. In Figure-8, the four results found from the four codes are plotted on the same graph, which can work as a conclusion figure for the results of this work.

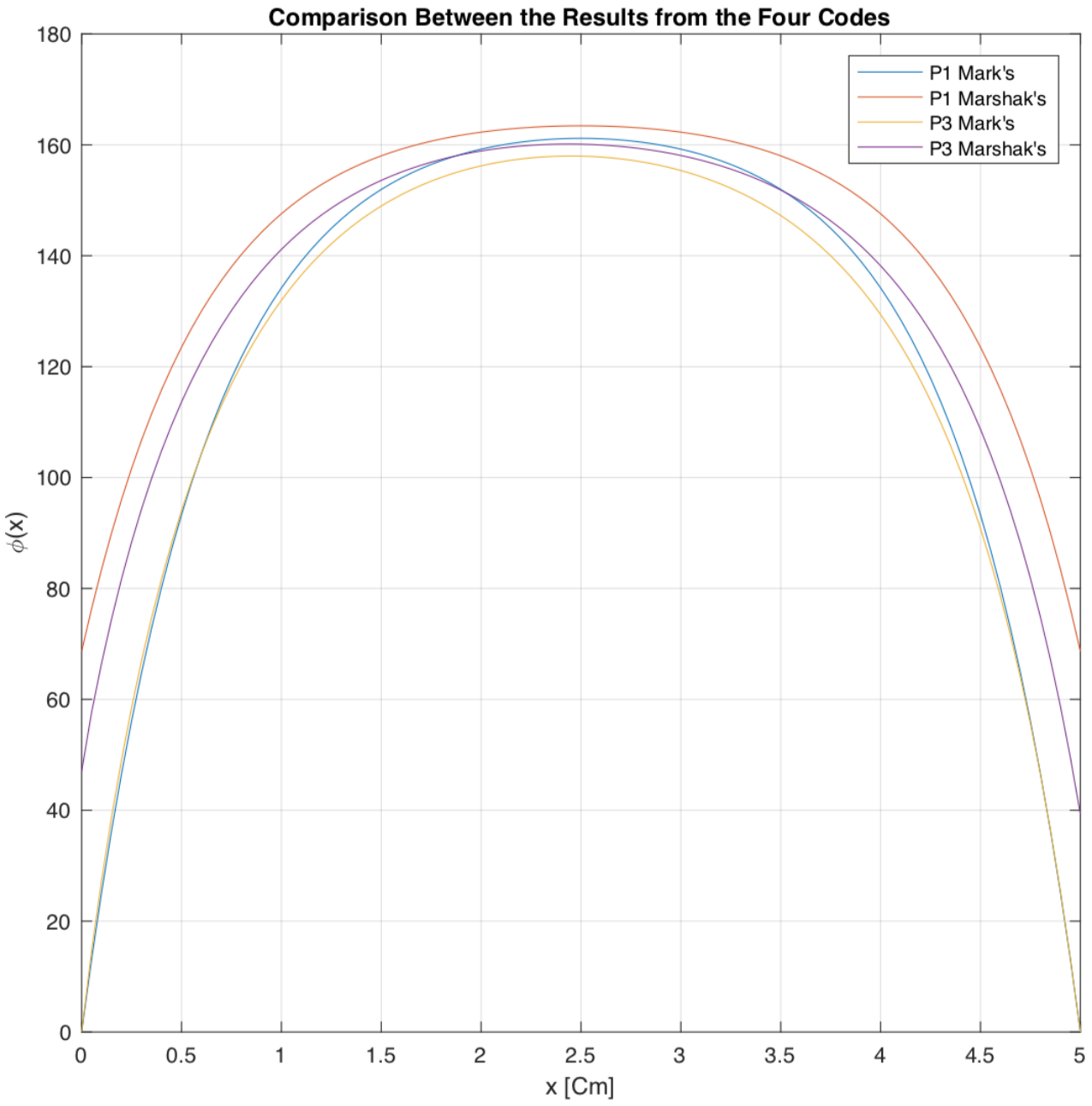

Figure 8: Comparison Between All the Four Fluxes

V. References

[1] K. Huff, “NPRE 555 Lecture Notes”, NPRE 555 Class, University of Illinois at Urbana-Champaign, 2018.

[2] M. Clark, K. Hansen, “Numerical Methods of Reactor Analysis”, Chapter 5, Academic Press, 1964.

[3] G. Bell, and S. Glasstone, “Nuclear Reactor Theory”, Chapter 3, Van Nostrand Reinhold Company, 1970.

Appendix A

P_1 Mark's BC

```
% P1 Mark's BC
clc;
clear;
close all;
seg_t=1.5;seg_s=0.9;Q=100;
L=5;                          % thickness of the whole slab
k=101;                        % number of mesh points
del=L/(k-1);

% Matrices Coefficents
a=-1/(3*del*seg_t);
b=(((seg_t-seg_s)*del)-(2*a));
c=a;
s=Q*del;

% i,j are indices which are indication on the number of mesh points
% Build the S Matrix
for i=1:k
    if i==1 | i==k
        S(i,1)=0;       % Because of the Mark's BCs
    else
    S(i,1)=s;
    end
end

% Specify the 1st and kth rows of the A Matrix from Mark's BCs
    A(1,1)=1;
    A(k,k)=1;
    for j=2:k
        A(1,j)=0;
    end
    for j=1:k-1
        A(k,j)=0;
    end

% Build the A Matrix
for i=2:k-1

    for j=1:k

        if j==i-1
            A(i,j)=a;
        elseif j==i
            A(i,j)=b;
        elseif j==i+1
            A(i,j)=c;
        else
            A(i,j)=0;
        end

     end
```

```
end

% Find the Inverse of (A) and computing phi
D=det(A);
P=inv(A);
phi=P*S

% Plot (phi) vs the width of the slab (x)
x=linspace(0,L,k);
plot(x,phi(:))
xlabel('x [Cm]')
ylabel('\phi(x)')
title('P1: Flux Distribution Inside the Slab Evaluated Using Mark''s BC')
grid on
```

Appendix B

P_1 Marshak's BC

```
% P1 Marshak's BC
clc;
clear;
close all;
seg_t=1.5;seg_s=0.9;Q=100;
L=5;                          % thickness of the whole slab
k=101;                        % number of mesh points
del=L/(k-1);

% Matrices Coefficents
a=-1/(3*del*seg_t);
b=(((seg_t-seg_s)*del)-(2*a));
c=a;
s=Q*del;

% i,j are indices which are indication on the number of mesh points
% Build the S Matrix
for i=1:k
    if i==1 | i==k
        S(i,1)=0;             % Because of the Marshak's BCs
    else
    S(i,1)=s;
    end
end

% Specify the 1st and kth rows of the A Matrix from Marshak's BCs
    A(1,1)=1-2*a;
    A(1,2)=2*a;
    A(k,k-1)=2*a;
    A(k,k)=1-2*a;
    for j=3:k
        A(1,j)=0;
    end
    for j=1:k-2
        A(k,j)=0;
    end

% Build the Coefficient Matrix A
for i=2:k-1

    for j=1:k

        if j==i-1
            A(i,j)=a;
        elseif j==i
            A(i,j)=b;
        elseif j==i+1
            A(i,j)=c;
        else
            A(i,j)=0;
        end
```

```
     end
end

% Find the Inverse of (A) and computing phi
D=det(A);
P=inv(A);
phi=P*S

% Plot (phi) vs the width of the slab (x)
x=linspace(0,L,k);
plot(x,phi(:))
    xlabel('x [Cm]')
    ylabel('\phi(x)')
    title('P1: Flux Distribution Inside the Slab Evaluated Using Marshak''s BC')
    grid on
```

Appendix C

P_3 Mark's BC

```
% P3 Mark's BC
clc;
clear;
close all;
seg_t=1.5;seg_s=0.9;Q=100;
seg_0=seg_t-seg_s;
L=5;                          % thickness of the whole slab
k=101;                        % number of mesh points
del=L/(k-1);

% F0 and F1 equations' coefficients
A=3*seg_t*(del)^2;
B=(7*seg_t*(del)^2)/9;

% (i) is the space mesh index, and (n) is the iterations counter
% Arbitrary intial guess of F0 and F1 of the 1st iteration
for i=1:k
    F0(i,1)=5;
    F1(i,1)=10;
end


n=0;
mx_dif=1;                     % intiating the convergance criterion
while mx_dif>0.00001          % convergance criterion for the flux

    n=n+1;                    % next iteration

% +++++ Finding (F0) and (F1) for each mesh point for iteration (n) ++++++
        for i=1:k-2

        F0(i+1,n+1)=0.5*((F0(i+2,n)+F0(i,n))-A*(seg_0*F0(i,n)-2*seg_0*F1(i,n)-Q));
        F1(i+1,n+1)=0.5*((F1(i+2,n)+F1(i,n))-B*(((4*seg_0+5*seg_t)*F1(i,n))-
(2*(seg_0*F0(i,n+1)+Q))));

        end

% Choose the two boundary values of F1 for each iteration to be equal the
neighborhood values this choice
% should be done by different approaches, but I found this most convenient
        F1(k,n+1)=F1(k-1,n+1);
        F1(1,n+1)=F1(2,n+1);

        % implementing Mark's BCs
        F0(1,n+1)=2*F1(1,n+1);
        F0(k,n+1)=2*F1(k,n+1);

        % find phi for each mesh point for this iterantion
        for i=1:k
            phi(i,n)=F0(i,n+1)-2*F1(i,n+1);
```

```
            % Specifying the difference in phi between each two successive
iteration

            if n>1   % skip evaluating the difference for the 1st iteration
            dphi(i,n-1)=phi(i,n)-phi(i,n-1);            % difference in phi
            else
            dphi=1;               % put the difference=1 for 1st iteration
            end

        end
        % find the maximum difference in flux between all the mesh points
        mx_dif=max(dphi);
end
n               % display the number of iterations required for convergance

% Plot (phi) vs the width of the slab (x)
x=linspace(0,L,k);
plot(x,phi(:,n))
xlabel('x [Cm]')
ylabel('\phi(x)')
title('P3: Flux Distribution Inside the Slab Evaluated Using Mark''s BC')
grid on
hold on
```

Appendix D

P_3 Marshak's BC

```
% P3 Marshak's BC
clc;
clear;
close all;
seg_t=1.5;seg_s=0.9;Q=100;
seg_0=seg_t-seg_s;
L=5;                          % thickness of the whole slab
k=101;                        % number of mesh points
del=L/(k-1);

% F0 and F1 equations' coefficients
A=3*seg_t*(del)^2;
B=(7*seg_t*(del)^2)/9;
C=2/(3*del*seg_t);

% (i) is the space mesh index, and (n) is the iterations counter
% Arbitrary intial guess of F0 and F1 of the 1st iteration
for i=1:k
    F0(i,1)=5;
    F1(i,1)=10;
end

n=0;
mx_dif=1;                      % intiating the convergance criterion
while mx_dif>0.00001           % convergance criterion for the flux

    n=n+1;                     % next iteration

% +++++ Finding (F0) and (F1) for each mesh point for iteration (n) ++++++
        for i=1:k-2

        F0(i+1,n+1)=0.5*((F0(i+2,n)+F0(i,n))-A*(seg_0*F0(i,n)-2*seg_0*F1(i,n)-Q));
        F1(i+1,n+1)=0.5*((F1(i+2,n)+F1(i,n))-B*(((4*seg_0+5*seg_t)*F1(i,n))-
(2*(seg_0*F0(i,n+1)+Q))));

        end

% Choose the two boundary values of F1 for each iteration to be equal the
neighborhood values this choice
% should be done by different approaches, but I found this most convenient
        F1(k,n+1)=F1(k-1,n+1);
        F1(1,n+1)=F1(2,n+1);

        % implementing Marshak's BCs
        F0(1,n+1)=(1/(1+C))*((3/4)*F1(1,n+1)+C*F0(2,n+1));
        F0(k,n+1)=(1/(1+C))*((3/4)*F1(k,n+1)+C*F0(k-1,n+1));

        % find phi for each mesh point for this iterantion
        for i=1:k
            phi(i,n)=F0(i,n+1)-2*F1(i,n+1);
```

```
            % Specifying the difference in phi between each two successive
iteration

            if n>1    % skip evaluating the difference for the 1st iteration
            dphi(i,n-1)=phi(i,n)-phi(i,n-1);           % difference in phi
            else
            dphi=1;               % put the difference=1 for 1st iteration
            end

        end
        % find the maximum difference in flux between all the mesh points
        mx_dif=max(dphi);
end
n               % display the number of iterations required for convergance

% Plot (phi) vs the width of the slab (x)
x=linspace(0,L,k);
plot(x,phi(:,n))
xlabel('x [Cm]')
ylabel('\phi(x)')
title('P3: Flux Distribution Inside the Slab Evaluated Using Marshak''s BC')
grid on
hold on
```