Homework 3

Name: Siyuan Song

Email: siyuan_song@brown.edu

Department: Engineering

Problem

Consider the one-dimensional Helmholtz equation in cylindrical coordinates:

$$\frac{1}{r}\frac{d}{dr}\left[r\frac{du}{dr}\right]-u=0 \ \ (1)$$

where $u(r=0)$ is bounded and $u(r=1)=1$.

1. Find the functional for the above problem and the corresponding variations at the end points.

2. Compare a numerical solution to this problem using linear finite elements. Plot the error in norms L_2 and H^1 versus the number of elements and compare with the known error estimates. (Notes: Find the exact solution first).

Solution

1. Functional

The one-dimensional Helmholtz equation is

$$\frac{1}{r}\frac{d}{dr}\left[r\frac{du}{dr}\right]-u=0 \Rightarrow \frac{d}{dr}\left[r\frac{du}{dr}\right]-ru=0 \ \ (2)$$

The boundary condition is

- Natural Boundary Condition: $u_r(r=0)=0$ (Notes: $u(r=0)<\infty$)
- Dirichlet Boundary Condition: $u(r=1)=1$ (Leading to $\delta u(r=1)=0$)

Take the variation on both sides

$$\delta I=\int_0^1\left[\frac{d}{dr}\left[r\frac{du}{dr}\right]-ru\right]\delta u dr=0 \ \ (3)$$

Due to the boundary conditions, the first part is simplified as

$$\int_0^1\frac{d}{dr}\left[r\cdot\frac{du}{dr}\right]\cdot\delta u dr=\left[\delta u\cdot r\cdot\frac{du}{dr}\right]\Bigg|_0^1-\int_0^1\left[r\cdot\frac{du}{dr}\right]\cdot\delta\frac{du}{dr}\cdot dr=-\delta\int_a^b\frac{1}{2}r\cdot\left(\frac{du}{dr}\right)^2\cdot dr \ \ (4)$$

The second part is simplified as

$$-\int_a^b ru\delta u dr = -\delta\int_a^b \frac{1}{2}ru^2 dr \quad (5)$$

Assemble the first part and the second part, the functional is

$$I = -\frac{1}{2}r\left[\left(\frac{du}{dr}\right)^2 + u^2\right] \quad (6)$$

At the end points, the boundary conditions are

$$u_r\left(r=0\right)=0, u\left(r=1\right)=1 \quad (7)$$

The variations are

$$\delta u_r\left(r=0\right)=0, \delta u\left(r=1\right)=0 \quad (8)$$

2. Exact solution

The exact solution is

$$u\left(r\right)=\frac{J_0\left(ri\right)}{J_0\left(i\right)}=\frac{I_0\left(r\right)}{I_0\left(1\right)} \quad (9)$$

where, $I_0\left(r\right)$ is the first kind modified Bessel function of the zero order.

3. The weak form

The Helmholtz equation is

$$\frac{d}{dr}\left[r\frac{du}{dr}\right]-ru=0 \, (10)$$

The weak form of the Helmholtz equation is

$$\int_0^1\left(\frac{d}{dr}\left[r\frac{du}{dr}\right]-ru\right)vdr=0 \quad (11)$$

It can be simplified as

$$\int_0^1\left(\frac{d}{dr}\left[r\frac{du}{dr}\right]-ru\right)vdr=0 \Rightarrow \left[\left(r\frac{du}{dr}\right)\cdot v\Big|_0^1 - \int_0^1\frac{dv}{dr}\left(r\frac{du}{dr}\right)dr\right]-\int_0^1\left(ru\right)vdr=0 \Rightarrow \int_0^1 r\left(uv+u'v'\right)dr \quad (12)$$

4. Finite element method

Uniform grid is utilized here with $r_0=0, r_1=\frac{1}{N}, r_2=\frac{2}{N}..., r_N=1$. The number of the elements is N. The number of the nodes is $N+1$. The space step is $h=\frac{1}{N}$.

Let $U = \sum_{i=0}^{N} U_i \phi_i$ and $v = \phi_i, i = 0,1,2,...N$ (Galerkin method). Then the week form of the equation can be expressed as

$$\int_0^1 r\left(U_i\phi_i\phi_j + U_i\phi'_i\,\phi'_j\right)dr = 0 \quad (13)$$

Here, summation notation is adopted here (i.e. $U_i\phi_i = \sum_{i=0}^{N} U_i\phi_i$). Eq. (13) can be written in matrix form,

$$U_i\int_0^1 r\left(\phi_i\phi_j + \phi'_i\,\phi'_j\right)dr = 0 \Rightarrow A_{ji}U_i = 0 \Rightarrow \mathbf{AU} = 0 \quad (14)$$

where $A_{ji} = \int_0^1 r\left(\phi_i\phi_j + \phi'_i\,\phi'_j\right)dr$, $U_i = U\left(r = r_i\right)$.

We need to determine the function ϕ_i , which satisfies

$$U\left(r_i\right) = U_i \quad (15)$$

Lagrange interpolation is adopted here in each element. In the range $r \in \left(r_{k-1}, r_k\right)$, the function U can be written as,

$$U = \frac{r - r_{k-1}}{h}U_k + \frac{r_k - r}{h}U_{k-1}, r \in \left(r_{k-1}, r_k\right) \quad (16)$$

Since U_k only appears in the range $r \in \left(r_{k-1}, r_k\right)$ and $r \in \left(r_k, r_{k+1}\right)$, the base function ϕ_i can be given by,

$$\phi_0 = \begin{cases} \dfrac{r_1 - r}{h}, & r \in \left(r_0, r_1\right) \\ 0, & r \in \text{others} \end{cases} \qquad i = 0 \quad (17)$$

$$\phi_i = \begin{cases} \dfrac{r - r_{i-1}}{h}, & r \in \left(r_{i-1}, r_i\right) \\ \dfrac{r_{i+1} - r}{h}, & r \in \left(r_i, r_{i+1}\right) \\ 0, & r \in \text{others} \end{cases} \qquad \mathrm{i} = 1,2,...,N-1 \quad (18)$$

$$\phi_N = \begin{cases} \dfrac{r - r_{N-1}}{h}, & r \in \left(r_{N-1}, r_N\right) \\ 0, & r \in \text{others} \end{cases} \qquad i = N \quad (19)$$

Then $A_{ji} = \int_0^1 r\left(\phi_i \phi_j + \phi'_i \phi'_j\right) dr$ can be calculated easily. Now we have the coefficient matrix A and the vector U.

$$\begin{bmatrix} A_{00} & A_{01} & \dots & A_{0(N-1)} & A_{0N} \\ A_{10} & A_{11} & \dots & A_{1(N-1)} & A_{1N} \\ \dots & \dots & \dots & \dots & \dots \\ A_{(N-1)0} & A_{(N-1)1} & \dots & A_{(N-1)(N-1)} & A_{(N-1)N} \\ A_{N0} & A_{N1} & \dots & A_{N(N-1)} & A_{NN} \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \dots \\ U_{N-1} \\ U_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix} \quad (20)$$

According to the boundary condition $U_N = 1$. The matrix equation is written as

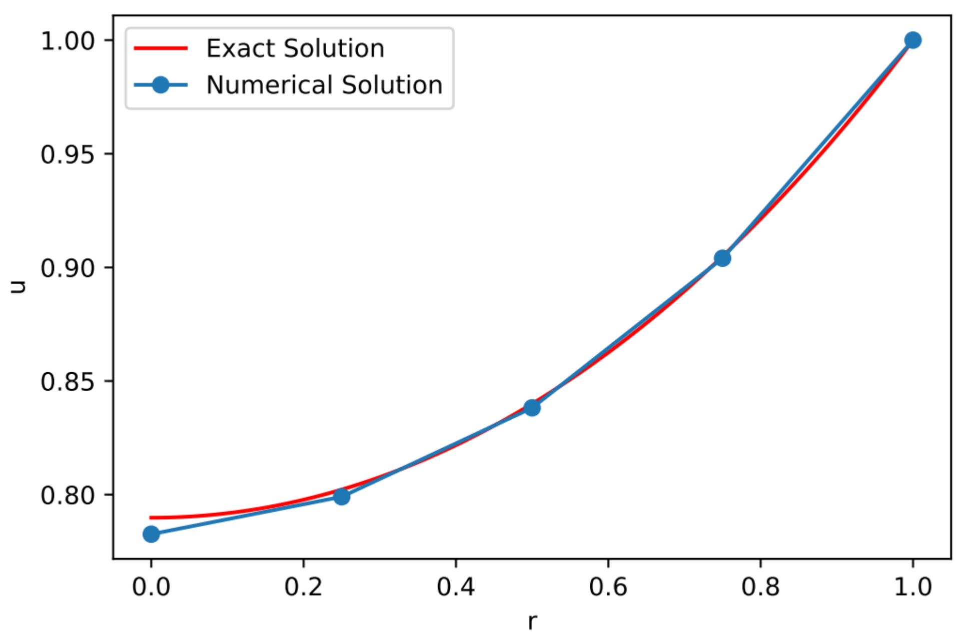
$$\begin{bmatrix} A_{00} & A_{01} & \dots & A_{0(N-1)} & A_{0N} \\ A_{10} & A_{11} & \dots & A_{1(N-1)} & A_{1N} \\ \dots & \dots & \dots & \dots & \dots \\ A_{(N-1)0} & A_{(N-1)1} & \dots & A_{(N-1)(N-1)} & A_{(N-1)N} \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \dots \\ U_{N-1} \\ U_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{bmatrix} \quad (21)$$

By solving this equation, we can obtain the numerical solution $\mathbf{U}$.

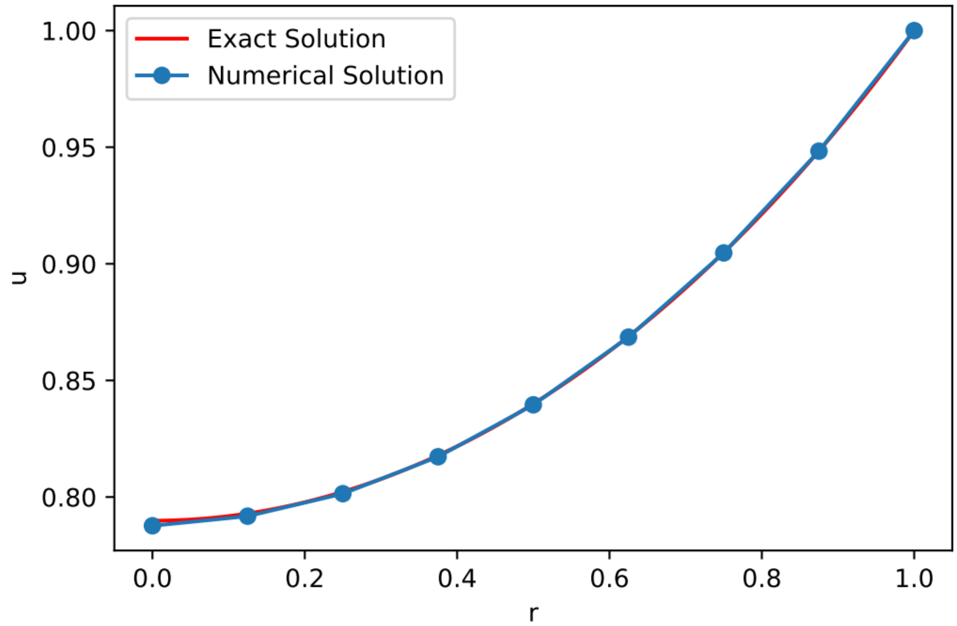
5. Solutions

Comparisons of theoretical predictions and numerical solutions

- N = 4:

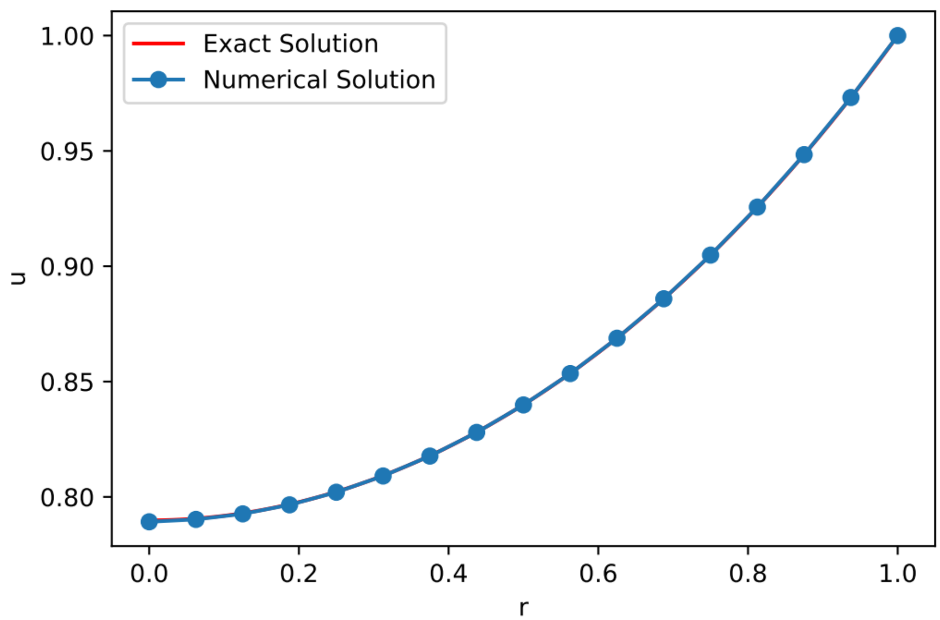


- N = 8:



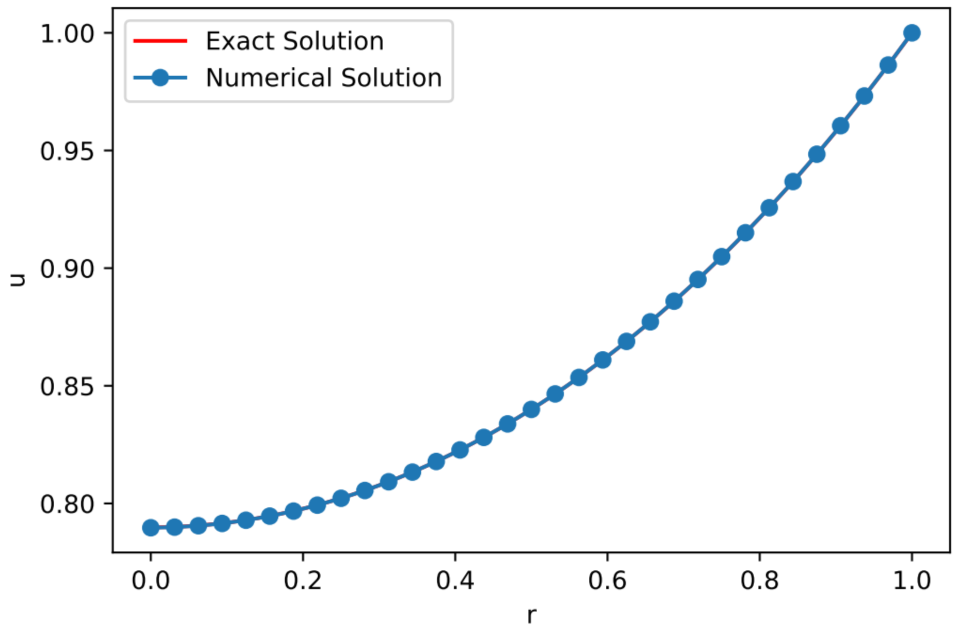
Exact Solution
Numerical Solution
1.00
0.95
0.90
0.85
0.80
u
0.0
0.2
0.4
0.6
0.8
1.0
r

- N = 16:

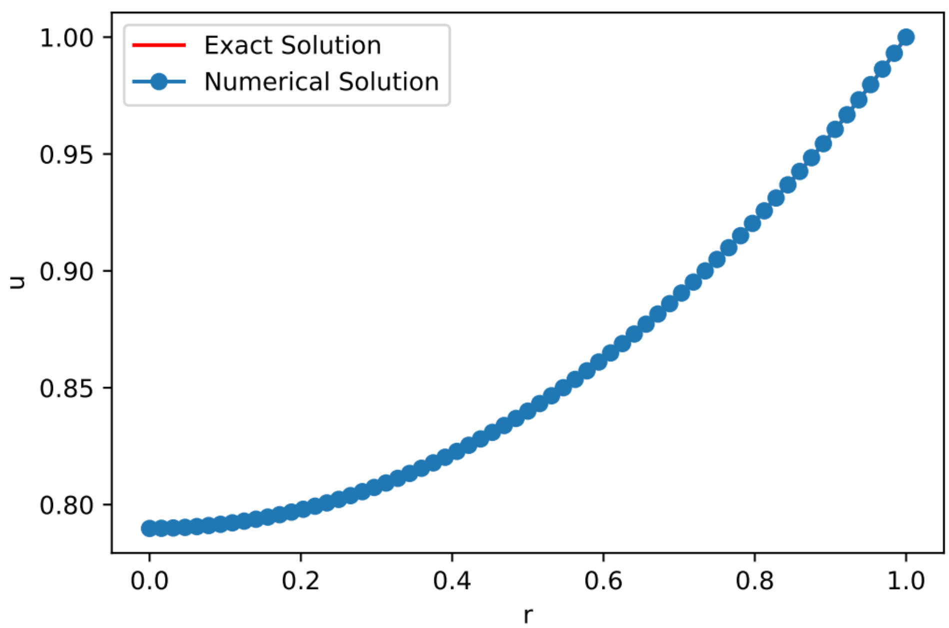


Exact Solution
Numerical Solution
1.00
0.95
0.90
0.85
0.80
u
0.0
0.2
0.4
0.6
0.8
1.0
r

- N = 32:



- N = 64:



6. Error Analysis (at nodes)

In this part, we consider the error at the nodes.

The L_2 error is defined as

$$e_{L2} = \sqrt{\sum_{i=0}^{N} \left(u_i - U_i\right)^2 h} \quad (22)$$

The H^1 error is defined as

$$e_{H1} = \sqrt{\sum_{i=0}^{N}\left(u_i - U_i\right)^2 h + \sum_{i=0}^{N}\left(\frac{du_i}{dr} - \frac{dU_i}{dr}\right)^2 h} \quad (23)$$

The exact derivative of u is,

$$\frac{du}{dr} = \frac{I_1(r)}{I_0(1)} \quad (24)$$

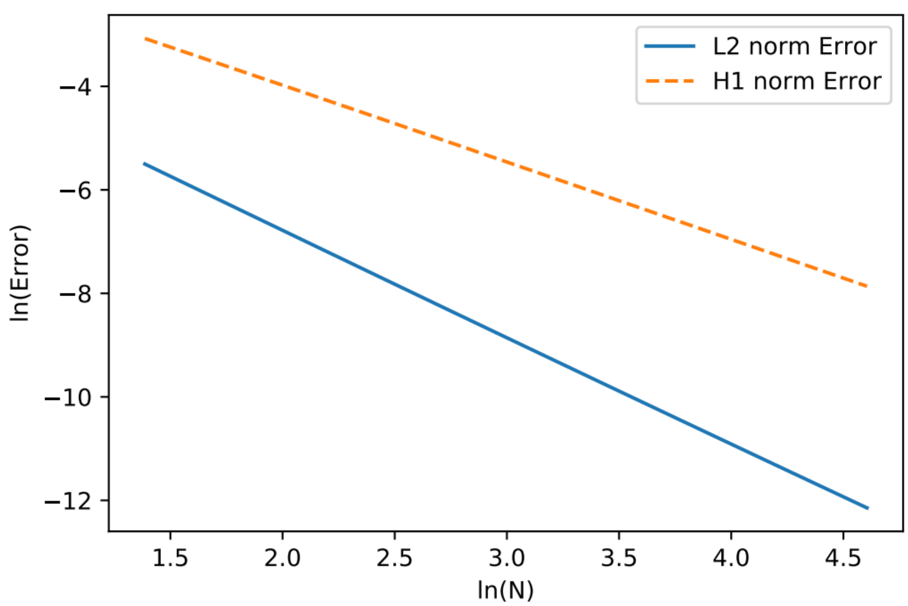
According to $U = \sum_{i=0}^{N} U_i \phi_i$, the numerical derivative of u is approximated by.

$$U = \sum_{i=0}^{N} U_i \phi'_i \quad (25)$$

Since ϕ_i is linear equation, its derivative is a constant in the range $r \in (r_{i-1}, r_i)$ and $r \in (r_i, r_{i+1})$. At the node point $r = r_i$, we take the average of the derivative on both sides, that is

$$\frac{dU_i}{dr} = \frac{U_{i+1} - U_{i-1}}{2h} \quad (26)$$

Specifically, at the points $r = r_0, r_N$, we only take the derivative on one side. The solution is given as below:



The order of the L2 norm is 2.0641 and the order of the H1 norm is 1.4874.

7. Error Analysis (in whole domain)

The L_2 error can be also obtained in the whole domain, defined as

$$e_{L2} = \sqrt{\int_0^1 (u-U)^2\, dr} \quad (27)$$

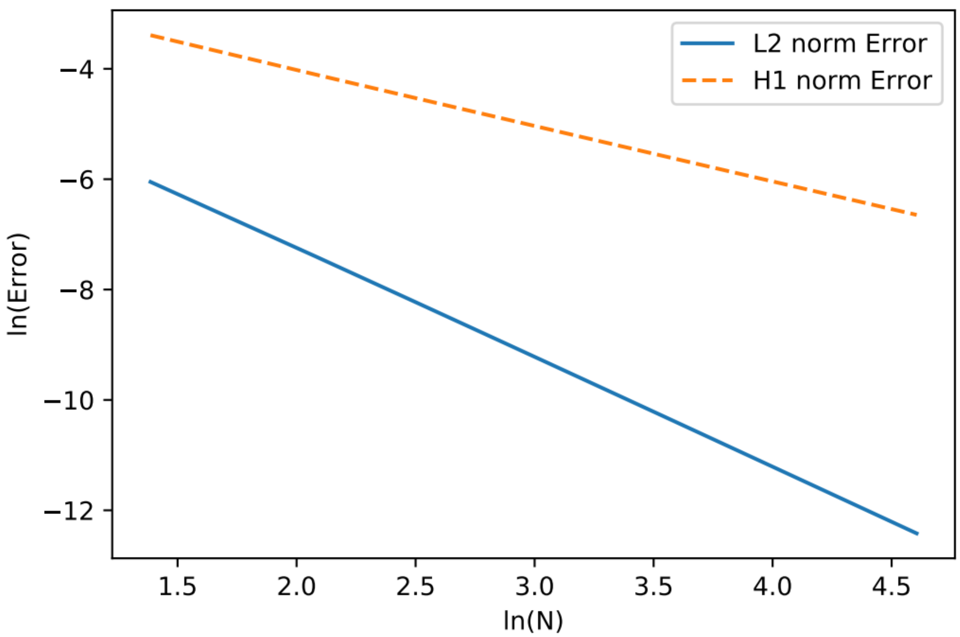
where, $U = \frac{r - r_{k-1}}{h} U_k + \frac{r_k - r}{h} U_{k-1}, r \in (r_{k-1}, r_k)$.

The H^1 error is defined as

$$e_{H1} = \sqrt{\int_0^1 \left[(u-U)^2 + \left(\frac{du}{dr} - \frac{dU}{dr} \right)^2 \right] dr} \quad (28)$$

where, $\frac{dU}{dr} = \frac{U_k - U_{k-1}}{h}, r \in (r_{k-1}, r_k)$.

The solution is given as below:



The order of the L2 norm is 1.97845278 and the order of the H1 norm is 1.01139933.

This conclusion is consistent with the theory given in the textbook.

Python Code

```
"""
Basic function: (Helmothoz Equation)
    1/r *d/dr(r*du/dr) = 0
Functional:
```

```
    1/2 * r * ((du/dr)^2 + u^2)
BC:
    Diriclet r = 1, u = 1, (set as right side colume)
    Natural r = 0, du/dr = 0, (don't need to do anything)
True solution:
    u(r) = bessel(0,r*i)/bessel(0,i)
"""
" ============================Finite Element Analysis ======================="
" ============================Import"
import numpy as np
from scipy import special
from matplotlib import pyplot
" ============================Basis parameters"
" The number of element"
def FEM(element_number):
    " The number of the node. Two nodes for each element"
    node_number = element_number + 1
    " For python convenience"
    N = node_number-1
    " The step size"
    h = 1/element_number
    " ============================Generate the grids"
    " Uniform grid is utilized here"
    grid = np.linspace(0,1,node_number)
    " ============================Finite element method"
    "%%%"
    " A is used to record the coefficient matrix"
    A = np.zeros([node_number,node_number])
    " Part B"
    A[0,0] = A[0,0] + (grid[1]**2-grid[0]**2)/(2*h**2)
    A[N,N] = A[N,N] + (grid[N]**2-grid[N-1]**2)/(2*h**2)
    for i in range(1,node_number-1):
        A[i,i] = A[i,i] + (grid[i]**2-grid[i-1]**2)/(2*h**2)
        A[i,i] = A[i,i] + (grid[i+1]**2-grid[i]**2)/(2*h**2)
    A[0,1] = A[0,1] - (grid[1]**2-grid[0]**2)/(2*h**2)
    A[N,N-1] = A[N,N-1] - (grid[N]**2-grid[N-1]**2)/(2*h**2)
    for i in range(1,node_number-1):
        A[i,i-1] = A[i,i-1] - (grid[i]**2-grid[i-1]**2)/(2*h**2)
        A[i,i+1] = A[i,i+1] - (grid[i+1]**2-grid[i]**2)/(2*h**2)
    " Part A"
    "fie_1 = a1*r + b1"
    "fie_2 = a2*r + b2"
    def intergral(a1,b1,a2,b2,r1,r2):
        return (a1*a2/4*(r2**4-r1**4)+(a1*b2+a2*b1)/3*(r2**3-r1**3)+b1*b2/2*(r2**2-r1**2))
    "i = 0"
    a1 = -1/h; b1 = h/h;
    a2 = -1/h; b2 = h/h;
    r1 = 0; r2 = h;
    A[0,0] = A[0,0] + intergral(a1,b1,a2,b2,r1,r2);
    a1 = -1/h; b1 = h/h;
    a2 = 1/h; b2 = -0/h;
    r1 = 0; r2 = h;
    A[0,1] = A[0,1] + intergral(a1,b1,a2,b2,r1,r2);
    "i = N"
    a1 = 1/h; b1 = -(N-1)*h/h;
    a2 = 1/h; b2 = -(N-1)*h/h;
```

```
r1 = (N-1)*h; r2 = N*h;
A[N,N] = A[N,N] + intergral(a1,b1,a2,b2,r1,r2);
a1 = 1/h; b1 = -(N-1)*h/h;
a2 = -1/h; b2 = (N)*h/h;
r1 = (N-1)*h; r2 = N*h;
A[N,N-1] = A[N,N-1] + intergral(a1,b1,a2,b2,r1,r2);
for i in range(1,node_number-1):
    "A[i,i]"
    a1 = 1/h; b1 = -((i-1)*h)/h;
    a2 = 1/h; b2 = -((i-1)*h)/h;
    r1 = (i-1)*h; r2 = i*h;
    A[i,i] = A[i,i] + intergral(a1,b1,a2,b2,r1,r2);
    a1 = -1/h; b1 = ((i+1)*h)/h;
    a2 = -1/h; b2 = ((i+1)*h)/h;
    r1 = i*h; r2 = (i+1)*h;
    A[i,i] = A[i,i] + intergral(a1,b1,a2,b2,r1,r2);
    "A[i,i-1]"
    a1 = 1/h; b1 = -(i-1)*h/h;
    a2 = -1/h; b2 = i*h/h;
    r1 = (i-1)*h; r2 = i*h;
    A[i,i-1] = A[i,i-1] + intergral(a1,b1,a2,b2,r1,r2);
    "A[i,i+1]"
    a1 = -1/h; b1 = (i+1)*h/h;
    a2 = 1/h; b2 = -i*h/h;
    r1 = i*h; r2 = (i+1)*h;
    A[i,i+1] = A[i,i+1] + intergral(a1,b1,a2,b2,r1,r2);
"%%%"
" b vector"
b = np.array([(np.zeros(node_number))])
for i in range(0,node_number):
    b[0,i] = 0
b = np.transpose(b)
b[node_number-1,0] = 1
for i in range(0,node_number):
    A[N,i] = 0
A[N,N] = 1
"%%%"
"Solve A u = b"
u_num = np.linalg.solve(A,b)
" ============================Exact solution"
""""
fig, ax = pyplot.subplots()
r_vector = np.linspace(0,1,100)
u_exact_vector = special.i0(r_vector) / special.i0(1)
ax.plot(r_vector,u_exact_vector,'r',label = 'Exact Solution')
ax.plot(grid,u_num,'-o',label = 'Numerical Solution')
ax.legend()
ax.set_xlabel('r')
ax.set_ylabel('u')
fig.savefig('pde.pdf')
""""
" ============================Calculate the Error"
u_exact = np.array([special.i0(grid) / special.i0(1)]);
u_exact = np.transpose(u_exact)
ur_exact = special.i1(grid) / special.i0(1);
" L2 norm"
```

```
    u_error_L2 = np.sqrt(np.sum((u_num - u_exact)**2*h))
    " Calculate ur"
    ur_num = np.zeros(node_number)
    ur_num[0] = (u_num[1]-u_num[0])/h
    ur_num[N] = (u_num[N]-u_num[N-1])/h
    for i in range(1,N):
        ur_num[i] = (u_num[i+1]-u_num[i-1])/2/h
    ur_error_L2 = np.sqrt(np.sum((ur_num - ur_exact)**2*h))
    " H1 norm"
    u_error_H1 = np.sqrt(u_error_L2**2+ur_error_L2**2)
    return [u_error_L2,u_error_H1]
element_vector = []
L2 = []
H1 = []
for i in range(0,25):
    element_number = i * 4 + 4
    element_vector.append(element_number)
    error = FEM(element_number)
    L2.append(error[0])
    H1.append(error[1])
fig, ax = pyplot.subplots()
ax.plot(np.log(element_vector),np.log(L2),'-',label = 'L2 norm Error')
ax.plot(np.log(element_vector),np.log(H1),'--',label = 'H1 norm Error')
ax.legend()
ax.set_xlabel('ln(N)')
ax.set_ylabel('ln(Error)')
fig.savefig('pde.pdf')
```