SDSC PA2

Josyula V N Taraka Abhishek - 200010021
M V Karthik - 200010030

November 2022

Contents

List of Figures

1 Analytical Solution

1.1 Constant Area

We know the equation:

$$EA\frac{d^2u}{dx^2} + F = 0 \tag{1}$$

Where F is bA where b is body force and A is the area of the cross section. We know that body force in out context is zero, Hence the equation will become

$$\frac{d^2u}{dx^2} = 0 \tag{2}$$

For Which the general solution is

$$u = c_1 + c_2x \tag{3}$$

Boundary conditions given are $u = 0$ at $x = L$ and $EA\frac{du}{dx} = P$ where P is the force applied at the end of the rod. Solving the equation gives us:

$$u = \frac{P}{EA}(L - x) \tag{4}$$

1.2 Variable Area

we know the equation governing the given situation is

$$\frac{d(EA\frac{du}{dx})}{dx} = 0 \tag{5}$$

Here A is given as function of x. Using multiplication rule, and with help of boundary conditions as above we get

$$u = \frac{PL}{A_0E}[ln2 - ln(1 + \frac{x}{L})] \tag{6}$$

2 Finite Element Method

The stiffness matrix of each element here is $2X2$ matrix(2-noded rode element). The stiffness matrix of ith element is as follows

$$k^i = K = \begin{bmatrix} k_{11} & k_{11} \\ k_{21} & k_{22} \end{bmatrix} \tag{7}$$

where $k^l_{ij} = \int_{x_a}^{x_b} EA\frac{dN_i}{dx}\frac{dN_j}{dx}dx$, x_a, x_b are being end points of lth element. Merging all the element matrices to get global stiffness matrix K

$$K = \begin{bmatrix} k^1_{11} & k^1_{11} & 0 & \dots & 0 \\ k^1_{21} & k^1_{22} + k^2_{11} & \dots & & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & k^N_{22} \end{bmatrix} \tag{8}$$

The force vector F is

$$F^i = \begin{bmatrix} \left[N_1 EA \frac{d\widetilde{u}}{dx}\right]_0^L + \int_0^L N_1 F dx \\ \left[N_2 EA \frac{d\widetilde{u}}{dx}\right]_0^L + \int_0^L N_2 F dx \end{bmatrix} \tag{9}$$

Force vector in our case becomes for N elements

$$F = \begin{bmatrix} 5000 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{10}$$

We know that $K * U = F$ where $U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ u_N \end{bmatrix}$. Goal is to find U matrix

Applying Finite Element Method using below equations we the value of U.

$$\begin{aligned} K * U &= F \\ U &= K^{-1} * F \end{aligned} \tag{11}$$

To make K non-singular we are stripping off the last row and last column of K and last row of F and U matrices. Here the integration is done by guass-quadrature two point method and matrix inverse and multiplication are done using Eigen library.

3 Plots

X vs U graph for constant area:

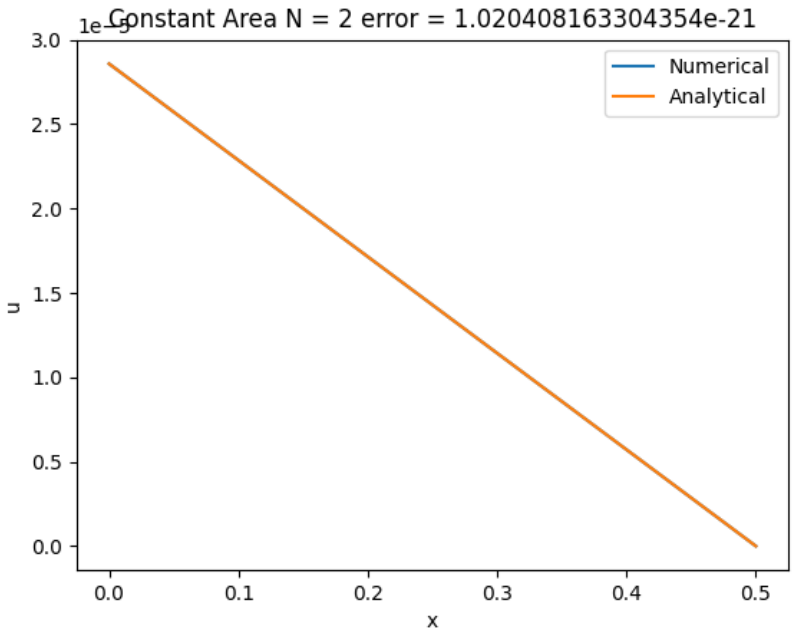
Constant Area N = 2 error = 1.020408163304354e-21
1e−5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
u
0.0
0.1
0.2
0.3
0.4
0.5
x
Numerical
Analytical

Figure 1: N=2

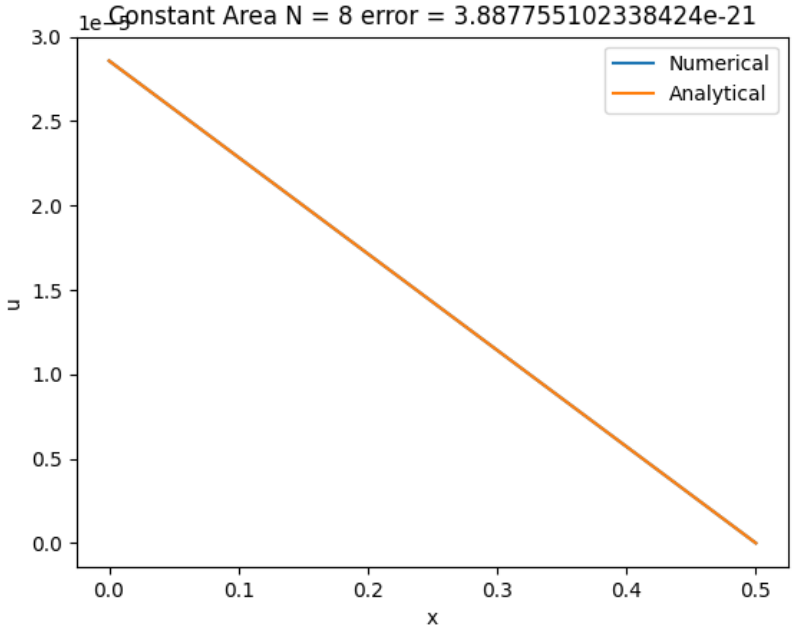
Constant Area N = 8 error = 3.887755102338424e-21
1e−5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
u
0.0
0.1
0.2
0.3
0.4
0.5
x
Numerical
Analytical

Figure 2: N=8

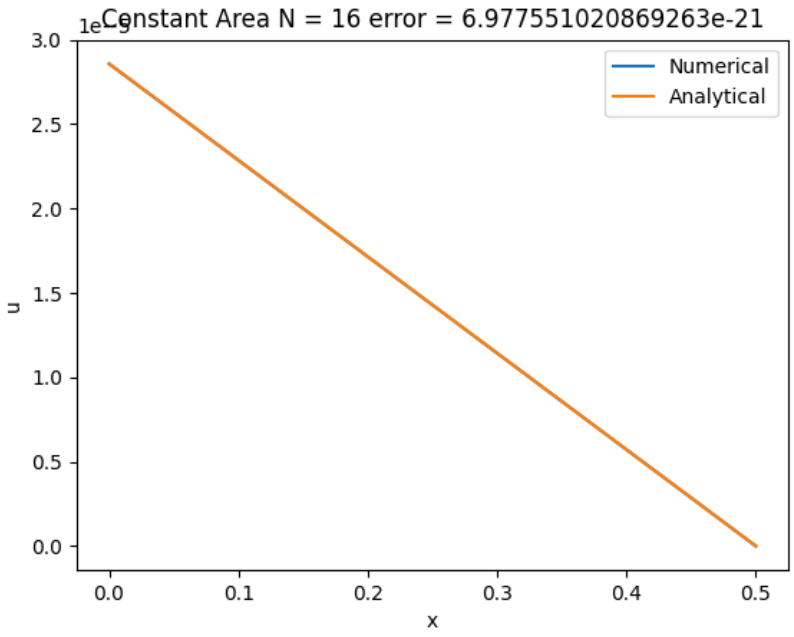
Constant Area N = 16 error = 6.977551020869263e-21
1e−9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
u
0.0
0.1
0.2
0.3
0.4
0.5
x
Numerical
Analytical

Figure 3: N=16

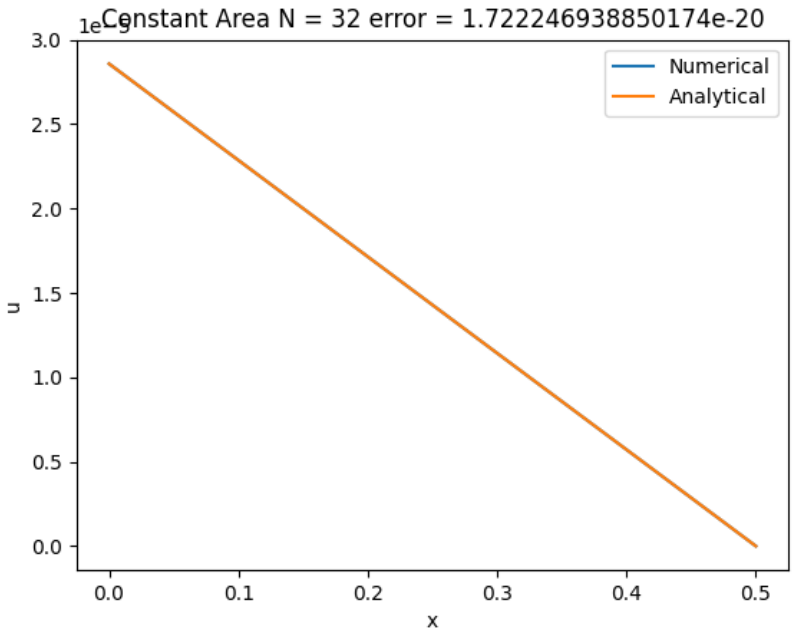
Constant Area N = 32 error = 1.722246938850174e-20
1e−9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
u
0.0
0.1
0.2
0.3
0.4
0.5
x
Numerical
Analytical

Figure 4: N=32

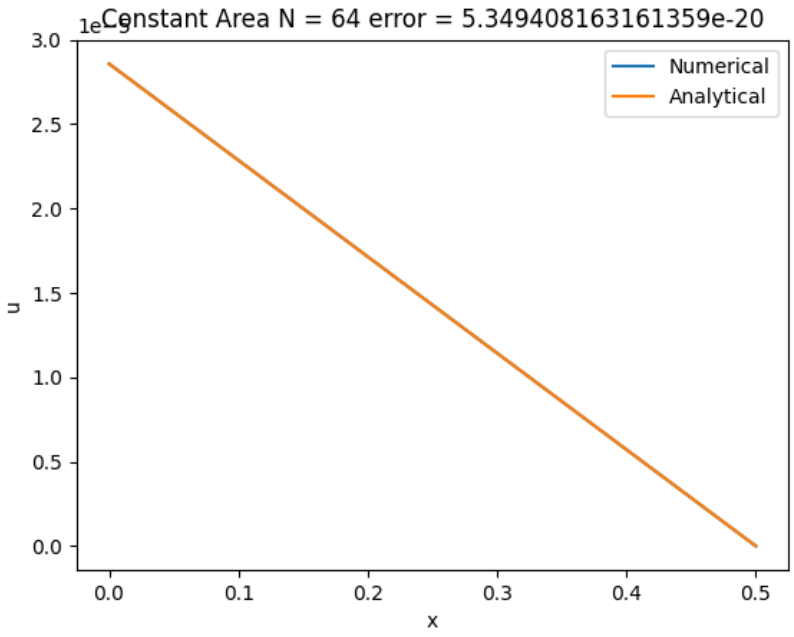
Constant Area N = 64 error = 5.349408163161359e-20
1e−9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
u
0.0
0.1
0.2
0.3
0.4
0.5
x
Numerical
Analytical

Figure 5: N=64

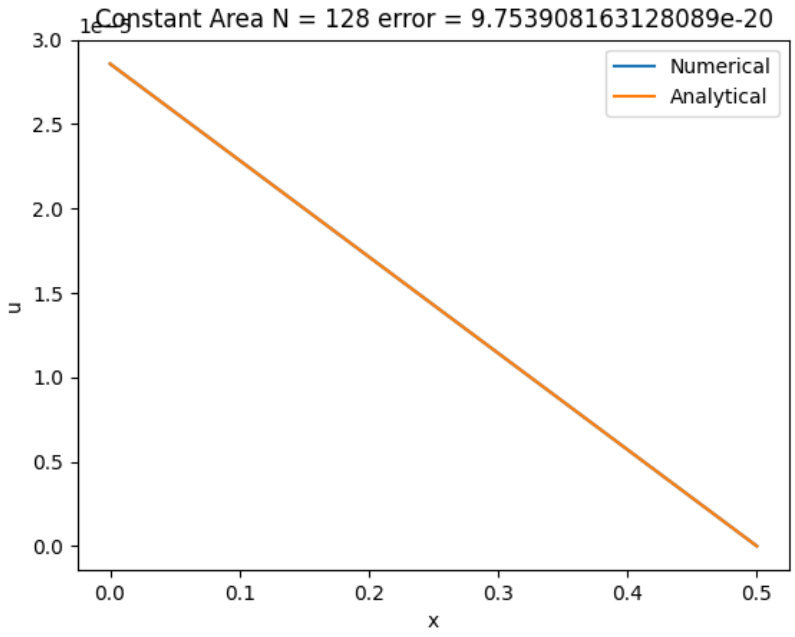

Figure 6: N=128

X vs U graph for variable area:

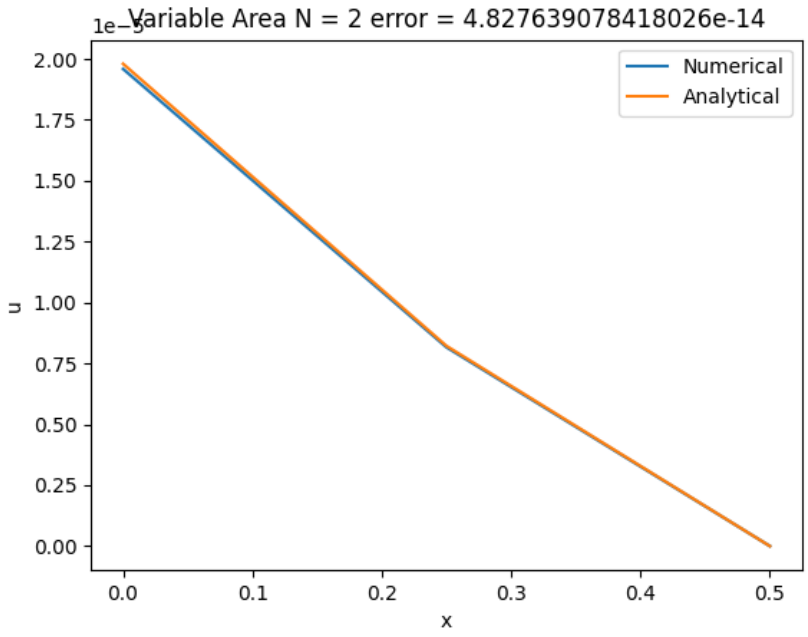

Figure 7: Variable Area N=2

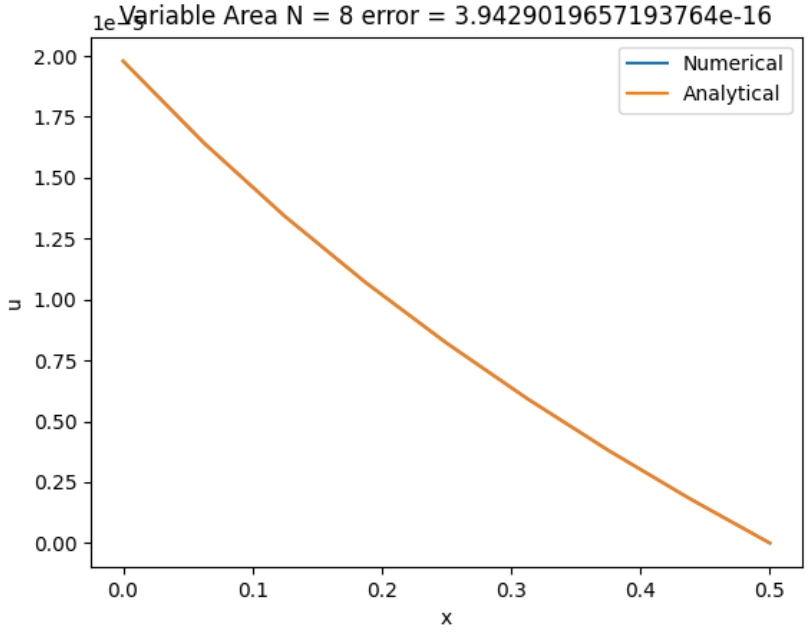

Figure 8: Variable Area N=8

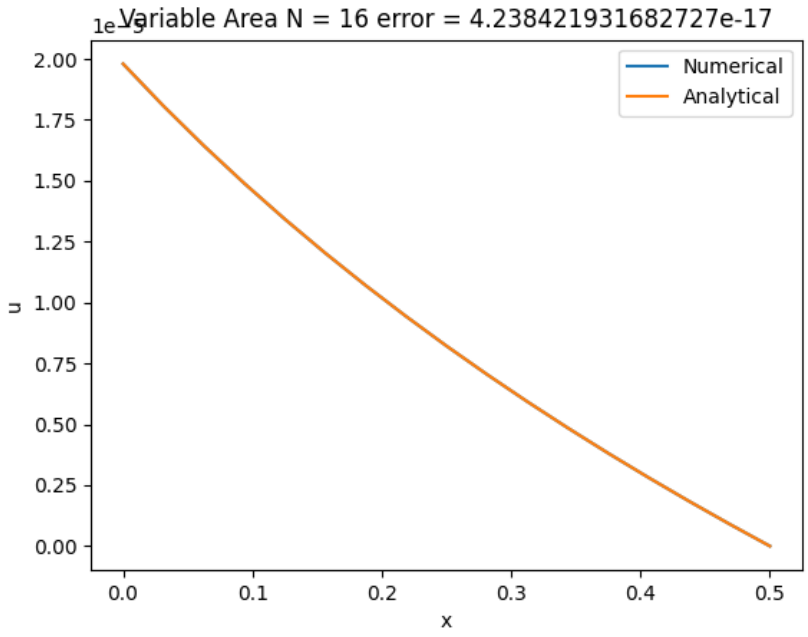

Figure 9: Variable Area N=16

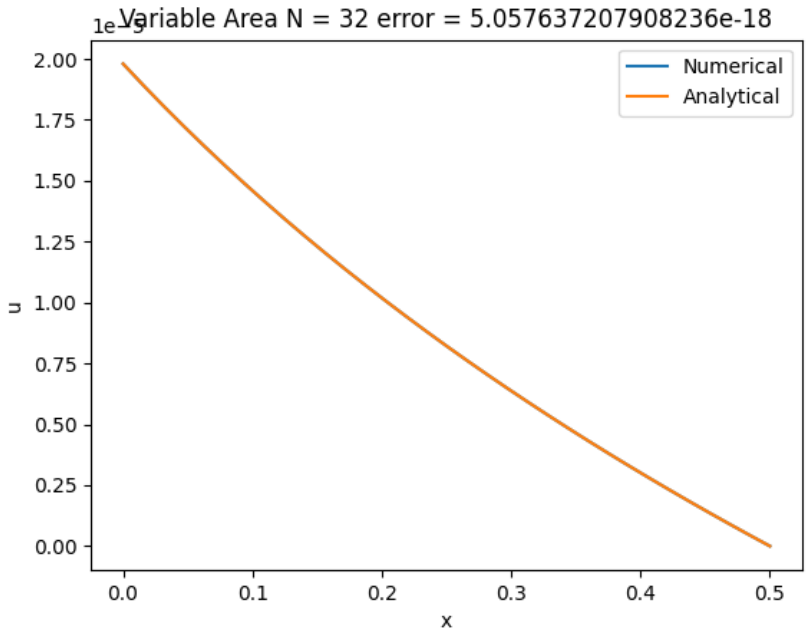

Figure 10: Variable Area N=32

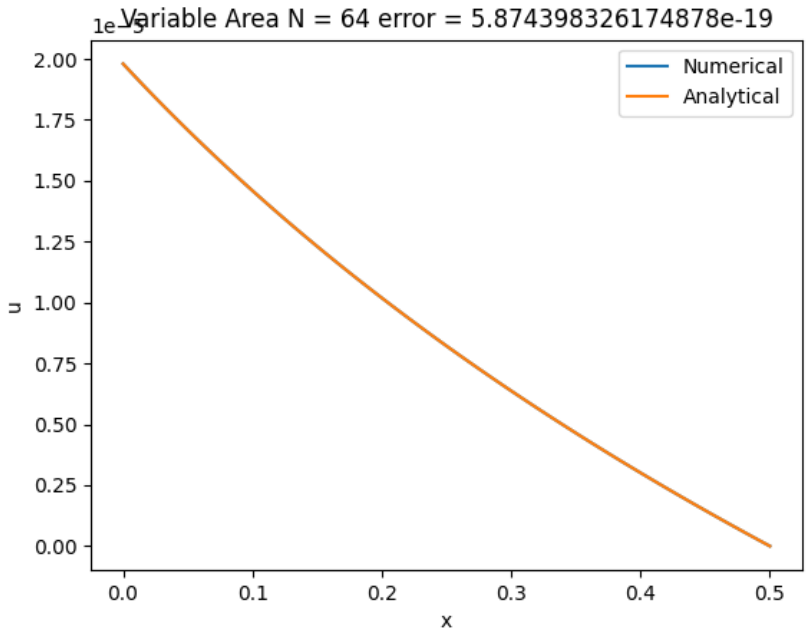

Figure 11: Variable Area N=64

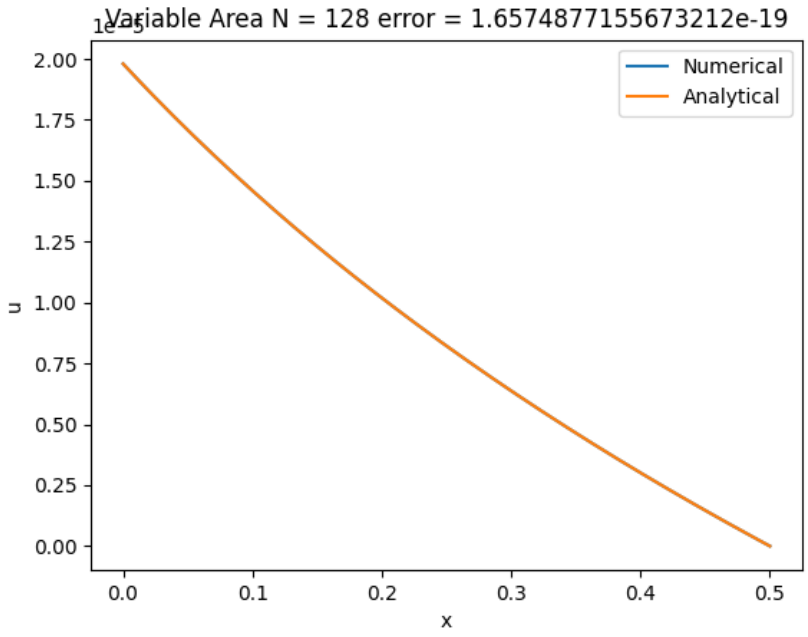

Figure 12: Variable Area N=128

4 Appendix

```
std::vector<std::vector<double>> Kvalues(N, std::vector<double>(4));
```

```
for (int i = 0; i < N; i++)
{
    double xa = i*h;
    double xb = (i+1)*h;
    double integral =
    ((Y*Ao)/ (h*h))*p1.twoPointGuassQuadratureIntegral(xb,xa);

    Kvalues[i][0] = integral;
    Kvalues[i][1] = -integral;
    Kvalues[i][2] = -integral;
    Kvalues[i][3] = integral;
```

```
}


for (int i = 0; i < N+1; i++)
{
    for (int j = 0; j < N+1; j++)
    {
        if (i == 0 && j == 0)
        {
            K(i,j) = Kvalues[0][0];
        }
        else if(i == N  && j == N){
            K(i,j) = Kvalues[N-1][3];
        }
        else if (i == j)
        {
            K(i,j) = Kvalues[i-1][3] + Kvalues[i][0];
        }
        else if (i == j-1)
        {
            K(i,j) = Kvalues[i][1];
        }
        else if (i == j+1)
        {
            K(i,j) = Kvalues[i-1][2];
        }
    }
}


F(0,0) = P;

// F.resize(N,1);
  Eigen::MatrixXd newK = K.block(0,0,N,N);
  Eigen::MatrixXd newF = F.block(0,0,N,1);
  Eigen::MatrixXd U = newK.inverse()*newF;
```

5 conclusion

The 2-norm error for N = 2,16,64,128 is in order of 10^{-20} when A is constant and also 10^{-19} when A is variable as given function. Hence the FEM method on given situation is highly accurate even if number elements discretized are less and the error decreases with increase in number of elements N.