CS601 : Software Development for Scientific Computing

Assignment 2

Sourabh Bhosale 200010004
Dibyashu Kashyap 200010013

November 8, 2022

1 Problem Statement

Consider a rod with cross sectional area $A(x)$ and length L. The rod is subjected to a constant load $P = 5000N$ at $x = 0$. At $x = L$ the rod is fixed. The length of the rod is $0.5m$ and the Young's modulus of the material of the rod is $70GPa$. Consider two problems:

1. The cross section of the rod is uniform with area $A(x) = A_0 = 12.5 * 10^4 m^2$
2. The cross sectional area is given by the formula $A(x) = A_0(1 + x/L)$. Here the cross section is not uniform, it increases linearly with x.

2 Code

```
// Program to illustrate the working of
// objects and class in C++ Programming
#include <iomanip>
#include <cmath>
#include <iostream>
#include <fstream>
#include <functional>
using namespace std;
double integrate(double a, double b, function<double(int, double, double)>
    f1,function<double(int, double, double)> f2,
    function<double(int, double, double)> f3,int i1,int i2,int i3,double le){
    double p = (b - a) / 2;
    double q = (b + a) / 2;

    double sum = 0;

    sum += f1(i1,p * sqrt(1 / 3) + q,le)*f2(i2,p * sqrt(1 / 3)
      + q,le)*f3(i3,p * sqrt(1 / 3) + q,le);
    sum += f1(i1,-1 * p * sqrt(1 / 3) + q,le)*f2(i2,p * sqrt(1 / 3)
      + q,le)*f3(i3,p * sqrt(1 / 3) + q,le);

    return p * sum;
}

double func_n(int i, double x, double le){
    if(i==1) return 1-x/le;
    if(i==2) return x/le;
    return 0;
}
double areaf(int i, double x, double le){
    return 1+x/(le*i);
}
double I(int i, double x, double le){
    return 1;
}
double func_dn(int i, double x, double le){
    if(i==1) return -1/le;
    if(i==2) return 1/le;
    return 0;
}
double func_anlitical_soln_const_area(double x, double f,double a, double y)
{
    return f*x/(a*y);
}
```

```
double func_anlitical_soln_var_area(double x, double f,double a,
    double y, double l){
    return f*l*log(1+x/l)/(a*y);
}
// create a class
class domain{
public:
    int elementindex;
    double xa;
    double xb;
    double k[2][2];
    double b[2];
public:
    // parameterized constructor to initialize variables
    void domainfoo(double x_a, double x_b, double le, int index,
    double E, double a, double fba,int N){
        xa = x_a;
        xb = x_b;
        double d = xb - xa;
        // k[0][0]=(E*a/(le*le))*(xb-xa);
        //k[0][0] = (E * a / (le * le)) * d;

        //for linear area
        k[0][0] =E * a *integrate(xa,xb,&areaf,&func_dn,&func_dn,N,1,1,le);
        k[0][1] =E * a *integrate(xa,xb,&areaf,&func_dn,&func_dn,N,1,2,le);
        k[1][0] = k[0][1];
        k[1][1] = k[0][0];
        b[0] = fba*integrate(xa,xb,&areaf,&I,&func_n,N,1,1,le);
        b[1] = fba*integrate(xa,xb,&areaf,&I,&func_n,N,1,2,le);
    }
};
int N = 8;
// function to find rms value
double rmsValue(double arr1[], double arr2[])
{
        double square1 = 0.0, square2 = 0.0;
        double diff = 0.0, root = 0.0;
        for (int i = 0; i < N; i++) {
                diff += pow(arr1[i] - arr2[i], 2);
        }
        // Calculate Root.
        root = sqrt(diff);
        return root;
}
```

```
int main(){
    // creating the file called 'file.txt'.
    ofstream MyFile("file.txt");
    // create object of Room class
    //int N = 8;
    double L = 0.5;
    double le = L / N;
    double l_a = 0;
    double l_b = le;
    double E = 70000000000;
    double A = 0.00125;
    double f = 5000;
    double z=integrate(0,le,&I,&func_dn,&func_dn,0,1,1,le);
    cout<<"Z is "<<z<<endl;
    domain rod_element[N];
    int i = 0;
    while (i < N){
        // rod_element[i].elementindex = 42;
        // rod_element[i].xa = 30.8;
        // rod_element[i].xb = 19.2;
        rod_element[i].domainfoo(l_a, l_b, le, i, E, A, f,N);
        l_a += le;
        l_b += le;
        i++;
    }
    // assign values to data members
    double kg[N][N];
    double bg[N];
    i = 0;
    while (i < N){
        int j = 0;
        while (j < N){
            kg[i][j] = 0;
            j++;
        }
        bg[i] = 0;
        i++;
    }
    double x[N - 1];
    // double fg[n];
    double f1 = 5000;
    double u[N];
    double a[N - 1][N];
    int m = 0;
```

```
while (m < N){
    kg[m][m] += rod_element[m].k[0][0];
    kg[m][m + 1] += rod_element[m].k[0][1];
    kg[m + 1][m] += rod_element[m].k[1][0];
    kg[m + 1][m + 1] += rod_element[m].k[1][1];
    bg[m] += rod_element[m].b[0];
    bg[m + 1] += rod_element[m].b[1];
    m++;
}
i = 0;
while (i < N - 1){
    int j = 0;
    while (j < N - 1){
        a[i][j] = kg[i][j];
        cout << a[i][j] << setw(16);
        MyFile << a[i][j] << setw(16);
        j++;
    }
    cout << "\n";
    MyFile << "\n";
    i++;
}
int j = 0;
while (j < N - 1){
    a[j][N - 1] = bg[j];
    j++;
}
a[0][N-1]+=f1;
//...........
int n = N - 1;
int k;
for (i = 0; i < n; i++) // Pivotisation
    for (k = i + 1; k < n; k++)
        if (abs(a[i][i]) < abs(a[k][i]))
            for (j = 0; j <= n; j++)
            {
                double temp = a[i][j];
                a[i][j] = a[k][j];
                a[k][j] = temp;
            }
cout << "\nThe matrix after Pivotisation is:\n";
MyFile << "\nThe matrix after Pivotisation is:\n";
for (i = 0; i < n; i++) // print the new matrix
{
    for (j = 0; j <= n; j++)
        cout << a[i][j] << setw(16);
        MyFile << a[i][j] << setw(16);
    cout << "\n";
    MyFile << "\n";
}
```

```
    for (i = 0; i < n - 1; i++) // loop to perform the gauss elimination
        for (k = i + 1; k < n; k++){
            double t = a[k][i] / a[i][i];
            for (j = 0; j <= n; j++)
                a[k][j] = a[k][j] - t * a[i][j]; // make the elements below the pivo
        }
    cout << "\n\nThe matrix after gauss-elimination is as follows:\n";
    MyFile << "\n\nThe matrix after gauss-elimination is as follows:\n";
    for (i = 0; i < n; i++) // print the new matrix{
        for (j = 0; j <= n; j++)
            cout << a[i][j] << setw(16);
            MyFile << a[i][j] << setw(16);
        cout << "\n";
        MyFile << "\n";
    }
    for (i = n - 1; i >= 0; i--) // back-substitution
    {                            // x is an array whose values correspond to the val
        x[i] = a[i][n];          // make the variable to be calculated equal to the
        for (j = i + 1; j < n; j++)
            if (j != i) // then subtract all the lhs values except the coefficient o
                x[i] = x[i] - a[i][j] * x[j];
        x[i] = x[i] / a[i][i]; // now finally divide the rhs by the coefficient of t
    }
    cout << "\nThe values of the variables are as follows:\n";
    MyFile << "\nThe values of the variables are as follows:\n";
    for (i = 0; i < n; i++)
        cout << x[i] << endl; // Print the values of x, y,z,....
        MyFile << x[i] << endl;

    //.......
    double anlitcal_sol_cont_area[N];
    double anlitcal_sol_var_area[N];

    double temp_x=le;
           cout << "N:"<< N<<endl;
           MyFile << "N:"<< N<<endl;

    for (i = 0; i < N; i++) // print the new Analytical solution{
      anlitcal_sol_cont_area[i]=func_anlitical_soln_const_area(
          temp_x,f1,A,E);
      anlitcal_sol_var_area[i]=func_anlitical_soln_var_area(temp_x,
            f1,A,E,L);
        temp_x+=le;
       cout <<  anlitcal_sol_cont_area[i] <<"   "<< anlitcal_sol_var_area[i]<<endl;
       MyFile <<  anlitcal_sol_cont_area[i] <<"   "<< anlitcal_sol_var_area[i]<<endl
    }
```

```
    // CPP program to calculate Root Mean Square

    // int sizearr1 = sizeof(anlitcal_sol_cont_area) /
     //     sizeof(anlitcal_sol_cont_area[0]);
    // int sizearr2 = sizeof(anlitcal_sol_var_area) /
     //     sizeof(anlitcal_sol_var_area[0]);

    // cout <<"RMS value of anlitcal_sol_cont_area is: " <<
      //  rmsValue(anlitcal_sol_cont_area, sizearr1)<<endl;
    // cout <<"RMS value of anlitcal_sol_var_area is: " <<
     //   rmsValue(anlitcal_sol_var_area, sizearr2)<<endl;

    cout <<"\n"<<"RMS error: " << rmsValue(anlitcal_sol_cont_area,
        anlitcal_sol_var_area)<<endl;
    MyFile <<"\n"<<"RMS error: " << rmsValue(anlitcal_sol_cont_area,
        anlitcal_sol_var_area)<<endl;

    // closing the file
    MyFile.close();

    return 0;
}
```

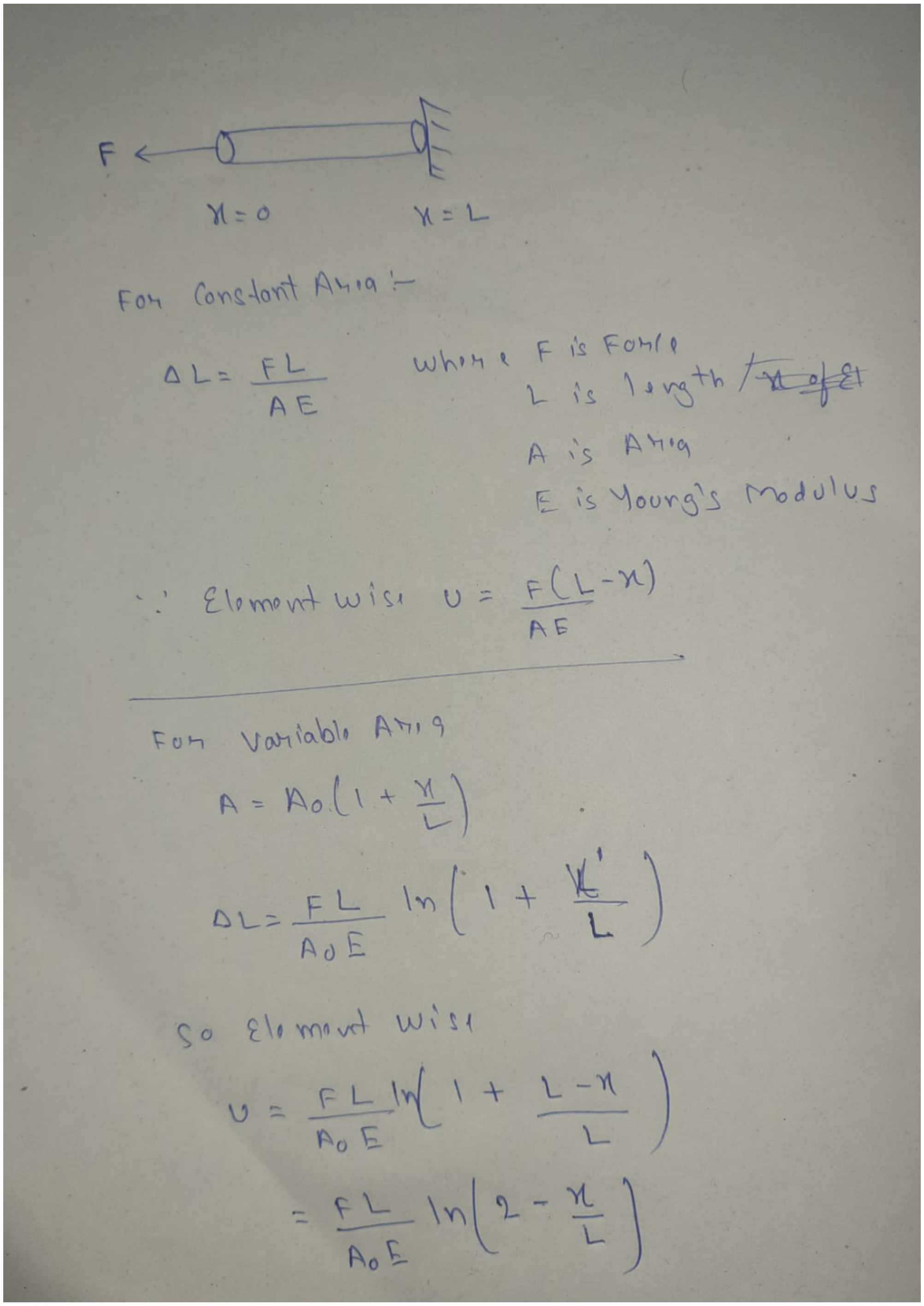
3 Problem 3

Write an Finite Element code to find the displacement at the nodal points on the rod. You need to discretize the rod into N = 2, 8, 32, 128 elements of equal length for problems in 1 and 2.

For implementing this part we divide domain into elements and making them into objects with xa, xb, K,B as attributes. them we calculated values of attributes of element object. Then we assembled these values to get various Global matrices and applied gauss elimination to find our final displacement (U) values.

4 Problem 4

Find analytical solution of the problems stated above. For constant area:-


F

$x = 0$ $\qquad$ $x = L$

For Constant Area :-

$$\Delta L = \frac{FL}{AE}$$

where F is Force
L is length / ~~x of El~~
A is Area
E is Young's Modulus

$\because$ Element wise $\quad U = \frac{F(L-x)}{AE}$

For Variable Area

$$A = A_0\left(1 + \frac{x}{L}\right)$$

$$\Delta L = \frac{FL}{A_0 E} \ln\left(1 + \frac{x'}{L}\right)$$

So Element wise

$$U = \frac{FL}{A_0 E} \ln\left(1 + \frac{L-x}{L}\right)$$

$$= \frac{FL}{A_0 E} \ln\left(2 - \frac{x}{L}\right)$$

Figure 1:

5 Problem 5

Plot and compare your solutions, both analytical and Numerical. Write your observations. Plot the error: norm(uN uA) over the length of the rod. Here uN represents numerical solution and uA represents analytical solution.

For the first case where area is constant behaviour of both the curves will be same as both U's vary linearly with x But in the second case U varies linearly for numerical solution but logarithmicly for analytical solution.

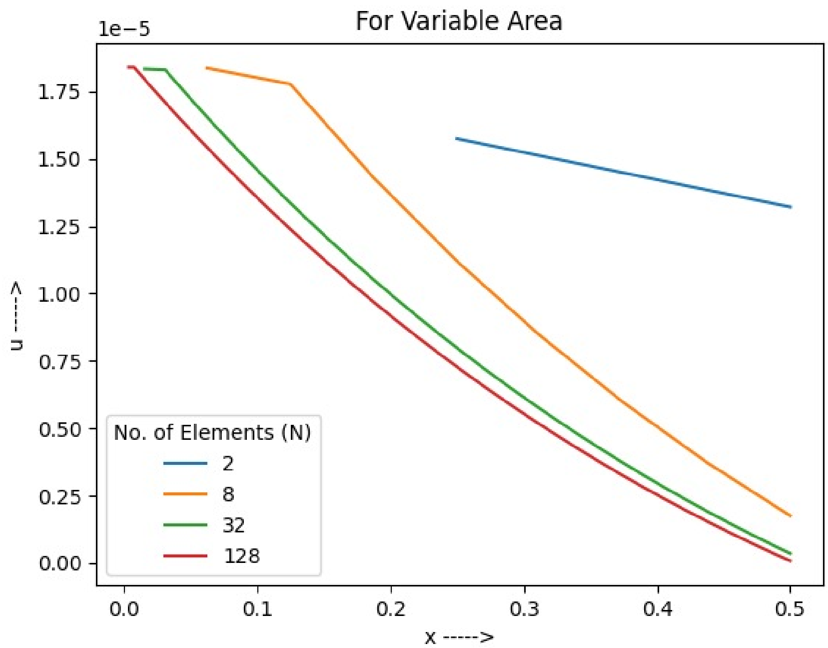


Figure 2: Plot 1

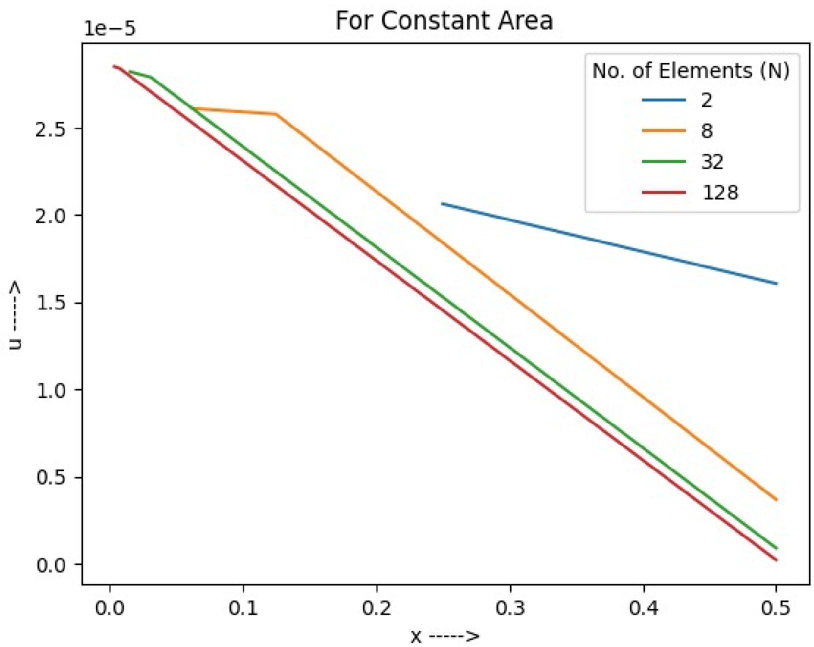


Figure 3: Plot 2

6 Conclusions

We were able to deduce the following points from the above analyses,

- The displacements of nodal points were almost same for analytical and numerical approach, and we can have error as almost 0.
- We get almost 0 error as FEM approximation and analytical approximations are linear in nature but will get some error for small values N when area is not constant and as here will lead to non liner change of U wrt x.
- We can infer the above conclusions with plots of uniform cross section as both curve overlap throughout when area was constant.
- The error in the nodal displacements decreases and accuracy of FEM increases as the number of elements.