Assignment: Solving a Non-Linear Equation

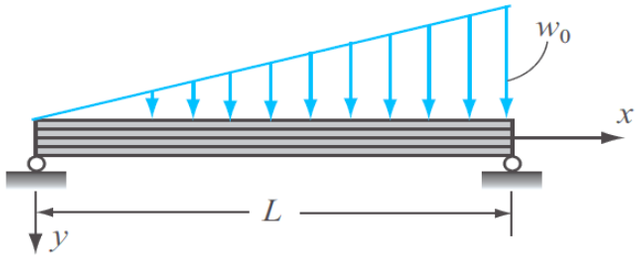
Name: 안견힐 ID: 21900416

Instructions:

You must submit, report file and program file: “myNM.h”, “myNM.c”, “Assignment_nonlinear.c” on Hisnet.

Problem: Solve the following non-linear equation [30pt]

A simply supported I-beam is loaded with a distributed load, as shown. The deflection, y, of the center line of the beam as a function of the position, x, is given by the equation:



From the solution of the equation of motion for this model, the steady-state up-and-down motion of the car (mass) is given by x(t), due to the wheel motion of y(t). The ratio between amplitude *X* and amplitude *Y* is given by:

$$y = \frac{w_0 x}{360LEI}(7L^4 - 10L^2x^2 + 3x^4)$$

where L=4m is the length, E=70 GPa is the elastic modulus, I=52.9x10^{-6} m^4 is the moment of inertia, and w_0=20 kN/m.

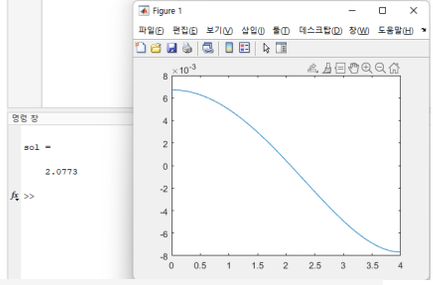
Find the position *x* where the deflection at the point where the deflection of the beam is maximum and determine the deflection at this point.

HINT: The maximum deflection is at the point where dy/dx=0=f(x). First, find dy/dx=f(x). Then solve for f(x).

a. Solve for the solution using MATLAB’s functions of *fzero( )* [5pt]

b. Use your defined Newton-Rhapson method to solve for the solution. [20pt]

c. Compare the result with MATLAB’s solution [5pt]




```
clc; clear; close all;

L=4;
E=70*10^9;
I=52.9*10^-6;
W0=20*10^3;
x=0:0.001:L;

y=(W0.*x/(360*L*E*I)).*(7*(L^4)-10*(L^2).*(x.^2)+3.*x.^4);
F=inline('(20000.*x/(360*4*70*10^9*52.9*10^-6)).*(7*(4^4)-10*(4^2).*(x.^2)+3.*x.^4)',x);
dF=inline('(20000/(360*4*70*10^9*52.9*10^-6)).*(7*(4^4)-10*(4^2).*(x.^2)+3.*x.^4)+(20000.*x/(360*4*70*10^9*52.9*10^-6)).*(-2*10*(4^2).*x+3.*4*x.^3)',x);
% F'

plot(x,dF(x));
x0=3;              %initial x

sol=fzero(dF,x0)   %dy/dx=0=f(x)
```

Procedure:

You need to create the main source file ('Assignment_nonlinear.c'). You should fill in your codes on the main source file and in your library. Show the output results on the report.

Upload (1) Assignment_nonlinear.c, (2) myNM.h, (3) myNM.c) as a zip file

- Create newtonRaphson function to solve a non-linear equation.

```
double newtonRaphson (double func(double x), double dfunc(double x),
                      double x0, double tol)
```

  - The declaration of the function must be in "myNP.h"
  - The definition of the function must be in "myNP.cpp"

- User defined non-linear function, f(x), and the derivative function f'(x) must be defined in main source file (e.g. .Assignment_nonlinear.c)

```
double func(double x);
double dfunc(double x);
```

- Show the output results by capturing the output screen by running the main source. Display the output value of x(n), iteration number, and the relative error or tolerance.

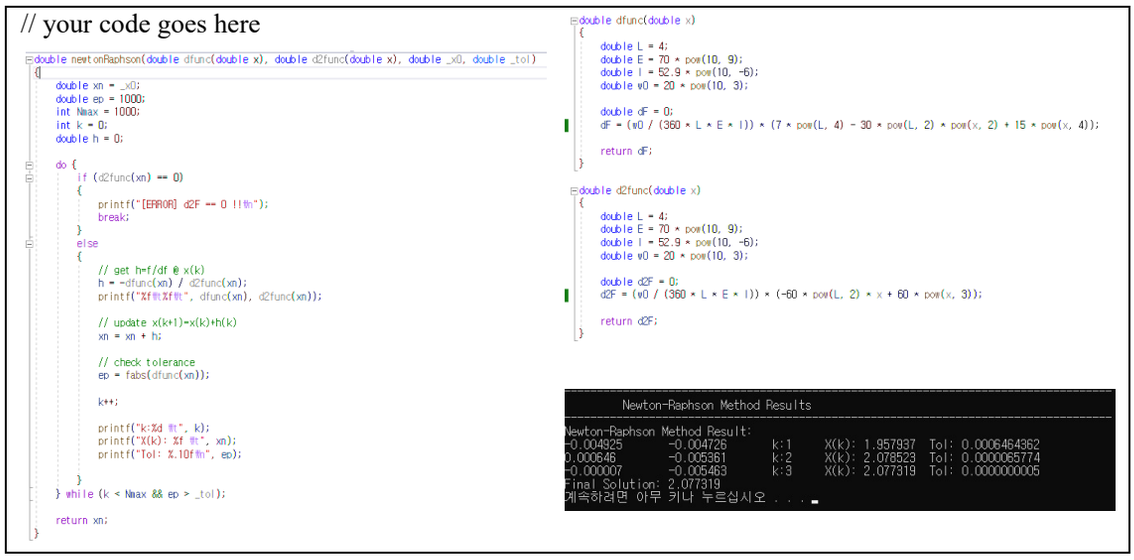
  Example)

```
--------------------------------------------------------------------------------
                    Newton-Raphson Method Results
--------------------------------------------------------------------------------
Newton-Raphson Method Result:
Iteration:0     X(n): -10.000000        Tolerance: 1.0000000000
Iteration:1     X(n): -5.764706         Tolerance: 0.7346938776
Iteration:2     X(n): -3.661663         Tolerance: 0.5743407933
Iteration:3     X(n): -2.638657         Tolerance: 0.3876993534
Iteration:4     X(n): -2.179107         Tolerance: 0.2108892813
Iteration:5     X(n): -2.023619         Tolerance: 0.0768367851
Iteration:6     X(n): -2.000533         Tolerance: 0.0115399761
Iteration:7     X(n): -2.000000         Tolerance: 0.0002662004
Final Solution: -2.000000
계속하려면 아무 키나 누르십시오 . . .
```

Pseudo Code

```
Double NewtonRapson ()
  while(K < N_max And ε > tol
  {  h = -dfunc(xn) / func(xn)
     xn = xn + h
     ε = | dfunc(xn) |
     K++
  }
```

- Paste your code of newtonRaphson()



// your code goes here

```
double newtonRaphson(double dfunc(double x), double d2func(double x), double _x0, double _tol)
{
    double xn = _x0;
    double ep = 1000;
    int Nmax = 1000;
    int k = 0;
    double h = 0;

    do {
        if (d2func(xn) == 0)
        {
            printf("[ERROR] d2F == 0 !!\n");
            break;
        }
        else
        {
            // get h=f/df @ x(k)
            h = -dfunc(xn) / d2func(xn);
            printf("%f\t%f\t", dfunc(xn), d2func(xn));

            // update x(k+1)=x(k)+h(k)
            xn = xn + h;

            // check tolerance
            ep = fabs(dfunc(xn));

            k++;

            printf("k:%d \t", k);
            printf("X(k): %f \t", xn);
            printf("Tol: %.10f\n", ep);

        }
    } while (k < Nmax && ep > _tol);

    return xn;
}
```

```
double dfunc(double x)
{
    double L = 4;
    double E = 70 * pow(10, 9);
    double I = 52.9 * pow(10, -6);
    double w0 = 20 * pow(10, 3);

    double dF = 0;
    dF = (w0 / (360 * L * E * I)) * (7 * pow(L, 4) - 30 * pow(L, 2) * pow(x, 2) + 15 * pow(x, 4));

    return dF;
}

double d2func(double x)
{
    double L = 4;
    double E = 70 * pow(10, 9);
    double I = 52.9 * pow(10, -6);
    double w0 = 20 * pow(10, 3);

    double d2F = 0;
    d2F = (w0 / (360 * L * E * I)) * (-60 * pow(L, 2) * x + 60 * pow(x, 3));

    return d2F;
}
```

```
        Newton-Raphson Method Results
-------------------------------------------------------------------------------
Newton-Raphson Method Result:
-0.004925       -0.004726       k:1     X(k): 1.957937  Tol: 0.0006464362
0.000646        -0.005361       k:2     X(k): 2.078523  Tol: 0.0000065774
-0.000007       -0.005463       k:3     X(k): 2.077319  Tol: 0.0000000005
Final Solution: 2.077319
계속하려면 아무 키나 누르십시오 . . .
```

HINT:

- Need to select MaxItr to prevent infinite loop errors
- Calculate for *f'(x)* analytically. If it is too complex to find *f'(x)*, then use the secant method technique to estimate *f'(x)*.
- Try to make a fail-safe program with error-handling techniques.
  - Check(Try-Catch) for possible errors such as *f'(x)==0*
- Example code for using MATLAB *fzero(function, x0)*

```
FUN = @ (x) 8-4.5*(x-sin(x));
x0=2;
x=fzero(FUN,x0)
```

- Examples of defining functions in C.

  Eg. $y = x^2 - x$

```
double func(x){
     return x*x-x;}

double dfunc(x){
    return 2*x-1;}
```