Assignment: Solving a Non-Linear Equation

Name: ID:

Instructions:

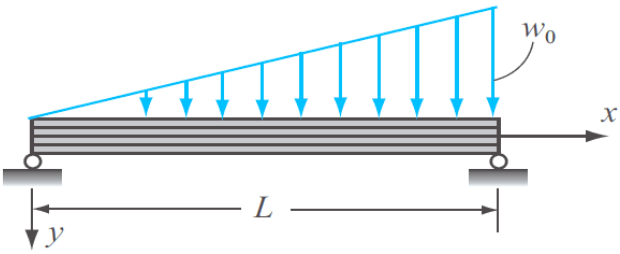
Declare and define the numerical method functions in your header file "myNM.h", "myNM.c", respectively.

The main program file, 'Assignment_nonlinear.c', contains the main function and calls yours NM functions to solve the assignment problems.

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

Problem: Solve the following non-linear equation [20pt]

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:

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

where L=4m is the length, E=70 GPa is the elastic modulus, $I=52.9*10^{-6}$ 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).

a. Solve for the solution using MATLAB's functions of *fzero( )*

b. Use your defined Newton-Rhapson method to solve for the solution.

c. Compare the result with MATLAB's solution

d. Compare the results of method a) with bisection method: (final answer, number of iterations, tolerance)

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. Also, you must upload the modified (Assignment_nonlinear.c, myNM.h, myNM.c) as a zip file on HISNET.

- The declaration of the required functions are expressed in "myNM.h"

```c
double bisectionNL(float _a, float _b, float _tol);   // given in tutorial
double newtonRaphson (float _x0, float _tol);
```

- Define *newtonRahpson()* function as a non-linear solver in file " myNM.c".
- The above functions call user defined non-linear function, f(x), and the derivative function f'(x).

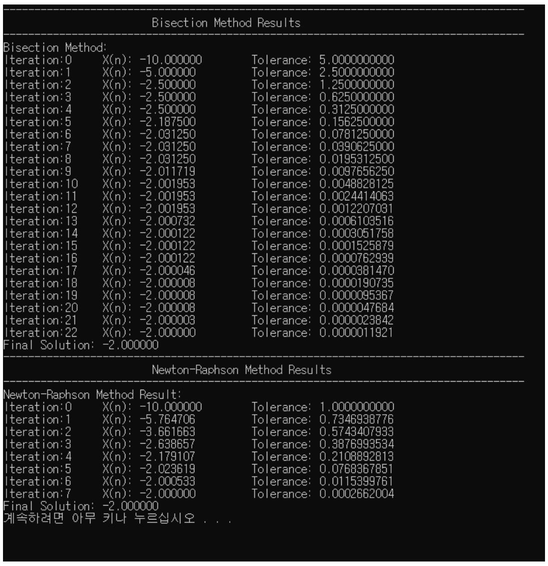
```c
float func(float _x);
float dfunc(float _x);
```

(For this problem, you can easily get df/dx expression).

You can also pass a function (or class) of f(x) and f'(x) into the function bisectionNL( ) or newtonRaphson() as the input argument, if you know how to do it.

- 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)



```
-----------------------------------------------------------------------
                    Bisection Method Results
-----------------------------------------------------------------------
Bisection Method:
Iteration:0     X(n): -10.000000        Tolerance: 5.0000000000
Iteration:1     X(n): -5.000000         Tolerance: 2.5000000000
Iteration:2     X(n): -2.500000         Tolerance: 1.2500000000
Iteration:3     X(n): -2.500000         Tolerance: 0.6250000000
Iteration:4     X(n): -2.500000         Tolerance: 0.3125000000
Iteration:5     X(n): -2.187500         Tolerance: 0.1562500000
Iteration:6     X(n): -2.031250         Tolerance: 0.0781250000
Iteration:7     X(n): -2.031250         Tolerance: 0.0390625000
Iteration:8     X(n): -2.031250         Tolerance: 0.0195312500
Iteration:9     X(n): -2.011719         Tolerance: 0.0097656250
Iteration:10    X(n): -2.001953         Tolerance: 0.0048828125
Iteration:11    X(n): -2.001953         Tolerance: 0.0024414063
Iteration:12    X(n): -2.001953         Tolerance: 0.0012207031
Iteration:13    X(n): -2.000732         Tolerance: 0.0006103516
Iteration:14    X(n): -2.000122         Tolerance: 0.0003051758
Iteration:15    X(n): -2.000122         Tolerance: 0.0001525879
Iteration:16    X(n): -2.000122         Tolerance: 0.0000762939
Iteration:17    X(n): -2.000046         Tolerance: 0.0000381470
Iteration:18    X(n): -2.000008         Tolerance: 0.0000190735
Iteration:19    X(n): -2.000008         Tolerance: 0.0000095367
Iteration:20    X(n): -2.000008         Tolerance: 0.0000047684
Iteration:21    X(n): -2.000003         Tolerance: 0.0000023842
Iteration:22    X(n): -2.000000         Tolerance: 0.0000011921
Final Solution: -2.000000
-----------------------------------------------------------------------
                    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 . . .
```

- Show your code for newtonRaphson()

Tip:

- Need to select MaxItr to prevent infinite loop errors
- Calculate for *f'(x)* analytically, if you can. 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'(xn)==0, f(a)*f(b)>0* etc.
- 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. $y = x^2 - x$

```
float func(_x){
return _x*_x-_x;}

float dfunc(_x){
return 2*_x-1;}
```

Advanced Problem: Bonus Point [10pt]

Q. You can combine bisection method and Newton-Raphson for a fail-safe routine. This hybrid algorithm can be designed as

- Set the bounds [a, b] on a root (x_T) as in the bisection method
- For each iteration,

  If Newton-Raphson gives the solution out of bounds, use bisection method for the next estimation of x(n+1)

  If Newton-Raphson is not decreasing fast enough, or seems to be diverging, use bisection method for x(n+1).

  Otherwise use Newton-Raphson
- Check your algorithm by solving for $f(x) = \frac{1}{x} - 2 = 0$. Use x_0=1.4 for the initial point. The exact solution is x=0.5.
- First, use Newton-Raphson method and analyze the solution. Use your hybrid method to find the solution. Compare these two solutions.