MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF MECHANICAL ENGINEERING
ME 310 NUMERICAL METHODS
FALL 2022
PROGRAMMING PROJECT 1

Assignment date : 09.11.2022
Due date : 23.11.2022

The programming project will be submitted through METU-Class, as described in the "Programming Project Assignment Guidelines", which is posted on METU-Class.

In False-position method (which is a bracketing method) the root of the function is estimated by the intersection of the line that passes through the function on two brackets. In order to have a better method, instead of a line we are going to use a second order polynomial. However, note that, in order to use a second order polynomial, 3 points are required. Therefore, in order to eliminate inputting 3 initial guesses to our root finding algorithm, the third point can be calculated as in the bisection method, i.e. $x_i = (x_l + x_u)/2$. The method is shown in the below figure.

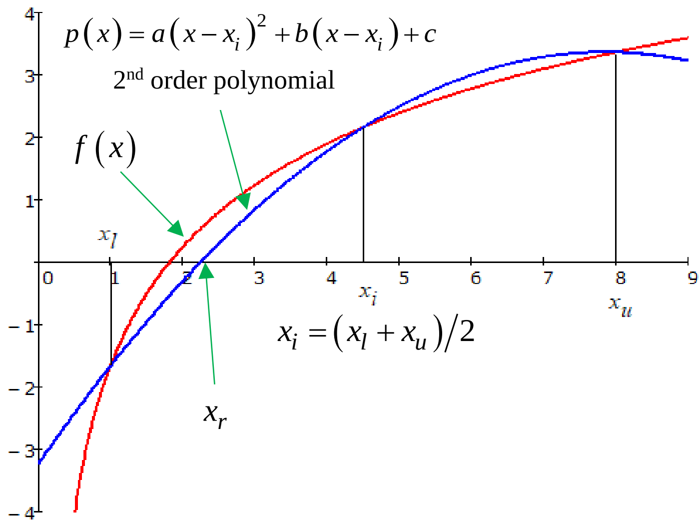

Figure 1 Third initial guess obtained as Bisection method

Second order polynomial passing through the intermediate point, x_i, can be writtens as follows

$$p(x) = a(x - x_i)^2 + b(x - x_i) + c,$$

where

$$f(x_l) = a(x_l - x_i)^2 + b(x_l - x_i) + c$$

$$f(x_u) = a(x_u - x_i)^2 + b(x_u - x_i) + c.$$

$$f(x_i) = c$$

From these equations, unknown constants of the 2nd order polynomial can be obtained as follows

‡ 123456 is the first six digits of the student number of the project group leader.

$$a=\frac{f(x_l)-f(x_i)}{(x_l-x_i)(x_l-x_u)}+\frac{f(x_i)-f(x_u)}{(x_u-x_i)(x_l-x_u)}$$
$$b=\frac{\left(f(x_l)-f(x_i)\right)(x_i-x_u)}{(x_l-x_i)(x_l-x_u)}-\frac{\left(f(x_i)-f(x_u)\right)(x_l-x_i)}{(x_u-x_i)(x_l-x_u)}.$$
$$c=f(x_i)$$

The roots of the 2nd order polynomial can be calculated as

$$x_{1,2}=x_i+\frac{-b\pm\sqrt{b^2-4ac}}{2a}=x_i-\frac{2c}{b\pm\sqrt{b^2-4ac}}.$$

From the two roots whichever is closer to x_i is selected, since x_i is the root estimate by using Bisection method. In order to minimize $|x_{1,2}-x_i|$, $\left|b\pm\sqrt{b^2-4ac}\right|$ need to be maximum; hence, the sign before square root must be equal to the sign of b. Therefore, iterative algorithm can be written as follows

$$x_r=x_i-\frac{2c}{b+sign(b)\sqrt{b^2-4ac}}.$$

The new bracket can be identified as in Bisection method by comparing the sign of $f(x_r)$ with the signs of $f(x_l)$ and $f(x_u)$. This method will be referred as Polynomial Method.

Write a computer program (named *e123456.ext*‡, where *.ext* is *.m, .py, .f90, .c, .cpp* etc. depending on your language of preference) to find the roots of a function $f(x)$ by using Polynomial method, Bisection method, False-position method, Secant method and Newton-Raphson method, separately. Function $f(x)$ and $f'(x)$ will be defined in a separate file named as *f.ext* and *fp.ext* (i.e. consistent with your program language), where the name of the function in this file should also be ***f*** and ***fp***.

Initial bracket containing the root (x_l, x_u), the pre-specified error tolerance (ε_s) in % and the maximum number of iterations ($i_{\max}$) will be given in an input file named as "*input.txt*" where each value is on a different line respectively. Once your program file (i.e. *e123456.ext*) is called / executed, it should read the file "*input.txt*" to initialize the problem parameters, and generate solutions for the 3 methods mentioned above. Results should be saved into the following files named "*output_polynomial.txt*", "*output_bisection.txt*", "*output_falseposition.txt*" "*output_secant.txt*" and "*output_newton.txt*" for each method, respectively. Each line in an output file should have $i, x_r, f(x_r), \varepsilon_a$ where i is the iteration number and x_r and ε_a are the corresponding root estimate and approximate percent error, respectively.

The convergence performance of all methods should be compared by finding the root of the equation using all methods.

A sample *input.txt* file can be as follows:

```
1.0
2.0
0.000001
30
```

This means that $x_l = 1.0$, $x_u = 2.0$, $\varepsilon_s = 0.000001\%$ and $i_{\max} = 30$. As a result, your code should stop if $\varepsilon_a < \varepsilon_s$ or after $i_{\max}$ iterations independent of what ε_a is. Hence, for this particular case, your output files should not have more than 30 lines.

A sample *output.txt* file will have the following format:

```
1  0.85000112 -1.25687925 ----
2  0.75000112 -0.57894156 11.76
```
...

According to this output file, the first iteration yields $x_{r1} = 0.85000112$, $f(x_{r1}) = -1.25687925$ and approximate relative error in percent is undefined, second iteration yields $x_{r2} = 0.75000112$ $f(x_{r2}) = -0.57894156$, $\varepsilon_{a2} = 11.76$ and so on. All values should be in floating point format.

To sum up, during grading we expect your code to perform as follows:

- To start with, *f.ext*, *e123456.ext* and *input.txt* will be in the same folder.
- Your solver implementation (i.e. *e123456.ext*) will be run in the proper environment (C, Fortran, Matlab, Python etc.)
- Making use of *f.ext* and *input.txt* your solver code will generate 5 files "*output_polynomial.txt*", "*output_bisection.txt*", "*output_falseposition.txt*" "*output_secant.txt*" and "*output_newton.txt*" as defined above and stop.
- Once execution is completed and output files are generated, a summary for each method should be printed to the computer screen: the last estimate of the root (x_{rn}), approximate percent relative error for this estimate (ε_{an}), total number of iterations (n) and the function value at the calculated root ($f(x_{rn})$).
- Your code should never crash.
- If there is an abnormality (such as problems with function *f* or *input.txt* file) your code should print a proper message to the screen and to the end of the output files and terminate.

Present your results in a short report (a few pages of a word document only, saved as a pdf document) which should include the following:

- A basic introduction paragraph,
- Necessary formulations and hand calculations to write your code (type it in the word document),
- Your numerical results for an example *f(x)* for each algorithm (i.e. Polynomial, Bisection etc.),
- Your plotted graphics including the evolution of the numerical computation process by plotting the root estimates and errors, etc. versus iteration number for each algorithm
- A single plot comparing ε_S values generated for both methods,
- Discussion of the results and conclusion,
- Appendix section including your code and executable file in the case programing languages such as C/C++, Fortran etc. are used.

Use the following functions and the given upper and lower bounds in your project report. If the method requires a single initial guess, then use the mean of the given limit.

$x^2 - (1-x)^5$	$[0.1, 1]$	|	$xe^x - 1$	$[-1, 1]$

$\cos(x) - x^3$	$[0.1, 1]$	$\vert$	$\ln(x)$	$[0.5, 5]$
x^3	$[-0.5, 1/3]$	$\vert$	$e^{x^2+7x-30} - 1$	$[2.8, 3.1]$
$x^{-1} - \sin(x) + 1$	$[-1.3, -0.5]$	$\vert$		