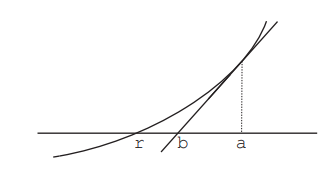
**Assignment no.**

**Problem Statement: WRITE A PROGRAM IN "c" TO DETERMINE THE ROOTS (CORRECT UPTO 5TH DECIMAL PLACES)**

**OF THE FOLLOWING EQUATION BY USING "Newton-Raphson Method"**

**f(x) = x^3 - 3\*x + 1**

**Theory:**

Let x0 be a good estimate of r and let r = x0 + h. Since the true root is r, and h = r − x0, the number h measures how far the estimate x0 is from the truth. Since h is ‘small,’ we can use the linear (tangent line) approximation to conclude that 0 = f(r) = f(x0 + h) ≈ f(x0) + hf0 (x0), and therefore, unless f0 (x0) is close to 0, h ≈ − f(x0) f0 (x0) . It follows that r = x0 + h ≈ x0 − f(x0) f0 (x0). Our new improved (?) estimate x1 of r is therefore given by x1 = x0 − f(x0) f0 (x0). The next estimate x2 is obtained from x1 in exactly the same way as x1 was obtained from x0: x2 = x1 − f(x1) f0 (x1). Continue in this way. If xn is the current estimate, then the next estimate xn+1 is given by xn+1 = xn − f(xn) f0 (xn) 

In the picture below, the curve y = f(x) meets the x-axis at r. Let a be the current estimate of r. The tangent line to y = f(x) at the point (a, f(a)) has equation y = f(a)+(x − a)f0 (a). Let b be the x-intercept of the tangent line. Then b = a − f(a) f0 (a)

Compare with Equation 1: b is just the ‘next’ Newton-Raphson estimate of r. The new estimate b is obtained by drawing the tangent line at x = a, and then sliding to the x-axis along this tangent line. Now draw the tangent line at (b, f(b)) and ride the new tangent line to the x-axis to get a new estimate c. Repeat. We can use the geometric interpretation to design functions and starting points for which the Newton Method runs into trouble. For example, by putting a little bump on the curve at x = a we can make b fly far away from r. When a Newton Method calculation is going badly, a picture can help us diagnose the problem and fix it. It would be wrong to think of the Newton Method simply in terms of tangent lines. The Newton Method is used to find complex roots of polynomials, and roots of systems of equations in several variables, where the geometry is far less clear, but linear approximation still makes sense.

**Variable Listing:**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| a, b | float | For storing intervals of a given function |
| c | float | Stores the formula of newton-raphson |
| c\_prev | float | Stores the previous value of c |
| error | float | Stores the errors of c and c\_prev |
| i | integer | Loop variable |

**Algorithm:**

1. Read a and b from the user
2. If (f(a) \* f(b) > 0) then go to next step, otherwise go to step 5
3. Display invalid interval
4. Exits from the program
5. If f(a) and f(b) both gets 0, then go to next step, otherwise go to step 8
6. Display root a whether f(a) = 0
7. Exits from the program
8. Repeat through step 9 to step 17 until error > 0.0005
9. Stores c in c\_prev
10. Store (a - (f(a) / fd(a))) in c
11. Display i, a, b, c in a tabloid form
12. Store c in a
13. Store c-c\_prev ‘s absolute value in error
14. If i gets 1, display “----“, otherwise go to next step
15. Display error

[End of do-while loop]

1. Display “The approximate root: c”
2. End.

**Source Code:**

#include "stdio.h"

#include "stdlib.h"

#include "math.h"

#define f(x) (x \* x \* x - 3 \* x + 1)

#define fd(x) (x \* x - 3)

int main()

{

float a = 0.0, b = 0.0, c\_prev, c, error = 0.0;

int i = 0;

printf("-----------------\n");

printf("Newton Raphson Method\n");

printf("-----------------\n");

printf("Enter two intervals: ");

scanf("%f %f", &a, &b);

if (f(a) == 0)

{

printf("The root is: %4.3f\n", a);

exit(1);

}

else if ((f(a) \* f(b)) > 0)

{

printf("Invalid Interval!!!\n");

exit(0);

}

printf("Iter.\ta\tc\tError\n");

do

{

c\_prev = c;

c = (a - (f(a) / fd(a)));

printf("%2d\t%4.3f\t%4.3f\t", i++, a, c);

a = c;

error = fabs(c - c\_prev);

if (i == 1)

{

printf("----\n");

}

else

{

printf("%4.3f\n", error);

}

} while (error > 0.0005);

printf("The approximate root is: %4.3f\n", c);

return 0;

}

**Input/Output:**

-----------------

Newton Raphson Method

-----------------

Enter two intervals: 0 1

Iter. a c Error

0 0.000 0.333 ----

1 0.333 0.346 0.013

2 0.346 0.347 0.001

3 0.347 0.347 0.000

The approximate root is: 0.347

**Discussion:**

1. This program doesn’t run for a very large value.
2. The program exits when user puts a wrong interval.