Program-M1

Devansh Shukla
I18PH021

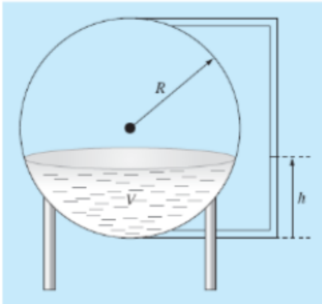
Write and execute a FORTRAN program for finding roots of an equation by Newton Raphson method, and solve the following problems both numerically and using your written program and compare them:

(a) Find the roots of the following equations correct upto two places of decimals:

$$x^3 - 3x^2 + 2x - 1 = 0 \quad \text{and} \quad -2x^2 + 3x + 2 = 0$$

(b) You are designing a spherical tank to hold water for a small village in a developing country. The volume of liquid it can hold can be computed as

$$V = \pi h^2 \frac{3R - h}{3}$$



where V=volume (m3), h=depth of water in tank (m) and R=the tank radius (m). If $R = 3\ m$, what depth must the tank be filled to so that it holds 30 m^3? Use three iterations to determine your answer.

Theory

Newton-Raphson iteration method

Newton-Raphson is a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function. It works by first guessing a inital guess, then approximating it to the actual root by using the tanget at that point.

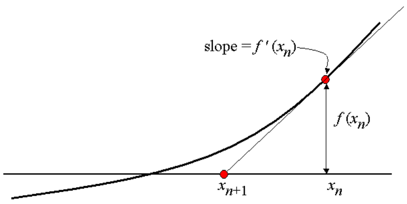


Figure 1: src:brillant.org

Let $f(x)$ be a function of x. Using Taylor expansion,

$$f(x - x_0) = f(x_0) + \frac{(x - x_0)}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \ldots \tag{1}$$

Let $h = x - x_0$ and the root be $r = x_0 + h$

$$f(r) = f(x_0) + \frac{(h)}{1!} f'(x_0) + \frac{(h)^2}{2!} f''(x_0) + \ldots \tag{2}$$

Considering r is the root and neglecting higher order terms

$$0 = f(x_0) + \frac{h}{1!} f'(x_0) \tag{3}$$

$$\implies h = -\frac{f(x_0)}{f'(x_0)}$$

$$\therefore \quad x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4}$$

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}} \tag{5}$$

Numerical Solution

(a) (i)

$$f(x) = x^3 - 3x^2 + 2x - 1 \tag{6}$$
$$f'(x) = 3x^2 - 3x + 2 \tag{7}$$

Computing initial guesses:

$$\begin{aligned} f(0) &= -1 \\ f(-1) &= -7 \\ f(2) &= -1 \\ f(3) &= 5 \end{aligned}$$

Therefore, the root is between $(2,\ 3) \implies x_0 = 2,\ f(x_0) = -1$

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= 2 - \frac{-1}{2} \\ &= 2.5 \end{aligned}$$

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\ &= 2.5 - \frac{0.875}{5.75} \\ &= 2.3478 \end{aligned}$$

$$\begin{aligned} x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} \\ &= 2.3478 - \frac{0.1001}{4.4497} \\ &= 2.3253 \end{aligned}$$

$$\begin{aligned} x_4 &= x_3 - \frac{f(x_3)}{f'(x_3)} \\ &= 2.3253 - \frac{2.4838 \times 10^{-3}}{4.2692} \\ &= 2.3247 \end{aligned}$$

At $x = 2.3247$, $f(x) = -7.6771 \times 10^{-5}$, hence $x = 2.3247$ is the a root of the equation.

(a) (ii)

$$f(x) = -2x^2 + 3x + 2 \tag{8}$$
$$f'(x) = -4x + 3 \tag{9}$$

Computing initial guesses:

$$\begin{aligned} f(0) &= 2 \\ f(-1) &= -3 \\ f(1) &= 3 \\ f(2) &= 0 \end{aligned}$$

Therefore, the root is between $(0,\ -1) \implies x_0 = 0,\ f(x_0) = 2,\ f'(x_0) = 3$

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= 0 - \frac{2}{3} \\ &= -0.6667 \end{aligned}$$

$$
\begin{aligned}
x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\
&= -0.6667 - \frac{-0.8891}{5.6668} \\
&= -0.5098
\end{aligned}
$$

$$
\begin{aligned}
x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} \\
&= -0.5098 - \frac{-4.9192 \times 10^{-2}}{5.0392} \\
&= -0.5088
\end{aligned}
$$

$$
\begin{aligned}
x_4 &= x_3 - \frac{f(x_3)}{f'(x_3)} \\
&= -0.5088 - \frac{-4.4155 \times 10^{-2}}{5.0352} \\
&= -0.5079
\end{aligned}
$$

At $x = -0.5079,\ f(x) = -3.9624 \times 10^{-2}$, hence $x = -0.5079$ and $x = 2$ are the roots of the equation.

(b)

$$V = \frac{\pi h^2\ (3R - h)}{3} \tag{10}$$

$$f(h) = h^3 - 3h^2 R + \frac{3V}{\pi} \tag{11}$$

For $R = 3\ m$ and $V = 30\ m^3$, $h = ?$

$$f(h) = h^3 - 9h^2 + 28.6479 \tag{12}$$

$$f'(h) = 3h^2 - 18h \tag{13}$$

Computing initial guesses:

$$
\begin{aligned}
f(0) &= 28.6479 \\
f(-1) &= 18.6479 \\
f(-2) &= -15.3521 \\
f(1) &= 20.6479 \\
f(2) &= 0.6479 \\
f(3) &= -25.3521 \\
f(8) &= -35.35 \\
f(9) &= 28.65
\end{aligned}
$$

Therefore, the roots are between $(2,\ 3)$ and $(8,\ 9) \implies h_0 = 2,\ f(h_0) = 0.6479,\ f'(h_0) = -24$

$$
\begin{aligned}
h_1 &= h_0 - \frac{f(h_0)}{f'(h_0)} \\
&= 2 - \frac{0.6479}{-24} \\
&= 2.0270
\end{aligned}
$$

$$
\begin{aligned}
h_2 &= h_1 - \frac{f(h_1)}{f'(h_1)} \\
&= 2.0270 - \frac{-2.2697 \times 10^{-3}}{-24.1598} \\
&= 2.0269
\end{aligned}
$$

$$
\begin{aligned}
h_3 &= h_2 - \frac{f(h_2)}{f'(h_2)} \\
&= 2.0269 - \frac{1.4687 \times 10^{-4}}{-24.1592} \\
&= 2.0269
\end{aligned}
$$

At $h = 2.0269,\ f(h) = -1.4687 \times 10^{-4}$, hence $h = 2.0269$ is a root of the equation.

Now, $h_0 = 8,\ f(h_0) = -35.35,\ f'(h_0) = 48.00$

$$\begin{aligned} h_1 &= h_0 - \frac{f(h_0)}{f'(h_0)} \\ &= 8 - \frac{-35.35}{48.00} \\ &= 8.7364 \end{aligned}$$

$$\begin{aligned} h_2 &= h_1 - \frac{f(h_1)}{f'(h_1)} \\ &= 8.7364 - \frac{8.5286}{71.7188} \\ &= 8.6175 \end{aligned}$$

$$\begin{aligned} h_3 &= h_2 - \frac{f(h_2)}{f'(h_2)} \\ &= 8.6175 - \frac{0.2530}{67.6689} \\ &= 8.6138 \end{aligned}$$

$$\begin{aligned} h_4 &= h_3 - \frac{f(h_3)}{f'(h_3)} \\ &= 8.6138 - \frac{-7.1907 \times 10^{-3}}{67.5442} \\ &= 8.6139 \end{aligned}$$

At $h = 8.6139,\ f(h) = -4.1580 \times 10^{-4}$, hence $h = 8.6139$ is a root of the equation.

Now, $h_0 = -1.0,\ f(h_0) = 18.6479,\ f'(h_0) = 21.0000$

$$\begin{aligned} h_1 &= h_0 - \frac{f(h_0)}{f'(h_0)} \\ &= -1.0 - \frac{18.6479}{21.00} \\ &= -1.8880 \end{aligned}$$

$$\begin{aligned} h_2 &= h_1 - \frac{f(h_1)}{f'(h_1)} \\ &= -1.8880 - \frac{-10.1628}{44.6776} \\ &= -1.6605 \end{aligned}$$

$$\begin{aligned} h_3 &= h_2 - \frac{f(h_2)}{f'(h_2)} \\ &= -1.6605 - \frac{-0.7459}{38.1608} \\ &= -1.6410 \end{aligned}$$

$$\begin{aligned} h_4 &= h_3 - \frac{f(h_3)}{f'(h_3)} \\ &= -1.6410 - \frac{-7.0475 \times 10^{-3}}{37.6166} \\ &= -1.6408 \end{aligned}$$

At $h = -1.6408,\ f(h) = -4.7493 \times 10^{-4}$, hence $h = -1.6408$ is a root of the equation.

All roots are: $h = -1.6408$, $h = 2.0269$ and 8.6139 and 2nd and 3rd are possible solution for the size of the tank.

Program Algorithm

NOTE: Blue-colored text represents variables in the algorithm, eg. TEST.

1. Program open.
2. Define j, m, n, big, iterations, x, i, h, f, df, f_value, old_f_value, initial_counter, final_counter, guess_increment, tolerance and arrays guesses and roots.
3. Define the function, f(x) and its derivative f'(x).
4. Define a goto loop which runs index i from initial_counter to final_counter and checks if the adjacent function value changes its sign, if yes then it saves that particular arg to the array named guesses; the index increments with guess_increment.
5. Print the inital guesses from guesses.
6. Define a do-loop which runs index big for the no. of initial guesses.
7. Define another nested inner-do-loop which runs index j from 0 to variable iterations.
8. Inside the nested loop, the working formula for Newton-Raphson is used to compute the next approximate guess; it computes h : $h = \frac{f(x_n)}{f'(x_n)}$ and root x : $x = x - h$.
9. When the required tolerance is reached or abs(root − prev_root) < tolerance the root is saved in array named roots and nested inner-do-loop breaks.
10. The steps 7 to 9 are repeated for each initial guess.
11. Outer-do-loop breaks.
12. Flag m saves the index for root; if m not equal to 1, then all roots are printed using a do-loop with index big from 1 to m-1, else "Not convergent" is printed.
13. Program close.

Program

```fortran
program newton_raphson
    ! Program to compute approximate roots of polynomials using Newton-Raphson method of iterations.
    ! Author: Devansh Shukla

    implicit none
    integer :: j, m=1, n=1, big=0, iterations=50
    real :: x, i, h, f, df, f_value, old_f_value, initial_counter=-100, final_counter=100, guess_increment=0.5,
         tolerance=0.001
    real, dimension(10) :: guesses, roots

    ! M1 (a) (i)
    ! f(x) = x**3 - 3*x**2 + 2*x - 1
    ! df(x) = 3*x**2 - 6*x + 2

    ! M1 (a) (ii)
    ! f(x) = -2*x**2 + 3*x + 2
    ! df(x) = -4*x + 3

    ! M1 (b)
    ! f(x) = x**3 - 9*x**2 + 28.6479
    ! df(x) = 3*x**2 - 18*x

    ! Computing initial guess(es)
    print *,"----------------------------------------------------"
    print *, "Computing inital guesses"
    i = initial_counter
    old_f_value = f(i)
10 i = i + guess_increment
    f_value = f(i)
    if (f_value <= 0 .and. old_f_value >= 0) then
        guesses(n) = i
        n = n + 1
        print "(xxxx, F10.4, x, F10.6, x, F10.6)", i, f_value, old_f_value
    elseif (f_value >= 0 .and. old_f_value <= 0) then
        guesses(n) = i
        n = n + 1
        print "(xxxx, F10.4, x, F10.6, x, F10.6)", i, f_value, old_f_value
    endif
    if (i<final_counter) then
        old_f_value = f_value
        GOTO 10
    endif

    print *, "Initial guess ranges are:"
```

```fortran
    do big=1, n-1, 1
        print "(xxxx,I1,A,F10.4,A,F10.4,A)", big, ". (", guesses(big), ",", guesses(big)-guess_increment, ")"
    enddo

    ! Newton-raphson method
    print *,"----------------------------------------------------"
    print "(A)", "Computing root(s) via Newton-Raphson method"
    do big=1, n-1, 1
        x = guesses(big)
        print "(x, A10,F10.4)", "Tying x = ", x
        do j=0,iterations,1
            f_value = f(x)
            h = f_value/df(x)
            old_f_value = x
            x = x - h
            print "(xxxx,A6,I1,xxx,A3,F10.6,xxx,A3,F10.6)", "itr = ", j, "x =", x, "f =", f(x)
            if (abs(x-old_f_value) < tolerance) then
                roots(m) = x
                m = m + 1
                print "(A17, F10.6)", "Converged at x =", x
                exit
            endif
        enddo
    enddo
    print *,"----------------------------------------------------"
    if (m .ne. 1) then
        print *, "Roots are:"
        do big=1, m-1, 1
            print "(xxxx,A4,F10.6)", "x = ", roots(big)
        enddo
    else
        print *, "Not convergent"
    endif
    print *,"----------------------------------------------------"

end program newton_raphson
```

Results

(a) (i)

```
 ----------------------------------------------------
 Computing inital guesses
        2.5000   0.875000  -1.000000
 Initial guess ranges are:
    1. (    2.5000,    2.0000)
 ----------------------------------------------------
Computing root(s) via Newton-Raphson method
 Tying x =     2.5000
    itr = 0   x =  2.347826   f =  0.100681
    itr = 1   x =  2.325201   f =  0.002059
    itr = 2   x =  2.324718   f =  0.000002
 Converged at x =  2.324718
 ----------------------------------------------------
 Roots are:
    x =   2.324718
 ----------------------------------------------------
```

(a) (ii)

```
 ----------------------------------------------------
 Computing inital guesses
       -0.5000   0.000000  -3.000000
        0.0000   2.000000   0.000000
        2.0000   0.000000   2.000000
        2.5000  -3.000000   0.000000
 Initial guess ranges are:
    1. (   -0.5000,   -1.0000)
    2. (    0.0000,   -0.5000)
    3. (    2.0000,    1.5000)
    4. (    2.5000,    2.0000)
 ----------------------------------------------------
Computing root(s) via Newton-Raphson method
 Tying x =    -0.5000
    itr = 0   x = -0.500000   f =  0.000000
 Converged at x = -0.500000
 Tying x =     0.0000
    itr = 0   x = -0.666667   f = -0.888889
    itr = 1   x = -0.509804   f = -0.049212
```

```
   itr = 2   x = -0.500038   f = -0.000191
   itr = 3   x = -0.500000   f =  0.000000
Converged at x = -0.500000
Tying x =     2.0000
   itr = 0   x =  2.000000   f =  0.000000
Converged at x =  2.000000
Tying x =     2.5000
   itr = 0   x =  2.071429   f = -0.367347
   itr = 1   x =  2.001930   f = -0.009660
   itr = 2   x =  2.000001   f = -0.000007
   itr = 3   x =  2.000000   f =  0.000000
Converged at x =  2.000000
----------------------------------------------------
Roots are:
   x =  -0.500000
   x =  -0.500000
   x =   2.000000
   x =   2.000000
----------------------------------------------------
```

(b)

```
----------------------------------------------------
Computing inital guesses
      -1.5000   5.022900 -15.352100
       2.5000 -11.977100   0.647900
       9.0000  28.647900  -7.477100
Initial guess ranges are:
   1. (   -1.5000,   -2.0000)
   2. (    2.5000,    2.0000)
   3. (    9.0000,    8.5000)
----------------------------------------------------
Computing root(s) via Newton-Raphson method
 Tying x =    -1.5000
   itr = 0   x = -1.648827   f = -0.302311
   itr = 1   x = -1.640836   f = -0.000893
   itr = 2   x = -1.640813   f =  0.000002
Converged at x = -1.640813
Tying x =     2.5000
   itr = 0   x =  2.043730   f = -0.407261
   itr = 1   x =  2.026940   f = -0.000816
   itr = 2   x =  2.026906   f =  0.000004
Converged at x =  2.026906
Tying x =     9.0000
   itr = 0   x =  8.646322   f =  2.207348
   itr = 1   x =  8.614165   f =  0.017467
   itr = 2   x =  8.613907   f =  0.000011
Converged at x =  8.613907
----------------------------------------------------
Roots are:
   x =  -1.640813
   x =   2.026906
   x =   8.613907
----------------------------------------------------
```

Remarks

The program can be used to compute roots of real-valued functions. It automatically guesses the initial root but is currently limited to search from -100 to 100, though it could be extended to search in more wider range by setting the inital_counter and final_counter.

The roots computed numerically and using the program are in agreement.