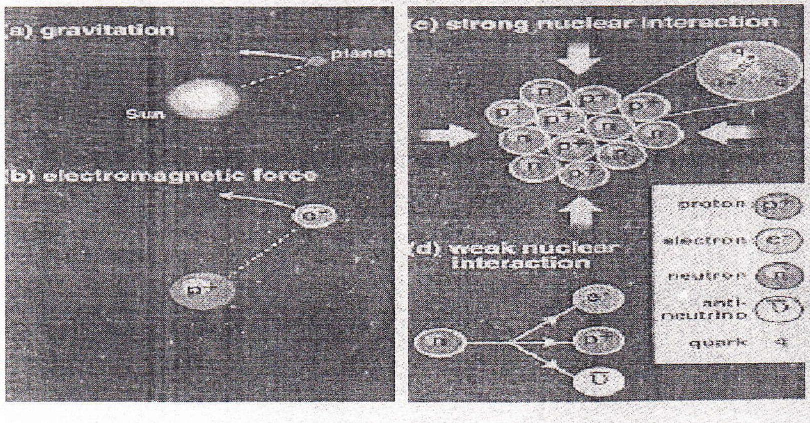
BULLETIN

THE INDIAN ASSOCIATION OF PHYSICS TEACHERS

VOLUME 25 NUMBER 9 NOVEMBER 2008

Fundametal forces of nature


(a) gravitation
planet
Sun
(b) electromagnetic force
(c) strong nuclear interaction
(d) weak nuclear interaction
proton
electron
neutron
anti-neutrino
quark

A MONTHLY JOURNAL OF EDUCATION IN

COMPUTATION OF ROOTS OF AN ALGEBRAIC EQUATION BY NEWTON-RAPSON'S METHOD - AN INTERACTIVE APPROACH

S W ANWANE
Department of Physics, Shri Shivaji Education Society Amravati's, Science College, Congress Nagar, Nagpur

roduction

The solution of numerical problems in theoretical physics was, until a few years ago, the domain of large computers. In recent years, personal computers have reached the power at par with large computers of early sixities. Apart from their high computational performances, personal computers offer interactive capabilities and rapid graphical output of results. Thus, the personal computers offer us a wide field of possibilities of education and research. The present work is an attempt to visualize the Numerical Method of Newton-Rapson and attempt calculations in steps until we find the roots of the equations. The C^{++} programme depicted in the Appendix will help the users to take a ride on their PC to find the roots of equation.

Computation using Newton-Rapson's method- An approach to practical problem

One of the oldest problems in Mathematics is that of finding the roots of an equation, which has given rise to a whole branch of mathematics known as *theory of equations.* It deals with those methods which are applicable to finding the real roots of equation

$$f(x)=0 \qquad (1)$$

where $f(x)$ is any precise piecewise continuous function of x having numerical cofficients a_i's giving a polynomial of degree n

$$y=f(x)=a_0+a_1x+a_2x^2+a_3x^3+\ldots+a_mx^m=\sum_{i=0}^{m}a_ix^i \qquad (2)$$

In order to have concise discussion we shall begin with a polynomial of degree 2 having its roots known and then search them with famous Newto-Rapson's method to find the roots. Later, we shall generalize the problem for a polynomial of degree 2 and write a programme in Turbo C++ which then will be a tool between Polynomial and Roots.

Graphical Approach

Consider a function

$$y=f(x)=(x-4)(x-9)=x^2-13x+36$$

Obviously roots are 4 and 9 while a polynomial to be solved for roots has coefficients

$$a_0=36, a_1=-13 \text{ and } a_2=1.$$

If the referred polynomial is plotted, it appears as shown in the Fig. 1. The curve intersects x-axis twice indicating existance of two real roots.

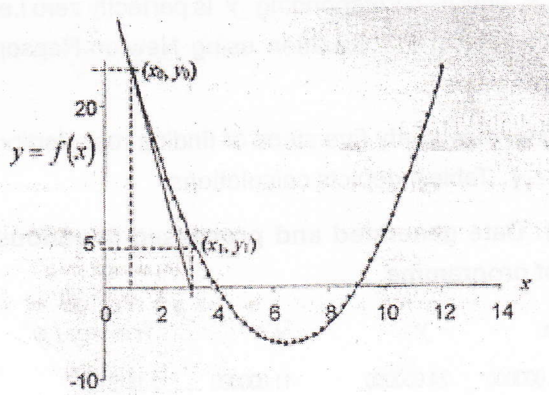


Fig.1: Interpretation of Newton-Rapson's formula in terms of plots of $f(x)=0$

To begin search of roots we have to begin from arbitrary point (x_0, y_0). Draw a tangent to the curve at (x_0, y_0). Mathematically, the slop of the tangent will be

$y' = f'(x) = 2x - 13$ i.e. $y_0' = f'(x_0) = 2x_0 - 13$ (4)

This line intercepts x-axis at a point x_1 (say) which is closer to the root than x_0. Thus we know that tangent is a line

$$y = mx + c \quad (5)$$

passing through points (x_0, y_0), $(x_1, 0)$ and and having slope $m = y_0'$.

To know *y*-intercept *c* of the line, substituting (x_0, y_0) in (5), we get

$$c = y_0 - y'_0\, x_0 \quad (6)$$

For this point $(x_1, 0)$ the equation of line will be $0 = mx_1 + c$. Thus the new point

$$x_1 = -\frac{c}{m} = \frac{y_0 - y'_0\, x_0}{y'_0}.$$

$$x_1 = x_0 - \frac{y_0}{y'_0} = x_0 + h \quad \text{for} \quad \frac{-y_0}{y'_0} = h \text{ (say)} \quad (6a)$$

It may be generalized as

$$x_{n+1} = x_n = \frac{y_n}{y'_n} = x_n + h \quad (6b)$$

Now, we can find a point y_1 on the curve corresponding to x_1 as shown in Fig. 1. Mathematically obtain y_1 from (2). Now the process can be repeated for (x_1, y_1) to search x_1 until its corresponding y is perfectly zero i.e. we reach root of the equation using Newton-Rapson formula (6).

The reader may verify five steps of finding root starting from $x = 1$. Table 1 depicts calculations.

Table 1: Date generated and processed in various steps of programme

n	x	y	y'	$h = -y/y'$
0	1.000000	24.000000	-11.000000	2.181818
1	3.181818	4.760331	-6.636340	0.717310
2	3.899128	0.514534	-5.201743	0.098916
3	3.998044	0.009784	-5.003912	0.001955
4	3.999999	0.000004	-5.000002	0.000001
5	4.000000	0.000000	-5.000000	0.000000

The programme proposed in the present work repeatedly uses Newton-Rapson formula (6). Moreover a generalization is developed to key in the polynomial equation of the form (3) by accepting coefficients $a_0, a_1, a_2, a_3, \ldots a_m$. By arbitrarily supplying the starting point x_0 one can reach to the exact root of the polynomial provided it is real and does exist.

Mathematical approach using Taylor's series

The Newton-Rapson's mathod is based on the process of successive approximation by iteration. The formula for successive aproximation may be obtained from Taylor's Series with a remainder about the point $x = x_0$.

$$f(x) = f(x_i) + (x - x_i) f'(x_i) + \frac{(x - x_i)^2}{2!} f''(\xi) \quad (7)$$

ξ lies between x and x_i. Sustituting $x = p$ (where p is root) and solving for p in the second term on the right-hand side, we have

$$p = x_i - \frac{f(x)}{f'(x)} - (p - x_i)^2 \frac{f''(\xi)}{2 f'(x_i)} \quad (8)$$

therefore, by equation (6b)

$$x_{i+1} - p = (x_i - p)^2 \frac{f''(\xi)}{2 f'(x_i)} \quad (9)$$

since $x_i - p$ is the error in the j th iterate to the roo' equation (8) states that each iteration squares the error

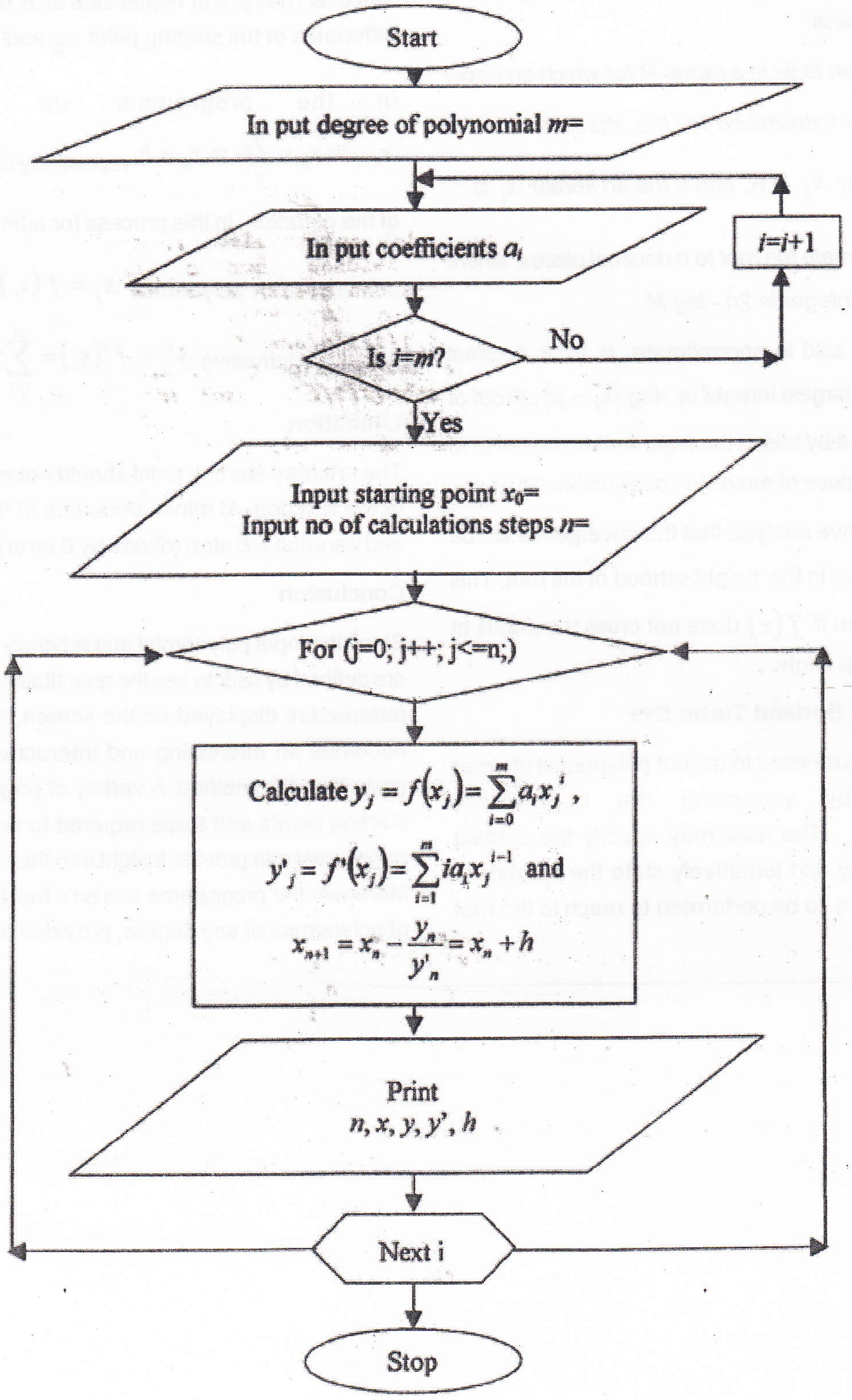
and then multiplie it by the factor $\frac{f''(\xi)}{2 f'(x_i)}$, where ξ lies

somewhere between x_i and the root p.

Note that the sign of the error in x_{i+1} is independent of the sign of the error in x_i. It is positive if $f''(x_i) f'(x_i) > 0$ in the neighborhood of p and negative if $f''(x_i) f'(x_i) < 0$. Thus, the iterate approach the root from one particular side that is independent of the initial interate x_0.

Once one is close enough to the root so that the factor $\frac{f''}{2f'}$ in equation (9) is roughly constant from one iteration

Flow Chart


Start
In put degree of polynomial m=
In put coefficients a_i
$i=i+1$
Is $i=m$?
No
Yes
Input starting point x_0=
Input no of calculations steps n=
For (j=0; j++; j<=n;)
Calculate $y_j = f(x_j) = \sum_{i=0}^{m} a_i x_j^{\,i}$,
$y'_j = f'(x_j) = \sum_{i=1}^{m} i a_i x_j^{\,i-1}$ and
$x_{n+1} = x_n - \frac{y_n}{y'_n} = x_n + h$
Print
n, x, y, y', h
Next i
Stop

to the next, one may successfully employ the following rule based on equation (9), for the rate of convergence of the iteration process:

If the root is known to lie in a range R for which an upper bound M can be established for the absolute value of $\frac{f''(\xi)}{2f'(x_i)}$, *for any* x_j *in R, and if the ith iterate* x_i *is known to approximate the root to n decimal places, where m is the longest integer in 2n - log M.*

In this *rule* x_i is said to approximate p to n decimal places if n is the largest integer in -log $|x_i - p|$. Proof of the rule is obtained by taking the logarithm to the base 10 of the absolute values of each side of equation (9).

It is clear from above analysis that the covergence will be poor if $\frac{f''}{2f'}$ is large in the neighborhood of the root. This will usually happen if $f(x)$ does not cross the x-axis at a sufficiently steep angle.

Programming in Borland Turbo C++

The programme is devised to accept polynomial of order m followed by accepting the coefficients $a_0, a_1, a_2, a_3, ... a_m$. The user may specify the starting point x_0 arbitrarily and tentatively state the number of calculation steps n to be performed to reach to the root of equation. The number of step n is also to be chosen arbitrarily such that root finding attepts should be sufficient. The user may prefer higher side of n depending upon the difference of the starting point x_0 and root x_n.

In the programme we use equation $x_{n+1} = x_n - \frac{y_n}{y'_n} = x_n + h$ repeatedly until we reach to root of the equation. In this process for arbitrary x_j, y_j and be obtained from polynomial $x_j = f(x_i) = \sum_{i=0}^{m} a_i x_j^{\,i}$ and its first order derivative $y'_j = f'(x_i) = \sum_{i=1}^{m} i a_i x_j^{\,i-1}$.

Limitation

The arbitrary starting point should not be minima/maxima of the function. At minima/maxima of the function $y' = 0$ and iteration will stop (divide by 0 error).

Conclusion

Since the input polynomial and arbitrary starting conditions are defined by user to see the quantitative effects on various parameters displayed on the screen, this C programme becomes an interesting and interactive teaching aid to assimilate the method. A variety of polynomials, arbitrary starting points and steps required to find out roots of the polynomial can provide insight into the Numerical Method. Moreover the programme can be a tool to find the real roo of polynomial of any degree, provided root exists.