TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

Dharan-8

A Lab Report On: To Find A Root Of Non-Linear Using Bisection Method

Submitted By

Name: Dhiraj KC
Roll No. : PUR077BEI014
Faculty: Electronics, Communication and Information
Group: A

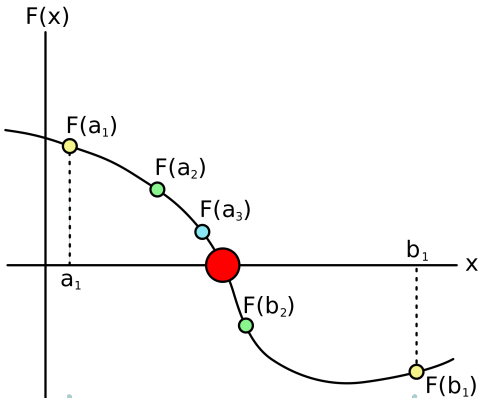
Submitted To

Department Of Electronics and Computer Engineering
Checked By: Pukar Karki

TITLE: TO FIND A ROOT OF NON-LINEAR USING BISECTION METHOD

THEORY

Bisection Method is one of the simplest, reliable, easy to implement and convergence guaranteed method for finding real root of non-linear equations. It is also known as Binary Search or Half Interval or Bolzano Method.



Bisection method is bracketing method and starts with two initial guesses say x_0 and x_1 such that x_0 and x_1 brackets the root i.e. $f(x_0)f(x_1) < 0$ Bisection method is based on the fact that if $f(x)$ is real and continuous function, and for two initial guesses x_0 and x_1 brackets the root such that: $f(x_0)f(x_1) < 0$ then there exists at least one root between x_0 and x_1. Root is obtained in Bisection method by successive halving the interval i.e. If x_0 and x_1 are two guesses then we compute new approximated root as:

$$x_2 = \frac{(x_0 + x_1)}{2}$$

Now we have following three different cases:

1. If $f(x_2) = 0$ then the root is x_2.
2. If $f(x_0)f(x_2) < 0$ then root lies between x_0 and x_2.
3. If $f(x_0)f(x_2) > 0$ then root lies between x_1 and x_2. And then process is repeated until we find the root within desired accuracy.

ALGORITHM

1. Start
2. Define function $f(x)$
3. Choose initial guesses x_0 and x_1 such that $f(x_0)f(x_1) < 0$
4. Choose pre-specified tolerable error e.
5. Calculate new approximated root as $x_2 = \frac{(x_0 + x_1)}{2}$
6. Calculate $f(x_0)f(x_2)$
    1. If $f(x_0)f(x_2) < 0$ then $x_0 = x_0$ and $x_1 = x_2$
    2. If $f(x_0)f(x_2) > 0$ then $x_0 = x_2$ and $x_1 = x_1$
    3. If $f(x_0)f(x_2) = 0$ then goto (8)
7. If $|f(x_2)| > e$ then goto (5) otherwise goto (8)
8. Display x_2 as root.
9. Stop

PROGRAM

```
import math

def bisectionFunc(x):
 return pow(x,3)-4*x+27

def calculateBisection(a, b, err):
 step = 1
 x0 = a
 x1 = b
 x2 = 0
 flag = True
 f0 = bisectionFunc(a)
 f1 = bisectionFunc(b)
 if (f0*f1 > 0.0):
  print("Incorrect Guesses and the solution doesnot lie between this")
 while flag:
  x2 = (x0+x1)/2
  f2 = bisectionFunc(x2)
  print('Iteration - %d, x2 = %0.8f and f(x2) = %0.8f' % (step, x2, f2))
  # means If  the solution converges
  if (f0*f2 < 0):
   x1 = x2
  else:
   x0 = x2
  step += 1
  flag = abs(f2) > err
 print("Root is %0.8f", x2)

calculateBisection(2, 3, 0.0001)
```

OUTPUT

```
Iteration - 1, x2 = 3.50000000 and f(x2) = 1.87500000
Iteration - 2, x2 = 3.25000000 and f(x2) = -5.67187500
Iteration - 3, x2 = 3.37500000 and f(x2) = -2.05664062
Iteration - 4, x2 = 3.43750000 and f(x2) = -0.13110352
Iteration - 5, x2 = 3.46875000 and f(x2) = 0.86178589
Iteration - 6, x2 = 3.45312500 and f(x2) = 0.36281204
Iteration - 7, x2 = 3.44531250 and f(x2) = 0.11522341
Iteration - 8, x2 = 3.44140625 and f(x2) = -0.00809759
Iteration - 9, x2 = 3.44335938 and f(x2) = 0.05352350
Iteration - 10, x2 = 3.44238281 and f(x2) = 0.02270311
Iteration - 11, x2 = 3.44189453 and f(x2) = 0.00730030
Iteration - 12, x2 = 3.44165039 and f(x2) = -0.00039926
Iteration - 13, x2 = 3.44177246 and f(x2) = 0.00345036
Iteration - 14, x2 = 3.44171143 and f(x2) = 0.00152551
Iteration - 15, x2 = 3.44168091 and f(x2) = 0.00056312
Iteration - 16, x2 = 3.44166565 and f(x2) = 0.00008193
Root is %0.8f 3.4416656494140625
```

DISCUSSION

From this practical, it can be concluded that the bisection method is an iteration method with faster rate of getting roots of equation and we found out it to be 3.44166.