Bisection Method

©Malay K. Das, mkdas@iitk.ac.in

1 Introduction

In this lecture, we focus on the numerical solution of equation involving one unknown. The general form of such equation is written as

$$f(x) = 0 \tag{1}$$

The above equation may have many real and complex roots. For brevity, however, here we focus only on finding single real root using two methods: bisection method and Newton-Raphson method. Being iterative in nature, both methods starts from arbitrary initial guess and likely to converge to a solution that is close to the initial guess. Let us assume $x = x_1$ to be a solution of the equation 1. We can test our assumption very easily by evaluating $f(x = x_1)$. If we find $f(x = x_1) \approx 0$, we have just solved the problem. Unfortunately, one has be very lucky to find the solution just from the first guess. In general, we are likely to find $f(x = x_1) \neq 0$.

2 Bisection Method

In bisection method our initial goal is to find x_1 and x_2 such that The steps involved, in the Bisection method, are described below

1. Take two trial vales x_1 and x_2 such that $f(x = x_1) > 0$ and $f(x = x_2) < 0$
2. Set $x_0 = \frac{x_1+x_2}{2}$
3. Calculate $f(x = x_0)$
4. If $f(x = x_0) > 0$ set $x_1 = x_0$, else set Calculate $x_2 = x_0$
5. Go to step 2 and continue till $f(x = x_1)$ or $f(x = x_2)$ or $f(x = x_0)$ becomes small enough to be acceptable

The algorithm (pseudocode) of bisection method may be written in various ways. One example is given below

Algorithm 1: Bisection Method

Data: $f(x), x_1, x_2$ where $f(x = x_1) f(x = x_2) < 0$, $maximumError$
Result: x_0 where $f(x = x_0) < maximumEerror$
$y_1 = f(x = x_1)$
$y_2 = f(x = x_2)$
if $(y_1 * y_2) < 0$ **then**
  $error = \min(|y_1|, |y_2|)$
  while $error > maximumError$ **do**
    $x0 = (x1 + x2)/2; y1, y0 = y(x1), y(x0)$
    $error = \min(|y_1|, |y_0|)$
    if $error < maximumError$ **then**
      if $|y1| < maximumError$ **then**
        $x0 = x1$
      end
    else if $(y1 * y0) > 0$ **then**
      $x1 = x0$
    else
      $x2 = x0$
      print$(error, x0, x1, x2)$
    end
  end
else
  print$(y1, y2)$;print("invalid input")
end

To see an example, let us try to evaluate $\sqrt{2}$ using the bisection method. One way to find the same would be to solve the following equation and pick the positive root

$$f(x) = x^2 - 2 = 0 \tag{2}$$

We know that the solutions of the above equation must satisfy $1 < x < 2$ since $f(1) < 0$ and $f(2) > 0$. With this initial guess, we can now follow the above algorithm. If we run the computer program, shown below, we will find that, as the iterations progress, the values of both x_1 and x_2 approaches the value 1.41421356 while both $f(x_1)$ and $f(x_1)$ approach 10^{-9}. The bisection method therefore intends to generate two sequences of x_1 and x_2 such that the $|x_1 - x_2|$ decreases with iteration. It is trivial to show that after n-th iteration

$$\left|x_1^{(n)} - x_2^{(n)}\right| = \frac{1}{2}\left|x_1^{(n-1)} - x_2^{(n-1)}\right| = \cdots = \frac{1}{2^n}\left|x_1^{(0)} - x_2^{(0)}\right| \tag{3}$$

Thus we see that the bisection method *converges* to accurate solution in finite number of iterations. The method, however, can be quite slow and thus may not be usable in some cases.

Computer Program 1: Computer Program for Bisection Method

```
import math as m

def main():
    y=lambda x: x**2-2    #solving x**2-2=0
    maximum_error=1.e-08
    x1,x2=1,2                  #initial guess
    y1,y2=y(x1),y(x2)
    if (y1*y2)<0:
        error=min(m.fabs(y1),m.fabs(y2))
        while error>maximum_error:
            x0=(x1+x2)/2;y1,y0=y(x1),y(x0);error=min(m.fabs(y1),m.fabs(y0
   ))
            if error<maximum_error:
                if m.fabs(y1)<maximum_error: x0=x1
            elif (y1*y0)>0: x1=x0
            else: x2=x0
            print(error,x0,x1,x2)
    else:
        print(y1,y2);print("invalid input")
    return

if __name__=="__main__":
    main()
```