- Numerical Methods

Functions and Roots

- Roots are the possible solutions to a given function.

- There are many ways to find a solution(or a root) of a function by analytical ways.

Finding the Roots of a function!!!

- We try to find the root of a function by discussing of the oldest numerical problem: finding the x value for which a given function f(x) is zero.
- For low order polynomials like; f(x)=x-3, we set f(x)=0 i.e.

$$x-3=0$$

  For which the solution is x=3.
- We can find closed form solutions for quadratic, cubic, and quartic equations as well but for polynomials of fifth order and higher, there is no general solutions.
- Many equations involving functions other than Polynomials have no analytical solutions at all.
- We try to find the solution(or root) of non linear and transcendental equations.

Bisection Method

Theory: If f(x) is continuous in the interval [a, b] and f(a) f(b)$<$ 0 then a root exist in the interval (a, b).

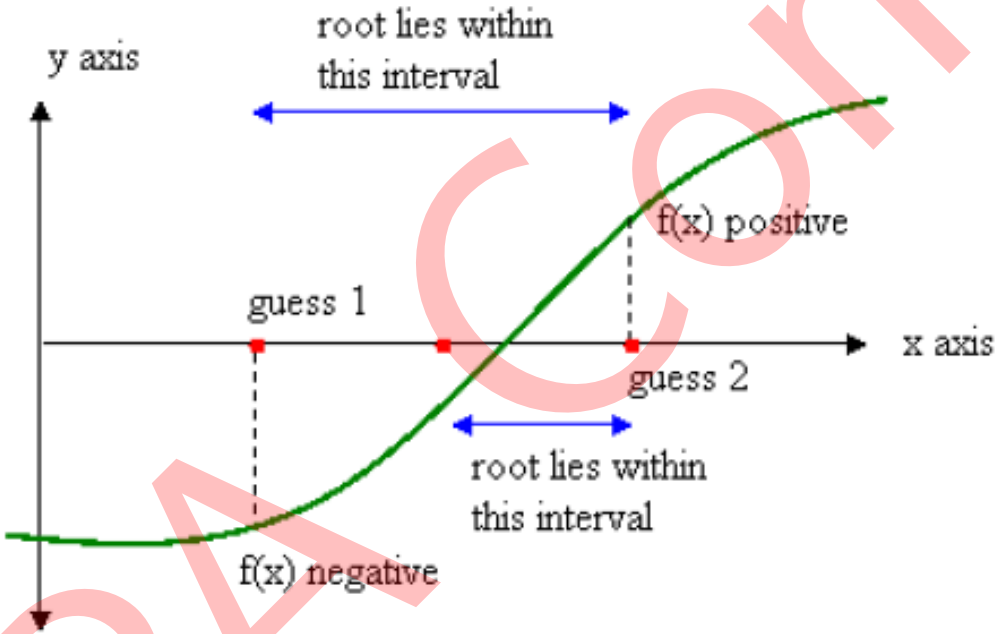


Figure: Bisection method for finding root

Question: Find the root of

$$x^2 + x - 4 = 0 \tag{1}$$

If x=1, f(x) = -ve,
If x=2, f(x) = +ve,
Root lies between 1 and 2.

n	a_n	b_n	$x_{n+1} = (a_n + b_n)/2$	$f(x_{n+1})$
0	1	2	1.5	-0.25 (-ve)
1	1.5	2	1.75	0.81 (+ve)
2	1.5	1.75	1.625	0.26 (decreasing)
3	1.5	1.625	1.5625	0.00390625
4	**1.5**	**1.5625**	**1.53125**	**-0.1240234375**

So on.....

Bisection Method

- Let us consider a transcendental equation

$$x = \cos x \tag{2}$$

- We can solve this equation by using graphical method. Rewriting above equation as,

$$f(x) = \cos x - x \tag{3}$$

- and we set f(x)=0 which gives,

$$\cos x - x = 0 \tag{4}$$

Now we draw graph between cos(x) and x. Simply the transaction point of these two lines gives the solution of the above equation.

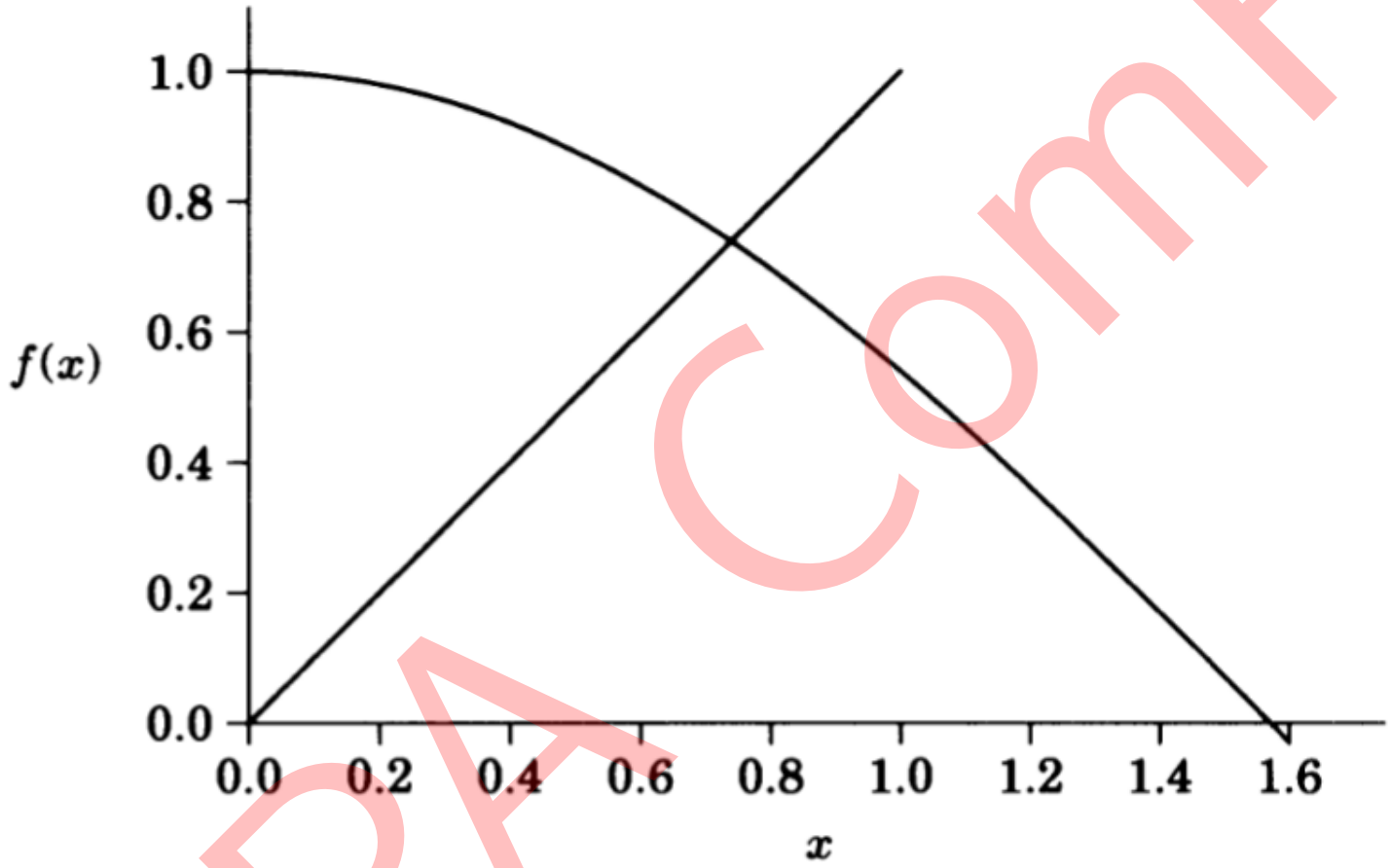


Figure: The functions cos(x) and x

Bisection Method

Solution lies near x=0.75 as in figure. But we can't find exact root of this equation. Moreover, if there are large no. of roots to find and the function is not easy to plot then this graphical method is not suitable. Thus the best known idea to solve such transcendental and non linear equations is by a numerical method: Bisection Method. The main idea of this Bisection Method are as follows:

- Choose two extrema value of x such that the root $x = x_0$ lies inside these two values. Let these values be left and right.

$$x_0 \in [\, left, right] \tag{5}$$

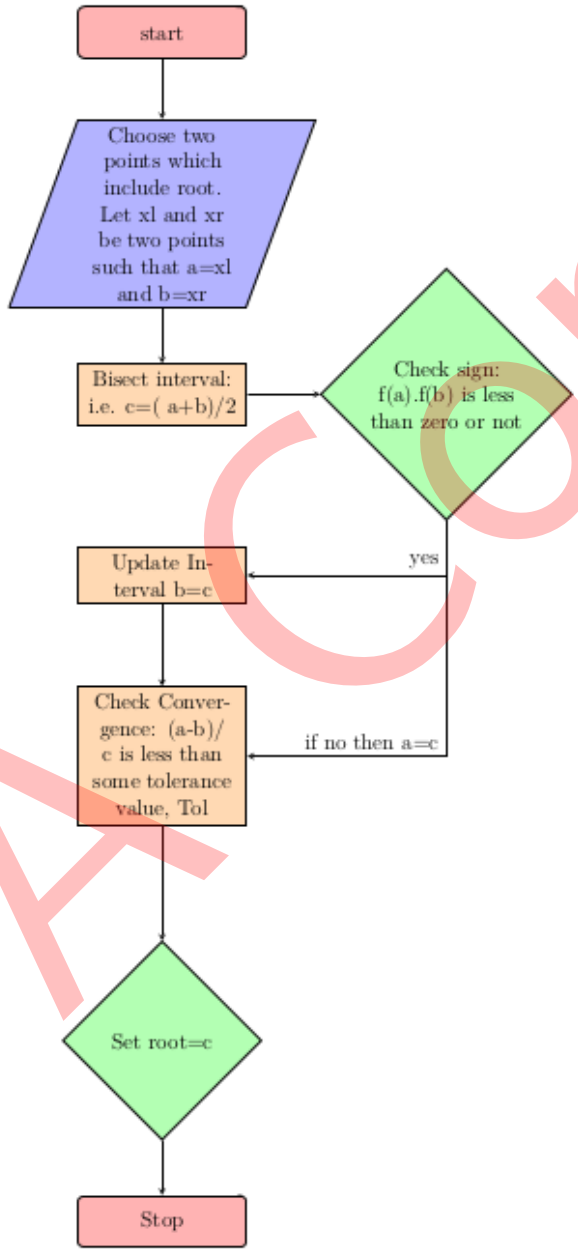
- Find f(left) and f(right) and Check whether f(left)*f(right) is less than zero or not. If the product is less than zero than it is clear that the limit we have taken covers the root value.

- Now find mean of the left and right. Let it be m_1,

$$m_1 = \frac{(left + right)}{2} \tag{6}$$

- Now check whether the root lies between (left,m_1) or (m_1,right). If f(left)*f(m_1) is less than zero then root lies between (left and m_1) otherwise root lies between (m_1 and right).
- Continue the third and fourth steps until we get closer value of x_0(i.e x_0 is less than some tolerance value).

We follow these steps to write Fortran codes and find the root. The flow chart for Bisection method is as shown below:



- program bisection1
- implicit none
- real:: a, b, c, fa, fb, fc, fx, x,tol

! a and b are also left and right values
!c is the middle point of a and b
! fa is the f(a),fb is f(b) and fc is f(c)

- external fx
- integer::n
- tol=0.001
- a=0.0
- b=1.60

!fx=f(x) !since f(x) is an external variable type so we have to define external variable !which we define after the program ends.

- n=0

!here giving n is to find after how many times of iteration we got the result

- 111 c=(a+b)/2.
- fa=fx(a)
- fb=fx(b)
- fc=fx(c)

- if(fa*fc<0.) then

b=c fb=fc

- else

a=c fa=fc

- endif
- if ((abs(a-b)/c)<tol) then

!here we set a tolerance value to give nearest value of root

- print*,"the root is",c,n
- else

n=n+1

- goto 111
- endif
- end program

!since cos(x)-x is an external function so !it should be defined after the program ends !or before the program starts

- real function fx(x)
- real::x
- fx=cos(x)-x
- end function fx

When we run this program then the output obtained is as shown below:
In lectures
The root of the transcendental equation: x=cos(x) is found to be 0.739553197 using the above programming, which is more significant than we obtained using the graphical method.

Newton-Raphson Method

- Widely used method.
- Newton-Raphson Newton's method is a fast algorithm for finding better approximations to the zeros (roots) of a real-valued function.
- It depends on initial guess which is close to true root
- This method is effective as long as the function does not have a minimum near its root.

Newton-Raphson Method

Newton-Raphson Method is based on the Taylor series expansion of a function f(x) which is given by,

$$f(x) = f(a) + (x-a)\frac{df}{dx}|_{x=a} + \frac{1}{2!}(x-a)^2\frac{d^2f}{dx^2}|_{x=a} + ... \quad (7)$$

This is the Taylor series expansion of function f(x) about a point 'a'. Now taking upto first order differentiation of f(x), we get

$$f(x) = f(a) + (x-a)\frac{df}{dx}|_{x=a} \quad (8)$$

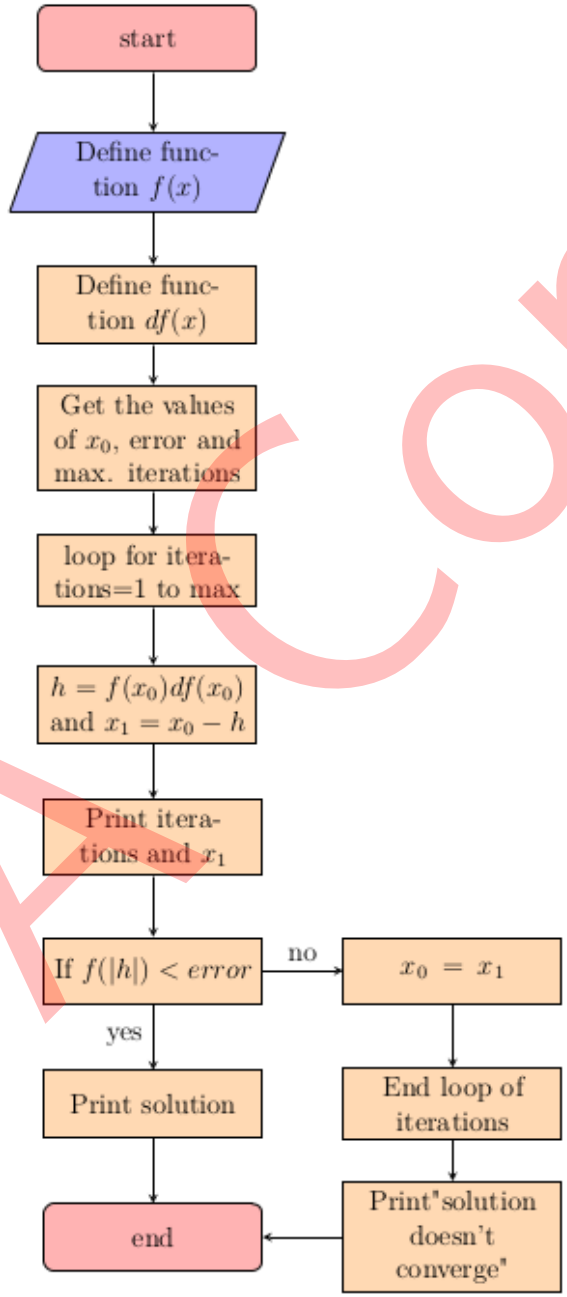
$$f(x) = f(a) + (x-a)f'(a) \quad (9)$$

To find root, we set f(x)=0

$$0 = f(a) + (x-a)f'(a) \quad (10)$$

$$x = a - \frac{f(a)}{f'(a)} \quad (11)$$

The flow chart for the the Newton-Raphson Method is as shown below:



Now we find the root and error for the function $f(x) = cos(x) - x$. The code and the result for this function using Newton-Raphson Method is as shown below:

```
program roots
implicit none
real::x,fx,dfx,tol,error,h
integer::n
External fx,dfx
tol=0.01d0
!give initial guess
x=1.0
111h=-fx(x)/dfx(x)
x=x+h
error = abs(h/x)
if (error .gt. tol) then
goto 111
!tolerance value is set to minimize the error
!as far as possible
else
```

```
print*,"the root and error are",x,"and",error,"respectively"
end if
end program
!since fx and dfx are two external functions
!so they should be defined separately after the program ends
!or before the program begins
real function fx(x)
real::x
fx=cos(x)-x
end function
real function dfx(x)
real::x
dfx=-sin(x)-1.0
end function
```

We obtain root and error as 0.739085138 and 3.75854797E-05 respectively after run the program using Fortran compiler.
We discuss in lectures ...