Program -1 (Bisection)

```
Program Bisection_method
Implicit none
Real ::x,a,b,toll,e,f
Integer ::i,n

F(x)=x*exp(x)-1.0

10 write(*,1)
1 format('Input initial and Ending value of an interval')
  Read(*,*)a,b


If(f(a)*f(b)>0)then
        Write(*,2)
    2 format('No root in this interval',/,'Enter new interval')
        Goto 10
End if

Write(*,3)
3 format('Enter total number of iteration',/,'Enter Tollerance')
Read(*,*)n,toll

Write(*,4)
4 format('Iteration',17x,'A',20x,'B',22x,'X',23x,'Error',20x,"F(x)")

Do i=1,n

    X=(a+b)/2.0
    E=abs(a-b)

       Write(*,5)i,a,b,x,e,f(x)
    5 format(i5,5(5x,f20.10))

    If(f(x)==0.Or.E<=toll)then

    Write(*,6)x,i
    6 format('Approximate root =',f20.15,/,'Number of total iteration=',i5)

    Stop
    Else If(i==n)then
        Write(*,7)n
        7 format('Mehtod failed after',i3,' iteration')
    End if


    If(f(a)*f(x)>0.0)then
    A=x
    Else
    B=x
    End if

End do
Stop
End program
```

Bisection Method

1.Interval Validity Check

a. f(a)*f(b)>0 ----no root

2.Calculation Part

a.fromlula,

x=(a+b)/2.0

b. Error,

er=abs(a-b)

Value Print then

c. Root Check ,

If(f(x)==0.Or.E<=toll)then ----root exist.

d. Failure check , If(i==n)-----failed.

e. Value update

```
If(f(a)*f(x)>0.0)then
A=x
Else
B=x
End if
```

Program -2 (False Position)

```
Program False_Position
  Implicit none
  Real :: f,fa,fb,fx,x,a,b,er,toll
  Integer :: i,n
  F(x)=x*exp(x)-1.0

10 write(*,1)
1 Format('Input initial and final value of an interval')
Read(*,*)a,b

If(f(a)*f(b)>0)then
    Write(*,2)
  2 format('No root in this interval',/,'Enter new interval')
    Goto 10
End if

Write(*,3)
3 format('Enter total number of iteration',/,'Enter tollerance')
Read(*,*)n,toll

Write(*,4)
4 format('Iteration',12x,'A',20x,'B',20x,'X',20x,'Toll')
fa=f(a)
fb=f(b)
Do i=1,n
  x=((a*fb-b*fa)/(fb-fa))
  if(abs(a-x)<=abs(x-b))then
  er=abs(a-x)
  else
    er=abs(b-x)
  end if

  Write(*,5)i,a,b,x,er
  5 format(i5,1x,f20.10,1x,f20.10,1x,f20.10,1x,f20.10)


  If(f(x)==0.Or.Er<toll)then
    Write(*,6)x,i
   6 format('Approximate root =',f20.15,/,'Number of total iteration=',i5)
    Stop
  Else If(i==n)then
  Write(*,7)n
  7 format('Mehtod failed after',i3,' iteration')
  stop
  End if

  If(f(x)*fa>0.0)then
  a=x
  fa=f(x)
  Else
  b=x
  fb=f(x)
```

<u>Regular Falsi Method or Method of false Positio</u>

1.Interval Validity Check

a. f(a)*f(b)>0 ----no root

2.Calculation Part

a.fromlula,

x=((a*fb-b*fa)/(fb-fa))

b. Error,
if(abs(a-x)<=abs(x-b))then
er=abs(a-x)
else

Value Print then

c. Root Check ,

If(f(x)==0.Or.Er<toll)then ----root exist.

d. Failure check ,

If(i==n)-----failed.

e. Value update

If(f(x)*fa>0.0)then
a=x
fa=f(x)
Else
b=x
fb=f(x)

```
   End if

End do
Stop
End program
```

Program -3 (Fixed Point)

```
Program Fixed_Point
Implicit none
Real ::x,f,df,toll,x1,er
Integer ::i,n

F(x)=exp(-x)
Df(x)=-exp(-x)

10 write(*,1)
1 Format('Initial Approximation=')

Read(*,*)x

If(abs(df(x))>1.0)then
   Write(*,2)
   2 format('No root for this point ')
   Goto 10
End if

Write(*,3)
3 format('Enter total number of iteration',/,'Enter Tollerance')
Read(*,*)n,toll


Write(*,4)
4 format('Iteration',17x,'X',20x,'X1',22x,'Error')
Do i=1,n

   X1=f(x)
   Er=abs(x-x1)

   Write(*,5)i,x,x1,er
   5 format(i5,3(5x,f20.10))

   If(er<=toll)then
   Write(*,6)x,i
    6 format('Approximate root=',f20.15,5x,/,'Number of iteration=',i5)
   Stop
   Else If(i==n)then
   Write(*,7)n
   7 format('Mehtod failed after',i3,' iteration')
   End if

   X=x1
End do
Stop
End program
```

Regular Falsi Method or Method of false Positio

```
1.Convergency  Check

   a. If(abs(df(x))>1.0)then
      ----no root

2.Calculation Part

   a.fromlula,

       X1=f(x)
   b. Error,
   Er=abs(x-x1)

    Value Print then

   c. Root Check ,

       If(er<=toll)then----root exist.
   d. Failure check ,

      If(i==n)-----failed.

   e. Value update

      X=x1
```

Program -4 (Nr)

```
Program NR_Method
Implicit none
Real :: x,x1,b,toll,f,er,df
Integer :: i,n

F(x)=x*exp(x)-1.0
Df(x)=exp(x)*(x+1.0)

10 write(*,1)
1 Format('Initial Approximation=')

Read(*,*)x

If(df(x)==0.0)then
    Write(*,2)
    2 format('No root for this point ')
    Goto 10
End if

Write(*,3)
3 format('Enter total number of iteration',/,'Enter Tollerance')
Read(*,*)n,toll

Write(*,4)
4 format('Iteration',17x,'X',20x,'X1',22x,'Error')
Do i=1,n

   X1=x-f(x)/df(x)
   Er=abs(x-x1)

   Write(*,5)i,x,x1,er
   5 format(i5,3(5x,f20.10))

   If(f(x)==0.or.er<=toll)then
    Write(*,6)x1,i
6   format(2x,'Approximate Root=',f30.20,3x,/,'Number of iteration=',i5)
    Stop
   Else If(i==n)then
       Write(*,7)n
       7 format('Mehtod failed after',i3,' iteration')
    End if

   X=x1

End do
Stop
End program
```

Newton Rapshon Method

1.Convergency Check

a . If(df(x)==0.0)then ----no root

2.Calculation Part

a.fromlula,

X1=x-f(x)/df(x)

b. Error,

Er=abs(x-x1)

Value Print then

c. Root Check ,

If(er<=toll)then----root exist.

d. Failure check ,

If(i==n)-----failed.

e. Value update

X=x1

Program -5(NFIF)

```
Program NFIF
```

```
    implicit none
    integer ::i,j,k,n
    real ::x(10),y(10),p,s,t,d(10,10),u,h,f

   !data(x(i),i=1,5)/0.10,0.15,0.20,0.25,0.30/
   !data(y(i),i=1,5)/0.1003,0.1511,0.2027,0.2553,0.3093/

    write(*,1)
    1 format('Enter the total number of point:')
    read(*,*)n

    write(*,3)
    3 format('Enter values of x and y : ')

        Do i=1,n
        read(*,*)x(i),y(i)
       d(i,1)=x(i)
       d(i,2)=y(i)
        End do

    write(*,5 )
    5 format('Enter the value of x at which the functional value is required:')
    read(*,*)t

    h=x(2)-x(1)
    u=(t-x(1))/h
    s=y(1)


    do j=3,n+1
        do i=1,n-j+2
            d(i,j)=d(i+1,j-1)-d(i,j-1)
        end do
    end do

    write(*,9)
    9 format(18x,'Difference table:',/,7x,'X',17x,'Y',12x,'Y1',12x,'Y2',15x,'y3',15x,'y4')

do i=1,n
    write(*,10)(d(i,k),k=1,n-i+2)
    10 format(6(5x,f10.6))
end do

do i=1,n-1
    f=1.0
    p=1.0
    do j=1,i
        p=p*(u-j+1)
        f=f*j
    end do

    s=s+(p/f)*d(1,i+2)
```

NFIF

1.Input value

```
  a. Do i=1,n
       read(*,*)x(i),y(i)
      d(i,1)=x(i)
      d(i,2)=y(i)
       End do
```

2. Find h,u,s

```
   h=x(2)-x(1)
    u=(t-x(1))/h
    s=y(1)
3.Difference Table
    do j=3,n+1
        do i=1,n-j+2
            d(i,j)=d(i+1,j-1)-d(i,j-1)
        end do
    end do
4.Differnce Table Print
do i=1,n
    write(*,10)(d(i,k),k=1,n-i+2)
    10 format(6(5x,f10.6))
end do
5. Find p,f,s
a.    do j=1,i
        p=p*(u-j+1)
        f=f*j
    end do
b. s=s+(p/f)*d(1,i+2)
```

```
end do

write(*,11)t,s
11 format('The corresponding value of (',f4.2,')is=',f10.6)
end program
```

Program -6(NBIF)

```
Program NBIF
   implicit none
   integer ::i,j,k,n
   real ::x(10),y(10),p,s,t,d(10,10),u,h,f

  !data(x(i),i=1,5)/0.10,0.15,0.20,0.25,0.30/
  !data(y(i),i=1,5)/0.1003,0.1511,0.2027,0.2553,0.3093/


   write(*,1)
   1 format('Enter the total number of point:')
   read(*,*)n

   write(*,3)
   3 format('Enter values of x and y : ')

      Do i=1,n
      read(*,*)x(i),y(i)
     d(i,1)=x(i)
     d(i,2)=y(i)
      End do

   write(*,5 )
   5 format('Enter the value of x at which the functional value is required:')
   read(*,*)t

   h=x(2)-x(1)
   u=(t-x(n))/h
   s=y(n)


   do j=3,n+1
      do i=1,n-j+2
         d(i,j)=d(i+1,j-1)-d(i,j-1)
      end do
   end do

   write(*,9)
   9 format(18x,'Difference table:',/,9x,'X',19x,'Y',14x,'Y1',14x,'Y2',17x,'y3',17x,'y4')

do i=1,n
   write(*,10)(d(i,k),k=1,n-i+2)
   10 format(6(5x,f12.8))
end do
```

NBIF

1.Input value

```
 Do i=1,n
    read(*,*)x(i),y(i)
   d(i,1)=x(i)
   d(i,2)=y(i)
    End do
```

2. Find h,u,s

```
 h=x(2)-x(1)
  u=(t-x(1))/h
  s=y(n)
```

3.Difference Table

```
  do j=3,n+1
     do i=1,n-j+2
        d(i,j)=d(i+1,j-1)-d(i,j-1)
     end do
  end do
```

4.Differnce Table Print

```
do i=1,n
  write(*,10)(d(i,k),k=1,n-i+2)
  10 format(6(5x,f10.6))
end do
```

5. Find p,f,s

```
a.  do j=1,i
       p=p*(u+j-1.0)
      f=f*j
    end do
b.    s=s+(p/f)*d(n-i,i+2)
```

```
do i=1,n-1
   f=1.0
   p=1.0
   do j=1,i
      p=p*(u+j-1.0)
      f=f*j
   end do

   s=s+(p/f)*d(n-i,i+2)
end do

write(*,11)t,s
11 format('The corresponding value of (',f5.2,')is=',f12.8)
end program
```

Program -7(Divided)

```
Program divided_difference
Implicit None
Real :: x(10),y(10),d(10,10),t,p,s
Integer ::i,j,n
  ! data(x(i),i=1,5)/0.10,0.15,0.20,0.25,0.30/
   !data(y(i),i=1,5)/0.1003,0.1511,0.2027,0.2553,0.3093/


   write(*,1)
   1 format('Enter the total number of point:')
   read(*,*)n

   write(*,2)
   2 format('Enter values of x and y : ')
      Do i=1,n
      read(*,*)x(i),y(i)
       d(i,1)=x(i)
       d(i,2)=y(i)
      End do

   write(*,3)
   3 format('Enter the value of x at which the functional value is required:')
   read(*,*)t

do j=3,n+1
   do i=1,n-j+2
      d(i,j)=(d(i+1,j-1)-d(i,j-1))/(x(i+j-2)-x(i))
   end do
end do

write(*,4)
```

NBIF

1.Input value

```
  Do i=1,n
     read(*,*)x(i),y(i)
    d(i,1)=x(i)
    d(i,2)=y(i)
     End do
```

3.Difference Table

```
   do j=3,n+1
   do i=1,n-j+2
      d(i,j)=(d(i+1,j-1)-d(i,j-1))/(x(i+j-2)-x(i))
   end do
```

4.Differnce Table Print

```
do i=1,n
   write(*,5)(d(i,j),j=1,n-i+2)
   5 format(6(f10.6))
end do
```

5. Find p,f,s

a.

```
S=y(1)
 do j=1,i
      p=p*(t-x(j))
   end do
```

b. s=s+p*d(1,i+2)

```
4 Format(15x,'Difference Table :',/,5x,'X',8x,'y',8x,'y1',8x,'y2',8x,'y3',8x,'y4')

do i=1,n
   write(*,5)(d(i,j),j=1,n-i+2)
   5 format(6(f10.6))
end do

s=d(1,2)
do i=1,n-1
   p=1.0
   do j=1,i
      p=p*(t-x(j))
   end do
   s=s+p*d(1,i+2)
end do
write(*,11)t,s
11 format('The corresponding value of (',f4.2,')is=',f10.6)

end program
```

Program -8(Lagrange)

```
Program Lagrange_Interpolation
   Implicit None
   Integer :: i,j,n
   Real ::x(10),y(10),s,p,t
   !data(x(i),i=1,5)/0.10,0.15,0.20,0.25,0.30/
   !data(y(i),i=1,5)/0.1003,0.1511,0.2027,0.2553,0.

   write(*,1)
   1 format('How many values : ')
   read(*,*)n

   write(*,2)
   2 format('Enter value of x and y : ')

   do I=1,n
   read(*,*)x(i),y(i)
   end do

write(*,3)
3 format('Enter X for required value Of y:')
read(*,*)t

write(*,4)(x(i),i=1,n)
4 format('Value of x : ',5(f15.10))
write(*,5)(y(i),i=1,n)
5 format('Value of y : ',5(f15.10))


s=0.0
```

Lagrange

1.Calculation

a.

```
 Do j=1,n
   If(i/=j)p=p*(t-x(j))/(x(i)-x(j))
   End do
```

b.

```
s=s+y(i)*p
```

```
Do i=1,n

   p=1.0
   Do j=1,n
   If(i/=j)p=p*(t-x(j))/(x(i)-x(j))
   End do

   s=s+y(i)*p
End do

write(*,6)t,s
6 format('Interpolated value of (',f15.10,')=',f15.10)
stop
End program
```

Program -9(Trapezoidal)

```
Program Trapezoidal_rule
   Implicit none
   Integer ::i,j,n
   Real ::h,f,s,x,a,b,g,er

   f(x)=1/(1+x)
   g(x)=Log(1+x)

   write(*,1)
1   format('Enter value of a,b and n :')
   read(*,*)a,b,n

   h=(b-a)/n
   s=0.0
   do i=1,n-1
   s=s+2*f(a+i*h)
   end do
   s=(h/2.0)*(s+f(a)+f(b))
   p=g(b)-g(a)
   er=abs(p-s)
    write(*,2)s,p,er
    2 format('Approximate Value =',f10.5,/,'Exact value=',f10.5,/,'Error',f10.5)
end Program
```

Trapezoidal

1. h=(b-a)/n
2. s=s+2*f(a+i*h)
3. s=(h/2.0)*(s+f(a)+f(b))
4. p=g(b)-g(a)
5. er=abs(p-s)

Program -10(Simpson’s 1/3)

Simpson’s 1/3

1.h=(b-a)/n
2.A=h/3(y0+2(y2+y4+…+y(n-2))+4(y1+y3+…+y(n-1))+yn)
3.s=(h/3.0)*(s+f(a)+f(b))

4. p=g(b)-g(a)

5. er=abs(p-s)

```
Program Simpsons1_3_rule
   Integer ::i,j,n
   Real ::h,f,s,a,b,g,er,p

   f(x)=1/(1+x)
   g(x)=Log(1+x)

   write(*,1)
1   format('Enter value of a,b and n:')
   read(*,*)a,b,n

   h=(b-a)/n
   s=0.0
   do i=1,n-1
      if(mod(i,2)==0.0)then
      s=s+2*f(a+i*h)
      else
      s=s+4.0*f(a+i*h)
      end if
   end do

   s=(h/3.0)*(s+f(a)+f(b))
    p=g(b)-g(a)
   er=abs(p-s)
    write(*,2)s,p,er
    2 format('Approximate Value =',f10.5,/,'Exact value=',f10.5,/,'Error',f10.5)
```

```
Endprogram
```

Program -11(Simpson’s 3/8)

Simpson’s 3/8

1.h=(b-a)/n
2.A=3h/8(y0+2(y3+y6+…+y(n-3))+
3(y1+y2+y4+y5…+y(n-3))+yn)
3.s=(3h/8.0)*(s+f(a)+f(b))

4. p=g(b)-g(a)

5. er=abs(p-s)

```
Program Simpson3_8_rule
   Integer ::i,j,n
  Real ::h,f,s,a,b,g,er,p
   f(x)=1/(1+x)
   g(x)=Log(1+x)

   write(*,1)
1   format('Enter value of a,b and n:')
   read(*,*)a,b,n

   h=(b-a)/n
   s=0.0

   do i=1,n-1
      if(mod(i,3)==0.0)then
      s=s+2*f(a+i*h)
      else
      s=s+3.0*f(a+i*h)
      end if
   end do

   s=(3.0*h/8.0)*(s+f(a)+f(b))

   p=g(b)-g(a)
   er=abs(p-s)
```

```
    write(*,2)s,p,er
    2 format('Approximate Value =',f10.5,/,'Exact value=',f10.5,/,'Error',f10.5)
End program
```

Program -12(Weddle's)

```
Program Weddles_rule
   Integer ::i,j,n
   Real ::h,f,s,x,a,b,g,er

   f(x)=1/(1+x**2)
   g(x)=atan(x)

   write(*,1)
1   format('Enter value of a,b and n :')
   read(*,*)a,b,n

   h=(b-a)/n
   s=0.0
   do i=1,n-1
      if(mod(i,6)==0.0)then
      s=s+2*f(a+i*h)
      else if(mod(i,3)==0.0)then
      s=s+6*f(a+i*h)
      else if(mod(i,2)==0.0)then
      s=s+f(a+i*h)
      else
         s=s+5*f(a+i*h)
      end if
```

Weddle's

1.h=(b-a)/n
2.A=3h/10(y0+2(y6+y12+…+y(n-6))+
6(y3+y9+y15+y21…+y(n-3))+
(y2+y4+y8+y10…+y(n-2))+
5(y1+y5+y7+y11…+y(n-5))+yn)
3.s=(3h/8.0)*(s+f(a)+f(b))

4. p=g(b)-g(a)

5. er=abs(p-s)

```
    end do
    s=(3.0*h/10.0)*(s+f(a)+f(b))
    p=g(b)-g(a)
    er=abs(p-s)
     write(*,2)s,p,er
     2 format('Approximate Value =',f10.5,/,'Exact value=',f10.5,/,'Error',f10.5)
Endprogram
```

Program -13(TSSW)

```
Program integration
Implicit None
Integer :: i,n
Real :: a,b,h,s,ex,f,g,x,r(10),er(10)
g(x)=atan(x)

write(*,1)
1 format("Enter a ,b,n : ")
read(*,2)a,b,n
2 format(f3.1,1x,f3.1,1x,i2)

h=(b-a)/n
write(*,*)a,b,n,h

ex=g(b)-g(a)

write(*,3)
3 format(15x,"Approximate value : ",5x,'Exact value : ',5x,'Error : ')

call tr(a,b,h,s,n,f)
r(1)=s
er(1)=abs(s-ex)
write(*,4)r(1),ex,er(1)
    4 format('Trapezoidal=',3(8x,f10.6))

call sot(a,b,h,s,n)
r(2)=s
er(2)=abs(s-ex)
write(*,5)r(2),ex,er(2)
```

```
    5 format('Simpsons 1/3',3(8x,f10.6))

call ste(a,b,h,s,n)
r(3)=s
er(3)=abs(s-ex)
write(*,6)r(2),ex,er(2)
    6 format('Simpsons 3/8',3(8x,f10.6))

call wed(a,b,h,s,n)
r(4)=s
er(4)=abs(s-ex)
write(*,7)r(3),ex,er(3)
    7 format('Weddles',5x,3(8x,f10.6))

Stop
End Program

real function f(x)
f=1.0/(1.0+x**2)
return
end
Subroutine tr(a,b,h,s,n)
Implicit None
integer ::i,n
Real ::a,b,h,s,f
s=0.0
Do i=1,n-1
    s=s+2.0*f(a+i*h)
End do
s=h/2.0*(f(a)+f(b)+s)
End subroutine

Subroutine sot(a,b,h,s,n)
Implicit None
integer :: i,n
Real ::a,b,h,s,f
s=0.0
Do i=1,n-1
    if(Mod(i,2)==0)then
    s=s+2*f(a+i*h)
    else
    s=s+4*f(a+i*h)
    end if
End do

s=h/3.0*(f(a)+f(b)+s)

End subroutine

Subroutine ste(a,b,h,s,n)
Implicit None
integer :: i,n
Real ::a,b,h,s,f
s=0.0
Do i=1,n-1
    if(Mod(i,3)==0)then
    s=s+2*f(a+i*h)
    else
    s=s+3*f(a+i*h)
    end if
End do

s=3.0*h/8.0*(f(a)+f(b)+s)
End subroutine

Subroutine wed(a,b,h,s,n)
Implicit None
integer :: i,n
Real ::a,b,h,s,f
s=0.0
Do i=1,n-1
    if(Mod(i,6)==0)then
    s=s+2*f(a+i*h)
    else if(mod(i,3)==0)then
    s=s+6.0*f(a+i*h)
    else if(mod(i,2)==0)then
    s=s+f(a+i*h)
    else
    s=s+5*f(a+i*h)
    end if
End do
```

```
s=3.0*h/10.0*(f(a)+f(b)+s)
End subroutine
```

Program -13(Romberg Integration)

```
 Program Romberg_Integration
 Implicit none
Integer :: i,j,k,n
Real ::a,b,h(10),f,g,er,ex,r(10,10),x,s,toll
f(x)=1.0/(1.0+x**2)
g(x)=atan(x)

write(*,1)
1 format("How many partition : ")
read(*,*)n
write(*,2)
2 format("Enter value of a and b: ")
read(*,*)a,b


write(*,3)
3 format("Enter tolerance  : ")
read(*,*)toll


 Do i=1,n
   h(i)=(b-a)/2**(i-1)
 End do
 r(1,1)=0.5*h(1)*(f(a)+f(b))

 write(*,4)r(1,1)
 4 format(2x,f12.8)

 do i=2,n
   s=0.0
   do j=1,2**(i-2)
      s=s+f(a+(2.0*j-1)*h(i))
   End do
   r(i,1)=0.5*(r(i-1,1)+h(i-1)*s)

   do j=2,i
      r(i,j)=r(i,j-1)+(r(i,j-1)-r(i-1,j-1))/(4**(j-1)-1.0)
   End do

   write(*,5)(r(i,j),j=1,i)
   5 format(15(2x,f12.8))

    if(abs(r(i,i)-r(i-1,i-1))<=toll)then
```

Romberg Integration

1.Find H

```
Do i=1,n
   h(i)=(b-a)/2**(i-1)
 End do
```

2. Find r(1,1)

```
r(1,1)=0.5*h(1)*(f(a)+f(b))
```

3.Find for r(i,1)

a.

```
   do j=1,2**(i-2)
      s=s+f(a+(2.0*j-1)*h(i))
   End do
```

b.

```
r(i,1)=0.5*(r(i-1,1)+h(i-1)*s)
```

c.use r(I,1) for r(I,j)

```
   do j=2,i
      r(i,j)=r(i,j-1)+(r(i,j-1)-r(i-1,j-1))/(4**(j-1)-1.0)
   End do
```

d.value print

```
write(*,5)(r(i,j),j=1,i)
   5 format(15(2x,f12.8))
```

3.Root check

a. if(abs(r(i,i)-r(i-1,i-1))<=toll)then----root exist

b.exact,   ex=g(b)-g(a)

c.error, er=abs(ex-r(i,i))

d.Failure check

if (i==n)then---failed

```
    ex=g(b)-g(a)
    er=abs(ex-r(i,i))
    write(*,6)r(i,i),ex,er
    6 format('Approximate value =',f20.10,/,'Exact value =',f20.10,/,'Error =',f20.10)
    stop
    else if (i==n)then
      write(*,7)i
      7 format("Method Failed after ",i2,' Iteration ')
    end if
 End do

Stop
End Program
```

Set A :

1. Use bisection method with FORTRAN Program to approximate one of the roots $x^2 - 2x - 5 = 0$ by pecifying 15 iterations.
2. The populations of a town in the decimal census was given below . Estimate the population for the year 1895 using Newton's Interpolation formula.

Year x	1891	1901	1911	1921	1931
Population y	46	66	81	93	101

3. Determine the value of $\int_3^7 x^2 \log(x) dx$ by using Simpson's rule. Also compare the solution with exact solution

Set B :

4. Use fixed Point iteration method to determine a real root accurate to within 10^{-2} for $x^4 - 3x^2 - 3 = 0$ on [1,2]
5. Write a program to find $\log_{10} 301$, Certain values are given .

x	300	304	305	307
y	46	66	81	93

6. Determine the value of $\int_0^1 \sqrt{1 - x^2}\, dx$ by using Simpson's rule 3/8. Also compare the solution with exact

Set C :

7.Use Newton Raphson method to determine a real root for $2x - Cos(x) - 3 = 0$ specifying 0.0001 tolerance.

8.The populations of a town in the decimal census was given below . Estimate the population for the year 1895 using Newton's Interpolation formula.

Year x	1891	1901	1911	1921	1931
Population y	46	66	81	93	101

9. Use trapezoidal rule to approximate $\int_0^e Ln(x)\, dx$ with n=36 . Also compare the solution with exact solution.

Set D :

10. Use Regular Falsi method with FORTRAN Program to approximate a root lies between 0 to 0.5 of the equation $f(x) = 4e^{-x}Sin(x) - 1 = 0$ by pecifying 20 iterations.

11.Find the annual premium at the age of 31 from the followings table by using newton's Divided Difference formula .

Age	21	25	29	33
Premium	14.27	15.81	17.72	19.96

12. Determine the value of $\int_3^7 x^2 \log(x) dx$ by using Simpson's 3/8 rule. Also compare the solution with exact solution

Program -(Richardson Extrapolation)

```
program richarson_extrapolation
    implicit none
    integer :: i,j,n
    real :: x,h(20),r(20,20),f,g,ex,toll,er

    write(*,11)
    11 format('Give difference h: ',/,'Differentiating
     point x',/,'Give n and  tollerance :')
    read(*,*)h(1),x,n,toll

    Do i=2,n
    h(i)=h(1)/2**(i-1)
    End do

    r(1,1)=(f(x+h)-f(x-h))/(2.0*h(1))
    write(*,1)
    1 format(30x,"Richardson Extrpolation ",/,5x,
'Iteration,',10x,'O(h)',10x,"O(h^2)",17x,"O(h^3",
10x,"O(h^4)")
    write(*,4)1,r(1,1)
    4 format(i10,8x,f12.8)

    do i = 2, n
        r(i,1)=(f(x+h(i))-f(x-h(i)))/(2*h(i))
    do j=2,i
    r(i,j)=r(i,j-1)+(r(i,j-1)-r(i-1,j-1))/(4**(j-1)-1.0)
    End do

    write(*,5)i,(r(i,j),j=1,i)
    5 format(i10,15(8x,f12.8))

     if(abs(r(i,i)-r(i-1,i-1))<=toll)then
     ex=g(x)
```

Romberg Integration

1.Find H

```
Do i=1,n
    h(i)=(b-a)/2**(i-1)
 End do
```

2. Find r(1,1)

```
    r(1,1)=(f(x+h)-f(x-h))/(2.0*h(1))
```

3.Find for r(i,1)

a.

```
do i = 2, n
        r(i,1)=(f(x+h(i))-f(x-h(i)))/(2*h(i))
    do j=2,i
```

b.use r(I,1) for r(I,j)

```
do j=2,i
    r(i,j)=r(i,j-1)+(r(i,j-1)-r(i-1,j-1))/(4**(j-1)-1.0)
    End do
```

c.value print

```
    write(*,5)i,(r(i,j),j=1,i)
    5 format(i10,15(8x,f12.8))
```

3.Root check

a. if(abs(r(i,i)-r(i-1,i-1))<=toll)then----root exist

b.exact, ex=g(b)-g(a)

c.error, er=abs(ex-r(i,i))

d.Failure check

if (i==n)then---failed

```
     er=abs(ex-r(i,i))
     write(*,6)r(i,i),ex,er
     6 format('Approximate value =',f20.10,/,'Exact value =',f20.10,/,'Error =',f20.10)
     stop
     else if (i==n)then
        write(*,7)i
        7 format("Method Failed after ",i3,' Iteration ')
     end if

    End do
stop
end

real function f(x)
    f = x**2*Sin(x)
end function
real function g(x)
     g=2*x*Sin(x)+x**2*cos(x)
end function
```

Program -(LU Decomposition)

```
PROGRAM LU
    IMPLICIT NONE
    INTEGER,PARAMETER::n=3
    INTEGER::i,j,k
    real ::a(n,n),L(n,n)=0.0,u(n,n)=0.0,b(n),x(n),y(n),s

    a=reshape((/2,1,3,3,2,1,1,3,2/),shape(a))
    data(b(i),i=1,3)/9,6,8/

   ! READ(*,*)((a(i,j),j=1,n),b(i),i=1,n)

    WRITE(*,7)"A=",((a(i,j),j=1,n),i=1,n)
    WRITE(*,8)"b=",(b(i),i=1,n)


    l=0
    u=a
    DO k=1,n

        l(k,k)=1
        DO i=k+1,n
            l(i,k)=u(i,k)/u(k,k)

            DO j=1,n
                u(i,j)=u(i,j)-l(i,k)*u(k,j)
            END DO

        END DO
    END DO

    WRITE(*,7)"L=",((l(i,j),j=1,n),i=1,n)
```

LU -Decomposition

```
1.  L=0,u=a(u11=a11,u12=a12,u13=a13)
2.  Let,(l11,l22,l33=1)
3.  Find (l21,l31,l32)
    a.  DO i=k+1,n
          l(i,k)=u(i,k)/u(k,k)

    b.  Find (u22,u23,u33)
         DO j=1,n
          u(i,j)=u(i,j)-l(i,k)*u(k,j)
4.  Find(y1,y2,y3)
     DO i=1,n
          s=b(i)
          DO j=1,i-1
              s=s-l(i,j)*y(j)
          END DO
          y(i)=s/l(i,i)
    END DO
5.Find(x1,x2,x3)
 DO i=n,1,-1
    s=y(i)
    DO j=i+1,n
       s=s-u(i,j)*x(j)
    END DO
    x(i)=s/u(i,i)
 END DO
```

```
  WRITE(*,7)"U=",((u(i,j),j=1,n),i=1,n)

  DO i=1,n
    s=b(i)
    DO j=1,i-1
      s=s-l(i,j)*y(j)
    END DO
    y(i)=s/l(i,i)
  END DO

  WRITE(*,8)"y=",(y(i),i=1,n)

  DO i=n,1,-1
    s=y(i)
    DO j=i+1,n
      s=s-u(i,j)*x(j)
    END DO
    x(i)=s/u(i,i)
  END DO

  WRITE(*,8)"x=",(x(i),i=1,n)

  7 FORMAT(a,/,3(3(f9.3,2x),/))
  8 FORMAT(a,/,3(f9.3,/))

END PROGRAM
```

Program –(Gauss Elimination with pivoting)

Gauss Elimination with Pivoting

1. Pivoting

2.Eliminate variable
3 .Back substitution
4. Print Result

```
program Gaussian_Elimination_with_pivoting
 implicit none
 integer :: i,j,k,n, mr
 real :: a(3,4),x(10),mv,p

  a=reshape((/2,1,3,3,2,1,1,3,2,9,6,8/),(/3,4/))

 print*,"Enter the number of equations (max 10):"
 read*,n

 print*,"Enter the augmented matrix (n x n+1 elements):"
 do i = 1, n
   do j = 1, n+1
     !read *, A(i, j)
   end do
 end do

 do i=1,N-1
  mr=i
  mv=abs(A(i,i))

  do j=i+1,N
   if (abs(A(j,i))> mv) then
    mr=j
```

```
        mv=abs(A(j, i))
      end if
    end do

    if (mr/=i) then
      do k = 1,N+1
        p= A(i,k)
        A(i,k)= A(mr,k)
        A(mr,k)=p
      end do
    end if

    ! Eliminate variables
    do j=i+1,N
      p= A(j,i)/A(i, i)
      do k=i,N+1
        A(j,k) =A(j,k)-p*A(i,k)
      end do
    end do
end do

  ! Back substitution
  x(N)=A(N,N+1)/A(N, N)
  do i=N-1,1,-1
    x(i)=A(i,N+1)
    do j=i+1,N
      x(i)=x(i)-A(i,j)*x(j)
    end do
    x(i)=x(i)/A(i, i)
  end do

  ! Print the solution
  print *, 'Solution:'
  do i = 1, N
    write(*,*)'x(', i,') = ', x(i)
  end do
end program
```

Program –(Gauss Elimination without pivoting)

Gauss Elimination without Pivoting

1. Eliminate variable
2. Back substitution
3. Print Result

```
program B19_05_Gaussian_Elimination_without_pivoting
  implicit none
  integer :: n,i,j,k
  real :: A(3,4), X(10),s,p
  a=reshape((/2,1,3,3,2,1,1,3,2,9,6,8/),(/3,4/))

  print*,"Enter the number of equations (max 10):"
  read*,n

  print*,"Enter the augmented matrix (n x n+1 elements):"
  do i=1,n
     do j=1,n+1
        !read (*,*)A(i,j)
     end do
  end do

  do i=1,n-1
     do j=i+1,n
        p=A(j,i)/A(i,i)
        do k=i,n+1
           A(j,k)=A(j,k)-p*A(i,k)
        end do
     end do
  end do

  X(n)=A(n,n+1)/A(n, n)
  do i=n-1,1,-1
     s=0.0
     do j=i+1,n
        s=s+A(i,j)*X(j)
     end do
     X(i)=(A(i,n+1)-s)/A(i, i)
end do
print*,"Solution:"
do i = 1, n
    write(*,*) "X(", i, ") = ", X(i)
end do
end program
```

Program -(Jacobii)

```
Program B19_05_Jacobii
Implicit none
Integer ::i,j,k,m,n,r
Real :: a(3,3),b(10),x0(10),x(10),toll,s,n1,n2
 a=reshape((/2,1,3,3,2,1,1,3,2/),shape(a))
data(b(i),i=1,3)/9,6,8/

write(*,1)
1 format('How many equation : ',/,"Enter tolerance :",/,'Enter Number of iterations: ')
read(*,*)n,toll,m

do i=1,n
    print*,"Enter equation= ",i
    do j=1,n
        !read(*,*)a(i,j)
    end do
    !read*,b(i)
end do

write(*,*),'Give initial approximation'
read(*,*),(x0(i),i=1,n)

 write(*,10)
10 format('Iteration',12X,'x1',10X,'x2',10X,"x3",10X,"x4")
```

```
do k=1,m
    do i=1,n
        s=0.0
        do j=1,n
            if(i/=j) then
                s=s+a(i,j)*x0(j)
            end if
        end do
        x(i)=(b(i)-s)/a(i,i)
    end do

write(*,2)k,(x(i),i=1,n)
2 format(i4,4(5x,f10.6))

n1=abs(x(1)-x0(1))
n2=abs(x(1))

 do r=2,n
    if(n1<abs(x(r)-x0(r)))n1=abs(x(r)-x0(r))
    if(n2<abs(x(r)))n2=abs(x(r))
 end do
        if((n1/n2)<toll)then
        write(*,22)(x(i),i=1,n)
       22 format('Approximate root :',4(f20.15))
        stop
       else if(k==m)then
        write(*,11)k
        11 format('Method failed after ',i3,' iterations')
        stop
      else
        do i=1,n
        x0(i)=x(i)
        end do
    end if
end do
stop
End Program
```

Program -(Gauss Seidal method)

```
Program B19_05_Gauss_Seidal_Method
Implicit none
Integer ::i,j,k,m,n,r
Real :: a(3,3),b(10),x0(10),x(10),toll,s,n1,n2
 a=reshape((/2,1,3,3,2,1,1,3,2/),shape(a))
data(b(i),i=1,3)/9,6,8/
write(*,1)
1 format('How many equation : ',/,"Enter tolerance :",/,'Enter Number of iterations: ')
read(*,*)n,toll,m

do i=1,n
    print*,"Enter equation= ",i
    do j=1,n
        !read(*,*)a(i,j)
    end do
    !read*,b(i)
end do


write(*,*),'Give initial approximation'
read(*,*),(x0(i),i=1,n)

 write(*,10)
10 format('Iteration',12X,'x1',10X,'x2',10X,"x3",10X,"x4")

do k=1,m
    do i=1,n
        s=0.0
        do j=1,n
            if(i<j) then
                s=s+a(i,j)*x0(j)
            else if(i>j)then
                s=s+a(i,j)*x(j)
            end if
        end do
        x(i)=(b(i)-s)/a(i,i)
    end do

write(*,2)k,(x(i),i=1,n)
2 format(i4,4(5x,f10.6))

n1=abs(x(1)-x0(1))
n2=abs(x(1))

 do r=2,n
    if(n1<abs(x(r)-x0(r)))n1=abs(x(r)-x0(r))
    if(n2<abs(x(r)))n2=abs(x(r))
 end do

        if((n1/n2)<toll)then
        write(*,22)(x(i),i=1,n)
       22 format('Approximate root :',4(f20.15))
        stop
       else if(k==m)then
        write(*,11)k
        11 format('Method failed after ',i3,' iterations')
        stop
      else
        do i=1,n
        x0(i)=x(i)
        end do
    end if
end do
stop
```

```
End Program
```

Program -(SOR Method

```
Program B19_05_Successive_Over_Relaxation
Implicit none
Integer ::i,j,k,m,n,r
Real :: a(10,10),b(10),x0(10),x(10),toll,s,n1,n2,w

write(*,1)
1 format('How many equation : ',/,"Enter tolerance :",/,'Enter Number of iterations: ',/,'Give value of the Omega:
')
read(*,*)n,toll,m,w

do i=1,n
    print*,"Enter equation= ",i
    do j=1,n
        read(*,*)a(i,j)
    end do
    read*,b(i)
end do


write(*,*),'Give initial approximation'
read(*,*),(x0(i),i=1,n)

 write(*,10)
10 format('Iteration',12X,'x1',10X,'x2',10X,"x3",10X,"x4")

do k=1,m
    do i=1,n
        s=0.0
        do j=1,n
            if(i<j) then
                s=s+a(i,j)*x0(j)
            else if(i>j)then
                s=s+a(i,j)*x(j)
            end if
        end do
        x(i)=(1-w)*x0(i)+(w/a(i,i))*(b(i)-s)
    end do

write(*,2)k,(x(i),i=1,n)
2 format(i4,4(5x,f10.6))

n1=abs(x(1)-x0(1))
n2=abs(x(1))

 do r=2,n
    if(n1<abs(x(r)-x0(r)))n1=abs(x(r)-x0(r))
    if(n2<abs(x(r)))n2=abs(x(r))
 end do
```

```
        if((n1/n2)<toll)then
        write(*,22)(x(i),i=1,n)
       22 format('Approximate root :',4(f20.15))
        stop
       else if(k==m)then
        write(*,11)k
        11 format('Method failed after ',i3,' iterations')
        stop
      else
        do i=1,n
        x0(i)=x(i)
        end do
    end if
end do
stop
End Program
```

Program -(Rk-2,Rk-4 Method   )

```
Program Rk_2_4
Implicit None
Integer :: i,n
Real ::a,b,h,x,y,y1,er1,er2,f,g

write(*,*)'Enter Intinal and Ending value of x :'
read(*,*)a,b

x=a
write(*,1)
1 Format('Enter value of n  :',/,' Enter Inital value of y :')
read(*,*)n,y

y1=y
h=(b-a)/n

write(*,10)
10 format('Iteration',5x,'x(i)',10x,'Exact',8x,'Rk_2(yi)',7x,'Rk_2 Rrror',6x,'Rk_4(yi)',7x,'Rk_4 Error')
```

```
Do i=1,n+1
er1=abs(g(x)-y)
er2=abs(g(x)-y1)

write(*,2)i,x,g(x),y,er1,y1,er2
2 format(i5,6(5x,f10.7))
call Rk_2(h,x,y)
call Rk_4(h,x,y1)
x=a+i*h

End do

stop
End Program

real function f(x,y)
f=y-x**2+1
end function
real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function

Subroutine Rk_2(h,x,y)
 implicit none
 real:: k1,k2,h,x,y,f

  k1=h*f(x,y)
  k2=h*f(x+h,y+k1)
  y=y+(k1+k2)/2

End Subroutine
```

```
Subroutine Rk_4(h,x,y1)
    Implicit none
    real :: k1,k2,k3,k4,h,x,y1,y,f
  k1=h*f(x,y1)
  k2=h*f(x+h/2,y1+k1/2)
  k3=h*f(x+h/2,y1+k2/2)
  k4=h*f(x+h,y1+k3)

  y1=y1+(k1+2*k2+2*k3+k4)/6

End Subroutine
```

Program -(Rk-2)

```
Program Rk_2
Implicit None
Integer ::i,j,n
Real :: a,b,h,x,y,f,k1,k2,er,g

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
x=a
Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y,n

h=(b-a)/n

write(*,3)
3 format('Iteration',3x,'x',10x,'y',10x,'Exact',15x,'Error')
Do i=1,n+1
```

```
 er=abs(g(x)-y)
 write(*,4)i,x,y,g(x),er
 4 format(i4,5x,f5.2,3(4x,f12.8))
 k1=h*f(x,y)
 k2=h*f(x+h,y+k1)

 y=y+(k1+k2)/2

 x=a+i*h

End do

stop
End Program

real function f(x,y)
f=y-x**2+1
end function
real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function
```

Program -(Rk-4)

```
Program Rk_4
Implicit None
Integer ::i,j,n
Real :: a,b,h,x,y,f,k1,k2,k3,k4,er,g

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
```

```
x=a

Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y,n

h=(b-a)/n

write(*,3)
3 format('Iteration',3x,'x',10x,'y',10x,'Exact',15x,'Error')
Do i=1,n+1

 er=abs(g(x)-y)
 write(*,4)i,x,y,g(x),er
 4 format(i4,5x,f5.2,3(4x,f12.8))
 k1=h*f(x,y)
 k2=h*f(x+h/2.0,y+k1/2.0)
 k3=h*f(x+h/2.0,y+k2/2.0)
 k4=h*f(x+h,y+k3)
 y=y+(k1+2.0*k2+2.0*k3+k4)/6

 x=a+i*h

End do

stop
End Program

real function f(x,y)
f=y-x**2+1
end function
```

```
real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function
```

Program –(Eulers Method)

```
Program B19_05_Eulers_Method
Implicit none
Integer :: i,n
Real ::a,b,x,y,g,er,y0,h,y1,er1,f

Write(*,1)
1 format('Enter Initial and Ending value of (x) a and b : ',/,'Value of n : ',/,"Initial value of Y :")
read(*,*)a,b,n,y



h=(b-a)/n
x=a
y=y

write(*,2)
2 format(3x,'x(i)',5x,'exact value',5x,'eu_y(i)',5x,'eu error')

do i=1,n+1
   er=abs(g(x)-y)
   write(*,3)x,g(x),y,er
   3 format(2x,f5.2,3(5x,f8.5))
   call euler(h,x,y)
   x=a+i*h
end do
End Program

real function f(x,y)
```

```
f=y+cos(x)
end function
real function g(x)
g=0.5*(3.0*exp(x)-cos(x)+sin(x))
end function

Subroutine euler(h,x,y)
   implicit none
   real :: x,y,h,f
   y=y+h*f(x,y)
end subroutine
```

Program –(modified Eulers Method)

```
Program B19_05_M_Eulers_Method
Implicit none
Integer :: i,n
Real ::a,b,x,y,g,er,y0,h,y1,er1,f



Write(*,1)
1 format('Enter Initial and Ending value of (x) a and b : ',/,'Value of n : ',/,"Initial value of Y :")
read(*,*)a,b,n,y


h=(b-a)/n
x=a
y1=y

write(*,2)
2 format(3x,'x(i)',5x,'exact value',5x,'meu_y(i)',5x,'meu_error')
```

```
do i=1,n+1

    er1=abs(g(x)-y1)
    write(*,14)x,g(x),y1,er1
    14 format(2x,f5.2,3(5x,f8.5))
    call m_euler(h,x,y1)
    x=a+i*h
end do
End Program

real function f(x,y)
f=y+cos(x)
end function
real function g(x)
g=0.5*(3.0*exp(x)-cos(x)+sin(x))
end function



Subroutine m_euler(h,x,y1)
    implicit none
    real :: x,y1,h,f
    y1=y1+h*0.5*(f(x,y1)+f(x+h,y1+h*f(x,y1)))
end subroutine
```

Program –(Combined Eulers Method)

```
Program B19_05_Combine_Euler_Method
Implicit none
Integer :: i,n
Real ::a,b,x,y,g,er,y0,h,y1,er1,f
```

```
Write(*,100)
100 format('Enter Initial and Ending value of (x) a and b : ',/,'Value of n : ',/,"Initial value of Y :")
read(*,*)a,b,n,y


h=(b-a)/n
x=a
y=y
y1=y

write(*,1)
1 format(3x,'x(i)',5x,'exact value',5x,'eu_y(i)',5x,'eu error',4x,'meu_y(i)',5x,'meu_error')

do i=1,n+1
   er=abs(g(x)-y)
   er1=abs(g(x)-y1)
   write(*,14)x,g(x),y,er,y1,er1
   14 format(2x,f5.2,5(5x,f8.5))
   call euler(h,x,y)
   call m_euler(h,x,y1)
   x=a+i*h
end do
End Program

real function f(x,y)
f=y+cos(x)
end function
real function g(x)
g=0.5*(3.0*exp(x)-cos(x)+sin(x))
```

```
end function

Subroutine euler(h,x,y)
   implicit none
   real :: x,y,h,f
   y=y+h*f(x,y)
end subroutine
Subroutine m_euler(h,x,y1)
   implicit none
   real :: x,y1,h,f
   y1=y1+h*0.5*(f(x,y1)+f(x+h,y1+h*f(x,y1)))
end subroutine
```

Program –(Adam bshforhth 3 step)

```
Program Adam_3
Implicit None
Integer ::i,j,n
Real :: a,b,h,y0=0.5,x(100),y(100),f,k1,k2,k3,k4,er(50),g

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
x(1)=a

Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y(1),n
h=(b-a)/n
```

```
er(1)=abs(y(1)-g(x(1)))
Do i=2,3

 x(i)=x(i-1)+h

 k1=h*f(x(i-1),y(i-1))
 k2=h*f(x(i-1)+h/2.0,y(i-1)+k1/2.0)
 k3=h*f(x(i-1)+h/2.0,y(i-1)+k2/2.0)
 k4=h*f(x(i-1)+h,y(i-1)+k3)
 y(i)=y(i-1)+(k1+2.0*k2+2.0*k3+k4)/6

 er(i)=abs(y(i)-g(x(i)))

End do

do i=4,n+1
     x(i)=x(i-1)+h
y(i)=y(i-1)+(h/12.0)*(23*f(x(i-1),y(i-1))-16*f(x(i-2),y(i-2))+5*f(x(i-3),y(i-3)))
er(i)=abs(y(i)-g(x(i)))

End do
write(*,3)
3 format('Iteration',3x,'x',10x,'y',14x,'Exact',12x,'Error')
do i=1,n+1
 write(*,4)i,x(i),y(i),g(x(i)),er(i)
 4 format(i4,5x,f5.2,3(4x,f12.8))
End do
```

```
stop
End Program



real function f(x,y)
f=y-x**2+1
end function

real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function
```

Program –(Adam bshforhth 4 step)

```
Program Adam_4
Implicit None
Integer ::i,j,n
Real :: a,b,h,x(100),y(100),f,k1,k2,k3,k4,er(50),g

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
x(1)=a

Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y(1),n
```

```
h=(b-a)/n



er(1)=abs(y(1)-g(x(1)))

Do i=2,4

 x(i)=x(i-1)+h

 k1=h*f(x(i-1),y(i-1))
 k2=h*f(x(i-1)+h/2.0,y(i-1)+k1/2.0)
 k3=h*f(x(i-1)+h/2.0,y(i-1)+k2/2.0)
 k4=h*f(x(i-1)+h,y(i-1)+k3)
 y(i)=y(i-1)+(k1+2.0*k2+2.0*k3+k4)/6
 er(i)=abs(y(i)-g(x(i)))
End do


do i=5,n+1
     x(i)=x(i-1)+h
    y(i)=y(i-1)+(h/24.0)*(55*f(x(i-1),y(i-1))-59*f(x(i-2),y(i-2))+37*f(x(i-3),y(i-3))-9*f(x(i-4),y(i-4)))
    er(i)=abs(y(i)-g(x(i)))
End do
write(*,3)

3 format('Iteration',3x,'x',10x,'y',14x,'Exact',12x,'Error')
```

```
do i=1,n+1
 write(*,4)i,x(i),y(i),g(x(i)),er(i)
 4 format(i4,5x,f5.2,3(4x,f12.8))
End do

stop
End Program



real function f(x,y)
f=y-x**2+1
end function

real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function
```

Program –(p_c 3step)

```
Program B19_05_p_c_3
Implicit None
Integer ::i,j,n
Real :: a,b,h,y0=0.5,x(100),y(100),yp(100),f,k1,k2,k3,k4,er(50),g

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
x(1)=a
```

```
Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y(1),n
h=(b-a)/n

er(1)=abs(y(1)-g(x(1)))
Do i=2,3

 x(i)=x(i-1)+h

 k1=h*f(x(i-1),y(i-1))
 k2=h*f(x(i-1)+h/2.0,y(i-1)+k1/2.0)
 k3=h*f(x(i-1)+h/2.0,y(i-1)+k2/2.0)
 k4=h*f(x(i-1)+h,y(i-1)+k3)
 y(i)=y(i-1)+(k1+2.0*k2+2.0*k3+k4)/6

 er(i)=abs(y(i)-g(x(i)))

End do

do i=4,n+1
     x(i)=x(i-1)+h
    y(i)=y(i-1)+(h/12.0)*(23*f(x(i-1),y(i-1))-16*f(x(i-2),y(i-2))+5*f(x(i-3),y(i-3)))
    yp(i)=y(i)
    y(i)=y(i-1)+(h/24.0)*(9*f(x(i),y(i))+19*f(x(i-1),y(i-1))-5*f(x(i-2),y(i-2))+f(x(i-3),y(i-3)))
    er(i)=abs(y(i)-g(x(i)))

End do
```

```
write(*,3)
3 format('Iteration',3x,'x',10x,'yp',12x,'yc',14x,'Exact',12x,'Error')


do i=1,n+1
    if(i<5) yp(i)=0
 write(*,4)i,x(i),yp(i),y(i),g(x(i)),er(i)
 4 format(i4,5x,f5.2,4(4x,f12.8))
End do



stop
End Program


real function f(x,y)
f=y+cos(x)
end function
real function g(x)
g=0.5*(3.0*exp(x)-cos(x)+sin(x))
end function
```

Program –(p_c _4)

```
Program p_c_4
Implicit None
Integer ::i,j,n
Real :: a,b,h,x(100),y(100),f,k1,k2,k3,k4,er(50),g,yp(100)

Write(*,1)
1 format("Enter Inital and Ending value of x : ")
read(*,*)a,b
x(1)=a
```

```
Write(*,2)
2 format("Enter Inital  y: ",/,'Enter value of n :')
read(*,*)y(1),n
h=(b-a)/n

er(1)=abs(y(1)-g(x(1)))

Do i=2,4

 x(i)=x(i-1)+h

 k1=h*f(x(i-1),y(i-1))
 k2=h*f(x(i-1)+h/2.0,y(i-1)+k1/2.0)
 k3=h*f(x(i-1)+h/2.0,y(i-1)+k2/2.0)
 k4=h*f(x(i-1)+h,y(i-1)+k3)
 y(i)=y(i-1)+(k1+2.0*k2+2.0*k3+k4)/6
 er(i)=abs(y(i)-g(x(i)))
End do



do i=5,n+1
     x(i)=x(i-1)+h
    y(i)=y(i-1)+(h/24.0)*(55*f(x(i-1),y(i-1))-59*f(x(i-2),y(i-2))+37*f(x(i-3),y(i-3))-9*f(x(i-
4),y(i-4)))
    yp(i)=y(i)
    y(i)=y(i-1)+(h/24.0)*(9*f(x(i),y(i))+19*f(x(i-1),y(i-1))-5*f(x(i-2),y(i-2))+f(x(i-3),y(i-3)))
    er(i)=abs(y(i)-g(x(i)))
End do

write(*,3)
3 format('Iteration',3x,'x',10x,'yp',12x,'yc',14x,'Exact',12x,'Error')

do i=1,n+1
    if(i<5) yp(i)=0
 write(*,4)i,x(i),yp(i),y(i),g(x(i)),er(i)
 4 format(i4,5x,f5.2,4(4x,f12.8))
End do

stop
End Program
```

```
real function f(x,y)
f=y-x**2+1
end function

real function g(x)
g=(x+1.0)**2-0.5*exp(x)
end function
```

Program –(p_c _4 explicit)

```
PROGRAM AB4explicit
    IMPLICIT NONE

    INTEGER ::i,n
    REAL::a,b,h,f,df,x,alpha,k(4),t(0:1000),w(0:1000)


    READ(*,*)a,b,alpha,n

    h=(b-a)/n

    t(0)=a
    w(0)=alpha

    WRITE(2,*)"Step      x          y          AB step 4         error"
    i=0
    WRITE(2,8)i,t(i),f(t(i)),w(i)

    DO i=1,3

        k(1)=h*df(t(i-1),w(i-1))
        k(2)=h*df(t(i-1)+h/2,w(i-1)+k(1)/2)
        k(3)=h*df(t(i-1)+h/2,w(i-1)+k(2)/2)
        k(4)=h*df(t(i-1)+h,w(i-1)+k(3))

        w(i)=w(i-1)+(k(1)+2*k(2)+2*k(3)+k(4))/6
        t(i)=a+i*h

        WRITE(2,8)i,t(i),f(t(i)),w(i)
    END DO

    DO i=4,n
        t(i)=a+i*h
        w(i)=w(i-1)+h/24*(55*df(t(i-1),w(i-1))-59*df(t(i-2),w(i-2))+37*df(t(i-3),w(i-3))-9*df(t(i-4),w(i-4)))
        WRITE(2,8)i,t(i),f(t(i)),w(i),ABS(f(t(i))-w(i))
    END DO

    8 FORMAT(i4,3x,f6.2,10(3x,f12.8))
END PROGRAM

FUNCTION f(x)
    IMPLICIT NONE
    REAL::f,x
    f=(1+x)**2-0.5*EXP(x)
END FUNCTION

FUNCTION df(x,y)
    IMPLICIT NONE
    REAL::df,x,y
    df=y-x**2+1
END FUNCTION
```

Program –(p_c _4 implicit)

```
PROGRAM AM4Imp
    IMPLICIT NONE

    INTEGER ::i,n
    REAL::a,b,h,f,df,x,alpha,k(4),t(0:1000),w(0:1000)
    READ(1,*)a,b,alpha,n
    h=(b-a)/n
    t(0)=a
    w(0)=alpha
    WRITE(2,*)"Step      x          y          AM step 4         error"
    i=0
    WRITE(2,8)i,t(i),f(t(i)),w(i)

    DO i=1,3
```

```
        k(1)=h*df(t(i-1),w(i-1))
        k(2)=h*df(t(i-1)+h/2,w(i-1)+k(1)/2)
        k(3)=h*df(t(i-1)+h/2,w(i-1)+k(2)/2)
        k(4)=h*df(t(i-1)+h,w(i-1)+k(3))

        w(i)=w(i-1)+(k(1)+2*k(2)+2*k(3)+k(4))/6
        t(i)=a+i*h

        WRITE(2,8)i,t(i),f(t(i)),w(i)
    END DO

    DO i=4,n
        t(i)=a+i*h
        w(i)=(w(i-1)+h/720*(251*(1-(t(i))**2)+646*df(t(i-1),w(i-1))-264*df(t(i-2),w(i-2)) &
        & +106*df(t(i-3),w(i-3))-19*df(t(i-4),w(i-4))))/(1-251*h/720)
        WRITE(2,8)i,t(i),f(t(i)),w(i),ABS(f(t(i))-w(i))
    END DO

    8 FORMAT(i4,3x,f6.2,10(3x,f12.8))
END PROGRAM

FUNCTION f(x)
    IMPLICIT NONE
    REAL::f,x
    f=(1+x)**2-0.5*EXP(x)
END FUNCTION

FUNCTION df(x,y)
    IMPLICIT NONE
    REAL::df,x,y
    df=y-x**2+1
END FUNCTION
```

Program -(Linear Shotting Method   )

```
Program Linear_Shotting_Method

implicit none

integer::i,j,n

real ::f,g,x,k1,k2,k3,k4,l1,l2,l3,l4,a,b,u,u1,v,v1,al,be,y(100),yy(100),w,h,gg,ff,e
```

```
read*,a,b,al,be,n
h=(b-a)/n
u=al
u1=0
v=0
v1=be
x=a
y(1)=u
yy(1)=v

do i=2,n+1

    x=x+h

    k1=h*f(x,u,u1)
    l1=h*(g(x,u,u1))

    k2=h*(f(x+h/2.0,u+k1/2.0,u1+l1/2.0))
    l2=h*(g(x+h/2.0,u+k1/2.0,u1+l1/2.0))

    k3=h*f(x+h/2.0,u+k2/2.0,u1+l2/2.0)
    l3=h*g(x+h/2.0,u+k2/2.0,u1+l2/2.0)

    k4=h*f(x+h,u+k3,u1+l3)
    l4=h*g(x+h,u+k3,u1+l3)

    u=u+(k1+2.0*k2+2.0*k3+k4)/6
    u1=u1+(l1+2.0*l2+2.0*l3+l4)/6
    y(i)=u
    k1=h*ff(x,v,v1)
    l1=h*gg(x,v,v1)

    k2=h*ff(x+h/2.0,v+k1/2.0,v1+l1/2.0)
    l2=h*gg(x+h/2.0,v+k1/2.0,v1+l1/2.0)

    k3=h*ff(x+h/2.0,v+k2/2.0,v1+l2/2.0)
    l3=h*gg(x+h/2.0,v+k2/2.0,v1+l2/2.0)

    k4=h*ff(x+h,v+k3,v1+l3)
    l4=h*gg(x+h,v+k3,v1+l3)
    v=v+(k1+2.0*k2+2.0*k3+k4)/6
    v1=v1+(l1+2.0*l2+2.0*l3+l4)/6
    yy(i)=v



End do
x=a
write(*,11)
11 format('iteration',3x,'x',10x,'Y1',15x,"y2",13x,'y',12x,'Exact',10x,'Error')
do i=1,n+1
   w=y(i)+(be-y(n+1))*yy(i)/yy(n+1)

    write(*,10)i,x,y(i),yy(i),w,e(x),abs(w-e(x))
    10 format(i3,7x,f5.2,5(3x,f12.8))
    x=x+h
End do



stop
end
```

```
    real function f(x,u,u1)
       f=u1
    End function

    real function g(x,u,u1)
       g=(-2.0*u1)/x+2.0*u/(x**2)+sin(log(x))/(x**2)
    End function

    real function ff(x,v,v1)
       ff=v1
    End function

    real function gg(x,v,v1)
       gg=(-2.0*v1)/x+2.0*v/(x**2)
    End function

    Real function e(x)
    e=1.139207*x-0.039207/(x**2)-(3/10.0)*Sin(log(x))-(1/10.0)*cos(log(x))
End function
```