BISECTION METHOD

```
implicit none
integer::n,i
real::a,b,c,f,x,f0,t,f1,f2,f3,e
f(x)=x**3+x**2+x+7
11 write(*,*)"Enter Initial point a="
read*,a
write(*,*)"Enter End point b="
read*,b
f1=f(a)
f2=f(b)
if(f1*f2>0)then
    write(*,10)
    10 format('No Root Found in the Interval'/'Enter New Interval')
    goto 11
end if
write(*,*)"Enter Iteration n="
read*,n
write(*,*)"Enter Tolerance t="
read*,t
write(*,12)
12 format('Iteration',8x,'a',14x,'b',14x,'x',12x,'f(x)',10x,'Error')
do i=1,n
    c=(a+b)/2.0
    f0=f(c)
    e=abs(b-a)
    write(*,13)i,a,b,c,f0,e
    13 format(2x,i3,6x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6)
    if(abs(b-a)<t)then
        write(*,14)c,i
        14 format("Approximate Root=",f10.6/"Number of Iteration=",i5)
        stop
    end if
    f3=f(c)
    if(f2*f3<0.0)then
        a=c
        f1=f3
    else
        b=c
        f2=f3
    end if
    if(i==n)then
        write(*,15)
        15 format('Maximum Number of Iteration Exceeded')
    end if
end do
stop
end
```

Example

```
Enter Initial point a=
-3
Enter End point b=
-2
Enter Iteration n=
12
Enter Tolerance t=
0.001
Iteration        a              b              x            f(x)          Error
    1       -3.000000      -2.000000      -2.500000      -4.875000       1.000000
    2       -2.500000      -2.000000      -2.250000      -1.578125       0.500000
    3       -2.250000      -2.000000      -2.125000      -0.205078       0.250000
    4       -2.125000      -2.000000      -2.062500       0.417725       0.125000
    5       -2.125000      -2.062500      -2.093750       0.111481       0.062500
    6       -2.125000      -2.093750      -2.109375      -0.045498       0.031250
    7       -2.109375      -2.093750      -2.101562       0.033316       0.015625
    8       -2.109375      -2.101562      -2.105469      -0.006010       0.007812
    9       -2.105469      -2.101562      -2.103516       0.013673       0.003906
   10       -2.105469      -2.103516      -2.104492       0.003837       0.001953
   11       -2.105469      -2.104492      -2.104980      -0.001085       0.000977
Approximate Root= -2.104980
Number of Iteration=   11
```

FIXED POINT

```
implicit none
integer::n,i
real::a,c,t,f,g,x,f0,g0,e
f(x)=(cos(x)+3)/2
g(x)=(-sin(x))/2
write(*,*)"Enter Initial Approximation="
read*,a


if(abs(g(a))>1.0)then
    write(*,*)"Method is divergent"
    stop
    end if
write(*,*)"Enter Iteration n="
read*,n
write(*,*)"Enter Tolerance t="
read*,t
write(*,12)
12 format('Iteration',8x,'x',14x,'x1',12x,'f(x)',10x,'Error')
do i=1,n
    c=f(a)
    f0=f(c)
    e=abs(c-a)
    write(*,13)i,a,c,f0,e
    13 format(2x,i3,6x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6)
    if(abs(c-a)<t)then
        write(*,14)c,i
        14 format("Approximate Root=",f10.6/"Number of Iteration=",i5)
        stop
    end if
    a=c
    if(i==n)then
        write(*,15)
        15 format('Maximum Number of Iteration Exceeded')
    end if
end do
stop
end
```

Example

```
Enter Initial Approximation=
1.5
Enter Iteration n=
12
Enter Tolerance t=
0.001
Iteration        x              x1            f(x)          Error
    1        1.500000       1.535369       1.517710       0.035369
    2        1.535369       1.517710       1.526531       0.017658
    3        1.517710       1.526531       1.522126       0.008820
    4        1.526531       1.522126       1.524326       0.004405
    5        1.522126       1.524326       1.523227       0.002200
    6        1.524326       1.523227       1.523776       0.001099
    7        1.523227       1.523776       1.523502       0.000549
Approximate Root=  1.523776
Number of Iteration=    7
```

FALSE POSITION

```
implicit none
integer::n,i
real::a,b,c,f,x,p,f0,t,f1,f2,f3,e
f(x)=exp(x)-10
11 write(*,*)"Enter Initial point a="
read*,a
write(*,*)"Enter End point b="
read*,b
f1=f(a)
f2=f(b)
if(f1*f2>0)then
    write(*,10)
    10 format('No Root Found in the Interval'/'Enter New Interval')
    goto 11
end if
write(*,*)"Enter Iteration n="
read*,n
write(*,*)"Enter Tolerance t="
read*,t
write(*,12)
12 format('Iteration',8x,'a',14x,'b',14x,'x',12x,'f(x)',10x,'Error')
p=a
do i=1,n
    c=(a*f2-b*f1)/(f2-f1)
    f0=f(c)
    e=abs(b-a)
    write(*,13)i,a,b,c,f0,e
    13 format(2x,i3,6x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6)
    if(abs(c-p)<t)then
        write(*,14)c,i
        14 format("Approximate Root=",f10.6/"Number of Iteration=",i5)
        stop
    end if
       f3=f(c)
    if(f2*f3<0.0)then
        a=c
        f1=f3
        p=a
    else
        b=c
        f2=f3
        p=a
    end if
    if(i==n)then
        write(*,15)
        15 format('Maximum Number of Iteration Exceeded')
    end if
end do
stop
end
```

Example

```
Enter Initial point a=
2
Enter End point b=
3
Enter Iteration n=
12
Enter Tolerance t=
0.0001
Iteration        a              b              x            f(x)          Error
    1         2.000000       3.000000       2.205643      -0.923913       1.000000
    2         2.205643       3.000000       2.272305      -0.298258       0.794357
    3        2.272305       3.000000       2.293207      -0.093339       0.727695
    4        2.293207       3.000000       2.299689      -0.028923       0.706793
    5        2.299689       3.000000       2.301691      -0.008936       0.700311
    6        2.301691       3.000000       2.302309      -0.002758       0.698309
    7        2.302309       3.000000       2.302500      -0.000851       0.697691
    8        2.302500       3.000000       2.302559      -0.000262       0.697500
Approximate Root=  2.302559
Number of Iteration=    8
```

NEWTON RAPHSON

```
implicit none
integer::n,i
real::a,c,t,f,g,x,f0,g0,e
f(x)=x**3-x**2-x+1
g(x)=3*x**2-2*x-1
11 write(*,*)"Enter Initial Approximation="
read*,a
if(g(a)==0.0)then
    write(*,10)
    10 format('No Root Found at This Point'/'Enter New Approximation')
    goto 11
end if
write(*,*)"Enter Iteration n="
read*,n
write(*,*)"Enter Tolerance t="
read*,t
write(*,12)
12 format('Iteration',8x,'x',14x,'x1',12x,'f(x)',10x,"f'(x)",10x,'Error')
do i=1,n
    c=a-f(a)/g(a)
    f0=f(a)
    g0=g(a)
    e=abs(c-a)
    write(*,13)i,a,c,f0,g0,e
    13 format(2x,i3,6x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6,5x,f10.6)
    if(abs(c-a)<t)then
        write(*,14)c,i
        14 format("Approximate Root=",f10.6/"Number of Iteration=",i5)
        stop
    end if
    a=c
    if(i==n)then
        write(*,15)
        15 format('Maximum Number of Iteration Exceeded')
end if
end do
stop
end
```

Example

```
Enter Initial Approximation=
0.8
Enter Iteration n=
10
Enter Tolerance t=
0.001
Iteration        x              x1            f(x)          f'(x)          Error
    1        0.800000       0.905882       0.072000      -0.680000       0.105882
    2        0.905882       0.954133       0.016883      -0.349896       0.048250
    3        0.954133       0.977338       0.004111      -0.177158       0.023206
    4        0.977338       0.988734       0.001015      -0.089105       0.011396
    5        0.988734       0.994384       0.000252      -0.044682       0.005649
    6        0.994384       0.997194       0.000063      -0.022371       0.002811
    7        0.997194       0.998600       0.000016      -0.011198       0.001405
    8        0.998600       0.999303       0.000004      -0.005595       0.000703
Approximate Root=  0.999303
Number of Iteration=    8
```

NEWTON FORWARD INTERPOLATION

```
implicit none
real::x,y,t,h,u,fact,sm,prod
integer::n,i,j,k
dimension x(20),y(20,20)
n=6
data(x(i),i=1,6)/0.1,0.2,0.3,0.4,0.5,0.6/
data(y(i,1),i=1,6)/1.0596,1.6098,2.2804,2.912,3.5024,4.1139/
t=0.25
do i=2,n
    do j=1,n-i+1
        y(j,i)=y(j+1,i-1)-y(j,i-1)
    end do
end do
print*,"NEWTON'S FORWARD DIFFERENCE TABLE:"
write(*,30)
30 format(2x,"x",9x,"y",10x,"y1",12x,"y2",12x,"y3",10x,"y4",10x,"y5")
do i=1,n
    write(*,10)x(i),y(i,1),(y(i,j),j=2,n-i+1)
    10 format(f5.2,f10.4,5(3x,f10.4))
end do
h=x(2)-x(1)
u=(t-x(1))/h
sm=y(1,1)
do i=1,n
    prod=1.0
    fact=1.0
    do j=0,i-1
        prod=prod*(u-j)
    end do
    do k=1,i
        fact=fact*k
    end do
    sm=sm+(y(1,i+1)*prod)/fact
end do
write(*,20)t,sm
20 format("Interpolation of",f5.2,3x,"is",f10.4)
stop
end
```

Example

```
NEWTON'S FORWARD DIFFERENCE TABLE:
  x         y          y1            y2            y3          y4          y5
 0.10    1.0596       0.5502       0.1204      -0.1594       0.1572      -0.0927
 0.20    1.6098       0.6706      -0.0390      -0.0022       0.0645
 0.30    2.2804       0.6316      -0.0412       0.0623
 0.40    2.9120       0.5904       0.0211
 0.50    3.5024       0.6115
 0.60    4.1139
Interpolation of 0.25   is    1.9448
```

NEWTON BACKWARD INTERPOLATION

```
implicit none
real::x,y,t,h,u,fact,sm,prod
integer::n,i,j,k
dimension x(20),y(20,20)
n=6
data(x(i),i=1,6)/0.1,0.15,0.20,0.25,0.30,0.35/
data(y(i,1),i=1,6)/0.5053,1.0698,1.5804,2.0012,2.5024,3.1139/
t=0.33
do i=2,n
    do j=1,n-i+1
        y(j,i)=y(j+1,i-1)-y(j,i-1)
    end do
end do
print*,"NEWTON'S BACKWARD DIFFERENCE TABLE:"
write(*,30)
30 format(2x,"x",9x,"y",10x,"y1",12x,"y2",12x,"y3",10x,"y4",10x,"y5")
do i=1,n
    write(*,10)x(i),y(i,1),(y(i,j),j=2,n-i+1)
    10 format(f5.2,f10.4,5(3x,f10.4))
end do
h=x(2)-x(1)
u=(t-x(n))/h
sm=y(n,1)
do i=1,n-1
    prod=1.0
    fact=1.0
    do j=0,i-1
        prod=prod*(u+j)
    end do
    do k=1,i
        fact=fact*k
    end do
    sm=sm+(y(n-i,i+1)*prod)/fact
end do
write(*,20)t,sm
20 format("Interpolation of",f5.2,3x,"is",f10.4)
stop
end
```

Example

```
NEWTON'S BACKWARD DIFFERENCE TABLE:
  x         y          y1            y2            y3          y4          y5
 0.10    0.5053       0.5645      -0.0539      -0.0359       0.2061      -0.3464
 0.15    1.0698       0.5106      -0.0898       0.1702      -0.1403
 0.20    1.5804       0.4208       0.0804       0.0299
 0.25    2.0012       0.5012       0.1103
 0.30    2.5024       0.6115
 0.35    3.1139
Interpolation of 0.33   is    2.8704
```

LAGRANGE INTERPOLATION

```
implicit none
real::x,y,t,sm,prod
integer::n,i,j
dimension x(20),y(20,20)
n=6
data(x(i),i=1,6)/1.0,1.5,2.0,2.5,3.0,3.5/
data(y(i,1),i=1,6)/10.4560,11.9565,13.5571,15.2578,17.0586,18.9595/
t=2.85
print*,"LAGRANGE'S INTERPOLATION TABLE:"
write(*,30)
30 format(2x,"x",10x,"y")
do i=1,n
    write(*,10)x(i),y(i,1)
    10 format(f5.2,3x,f10.4)
end do
sm=0.0
```

```
do i=1,n
    prod=1.0
    do j=1,n
        if(j.ne.i)then
            prod=prod*(t-x(j))/(x(i)-x(j))
        end if
    end do
    sm=sm+prod*y(i,1)
end do
write(*,20)t,sm
20 format("Interpolation of",f6.2,3x,"is",f10.4)
stop
end
```

Example

```
LAGRANGE'S INTERPOLATION TABLE:
  x          y
 1.00      10.4560
 1.50      11.9565
 2.00      13.5571
 2.50      15.2578
 3.00      17.0586
 3.50      18.9595
Interpolation of  2.85   is   16.5079
```

DIVIDED DIFFERENCE

```
implicit none
real::x,y,t,h,u,sm,prod
integer::n,i,j,k,l
dimension x(20),y(20,20)
n=5
t=1.35
data(x(i),i=1,5)/1.1,1.2,1.3,1.4,1.5/
data(y(i,1),i=1,5)/1.1234,2.2241,3.4257,4.7284,6.1299/
k=n
l=1
do i=1,n
    do j=1,k-1
        y(j,i+1)=(y(j+1,i)-y(j,i))/(x(j+l)-x(j))
    end do
    l=l+1
    k=k-1
end do
print*,"NEWTON'S DIVIDED DIFFERENCE TABLE:"
write(*,30)
30 format(2x,"x",11x,"y",11x,"y1",12x,"y2",12x,"y3",10x,"y4")
do i=1,n
    write(*,10)x(i),y(i,1),(y(i,j),j=2,n-i+1)
    10 format(f5.2,3X,f10.4,4(3x,f10.4))
end do
sm=y(1,1)
do i=1,n
    prod=1.0
    do j=1,i
    prod=prod*(t-x(j))
    end do
        sm=sm+y(1,i+1)*prod
end do
write(*,20)t,sm
20 format("Interpolation of",f6.2,3x,"is",f10.4)
stop
end
```

Example

```
NEWTON'S DIVIDED DIFFERENCE TABLE:
  x           y           y1           y2           y3          y4
 1.10       1.1234      11.0070       5.0451       0.0330      -1.0406
 1.20       2.2241      12.0160       5.0549      -0.3833
 1.30       3.4257      13.0270       4.9400
 1.40       4.7284      14.0150
 1.50       6.1299
Interpolation of  1.35   is    4.0645
```

Jacobi Iteration

```
program Jacobi
    implicit none
    real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2
    integer::n,ite,h,i,j,k,r,s,t
    write(*,*)"Enter total row, iteration & tolerance:"
    read(*,*)n,ite,tol
    write(*,*)"Enter the matrix"
    read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n)
    write(*,*)"Enter initial condition"
    read*,(x0(i),i=1,n)
    write(6,10)
    10 format("iteration",10x,"x1",10x,"x2",15x,"x3",13x,"x4")
    do k=1,ite
        do i=1,n
            s1=0
            do j=1,n
                if(j.ne.i)s1=s1+a(i,j)*x0(j)
            end do
            x(i)=(b(i)-s1)/a(i,i)
        end do
        write(6,15)k,(x(h),h=1,n)
    15  format(2x,i2,1x,4(2x,f14.6))
        norm1=abs(x0(1)-x(1))
        norm2=abs(x(1))
        do r=2,n
            if(norm1<abs(x0(r)-x(r)))norm1=abs(x0(r)-x(r))
            if(norm2<abs(x(r)))norm2=abs(x(r))
        end do
        if((norm1/norm2)<tol)then
            write(6,*)'solution of the system:'
            do t=1,n
                write(6,20)t,x(t)
            end do
        20  format(2x,"x",i1,"=",f8.4)
        stop
        else
            do s=1,n
                x0(s)=x(s)
            end do
        end if
    end do
end program
```

Example

```
Enter total row, iteration & tolerance:
4 10 0.001
 Enter the matrix
10 -1 2 0
-1 11 -1 3
2 -1 10 -1
```

```
0 3 -1 8
6 25 -11 15
 Enter initial condition
0 0 0 0
iteration          x1          x2                x3              x4
   1         0.600000        2.272727        -1.100000         1.875000
   2         1.047273        1.715909        -0.805227         0.885227
   3         0.932636        2.053306        -1.049341         1.130881
   4         1.015199        1.953696        -0.968109         0.973843
   5         0.988991        2.011415        -1.010286         1.021351
   6         1.003199        1.992241        -0.994522         0.994434
   7         0.998129        2.002307        -1.001972         1.003594
   8         1.000625        1.998670        -0.999036         0.998888
   9         0.999674        2.000448        -1.000369         1.000619
 solution of the system:
  x1=  0.9997
  x2=  2.0004
  x3= -1.0004
  x4=  1.0006
```

Gauss Seidal iteration

```
program gauss_seidal
    implicit none
    real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2
    integer::n,ite,h,i,j,k,r,s,t
    write(*,*)"Enter total row, iteration & tolerance:"
    read(*,*)n,ite,tol
    write(*,*)"Enter the matrix"
    read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n)
    write(*,*)"Enter initial condition"
    read*,(x0(i),i=1,n)
    write(6,10)
    10 format("iteration",10x,"x1",10x,"x2",15x,"x3",13x,"x4")
    do k=1,ite
        do i=1,n
            s1=0
            do j=1,n
                if(i<j)s1=s1+a(i,j)*x0(j)
                if(i>j)s1=s1+a(i,j)*x(j)
            end do
            x(i)=(b(i)-s1)/a(i,i)
        end do
        write(6,15)k,(x(h),h=1,n)
    15  format(2x,i2,1x,4(2x,f14.6))
        norm1=abs(x0(1)-x(1))
        norm2=abs(x(1))
        do r=2,n
            if(norm1<abs(x0(r)-x(r)))norm1=abs(x0(r)-x(r))
            if(norm2<abs(x(r)))norm2=abs(x(r))
        end do
        if((norm1/norm2)<tol)then
            write(6,*)'solution of the system:'
            do t=1,n
                write(6,20)t,x(t)
            end do
        20  format(2x,"x",i1,"=",f8.4)
        stop
        else
            do s=1,n
                x0(s)=x(s)
            end do
        end if
    end do
end program
```

Example

```
Enter total row, iteration & tolerance:
4
10
0.001
 Enter the matrix
10 -1 2 0
-1 11 -1 3
2 -1 10 -1
0 3 -1 8
6 25 -11 15
 Enter initial condition
0 0 0 0
iteration          x1          x2               x3             x4
   1         0.600000        2.327273       -0.987273        0.878864
   2         1.030182        2.036938       -1.014456        0.984341
   3         1.006585        2.003555       -1.002527        0.998351
   4         1.000861        2.000298       -1.000307        0.999850
   5         1.000091        2.000021       -1.000031        0.999988
 solution of the system:
  x1=  1.0001
  x2=  2.0000
  x3= -1.0000
  x4=  1.0000
```

SOR iteration

```
program sor
    implicit none
    real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2
    integer::n,ite,h,i,j,k,r,s,t,w
    write(*,*)"Enter total row, iteration & tolerance:"
    read(*,*)n,ite,tol
    write(*,*)"Enter the matrix"
    read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n)
    write(*,*)"Enter initial condition"
    read*,(x0(i),i=1,n)
    write(6,10)
    10 format("iteration",10x,"x1",10x,"x2",15x,"x3",13x,"x4")
    w=1.1
    do k=1,ite
        do i=1,n
            s1=0
            do j=1,n
                if(i<j)s1=s1+a(i,j)*x0(j)
                if(i>j)s1=s1+a(i,j)*x(j)
            end do
            x(i)=(1-w)*x0(i)+(w*(b(i)-s1))/a(i,i)
        end do
        write(6,15)k,(x(h),h=1,n)
    15  format(2x,i2,1x,4(2x,f14.6))
        norm1=abs(x0(1)-x(1))
        norm2=abs(x(1))
        do r=2,n
            if(norm1<abs(x0(r)-x(r)))norm1=abs(x0(r)-x(r))
            if(norm2<abs(x(r)))norm2=abs(x(r))
        end do
        if((norm1/norm2)<tol)then
            write(6,*)'solution of the system:'
            do t=1,n
                write(6,20)t,x(t)
            end do
        20  format(2x,"x",i1,"=",f8.4)
        stop
        else
            do s=1,n
                x0(s)=x(s)
            end do
        end if
    end do
end program
```

Example

```
Enter total row, iteration & tolerance:
4
10
0.001
 Enter the matrix
10 -1 2 0
-1 11 -1 3
2 -1 10 -1
0 3 -1 8
6 25 -11 15
 Enter initial condition
0 0 0 0
iteration          x1          x2               x3             x4
   1         0.600000        2.327273       -0.987273        0.878864
   2         1.030182        2.036938       -1.014456        0.984341
   3         1.006585        2.003555       -1.002527        0.998351
   4         1.000861        2.000298       -1.000307        0.999850
   5         1.000091        2.000021       -1.000031        0.999988
 solution of the system:
  x1=  1.0001
  x2=  2.0000
  x3= -1.0000
  x4=  1.0000
```

Trapezoidal

```
program Trapizoidal
f(x)=1/(1.0+x)
g(x)=alog(1.0+x)
write(*,*)"Enter the value of a,b,n"
read(*,*)a,b,n
h=(b-a)/n
s=0.0
do i=1,n-1
    s=s+f(a+i*h)
end do
v=(h/2.0)*(f(a)+f(b)+2*s)
write(*,*)"Enter the approximate value"
write(*,*)v
exactvalue=g(b)-g(a)
write(*,*)"Exact value",exactvalue
err=exactvalue-v
```

```
write(*,*)"error",err
stop
end program
```

Example

```
Enter the value of a,b,n
1
3
12
 Enter the approximate value
  0.693580806
 Exact value  0.693147182
 error  -4.33623791E-04
```

Simpson 1/3

```
program simpson13
f(x)=1/(1.0+x)
g(x)=alog(1.0+x)
write(*,*)"Enter the value of a,b,n"
read(*,*)a,b,n
h=(b-a)/n
s=0.0
do i=1,n-1
    if(mod(i,2).eq.0)then
        s=s+4.0*f(a+i*h)
    end if
end do
v=(h/3.0)*(f(a)+f(b)+s)
write(*,*)"Enter the approximate value"
write(*,*)v
exactvalue=g(b)-g(a)
write(*,*)"Exact value",exactvalue
err=exactvalue-v
write(*,*)"error",err
stop
end program
```

Example

```
Enter the value of a,b,n
1
3
12
 Enter the approximate value
  0.421584874
 Exact value  0.693147182
 error  0.271562308
```

Simpson 3/8

```
program simpson38
f(x)=1/(1.0+x)
g(x)=alog(1.0+x)
```

```
write(*,*)"Enter the value of a,b,n"
read(*,*)a,b,n
h=(b-a)/n
s=0.0
do i=1,n-1
    if(mod(i,3).eq.0)then
        s=s+2*f(a+i*h)
    else
        s=s+3*f(a+i*h)
    end if
end do
v=(3.0*h/8.0)*(f(a)+f(b)+s)
write(*,*)"Enter the approximate value"
write(*,*)v
exactvalue=g(b)-g(a)
write(*,*)"Exact value",exactvalue
err=exactvalue-v
write(*,*)"error",err
stop
end program
```

Example

```
Enter the value of a,b,n
1
3
12
 Enter the approximate value
  0.693150401
 Exact value  0.693147182
 error  -3.21865082E-06
```

Weddle

```
program weddle
f(x)=1.0/(1.0+x)
g(x)=alog(1.0+x)
write(*,*)"Enter the value of a,b,n"
read(*,*)a,b,n
h=(b-a)/n
s=0.0
do i=1,n-1
    if(mod(i,6)==1)then
        s=s+5*f(a+i*h)
    else if(mod(i,6)==5)then
        s=s+5*f(a+i*h)
    else if(mod(i,6)==3)then
        s=s+6*f(a+i*h)
    else if(mod(i,6)==0)then
        s=s+2*f(a+i*h)
    else
        s=s+f(a+i*h)
    end if
end do
v=(3.0*h/10.0)*(f(a)+f(b)+s)
write(*,*)"Enter the approximate value"
write(*,*)v
exactvalue=g(b)-g(a)
write(*,*)"Exact value",exactvalue
err=exactvalue-v
```

```
write(*,*)"error",err
stop
end program
```

Example

```
Enter the value of a,b,n
1
3
12
 Enter the approximate value
  0.693147242
 Exact value  0.693147182
 error  -5.96046448E-08
```

Romberg Integration

```
program romberg
implicit none
integer::n,i,j,k,l
real::a,b,tol,h(20),r(20,20),f,g,sum1,er
write(*,*)"Enter interval"
read*,a,b
write(*,*)"Enter iteration & tolerance"
read*,n,tol
do i=1,n
    h(i)=(b-a)/2**(i-1)
end do
r(1,1)=(h(1)/2)*(f(a)+f(b))
write(*,*)"    O(h2)    ","     O(h4)     ","     O(h6)    ","    O(h8)   "
write(6,13)r(1,1)
13 format(2x,f12.8)
do i=2,n
    sum1=0.0
    do j=1,2**(i-2)
        sum1=sum1+f(a+(2*j-1)*h(i))
    end do
     r(i,1)=0.5*(r(i-1,1)+h(i-1)*sum1)
     do k=2,i
        r(i,k)=r(i,k-1)+(r(i,k-1)-r(i-1,k-1))/(4**(k-1)-1)
     end do
     write(6,14)(r(i,k),k=1,i)
     14 format(20(2x,f12.8))
     if(abs(r(i,i)-r(i-1,i-1))<tol)then
        write(6,*)"Value of the Itegration=",r(i,i)
        er=abs((g(b)-g(a))-r(i,i))
        write(6,15)er
        15 format(2x,"Error=",f12.8)
        stop
     end if
end do
end program
real function f(x)
f=1/(1+x**2)
return
end
real function g(x)
g=atan(x)
return
end
```

Example

```
Enter interval
1  3
 Enter iteration & tolerance
6   0.0001
     O(h2)         O(h4)          O(h6)        O(h8)
    0.60000002
    0.50000000    0.46666667
    0.47281167    0.46374890    0.46355438
    0.46593946    0.46364874    0.46364206    0.46364346
 Value of the Itegration=  0.463643461
  Error=  0.00000408
```

Gauss Elimination method with pivoting

```
PROGRAM gauss_eli_pivot
IMPLICIT NONE
REAL::A(20,20),k1,k2,v(20),c
INTEGER::i,j,n,k
PRINT *,'NO. OF ROWS'
READ(*,*)n
PRINT *,'ENTER ELEMENTS'
READ(*,*)((A(i,j),j=1,n+1),i=1,n)
PRINT *,'MATRIX - '
DO i=1,n
write(*,*)(A(i,j),j=1,n+1)
END DO
DO k=1,n-1
call pivot_sub(A,n,k)
k1=A(k,k)
   DO i=k+1,n
       k2=A(i,k)/k1
       DO j=k,n+1
         A(i,j)=A(i,j)-(k2*A(k,j))
       END DO
   END DO
END DO
PRINT *,'UPPER TRIANGULAR MATRIX - '
DO i=1,n
write(*,*)(A(i,j),j=1,n+1)
END DO
v(n)=A(n,n+1)/A(n,n)
DO i=n-1,1,-1
c=0.
DO j=i+1,n
   c=c+A(i,j)*v(j)
END DO
v(i)=(A(i,n+1)-c)/a(i,i)
END DO
PRINT *,'SOLUTIONS ARE - '
DO i=1,n
write(*,1)v(i)
1 format(f10.5)
END DO
END PROGRAM
SUBROUTINE pivot_sub(A1,n,k)
REAL::A1(20,20),big
INTEGER::i,n,k,rn
rn=k
```

```
big=abs(A1(k,k))
DO i=k+1,n
   IF((abs(A1(i,k)))>(abs(A1(k,k)))) THEN
     big=A1(i,k)
     rn=i
   END IF
  END DO
IF (rn .ne. k) THEN
DO j=1,n+1
   temp=A1(rn,j)
   A1(rn,j)=A1(k,j)
   A1(k,j)=temp
END DO
END IF
RETURN
END SUBROUTINE
```

Example

```
NO. OF ROWS
3
 ENTER ELEMENTS
1    6   10  -3
4   -10   1  -3
10   -5  -2    3
 MATRIX -
   1.00000000       6.00000000       10.0000000      -3.00000000
   4.00000000      -10.0000000       1.00000000      -3.00000000
   10.0000000      -5.00000000      -2.00000000       3.00000000
 UPPER TRIANGULAR MATRIX -
   10.0000000      -5.00000000      -2.00000000       3.00000000
   0.00000000      -8.00000000       1.79999995      -4.19999981
   0.00000000       0.00000000       11.6624994      -6.71249962
 SOLUTIONS ARE -
   0.38264
   0.39550
  -0.57556
```

Gauss Elimination method without pivoting

```
PROGRAM gauss_eli_without_pivot
IMPLICIT NONE
REAL::A(20,20),k1,k2,v(20),c
INTEGER::i,j,n,k
PRINT *,'NO. OF ROWS'
READ(*,*)n
PRINT *,'ENTER ELEMENTS'
READ(*,*)((A(i,j),j=1,n+1),i=1,n)
PRINT *,'MATRIX - '
DO i=1,n
  write(*,*)(A(i,j),j=1,n+1)
END DO
DO k=1,n-1
  k1=A(k,k)
    DO i=k+1,n
        k2=A(i,k)/k1
        DO j=k,n+1
          A(i,j)=A(i,j)-(k2*A(k,j))
        END DO
    END DO
END DO
PRINT *,'UPPER TRIANGULAR MATRIX - '
DO i=1,n
  write(*,*)(A(i,j),j=1,n+1)
END DO
v(n)=A(n,n+1)/A(n,n)
DO i=n-1,1,-1
c=0.
  DO j=i+1,n
    c=c+A(i,j)*v(j)
  END DO
v(i)=(A(i,n+1)-c)/a(i,i)
END DO
PRINT *,'SOLUTIONS ARE - '
DO i=1,n
  write(*,*)v(i)
END DO
END PROGRAM
```

Example

```
NO. OF ROWS
3
 ENTER ELEMENTS
1    6   10  -3
4   -10   1  -3
10   -5  -2    3
 MATRIX -
   1.00000000       6.00000000       10.0000000      -3.00000000
   4.00000000      -10.0000000       1.00000000      -3.00000000
   10.0000000      -5.00000000      -2.00000000       3.00000000
 UPPER TRIANGULAR MATRIX -
   1.00000000       6.00000000       10.0000000      -3.00000000
   0.00000000      -34.0000000      -39.0000000       9.00000000
   0.00000000       0.00000000      -27.4411774       15.7941170
 SOLUTIONS ARE -
  0.382636547
  0.395498335
 -0.575562656
```

Euler Method

```
program euler
    implicit none
    INTEGER::i,n
    REAL::a,b,x,y,f,g,er,y0,h,y1,er1
    write(*,*)"Enter value of a,b,n & y"
    read(5,*)a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    write(*,13)
    13 format(3x,'x(i)',7x,'eu_y(i)',3x,'exact value',4x,'eu_error')
    do i=1,n+1
        er=ABS(g(x)-y)
        write(*,14)x,y,g(x),er
        14 format(2x,f5.2,5(5x,f8.5))
        call eu(h,x,y)
        x=a+i*h
        end do
end program
subroutine eu(h,x,y)
    implicit none
    REAL::x,y,h,f
    y=y+h*f(x,y)
    end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0  0.5  5  0.5
   x(i)       eu_y(i)   exact value    eu_error
   0.00      0.50000      0.50000      0.00000
   0.10      0.65000      0.65741      0.00741
   0.20      0.81400      0.82930      0.01530
   0.30      0.99140      1.01507      0.02367
   0.40      1.18154      1.21409      0.03255
   0.50      1.38369      1.42564      0.04195
```

Modified Euler Method

```
program modified_euler
    implicit none
    INTEGER::i,n
    REAL::a,b,x,y,f,g,er,y0,h,y1,er1
    write(*,*)"Enter value of a,b,n & y"
    read(5,*)a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    y1=y0
    write(*,13)
    13 format(3x,'x(i)',3x,'exact value',4x,'meu_y(i)',4x,'meu_error')
    do i=1,n+1
        er1=ABS(g(x)-y1)
        write(*,14)x,g(x),y1,er1
        14 format(2x,f5.2,5(5x,f8.5))
        call m_euler(h,x,y1,y)
        x=a+i*h
        end do
end program
subroutine m_euler(h,x,y1,y)
    REAL::x,y1,y,h,f
     y=y+h*f(x,y)
    y1=y1+h*(.5*(f(x,y1)+f(x+h,y)))
    end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0  0.5  5  0.5
   x(i)   exact value    meu_y(i)    meu_error
   0.00      0.50000      0.50000      0.00000
   0.10      0.65741      0.65700      0.00041
   0.20      0.82930      0.82805      0.00125
   0.30      1.01507      1.01252      0.00255
   0.40      1.21409      1.20973      0.00436
   0.50      1.42564      1.41890      0.00674
```

Euler+Modified Euler Method

```
program euler_modified_euler
    implicit none
    INTEGER::i,n
    REAL::a,b,x,y,f,g,er,y0,h,y1,er1
    write(*,*)"Enter value of a,b,n & y"
    read(5,*)a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    y1=y0
    write(*,13)
    13 format(3x,'x(i)',7x,'eu_y(i)',3x,'exact
value',4x,'eu_error',5x,'meu_y(i)',4x,'meu_error')
    do i=1,n+1
        er=ABS(g(x)-y)
        er1=ABS(g(x)-y1)
        write(*,14)x,y,g(x),er,y1,er1
        14 format(2x,f5.2,5(5x,f8.5))
        call euler(h,x,y)
        call m_euler(h,x,y1,y)
        x=a+i*h
        end do
end program
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,y)
    REAL::x,y1,y,h,f
    y1=y1+h*(.5*(f(x,y1)+f(x+h,y)))
    end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0  0.5  5  0.5
   x(i)       eu_y(i)    exact value    eu_error      meu_y(i)     meu_error
   0.00       0.50000      0.50000      0.00000      0.50000      0.00000
   0.10       0.65000      0.65741      0.00741      0.65700      0.00041
   0.20       0.81400      0.82930      0.01530      0.82805      0.00125
   0.30       0.99140      1.01507      0.02367      1.01252      0.00255
   0.40       1.18154      1.21409      0.03255      1.20973      0.00436
   0.50       1.38369      1.42564      0.04195      1.41890      0.00674
```

Adam Bashfourth 4th step explicit method

```
program adams_bashforth
    implicit none
    real::a,b,f,g,y0,er1(20),x(20),y(20),h,k1,k2,k3,k4
    integer::i,n
    write(*,*)"Enter a,b,n & y"
    read(*,*)a,b,n,y0
    h=(b-a)/n
    x(1)=a
    y(1)=y0
    er1(1)=abs(y(1)-g(x(1)))
    do i=2,4
        k1=h*f(x(i-1),y(i-1))
        k2=h*f(x(i-1)+h/2,y(i-1)+k1/2)
        k3=h*f(x(i-1)+h/2,y(i-1)+k2/2)
        k4=h*f(x(i-1)+h,y(i-1)+k3)
        y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6
        x(i)=x(i-1)+h
        er1(i)=abs(y(i)-g(x(i)))
    end do
    do i=5,n+1
        y(i)=y(i-1)+(h/24)*(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)))
        x(i)=x(i-1)+h
        er1(i)=abs(y(i)-g(x(i)))
    end do
    write(*,13)
    13 format(3x,"x(i)",9x,"y(i)",8x,"Exact Value",6x,"Error")
    do i=1,n+1
        write(*,14)x(i),y(i),g(x(i)),er1(i)
        14 format(2x,f5.2,3(5x,f10.6))
    end do
end program
real function f(x,y)
    f=y-x**2+1
    return
end
real function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
end
```

Example

```
Enter a,b,n & y
0  1  10  0.5
   x(i)         y(i)         Exact Value      Error
   0.00       0.500000       0.500000       0.000000
   0.10       0.657414       0.657415       0.000000
   0.20       0.829298       0.829299       0.000000
   0.30       1.015070       1.015070       0.000000
   0.40       1.214089       1.214087       0.000002
   0.50       1.425644       1.425639       0.000004
   0.60       1.648948       1.648941       0.000007
   0.70       1.883135       1.883124       0.000011
   0.80       2.127245       2.127230       0.000015
   0.90       2.380219       2.380199       0.000021
   1.00       2.640886       2.640859       0.000027
```

Adam moulton 4th step implicit method

```
program adams_moulton
    implicit none
    real::a,b,f,g,y0,er1(10),x(10),y(10),h,k1,k2,k3,k4
    integer::i,n
    write(*,*)"Enter a,b,n & y0"
    read*,a,b,n,y0
    h=(b-a)/n
    x(1)=a
    y(1)=y0
    er1(1)=abs(y(1)-g(x(1)))
    do i=1,n
        k1=h*f(x(i-1),y(i-1))
        k2=h*f(x(i-1)+h/2,y(i-1)+k1/2)
        k3=h*f(x(i-1)+h/2,y(i-1)+k2/2)
        k4=h*f(x(i-1)+h,y(i-1)+k3)
        y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6
        x(i)=x(i-1)+h
        er1(i)=abs(y(i)-g(x(i)))
    end do
    do i=4,n
        y(i)=y(i-1)+(h/720)*(251*f(x(i),y(i))+646*f(x(i-1),y(i-1))-264*f(x(i-
2),y(i-2))+106*f(x(i-3),y(i-3))-19*f(x(i-4),y(i-4)))
        x(i)=x(i-1)+h
        er1(i)=abs(y(i)-g(x(i)))
    end do
    write(*,13)
    13 format(3x,"x(i)",9x,"y(i)",9x,"Exact_Value",6x,"Error")
    do i=1,n
        write(*,14)x(i),y(i),g(x(i)),er1(i)
        14 format(2x,f5.2,3(5x,f10.6))
    end do
end program
real function f(x,y)
    f=y-x**2+1
    return
end
real function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
end
```

Example

```
Enter a,b,n & y0
0  1  10  0.5
   x(i)         y(i)         Exact_Value      Error
   0.10       0.657414       0.657415       0.000000
   0.20       0.829298       0.829299       0.000000
   0.30       1.015070       1.015070       0.000000
   0.40       1.214087       1.214087       0.000000
   0.50       1.425639       1.425639       0.000001
   0.60       1.648940       1.648941       0.000001
   0.70       1.883123       1.883124       0.000001
   0.80       2.127228       2.127230       0.000001
   0.90       2.380197       2.380199       0.000001
   1.00       2.640858       2.640859       0.000001
```

Adam predictor corrector method

```
program adams_predictor_corrector
    implicit none
    real::a,b,f,g,y0,er1(20),x(20),y(20),h,k1,k2,k3,k4
    integer::i,n
    write(*,*)"Enter a,b,n & y0"
    read*,a,b,n,y0
    h=(b-a)/n
    x(1)=a
    y(1)=y0
    er1(1)=abs(y(1)-g(x(1)))
    do i=2,4
        k1=h*f(x(i-1),y(i-1))
        k2=h*f(x(i-1)+h/2,y(i-1)+k1/2)
        k3=h*f(x(i-1)+h/2,y(i-1)+k2/2)
        k4=h*f(x(i-1)+h,y(i-1)+k3)
        y(i)=y(i-1)+(k1+2*k2+2*k3+k4)/6
        x(i)=x(i-1)+h
        er1(i)=abs(y(i)-g(x(i)))
    end do
    do i=5,n+1
        y(i)=y(i-1)+(h/24)*(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)))
        x(i)=x(i-1)+h
        y(i)=y(i-1)+(h/24)*(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-2),y(i-2)))
        er1(i)=abs(y(i)-g(x(i)))
    end do
    write(*,13)
    13 format(3x,"x(i)",9x,"y(i)",8x,"Exact Value",6x,"Error")
    do i=1,n+1
        write(*,14)x(i),y(i),g(x(i)),er1(i)
        14 format(2x,f5.2,3(5x,f10.6))
    end do
end program
real function f(x,y)
    f=y-x**2+1
    return
end
real function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
end
```

Example

```
Enter a,b,n & y0
0  1  10  0.5
   x(i)        y(i)        Exact Value      Error
   0.00       0.500000       0.500000       0.000000
   0.10       0.657414       0.657415       0.000000
   0.20       0.829298       0.829299       0.000000
   0.30       1.015070       1.015070       0.000000
   0.40       1.214678       1.214087       0.000591
   0.50       1.426870       1.425639       0.001230
   0.60       1.650847       1.648941       0.001906
   0.70       1.885747       1.883124       0.002623
   0.80       2.130610       2.127230       0.003380
   0.90       2.384378       2.380199       0.004179
   1.00       2.645879       2.640859       0.005020
```

RK-2

```
program RK_2
    implicit none
    real::a,b,f,g,y0,er1,x,y,h,k1,k2
    integer::i,n
    write(*,*)"Enter value of a,b,n & y"
    read*,a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    write(*,20)
    20 format(5x,"x(i)",8x,"y(i)",8x,"Exact value",7x,"Error")
    do i=1,n+1
        er1=abs(y-g(x))
        write(*,30)x,y,g(x),er1
        30 format(2x,f5.2,3(5x,f10.6))
        call RK2(h,x,y)
        x=a+i*h
    end do
end program
subroutine RK2(h,x,y)
    real::f,x,y,k1,k2,h
    k1=h*f(x,y)
    k2=h*f(x+h,y+k1)
    y=y+(k1+k2)/2
end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0  0.5  5  0.5
     x(i)        y(i)        Exact value       Error
   0.00       0.500000       0.500000       0.000000
   0.10       0.657000       0.657415       0.000415
   0.20       0.828435       0.829299       0.000864
   0.30       1.013721       1.015070       0.001350
   0.40       1.212211       1.214087       0.001876
   0.50       1.423194       1.425639       0.002446
```

RK-4

```
program RK_4
    implicit none
    real::a,b,f,g,y0,er1,x,y,h,k1,k2
    integer::i,n
    write(*,*)"Enter value of a,b,n & y"
    read*,a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    write(*,20)
    20 format(5x,"x(i)",8x,"y(i)",8x,"Exact value",7x,"Error")
    do i=1,n+1
        er1=abs(y-g(x))
        write(*,30)x,y,g(x),er1
        30 format(2x,f5.2,3(5x,f10.6))
        call RK4(h,x,y)
        x=a+i*h
    end do
end program
subroutine RK4(h,x,y)
    real::f,x,y,k1,k2,k3,k4,h
    k1=h*f(x,y)
    k2=h*f(x+h/2,y+k1/2)
    k3=h*f(x+h/2,y+k2/2)
    k4=h*f(x+h,y+k3)
    y=y+(k1+2*k2+2*k3+k4)/6
end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0
0.5
5
0.5
     x(i)        y(i)        Exact value       Error
   0.00       0.500000       0.500000       0.000000
   0.10       0.657414       0.657415       0.000000
   0.20       0.829298       0.829299       0.000000
   0.30       1.015070       1.015070       0.000000
   0.40       1.214087       1.214087       0.000000
   0.50       1.425639       1.425639       0.000001
```

RK2 & RK4

```
program RK24_method
    implicit none
    integer::i,n
    real::a,b,x,y,g,er,y0,h,y1,er1
    write(*,*)"Enter value of a,b,n & y"
    read(*,*)a,b,n,y0
    h=(b-a)/n
    x=a
    y=y0
    y1=y0
    write(6,2)
    2 format(3x,'x(i)',7x,'rk2_y(i)',3x,'Exact
value',4x,'rk2_error',5x,'rk4_y(i)',4x,'rk4_error')
    do i=1,n+1
        er=abs(g(x)-y)
        er1=abs(g(x)-y1)
        write(*,14)x,y,g(x),er,y1,er1
        14 format(2x,f5.2,5(5x,f8.5))
        call rk2(h,x,y)
        call rk4(h,x,y1)
        x=a+i*h
    end do
end program
subroutine rk2(h,x,y)
    implicit none
    real::x,y,h,f,k1,k2
    k1=h*f(x,y)
    k2=h*f(x+h,y+k1/2.0)
    y=y+1.0/6*(k1+2*k2)
end subroutine
subroutine rk4(h,x,y1)
    implicit none
    real::x,y1,h,f,k1,k2,k3,k4,y
    k1=h*f(x,y)
    k2=h*f(x+h,y+k1/2.0)
    k3=h*f(x+h,y+k2/2.0)
    k4=h*f(x+h,y+k3)
    y1=y1+1.0/6*(k1+2.0*k2+2.0*k3+k4)
end subroutine
REAL function f(x,y)
    f=y-x**2+1
    return
    end
REAL function g(x)
    g=(x+1)**2-0.5*exp(x)
    return
    end
```

Example

```
Enter value of a,b,n & y
0  0.5  5  0.5
   x(i)       rk2_y(i)   Exact value    rk2_error     rk4_y(i)    rk4_error
   0.00      0.50000      0.50000      0.00000      0.50000      0.00000
   0.10      0.57717      0.65741      0.08025      0.60430      0.05311
   0.20      0.65714      0.82930      0.17216      0.70582      0.12348
   0.30      0.73902      1.01507      0.27605      0.80245      0.21262
   0.40      0.82189      1.21409      0.39220      0.89208      0.32201
   0.50      0.90475      1.42564      0.52089      0.97262      0.45302
```

LU Decomposition

```
PROGRAM LU_Decomposition
    implicit none
    integer,parameter::n=3
    real(8)::a(n,n)
    real(8)::l(n,n)
    real(8)::u(n,n)
    integer::i,j,k
    a=reshape([1.0,2.0,3.0,&
                4.0,5.0,6.0,&
                7.0,8.0,9.0],[n,n])
    l=0.0
    u=0.0
    do i=1,n
        u(i,i)=a(i,i)
        do k=i+1,n
            u(i,k)=a(i,k)
        end do
        do k=i+1,n
            l(k,i)=a(k,i)/u(i,i)
            do j=i+1,n
                a(k,j)=a(k,j)-l(k,j)*u(i,j)
            end do
        end do
    end do
    print*,"Original matrix A"
    do i=1,n
        print*,a(i,:)
    end do
    print*,"Lower triangle matrix L"
    do i=1,n
        print*,l(i,:)
    end do
    print*,"Upper triangle matrix U"
    do i=1,n
        print*,u(i,:)
    end do
end program
```

Example

```
Original matrix A
   1.0000000000000000         4.0000000000000000         7.0000000000000000
   2.0000000000000000         5.0000000000000000         8.0000000000000000
   3.0000000000000000         6.0000000000000000         9.0000000000000000
```

```
 Lower triangle matrix L
   0.0000000000000000        0.0000000000000000        0.0000000000000000
   2.0000000000000000        0.0000000000000000        0.0000000000000000
   3.0000000000000000        1.2000000000000000        0.0000000000000000
 Upper triangle matrix U
   1.0000000000000000        4.0000000000000000        7.0000000000000000
   0.0000000000000000        5.0000000000000000        8.0000000000000000
   0.0000000000000000        0.0000000000000000        9.0000000000000000
```

Richardson

```
program richardson
    implicit none
    integer, parameter :: n = 5
    integer :: i, j
    real :: xo, h, ri(n,n), l, k, m, f, exact
    h = 0.2; xo = 2.0; k = n; m = 0
    exact = (xo+1)*exp(xo)
    do i = 1, n
        do j = 1, k
            l = 2**(j-1)
            h = 0.2
            h = h/l
            ri(j+m,i) = h
            ri(j,1) = (f(xo+h)-f(xo-h))/(2*h)
        end do
        m = m + 1
        k = k - 1
    end do
    do j = 2, n
        do i = j, n
            ri(i,j)=ri(i,j-1)+(ri(i,j-1)-ri(i-1,j-1))/(4**(j-1)-1)
        end do
    end do
    write(*,*)"            Extrapolation table for Richardson is shown 
bellow"
    write(*,*)"
=================================================="
    write(*,*)"     O(h2)     ","      O(h4)   ","      O(h6)     ","
O(h8)     ","     O(h10)     "
    write(*,*)"   --------    ","    --------  ","    --------    ","   -----
---    ","   ---------    "
    k = n
    do i= 1, k
        write(*,'(5f15.8)')(ri(i,j), j = 1, i)!For table
        k = k + 1
    end do
    write(*,'(2/, A80)')"Order of (h)       approximate value       exact 
result       relative error"
    write(*,'(I10, F25.8, F23.8, F20.8)')8, ri(n,4), exact, abs(ri(n,4)-
exact)
    write(*,'(I10, F25.8, F23.8, F20.8)')10, ri(n,n), exact, abs(ri(n,n)-
exact)
end program
function f(x)
    real :: x, f
    f = x*exp(x)
end function
```

Example

```
             Extrapolation table for Richardson is shown bellow
             ==================================================
      O(h2)           O(h4)         O(h6)          O(h8)          O(h10)
    --------        --------      --------       --------       ---------
    22.41416740
    22.22877502    22.16697693
    22.18254089    22.16712952    22.16713905
    22.17105865    22.16723061    22.16723824    22.16724014
    22.16819763    22.16724396    22.16724396    22.16724396    22.16724396


    Order of (h)       approximate value       exact result       relative error
         8              22.16724396             22.16716766          0.00007629
        10              22.16724396             22.16716766          0.00007629
```

Linear Shooting

```
program linear_shooting
    implicit none
    integer::i,n
    real::a,b,h,x,y,alpha,beta,p,q,r,w1,w2
    real,dimension(4,2)::k,kp
    real,dimension(2,0:1000)::u,v,up,vp,w
    a=1
    b=2
    alpha=1
    beta=2
    N=10
    h=(b-a)/N
    u(1,0)=alpha
    u(2,0)=0
    v(1,0)=0
    v(2,0)=1
    do i=0,N-1
        x=a+i*h
        k(1,1) = h*u(2,i)
        k(1,2) = h*(p(x)*u(2,i) + q(x)*u(1,i) + r(x))
        k(2,1) = h*(u(2,i) + k(1,2)/2)
        k(2,2) = h*(p(x+h/2)*(u(2,i)+k(1,2)/2)+q(x+h/2))*(u(1,i)+k(1,1)/2+r(x+h/2))
        k(3,1) = h*(u(2,i) + k(2,2)/2)
        k(3,2) = h*(p(x+h/2)*(u(2,i)+k(2,2)/2)+q(x+h/2))*(u(1,i)+k(2,1)/2 + r(x + h/2))
        k(4,1) = h*(u(2,i) + k(3,2))
        k(4,2) = h*(p(x + h) *(u(2,i) + k(3,2))+q(x + h))*(u(1,i) +k(3,1) + r(x + h))
        u(1,i+1) = u(1,i) + (k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1))/6
        u(2,i+1) = u(2,i) + (k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2))/6
        kp(1,1) = h*v(2,i)
        kp(1,2) = h*(p(x)*v(2,i) + q(x)*v(1,i) )
        kp(2,1) = h*(v(2,i) + kp(1,2)/2)
        kp(2,2) = h*(p(x + h/2) *(v(2,i) + kp(1,2)/2)+q(x + h/2))*(v(1,i) +kp(1,1)/2)
        kp(3,1) = h*(v(2,i) + kp(2,2)/2)
        kp(3,2) = h*(p(x + h/2) *(v(2,i) + kp(2,2)/2)+q(x + h/2))*(v(1,i) +kp(2,1)/2)
        kp(4,1) = h*(v(2,i) + kp(3,2))
        kp(4,2) = h*(p(x + h) *(v(2,i) + kp(3,2))+q(x + h))*(v(1,i) +kp(3,1))
        v(1,i+1) = v(1,i) + (kp(1,1) + 2*kp(2,1) + 2*kp(3,1) + kp(4,1))/6
        v(2,i+1) = v(2,i) + (kp(1,2) + 2*kp(2,2) + 2*kp(3,2) + kp(4,2))/6
    end do
    w(1,0)=alpha
    w(2,0)=(beta - u(1,N))/v(1,N)
    write(*,*)"     x         Exact solution  LSM Approx.     Error in LSM"
    write(*,8)a,y(a),w(1,0),abs(y(x)-w(1,0))
    do i=1,N
        W1=u(1,i)+w(2,0)*v(1,i)
        W2=u(2,i)+w(2,0)*v(2,i)
        x=a+i*h
        write(*,8) x,y(x),w1,abs(y(x)-w1)
    end do
    8 format(4(f10.5,6x))
end program
function p(x)
    real::p,x
    p=1
end function
function q(x)
    real::q,x
    q=2
end function
function r(x)
    real::r,x
    r=cos(x)
end function
function y(x)
    real::y,x
    y=-(sin(x)+3*cos(x))/10
end function
```

Example

```
      x         Exact solution  LSM Approx.     Error in LSM
   1.00000        -0.24624         1.00000         0.99764
   1.10000        -0.22520         0.91888         1.14408
   1.20000        -0.20191         0.86633         1.06825
   1.30000        -0.17661         0.84318         1.01978
   1.40000        -0.14954         0.85109         1.00062
   1.50000        -0.12097         0.89308         1.01405
   1.60000        -0.09120         0.97433         1.06552
   1.70000        -0.06051         1.10363         1.16414
   1.80000        -0.02922         1.29606         1.32528
   1.90000         0.00236         1.57825         1.57590
   2.00000         0.03391         2.00000         1.96609
```