Trapizoidal,Simpson's 1/3 & 3/8 Program

```
program numerical_integration
   f(x)=1+x**2.0
   g(x)=x+(x**3.0/3.0)
   real::a,b,h,s1,s2,s3
   integer::n,i,k,j
   write(*,*)'enter the value of the limit a & b ='
   read(*,*)a,b
   write(*,*)'entre the number you want to divide the interval ='
   read(*,*)n
   h=(b-a)/Float(n)
   !trapezoidal part
   s1=f(a)+f(b)
   do i=1,n-1
      s1=s1+2.0*f(a+float(i)*h)
   end do
   trap_ar=(h/2.0)*s1
   ev=g(b)-g(a)
   trap_er=abs(ev-trap_ar)
   write(*,*)
   write(*,10)trap_ar,trap_er
   10 format(2x,'integral value by trapezoidal rule=',f20.5,10x,'Error is=',f20.5)
   write(*,*)
    !simpson 1/3 rules
    s2=f(a)+f(b)
    if(mod(n,2).ne.0)then
      write(*,*)'simpsons 1/3 rule is not applicable'
      stop
      else
      do k=1,n-1
```

```
    if(mod(k,2).eq.0)then
      s2=s2+2.0*f(a+k*h)
      else
          s2=s2+4.0*f(a+k*h)
    end if
    end do
       sim_ar=s2*(h/3.0)
       sim_er=abs(ev-sim_ar)
          write(*,100)sim_ar,sim_er
    100 format(2x,'integral value by simpsons 1/3 rule=',f20.5,10x,'Error is=',f20.5)
    end if
       write(*,*)
    !simpson 3/8 rule
      s3=f(a)+f(b)
     if(mod(n,3).ne.0)then
       write(*,*)'simpsons 3/8 rule is not applicable'
       stop
       else
       do j=1,n-1
    if(mod(j,3).eq.0)then
      s3=s3+2.0*f(a+j*h)
      else
          s3=s3+3.0*f(a+j*h)
    end if
    end do
       sim_ar2=s3*(3.0*h/8.0)
       sim_er2=abs(ev-sim_ar2)
          write(*,200)sim_ar2,sim_er2
    200 format(2x,'integral value by simpsons 3/8 rule=',f20.5,10x,'Error is=',f20.5)
    end if
end program
```

Weddel's Program

```
program weddels
   f(x)=x**2.0+1.0
   g(x)=x+(x**3.0/3.0)
   real::a,b,ev,s,h
   integer::i,j,k,l,m,n,num
    write(*,*)'enter the value of the limit a & b ='
   read(*,*)a,b
   write(*,*)'entre the number you want to divide the interval ='
   read(*,*)num
   if (mod(num, 6) /= 0) then
   write(*,*) "Error: n must be a multiple of 6."
   stop
  end if
   h=(b-a)/Float(num)
   s=f(a)+f(b)
   do i=1,num-1,6
      s=s+5.0*f(a+i*h)
   end do
    do j=5,num-1,6
      s=s+5.0*f(a+j*h)
   end do
    do k=2,num-1,6
      s=s+f(a+k*h)
   end do
    do l=4,num-1,6
      s=s+f(a+l*h)
   end do
    do m=3,num-1,6
```

```
      s=s+6.0*f(a+m*h)
    end do
     do n=6,num-1,6
      s=s+2.0*f(a+n*h)
    end do
    ev=g(b)-g(a)
    wdl_ar=s*(3.0*h/10.0)
    wdl_er=abs(ev-wdl_ar)
    write(*,10)wdl_ar,wdl_er
    10 Format('area is=',f20.5,10x,'error is=',f20.5)
end program
```

Romberg Program

```
program romberg
    implicit none
    integer::n,i,j,k
    real::a,b,tol,h(20),r(20,20),f,sum1,g,er
    read(5,*)a,b,n,tol
    write(*,*)'----------------------------------------------------------------------'
    do i=1,n
      h(i)=(b-a)/2**(i-1)
    end do
    r(1,1)=(h(1)/2)*(f(a)+f(b))
    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(*,*)'------------------------------------------------------------------------'
         write(6,*)'value of the integration=',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 function
real function g(x)
g=atan(x)
return
end function
```

Gausse elimination without pivotiog

```
PROGRAM gauss_eli
IMPLICIT NONE
REAL::A(20,20),k1,k2,v(20),c
INTEGER::i,j,n,k
PRINT *,'GAUSS ELIMINATION - WITHOUT PIVOTING'
PRINT *,'NO. OF ROWS'
```

```
READ(*,*)n
PRINT *,'ENTER ELEMENTS'
READ(*,*)((A(i,j),j=1,n+1),i=1,n)

PRINT *,'YOUR MATRIX - '
DO i=1,n
  write(*,30)(A(i,j),j=1,n+1)
END DO
30 format(30(2x,f12.4))
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(*,40)(A(i,j),j=1,n+1)
END DO
40 format(45(2x,f12.4))
!LAST ELEMENT
v(n)=A(n,n+1)/A(n,n)

!REST OF THE ELEMENTS

DO i=n-1,1,-1
c=0.
```

```
 DO j=i+1,n
  c=c+A(i,j)*v(j)!DETECTING LAST VALUE
 END DO

v(i)=(A(i,n+1)-c)/a(i,i)
END DO

PRINT *,'SOLUTIONS ARE - '
DO i=1,n
 write(*,50)i,v(i)
END DO
50 format(2x,'x',i1,'=',f12.4)
END PROGRAM
```

With pivoting

```
PROGRAM gauss_eli_pivot
IMPLICIT NONE
REAL::A(20,20),k1,k2,v(20),c
INTEGER::i,j,n,k
PRINT *,'GAUSS ELIMINATION - WITH PIVOTING'
PRINT *,'NO. OF ROWS'
READ(*,*)n
PRINT *,'ENTER ELEMENTS'
READ(*,*)((A(i,j),j=1,n+1),i=1,n)
PRINT *,'YOUR MATRIX - '
DO i=1,n
write(*,40)(A(i,j),j=1,n+1)
END DO
40 format(30(2x,f12.4))
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(*,50)(A(i,j),j=1,n+1)
END DO
50 format(40(2x,f12.4))
!LAST ELEMENT
v(n)=A(n,n+1)/A(n,n)
!REST OF THE ELEMENTS
DO i=n-1,1,-1
c=0.
DO j=i+1,n
  c=c+A(i,j)*v(j)!DETECTING LAST VALUE
END DO
v(i)=(A(i,n+1)-c)/a(i,i)
END DO
PRINT *,'SOLUTIONS ARE - '
DO i=1,n
write(*,60)i,v(i)
END DO
60 format(2x,'x',i1,'=',f12.4)
END PROGRAM
!PIVOT SUBROUTINE
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
```

Jacobi

```
program jacobi
   implicit none
      integer::i,j,k,t,iter,n,r
   real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2
   read(*,*)n,tol,iter
   read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n),(x0(i),i=1,n)
   do k=1,iter
   do i=1,n
     s1=0
   do j=1,n
```

```
    if(i/=j)s1=s1+a(i,j)*x0(j)
    end do
    x(i)=(b(i)-s1)/a(i,i)
    end do
    write(*,*)(x(i),i=1,n)
    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 i=1,n
       x0(i)=x(i)
       end do
       end if
    end do
 end program
```

Gausse Seidel

```
program gauss_seidal
    implicit none
       integer::i,j,k,t,iter,n,r
    real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2
```

```
read(*,*)n,tol,iter
read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n),(x0(i),i=1,n)
do k=1,iter
do i=1,n
  s1=0
do j=1,n
if(i<j)then
s1=s1+a(i,j)*x0(j)
else if(i>j)then
s1=s1+a(i,j)*x(j)
end if
end do
x(i)=(b(i)-s1)/a(i,i)
end do
write(*,*)(x(i),i=1,n)
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 i=1,n
   x0(i)=x(i)
```

```
      end do
      end if
   end do
 end program
```

Sor Program

```
program sor
   implicit none
      integer::i,j,k,t,iter,n,r
   real::a(5,5),x(5),b(5),x0(5),tol,s1,norm1,norm2,w=1.25
   read(*,*)n,tol,iter
   read(*,*)((a(i,j),j=1,n),i=1,n),(b(i),i=1,n),(x0(i),i=1,n)
   do k=1,iter
   do i=1,n
     s1=0
   do j=1,n
   if(i<j)then
   s1=s1+a(i,j)*x0(j)
   else if(i>j)then
   s1=s1+a(i,j)*x(j)
   end if
   end do
   x(i)=(1-w)*x0(i)+w*(b(i)-s1)/a(i,i)
   end do
   write(*,*)(x(i),i=1,n)
   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 i=1,n
     x0(i)=x(i)
     end do
     end if
  end do
end program
```