Aim : To find the 1D and 2D integration using the Gauss Quadrature formula.

Theory:

Gauss Quadrature Formula:

It is type of finding the integration between the -1 to 1 for a function using weighted sum of the value of the function at some points that depend upon the degree of the polynomial function we use to mimic the function in the range.

Formula:

For 1 D:

$$\int_{-1}^{1} f(x)\,dx = \sum_{i=1}^{n} c_i * f(x_i)$$

For 2 D:

$$\int_{-1}^{1}\int_{-1}^{1} f(x)\,dx dy = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j f(x_i) f(x_j)$$

here c_i and x_i are the weight and point at which we find the integration and n is degree of polynomial.

For range from a to b the x change into:

$$x = \frac{(b-a)*t+(a+b)}{2} \qquad \text{and} \qquad dx = \frac{(b-a)}{2} * dt$$

and using this we can find the integration between any two points.

Algorithm:

1 D: First we need to take all value from user and also define the points and weight for the integration. And then start a loop from 1, n and calculate the weighted sum and print the result.

2 D: This is same as the 1 D but this time we need to loop to calculate the weighted sum and print the result.

For One Dimension:

Program in FORTRAN95 :

```
program main
   implicit none     ! declaring the variables need
   real :: upper,lower , integration = 0
   real , dimension(:,:), allocatable  :: optimizer
   integer :: i , degree

   ! getting the input about the upper and the lower limit and also value of n (number of points)

   print *, "Enter the number of points taken (Hint : 2 to 4):: "
   read *, degree
   print *, "Enter the upper limit :: "
   read *, upper
   print *, "Enter the lower limit :: "
   read *, lower
```

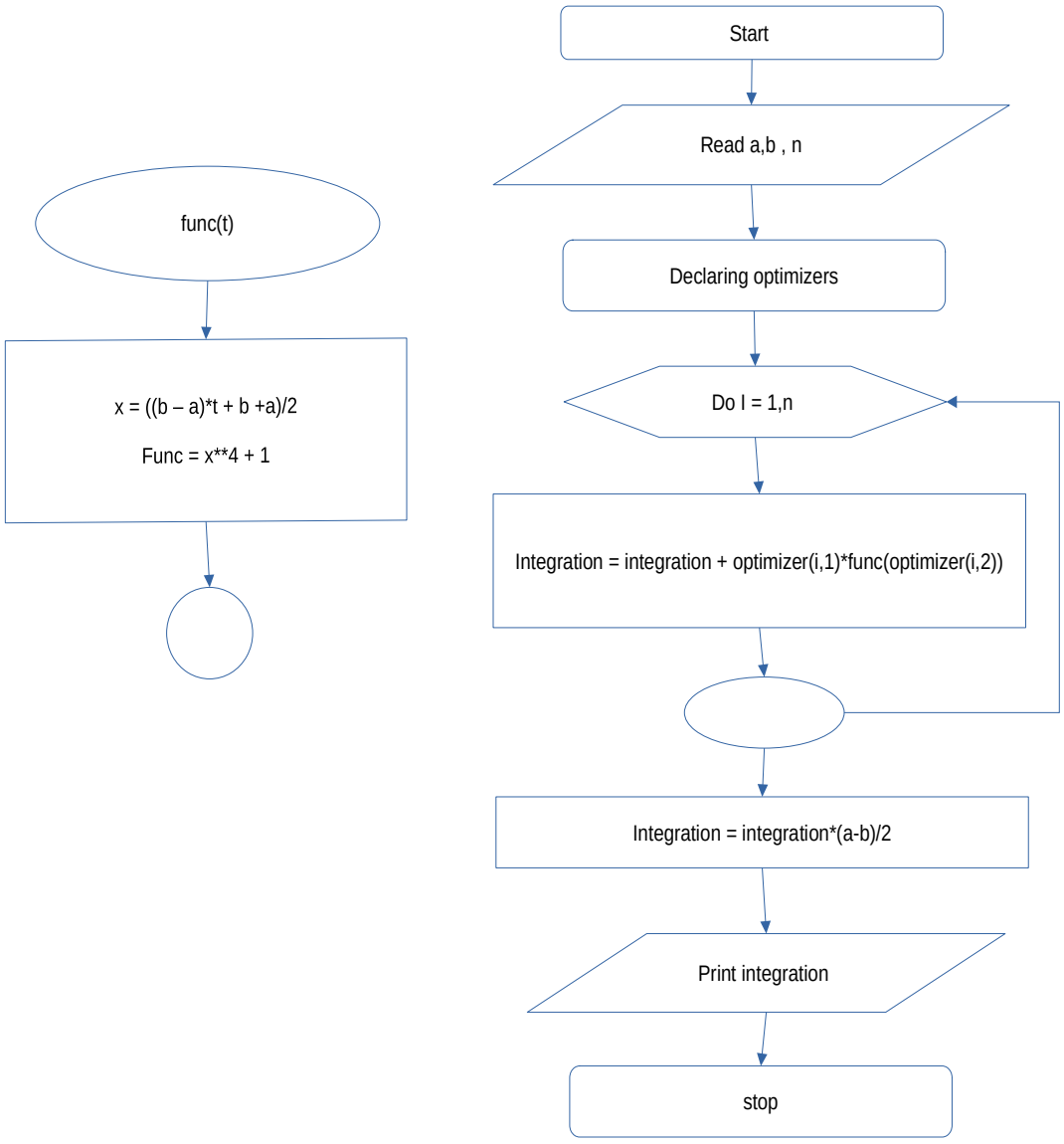
```
if (degree == 2) then     ! declaring the optimizer points value in the array
   allocate(optimizer(2,2))
   optimizer(1,1) = 1
   optimizer(1,2) = -0.5773502692
   optimizer(2,1) = 1
   optimizer(2,2) = 0.5773502692

else if (degree == 3) then
   allocate(optimizer(3,2))
   optimizer(1,1) = 0.5555555
   optimizer(1,2) = -0.7745967
   optimizer(2,1) = 0.8888888
   optimizer(2,2) = 0
   optimizer(3,1) = 0.5555555
   optimizer(3,2) = 0.7745967

else if (degree == 4) then
   allocate(optimizer(4,2))
   optimizer(1,2) = 0.8611363116
   optimizer(1,1) = 0.3478548451
   optimizer(2,2) = 0.3399810436
   optimizer(2,1) = 0.6521451549
   optimizer(3,2) = -0.3399810436
   optimizer(3,1) = 0.6521451549
   optimizer(4,2) = -0.8611363116
   optimizer(4,1) = 0.3478548451

else
   stop "Unable to do this operation for this degree of polynomial"
endif

! taking loop for all the value for calculating the integration

do i = 1,degree
   integration = integration + optimizer(i,1)*(func(optimizer(i,2)))
enddo
deallocate(optimizer) ! deallocate the array
integration = (integration*(upper  - lower))/2.0
print *, "The integration is :: ",integration !printing the result
stop
contains
   real function func(t) ! function for making change in variable for condition
      real , intent(in) :: t !of integration limit satisfied
      real :: output , x
      x = ((upper - lower)*t + upper + lower)/2.0
      output = x**4+1
      func = output
      return
   end function func
```

end program

Output :

Enter the number of points taken (Hint : 2 to 4)::
4
Enter the upper limit ::
10
Enter the lower limit ::
2
The integration is :: 20001.6016

Flow Chart :



For Two Dimension:

Program in FORTRAN95:

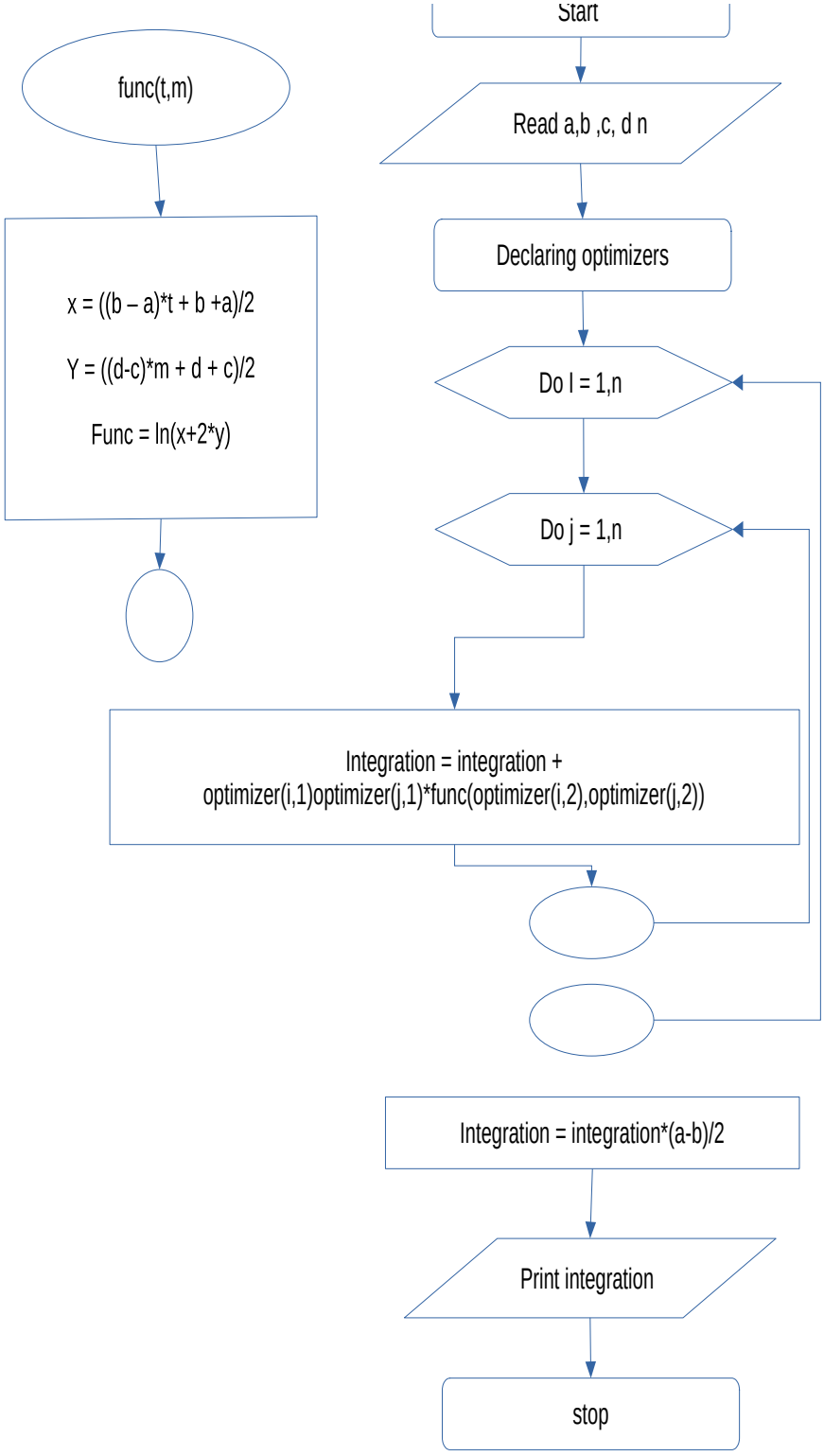
```
program main

   implicit none  ! declaring the variables for the program

   real :: upper_y,upper_x,lower_y,lower_x , integration = 0
   real , dimension(:,:), allocatable :: optimizer
   integer :: i , j , degree

! getting all the input form the user

   print *, "Enter the order of the polynomial  (Hint : 2 to 4):: "
   read *, degree
   print *, "Enter the upper limit of x axis :: "
   read *, upper_x
   print *, "Enter the lower limit of x axis:: "
   read *, lower_x
   print *, "Enter the upper limit of y axis:: "
   read *, upper_y
   print *, "Enter the lower limit of y axis:: "
   read *, lower_y

! declaring the optimizer for the weight and points for the Gauss Quadrature Formula

   if (degree == 2) then
      allocate(optimizer(2,2))
      optimizer(1,1) = 1
      optimizer(1,2) = -0.5773502692
      optimizer(2,1) = 1
      optimizer(2,2) = 0.5773502692

   else if (degree == 3) then
      allocate(optimizer(3,2))
      optimizer(1,1) = 0.5555555
      optimizer(1,2) = -0.7745967
      optimizer(2,1) = 0.8888888
      optimizer(2,2) = 0
      optimizer(3,1) = 0.5555555
      optimizer(3,2) = 0.7745967

   else if (degree == 4) then
      allocate(optimizer(4,2))
      optimizer(1,2) = 0.8611363116
      optimizer(1,1) = 0.3478548451
      optimizer(2,2) = 0.3399810436
```

```
        optimizer(2,1) = 0.6521451549
        optimizer(3,2) = -0.3399810436
        optimizer(3,1) = 0.6521451549
        optimizer(4,2) = -0.8611363116
        optimizer(4,1) = 0.3478548451

    else
        stop "Unable to do this operation for this degree of polynomial"
    endif

    do i = 1,degree
        do j = 1,degree
            integration = integration +
optimizer(i,1)*optimizer(j,1)*func(optimizer(i,2),optimizer(j,2))
        enddo
    enddo

    deallocate(optimizer)

    integration = (integration*(upper_y  - lower_y)*(upper_x – lower_x))/4.0

    print *, "The integration is :: ",integration
    stop

    contains
        real function func(m,t)
            real , intent(in) :: t , m
            real :: output , x ,y
            x = ((upper_x - lower_x)*t + upper_x + lower_x)/2.0
            y = ((upper_y - lower_y)*m + upper_y + lower_y)/2.0
            output = log(x+2*y)
            func = output
            return
        end function func
end program
```

Output:

```
 Enter the order of the polynomial  (Hint : 2 to 4)::
3
 Enter the upper limit of x axis ::
2
 Enter the lower limit of x axis::
1.4
 Enter the upper limit of y axis::
1.5
 Enter the lower limit of y axis::
1
 The integration is ::   0.429554492
```

Flow Chart :


Start
Read a,b ,c, d n
Declaring optimizers
Do I = 1,n
Do j = 1,n
Integration = integration +
optimizer(i,1)optimizer(j,1)*func(optimizer(i,2),optimizer(j,2))
Integration = integration*(a-b)/2
Print integration
stop
func(t,m)
x = ((b – a)*t + b +a)/2
Y = ((d-c)*m + d + c)/2
Func = ln(x+2*y)