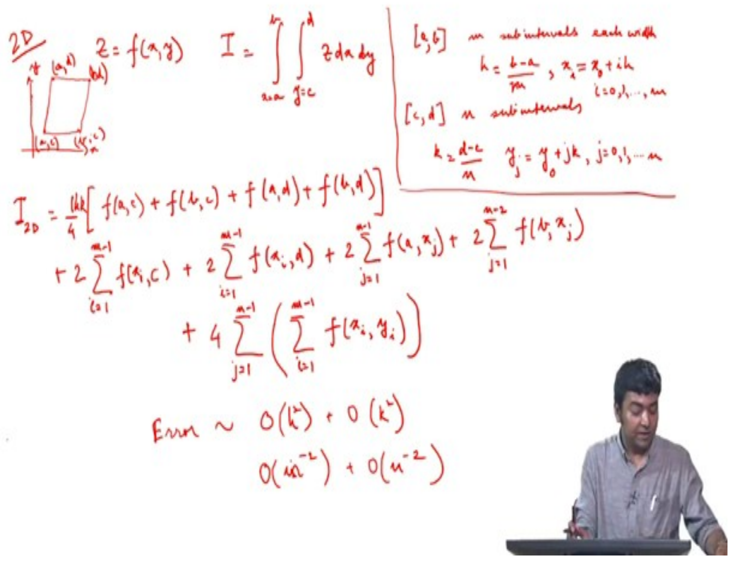
Computational Physics
Dr. Apratim Chatterji
Dr. Prasenjit Ghosh
Department of Physics
Indian Institute of Science Education and Research, Pune

Lecture - 07
Numerical Integration Part 02

Now, what happens if I have a two-dimensional function?

(Refer Slide Time: 00:24)



2D $z = f(x,y)$ $I = \int_{x=a}^{b}\int_{y=c}^{d} z\,dx\,dy$

$[a,b]$ m subintervals each width $h = \frac{b-a}{m}$, $x_i = x_0 + ih$, $i = 0,1,\dots,m$

$[c,d]$ n subintervals $k = \frac{d-c}{n}$ $y_j = y_0 + jk$, $j = 0,1,\dots n$

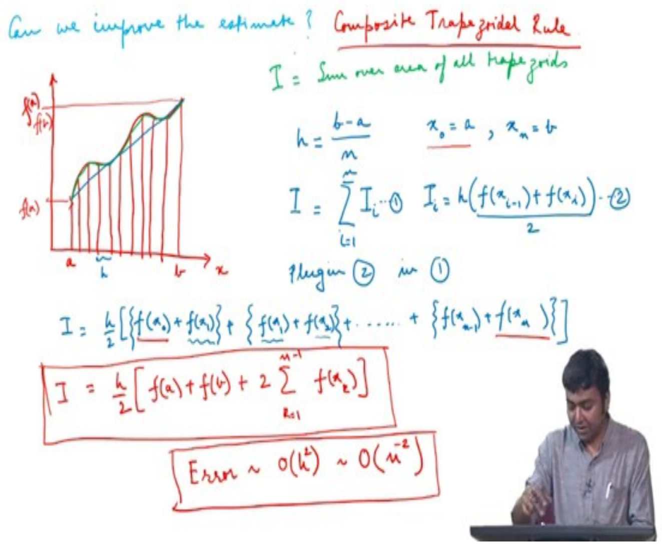
$$I_{2D} = \frac{hk}{4}\Big[f(a,c) + f(b,c) + f(a,d) + f(b,d)\Big] + 2\sum_{i=1}^{m-1} f(x_i,c) + 2\sum_{i=1}^{m-1} f(x_i,d) + 2\sum_{j=1}^{n-1} f(a,x_j) + 2\sum_{j=1}^{n-2} f(b,x_j) + 4\sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} f(x_i,y_i)\right)$$

$$\text{Err} \sim O(h^2) + O(k^2)$$
$$O(m^{-2}) + O(n^{-2})$$

For example, now I am moving on to 2D. So, suppose I have a function z which I am calling as a function of two variables x and y and I want to compute this integral. So, x equals to a to b and then y is equal to c to d, z dx dy. So, one can use the same thing. So, what I do is that interval between a to b that I split into m sub intervals, each with each having a width h say. So, what that means, is h is equal to b minus a by m.

Similarly, the interval c to d that I split into n sub intervals, this will be n, where each width is given width of the width of each one is given by d minus c by n. And since we are discretizing it the value of the ith point xi that will be given by x0 plus i h where my i goes from 0, 1 to all the way to n. In this case the yj will be given by y0 plus j k, where my j goes again from 0, 1 all the way to m, n.

(Refer Slide Time: 02:42)



Can we improve the estimate? Composite Trapezoidal Rule

I = Sum over area of all trapezoids

$h = \frac{b-a}{m}$ $x_0 = a$, $x_m = b$

$I = \sum_{i=1}^{m} I_i \cdots ①$ $I_i = h\frac{(f(x_{i-1}) + f(x_i))}{2} \cdots ②$

Plugin ② in ①

$I = \frac{h}{2}\left[\{f(x_0) + f(x_1)\} + \{f(x_1) + f(x_2)\} + \cdots + \{f(x_{m-1}) + f(x_m)\}\right]$

$$\boxed{I = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{k=1}^{m-1} f(x_k)\right]}$$

$$\boxed{\text{Error} \sim O(h^2) \sim O(m^{-2})}$$

And then if we do a similar exercise that we had done in this case, I am not going into the derivation, I will just directly write down the equation. So, my integral two-dimensional integral under the trapezoidal rule will be given by this function.

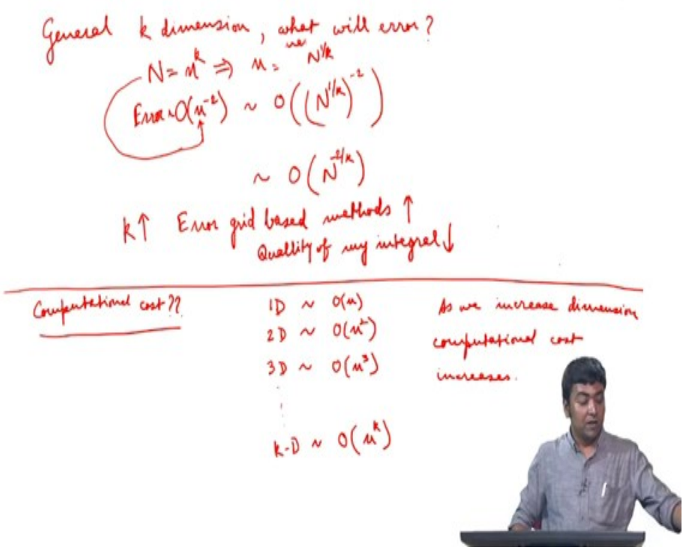
So, schematically what we can think of is suppose this is my x and y axis. So, I have my point a here, b here, somewhere here I have c and I have d here. So, basically the coordinate of this point is a c, the coordinate of this point is b c, the coordinate of this point is a d and the coordinate of this point one can write as b d. And so, I have trapezium in the base in the x y plane and I want to compute now the volume of this trapezium, ok. So, z is perpendicular to this plane.

So, if you remember the previous case. So, here you get you need to compute the value of the function at f(a) and at f(b) the two points of the integral. In a similar way here also you need to compute the value of the integral at all these points. So, that I write as one-fourth h k, where h and k are these two quantities here and then what we will get as function, the value of the function at the coordinate (a ,c) plus value of the function at the coordinate (b,c) plus value of the function at the coordinate (a,d) plus value of the function at the coordinate (b,d). So, this is one term I will get.

So, apart from that I will get another term which is for example, along the edges of this trapezium. So, that will be given by, so plus 2 sum over i equals to 1 to m minus 1 f x i at c, so basically I am looking along this line plus 2 sum over i equals to 1 to m minus 1 f of xi . So, basically along this line, function at along this line plus i get 2 sum over j equals to 1 to m,.... sorry this will not be m..., this will be n here, (jn-1) with f a x j plus 2 into sum over j equals to 1 to n minus 2 f(b) * xj. So, this goes along this 4 sides of the trapezium. And then we also need to evaluate the points the value of the functions at the points which lies in within this trapezium. So, for that we will get double sum, sum is j equals to 1 to n minus 1, then sum over i equals to 1 to m minus 1 f (xi ,y i). So, this is my expression for the integral.

So, in this case also the error typically scales as of the order of h square plus of the order of k square or in other words you can think of the order of n to the power minus 2 plus of the order of n to the power minus 2. So, this is the how the error is scalable.

(Refer Slide Time: 07:30)



General k dimension, what $\overset{we}{\text{will}}$ error?

$N = n^k \Rightarrow n = N^{1/k}$

$\text{Error} \sim O(n^{-2}) \sim O\left(\left(N^{1/k}\right)^{-2}\right)$

$\sim O\left(N^{-2/k}\right)$

$k\uparrow$ Error grid based methods $\uparrow$

Quality of my integral $\downarrow$

Computational cost??

$1D \sim O(n)$

$2D \sim O(n^2)$

$3D \sim O(n^3)$

$\vdots$

$k\text{-}D \sim O(n^k)$

As we increase dimension computational cost increases.

So, if we think of again doing the same integral. So, for a general k dimension, what will be the error? So, we will look into that now. Or in other words how the error in these type of methods typically scales as one increases the dimension or is the function of the dimension size.

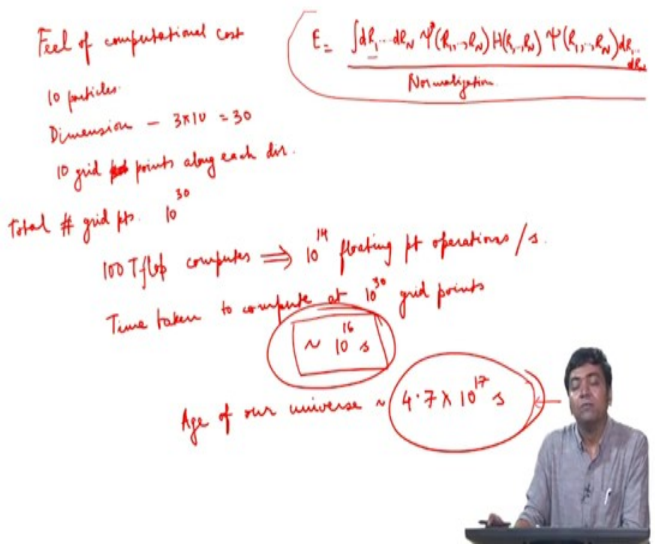
So, now, suppose I have N number of points, these are the total number of points in my N dimensional grid and along each dimension I have n, small n points. So, the total number of points N, capital N is equal to n to the power k. And as I mentioned before that the error scales as is of the order of n to the power minus 2. In the same way as we have done for the 2D case, if we plugin this equation here the value of, if we write the small n in terms of the capital N what we will get is that my order scales as n. So, this equation gives me small n is equal to n to the power 1 by k, I can write. So, in other words what I we get is 1 by k to the power minus 2.

So, what it tells us is that the error scales of the order of minus 2 by k. So, what does this mean? So, what it means is that as we increase the dimension, as we go to higher and higher dimension the error in the grid based method goes to increases or in other words the quality of my integral goes down.

Now, the second thing is what about the computational cost. So, for the 1D case your computational cost scales as of the order of n because you are evaluating the functions at n different points. The moment as we saw in the last slide that when we go to the 2D case we have to evaluate the function on an m into n grid. So, what it implies is that, it scales as order n square. Similarly, for a 3D case the computational cost will scale as order n cube.

So, for a k dimensional case the computational cost is scales as n to the power k. So, basically as we increase dimension the computational cost increases which is a bad thing and we do not want that. So, to get a feel of how much the computational cost increase what we will do is I will take a very common example. So, particularly, so I will take a case where these evaluating this multidimensional integrals are very necessary.

(Refer Slide Time: 11:25)



Feel of computational cost

$$E = \frac{\int dR_1 \cdots dR_N \, \Psi^*(R_1, \ldots, R_N) H(R_1, \ldots, R_N) \Psi(R_1, \ldots, R_N)}{\text{Normalization}}$$

10 particles

Dimension $- 3 \times 10 = 30$

10 grid ~~pt~~ points along each dir.

Total # grid pts. 10^{30}

100 Tflop computer $\Rightarrow 10^{14}$ floating pt operations/s.

Time taken to compute at 10^{30} grid points $\boxed{\sim 10^{16}\,s}$

Age of our universe $\sim 4.7 \times 10^{17}\,s$

So, for example, so what we are trying to do is now to get a feel of computational cost. So, those of you who have a background in quantum mechanics you know that we are often interested in computing the expectation value of my Hamiltonian because that gives me the energy of my system. So, how does we write the expectation value of my Hamiltonian? So, we can write E equals to the integral dR 1 dot dot dot dR N H which is a function of R 1 dot dot dot R N, psi star which is my wave function R 1 to R N, then H which is a function of R 1 to R N and then psi goes as R 1 to R N. And then here it goes to dR 1 to dR N and then we have the normalization below, ok.

So, suppose we have we have a system where we have 10 particles, ok. So, if and we want to evaluate the integral for this. So, the number of the dimension of this integral, so R here is the position vector, ok. So, each vector has 3 directions. The dimension of this integral will be 3 into 10 equals to 30. Now, suppose along each direction I use 10 grid points along each directions. This is the smallest one can think of. So, what that will result is that, so the number of mesh points, number of grid points, the total number of grid points I would say will then be 10 to the power 30, ok. So, this we have to evaluate the function at these many grid points to compute the integral.

Now, the let us suppose we have a 100 teraflop computer, ok. What this means is that this computer can perform 10 to the power 14 floating point operations per second, ok. So, now we compute the time that will take and also we assume that to evaluate the function at each grid point we need to do one floating point operation, then the time taken to compute at 10 to the power 30 grid points will be of the order of 10 to the power 16 seconds, ok. So, this will be the my computational cost, if I want to do compute this integral for a system of 10 particles using a relatively coarse grid where each along each direction I am using a 10 using just 10 grid points.

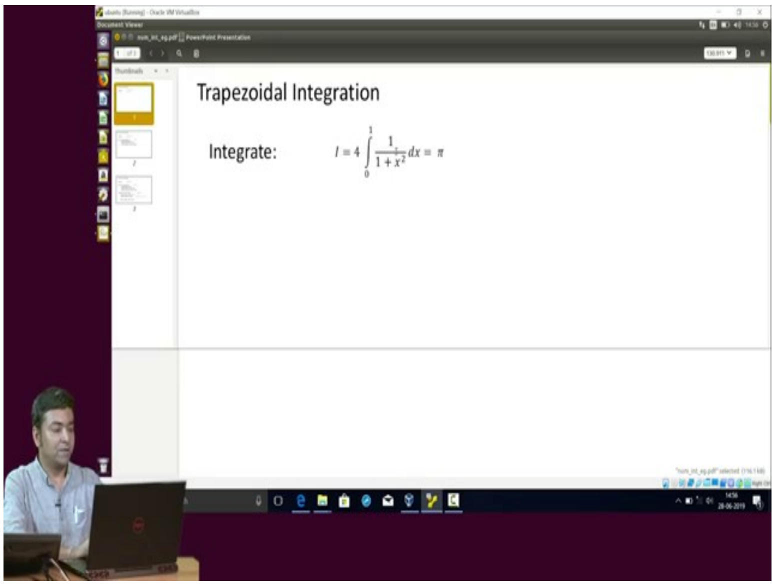
Now, how large is 10 to the power 16 seconds? To give that perspective let me just code the age of our universe. So, hat is typically of the order of 4.7 into 10 to the power 17 second. So, now, look at this two numbers here, ok. So, these numbers are very much comparable. So, what this numbers tells us that in terms of the computational cost if we use this grid based method to evaluate an integral which is something like this or in other words evaluate a n-dimensional integral then we will never be able to use it. I mean find out the solution for that in our lifetime.

So, now the question is what to do but before we go to what to do. So, let me just tell you one thing. So, trapezoidal rule is one of the simplest rules. So, there are some other rules also for example, the Simpsons one-third rule, then there is the Gaussian quadrature rule and all. So, those are much more sophisticated and computationally efficient, but having said that those methods are still not good enough or computationally fast enough or accurate enough to evaluate these type of multidimensional integrals.

Now, all this methods which we have seen so far these are called grid based method, deterministic methods because we are evaluating the function at a predetermined set of grid points. So, we know that we are going to evaluate the functions at this value of my grid. So, in that way it is a deterministic method, ok.

So, now, we come to the next question as to how do we get rid of this problem. So, we have two problems here. We have a method here which is computationally expensive and in a realistic way it is computationally unfeasible to use it and which has a error which increases as we go to higher and higher dimensions. So, how do we solve this problem and now one line answer to this is we move to the stochastic methods of integral.

(Refer Slide Time: 17:48)

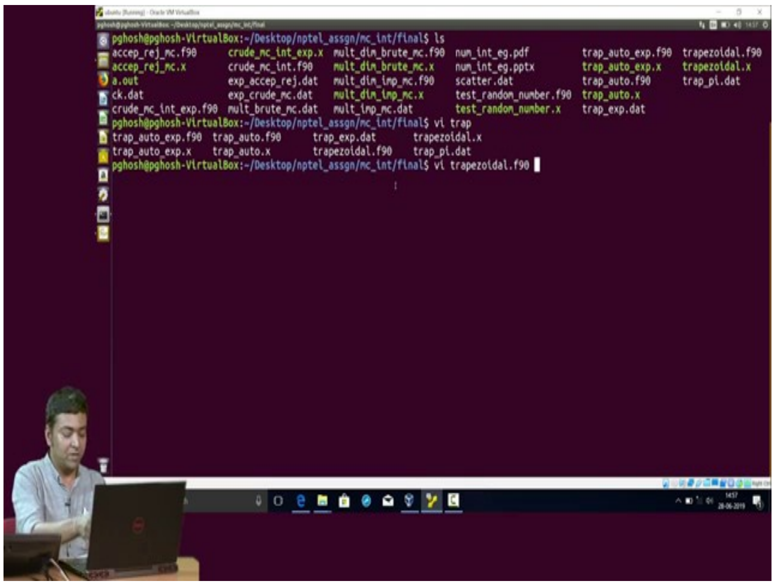


Trapezoidal Integration

Integrate: $I = 4\int_0^1 \frac{1}{1+x^2}dx = \pi$

So, what we will do is we will look into how to do the numerical integration using the trapezoidal method for that what we will try to do is we will try to evaluate this following integral. So, basically the limits are from 0 to 1, 4 by 1 plus x square dx and we know that the value of this integral is pi.

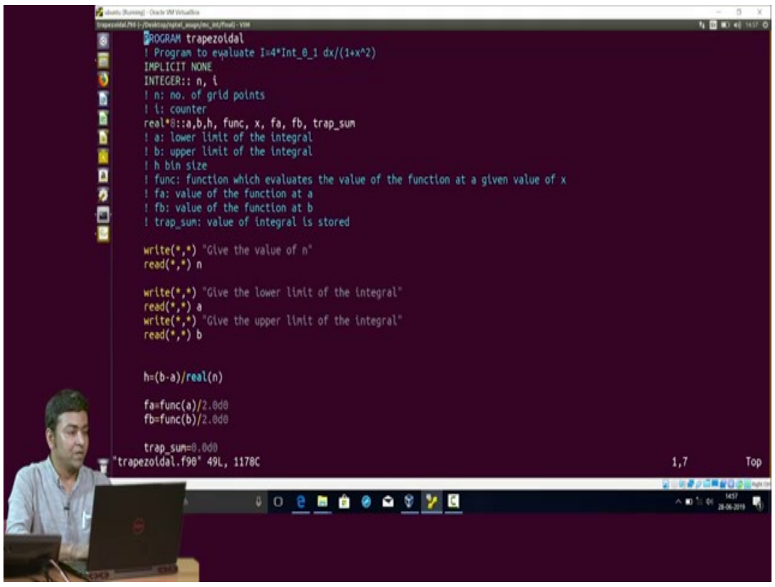
(Refer Slide Time: 18:12)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls
accep_rej_mc.f90      crude_mc_int_exp.x  mult_dim_brute_mc.f90  num_int_eg.pdf          trap_auto_exp.f90  trapezoidal.f90
accep_rej_mc.x        crude_mc_int.f90    mult_dim_brute_mc.x    num_int_eg.pptx         trap_auto_exp.x    trapezoidal.x
a.out                 exp_accep_rej.dat   mult_dim_imp_mc.f90    scatter.dat             trap_auto.f90      trap_pi.dat
ck.dat                exp_crude_mc.dat    mult_dim_imp_mc.x      test_random_number.f90  trap_auto.x
crude_mc_int_exp.f90  mult_brute_mc.dat   mult_imp_mc.dat        test_random_number.x    trap_exp.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trap
trap_auto_exp.f90  trap_auto.f90    trap_exp.dat      trapezoidal.x
trap_auto_exp.x    trap_auto.x      trapezoidal.f90   trap_pi.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trapezoidal.f90
```

So, what I have here is the code. So, this is the directory where all the codes are if I do 'l s' you can see all the different codes here and the code which does this is this trapezoidal vi underscore trapezoidal dot f 90. So, I am opening it in a vi editor and this is how the code looks like.

(Refer Slide Time: 18:36)



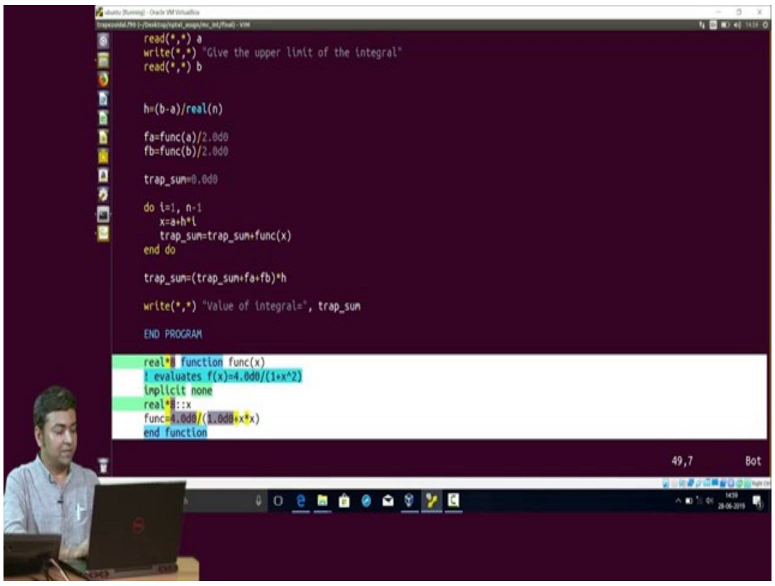
```
PROGRAM trapezoidal
! Program to evaluate I=4*Int_0_1 dx/(1+x^2)
IMPLICIT NONE
INTEGER:: n, i
! n: no. of grid points
! i: counter
real*8::a,b,h, func, x, fa, fb, trap_sum
! a: lower limit of the integral
! b: upper limit of the integral
! h bin size
! func: function which evaluates the value of the function at a given value of x
! fa: value of the function at a
! fb: value of the function at b
! trap_sum: value of integral is stored

write(*,*) "Give the value of n"
read(*,*) n

write(*,*) "Give the lower limit of the integral"
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b


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

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0
"trapezoidal.f90" 49L, 1178C                                    1,7           Top
```

So, as I mentioned in the lectures of the first module that whenever one writes a FORTRAN program. So, it starts with this program keyword and then the name of the program, and then I have put in comment a line which says that I will be doing this integral using the trapezoidal rule.

So, what do I need to do? So, I need to have some variables, some of them are integers, and some of them are real numbers. So, amongst the integers I am using two integer variables here n and i. So, n is the total number of grid points that I will be using for my to evaluate the integral along the x axis and i is the counter. Then I have some real variables, ok. One thing I have not mentioned is this real star 8. So, far what we have been doing is we are just using real. So, star 8 basically what it tells the program or tells the compiler is that I want to do this calculations in double precision.

So, here I have a set of variables which are already defined below in the comments. For example, my a is the lower limit of the integral, then the b is the upper limit of the integral, h will be the bin size. By bin size what I mean is, so there is limit on the integral from 0 to 1 as you see here. So, from 0 to 1, this I will be dividing into n bins. So, h stores the value of the size of each bin. This variable 'func', this is a function and this function I am using to evaluate this particular function 4 by 1 plus x square.

(Refer Slide Time: 20:27)



```
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b


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

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0

do i=1, n-1
   x=a+h*i
   trap_sum=trap_sum+func(x)
end do

trap_sum=(trap_sum+fa+fb)*h

write(*,*) "Value of integral=", trap_sum

END PROGRAM

real*8 function func(x)
! evaluates f(x)=4.0d0/(1+x^2)
implicit none
real*8::x
func=4.0d0/(1.0d0+x*x)
end function
```

So, this you can see I have the function here at the end of the program. So, what it does is the program sends to this function the main program send to this function, the argument x which is the value of x at which this function needs to be evaluated and then it evaluates this functions and return it back to the main program. So, that is what information that is stored in this ‘func’ variable.

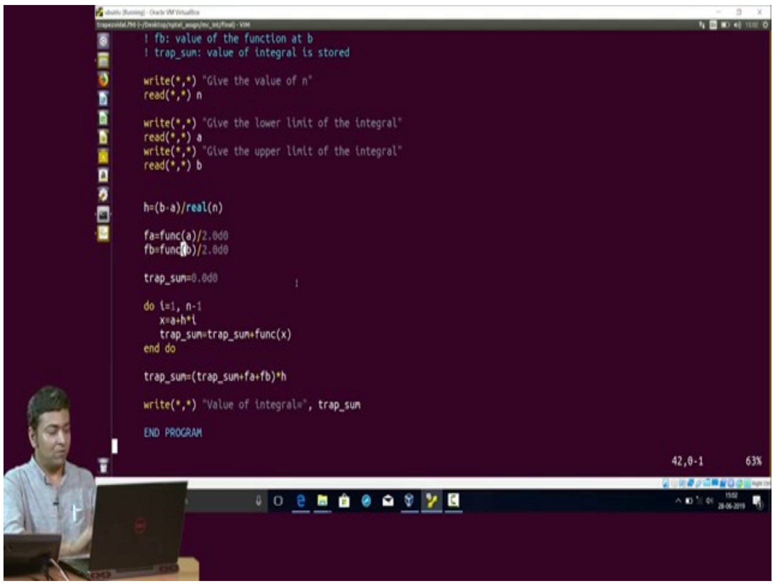
Then I have some other temporary variables. Then f(a) is the value of the function at a and f(b) is the value of the function at b. And then what I do is I store the value of the integral in trap underscore sum. So, this goes the declaration of variables, now we begin the program.

So, to start the program what I, the program needs to know from the user is how many points do I want my interval to be divided into. So, that is why I write using this write statement, I give a I prompt on the screen that gives the value of n and when the user sends in the value of n. This is reading in this statement. So, once I know the number of bins I also need to know what are the limits of my integral.

So, again as before I give using the write statement I prompt on the screen that gives the lower limit of the integral, then I read that value that uses separate value. Then, I again ask for the code again asks for the upper limit if the integral and then it reads in the user supplied value which is stored in this variable b.

So, now once I know my n and I know my a and b what I am doing here this part of the code what it does is it evaluates the size of each bin. So, that is h the variable I used to store that is h. So, the size of each bin is given by the difference between the upper and the lower limit divided by the number of bins that is b minus a divided by n. So, you note that in the beginning of the code I have put n as an integer and now I am doing a division here which is not an integer type of division. So, what I am changing it into a real number because b minus a is also real. So, I would like to have everything real number here. So, that is why I am using this FORTRAN attribute real which changes this integer variable n to a real one.

(Refer Slide Time: 23:04)



```
! fb: value of the function at b
! trap_sum: value of integral is stored

write(*,*) "Give the value of n"
read(*,*) n

write(*,*) "Give the lower limit of the integral"
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b


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

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0

do i=1, n-1
   x=a+h*i
   trap_sum=trap_sum+func(x)
end do

trap_sum=(trap_sum+fa+fb)*h

write(*,*) "Value of integral=", trap_sum

END PROGRAM
```

```
42,0-1        63%
```

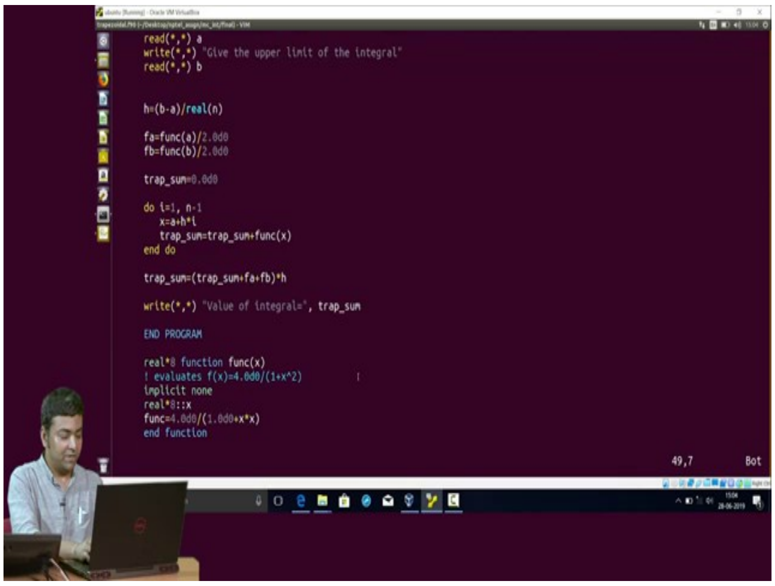
So, once I have the bin size what I also need to do know is what are the values of the function at the two limits of the integral that is at f(a), where x equals to 0 and f(b) that is the upper limit in this case it happens to be 1.

So, now once I know these things then what else do I need to know. I need to know what are the values of my functions at the different; at the different grid points that lie between 0 and 1. So, and then once I know this functions, so what I need to do is I add them up. So, how do I do that? What I do is I use a do loop. So, my do loop goes from i equals to 1 to n minus 1. The reason it goes to n minus 1 is because the nth point is my end point which is the b and for that I already have the function. So, this do loop goes from 1 to n minus 1 and at each iteration of this do loop the first thing is done is we evaluate the value of x. So, we start at the lower limit then to that we add the width of the bin size into i which is the increment of the step size.

So, and once we know the value of x what we do is we call this function and this functions returns me or returns the code or the main program the value of the function at the point x. And this is then added up. So, this is done in this line here. So, each time this do loop is executed the function is evaluated and is added to the previous value. So, that is why one should always be careful that before I start this do loop I should set this to a value 0, otherwise FORTRAN might take some arbitrary initial value in this variable trap underscore sum which might give me erroneous results.

So, once I have now summed up the values of the functions at the different grid points, so what I do is now I evaluate the value of the integral which is given by sum of all the grid points plus f(a) and f(b) into the height of my trapezoidal which is h.

(Refer Slide Time: 25:22)

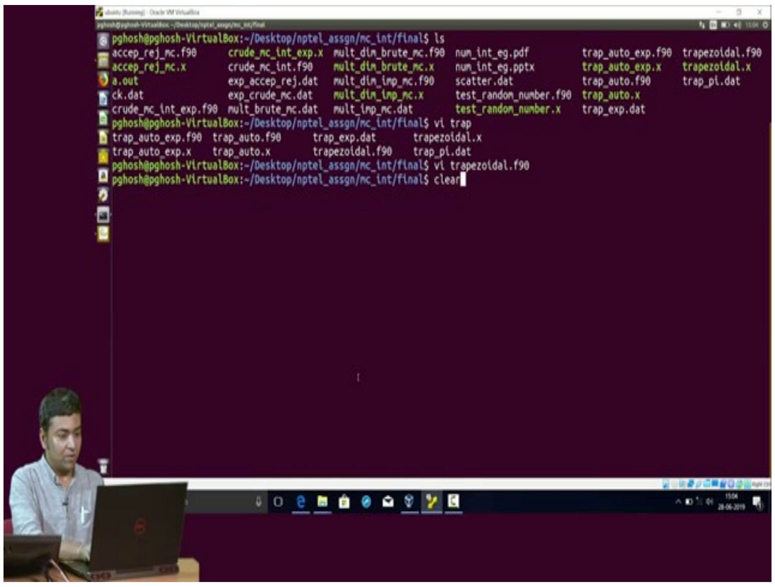


```
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b


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

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0

do i=1, n-1
   x=a+h*i
   trap_sum=trap_sum+func(x)
end do

trap_sum=(trap_sum+fa+fb)*h

write(*,*) "Value of integral=", trap_sum

END PROGRAM

real*8 function func(x)
! evaluates f(x)=4.0d0/(1+x^2)
implicit none
real*8::x
func=4.0d0/(1.0d0+x*x)
end function
```

```
49,7          Bot
```

So, once that is done what I do is I write on the screen the value of the integral equals to trap underscore sum. So, this is the code and it is a very simple few lines of code, more precisely it is just 49 lines of code and let us see what we get after we run it.

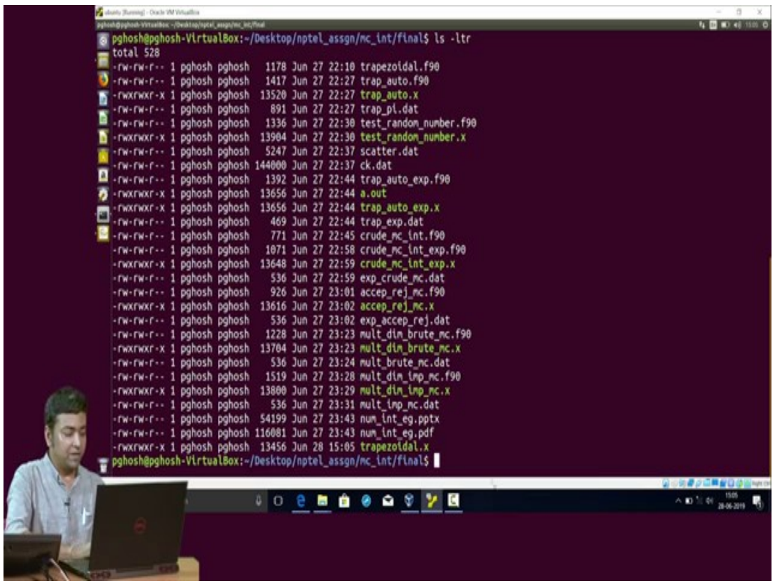
(Refer Slide Time: 25:39)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls
accep_rej_mc.f90      crude_mc_int_exp.x  mult_dim_brute_mc.f90  num_int_eg.pdf          trap_auto_exp.f90  trapezoidal.f90
accep_rej_mc.x        crude_mc_int.f90    mult_dim_brute_mc.x    num_int_eg.pptx         trap_auto_exp.x    trapezoidal.x
a.out                 exp_accep_rej.dat   mult_dim_imp_mc.f90    scatter.dat             trap_auto.f90      trap_pi.dat
ck.dat                exp_crude_mc.dat    mult_dim_imp_mc.x      test_random_number.f90  trap_auto.x
crude_mc_int_exp.f90  mult_brute_mc.dat   mult_imp_mc.dat        test_random_number.x    trap_exp.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trap
trap_auto_exp.f90  trap_auto.f90     trap_exp.dat      trapezoidal.x
trap_auto_exp.x    trap_auto.x       trapezoidal.f90   trap_pi.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trapezoidal.f90
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ clear
```

So, to run it, as I told you first we need to compile the code.

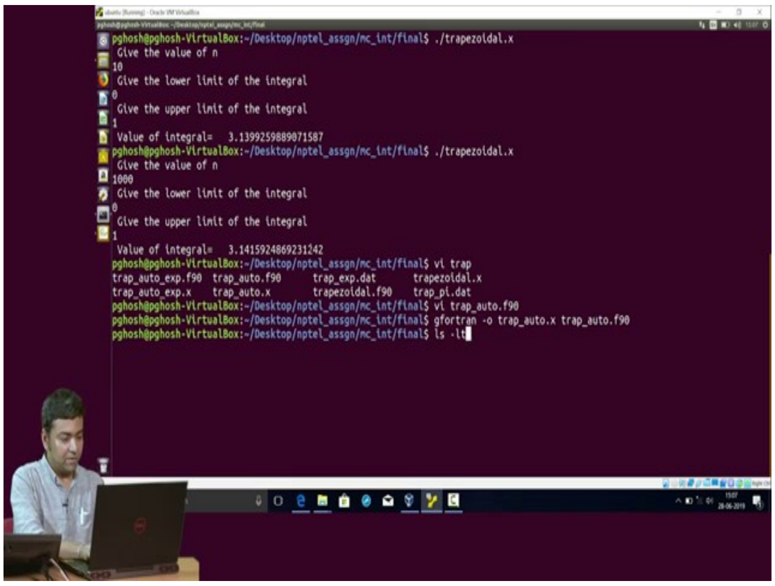
(Refer Slide Time: 25:45)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls -ltr
total 528
-rw-rw-r-- 1 pghosh pghosh   1178 Jun 27 22:10 trapezoidal.f90
-rw-rw-r-- 1 pghosh pghosh   1417 Jun 27 22:27 trap_auto.f90
-rwxrwxr-x 1 pghosh pghosh  13520 Jun 27 22:27 trap_auto.x
-rw-rw-r-- 1 pghosh pghosh    891 Jun 27 22:27 trap_pi.dat
-rw-rw-r-- 1 pghosh pghosh   1336 Jun 27 22:30 test_random_number.f90
-rwxrwxr-x 1 pghosh pghosh  13904 Jun 27 22:30 test_random_number.x
-rw-rw-r-- 1 pghosh pghosh   5247 Jun 27 22:37 scatter.dat
-rw-rw-r-- 1 pghosh pghosh 144000 Jun 27 22:37 ck.dat
-rw-rw-r-- 1 pghosh pghosh   1392 Jun 27 22:44 trap_auto_exp.f90
-rwxrwxr-x 1 pghosh pghosh  13656 Jun 27 22:44 a.out
-rwxrwxr-x 1 pghosh pghosh  13656 Jun 27 22:44 trap_auto_exp.x
-rw-rw-r-- 1 pghosh pghosh    469 Jun 27 22:44 trap_exp.dat
-rw-rw-r-- 1 pghosh pghosh    771 Jun 27 22:45 crude_mc_int.f90
-rw-rw-r-- 1 pghosh pghosh   1071 Jun 27 22:58 crude_mc_int_exp.f90
-rwxrwxr-x 1 pghosh pghosh  13648 Jun 27 22:59 crude_mc_int_exp.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 22:59 exp_crude_mc.dat
-rw-rw-r-- 1 pghosh pghosh    926 Jun 27 23:01 accep_rej_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13616 Jun 27 23:02 accep_rej_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:02 exp_accep_rej.dat
-rw-rw-r-- 1 pghosh pghosh   1228 Jun 27 23:23 mult_dim_brute_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13704 Jun 27 23:23 mult_dim_brute_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:24 mult_brute_mc.dat
-rw-rw-r-- 1 pghosh pghosh   1519 Jun 27 23:28 mult_dim_imp_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13800 Jun 27 23:29 mult_dim_imp_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:31 mult_imp_mc.dat
-rw-rw-r-- 1 pghosh pghosh  54199 Jun 27 23:43 num_int_eg.pptx
-rw-rw-r-- 1 pghosh pghosh 116081 Jun 27 23:43 num_int_eg.pdf
-rwxrwxr-x 1 pghosh pghosh  13456 Jun 28 15:05 trapezoidal.x
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$
```

So, how do I compile the code? Here I am using gfortran compiler. So, gfortran minus o trapezoidal dot x, that is executable file that needs to be created and then I have I give the FORTRAN program name. So, once it is compiled I do a ls minus ltr and then I see that a new file has been created which is the trapezoidal dot x which contains my executable file.

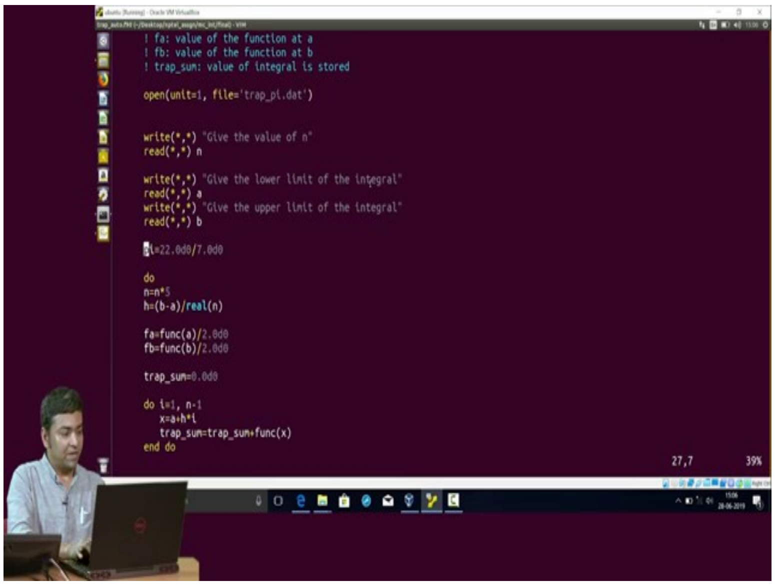
(Refer Slide Time: 26:22)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ./trapezoidal.x
 Give the value of n
10
 Give the lower limit of the integral
0
 Give the upper limit of the integral
1
 Value of integral=   3.1399259889071587
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ./trapezoidal.x
 Give the value of n
1000
 Give the lower limit of the integral
0
 Give the upper limit of the integral
1
 Value of integral=   3.1415924869231242
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trap
trap_auto_exp.f90  trap_auto.f90      trap_exp.dat       trapezoidal.x
trap_auto_exp.x    trap_auto.x        trapezoidal.f90    trap_pi.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ vi trap_auto.f90
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ gfortran -o trap_auto.x trap_auto.f90
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls -lt
```

So, now I will run it and we will see what the code gives. So, this is my trapezoidal. So, I give first a very small value of n. And then it asks for the lower limit of the integral which is 0 in my case and then the upper limit which is 1, and then I get the value of the integral in the following way. So, this is my evaluated value of the integral which is very close to the value of the pi which is 3.14 something. So, you see if we now increase the number of grid points say from 10 I make it to 1000 and redo the calculation what I will see is this value is slowly going closer and closer to the actual value. So, the same thing I have automated it and this is written in this code trapezoidal auto dot f 90.

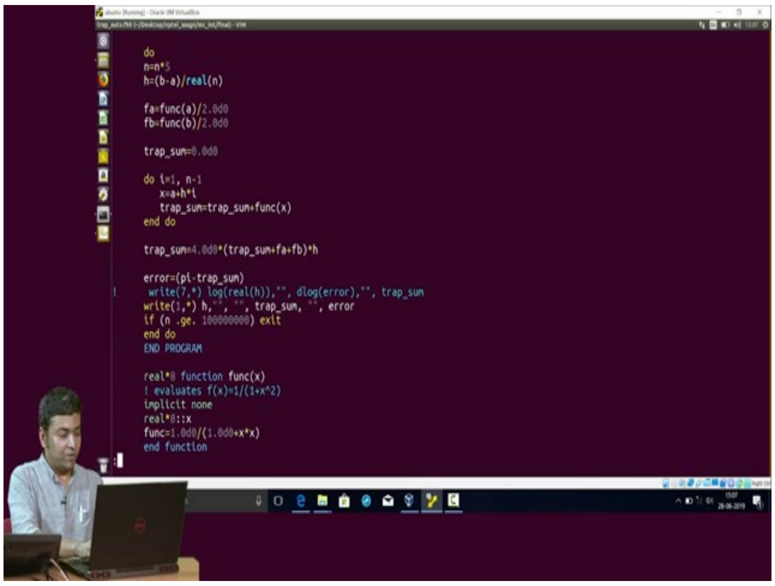
(Refer Slide Time: 27:21)



```
! fa: value of the function at a
! fb: value of the function at b
! trap_sum: value of integral is stored

open(unit=1, file='trap_pi.dat')


write(*,*) "Give the value of n"
read(*,*) n

write(*,*) "Give the lower limit of the integral"
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b

pi=22.0d0/7.0d0

do
n=n*5
h=(b-a)/real(n)

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0

do i=1, n-1
   x=a+h*i
   trap_sum=trap_sum+func(x)
end do
```

So, what it is basically the same code. What I have done is this thing is evaluated, I increase the value of n by a factor of 5, so n becomes n into 5.

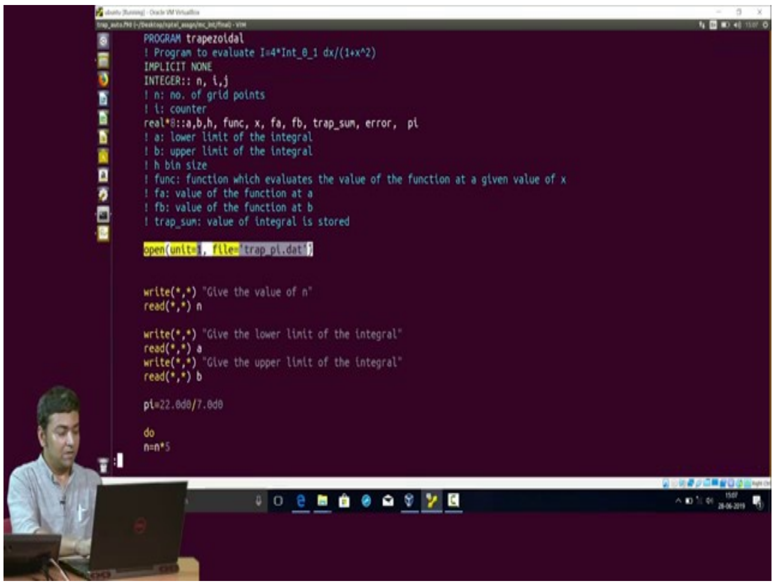
(Refer Slide Time: 27:34)



```
do
n=n*5
h=(b-a)/real(n)

fa=func(a)/2.0d0
fb=func(b)/2.0d0

trap_sum=0.0d0

do i=1, n-1
   x=a+h*i
   trap_sum=trap_sum+func(x)
end do

trap_sum=4.0d0*(trap_sum+fa+fb)*h

error=(pi-trap_sum)
! write(7,*) log(real(h)),"", dlog(error),"", trap_sum
write(1,*) h,"", "", trap_sum, "", error
if (n .ge. 100000000) exit
end do
END PROGRAM

real*8 function func(x)
! evaluates f(x)=1/(1+x^2)
implicit none
real*8::x
func=1.0d0/(1.0d0+x*x)
end function
```

And then this loop is executed till n is larger than this particular number. So, this is the only change. So, that I do not need to give in the value of n every time and also what I am doing is I am printing the bin size h, then the value of the integral and then the error. And the error I am calculating as the actual value of pi minus the computed value of the integral.

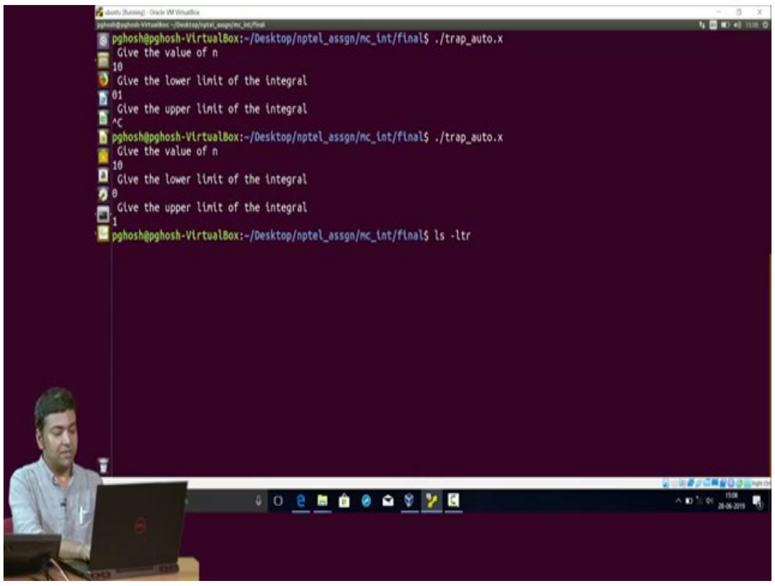
(Refer Slide Time: 28:01)



```
PROGRAM trapezoidal
! Program to evaluate I=4*Int_0_1 dx/(1+x^2)
IMPLICIT NONE
INTEGER:: n, i,j
! n: no. of grid points
! i: counter
real*8::a,b,h, func, x, fa, fb, trap_sum, error,  pi
! a: lower limit of the integral
! b: upper limit of the integral
! h bin size
! func: function which evaluates the value of the function at a given value of x
! fa: value of the function at a
! fb: value of the function at b
! trap_sum: value of integral is stored

open(unit=1, file='trap_pi.dat')


write(*,*) "Give the value of n"
read(*,*) n

write(*,*) "Give the lower limit of the integral"
read(*,*) a
write(*,*) "Give the upper limit of the integral"
read(*,*) b

pi=22.0d0/7.0d0

do
n=n*5
```

And all this thing I am storing in a file which contains this name trap underscore pi dot that using this ‘open’ attribute, ‘open’ keyword of the of fortran. So, again what I need to do is I need to compile it. So, it is the same way minus o trap underscore auto dot dot x, then trap underscore auto dot f 90. So, after compile, getting compiled we have this new executable which is being produced here which is trap underscore auto dot x.

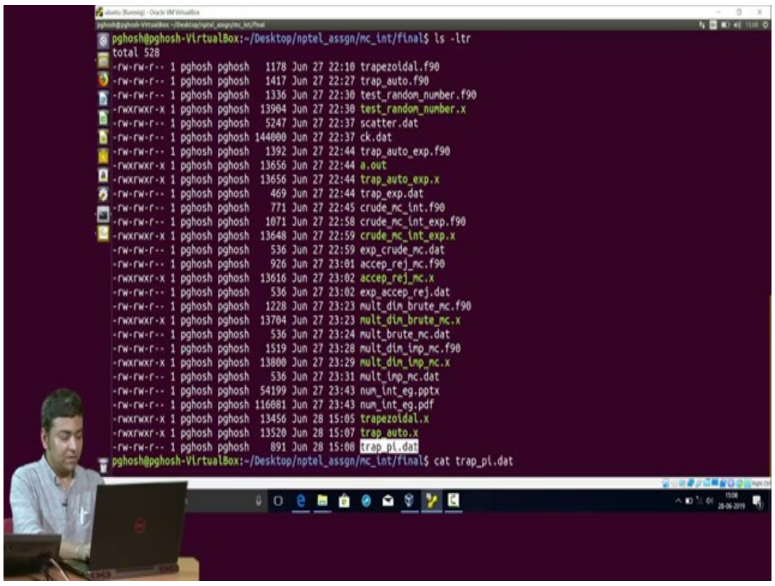
(Refer Slide Time: 28:44)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ./trap_auto.x
 Give the value of n
10
 Give the lower limit of the integral
01
 Give the upper limit of the integral
^C
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ./trap_auto.x
 Give the value of n
10
 Give the lower limit of the integral
0
 Give the upper limit of the integral
1
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls -ltr
```

And what I do is now I run it trap underscore auto dot x. So, I give a starting value of n 10, then the lower.... sorry I made a mistake in giving the lower limit..., so I will run it again. So, I give a value of 10 and the lower limit is 0, the upper limit is 1 and then it will take some time because it is doing for now several values of the integral and it is done.

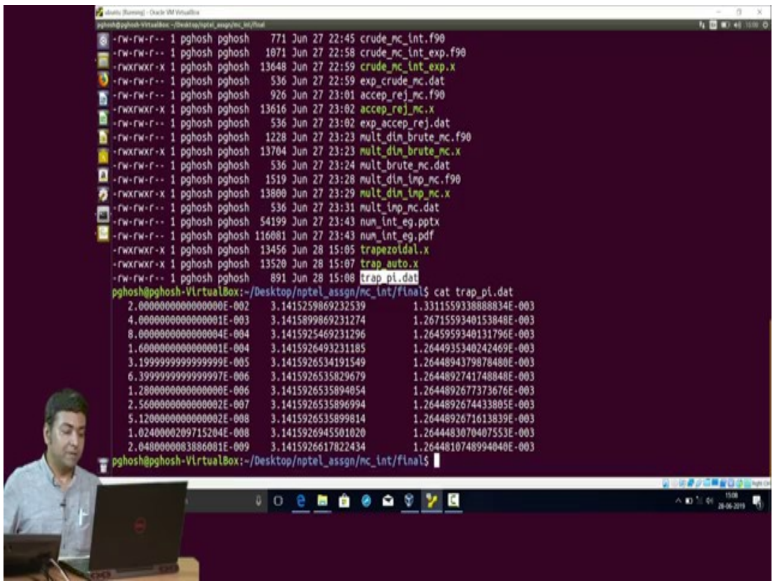
(Refer Slide Time: 29:17)



```
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ ls -ltr
total 528
-rw-rw-r-- 1 pghosh pghosh   1178 Jun 27 22:10 trapezoidal.f90
-rw-rw-r-- 1 pghosh pghosh   1417 Jun 27 22:27 trap_auto.f90
-rw-rw-r-- 1 pghosh pghosh   1336 Jun 27 22:30 test_random_number.f90
-rwxrwxr-x 1 pghosh pghosh  13904 Jun 27 22:30 test_random_number.x
-rw-rw-r-- 1 pghosh pghosh   5247 Jun 27 22:37 scatter.dat
-rw-rw-r-- 1 pghosh pghosh 144000 Jun 27 22:37 ck.dat
-rw-rw-r-- 1 pghosh pghosh   1392 Jun 27 22:44 trap_auto_exp.f90
-rwxrwxr-x 1 pghosh pghosh  13656 Jun 27 22:44 a.out
-rwxrwxr-x 1 pghosh pghosh  13656 Jun 27 22:44 trap_auto_exp.x
-rw-rw-r-- 1 pghosh pghosh    469 Jun 27 22:44 trap_exp.dat
-rw-rw-r-- 1 pghosh pghosh    771 Jun 27 22:45 crude_mc_int.f90
-rw-rw-r-- 1 pghosh pghosh   1071 Jun 27 22:58 crude_mc_int_exp.f90
-rwxrwxr-x 1 pghosh pghosh  13648 Jun 27 22:59 crude_mc_int_exp.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 22:59 exp_crude_mc.dat
-rw-rw-r-- 1 pghosh pghosh    926 Jun 27 23:01 accep_rej_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13616 Jun 27 23:02 accep_rej_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:02 exp_accep_rej.dat
-rw-rw-r-- 1 pghosh pghosh   1228 Jun 27 23:23 mult_dim_brute_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13704 Jun 27 23:23 mult_dim_brute_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:24 mult_brute_mc.dat
-rw-rw-r-- 1 pghosh pghosh   1519 Jun 27 23:28 mult_dim_imp_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13800 Jun 27 23:29 mult_dim_imp_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:31 mult_imp_mc.dat
-rw-rw-r-- 1 pghosh pghosh  54199 Jun 27 23:43 num_int_eg.pptx
-rw-rw-r-- 1 pghosh pghosh 116081 Jun 27 23:43 num_int_eg.pdf
-rwxrwxr-x 1 pghosh pghosh  13456 Jun 28 15:05 trapezoidal.x
-rwxrwxr-x 1 pghosh pghosh  13520 Jun 28 15:07 trap_auto.x
-rw-rw-r-- 1 pghosh pghosh    891 Jun 28 15:08 trap_pi.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ cat trap_pi.dat
```

Then if I do ls minus ltr which basically lists gives a list of the file as per the timestamp. So, I see that a new file has been created.

(Refer Slide Time: 29:32)



```
-rw-rw-r-- 1 pghosh pghosh    771 Jun 27 22:45 crude_mc_int.f90
-rw-rw-r-- 1 pghosh pghosh   1071 Jun 27 22:58 crude_mc_int_exp.f90
-rwxrwxr-x 1 pghosh pghosh  13648 Jun 27 22:59 crude_mc_int_exp.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 22:59 exp_crude_mc.dat
-rw-rw-r-- 1 pghosh pghosh    926 Jun 27 23:01 accep_rej_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13616 Jun 27 23:02 accep_rej_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:02 exp_accep_rej.dat
-rw-rw-r-- 1 pghosh pghosh   1228 Jun 27 23:23 mult_dim_brute_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13704 Jun 27 23:23 mult_dim_brute_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:24 mult_brute_mc.dat
-rw-rw-r-- 1 pghosh pghosh   1519 Jun 27 23:28 mult_dim_imp_mc.f90
-rwxrwxr-x 1 pghosh pghosh  13800 Jun 27 23:29 mult_dim_imp_mc.x
-rw-rw-r-- 1 pghosh pghosh    536 Jun 27 23:31 mult_imp_mc.dat
-rw-rw-r-- 1 pghosh pghosh  54199 Jun 27 23:43 num_int_eg.pptx
-rw-rw-r-- 1 pghosh pghosh 116081 Jun 27 23:43 num_int_eg.pdf
-rwxrwxr-x 1 pghosh pghosh  13456 Jun 28 15:05 trapezoidal.x
-rwxrwxr-x 1 pghosh pghosh  13520 Jun 28 15:07 trap_auto.x
-rw-rw-r-- 1 pghosh pghosh    891 Jun 28 15:08 trap_pi.dat
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$ cat trap_pi.dat
   2.0000000000000000E-002    3.1415259869232539         1.3311559338888834E-003
   4.0000000000000001E-003    3.1415899869231274         1.2671559340153848E-003
   8.0000000000000004E-004    3.1415925469231296         1.2645959340131796E-003
   1.6000000000000001E-004    3.1415926493231185         1.2644935340242469E-003
   3.1999999999999999E-005    3.1415926534191549         1.2644894379878480E-003
   6.3999999999999997E-006    3.1415926535829679         1.2644892741748848E-003
   1.2800000000000000E-006    3.1415926535894054         1.2644892677373676E-003
   2.5600000000000002E-007    3.1415926535896994         1.2644892674433805E-003
   5.1200000000000002E-008    3.1415926535899814         1.2644892671613839E-003
   1.0240000209715204E-008    3.1415926945501020         1.2644483070407553E-003
   2.0480000083886081E-009    3.1415926617822434         1.2644810748994040E-003
pghosh@pghosh-VirtualBox:~/Desktop/nptel_assgn/mc_int/final$
```

So, so if I grip out the contents of this file. So, this is what I have. So, this, so my first column is if you remember its h, the second column is the value of the integral and the third column is my error. So, as you see if we go smaller and smaller value of h the error is decreasing. So, this is how trapezoidal rule is used to compute the value of the integral of this function.