

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

Lecture - 11
Numerical Integration Part 06

So, far we have seen how to do Numerical Integration, in particular, multi dimensional integration numerically, using the stochastic Monte Carlo integration base method. So, far what we have been doing is we have been doing it in a very brute way. So, in order to get good converged value of the integral with few random numbers or few Monte Carlo steps, we need to improve the present or the existing method which we are doing.

(Refer Slide Time: 00:57)

Acceptance / Rejection Method

Affen von Neumann

$$I \approx \int_0^3 \exp(x) dx$$

Hit or Miss Method


uniform dist $[0, 1]$

$x \in [0, 3]$
 $y \in [0, e^3]$

$x = 3 * \text{ran}$
 $y = \exp(3) * \text{ran}$
 $s = 0$
 If $y < \exp(x)$ $s = s + 1$
 Repeat N no. of times

Fraction of #'s in shaded region = $\frac{s}{N}$

$$I \approx 3 * \exp(3) * \frac{s}{N}$$



So, one such method which helps in improving the sampling is called a acceptance or the acceptance rejection method. It is also called the hit or miss method. So, it is based on the famous scientist Von Neumann. So, let us let me try to explain what this method is through a through an example. So, the example which I will be using is basically an one dimensional integral. The reason I am using a one dimensional integral though I have already said before, is that one dimensional integrals with Monte Carlo is not very efficient, that is because one can then visualize the things much more clearly rather than

trying to visualize or imagine a multi dimensional space. So, what I am trying to do is, I am trying to evaluate this integral $\int_0^3 e^x dx$ which equals to integration 0 to 3 exponential x d x.

So, this is the integral I am trying to evaluate. So, if I plot this function e^x to the power x. So, what it will look like is the following. So, I have x here y axis here. So, at x equals to 0. So, e^0 is 1. So, it will go from 1 from here and it goes all the way up here. So, this is my x equals to 0 and this is my x equals to 3. Similarly here my $f(x)$ is equal to 1 and here my $f(x)$ is equal to e^3 .

So, what is done in this method is so, I consider this rectangle. So, my rectangle is defined by x varying from 0 to 3 and y belonging from 0 to e^3 . And then what I do is I generate two random numbers, I generate a random number x which is 3 times my original random number. So, remember so, my Fortran random number generator returns me uniform distribution random the set of random numbers with uniform distribution between 0 and 1 .

So, for my x I need to scale this distribution from 0 to 3. So, I need to map these to a distribution to a set of random members which spans from 0 to 3. So, the way to do it is 3 into this random similarly for the y also I need to map. So, this is for x; for y also I need to map this distribution to a set of numbers which would lie between 0 to e^3 . So, what I do is I take y equals to exponential to the power 3 into my random number. So now, I have got two random numbers y and x. Then what I do is I check whether my this random number y, the second random number which I have generated here, whether it lies inside this shaded area or it is outside.

So, how do I check this? I use the if statement. So, I check if y is less than exponential x. Reason I am taking exponential x is because this curve is given by the equation e^x to the power x. So, if y is less than x what it will mean is that random number is somewhere in this shaded region and I include that number in my count. So, what I have to do is I have to keep a counter, and to find out how many such random numbers are there in the shaded region. So, suppose such counter is given by s. So, I increase my counter to, s equals to s plus 1.

So, I start with s equals to 0. So, I do what this I repeat this step these therefore, say n number of times. So, what I will get is, fraction of numbers that lie in this shaded region, which is my area of the curve and that is given fraction of numbers in shaded region and

that is given by ratio of s by the total number. So, once I know the fraction of such points lying in the shaded region. So, what I need to do is, find out the area, find out I need to multiply this fraction with the total area of this rectangle. So, then that will give me the estimate of this area over under the curve.

So, my value of the integral will be equal to 3 that is the length of the rectangle into exponential 3 that is the breath of the rectangle and the fraction of points which falls under this shaded region. So, this is my estimate of the integral. So, the reason we are calling it as the acceptance rejection method or hit and miss method is because of the following. So, what we are doing is we are generating random number and then checking whether it is within the concerned area or within the concerned volume. If that particular random number do not lie within the volume of interest or in this case the area of interest we are throwing away that random number.

So, we are accepting some and we are rejecting other and as a result this method is called the acceptance rejection method rather than we generating a set of random numbers over all this area, we are just now using a fraction of them to evaluate the integral. So, this is what is the summery of the acceptance rejection method, but still as we will see that this method is also very brutal. If your function is not a well behaved one it you need to generate still a large number of random numbers to have a reasonable estimate.

(Refer Slide Time: 08:59)

The image shows a presentation slide with the following content:

Trapezoidal Integration

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

Monte Carlo Integration

$I = \int_0^3 e^x dx$

- 1) With Trapezoidal rule
- 2) With brute force monte carlo
- 3) With acceptance rejection method

Below the slide, a terminal window displays a grid of random numbers:

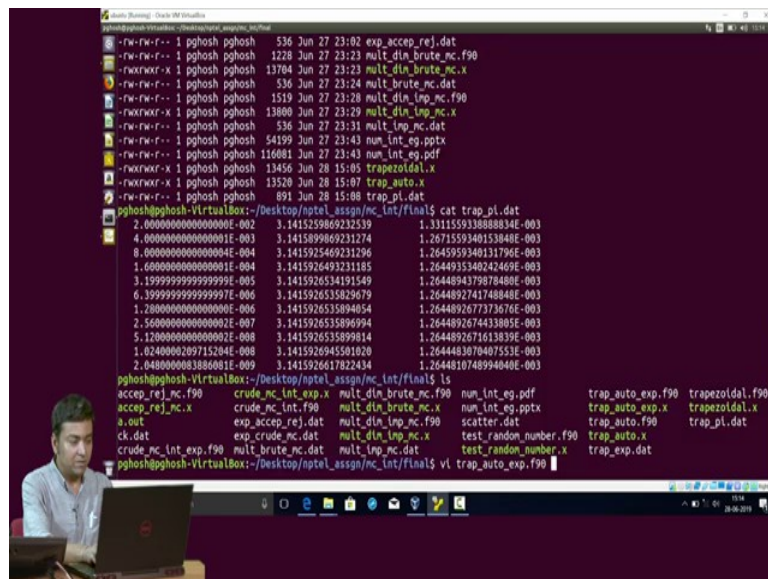
4.1000000000000000E-004	2.1492269492261100	1.1094722540246400E-002
3.1999999999999999E-005	3.1415926534191549	1.2644894379878480E-003
6.3999999999999997E-006	3.1415926535829679	1.2644892741748848E-003
1.2800000000000000E-006	3.1415926535894054	1.2644892677373676E-003
2.5600000000000002E-007	3.1415926535896994	1.2644892674433805E-003
5.1200000000000002E-008	3.1415926535899814	1.2644892671613839E-003
1.0240000289715204E-008	3.1415926535891020	1.2644830784097553E-003
2.048000083886081E-009	3.1415926617822434	1.2644810748994040E-003

The terminal prompt is `pgsh@pgsh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$`.

So, now we will use the Monte Carlo integration how using Monte Carlo integration we can numerically evaluate this integral. So, what we want to do is, we want to find out the value of the integral e^x to the power x with a lower limit from 0 to 3. So, this we can solve analytically and we know that the exact value is e^3 minus 1. So, what we will do is, first we will just solve it with the trapezoidal rule and then we will use two different flavors of Monte Carlo.

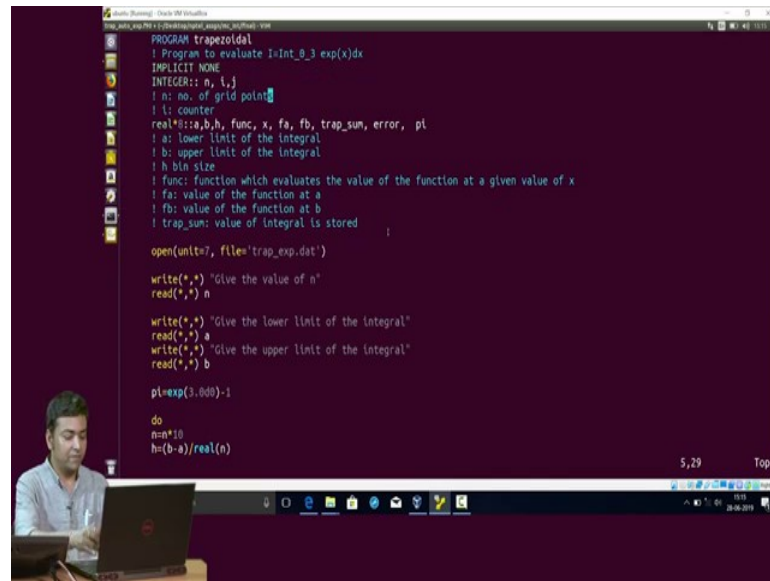
In the first flavor of Monte Carlo this one this brute force Monte Carlo we will just use the conventional philosophy that the integral of a function in m th dimension gives me the volume in at m n close by that function in the m plus 1th dimension. So, that is what is this brute force Monte Carlo and then we will again evaluate the same thing with the acceptance rejection method. So, let us first see the code for then evaluating the integral within the trapezoidal rule.

(Refer Slide Time: 10:10)



So, that is contained in this file called trap underscore x where is contained let us see what is the file name. So, the file name is trap underscore auto underscore e x p dot f 90.

(Refer Slide Time: 10:35)



```
PROGRAM trapezoid1
! Program to evaluate  $\int_{a}^{b} \exp(x) dx$ 
IMPLICIT NONE
INTEGER:: n, i, j
! n: no. of grid point
! i: counter
real:: 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=7, file='trap_exp.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

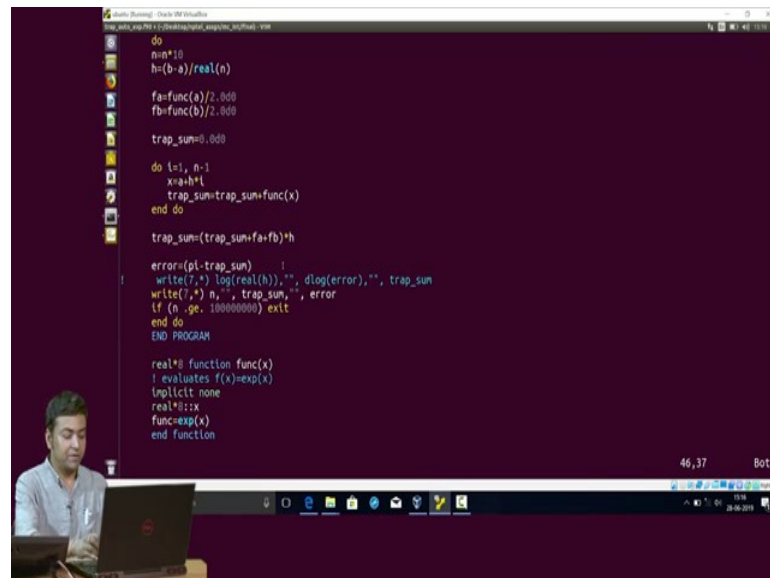
pi=exp(3.000)-1

do
n=n*10
h=(b-a)/real(n)


```

So, basically what I have done is I have used same file with just modifying my function. So, this will be integration 0 to 3 just to correct it, then we have e^x exponential $x dx$. So, this is the function which I am trying to integrate. As before these part is all same the only change which I have made here is I have change the function.

(Refer Slide Time: 11:12)



```
do
n=n*10
h=(b-a)/real(n)

fa=func(a)/2.000
fb=func(b)/2.000

trap_sum=0.000

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

trap_sum=(trap_sum+fa+fb)*h

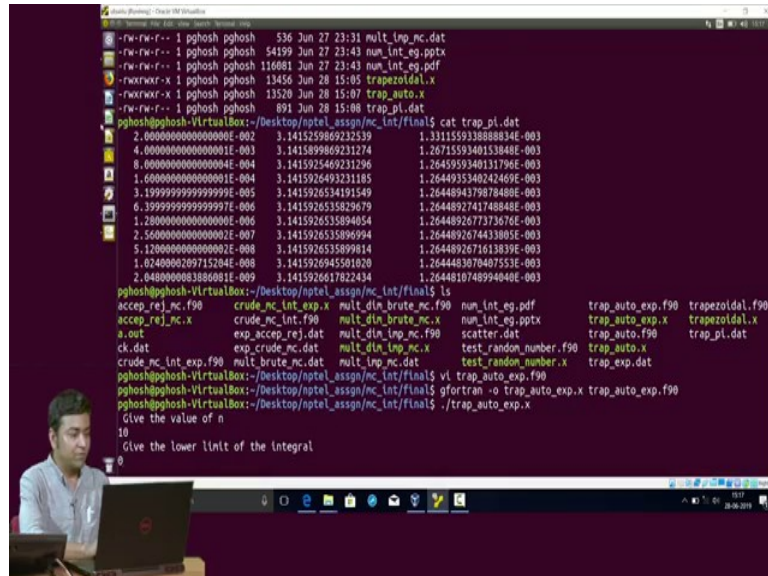
error=(pi-trap_sum)
write(7,*) log(real(h)), " ", dlog(error), " ", trap_sum
write(7,*) n, " ", trap_sum, " ", error
if (n .ge. 100000000) exit
end do
END PROGRAM

real*8 function func(x)
! evaluates f(x)=exp(x)
implicit none
real*8:: x
func=exp(x)
end function
```

So, instead of the evaluating the function 4 by 1 plus x square, what I will do is I will evaluate this function e^x p 2 e x p x and then rest part is completely same. And, then I write down the bin size the value of the integral as a function of the bin size and also the

error has the function of the bin size in this file with this name trap underscore e x p dot dat. So, let us run it and see what we get.

(Refer Slide Time: 11:51)



So, as before we compile it with my gfortran compiler, gfortran minus o trap underscore auto dot underscore e x p dot x then that is the name of the executable look and then trap auto underscore exponential dot f 90.

So, I have compiled it and then I will now I am running it dot slash trap underscore auto underscore exponential dot x. So, I give the starting value of n let me start with the small value say 10 the limit of the integral if you remember it is from 0 to 3 as I have written it here 0 to 3.

(Refer Slide Time: 12:33)

Trapezoidal Integration

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

Monte Carlo Integration $I = \int_0^3 e^x dx$

- 1) With Trapezoidal rule
- 2) With brute force monte carlo
- 3) With acceptance rejection method

```
crude_mc_int_exp.f90  mult_brute_mc.dat  mult_inp_mc.dat  test_random_number.x  trap_exp.dat
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ vi trap_auto_exp.f90
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ gfortran -o trap_auto_exp.x trap_auto_exp.f90
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ./trap_auto_exp.x
Give the value of n
10
Give the lower limit of the integral
0
```

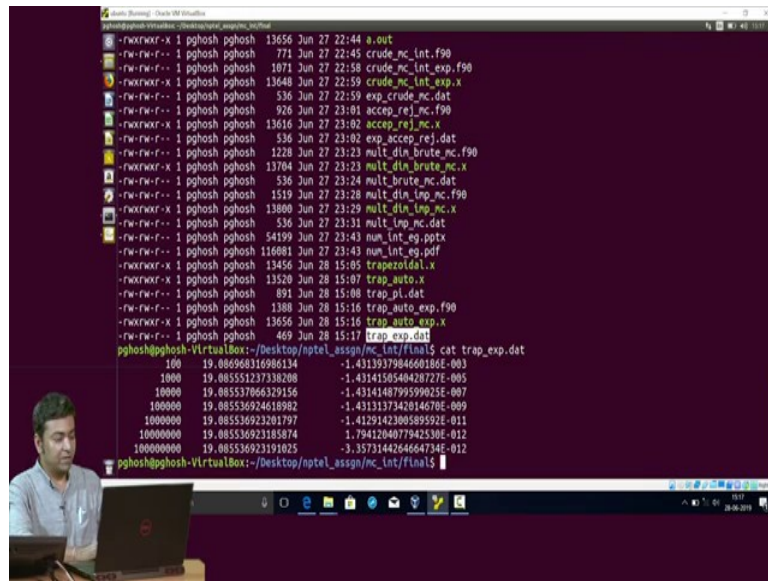
So, the lower limit is 0 the upper limit is 3.

(Refer Slide Time: 12:40)

```
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ls
-rwxr-xr-x 1 pgshosh pgshosh 13456 Jun 28 15:05 trapezoidal.x
-rwxr-xr-x 1 pgshosh pgshosh 13520 Jun 28 15:07 trap_auto.x
-rw-rw-r-- 1 pgshosh pgshosh 891 Jun 28 15:08 trap_pi.dat
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ cat trap_pi.dat
2.0000000000000000E-002  3.141529869232539  1.311155938888834E-003
4.0000000000000000E-003  3.1415899869231274  1.2671559340153848E-003
8.0000000000000000E-004  3.1415925469231296  1.2645959340131796E-003
1.6000000000000000E-004  3.1415926493231185  1.2644935340242469E-003
3.1999999999999999E-005  3.1415926534191549  1.2644894379878480E-003
6.3999999999999997E-006  3.1415926535829679  1.2644892741748848E-003
1.2000000000000000E-006  3.1415926535894854  1.2644892677373676E-003
2.5000000000000000E-007  3.1415926535896994  1.2644892674433885E-003
5.1200000000000000E-008  3.1415926535899814  1.26448926761613839E-003
1.0240000000000000E-008  3.1415926945501020  1.2644483070407553E-003
2.0480000000000000E-009  3.1415926617822434  1.2644810748994404E-003
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ls
accep_rej.f90  crude_mc_int_exp.x  mult_din_brute_mc.f90  num_int_eg.pdf  trap_auto_exp.f90  trapezoidal.f90
accep_rej.x   crude_mc_int.f90      mult_din_brute_mc.x  num_int_eg.pptx  trap_auto_exp.x   trap_auto_exp.f90
a.out        exp_accep_rej.dat    mult_din_inp_mc.f90  scatter.dat      trap_auto.f90     trap_pi.dat
ck.dat       exp_crude_mc.dat    mult_din_inp_mc.x   test_random_number.f90  trap_auto.x
crude_mc_int_exp.f90  mult_brute_mc.dat  mult_inp_mc.dat     test_random_number.x  trap_exp.dat
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ vi trap_auto_exp.f90
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ gfortran -o trap_auto_exp.x trap_auto_exp.f90
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ./trap_auto_exp.x
Give the value of n
10
Give the lower limit of the integral
0
Give the upper limit of the integral
3
pgshosh@pgshosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ls -ltr
```

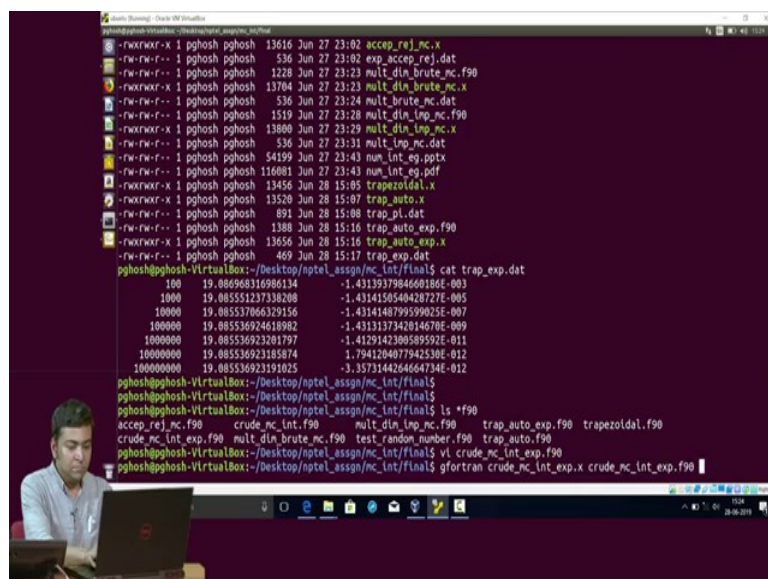
So, it will take some time to evaluate and then. So, we have to wait till it is done because it is doing for several values of the integral.

(Refer Slide Time: 12:49)



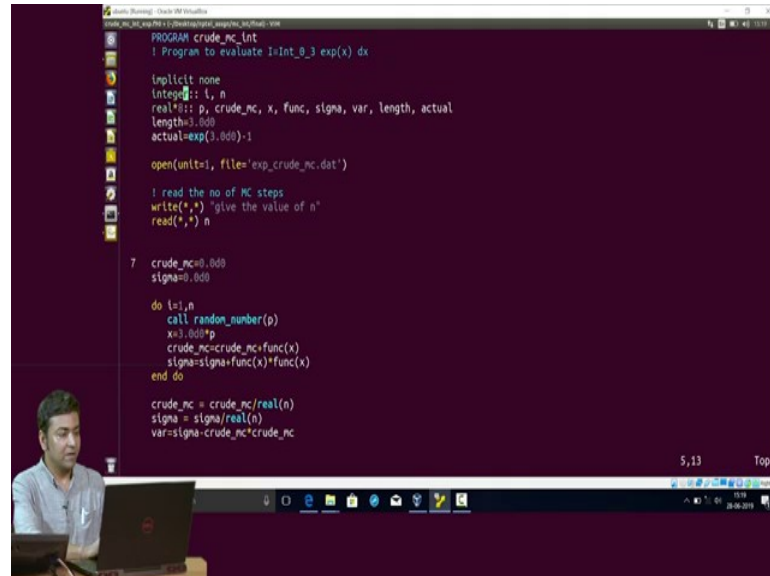
So, now it is done and then what I have is the file containing the data `trap_exp.dat`. So, like before if I see what are the contents of the file let me just grab them out and this is what I get. So, these are the that is now here I do not have it as a bin size rather I have written it as a function of the number of grid points. So, as I increase the grid points, you see that the error is decreasing very fast. So, with the hundred grid points I am already doing very good with a error of 10 to the power minus 3, but the movement I went to this grid size the error decreases to 10 to the power minus 12. Now the same thing how will I do it using Monte Carlo integration?

(Refer Slide Time: 13:41)



So, the code that does that is the following. So, the name of the code is crude underscore m c underscore int underscore exp. So, let us open this file.

(Refer Slide Time: 14:04)



```
PROGRAM crude_mc_int
! Program to evaluate  $\int_{0}^{3} \exp(x) dx$ 

implicit none
integer :: i, n
real :: p, crude_mc, x, func, sigma, var, length, actual
length=3.0d0
actual=exp(3.0d0)-1

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

! read the no of MC steps
write(*,*) 'give the value of n'
read(*,*) n

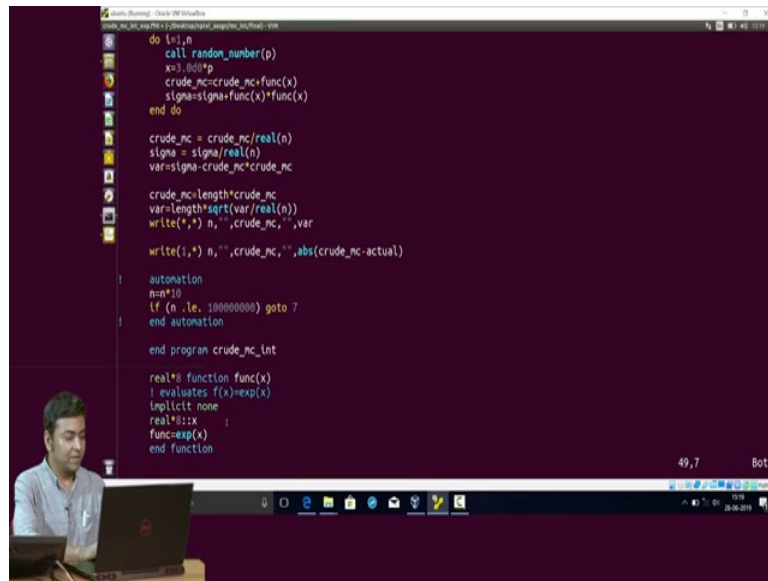
7 crude_mc=0.0d0
sigma=0.0d0

do i=1,n
  call random_number(p)
  x=3.0d0*p
  crude_mc=crude_mc+func(x)
  sigma=sigma+func(x)*func(x)
end do

crude_mc = crude_mc/real(n)
sigma = sigma/real(n)
var=sigma-crude_mc*crude_mc
```

So, again I here in the comment line, I have to correct this. This is integration 0 to 3, then I have exponential $x dx$ ok. So, what I have here is I have 2 integers the n is the number of points now remember unlike the trapezoidal rule these points will be random points which will be given by which will generate by using my random number generator. And, I is the counter p is the dummy variable used for the subroutine called random underscore number, which returns the random number when I call this random number generator crude underscore m c is the variable where I am storing the value of the integral, then x is a number which basically lies between 0 to 3.

(Refer Slide Time: 15:13)



```
do i=1,n
  call random_number(p)
  x=3.000*p
  crude_nc=crude_nc+func(x)
  sigma=sigma+func(x)*func(x)
end do

crude_nc = crude_nc/real(n)
sigma = sigma/real(n)
var=sigma-crude_nc*crude_nc

crude_nc=length*crude_nc
var=length*sqrt(var/real(n))
write(*,*) n, ", ", crude_nc, ", ", var

write(i,*) n, ", ", crude_nc, ", ", abs(crude_nc-actual)

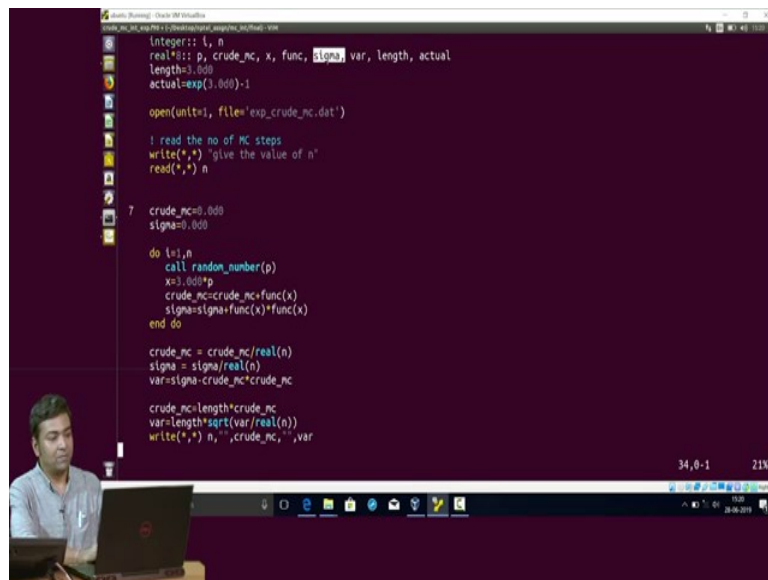
! automation
n=n*10
if (n .le. 100000000) goto 7
end automation

end program crude_nc_int

real*8 function func(x)
! evaluates f(x)=exp(x)
implicit none
real*8::x
func=exp(x)
end function
```

And, then I have my function `func` is a function as we have seen before which evaluates the value of the function e to the power x , when I give the value of x .

(Refer Slide Time: 15:22)



```
integer:: i, n
real*8:: p, crude_nc, x, func, sigma, var, length, actual
length=3.000
actual=exp(3.000)-1

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

! read the no of MC steps
write(*,*) "give the value of n"
read(*,*) n

7 crude_nc=0.000
sigma=0.000

do i=1,n
  call random_number(p)
  x=3.000*p
  crude_nc=crude_nc+func(x)
  sigma=sigma+func(x)*func(x)
end do

crude_nc = crude_nc/real(n)
sigma = sigma/real(n)
var=sigma-crude_nc*crude_nc

crude_nc=length*crude_nc
var=length*sqrt(var/real(n))
write(*,*) n, ", ", crude_nc, ", ", var
```

And then what we have is then `sigma` stores the value of the sigma, `var` stores the value of the variance, `length` stores the total length of the interval over which we want to do the evaluate the integral and since here the it is done from 0 to 3. So, I have set `length` to 3. Again I am using double precision here and `actual` stores the actual value of the integral.

And actual value of the integral as I mentioned before is exponential e to the power 3 minus 1. So, this is my actual value of the integral here I do it for several value of n . So, I want to store the data either I mean I want to store the value of my integral as a function of different values of n . So, that information is stored in this particular file. So, how do I do it? So, first I set this crude underscore m c , where I will be storing the value to 0 and sigma to 0

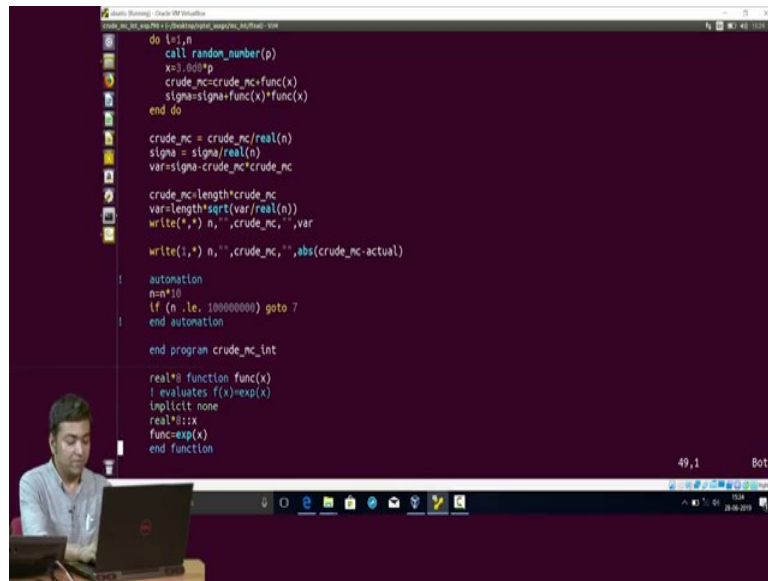
So, once I do that what I do is, I follow these steps n number of times. So, each time what I do is. So, basically what here I am doing is I am computing the average value of the function, but instead of taking a fixed grid I am taking the grid is a random one depending on the random numbers which is generated by my random number generator. So, and also remember that this random number generator what it does is, it gives me random numbers from 0 to 1. So, that random numbers from 0 to 1 I need to distribute it from 0 to 3, and this is how I do it. I just simply multiply it by 3 times the random number p where p is a random is a random number which lies between 0 to 1.

So, this is my new grid point x which is which I find it this way and once I find it this way then what I can do is, I can evaluate using the function I can evaluate the value of the function at this particular x point, similarly and then I keep on adding it up and similarly for the sigma also I do the same thing, but sigma you know it is square of the function which I need to do.

So, because this 1 the integral is the average, now I am computing this I want to compute the variance and so, I need to sigma which is the average of the square of the function. And once this do loop is executed what I do is, I to compute the average and need to divide it by the real n . Here is the number of time. So, I compute the value of the function similarly for sigma and then my a integral the actual value of the integral, I need to multiply it by the volume of which in this case is one dimensional, it is length.

So, length into crude underscore m c and similar way I also compute my variance and once that is done I write it here in the screen and I also write the error in the file the value of n , the number of iterations or the number of times I am evaluating out of the number of random numbers; I am using to make this estimate and then the crude value I mean of the integral and then the error compared to the actual value of the integral.

(Refer Slide Time: 18:56)



```
do i=1,n
  call random_number(p)
  x=3.000*p
  crude_mc=crude_mc+func(x)
  sigma=sigma+func(x)*func(x)
end do

crude_mc = crude_mc/real(n)
sigma = sigma/real(n)
var=sigma-crude_mc*crude_mc

crude_mc=length*crude_mc
var=length*sqrt(var/real(n))
write(*,*) n, " ", crude_mc, " ", var

write(*,*) n, " ", crude_mc, " ", abs(crude_mc-actual)

! automation
n=n*10
if (n .le. 100000000) goto 7
end automation

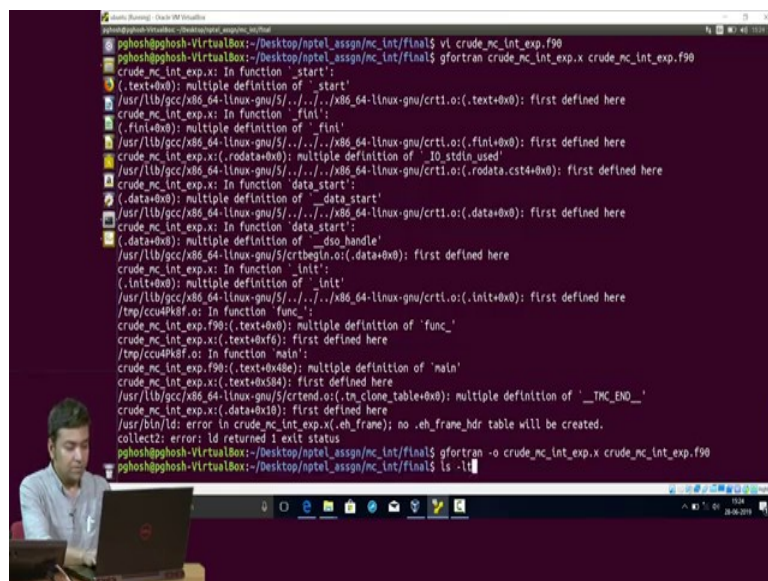
end program crude_mc_int

real*8 function func(x)
! evaluates f(x)=exp(x)
implicit none
real*8::x
func=exp(x)
end function
```

Once this is done what I do is, here I change n to n into 10, I increase it n fold, and then until and unless the n is or as long as n is less than this number, I repeat these steps.

So, I go back to this point go to 7, 7 is this point I again set my this crude underscore mc and sigma to 0 and repeat all the steps. So, once this is done, then it will come out of the loop. So, let us again compile and run the code and see what happens crude m c integration.

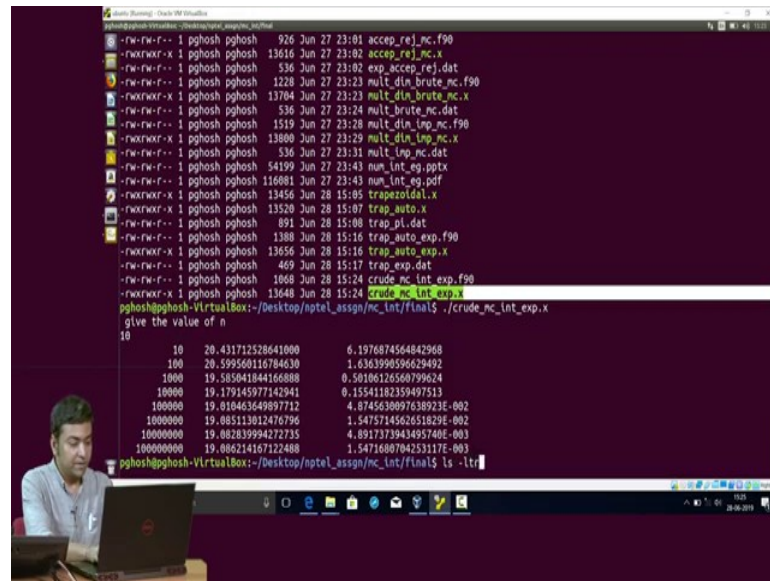
(Refer Slide Time: 19:48)



```
gghosh@gghosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ vi crude_mc_int_exp.f90
gghosh@gghosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ gfortran crude_mc_int_exp.x crude_mc_int_exp.f90
crude_mc_int_exp.x: In function 'start':
(.text+0x0): multiple definition of 'start'
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o:(.text+0x0): first defined here
crude_mc_int_exp.x: In function 'fini':
(.fini+0x0): multiple definition of 'fini'
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o:(.fini+0x0): first defined here
crude_mc_int_exp.x:(.rodata+0x0): multiple definition of '.IO.stdin.used'
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o:(.rodata.cst4+0x0): first defined here
crude_mc_int_exp.x: In function 'data_start':
(.data+0x0): multiple definition of 'data_start'
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o:(.data+0x0): first defined here
crude_mc_int_exp.x: In function 'data_start':
(.data+0x0): multiple definition of 'do_handle'
/usr/lib/gcc/x86_64-linux-gnu/5/crtbegin.o:(.data+0x0): first defined here
crude_mc_int_exp.x: In function 'init':
(.init+0x0): multiple definition of 'init'
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crti.o:(.init+0x0): first defined here
/tmp/ccc4PK8f.o: In function 'func':
crude_mc_int_exp.f90:(.text+0x0): multiple definition of 'func'
crude_mc_int_exp.x:(.text+0x0): first defined here
/tmp/ccc4PK8f.o: In function 'main':
crude_mc_int_exp.f90:(.text+0x48e): multiple definition of 'main'
crude_mc_int_exp.x:(.text+0x584): first defined here
/usr/lib/gcc/x86_64-linux-gnu/5/crtend.o:(.tn_clone_table+0x0): multiple definition of '___TKN_END___'
crude_mc_int_exp.x:(.data+0x10): first defined here
/usr/bin/ld: error in crude_mc_int_exp.x(.eh_frame); no .eh_frame_hdr table will be created.
collect2: error: ld returned 1 exit status
gghosh@gghosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ gfortran -o crude_mc_int_exp.x crude_mc_int_exp.f90
gghosh@gghosh-VirtualBox:~/Desktop/nptel_assign/mc_int/final$ ls -lt
```

Sorry I missed the minus .

(Refer Slide Time: 20:02)

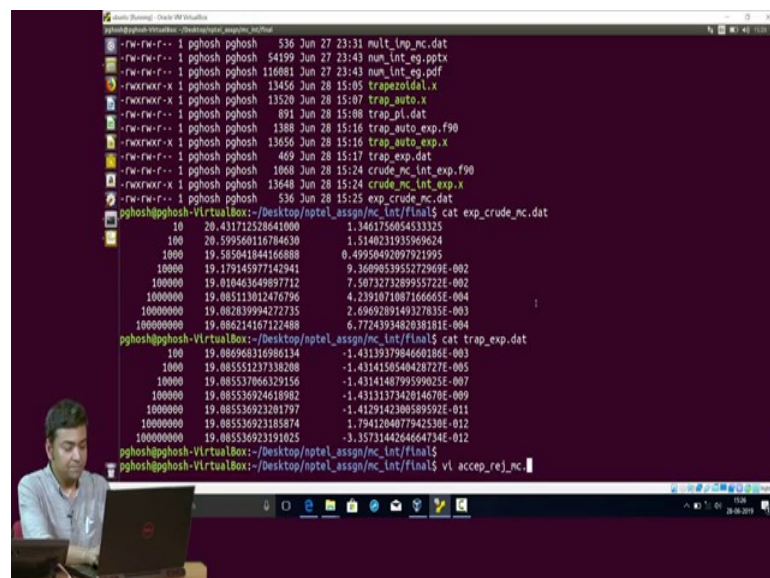


```
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$ ./crude_nc_int_exp.x
give the value of n
10
10      20.431712528641000      6.1976874564842968
100     29.599560110784630      1.6363990596629492
1000    19.585041844166888      0.50106126560799624
10000   19.179145977142941      0.15541182359497513
100000  19.810463649897712      4.8745630097638923E-002
1000000 19.885113012476796      1.547571562651829E-002
10000000 19.88283994272735      4.891737943495740E-003
100000000 19.88624167122488      1.5471680784253117E-003
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$ ls -ltr
```

So, it has produced this executable file this one here. So, let us run this . So, I start with a very small value say 10 and then it automatically does it for several values.

So, to wait for some time for it to come out of the loop and do that and finish the calculation it is done.

(Refer Slide Time: 20:35)



```
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$ cat exp_crude_nc.dat
10      20.431712528641000      1.346175605453325
100     29.599560110784630      1.534021395896624
1000    19.585041844166888      0.49950492097921995
10000   19.179145977142941      9.3609953955272969E-002
100000  19.810463649897712      7.5073273289955722E-002
1000000 19.885113012476796      4.2391071087166665E-004
10000000 19.88283994272735      2.6969289149327835E-003
100000000 19.88624167122488      6.7724393482038181E-004
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$ cat trap_exp.dat
100     19.88696831690634      -1.431393790460280E-003
1000    19.88551237338208      -1.4314150540428727E-005
10000   19.885537866329156      -1.4314148799599025E-007
100000  19.885536924618982      -1.4313137342014670E-009
1000000 19.885536923201797      -1.4129142300589592E-011
10000000 19.885536923185874      1.7941204077942530E-012
100000000 19.885536923191025      -3.3573144264664734E-012
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$
gghosh@pghosh-VirtualBox:~/Desktop/nptel_assign/nc_int/final$ vi accep_rej_mc.
```

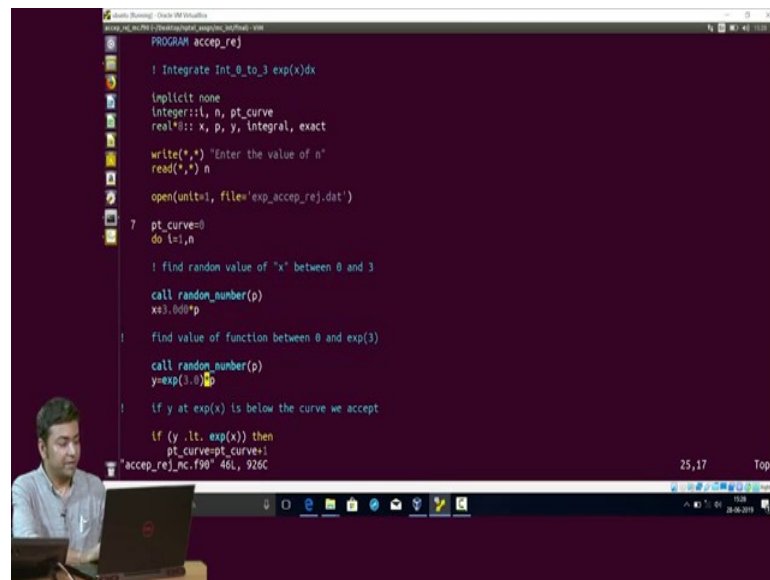
So, if I do `ls -ltr`. So, I have this file which has been produced which contains the value of the integral as a function of the number of random numbers I used to compute the average. So, if I do it. So, this is what I get. Now, let us compare it with my

trapezoidal rule. So, this first part here is the output of the value of my integral as a function of n using the crude Monte Carlo and this one here is the value of the integral for the same thing using the trapezoidal rule.

So, what we see is that, in comparison to this Monte Carlo simulations Monte Carlo integration here the accuracy achieved by the trapezoidal rule is much higher, but this is true in 1 dimension and in one dimension typically your trapezoidal rule is much better and much more faster also compared to the till the Monte Carlos integration. So now, we will do the same integration using the acceptance rejection method. So, in the acceptance rejection method what one needs to do is so, this is the integral which I am evaluating.

So, basically what I will do is, I will consider a square whose origin is a which are formed along the x direction it goes from 0 to 3 and in the y direction it goes from 1 to e to the power 3. So, what I will generate randomly through points inside this particular square and then I will find out, how many points are there which lies and within this curve. So, that fraction gives me the value of the integral.

(Refer Slide Time: 22:30)



```
PROGRAM accep_rej
! Integrate Int_0_to_3 exp(x)dx
implicit none
integer:: i, n, pt_curve
real:: x, y, integral, exact

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

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

7 pt_curve=0
do i=1,n

! Find random value of "x" between 0 and 3
call random_number(p)
x=3.000*p

! find value of function between 0 and exp(3)
call random_number(p)
y=exp(3.0)

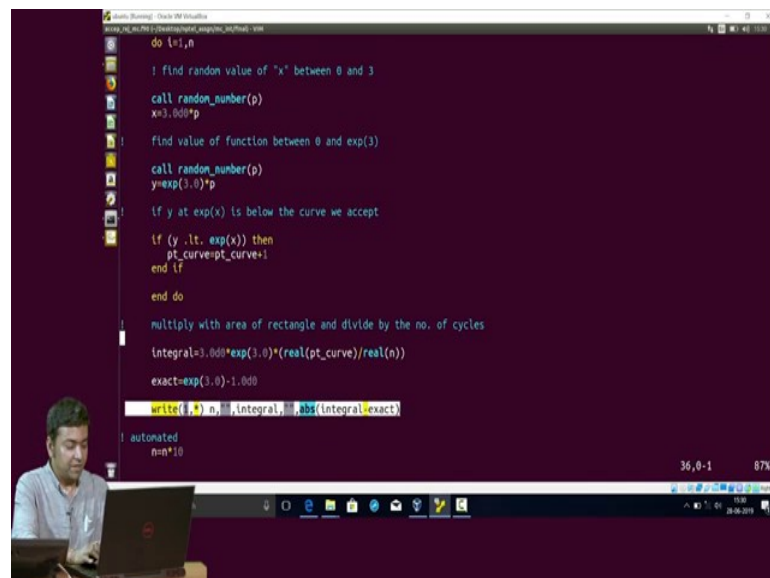
! if y at exp(x) is below the curve we accept
if (y .lt. exp(x)) then
pt_curve=pt_curve+1
"accep_rej_mc.f90" 461, 9266
```

So, again as before what I have here is integer variable set of integer variables I start with i the counter, then I have n which gives the total number of points and then I and this p t underscore curve will calculate how many points will that lie within the curve, then I have the value at of x and then p is the dummy variable from the random number

generator y is the value of the function and integral is the value of the integral and then exact is the value of the analytical value of the integral.

So, I ask a user to send me the value of n and since I want to do it again for several values of n and do it in automated process and I stored them here. So, I said the counter to the number of points that lie within the curve to be 0 each time I want to evaluate this integral for each value of n , then what I do is the first step is I find a random value of x between 0 and 3 and this is done by this two lines. So, first I call a random number a random number generator which gives me the value of a number between 0 and 1 which is stored in p and this number I convert to a number between 0 and 3 by multiplying it by 3 and I store it in x .

(Refer Slide Time: 24:00)



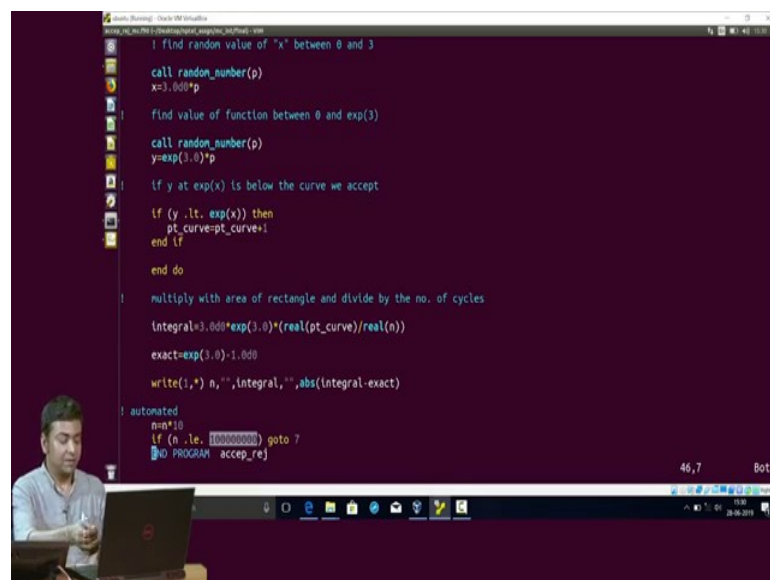
```
do i=1,n
! find random value of "x" between 0 and 3
call random_number(p)
x=3.000*p
! find value of function between 0 and exp(3)
call random_number(p)
y=exp(3.0)*p
! if y at exp(x) is below the curve we accept
if (y .lt. exp(x)) then
pt_curve=pt_curve+1
end if
end do

! multiply with area of rectangle and divide by the no. of cycles
integral=3.000*exp(3.0)*(real(pt_curve)/real(n))
exact=exp(3.0)-1.000
write(*,*) n, integral, abs(integral-exact)
! automated
n=n*10
```

So, once that is done, then I evaluate the value of the function at a random point between 0 and e to the power 3 which is exponential 3, which is the maximum value of the function, that I do it in a similar way as I have done to find a random number x . So, I call a random number p which lies between 0 and 1 and then I multiply the random number to e to the power exponential 3. Now, once I have got these two numbers what I do is, I check whether this random number y sub which is the value of the function any value of the random value of the function that lies between 0 and e to the power 3 whether it is less than that exponential of the random number x .

If it is so, then we can assume there then we can consider that the point is lies within the within the curve enclosed by or the area enclosed by the curve and I take it, I count it. So, I repeat this process several times and I count say if I throw 1000 times such numbers on my square how many such numbers lie within the area enclosed by the curve. So, once I know it, then I can compute the integral in the following way. So, this gives my so, this part is basically the fraction of the points which lies within the curve and then this is the area under the curve form. So, if I do the integral and then what I am doing here is I am also evaluating the exact value of the integral and then I am writing it in this file.

(Refer Slide Time: 25:46)



```
! find random value of "x" between 0 and 3
call random_number(p)
x=3.000*p

! find value of function between 0 and exp(3)
call random_number(p)
y=exp(3.0)*p

! if y at exp(x) is below the curve we accept
if (y .lt. exp(x)) then
  pt_curve=pt_curve+1
end if

end do

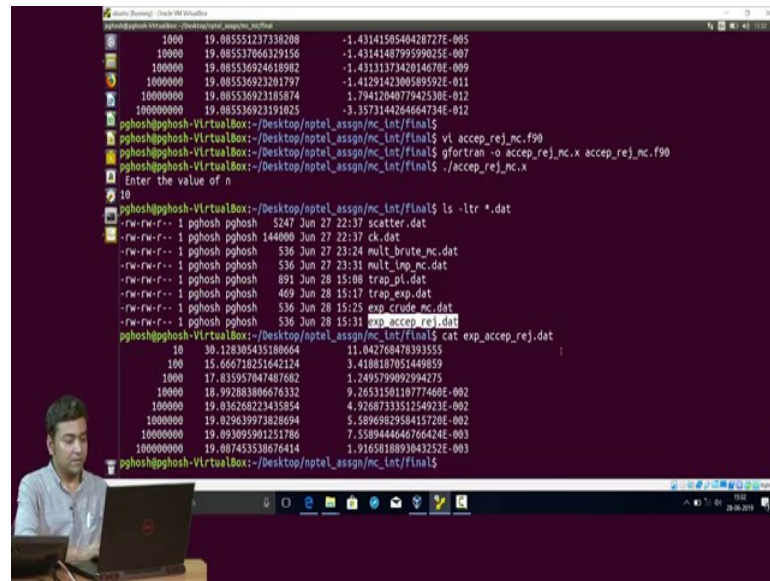
! multiply with area of rectangle and divide by the no. of cycles
Integral=3.000*exp(3.0)*(real(pt_curve)/real(n))
exact=exp(3.0)-1.000

write(1,*) n," ",Integral," ",abs(Integral-exact)

! automated
n=n+10
if (n .le. 1000000) goto 7
@ PROGRAM accep_rej
```

So, and then as before I am automating the process and it is done till the n takes a value of which is larger than this one.

(Refer Slide Time: 25:55)



```
1000 19.88551237338208 -1.4314150540428727E-005
10000 19.885537866329156 -1.4314148799599025E-007
100000 19.885536924618982 -1.4313137342814670E-009
1000000 19.885536923201797 -1.4129142300839592E-011
10000000 19.885536923183874 1.7941204077942530E-012
100000000 19.885536923191825 -3.3573144264664734E-012
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$ vi accep_rej_mc.f90
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$ gfortran -o accep_rej_mc.x accep_rej_mc.f90
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$ ./accep_rej_mc.x
Enter the value of n
10
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$ ls -ltr *.dat
-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 536 Jun 27 23:24 mult_brute_mc.dat
-rw-rw-r-- 1 pghosh pghosh 536 Jun 27 23:31 mult_lm_mc.dat
-rw-rw-r-- 1 pghosh pghosh 891 Jun 28 15:08 trap_pl.dat
-rw-rw-r-- 1 pghosh pghosh 469 Jun 28 15:17 trap_exp.dat
-rw-rw-r-- 1 pghosh pghosh 536 Jun 28 15:25 exp_crude_mc.dat
-rw-rw-r-- 1 pghosh pghosh 536 Jun 28 15:31 exp_accp_rej.dat
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$ cat exp_accp_rej.dat
10 30.128305435180664 11.042760478393555
100 15.466718251642124 3.4188187051449859
1000 17.835957847487682 1.2495799092994275
10000 18.992883806676332 9.265315010777460E-002
100000 19.836268223435854 4.9268733351254923E-002
1000000 19.829639973828694 5.5896982958415720E-002
10000000 19.893095901251786 7.5580444640766424E-003
100000000 19.887453330676414 1.9165818893043252E-003
pghosh@pghosh-VirtualBox:~/Desktop/npTEL_assign/mc_int/final$
```

So, let us compile it as before and see what we get. So, I start with the very means small value and let the code run it will take some time to run it. So, once it is done. So, what we will do is let us see what is the file. So, you see it has produced this particular file. So, let me just copy it and let us see what it has done. So, these are the set of numbers. So, see again this method also the accuracy is given with such a huge number accuracy is few orders of magnitude what is then what we achieved with a trapezoidal rule. So, this is how one does the integration using the acceptance rejection method and the crude Monte Carlo and we have also seen how these compares with the trapezoidal rule for a low dimensional case. So, for a low dimensional case definitely the trapezoidal rule works much better.