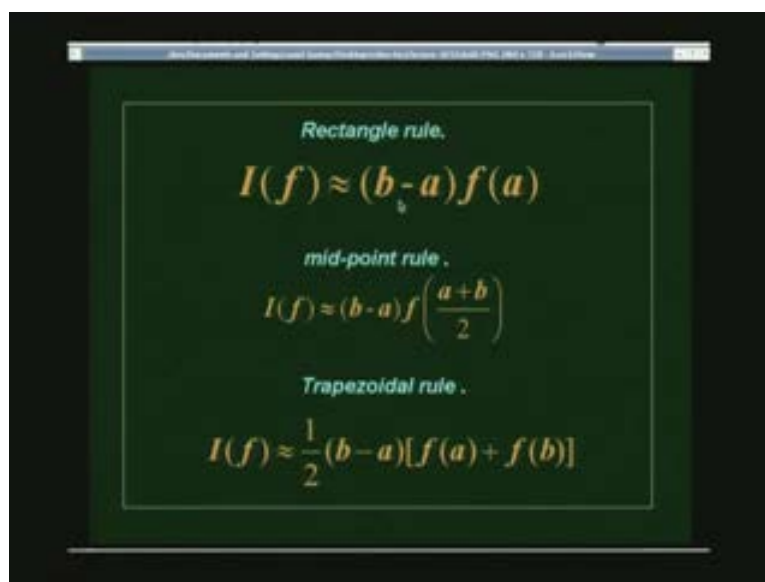


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 30
Numerical Integration: Comparison of
Gaussian Rules

In the last couple of classes we have been looking at the numerical methods to calculate integrals of a function that is we want to evaluate the integrals of a function which is between the limits a to b $\int_a^b f(x) dx$ and we call that as I of f , that is what we have been looking at. So we have been looking at a calculation of this quantity $\int_a^b f(x) dx$ as of for any function f . So we looked at various methods for doing such integrals okay and some of them I will just summarize here is one is the what is called the rectangular rule, okay in this rule we wrote, we approximate the function I the integral of the function by the function evaluated at one point multiplied by the whole interval b minus a , remember b and a are the limits of this integral. So it is integral a to b and then we evaluate the function at the first point a and then multiplied by the interval b minus a .

So that is one method which we learnt and then another method again which involve evaluating the function only at one point which is now **in the** in between a and b that is a plus b by 2 and multiplying it by the interval b minus a that is again another approximation we call that as a midpoint rule. These are the two methods which involve evaluating the function value at one point and then they were other methods which is I have involved for evaluating the function at two points and one is trapezoidal rule, in the trapezoidal rule we approximate the integral of the function in the interval a to b by f of a plus f of b that is evaluated at the two ends divided by 2 into the interval b minus a .

(Refer Slide Time: 2:01)



Rectangle rule.

$$I(f) \approx (b-a)f(a)$$

mid-point rule.

$$I(f) \approx (b-a)f\left(\frac{a+b}{2}\right)$$

Trapezoidal rule.

$$I(f) \approx \frac{1}{2}(b-a)[f(a) + f(b)]$$

So this is 3 methods which we looked at using called the rectangle rule, midpoint rule and trapezoidal rule these are all of methods which is a polynomial of order 0, approximation for a polynomial of order 0 and then we looked at this first two methods polynomial of order 0 and this is 0 th order polynomial, this first order polynomial and then we looked at the next order polynomial which is the Simpson's rule. So now, that involves evaluating the function at 3 different points that is f of a, f of b and f of a plus b by 2. Now you evaluate the function at the 2 ends and also in the middle and then write the formula for the integral as b minus a by 6, f of a plus 4 f of a plus b by 2 plus 4b.

So all these methods the common features is once again you can notice is this that it is the integral become a summation over a few points a function evaluated at a few points multiplied by some appropriate weight factor ,so all of them actually writes it as some w_i f of x_i . So on the first two methods we had only one point and then we had the second the trapezoidal rule had 2 points and the Simpson's rule had 3 points these are all. So all these methods reduces this into have in integral will finally become something like w_i f of x_i summed over i. So that is what the method finally would become, so all of them.

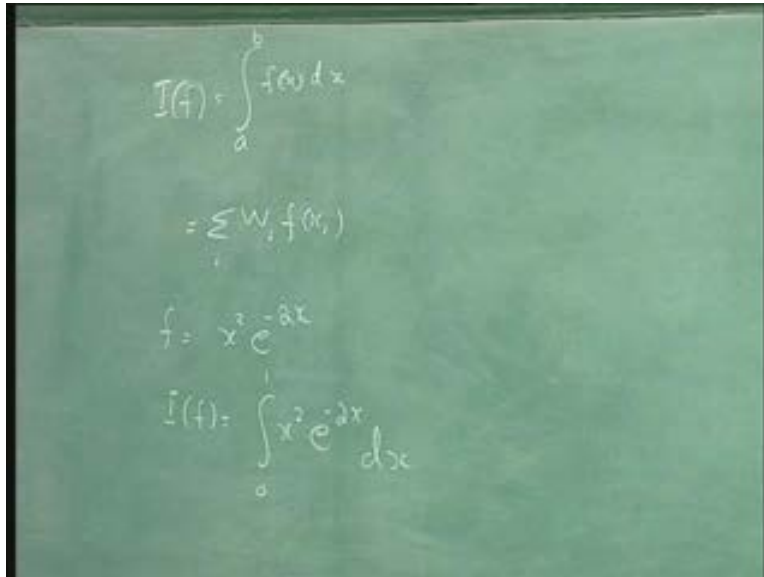
(Refer Slide Time: 3:38)

Simpsons rule

$$I(f) \approx \frac{b-a}{6} \left\{ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right\}$$

So now we look at some actual a program which actually implements of this rule various different rules rectangle rule, midpoint rule trapezoidal and Simpson's rule and compare get an idea about the accuracy of this methods. So we look at the function of the form f equal to x squared e to the power of minus 2x, some function of this form and then we will evaluate the integral the I of f will be interested in is integral limits 0 to 1, 0 to 1, x squared e to the power of minus 2 x dx.

(Refer Slide Time: 5:47)

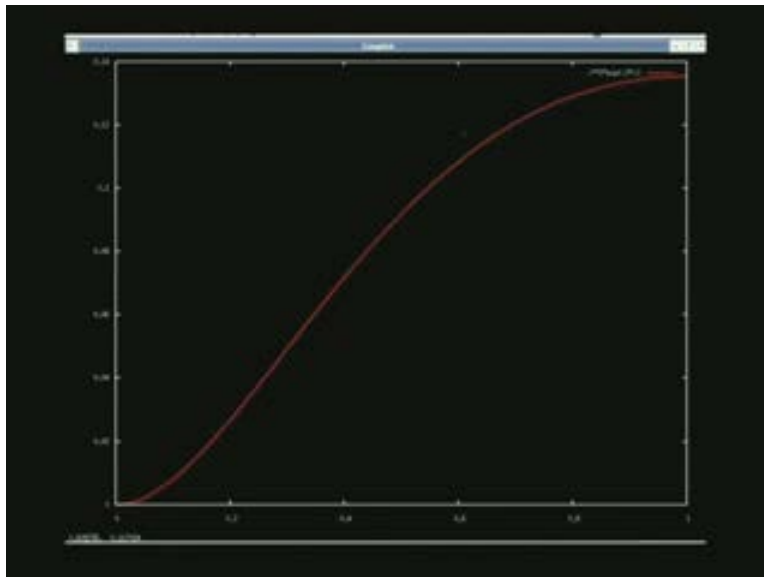


The image shows a chalkboard with the following handwritten equations:

$$I(f) = \int_a^b f(x) dx$$
$$= \sum_i w_i f(x_i)$$
$$f = x^2 e^{-2x}$$
$$I(f) = \int_0^1 x^2 e^{-2x} dx$$

Okay that is what we our interest would be to calculate this. So let us first plot this function and then look at, so here is I am just trying to plot this function between the interval 0 to 1, x squared e to the exponential of minus 2x. Okay between the interval 0 to 1 okay that is what I am going to plot. So that is the plot the function is like this, so that is my functional form so you have some form like this I will draw that here again.

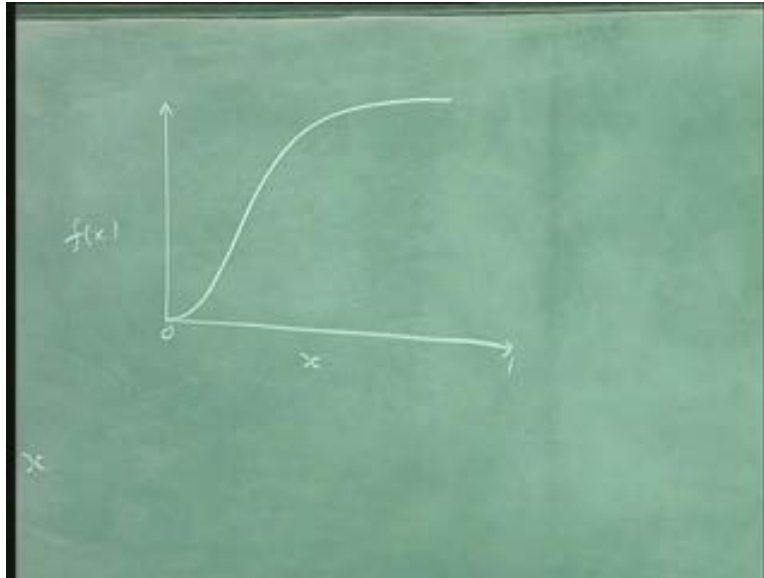
(Refer Slide Time: 5:50)



So you have x, this is f of x and we have from 0 to 1 this integral and it goes like something like this. Okay that is what it is that is the form of that that function

approximately it is this way. So now we want to integrate the function from 0 to 1 and using one of the methods which we have just seen.

(Refer Slide Time: 6:43)



So we will go back to that, so we are going to use one of these methods. So let us first start with the trapezoidal rule so that is saying $\int_a^b f(x) dx$ is approximately $\frac{b-a}{2} (f(a) + f(b))$. So now as I said earlier that it is this if you this is actually approximating the area under this curve by just a rectangle right as we can see evaluate $f(a)$ and $f(b)$.

So $f(a)$ is 0 here. So if you evaluate just only at this point and then take this integral you will get obviously you completely a wrong answer that this integral will be 0 right because, the function is this so it is evaluated at $a=0$ because our interval is now $a=0$ and $b=1$. So if I evaluate this function at $a=0$ then at $x=0$ then you are going to get 0 then we are getting obviously a wrong answer, okay and we know the area under the curve is not 0.

So and so what we going to do is to break the limit, the trivial limit that is where we take the whole interval 0 to 1 as just 1 we just mark this as one the whole interval 0 to 1 as just one quantity and then and then evaluate this integral gives us a wrong answer. So then we shall look we can split this into 2 for 2 equal intervals, let us say we have split this into $\frac{b-a}{2}$ and then do this separately.

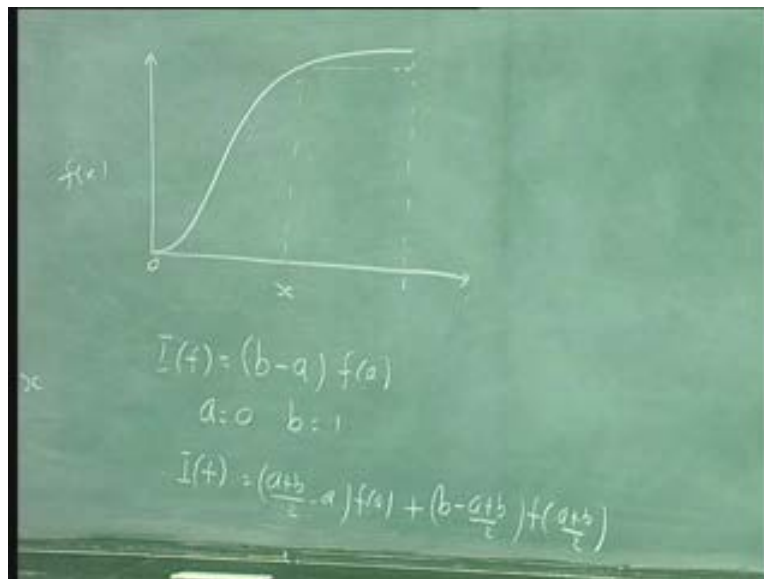
So that we can do right, so that is we split this interval a to b as 2 intervals and then apply the rectangular rule for each of this interval that is $\int_a^b f(x) dx$ now, okay $\int_a^b f(x) dx$ I will now write as one goes from $a + \frac{b-a}{2}$ to f . So I will have this going from $a + \frac{b-a}{2}$ to a so that is $a + \frac{b-a}{2} - a$ that is the first interval and the function

evaluated at a and then I have $b - a$ plus b by 2 and the function evaluated at a plus b by 2.

So that this trapezoidal, this rectangular rule is splitted into 2, okay one so 2 intervals and then I will get now I can see that again this part will be 0 because f of a is 0 but this part would be this part would be non zero which would be the area of this it will be area of this rectangle.

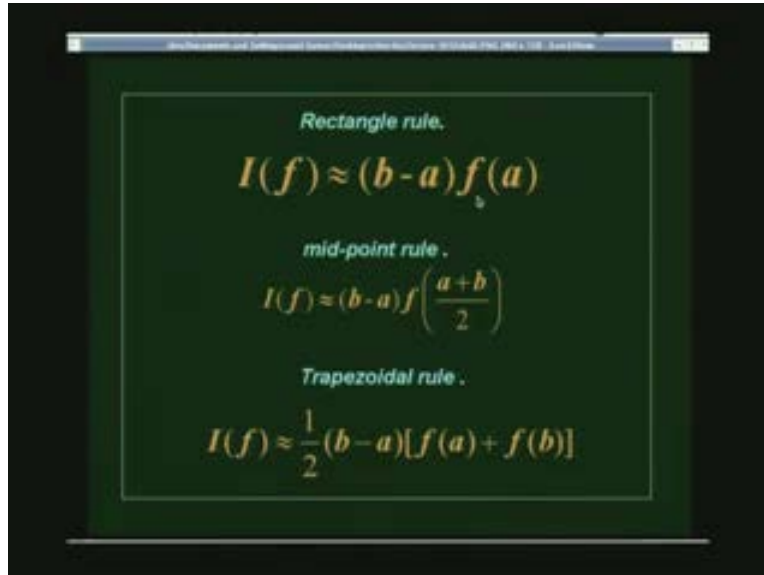
So again there is error because obviously this whole area is left out so you have this error but then we can improve the accuracy by splitting of this into more and more intervals that is what we are going to look at. So we look at how the limit as in what limit or how many intervals we need to get a realistic answer to this integral that is $\int_0^1 x^2 e^{-x} dx$. So the question to ask is that if I use simple rectangular rule okay and evaluate this integral how many intervals I need means how many breakups I should do for this whole interval 0 to 1, how many breakups I should do to get a realistic answer, the realistic answer turns “.80831”. So we know the actual answer is “.80831” to this. So we will see how we would how many points which we have to use to get that answer.

(Refer Slide Time: 10:28)



Okay that is something at which we would be looking at. So look at the program which implements that rectangular rule, okay here is a is a small program which, so this program has this program actually has for it has some it tabulates the points at x is our number of points which we I am which we are going to evaluate the function that means its equal to that the dimension of that should be equal to the interval a to b divided by the number of points which number of divisions which we want. Okay that many points we are going to evaluate this function.

(Refer Slide Time: 10:29)



(Refer Slide Time: 10:48)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
    FILE *fp1;
    int i,j,Npoints,n;
    float *xp,upper_lmt=1.0,lower_lmt=0.0
    ,S;
    char fname[20];

    printf("\n Here, upper limit=%f; lower l
    imit=%f\n",upper_lmt,lower_lmt);
    printf("\n Enter no of pts you want:");
    8,9-16      Top
```

So we are going to take the whole interval a to b and we are going to split up that into some number of divisions and then use the rectangular rule for each of this divisions. So that is the method we should be looking at, okay so this program to be given in an upper limit to the integral and a lower limit to the integral we have given that as 0 and one already. So here in the declaration of the variables I have already fixed this values, these values are already been given. Okay they are floating points and but also given the values. So upper limit is 1 and lower limit is 0 and then x is in a pointer as here and it is going to be an our array. So to before I put in the variables into that I need to locate some memory to that, so I use this malloc function to allocate memory to that that is here. So x

is been allocated some memory using this malloc function. So let us going to a detail of that here okay.

(Refer Slide Time: 11:34)



So I will have I am just splitting up the upper limit and lower limit here. So ignore this particular statement we just printing out the upper limit and lower limit okay and then here it ask this program as for the number of points you want to use, that is the number of divisions you want to use, so number of points at which you evaluate the function, so you enter that here okay.

(Refer Slide Time: 11:37)

```
FILE *fp1;
int i,j,Npoints,n;
float *x,xp,upper_lmt=1.0,lower_lmt=0.0
,S;
char fname[20];

printf("\n Here, upper limit=%f; lower l
imit=%f\n",upper_lmt,lower_lmt);

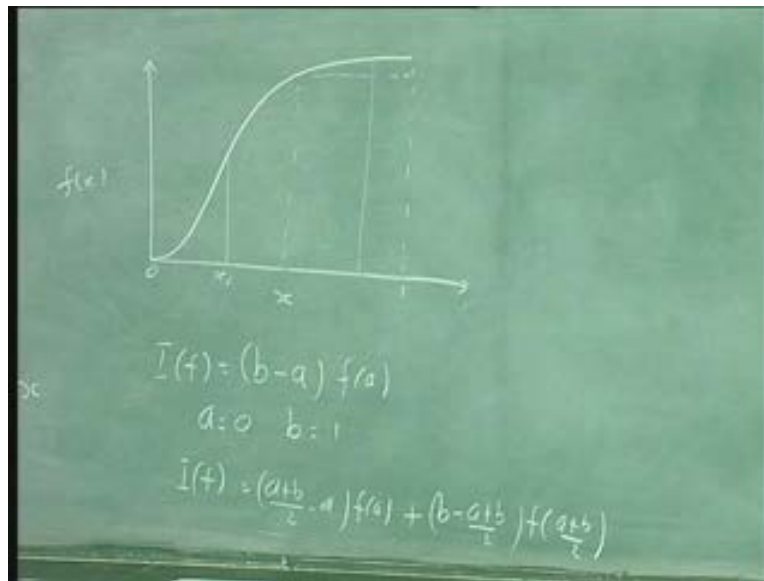
printf("\n Enter no of pts you want:");
scanf("%d",&Npoints);
x=(float *)malloc((Npoints+2)*sizeof(fl
oat));

-- INSERT --                14,21                29%
```

So that minus 1 will be the number of divisions. So the number of points you want to evaluate the function at and that will read off from the screen as the number of end points, end points is the number of points at which we want to evaluate the function. Okay that is in this interval we would be just splitting into many intervals.

Let us say right each of this interval, so now I will label this as x_i 's okay so x_0, x_1, x_2, x_3 etcetera. Okay that is the number of points which I want to evaluate, so in this particular graph now I have drawn that 1, 2, 3, 4, 5 points at which I am evaluating this.

(Refer Slide Time: 13:28)



So that is the number of points and once you given the number of points that many points I have to store in the x variable right, the x is my array which has this x, x_0, x_1, x_2 etcetera but x is declared as a pointer. So I need to now allocate memory to that before I put in some values into it, so that is here. Okay so number of points plus 2 into size of float is the memory I had to because x is the floating point, so I need to allocate memory to that. We have seen this malloc function here is the use of that malloc function to allocate memory to this x , okay so then now the x_p is my interval. So I have dividing this into equal intervals okay this inter this whole this whole distance 0 to 1, I am going to divide them into equal intervals.

Okay so I am just going to say that it is the interval distance is the upper limit minus the lower limit divided by the number of points that is the interval then we have x_i values, x_i values are then lower limit plus i , i goes from 0 to n points i lower limit x of i plus x of i is equal to lower limit plus i times the interval right, when i is equal to 0 x of i is 0 x of i starts from 0 and when i is n points that is i is the number of points and then this equal to the upper limit right, because x_p is upper limit minus lower limit by n points.

So if I substitute that here you see that at n points it is upper limit and i equal to n points it is the upper limit and i equal to 0 it is the lower limit. So x of i was from lower limit to

the upper limit in n point in x_p interval using n points. Okay so that is what it is and now up to this it should this part of the program this whole code up to this it should be common for all methods we are going to mention here that is the rectangular rule, the midpoint rule, the trapezoidal and the Simpson's rule, all of these we will use the same part same program for all this except the last part where we actually compute the integral, that is this part when we actually compute this part we would change the part. Okay now here let us say we are putting n points equal to 1, okay that is only one whole the whole thing is just evaluated at only one point okay then this will be basically 0 right because x of i is 0 and we know that it is this integral will be 0 because f of a will be 0.

So we will see what the accuracy is we will run this program. So you can see that here what I am doing is I am just writing x_p that is the interval okay now if my n points is one the x_p is just whole of my b minus a , otherwise its b minus a by n points. Okay each interval this this this is the length of each interval multiplied by the function, the function is x square this power by using the power function which is built in the program built in the function in the math library that is it calculates the x to the power 2 and exponential another built in function that is minus 2 into x_i . Okay x squared exponential minus $2x_i$ that is what I am computing here okay and I just summed up, sum up all these quantities okay.

(Refer Slide Time: 13:30)

```

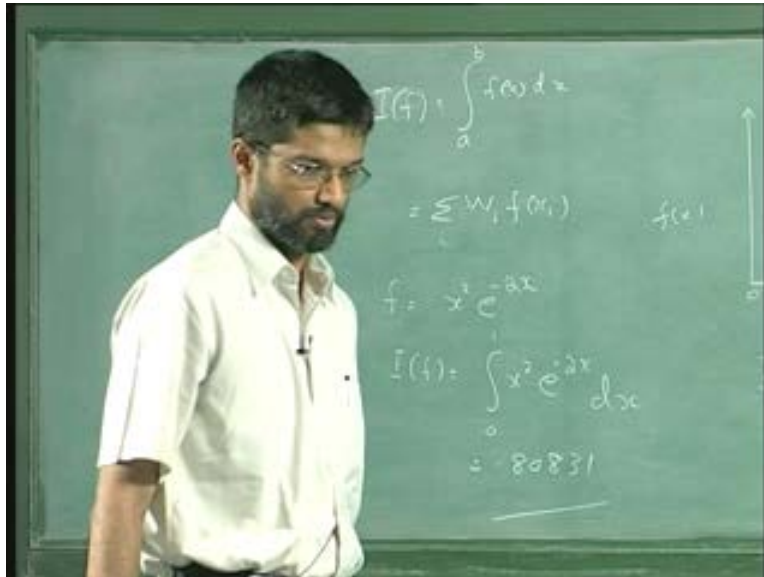
n=Npoints;
xp=(upper_lmt-lower_lmt)/Npoints;
for(i=0;i<=Npoints;i++)
{
x[i]=lower_lmt+i*xp;
}
S=0.0;
for(i=0;i<=Npoints-1;i++)
{ S=S+xp*pow(x[i],2)*exp(-2.0*x[i]);}

printf("%d %f\n",Npoints,S);
}
-- INSERT --                23,8-15                Bot

```

So I basically reduce that into this sum, so b minus a is now split into b minus a by n points and f of a is evaluated at 8 of that starting points then it become a sum okay so it becomes $w_i f$ of x_i kind of sum. Okay so it will become a sum like this that is what I have done where w_i being just b minus a is a simple weight, the weight is simply b minus a divided by the number of points, okay that is the weight w_i in this case okay that is what we see here and then I just print out the answer at the end. So this this the row which computes that sum is over here and then I just print out the answer here, okay that is the program is okay.

(Refer Slide Time: 17:16)



So let us look at this program and just run this and then this is rectangular rule and minus lm because I am using all those math library functions like exponential and power. So I need that, so I run let us compile and run the program so now upper limit is 1 printing out that here the upper limit is 1 and the lower limit is 0. Okay that is what it is printing out and its asking for the number of points you want to evaluate number of intervals basically number of points you want that is the splitting you want, let us put it as, let us say 5 okay.

(Refer Slide Time: 17:18)

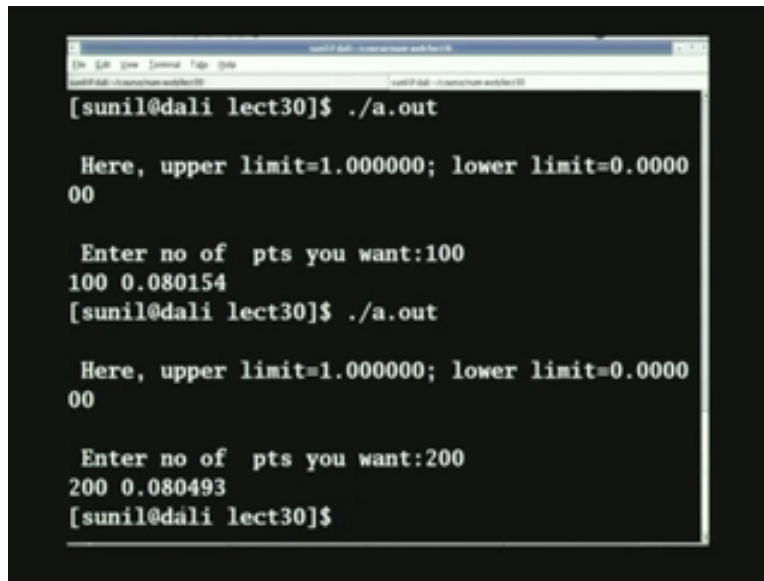
```
[sunil@dali lect30]$  
[sunil@dali lect30]$  
[sunil@dali lect30]$  
[sunil@dali lect30]$  
[sunil@dali lect30]$ gcc rectangular.c -lm  
[sunil@dali lect30]$ ./a.out  
  
Here, upper limit=1.000000; lower limit=0.000000  
00  
  
Enter no of pts you want:5  
5 0.067270  
[sunil@dali lect30]$
```

So we get the answer to be “.067270” so 5 values which we have evaluated the function and we are getting it as “.067270” while we know the correct answer is “.80831” okay

you can do this analytically you can see that this “.80831” is the correct answer to this equation to this integral.

Okay so we have to obviously have to increase the number of points okay we just increase it to be 10, okay so we go from “.067” to “.74” and we go into 20. So we get “.77” we go onto 100 points and then get “.801” so we are slowly approaching the correct answer but we already split this whole interval into 100 sub intervals to get the, to reach that answer. We have still an error in the second third decimal places so we need to get “.808” we are still at “.801” at 100 sub intervals so we go something like 200 we still have an error in the 3rd decimal place in this here okay this “.804” “.0804” 4 decimal place “.0804” well the answer is “.80 the answer is “.08” that is what the answer which we should get.

(Refer Slide Time: 18:44)



```
[sunil@dali lect30]$ ./a.out

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:100
100 0.080154
[sunil@dali lect30]$ ./a.out

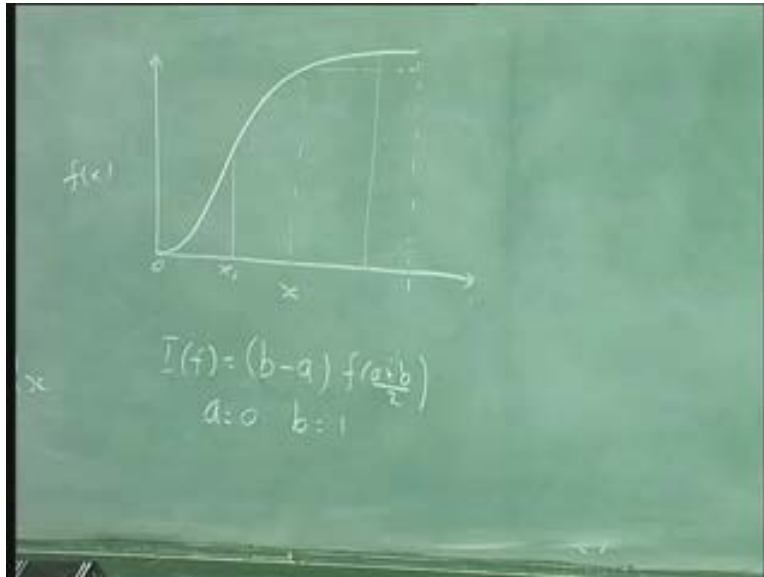
Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:200
200 0.080493
[sunil@dali lect30]$
```

So we saw that if we the simple rectangular, rectangle rule and split this interval even if even into 100 sub intervals, we still are very far from the actual answer to this interval which integral, which should be “.080831” so it will take it takes us lot of function evaluations and the sum to get this integral correctly and if we use a simple rectangle rule. Okay now let us do the same thing okay using a midpoint rule okay now we will use $b - a$ into f plus f into $a + b$ by 2.

So again I will use the next one that is will use function value evaluated at f plus a, b by 2. Okay that is what we are going to use in the in the midpoint rule. So that means now we going to evaluate for each sub interval the function evaluated is middle of that interval okay and then we will exactly do the same calculation then will see what happens.

(Refer Slide Time: 20:59)



Okay so now here is the program which we would use midpoint rule, so it is the same program now not a different the top part is exactly the same okay and only now the as I said where we exactly actually calculate the sum that is where the change is s equal to 0 before the loop and inside the loop you will take the number of points then evaluate the sum of all the function products. So that is x_p multiplied by the function values now the function value, now there is a change here the function value is not evaluated as x of i now.

(Refer Slide Time: 21:00)

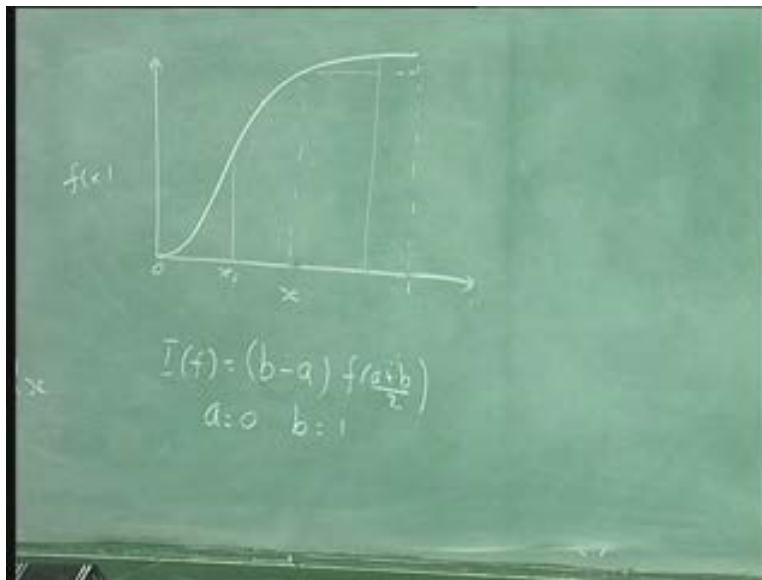
```

n=Npoints;
xp=(upper_lmt-lower_lmt)/Npoints;
for(i=0;i<=Npoints;i++)
{
x[i]=lower_lmt+i*xp;
}
S=0.0;
for(i=0;i<=Npoints-1;i++)
{ S=S+xp*pow((x[i]+x[i+1])/2.0,2)*exp(
-2.0*(x[i]+x[i+1])/2.0);}

printf("%d %f\n",Npoints,S);
}
24,23-30 Bot
```

Okay now I am taking x of i as x of i plus 1 by 2, so I want to keep this thing as constant that is what we give is the number of points at which the function is evaluated okay. So when I say n points that is the number of points at which the function is evaluated that is what we are going to give. Okay and then we are going to use this rule, okay so now when I so again I evaluated lets say at 5 points, okay now I am going to take this as an interval and that as another interval because I need to evaluate this f plus f of a plus b by 2. So I am going to do this as an interval and that as another interval, so actually when I again I need, I calculate the function at 0 $x_1, x_0, x_1, x_2, x_3, x_4$ okay and then I will evaluate the function at very alternate points.

(Refer Slide Time: 22:25)



Okay that is what I am going to here. Okay so I have evaluated x of i plus x of i plus 1 by 2 okay so in the middle of every interval, okay so I take this interval and I evaluate the function at this in the middle of that interval okay. So when I say here the number of intervals is 10 is actually we are evaluating the function only at 5 different points.

Okay so now let us look at this this points, so that is f of a so that is interval b minus a that is for each interval that is x_p now that is b minus a divided by n points and the function evaluated here at the midpoint that is x of i plus x of i plus 1 by 2 power 2 right that is x squared exponential minus $2x$ of i plus x of i plus 1 by 2. Okay that is I have written it that way, okay that is exactly this if I write this in this form I have writing this function now as this way. So the function is x_i plus x_i plus 1 by 2 it is see it. So it is x_i plus x_i plus 1 by 2 the whole squared right that is x squared and e to the power of minus 2, x_i plus x_i plus 1 by 2 that is x of i .

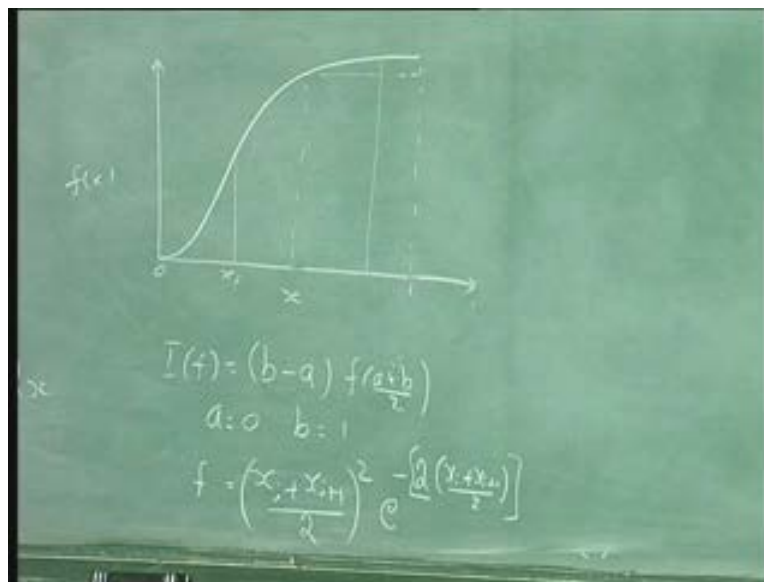
(Refer Slide Time: 22:26)

```
n=Npoints;
xp=(upper_lmt-lower_lmt)/Npoints;
for(i=0;i<=Npoints;i++)
{
x[i]=lower_lmt+i*xp;
}
S=0.0;
for(i=0;i<=Npoints-1;i++)
{ S=S+xp*pow((x[i]+x[i+1])/2.0,2)*exp(
-2.0*(x[i]+x[i+1])/2.0);}

printf("%d %f\n",Npoints,S);
}
```

24, 23-30 Bot

(Refer Slide Time: 23:57)

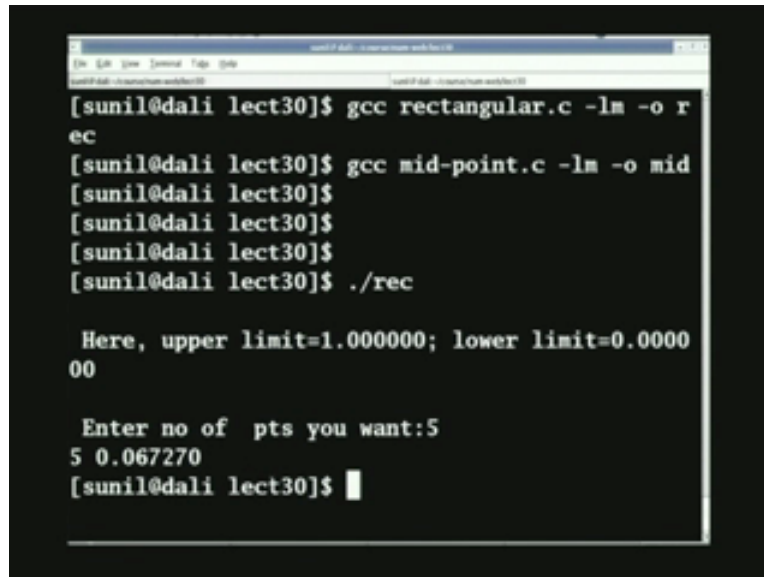


So that is, so that is what I am evaluating here, I have written like that okay. You could cancel this two and two but just for clarity just we show you that this actually evaluated at the midpoint between this intervals i and i plus 1 that is where I am evaluating the function. So we will run this program, so before we do that we will make two different output files as we compile the a rectangular method, the method we were using rectangular method. So that program we will compile as and store it as RAC.

So that **so** when you run RAC then I am running a rectangular method. Okay now similarly, I will do the midpoint method and then when you compile that and keep as

MID. Okay so if I run dot rec then that means I am running the rectangular method and so if I run dot mid then I am running the midpoint method. So if run dot RAC I run it with five points I get the answer as “.067270” where I should be getting as we have seen that “.8, .08, .3 08031831” that is what the answer should be.

(Refer Slide Time: 24:00)



```
[sunil@dali lect30]$ gcc rectangular.c -lm -o rec
[sunil@dali lect30]$ gcc mid-point.c -lm -o mid
[sunil@dali lect30]$
[sunil@dali lect30]$
[sunil@dali lect30]$ ./rec

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:5
5 0.067270
[sunil@dali lect30]$
```

So and then we will now run the midpoint program which is again we will use the 5 points actually when you are saying 5 points here. We are evaluating the function actually at less than 5 points here but we use the same thing here 5 points and then we get an accuracy much better than the rectangular rule. So you can see that simply by going to midpoint rule, we have a much higher accuracy and we have seen the reason for this this is actually of order, the second derivative of the function this accurate error in the midpoint rule okay.

So if you run RAC up to 100 in the 100 points then we get “.0801” we should get “.0808” “.0808” is the answer we are getting “.0801” while just running 5 points using the midpoint we are already at “.0808” so there is tremendous improvement in the, in this accuracy by just going to a midpoint method and now for this particular function, and now we look at the next method that is the trapezoidal rule in which we now use 2 points at which the function is evaluated. So we now use $b - a$ by 2 into f of a plus f of b by 2 f of a plus f of b . So now we are going to evaluate the function at 2 points again 2 points in each interval, so again I will split this function into I will split this function into many intervals and at each interval I will evaluate it at two points this thing that is what it is going to be the trapezoidal rule implementation. So let us look at that again.

(Refer Slide Time: 25:14)

```
[sunil@dali lect30]$ ./mid

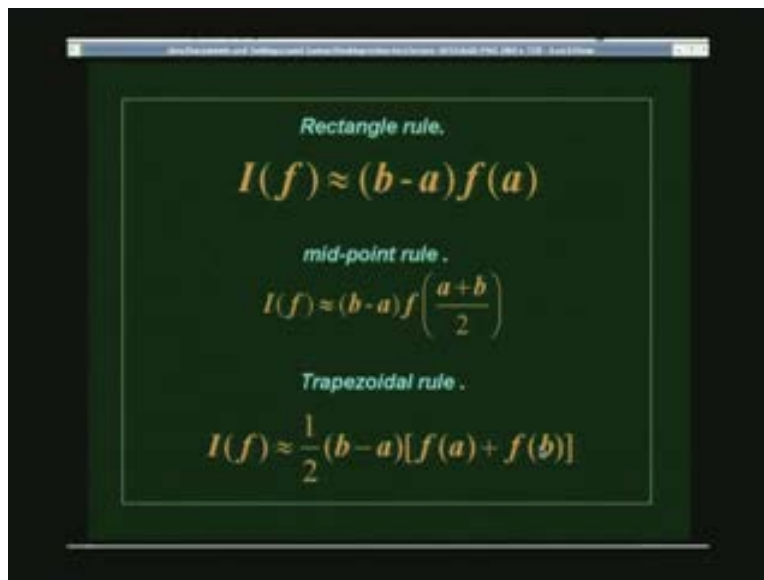
Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:5
5 0.080855
[sunil@dali lect30]$ ./rec

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:100
100 0.080154
[sunil@dali lect30]$
```

(Refer Slide Time: 26:25)



So here is the program which does that a trapezoidal, so that is so this program again the initial part have the same here n points at which it has to be evaluated the number of intervals etcetera so now here I change this now I need to evaluate for each interval x_p right and x_{p+1} by 2. So the end of this setting I have it by 2 here setting so you can see that here I divide that by two the function. So I have now x_p multiplied by the function value at x_i and then plus the function value at x_{i+1} so the 2 into 2 function values. So 1 is at x_i and the other is at x_{i+1} plus 1.

(Refer Slide Time: 27:01)

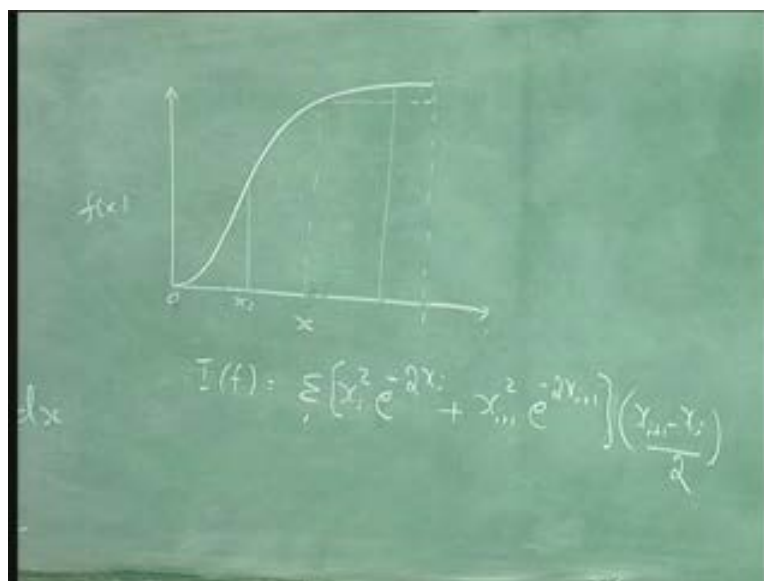
```
n=Npoints;
xp=(upper_lmt-lower_lmt)/Npoints;
for(i=0;i<=Npoints;i++)
{
x[i]=lower_lmt+i*xp;
}
S=0.0;
for(i=0;i<=Npoints-1;i++)
{ S=S+xp*(pow(x[i],2)*exp(-2.0*x[i])+
pow(x[i+1],2)*exp(-2.0*x[i+1]))
)/2.0;}

printf("%d %f\n",Npoints,S);
}
```

26,1-8 Bot

So you change this thing a little bit, so I would now write this in the case of trapezoidal rule now I am going to write this the program what it does is to compute the f the I as i of f is now computed as sigma I, x_i squared f exponential of minus 2 x_i plus x_i plus 1 squared exponential minus 2 x_i plus 1 inter interval x_i plus 1 minus x_i by 2 that is our trapezoidal rule. Okay that is going to be our trapezoidal rule that is that we are going to use and each of this for many intervals x_i . Okay so that is what is said here.

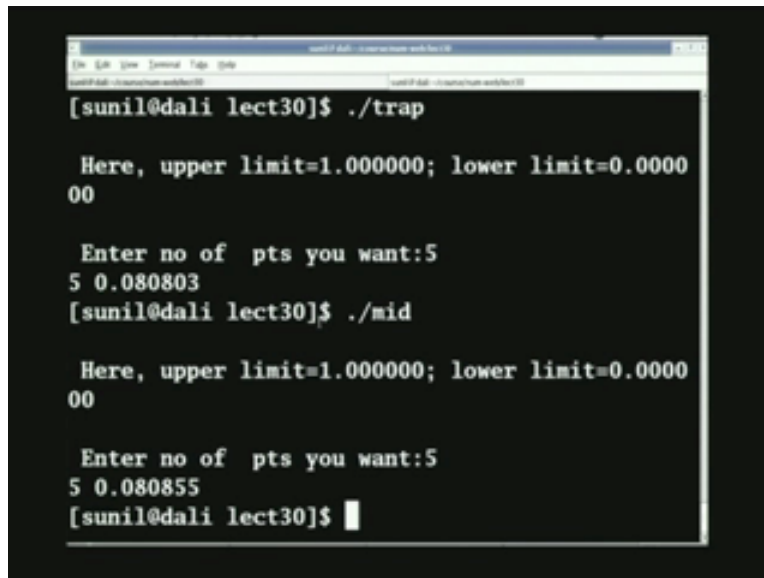
(Refer Slide Time: 28:42)



So you have this function evaluated at x_i and x_i plus 1 and multiplied by the $d x_p$ by 2 x_i plus 1 minus x_i and divided by 2 is here. Okay that is what we are going to run let

us run this, okay let us call that when we, when you compile that now. So let us call that as trap okay so that will be we run trap that is we are running trapezoidal okay let us run that and we get we will use again 5 points so we get “.80803” so compare that with midpoint for 5 points and we get “.80855”. So this is comparable okay so we see that we are not getting much higher accuracy by going to trapezoidal rule and using the same 5 points integration.

(Refer Slide Time: 28:44)



```
[sunil@dali lect30]$ ./trap

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:5
5 0.80803
[sunil@dali lect30]$ ./mid

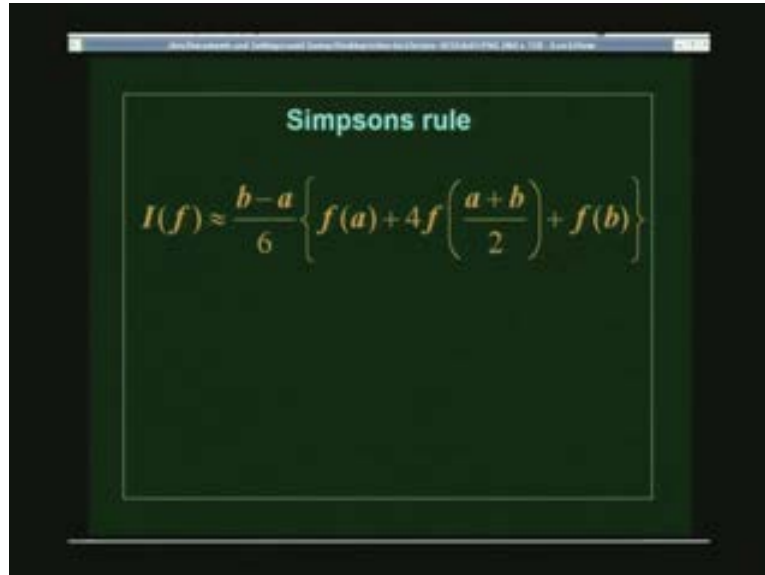
Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:5
5 0.80855
[sunil@dali lect30]$
```

So far we looked at 3 different methods that is rectangular rule, midpoint rule and trapezoidal rule. So now we look at the Simpson's rule so that is that way we have 3 function values. So we have $f(x_i)$ and $f(x_i + \frac{1}{2})$ and $f(x_i + 1)$. So now we will use 3 function values, okay and they have different weights now so now this is it is the first time we are actually using function with different weights. So here again we use function values at 2 points but both of them have the same weight but now here, we have this function this function value this point has a different weight this is 4 times $b - a$ while this is $b - a$ while this weight is $b - a$ by 6.

Okay now that is the program which does the Simpson's rule is here again similar to this one okay, except for the last part. Okay here is the part now we have to evaluate the function at x_i that is the first part here. Okay so x_p by 6 here that is x_p by 3, so that is this is the function evaluated at x_i and then the function evaluated at $x_i + 1$ that is, and then you have a function evaluated at $x_i + 2$ that is what I am using so x_i .

(Refer Slide Time: 29:44)

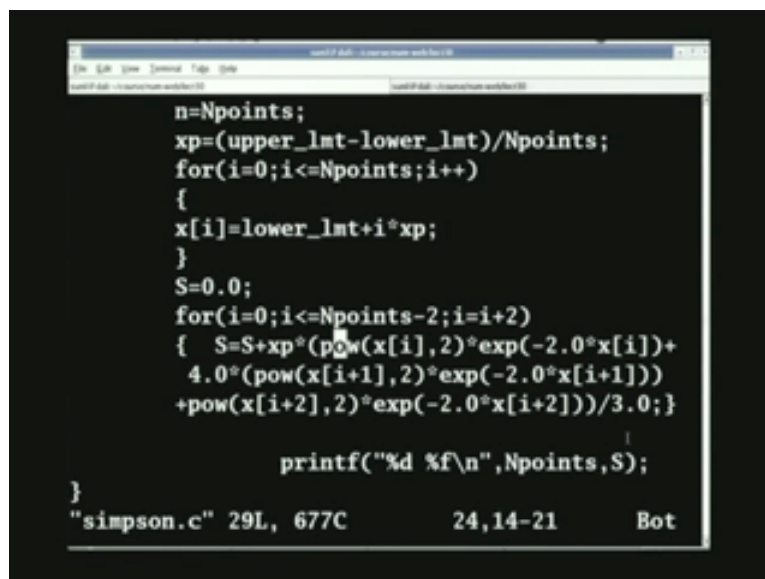


Simpsons rule

$$I(f) \approx \frac{b-a}{6} \left\{ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right\}$$

So now what I am going to do is I am going to split this interval like this okay and I am say that this is one interval and I evaluate the function at 0, x_2 and x_1 , x_0 , x_1 and x_2 like that instead of using the x_i plus 1, x_i minus 2, I am just using 0 x_1 and x_2 kind of 3 points in need to evaluate the function at 3 points I do that as x_i plus 1 and x_i plus 2 okay that is splitting this into 3 different in 3 different points at which we have to evaluate the function. So remember the quantity the point is this is f of a plus 4 f of a plus b by 2 and f of b that is b minus a by 6. Okay, so now I my interval has become 2 now here that is why I have here x of i , I put in here divide by 3 because my actual interval divide this is actually x of i m_i x of i plus 2 minus x of i .

(Refer Slide Time: 30:22)

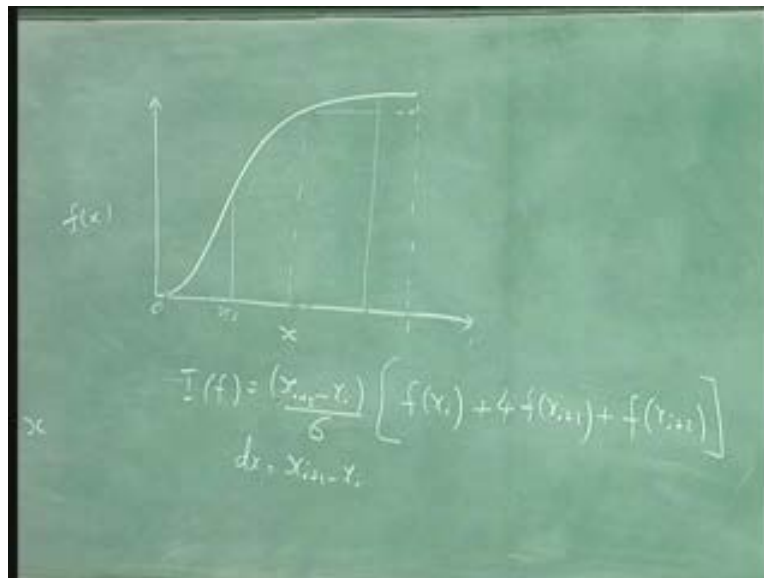


```
n=Npoints;
xp=(upper_lmt-lower_lmt)/Npoints;
for(i=0;i<=Npoints;i++)
{
x[i]=lower_lmt+i*xp;
}
S=0.0;
for(i=0;i<=Npoints-2;i=i+2)
{ S=S+xp*(pow(x[i],2)*exp(-2.0*x[i])+
4.0*(pow(x[i+1],2)*exp(-2.0*x[i+1]))
+pow(x[i+2],2)*exp(-2.0*x[i+2]))/3.0;}
printf("%d %f\n",Npoints,S);
}
"simpson.c" 29L, 677C          24,14-21      Bot
```

Okay so that is while x_p is x of i plus 1 minus x of i so I have to put divide multiply this divide that, by multiply that by 2 that is what it is become 3. Okay so let me write that here this what I am going to do here i of f is now is evaluated in the following form.

I am writing it as x of I , so I need an interval, so it is x of i plus 2 minus x of i that is my interval I am using and then I am evaluating the function at f of x_i and then I have here 4 times f of x_i plus 1 and then I have evaluating it at x_i plus 2. Okay that is the Simpson's rule that I am using here okay, where f where f is given by each point given by this okay this x so while my d my the dx which I was using was x_i plus 1 minus x_i okay this is actually 2 times dx , so actually this is divided by 6, so this is 2 times dx by 6 that is dx by 3, okay that is what I put it in here.

(Refer Slide Time: 33:18)

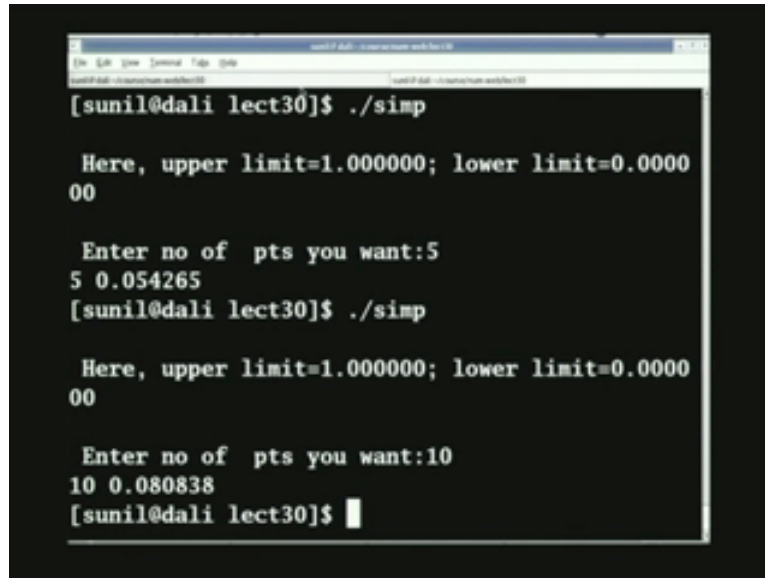


This program this x_p here I divide it by 3, okay so now let us run this. Okay so when I say now here again the number of points at which the function is evaluated is what we are going to give in the program. Okay so and then we are going to run this here will now compile this program we call it simp, we will run simp, we will run with 5 points okay and you have an accuracy of “.0454” okay now Simpson's rule does not seem to gives us very high accuracy but the number of interval is of course small. So let us run it with ten points and we get 10 points reasonable accuracy, so now this again a 5 intervals okay.

So when I say 10 points it is actually 5 intervals because this is one interval for now but the number of functions, number of points at which the function is evaluated is 10. Okay but the intervals is 5, so you see that in terms of the number of points at which the function is evaluated this, for this particular function this does not seem to give us much more accuracy than the trapezoidal rule but if you together interval then it is as good as the trapezoidal rule which we get for 10 points and we get “.80” “.0808”. Okay that is the simpler methods where we actually split the function into the various interval and then

evaluated this function value and multiplied them by the distance between those intervals that is what all these methods rely on okay.

(Refer Slide Time: 33:19)



```
[sunil@dali lect30]$ ./simp

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:5
5 0.054265
[sunil@dali lect30]$ ./simp

Here, upper limit=1.000000; lower limit=0.000000

Enter no of pts you want:10
10 0.080838
[sunil@dali lect30]$
```

So now let us go into the next set of rules which are the Gaussian rules right. So now this something which we discussed detail in the last class that is we now evaluate the function i of f , now this is slightly different from the earlier methods because we now evaluate the function i of f by writing i , i of f as this again we use the similar, we use the similar finally the interval, integral reduce into this but the technique here is writing i of f as now instead of f of x dx we are going to write it as a to b g of x , w of x dx right. That is what is so weight function here w of x this and then to distinguish this from this we call it a_i here a of x_i okay.

So now this is some weight function now depending upon this weight function and this limits we had different orthogonality properties and so of the polynomial and so we said that we discussed that in the last class and we said, we can have Legendary polynomial if it is minus 1 to plus 1 chebyshev if it is minus 1 to plus but we have a weight function one minus x squared here the weight function is 1 and Laguarre if it is 0 to infinity and Hermite minus infinity to plus infinity etcetera.

All these methods the idea is to approximate the function by a polynomial okay and then do the integral and finally the integral will then reduce to this form okay that is the g function evaluated at x_i multiplied by this particular polynomial and this is transform for the polynomial approximation. So we write that in the Lagrange form and then multiplied by g of x_i to if I said what we are going to do this we are going to approximate g by a polynomial. So in this method g is approximated by p_k of x okay and which is given by a sum over i going from 0 to k sum product j going from 0 to k j not equal to i right x minus x_j divided by x_i minus x_j into g of x_i okay that is what the approximation.

(Refer Slide Time: 35:40)

Gaussian Rules

$$I(p_k) = \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^k \frac{x-x_j}{x_i-x_j} g(x) W dx$$

$$= \sum_{i=0}^k A_i g(x_i)$$

Thus the integral is a weighted sum of the function values at a set of points with the weights given by

$$A_i = \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^k \frac{x-x_j}{x_i-x_j} W dx$$

So g here we write the integral in this form and then choose the appropriate w for limiting upon the limits. Okay and we write the polynomial this is approximated by a polynomial p okay which is given buy this quantity and that was the basic idea of this integral and then the weights and then we can write that as a_i g of x_i and the weights given by the integral of this part right now the g of x_i comes out of this integral because integral over dx. Okay it is only the function of x integral over x the function this is the only part which is the function of x. So we can take that out so then we can write this in this form with a of i given by this quantity.

(Refer Slide Time: 37:59)

The chalkboard shows the following derivations:

$$I(f) = \int_a^b f(x) dx$$

$$= \sum A_i f(x_i)$$

$$I(f) = \int_a^b g(x) W dx$$

$$g(x) = P_k(x) = \sum_{i=0}^k \prod_{\substack{j=0 \\ j \neq i}}^k \frac{x-x_j}{x_i-x_j} g(x_i)$$

On the right side of the board, there is a small graph of a curve on a coordinate system, with a point marked on the x-axis and a vertical line extending to the curve.

Okay so now we have seen that the weight do go about this is because of the orthogonality properties of the matrices that if you want to get very high accuracy using this that is if I choose the kth order polynomial I can get accuracy up to 2k plus 1 orders in derivative provided. I choose these points right this point x_j 's at which the function is tabulated okay now the function has to be tabulated at some values and that tabulated points if I choose them as the 0s of a polynomial I depending up on what the limits and the w is okay.

(Refer Slide Time: 38:00)

Gaussian Rules

$$I(p_k) = \int_a^b \sum_{i=0}^k \prod_{\substack{j=0 \\ j \neq i}}^k \frac{x-x_j}{x_i-x_j} g(x) W dx$$

$$= \sum_{i=0}^k A_i g(x_i)$$

Thus the integral is a weighted sum of the function values at a set of points with the weights given by

$$A_i = \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^k \frac{x-x_j}{x_i-x_j} W dx$$

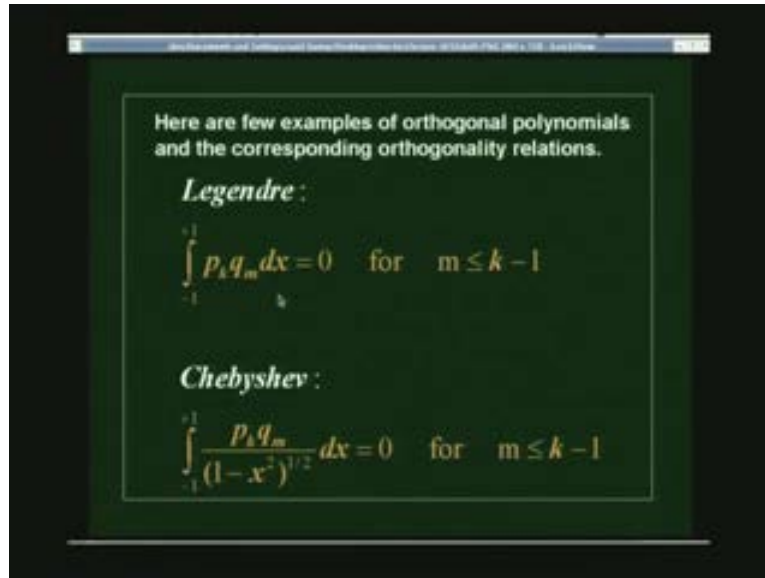
So for example if it is the limits were minus 1 to plus 1 and my weight function was here was 1 then I would use the tabulated points as a 0s of the legendary polynomial, okay and similarly for other cases. Okay so that is the method so first part in this thing would be to write the integral which we want to evaluate into this form into one of this form. Okay we have to change the limits such that the limits are minus 1 to plus 1 and we write that in this form. So that is what we had seen in this particular example.

For example when you want to these are particular function of interest right now, we just want to evaluate this integral that is x squared e to the power of minus 2x. Okay and then I have to change the limits of the integration from 0 to 1 from 0 to 1 to minus 1 to plus 1 and that do by changing the variable from x to t by going from minus 1 to from x to t that is t equal to minus 1 plus 2x and then I can write the integral now i as 1 by 4 integral minus 1 to plus 1 plus t whole squared e to the power of minus 1 plus t dt.

Okay and then evaluate the function and now this function is evaluated at various values of t and those values of t are the 0s of the legendary polynomial because of my limit and my weight is one so that is the idea, so now I hope this is clear. so we want to use the Gauss Legendary method then this function this integral is first converted into minus 1 to plus 1 by change of variable and then this function here 1 plus t squared e to the power of minus 1 plus t is tabulated at different values of at the 0s of the Legendary polynomial,

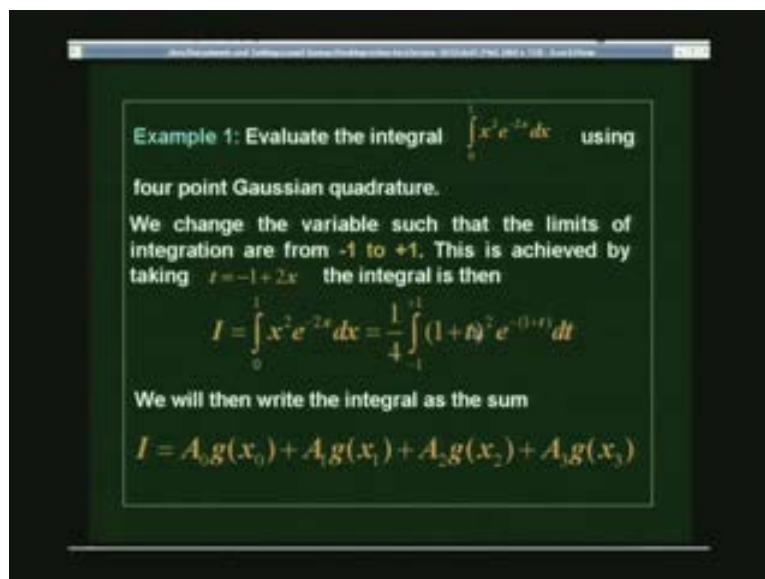
okay and appropriate weight functions are calculated using for example, this value. We do not use this value actually I will come to that later.

(Refer Slide Time: 39:06)



Okay but so that is we appropriate weight functions are calculated and the functions are evaluated at that point and we do a sum of all those functions. Okay now here is for example I summed over 4 points that is x_0, x_1, x_2 and x_4 . So this x_0, x_1, x_2 and x_4 should be the first 4, 0s of the fourth order **Lagrange, a Lagrange polynomial** Legendary polynomial. Okay in the in this particular gauss Legendary integration which is f of x integral minus 1 to plus 1 f of x dx with w equal to 1.

(Refer Slide Time: 39:41)

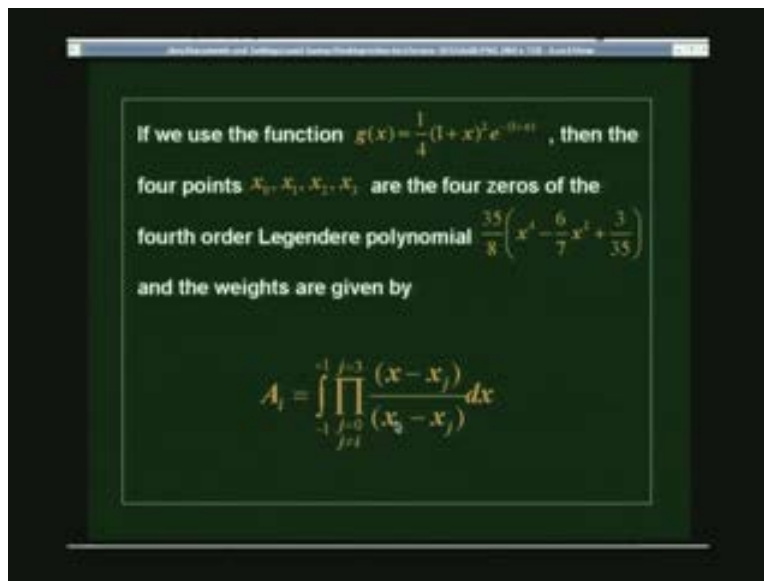


So it is just f of x dx or f of t dt in this particular case because that is what we want to look at, so remember that is we can so you can also incorporate that change of variable into the program itself and now you want to evaluate in the program in terms of x itself then what do you could do is instead of tabulate the t 's as the 0s of the Legendre polynomial and then evaluate this function at 1 plus x_0 by 2 , 1 plus x_1 by 2 , 1 plus x_2 by 2 because the relation between x and t was t equal to 1 minus 1 plus $2x$ or x equal to 1 plus t by 2 is that is also possible.

So that is what we have to do, okay so now we come to the computation of the weight okay we know where to choose the x to be okay so now the functions are evaluated tabulated at the 0s of the Legendre polynomial in this particular case because again I repeat because you are going to use minus 1 to plus 1 interval and the weight is one here of the function.

So now the question is how do I how am I going to compute this this weight because this weight itself involves an integral over minus 1 to plus 1 . Okay so if I use this form this is not very convenient because I can I have to integrate this from minus 1 to plus 1 . So this weight itself involves an integral I will not gain much advantage but it is not so complicated because what we can do is, we can actually write this particular function in terms of the Legendre polynomial itself okay and then we write the w as in this form.

(Refer Slide Time: 41:51)



So what we seen so far is that we have to write in this we can write it in this form and we said a_i is now given by integral a to b right and or in the particular in this case suppose, if it is converted in the limits etcetera then it is minus 1 to plus 1 . Okay so for each of this and ϕ_j going from 0 to k_j not equal to i , x minus x_j divided by x_i minus x_j . Okay so that is actually the weight which we have to compute, okay dx that is what the weight which we have to compute. Okay now this involves integration that is not something which we want to do because our idea is to actually do another integral.

So it does not make sense to compute the weights by doing integration and then doing a sum so we have to change this form. Okay and it turns out that this can be actually written as in terms of the Legendre polynomial whose 0s we have used here to tabulate the function value. Okay or this f of x_i here actually tabulated at the 0s of this polynomial of this Legendre polynomial and in terms of this Legendre polynomial we can write this weight and this that turns out to be 2 divided by 1 minus x_i squared into P_k prime P_k plus 1 prime of x_i P_k plus 1 prime of x_i whole squared okay now that is the derivative of the k plus 1th order Legendre polynomial.

So I should say one more thing, so that actually if we choose this this there order difference between these two because, these are now tabulated values the x_i are the tabulated 0s of the Legendre polynomial. So this goes from 0 to k plus 1 k so that is of order k plus 1 so I am using that polynomial Legendre polynomial here okay.

So I can write this I do not go into the details here of how we go from this to this but you can write the weight as the weight at i a of x_i at as 2 divided by 1 minus x_i squared into the derivative of the k plus 1th order polynomial at x_i evaluated at x_i squared. So now this is this the form which we are going to use for this computation. So once we know the 0s and the corresponding weights then we can simply compute the sum and then we get the integral so that is what something which we would look at okay.

(Refer Slide Time: 46:00)

$$I(f) = \int_a^b f(x) dx$$

$$= \sum A_i f(x_i)$$

$$A_i = \int_{-1}^1 \prod_{j=0, j \neq i}^k \left(\frac{x - x_j}{x_i - x_j} \right) dx$$

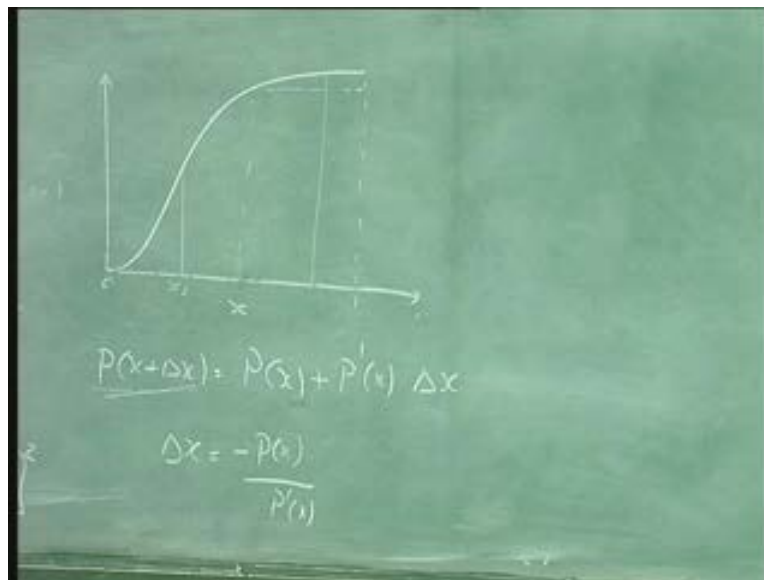
$$= \frac{2}{(1-x_i^2) [P_k'(x_i)]^2}$$

So now the question is how do I compute this derivative and how do I evaluate the function at various points, how do I find the compute the weights is the question and how do I know the 0s of this polynomial. Okay, so I said that I need the 0s of the Legendre polynomial here as the that is where I am going to tab tabulate it and I need the derivative of the Legendre polynomial. Okay, now these two things are required to actually compute this quantity okay but remember, both of those does not depend on the actual

function value itself so once you can actually construct a table of this quantity and keep it and you can use it for different values of the function okay the only thing we have to change rescale the variables at every different functions different limits and that is the only thing which we have to change but these a_i 's and the x_i 's are independent of the function values. So we can actually compute that.

Okay so now the question is how do we evaluate the derivatives and the and the 0s, okay so to evaluate the 0s of course we will use we have seen in the earlier methods we can use something like Newton Raphson, okay to find the 0s of the polynomial. Okay so if I know how to evaluate the derivatives of the polynomial and then I can use Newton Raphson to use the I get the 0s of the polynomial. So I just make a guess value for the 0 and then I write p of x plus Δx as p of x , okay p being the polynomial of some order plus p prime of x right into some Δx and then I say that this should be my 0. Okay so and then I get Δx as minus p of x divided by p prime of x and then I add the next point would be so I would I start with a guess value for x and then I go into the next iteration x plus Δx as my guess value and I repeat this quantity till the p goes to 0, so that is Newton Raphson, simple Newton Raphson scheme.

(Refer Slide Time: 46:02)



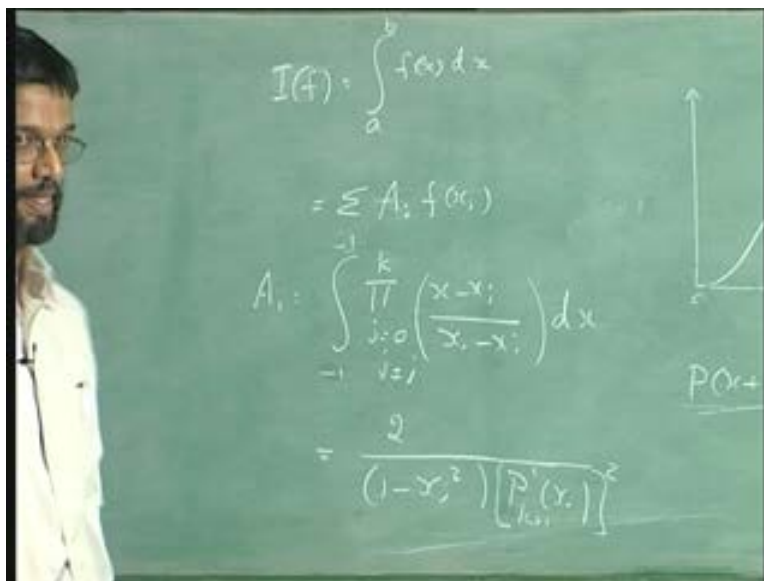
So I have all the tools I know the Newton Raphson to get the 0s of the polynomial and so once when if I know how to compute the derivative of the polynomial and how to evaluate the polynomial at that particular value of x then I can compute this integral here right. Okay because I can find the zeros using Newton Raphson and but only thing which I now need is the evaluation of the function p of x and its derivative and for that we again use some properties of the Legendary polynomial or the recursion relations which tells us that j the relation between the j th order polynomial the j th order polynomial and the j plus 1th order polynomial. Okay the recursion relations connect a polynomials of different order by a simple linear relations that is what we are going to use. So we are going to use the fact that the recursion relation j plus 1 and p_j plus 1 is equal to $2 j$ plus 1 times $x p_j$

minus p_j minus 1 that is what recursion relation is. So it connects the j th order polynomial in j plus 1th order j minus 1th order polynomial okay.

So that is this the relation which we can use and we know that p of 1 is 1 okay, so that is also something which we will use okay and then we start from this and then compute at a any polynomial of any order you want at any value of x . Okay so p of 1 is 1 for all x values so now if I want to evaluate a polynomial p_{10} then I will start from p of 1 and then I will just keep on doing the recursion relation and I obtain the, I use the recursion relation to obtain the next higher order polynomial values. Okay that is what the thing which I can use and then I have this relation the p_j prime is given by j into x p_j minus p_j plus 1 divided by x squared minus 1.

So this is another quantity which I can use okay, so I have basically the recursion relation to compute the p values, okay and another recursion another relation which connects the derivative of the j th order polynomial to the polynomial j th order polynomial and the j plus 1th order polynomial. Okay so I know how and this is connected to this through the recursion relation, so I can actually I can compute all these quantities and then and once I have the p and the p prime I can actually get the Os of the polynomial and once I have the Os of the polynomial and the derivative I can compute the weights and once I have the weights I can compute the integral, okay that is the whole scheme.

(Refer Slide Time: 50:54)

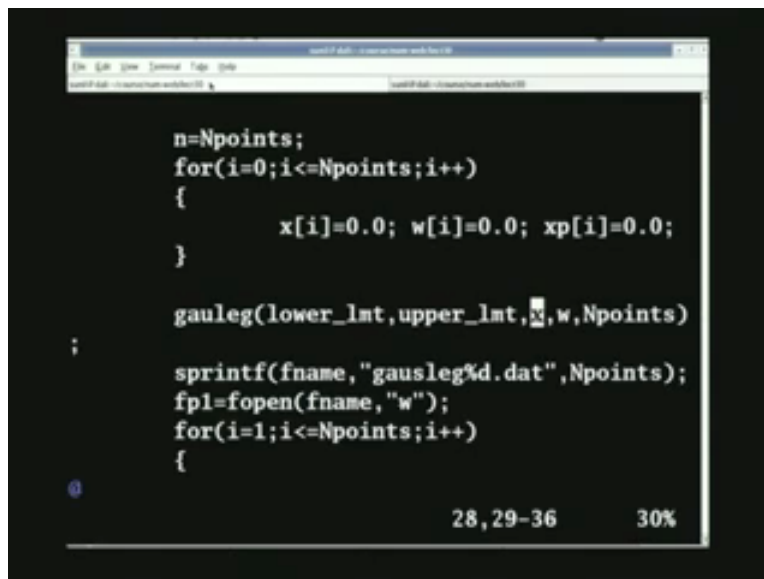


Okay so now we will just see the implementation of this in a in a program. So that is slightly more complicated simple code, okay now this is the program here. Okay so again, so the first part of this is the same that it actually asks you how many number of points you want etcetera and then this function um locates memory for different functions like weights now we need to we need to store the weights and the weights this x is the function values and x_p is again the values at which we have to tabulate the function.

Okay so once we have that we just call a program now you see as I said this independent of the function right. So the weights and the points at which it has to be tabulated is independent of the function. Okay so I the beginning of this program I just call a function another function called gauss Legendary, okay and I supply the lower limit and upper limit okay and then that this program would then return to me the x values and the weights at the number of points which I have given, I asked the program to do it.

So what we need to actually find out is how do we compute the x and w's that is the points at which the 0s of the polynomial and rescale it appropriately because I given the limits as this and then weights of the function at that. So if you know how to compute at it between minus 1 and plus 1 then we know by simple scaling we know that we can write it as in this fashion this is the 0s of the polynomial then I know where the functions has to be evaluated by this simple a plus b by 2 minus a minus b by 2 into x_j . So that is not a that is not a big problem which we have to find out is the 0s of the polynomial.

(Refer Slide Time: 50:55)



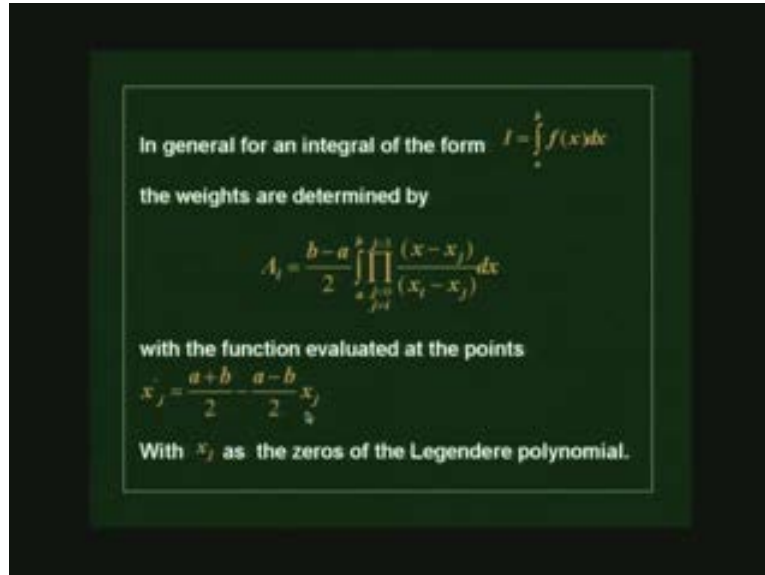
```
n=Npoints;
for(i=0;i<=Npoints;i++)
{
    x[i]=0.0; w[i]=0.0; xp[i]=0.0;
}

gauleg(lower_lmt,upper_lmt,n,w,Npoints)
;
sprintf(fname,"gauleg%d.dat",Npoints);
fp1=fopen(fname,"w");
for(i=1;i<=Npoints;i++)
{
```

28,29-36 30%

Okay so that is the what this gauss legendary actually does, okay so I do the series computation of the 0s of the polynomial by simply using the this method that this is Newton Raphson. So first you have to evaluate p of x and then I evaluate p prime of x, so p prime of x is given by that first I have to get the p of x for any order I given order is what the number of points I am going to ask for it is so if I say the number of points is 4 I have to do a 4th order gauss legendary that is I need four points etcetera. Okay so now this is what this part is doing, so I just it has a simple guess value for a0 so it has a guess value for a0 which is cos of phi into i minus this is the ith 0 is guess into phi minus 1 by four divided by n plus half this is simply a good guess value where it actually works and there are better methods of actually guessing the 0s of a of a Legendary polynomial.

(Refer Slide Time: 52:31)



We will not go into those details, so we are making some guess values of z okay and then we come down and then we just compute then now here so I am adjusting p_1 . So I just start with p_1 equal to 1 that I know and p_2 equal to 0 which I do not know what it is okay and so just put p_2 equal to 0. So p_1 is 1 that is something which I know that first order polynomial is 1. Okay and then p_2 equal to 0, I start with i initialize p_2 equal to 0 and then I compute as a in a loop all the way up to n where n is the number of points at which I want to the now the order of the polynomial that is the order of the polynomial is n , that is n points is taken as n here and then I just use this recursion relation you can see this recursion relation here. Okay that is the recursion relation I am using so 2 times j minus 1 into the guess values x that is I am using that that recursion relation which I have written down here.

To compute p_j plus 1 to p_j , so that is what I am using here. Okay so that recursion relation is used here to compute the value and I know this loop till I reach the order of the polynomial I want to evaluate okay, so n is the number of points I want to evaluate I want to evaluate the function n okay I do this till the order of the polynomial is reached then I use this the derivative of the function here. So this is the derivative of the polynomial which is computed again using that formula which I had which I had just written down because I use this formula to compute the derivative of the polynomial because I already used the recursion relation to compute p_1 and p_2 that is the then I can get the p_1 prime or if I use 4th order I can get p_4 prime from p_4 and p_5 , okay and I can compute p_5 starting using this starting from p_1 equal to 1 and using the recursion relation.

(Refer Slide Time: 52:44)

```
z=cos(3.141592654*(i-0.25)/(n+0
.5));
do {
    p1=1.0;
    p2=0.0;
    for (j=1;j<=n;j++) {
        p3=p2;
        p2=p1;
        p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
    }
    pp=n*(z*p1-p2)/(z*z-1.0
);
    z1=z;
    z=z1-p1/pp;
    67,12-33      86%
```

(Refer Slide Time: 55:41)

$$(j+1)P_{j+1} = (2j+1)xP_j - jP_{j-1}$$
$$P_1 = 1$$
$$P'_j = \frac{j(xP_j - P_{j+1})}{x^2 - 1}$$

x

That is all it is and then I improve on the z by using this Newton Raphson here. So I compute z₁, z₁ minus p₁ by pp. So pp is the derivative and p₁ is the function is the derivative, is the polynomial of the desired order I wanted, okay I am just using p₁ but it is the polynomial of the desired order I want because I gone through this loops n times in the recursion relation and I obtain the p₁ is now my order of the polynomial, I wanted to get and then I can use this to get my 0s okay, so now the 0s are tabulated here for different values.

(Refer Slide Time: 55:43)

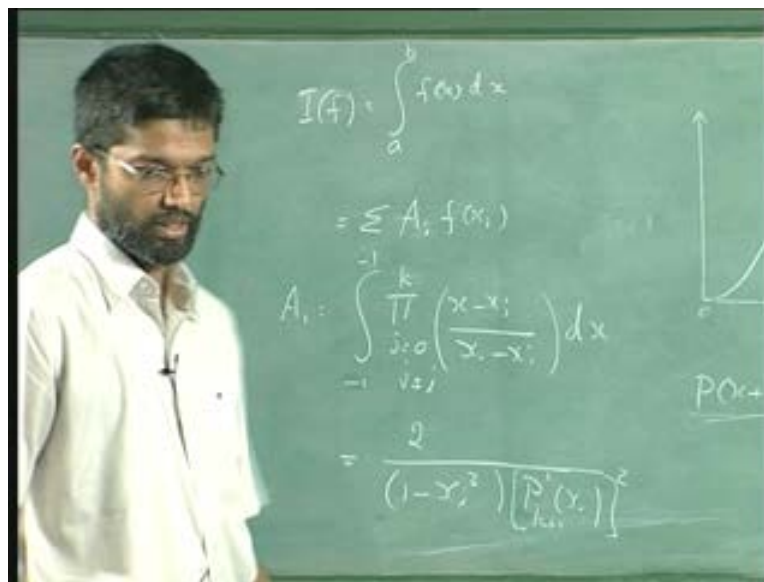
```

p3=p2;
p2=p1;
p1=((2.0*j-1.0)*z*p2-(j-1.0)*p3)/j;
}
pp=n*(z*p1-p2)/(z*z-1.0
);

z1=z;
z=z1-p1/pp;
} while (fabs(z-z1) > EPS);
x[i]=xm-x1*z;
x[n+1-i]=xm+x1*z;
w[i]=2.0*x1/((1.0-z*z)*pp*pp);
w[n+1-i]=w[i];
}
73,7-21 95%

```

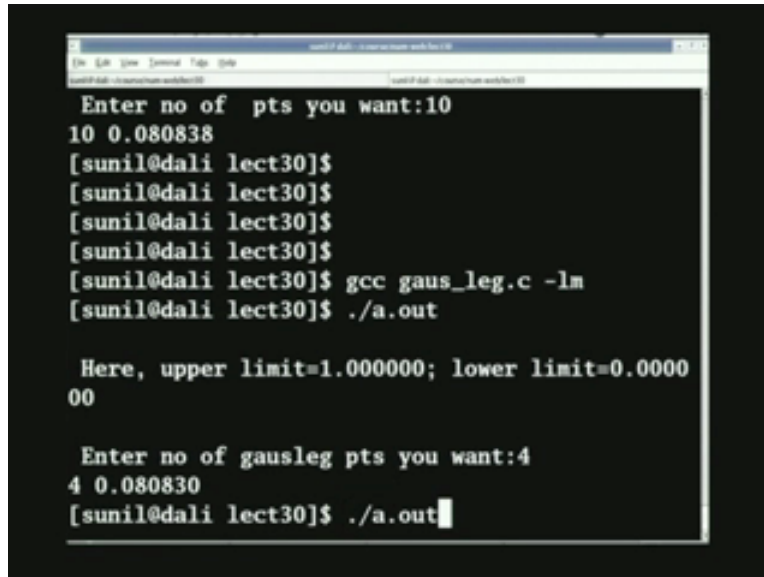
(Refer Slide Time: 56:37)



Okay and the corresponding weights are then calculated using this formula that is 2 by 1 minus x squared into p prime, once I get the p prime in the p this is pretty simple, okay once I got the zeros and the function and the polynomial derivative in that form then this is extremely simple I can write that down as the weights and then at this program just returns that that weight into the main program. So then I will just compute them as a and then I just simply sum over the weights into the function evaluated at that x_i values. Okay now as I see that this x_i values are actually rescaled by here, okay I rescale them okay so x_i values I get is values which are re scaled okay x_m into the rescaled values that is what I am getting. In the actual derivative, in the actual interval a to b, so we just run this

program now, okay so g okay, so we will just run it between the limits say we just run a four points Gauss Legendary we just run a₄ points and you can see that we get amazing accuracy we get “.080830” by running a₄ point Gauss Legendary.

(Refer Slide Time: 56:38)

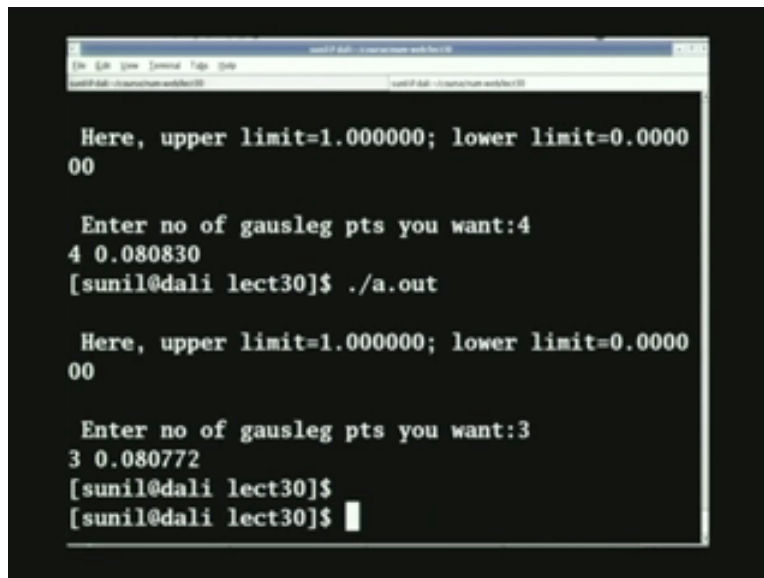


```
sunil@dali ~$ ./a.out
Enter no of pts you want:10
10 0.080838
[sunil@dali lect30]$
[sunil@dali lect30]$
[sunil@dali lect30]$
[sunil@dali lect30]$
[sunil@dali lect30]$ gcc gaus_leg.c -lm
[sunil@dali lect30]$ ./a.out

Here, upper limit=1.000000; lower limit=0.000000

Enter no of gausleg pts you want:4
4 0.080830
[sunil@dali lect30]$ ./a.out
```

(Refer Slide Time: 57:30)



```
Here, upper limit=1.000000; lower limit=0.000000

Enter no of gausleg pts you want:4
4 0.080830
[sunil@dali lect30]$ ./a.out

Here, upper limit=1.000000; lower limit=0.000000

Enter no of gausleg pts you want:3
3 0.080772
[sunil@dali lect30]$
[sunil@dali lect30]$
```

So we could we could get even very good accuracy by 3 to 4 is tremendous improvement so by 4 points we are getting a tremendous improvement that is the actual value remember is “.080831” okay that is the actual value we have already got up to a 6 decimal place by doing a 4 point Gauss Legendary. So you can see that this method is the

most accurate okay provided you can tabulate your function the points at which you want to use okay so we stop that here.

So we could actually compute this this for different functions and see how the compare this with the Gauss Legendary with other simpler methods and then and compare the accuracy you obtain in the next lecture. We look at the how to solve differential equations the next step is to actually look at differential equations. So we will look at that in the next lectures.