

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

Lecture – 32
Differential Equations with Specified Boundary Conditions Part 02

(Refer Slide Time: 00:16)

New improved value of the solution $y(x)$ available.

$y_{old} = y$ for all i

→ Calculate NEW IMPROVED VALUES OF y

→ $y_{old} = y$ for all i

$y = y_{init}$ $ll = 0$

do $y_{old} = y; ll = ll + 1$

do $ii = 2, nop - 1$

$y(i) = \frac{1}{(2-10k^2)} [d_1 y_{old}(ii) + d_2 y_{old}(i-1) - d_3 x(i)]$

enddo

→ CONDITION FOR EXIT? $COND = 1$

$if (COND = 1) THEN EXIT.$

enddo

$d_1 = (1 - \frac{5k}{2})$

$d_2 = (1 + \frac{5k}{2})$

$d_3 = 10k^2$

$x=0$ $x=1$

$y=0$ at $x=0$ $y=100$ at $x=1$

INITIAL guess

Now, here you would see that even before I enter this do loop where I keep on iteratively correcting the value of or improving the value of y for all the 'i's. You can set a variable ll some dummy variable equal to 0 and every time this is repeated. So, far if this condition is not satisfied then it will go back here, then you update ll equal to $ll + 1$ what is the advantage that you get? Once it comes out of this loop, when once this condition is satisfied you get to know in how many iterations, you get a solution to the differential equation.

(Refer Slide Time: 00:59)

$y(i) - y_{old}(i) < 10^{-3}$ for all values of i then $Cond = 1$.

```

    COND = 1.
    do ii = 2, ncp-1
    if (y(ii) - y_old(ii)) .gt. 10^-3 cond = 0.
    end
    if (COND.EQ.1) EXIT
    
```

CHECKING CONVERGENCE CRITERIA.

LIMIT.

JACOBI METHOD.

$$y' = \frac{y_{i+1} - y_{i-1}}{2h}$$

$$y'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

ERROR $\sim h^3$

So no point in setting LIMIT = 10^{-8} if $h = 0.05$ ERROR IN $y_i \approx 0.0025$.

Now, this method is called the Jacobi method the Jacobi method to calculate the solution of the difference equation what you have done is essentially using these two relationship, relations have substituted $d^2 y / dx^2$ and dy / dx in the differential equation, you calculate y_i for each of the i values right then you keep then you save this new values obtained of y_i into y_{old} and use y_{old} to calculate the new values of i iteratively right. Now the question is what limits should you set here? So, I said 10 to the power minus 3 it was arbitrarily chosen.

Now, the expressions that you use here they have errors at the order of h cube where h is the size of your the grid points basically x_2 minus x_1 or x_3 minus x_2 right. So, if the error anyways this is order of h cube you choose h equal to 0.05 there is no point in setting, this limit to be 10 to the power minus 8 because anyway you will have large errors orders of h cube.

So, you are never going to get a convergence if you expect that you shall accept the value if this limit is 10 to the power minus 8 only this condition is satisfied for all the values of y_i minus y_{old} for all values of i and if limit is equal to 10 to the power minus 8 when I am going to get it. You do expect an error in the value of y_i for each value of i order of 0.0025 because h is equal to 0.05 right x squared is 0.0025 . So, there is no point in setting a limit lower than this value you are not going to get; you are going to get a convergence solution right.

(Refer Slide Time: 03:19)

GAUSS SEIDEL.

$i=1$ $i=2$ $i=3$ $i=4$


y_1 y_2 y_3 y_p

→ UPDATE ALL VALUES OF y_i USING $y_{old}(i+)$ and $y_{old}(i-)$.

and then $y_{old} = y$ → CALCULATE NEW VALUES OF y_i

BUT $y(i-1)$ UPDATED BEFORE $y(i)$.

So $y(i) = \frac{1}{c_i} [d_i - y_{old}(i+1) + d_2 y(i-1) + d_3 x(i)]$.



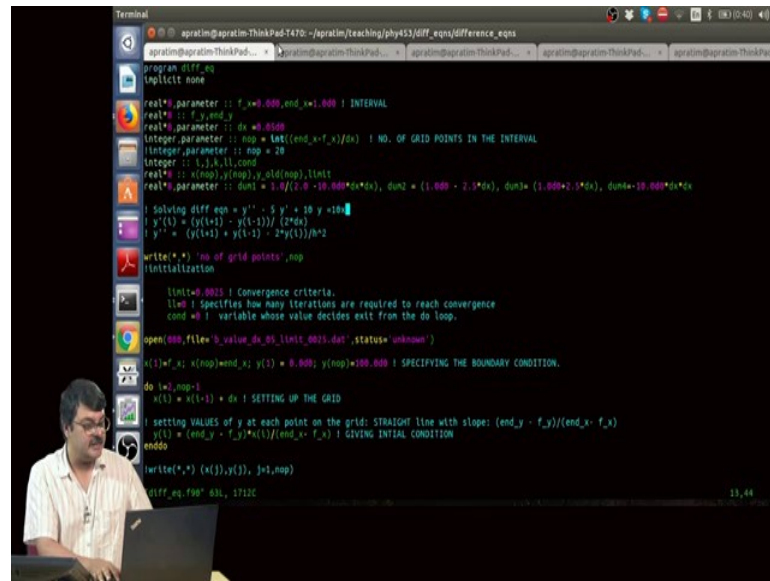
The other point is that in the Jacobi method you were calculating y at each point in i saving it into y old and we are using y i to calculate and we are using basically y old at i plus 1 and i minus 1 to calculate y i . Now, while you are updating y i from say y equal to 2, i equal to 2 to i equal to n minus 1 right before you update y at i you already have then updated value of y at i minus 1.

Now in this expression; in this expression y old at i plus 1 you do not have yet when you are running this loop right here both were old, but as you are updating you already when you have reached i you already have the new value of y i and in this expression if you use the new value of y i minus 1 then it is called the Gauss Seidel method and this often leads to a quicker convergence of the solution ok.

Now, you cannot use y old sorry the new value of y at i plus 1 this you do not have access to because you are basically changing i from 2 to n minus 1, but you do have an access to the newer value of y at i minus 1. So, that is the only expression that I have substituted here in this expression x i of course, does not i mean it is independent of the iteration it is basically just dependent upon i and not upon the iteration, but you can get more accurate values of y i and your solution can converge in a quicker manner if you use the Gauss Seidel method ok.

Now, with this background let us actually go to the code and see how this has been implemented and have an idea of the solution ok.

(Refer Slide Time: 05:53)

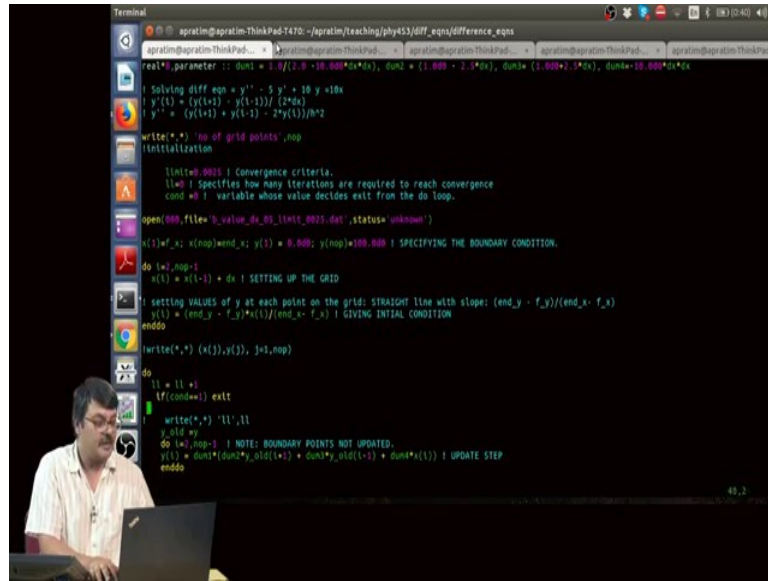


So, here is the code program diff equation and then basically you have the obvious implicit none and this is where I have defined the interval f_x is a 0. So, that is one end of the interval and end x is 1 the other end of the interval dx or h I have said to be 0.05 as we were discussing in class, number of points between the 2 end points is number of points is basically end x minus f_x minus dx divided by dx right.

And here you have some arrays defined there is the size of the arrays is y_{old} and y and the length of the arrays as number of points. So, if you change h to say or dx to 0.01 then the size of the array automatically changes. So, if you have h or dx equal to 0.01, then the number of points between the interval will be 100 and nop will be 100 right.

Now, these are those dummy variables called d_1 d_2 d_3 right which shall sit in front of the p which will be basically pre-factors in front of y_{i+1} y_{i-1} and so on so forth. So, those have been defined right at the beginning of the code so, that you do not have to calculate it multiple times.

(Refer Slide Time: 07:35)



```
Terminal
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/difference_eqns
real*8 parameter :: dnx1 = 1.0/(2.0*10.000*0*0*0), dnx2 = (1.000*2.1*0*0), dnx3 = (1.000*1.1*0*0), dnx4 = 10.000*0*0*0
! Solving diff eqn = y'' - 5 y' + 10 y = 10x
! y'(1) = (y(1+1) - y(1-1))/(2*dnx)
! y'' = (y(1+1) + y(1-1) - 2*y(1))/h^2
write(*,*) 'no of grid points', nop
! Initialization
! limit=0.0025 ! Convergence criteria.
! ll=1 ! Specifies how many iterations are required to reach convergence
cond = 0 ! variable whose value decides exit from the do loop.
open(000, file='b_value_vs_x_limit_0.0025.dat', status='unknown')
x(1)=f_x; x(nop)=end_x; y(1) = 0.000; y(nop)=100.000 ! SPECIFYING THE BOUNDARY CONDITION.
do i=1,nop-1
x(i) = x(i-1) + dnx ! SETTING UP THE GRID
! setting VALUES of y at each point on the grid: STRAIGHT line with slope: (end_y - f_y)/(end_x - f_x)
y(i) = (end_y - f_y)*(i)/(end_x - f_x) ! GIVING INITIAL CONDITION
enddo
write(*,*) (x(i),y(i)), i=1,nop
do
ll = ll + 1
if(cond==0) exit
! write(*,*) 'll',ll
y_old = y
do i=1,nop-1 ! NOTE: BOUNDARY POINTS NOT UPDATED.
y(i) = dnx*(dnx*y_old(i+1) + dnx*y_old(i-1)) + dnx*y(i) ! UPDATE STEP
enddo
enddo
```

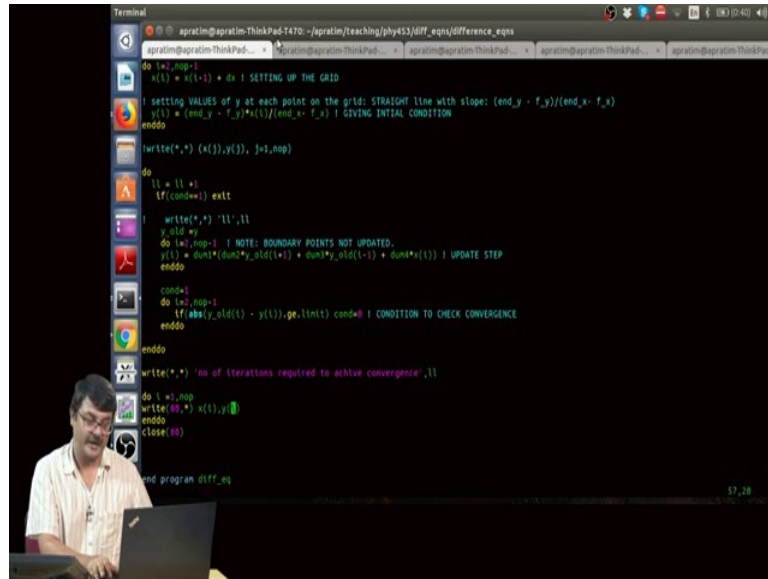
Now, here is a convergence criteria limit equal to 0.0025 because I have set h equal to 0.05 here is your ll which will give me in how many iterations the code has or the code has achieved convergence for a particular value of h right. And this is the condition the variable whose value decides whether you are going to exit this open ended do loop once the system has converged right.

Here I am specifying the boundary condition that y equal to 1 1. So, the number of grid points is number of point basically nop and y at 1 is 0 right at x equal to 0, y equal to at x equal to 0, i equal to 1 and y is equal to 0 and at x equal to 1, i equal to number of points nop and y is 100 and I have chosen a straight line right as the initial condition.

So, that is what I am doing here I am setting up the grid the value of x i for the different values of i by setting d x right x equal at i equal to 1, x equal to 0 at i equal to 2 the value of x will be 0.05 so, that is automatically set here. And the value the initial guess of y over the entire range is basically set here right it is basically a straight line with the appropriate slope end y.

So, basically $y_2 - y_1$ divided by $x_2 - x_1$ and that is exactly what is written here and the value of y_i is basically y equal to $m x$ so, this equation of a straight line. So, for each value of x_i the value of y_i has been calculated nothing more than that and here is the actual do loop right.

(Refer Slide Time: 09:49)



```
Terminal
apratim@apratim-ThinkPadT470: ~/apratim/teaching/phy453/diff_eqs/difference_eqs
apratim@apratim-ThinkPad: ~
do i=1,np-1
  x(i) = x(i-1) + dx ! SETTING UP THE GRID
! setting VALUES of y at each point on the grid: STRAIGHT line with slope: (end_y - f_y)/(end_x - f_x)
  y(i) = (end_y - f_y)*(i)/(end_x - f_x) ! GIVING INITIAL CONDITION
enddo
write(*,*) (x(i),y(i)), i=1,np)
do
  ll = ll + 1
  if(cond=1) exit
! write(*,*) 'll',ll
  y_old = y
  do i=np-1 ! NOTE: BOUNDARY POINTS NOT UPDATED.
    y(i) = dunt*(dunt*y_old(i+1) + dunt*y_old(i-1) + dunt*y(i)) ! UPDATE STEP
  enddo
  cond=1
  do i=np-1
    if(abs(y_old(i) - y(i)).ge.lim) cond=0 ! CONDITION TO CHECK CONVERGENCE
  enddo
enddo
write(*,*) 'no of iterations required to achieve convergence',ll
do i=1,np
  write(99,*) x(i),y(i)
enddo
close(99)
end program diff_eq
```

So, ll has been set equal to 0 right at the beginning of the code. So, this is the do loop to the open ended do loop ll equal to ll plus 1 if condition remains equal to 1, then x z this I have written right at the beginning. And here is where I write y old equal to y this is the actual line where you update the values of y where I am using y old at i plus 1 and y old at i minus 1 and x i at the end of this step I have the updated values of y over the entire range.

At each of these grid points and I set $cond$ equal to 1 and do i equal to 2 to number of point minus 1 at each point I am checking. If the absolute value of y old minus y i is greater than the limit then I set $cond$ equal to 0 this is the condition to check convergence right. And here I write down the number of iterations required to achieve convergence which is ll once it has come out of the loop, I have a fixed value ll they write down the value of x i and y over the entire range and i plot this right.

(Refer Slide Time: 11:18)

```
Terminal
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
no of iterations required to achieve convergence 103378
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns vi gauss_setdel.f90
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns vi gauss_setdel.f90
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran gauss_setdel.f90 -o gauss_setdel
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./gauss_setdel
no of iterations required to achieve convergence 335
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran gauss_setdel.f90 -o gauss_setdel
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./gauss_setdel
no of iterations required to achieve convergence 513370
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran gauss_setdel.f90 -o gauss_setdel
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./gauss_setdel
no of iterations required to achieve convergence 144548
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns wg
wg: command not found
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ls
backup_gauss_setdel.f90 b_value_dx_05_limit_0025.dat diff_eq.f90 gauss_setdel.f90
b_value_dx_001_limit_0001.dat de.stx fort.73 gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat de.stx.f90 gauss gs_value_dx_001_limit_0025.dat
b_value_dx_01_limit_0025.dat diff_eq gauss_setdel gs_value_dx_05_limit_0025.dat
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns xmgrace b_value_dx_05_limit_0025.dat b_value_dx_01_
.dat b_value_dx_01_limit_0001.dat b_value_dx_01_limit_0001.dat gs_value_dx_05_limit_0025.dat k
[2] 10325
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ls
backup_gauss_setdel.f90 de.stx fort.73 gs_value_dx_001_limit_0001.dat
b_value_dx_001_limit_0001.dat de.stx.f90 gauss gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq jacobi.gs.apr gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq.f90 gauss_setdel gs_value_dx_05_limit_0025.dat
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns xmgrace b_value_dx_05_limit_0025.dat b_value_dx_01_
.dat b_value_dx_01_limit_0001.dat b_value_dx_01_limit_0001.dat gs_value_dx_05_limit_0025.dat k
[2] 10325
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ls
backup_gauss_setdel.f90 de.stx fort.73 gs_value_dx_001_limit_0001.dat
b_value_dx_001_limit_0001.dat de.stx.f90 gauss gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq jacobi.gs.apr gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq.f90 gauss_setdel gs_value_dx_05_limit_0025.dat
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns xmgrace b_value_dx_05_limit_0025.dat b_value_dx_01_
.dat b_value_dx_01_limit_0001.dat b_value_dx_01_limit_0001.dat gs_value_dx_05_limit_0025.dat k
[2] 10325
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
no of iterations required to achieve convergence 623
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran diff_eq.f90 -o diff_eq
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./diff_eq
no of iterations required to achieve convergence 7202
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran diff_eq.f90 -o diff_eq
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./diff_eq
no of iterations required to achieve convergence 17306
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
```

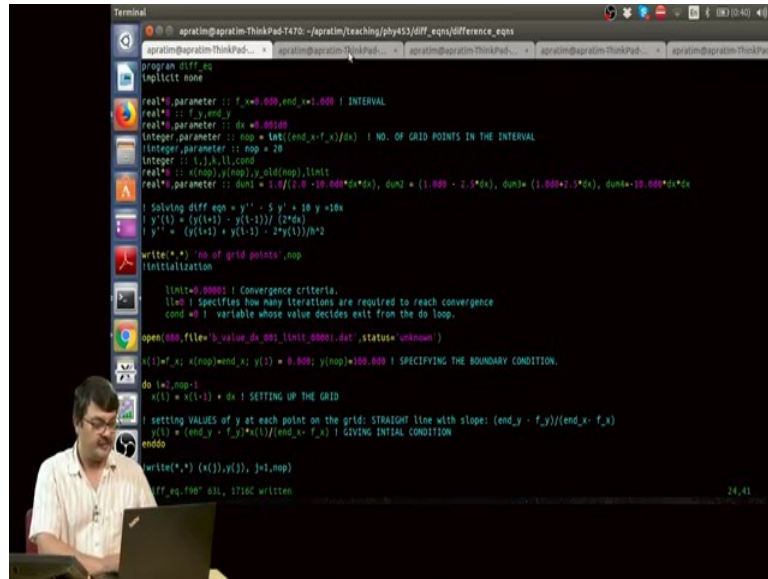
So, I compile this differential equation. So, in g fortran, so, this is the compilation of this code.

(Refer Slide Time: 11:37)

```
Terminal
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
no of iterations required to achieve convergence 144548
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns wg
wg: command not found
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ls
backup_gauss_setdel.f90 b_value_dx_05_limit_0025.dat diff_eq.f90 gauss_setdel.f90
b_value_dx_001_limit_0001.dat de.stx fort.73 gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat de.stx.f90 gauss gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq jacobi.gs.apr gs_value_dx_001_limit_0001.dat
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns xmgrace b_value_dx_05_limit_0025.dat b_value_dx_01_
.dat b_value_dx_01_limit_0001.dat b_value_dx_01_limit_0001.dat gs_value_dx_05_limit_0025.dat k
[2] 10325
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ls
backup_gauss_setdel.f90 de.stx fort.73 gs_value_dx_001_limit_0001.dat
b_value_dx_001_limit_0001.dat de.stx.f90 gauss gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq jacobi.gs.apr gs_value_dx_001_limit_0001.dat
b_value_dx_01_limit_0025.dat diff_eq.f90 gauss_setdel gs_value_dx_05_limit_0025.dat
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns xmgrace b_value_dx_05_limit_0025.dat b_value_dx_01_
.dat b_value_dx_01_limit_0001.dat b_value_dx_01_limit_0001.dat gs_value_dx_05_limit_0025.dat k
[2] 10325
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
no of iterations required to achieve convergence 623
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran diff_eq.f90 -o diff_eq
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./diff_eq
no of iterations required to achieve convergence 7202
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns gfortran diff_eq.f90 -o diff_eq
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns ./diff_eq
no of iterations required to achieve convergence 17306
apratin@apratin-ThinkPad-T470: ~/apratin/teaching/phy453/diff_eqns/difference_eqns
```

And basically what I have done carefully is that for a particular value of dx or h 0.05 a limit I have saved I am going to save the value of the solution x and y for different values of i in this file right. So, if I run it you see that I have essentially 20 grid points and the number of iterations required to achieve convergence was 623.

(Refer Slide Time: 12:12)



Now, suppose I changed this two. Now suppose I say well I choose my h to be 0.01, but I keep the same convergence right because this is the same and then of course, I want to change the name of the file where I am going to store the data right because I want to compare the data for these different values of h and the convergence factors.

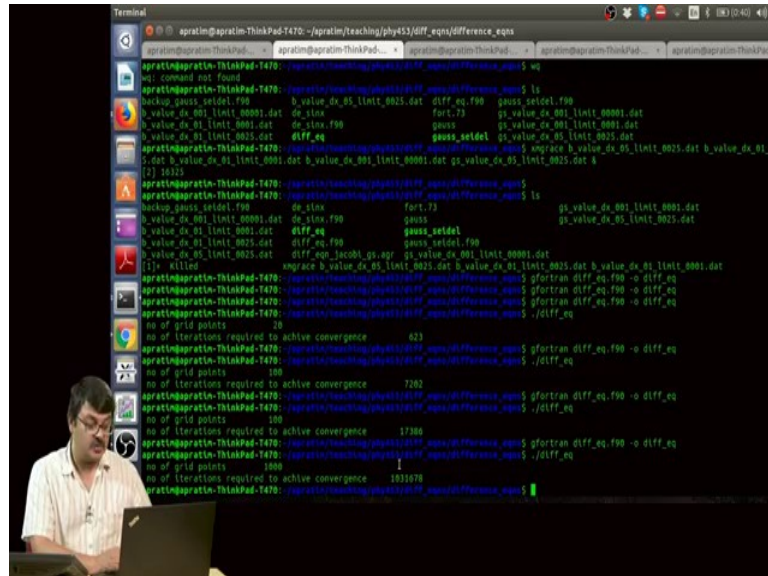
So, here what I have said is I am not going to change the limit, but I have changed the value of h if you like or dx if you like and I compile it again and run it again. Here I have now I have 100 points in the between x equal to 1 and x equal to 0 and now I needed 7,000 and 2 points 2 iterations to get converge to the right value right now what happens if I. So, now, have a smaller value of s , the tolerance should be even less. So, here I am changing. So, I am keeping dx to be the same as 0.01, but this limit I can easily change to where am I defining limit yes.

So, I am changing the limit 2 points x square again right and then I must change the name of this file as well because limit has been changed; dx remains the same as before and if I compile it and run it needs 17,386 iterations before it converges with a lower tolerance because I have lowered the because I have a smaller value of h , I have also lowered the tolerance for it right.

And now I can go to even smaller values suppose 0.0 two 0s as 1 and 1. So, 10 to the power minus 3 and I set my limit as just 10 to the power minus 5 say right though I can set the limit to be 10 to the power minus 6 as well, but I just choose to cheaper higher

tolerance and correspondingly I am changing the name of the files where dx equal to 0.001 and the limit is point. So, I am not explicitly written the point as 0000110 to the power minus 5 all right.

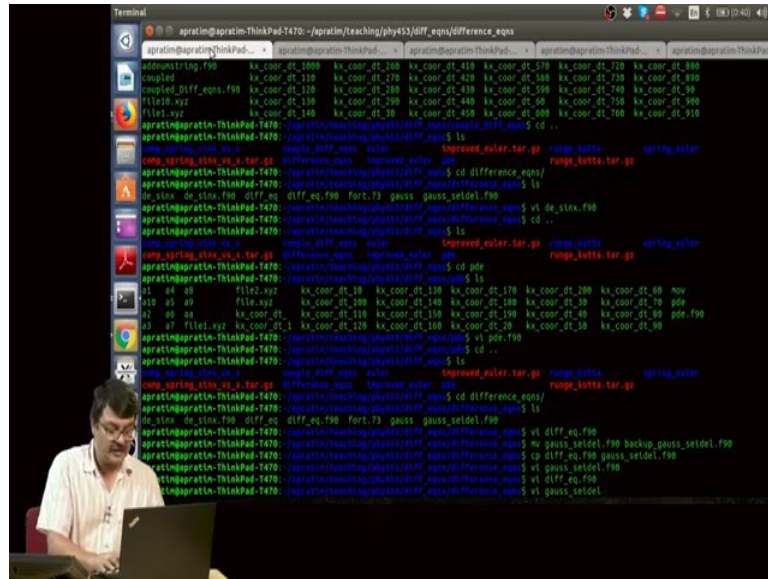
(Refer Slide Time: 15:23)



And if I run it, it does takes a number of grid points is 1000. So, bit the range between x equal to 0 and x equal to 1 has been changed has a 1000 grid point. So, basically the number of y calculations have significantly increased at each of these points, moreover the number of iterations required to basically converge to the right value with a higher accuracy because I have both decreased h.

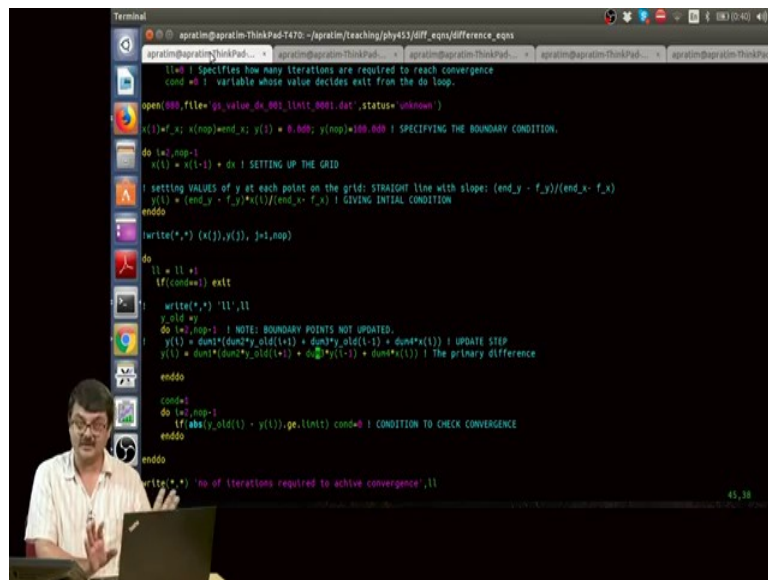
And hence I am allowed to decrease the tolerance and it requires nearly 10 lakhs right this is 10 lakh equations iterations to reach the; to reach the required accuracy. Basically it keeps on iterating it so many times before reach converges where the difference in the values of y old and y i at each of these i points is less than 10 to the power minus 5. So, this is supposed to be a much more accurate value.

(Refer Slide Time: 16:41)



Now, if you had Gauss Seidel. So, what would be the difference? Just all that you have in Gauss Seidel is.

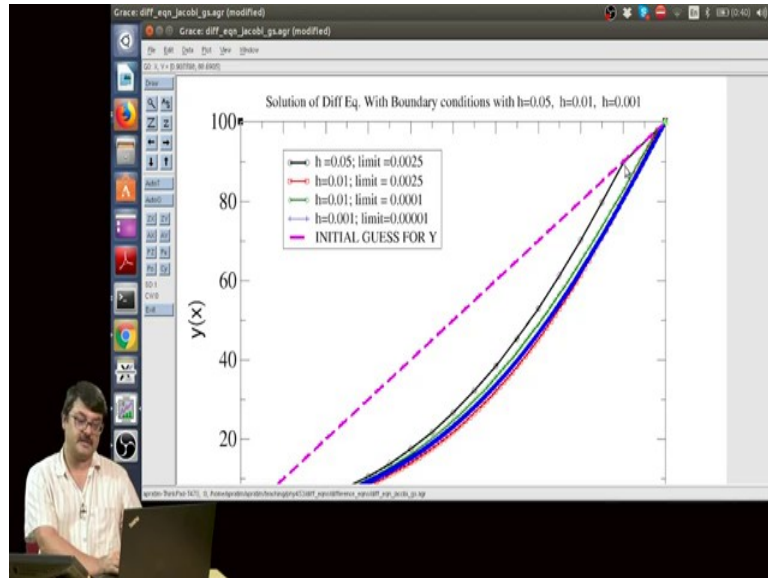
(Refer Slide Time: 16:51)



Basically the previous code it is just that if you compare this y old right has been changed by to y i minus 1. Because in the previous iteration when I run a loop from 1 I equal to 2 to nop minus 1, I already have the new updated value of y at i minus 1 though not the updated value of y at i plus 1. So, here I use the old value, but here I use the new value of y i and this is the only difference between Gauss and Seidel and you can do the

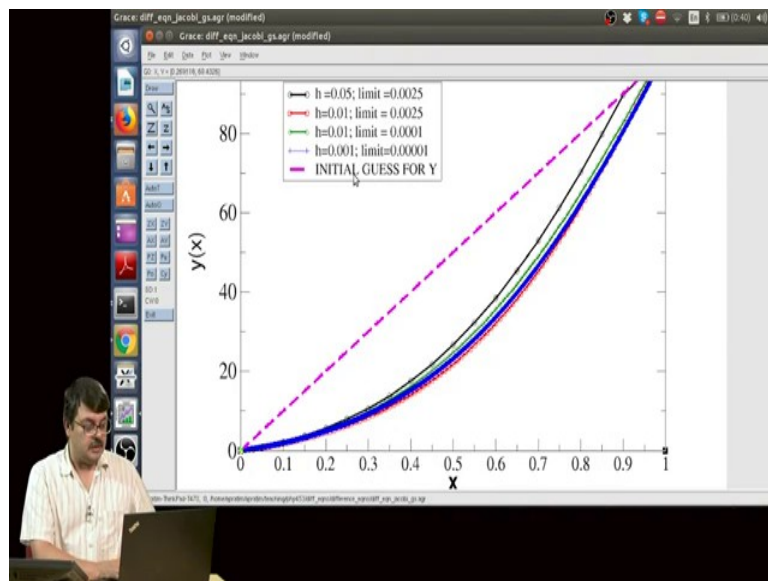
same thing and you will see that it will have a lower number of iterations to reach the converging value with the converged value.

(Refer Slide Time: 17:46)



I have already done that and I have plotted it for you right. So, here what I have done is solution of the differential equation with boundary conditions with h equal to 0.05, 0.01 and 0.001.

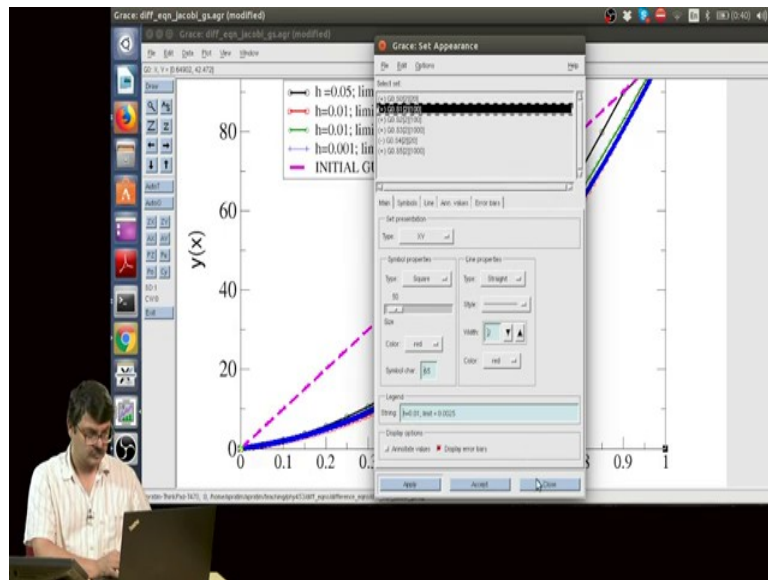
(Refer Slide Time: 17:58)



And here I am plotting y as a function of x versus x and this was this line is your initial straight line which was the guess with the values of y that I calculated for each value of i,

which was the initial guess bit for basically x equal to 0, y equal to 0 and at x equal to 1 y equal to 100 right. And if you have h equal to 0.05 and the limit is 0.0025 basically h square, this is the solution that you obtain after a certain I think 673 iterations or so, as we saw previously right.

(Refer Slide Time: 18:49)



Now, as we change. So, when h equal to 0.01 and limit is 0.0025, you have this red curve right. So, this is this I made more accurate, but I have not changed the limit then you have essentially this red curve when I change h equal to 0.01 and limit equal to point like 10 to the minus 4 compared corresponding to h square, you get this green curve. And when you have essentially h equal to point 10 to the power minus 3 right and limit you set to be 10 to the power minus 5 and it ran for 10 lakh iterations if you remember then you essentially have this blue curve which you can expect to be the most accurate.

When you have h equal to 0.05 of course, you have discretization errors here hardly surprising right because you have only 20 grid points between 0 and 1. But when you have h equal to 0.01 your convergence might more take more time, but it would be more accurate right and when you have h equal to 0.001 your tolerance is even lower and this blue one is expected to be the most accurate data right. So, you can try this out for various values of h and tolerance of course, there is no point in setting tolerance lesser than h square.

Now, the last thing I want to discuss in this class is that since these expressions of y' and y'' are accurate only as h^3 the error is an h^4 , then suppose you were had need to calculate more and more accurate values of the solution to the differential equation. Of course, one method would be to go to lower values of h and you already saw that over a such a small range and for a relatively simple differential equation, it took quite a few seconds around 10 seconds to get the solution.

And especially if you now increase the range and have more complicated differential equation it would take more time right. As you decrease and decrease h to get to the accurate solution. Is it possible to have a higher value of h lesser number of grid points, but yet obtain and more accurate solution y as a function of x at each of these grid points in that case you basically what you need is more accurate expressions of y' and y'' so, that you can use higher values of h without compromising on the accuracy or you can have lower tolerance.

(Refer Slide Time: 21:42)

MORE ACCURATE EXPRESSIONS FOR THE $Y'(X)$ and $Y''(X)$.
 → SO THAT YOU CAN USE HIGHER VALUES OF h (grid interval).

$$\frac{dy}{dx} = \frac{y(x-2h) - 8y(x-h) + 8y(x+h) - y(x+2h)}{12h} + O(h^4)$$

$$\rightarrow \frac{y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}}{12h} + O(h^4)$$

$$\frac{d^2y}{dx^2} = \frac{-y(x-2h) + 16y(x-h) - 30y(x) + 16y(x+h) - y(x+2h)}{12h^2} + O(h^4)$$

$$\rightarrow \frac{-y_{i-2} + 16y_{i-1} - 30y_i + 16y_{i+1} - y_{i+2}}{12h^2} + O(h^4)$$

Substitute in DIFF. EQNS
 Calculate $y_i = C_1 y_{i-2} + C_2 y_{i-1} + C_3 y_i + C_4 y_{i+1} + C_5 y_{i+2}$

And more accurate values of $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ have been derived in this book that I have already referred to (Refer Time: 22:12) you can look up the derivation and what you have is $\frac{dy}{dx}$ can be written as y at x minus $2h$ minus $8y$ at x minus h plus $8y$ at x plus h means and y at x plus $2h$.

So, previously all the expressions were calculated from just the first neighbor on the left and first neighbor on the right on the other hand here what you have to do? You are

taking the help of the y at a grid point 2 points removed 2 points to the left and 2 points to the right so, right. So, you are basically calculating your averaging $d y$ by $d x$ over a larger range in the grid point right.

And of course, this it can be written as y_{i-2} and this will be written as y_{i-1} and y_{i+1} and so on and so forth and when basically you are calculating the new value of y_i you have to use the old values in the Jacobi method and for some cases you can use for the Gauss Seidel of course, you can use the updated values similarly the expression for $d^2 y$ by $d x^2$ is this again it depends upon $x_{i-2} h$.

So, basically 2 points you move to the left and 2 points you move to the right and you have these pre factors here right and this expression. So, the expression here and here they are accurate to the order of h to the power 4. So, the error is you are neglecting higher order terms in the Taylor expansion this is nothing, but the way it is obtained is basically Taylor expansion and you basically match coefficients and neglect terms at the order of h^4 .

So, that is where the error will come in and basically what you have to do if you had wanted a higher accuracy is calculate $d^2 y$ by $d x^2$ well substitute $d^2 y$ by $d x^2$ and divide $d x$ and this differential equation whichever 1 you have to we want to solve right. And now similarly there are expressions for $d^3 y$ by $d x^3$ as well and what you have to do? One then you will get an algebraic equation and then solve for y_i which of course, now will be written in terms of y_{i-2} and y_{i-1} and y_i and y_{i+1} and y_{i+2} right.

So, when you actually use this there are some other set subtleties, but if and when you need to use it you can learn those up remember that since you have $x_{i-2} h$ or $x_{i-1} h$. So, then you have a . So, if you want to calculate the new value of y_i at just next to the boundary. So, suppose y_i equal to 1 the value of y is specified it does not change, but at y equal to 2 here it depends upon y_{i-2} .

So, basically y_{i-1} which is ill defined. So, there are some issues of how to solve it and similarly you will have a similar problem when you do y_{i+1} specially again near the other boundary, but there are methods how to solve for it I am not going to discuss this over in this lecture right.

When you need it you can learn it, that is anyway the scope of this course that we are giving you an introduction of how to develop different algorithms. So, that whenever you need to know more about a topic, you will read it up, learn it and implement it in the computer right. With this I shall end this class today thank you and in the next class we shall discuss partial differential equations.

Thanks.