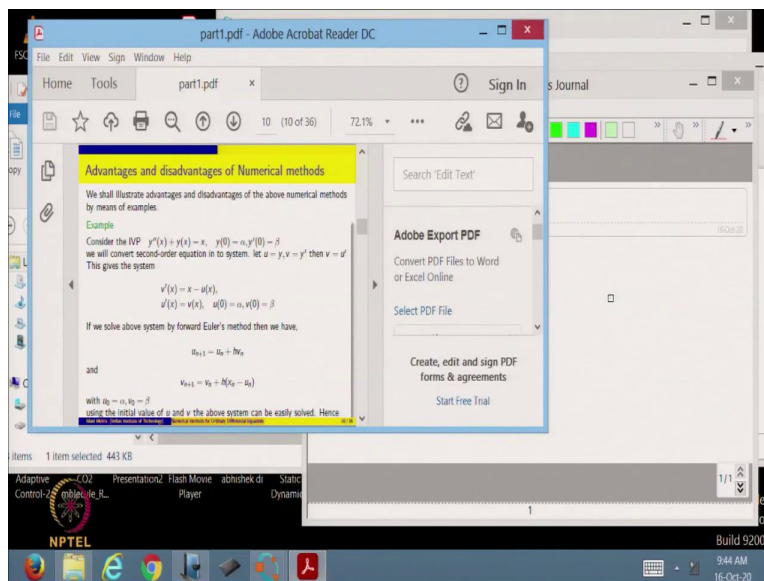


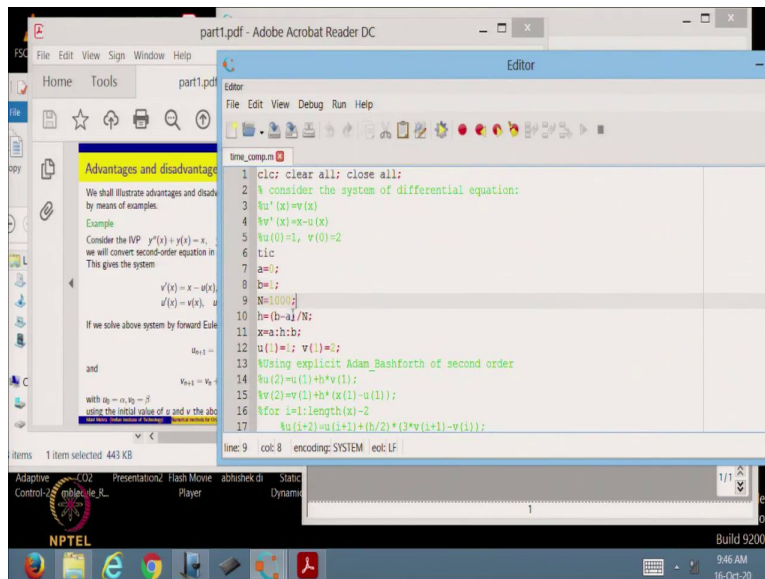
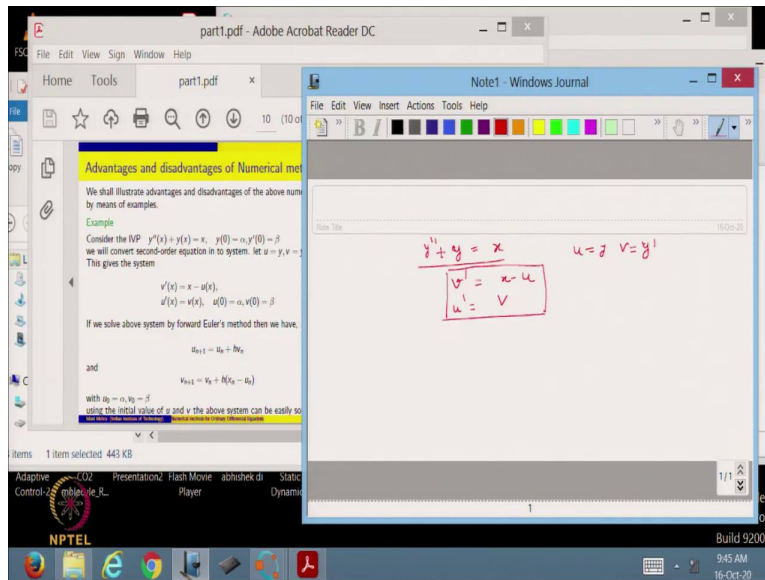
**Scientific Computing Using Matlab**  
**Professor Vivek Aggarwal**  
**Professor Mani Mehra**  
**Department of Mathematics**  
**Indian Institute of Technology Delhi**  
**Delhi Technological University**  
**Lecture 65**

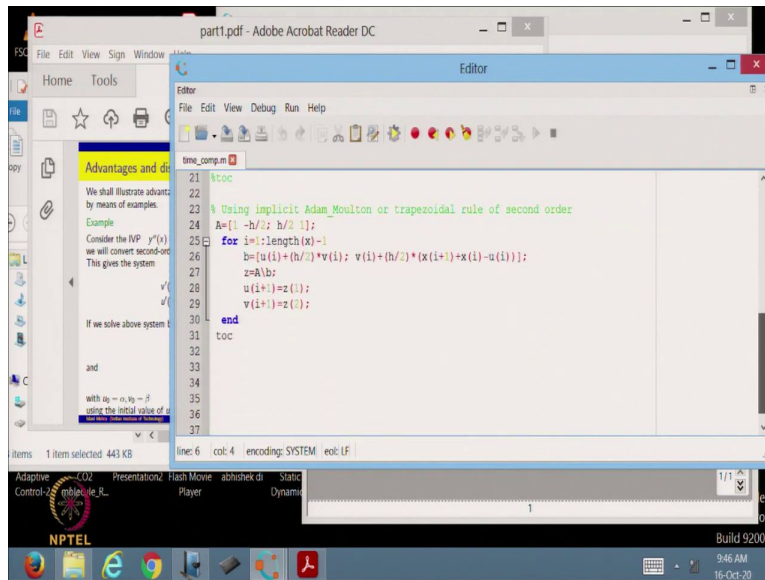
**Advantage of Implicit and Explicit Methods Over Each Other Via Matlab/Octave Codes for Initial Value Problem**

Welcome to all of you in the next class of this course. So, now let us recall what we have done in the last lecture. We have implemented various algorithms under different categories of methods, like Adam Bashforth, Adam Moulton's, Nystrom, Milne Simpson's, under these different categories; we have implemented the example of one particular linear initial value problem and today what we will see? We will see how to solve a system of differential equations with the help of explicit and one implicit method. So based on that we will conclude that what is the advantage of explicit methods. That is the aim of today's lecture.

(Refer Slide Time: 01:23)







So let us take the same example which we considered in one of our lectures which is here  $y'' + y = x$ . So we have to solve a second order differential equation. So, we are converting it

$$v' = x - u$$

to the system of differential equations which we have already seen earlier,  $u' = v$  by assuming  $u = y, v = y'$ , which you can also see from the slides which we have already seen in the lecture.

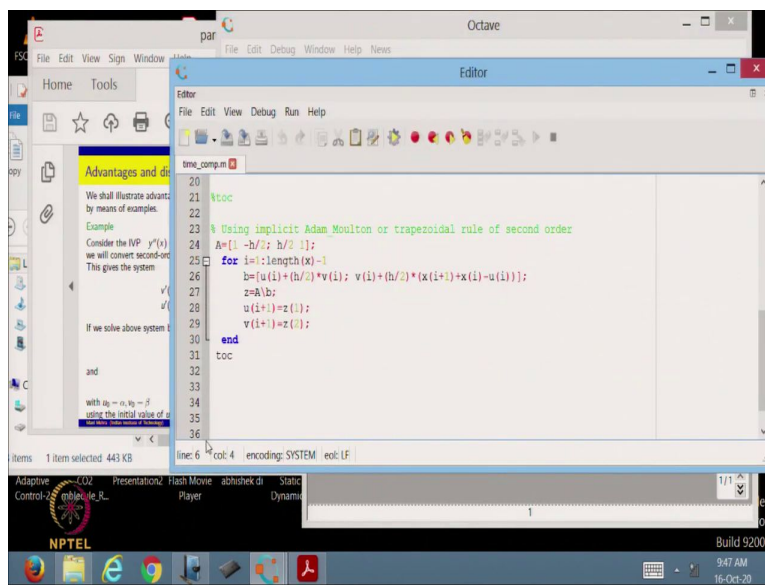
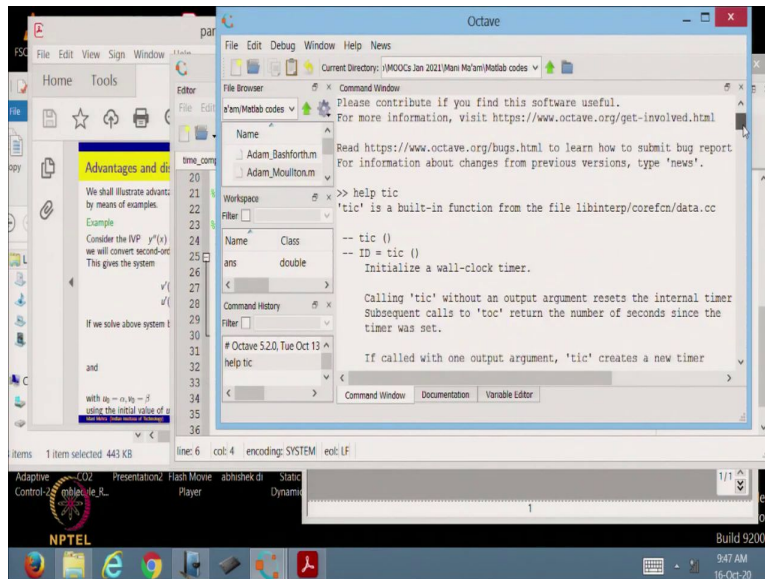
And now, so basically we are trying to implement this system of differential equations. So, let me open the octave. So this is the code which we have implemented. So, here the particular value of this initial conditions  $\alpha$  and  $\beta$  what we have chosen here is 1 and 2. So again we have chosen two variables a and b because a is 0, b is 1 like we were doing in case of a linear initial value problem.

$$h = \frac{b - a}{N}$$

And in this case, we have also chosen  $N = 1000$ , . Again we are creating an array of  $x = a:h:b$ . You are already familiar with these things now, so in the initial conditions we have implemented  $u(1) = 1$  and  $v(1) = 2$  because in Matlab indexing does not start from 0. So now, let us first look at the Adams Moulton method. We are also writing one another function here tic

toc. So tic is here and toc is at the end of the file, which you can see from here. You can see more about this function in an octave by typing `help tic`.

(Refer Slide Time: 3:52)



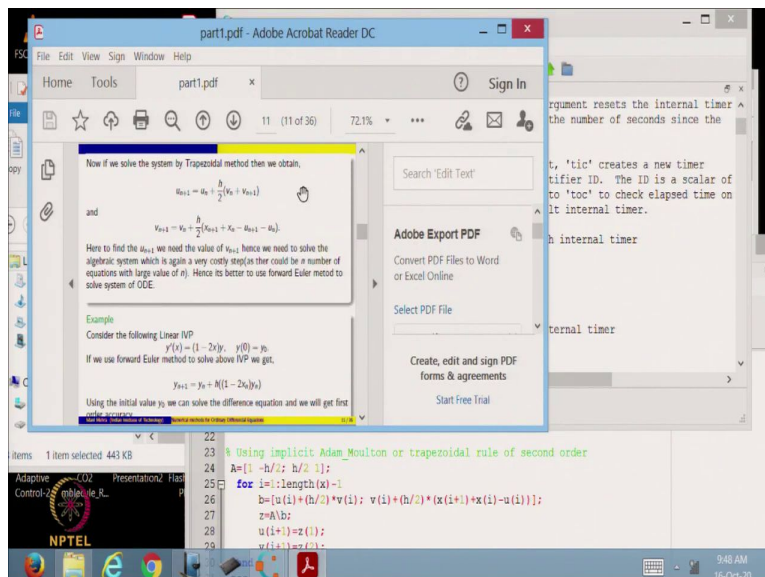
```

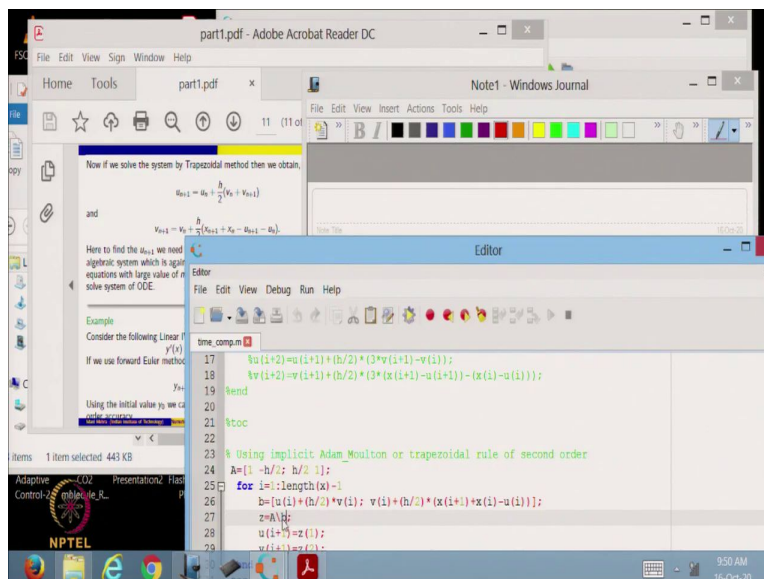
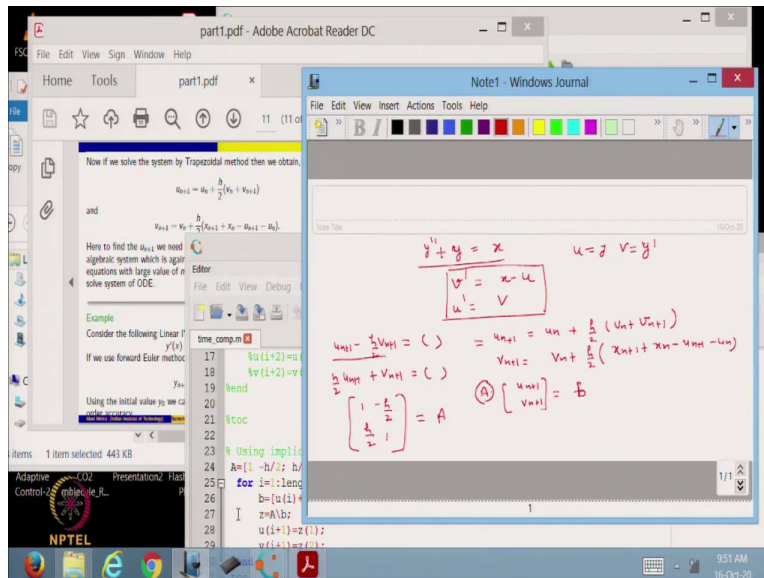
1 clc; clear all; close all;
2 % consider the system of differential equation:
3 %u'(x)=v(x)
4 %v'(x)=x-u(x)
5 %u(0)=1, v(0)=2
6 tic
7 a=0;
8 b=1;
9 N=1000;
10 h=(b-a)/N;
11 x=a:h:b;
12 u(1)=1; v(1)=2;
13 %Using explicit Adams_Bashforth of second order
14 %u(2)=u(1)+h*v(1);
15 %v(2)=v(1)+h*(x(1)-u(1));
16 %for i=1:length(x)-2
17     %u(i+2)=u(i+1)+h*(2)*v(i+1)-v(i);

```

Which I can show you if you want, help tic. So basically tic is an inbuilt function from the file, which basically tells you about the total elapsed time, wall clock time. So basically it will tell you how much time is taken by your programme. So a total that part of the programme where it is ended with toc, which you see here toc is here and at the initial we have typed tic. So it will tell you the total elapsed time of a wall clock.

(Refer Slide time: 5:05)





So, now we have chosen Adam Moulton's, first category of Adam Moulton's method, which is a trapezoidal method, which is a single step, which is self-starting as well. So I do not need to worry at all otherwise, for a two step method, this is a drawback. So now, if you look at what will be the, which we have seen here also, what will be the algorithm if this will be your



$$u_{n+1} = u_n + \frac{h}{2}(v_n + v_{n+1})$$

$$v_{n+1} = v_n + \frac{h}{2}(x_{n+1} + x_n - u_{n-1} - u_n)$$

. So we have already seen the algorithm of the trapezoidal method in the case in earlier lectures.

So now, the thing is we are trying to implement this. So now, if you look at  $v_{n+1}$  terms are also involved in the right hand side. So, basically we ended with this system of differential equations of 2 by 2. So that, for that reason basically we are ending with this, if you look at

$$u_{n+1} = u_n + \frac{h}{2}(v_n + v_{n+1})$$

$$v_{n+1} = v_n + \frac{h}{2}(x_{n+1} + x_n - u_{n-1} - u_n)$$

$$A \begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = b$$

So, basically I can always write in the following way. What will be A? You can figure out yourself and that is what we have implemented here; A is 1-h/2 at least in one

case, first row I can match because this equation will become  $u_{n+1} - \frac{h}{2}v_{n+1}$  is equal to something.

So that is why this has the first row of this matrix will become 1-h/2. Similarly, we can go for a

second row also. So this will become  $\frac{h}{2}u_{n+1} + v_{n+1}$  Rest terms you can collect in the right hand side yourself. So what will be the first row? 1-h/2; the second row will become h/2+1. So this will be our matrix A, which we have implemented here.

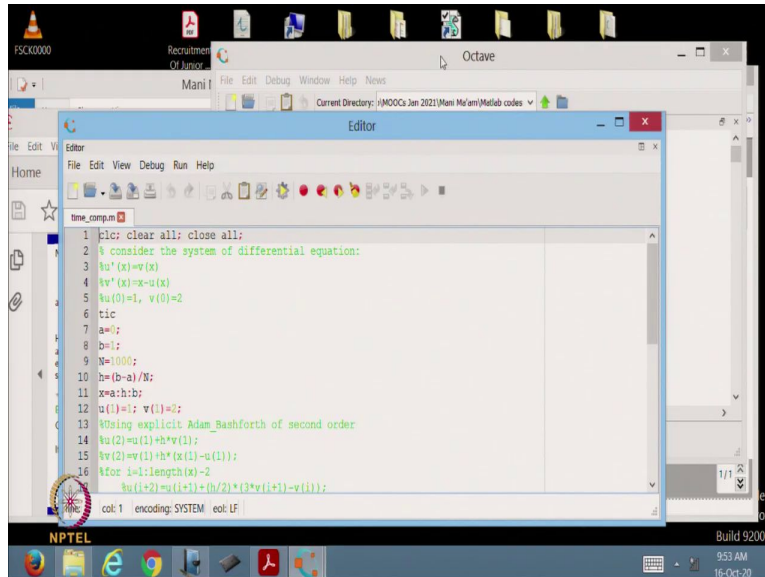
$$A = \begin{pmatrix} 1 & -h/2 \\ h/2 & 1 \end{pmatrix}$$

and similarly, we can collect the right hand side b and finally, what is the uses of this slash operator in a Matlab, not Matlab in Octave, in fact, in Matlab also if some

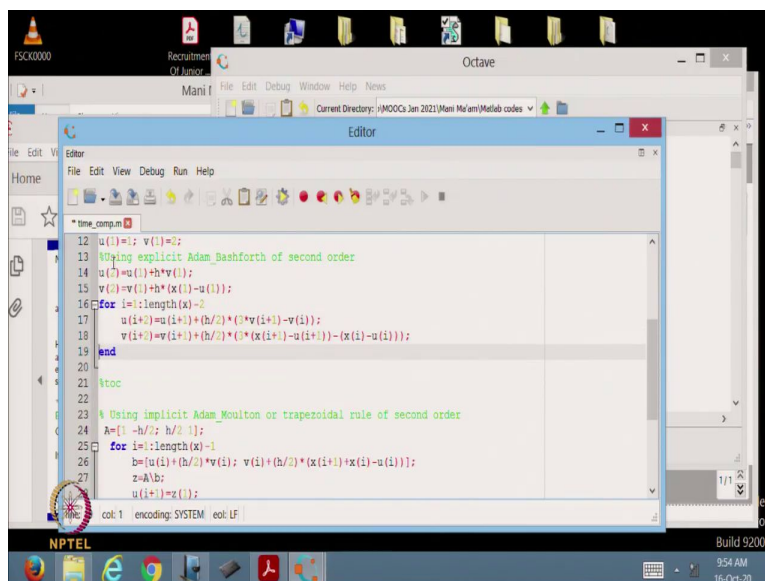


of you are having a Matlab that is what I have already explained earlier that you can run the same codes in Matlab. So slash operator, the application of the slash operator, you must have seen already in earlier lectures. So this is used to solve a system of algebraic equations. So in this case we are ending with a 2 by 2 system, which you can also see because this size of A will be 2 by 2 and after solving we can get  $u_{n+1}$  and  $v_{n+1}$ . So now, let me run this code.

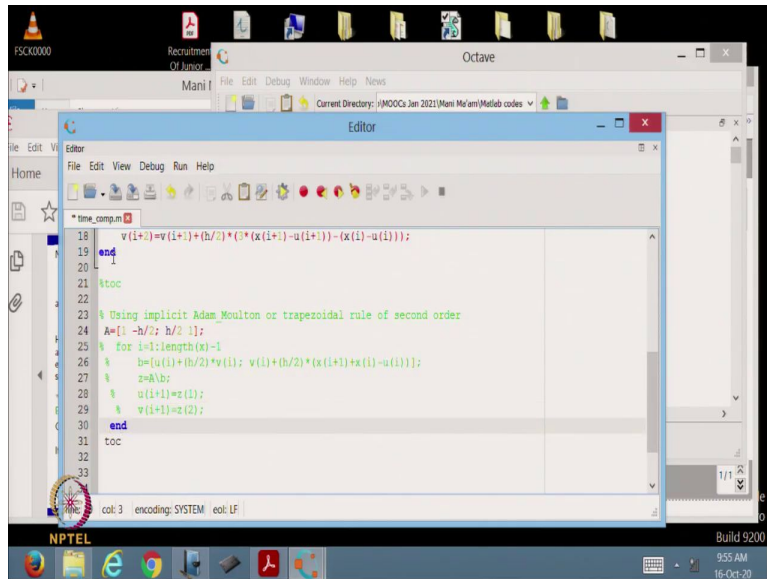
(Refer Slide Time: 8:52)



```
1 clear all; close all;
2 % consider the system of differential equation:
3 %u'(x)=v(x)
4 %v'(x)=x-u(x)
5 %u(0)=1, v(0)=2
6 tic
7 a=0;
8 b=1;
9 N=1000;
10 h=(b-a)/N;
11 x=a:h:b;
12 u(1)=1; v(1)=2;
13 %Using explicit Adams-Bashforth of second order
14 u(2)=u(1)+h*v(1);
15 %v(2)=v(1)+h*(x(1)-u(1));
16 for i=1:length(x)-2
    %u(i+2)=u(i+1)+(h/2)*(3*v(i+1)-v(i));
```



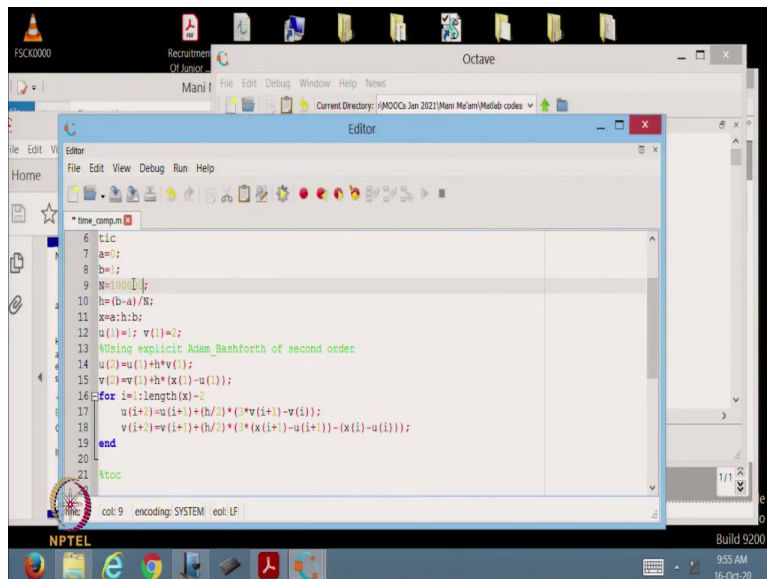
```
17 u(i+2)=u(i+1)+(h/2)*(3*v(i+1)-v(i));
18 v(i+2)=v(i+1)+(h/2)*(3*(x(i+1)-u(i+1))-(x(i)-u(i)));
19 end
20 %toc
21 % Using implicit Adams-Moulton or trapezoidal rule of second order
22 A=[1 -h/2; h/2 1];
23 for i=1:length(x)-1
24     b=[u(i)+(h/2)*v(i); v(i)+(h/2)*(x(i+1)-u(i)));
25     z=A\b;
26     u(i+1)=z(1);
```



The screenshot shows the Octave Editor window with a file named 'time\_comp.m'. The code implements the implicit Adams-Moulton method of second order. It starts with a function definition for 'v(i+1)' and 'u(i+1)' based on 'v(i)' and 'u(i)'. The code includes a loop for 'i' from 1 to 'length(x)-1'. The code is as follows:

```
18 v(i+1)=v(i)+h/2*(3*(x(i+1)-u(i+1))-(x(i)-u(i)));
19 end
20
21 tic
22
23 % Using implicit Adams-Moulton or trapezoidal rule of second order
24 A=[1 -h/2; h/2 1];
25 for i=1:length(x)-1
26     b=[u(i)+(h/2)*v(i); v(i)+(h/2)*(x(i+1)+x(i)-u(i))];
27     z=A\b;
28     u(i+1)=z(1);
29     v(i+1)=z(2);
30 end
31 toc
32
33
```

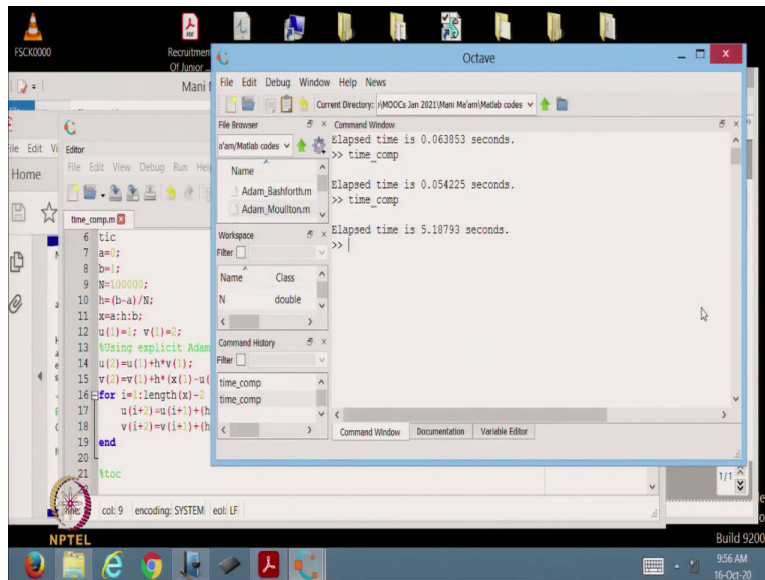
The status bar at the bottom indicates 'col: 3 encoding: SYSTEM eol: LF'.



The screenshot shows the Octave Editor window with a file named 'time\_comp.m'. The code implements the explicit Adams-Bashforth method of second order. It starts with a function definition for 'v(i+1)' and 'u(i+1)' based on 'v(i)' and 'u(i)'. The code includes a loop for 'i' from 1 to 'length(x)-1'. The code is as follows:

```
6 tic
7 a=0;
8 b=1;
9 N=1000;
10 h=(b-a)/N;
11 x=a:h:b;
12 u(1)=1; v(1)=2;
13 %Using explicit Adams-Bashforth of second order
14 u(2)=u(1)+h*v(1);
15 v(2)=v(1)+h*(x(1)-u(1));
16 for i=1:length(x)-2
17     u(i+1)=u(i)+h/2*(3*v(i+1)-v(i));
18     v(i+1)=v(i)+(h/2)*(3*(x(i+1)-u(i+1))-(x(i)-u(i)));
19 end
20
21 tic
22
```

The status bar at the bottom indicates 'col: 9 encoding: SYSTEM eol: LF'.



Yes, now it will work, elapsed time is what we see now, elapsed time is 0.0638 and now, so for a moment I am removing this command `clc`. So that earlier time is also clear to you. This screen is not cleared and so now, let me take out this comment from all the steps of Adam Bashforth algorithm which we have implemented here and let us comment all of them, in fact this A also, yes.

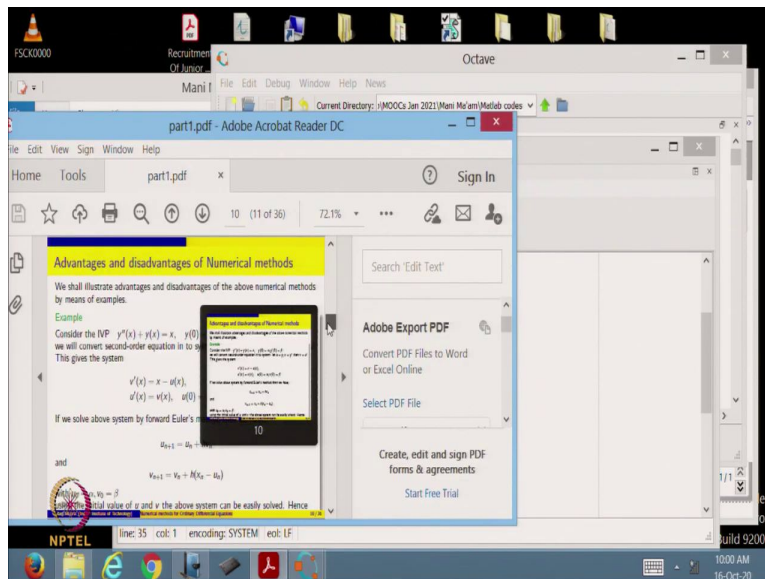
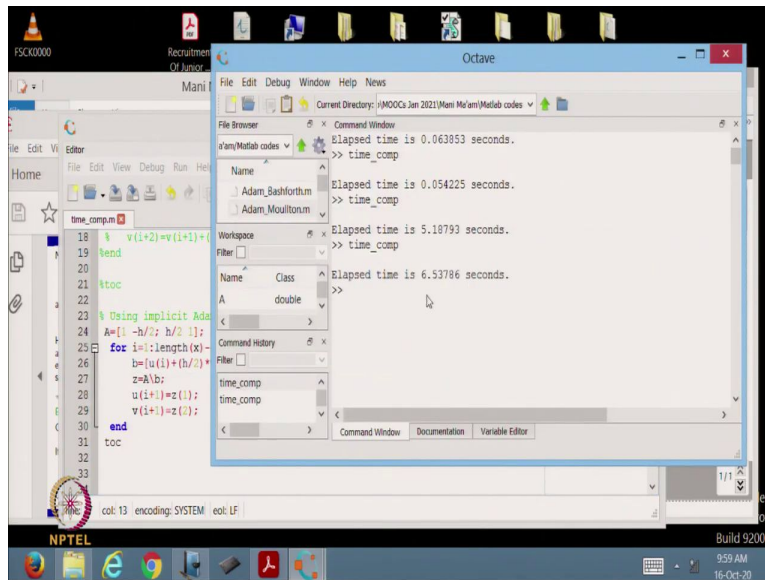
So now if I give the run command you will see that, what is the time, 0.05 with the explicit algorithms. So of course, there is a difference in the elapsed time, but that is very marginal and this will become very significant once you take more points. Let me add two more zeros. So it means we are dividing in so this is one lakh points between 0 to 1.

So now, you will see that that what is the time taken in Adam Bashforth method of second category, because we are comparing both the methods of same order that is why I have chosen here this as a Adam Bashforth method of second category of Adam Bashforth which is the 2 step method as well and for that reason it is not self-starting and that is why we have implemented  $u(2)$  and  $v(2)$  with the help of a forward Euler method. So of course, it will take some time here. The elapsed time is 5 seconds. Roughly it is 5 seconds which is very much visible on the screen.

(Refer Slide Time: 11:36)

```
10 h=(b-a)/N;  
11 x=a:h:b;  
12 u(1)=1; v(1)=0;  
13 %Using explicit Adams_Bashforth of second order  
14 u(2)=u(1)+h*v(1);  
15 v(2)=v(1)+h*(x(1)-u(1));  
16 for i=1:length(x)-2  
17     u(i+2)=u(i+1)+(h/2)*(3*v(i+1)-v(i));  
18     v(i+2)=v(i+1)+(h/2)*(3*(x(i+1)-u(i+1))-(x(i)-u(i)));  
19 end  
20  
21 %toc  
22  
23 % Using implicit Adams_Moulton or trapezoidal rule of second order  
24 A=[1 -h/2; h/2 1];  
25 for i=1:length(x)-1  
26     b=[u(i)+(h/2)*v(i); v(i)+(h/2)*(x(i+1)+x(i)-u(i))];  
27     z=A\b;  
28     u(i+1)=z(1);  
29     v(i+1)=z(2);  
30 end  
31 %toc
```

```
15 v(2)=v(1)+h*(x(1)-u(1));  
16 for i=1:length(x)-2  
17     u(i+2)=u(i+1)+(h/2)*(3*v(i+1)-v(i));  
18     v(i+2)=v(i+1)+(h/2)*(3*(x(i+1)-u(i+1))-(x(i)-u(i)));  
19 end  
20  
21 %toc  
22  
23 % Using implicit Adams_Moulton or trapezoidal rule of second order  
24 A=[1 -h/2; h/2 1];  
25 for i=1:length(x)-1  
26     b=[u(i)+(h/2)*v(i); v(i)+(h/2)*(x(i+1)+x(i)-u(i))];  
27     z=A\b;  
28     u(i+1)=z(1);  
29     v(i+1)=z(2);  
30 end  
31 %toc
```



Now again, this is with Adam Bashforth method and now. So again let me comment on all of them and uncomment the portion which will implement Adam Moulton's yes. So again the total number of discretized points N remains the same and this time Adam Moulton's method of second order. So again, let us wait how much time is taken by Adams Moulton's.

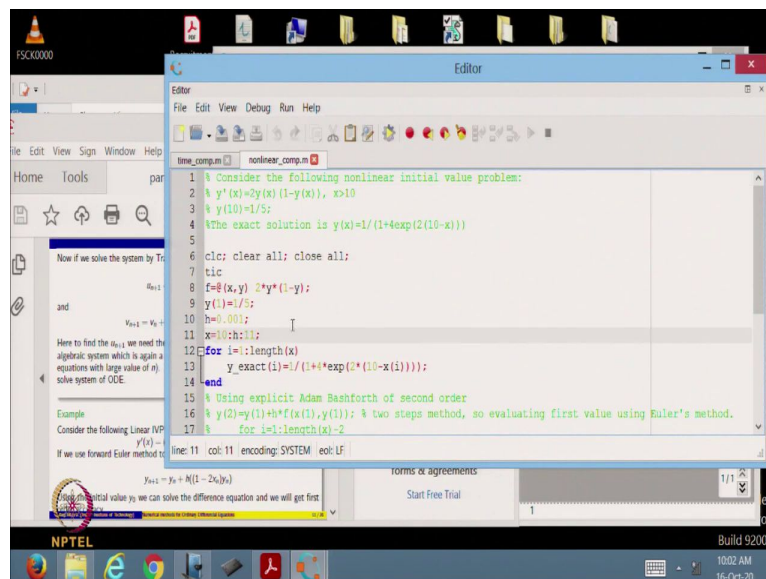
So of course, now you can see the difference; the time takes 1 second difference, even as you will keep increasing the N, as you will keep increasing the N that difference will be more significant. For a small N it is a very marginal difference, but it becomes very significant once you increase the N and in fact, this is a 2 by 2 system if you have to solve a nth order differential

equations in that case, you will end up with this algebraic system of order  $N$  and  $N$  can be more than 2 also in case of  $n$ th order differential equations.

So in that case in fact, the time which we are saving with the help of an explicit scheme is very significant. So in that sense, you can appreciate the beauty of explicit methods because you do not need to solve its algebraic system. So that is what we have learned with the help of the following example. Of course, if you wanted to look at the exact solution of this differential equation that also you can see, because I have already, here I have not written but you can find out the solution of this system also very easily.

And you can plot and you can see, you can also compute the error, like we have done in the case of a linear differential equations, you can play the same thing you will you can also verify the order of convergence, but here my aim was just to tell you that comparison about the timing. So the total time taken by implicit method is higher than total time taken by explicit method in case we are solving a system of differential equations. That way we can appreciate that implicit method is better than explicit method in that sense.

(Refer Slide Time: 15:19)



```
1 % Consider the following nonlinear initial value problem:
2 % y'(x)=2y(x)(1-y(x)), x>10
3 % y(10)=1/5;
4 %The exact solution is y(x)=1/(1+4exp(2(10-x)))
5
6 clc; clear all; close all;
7 tic
8 f=@(x,y) 2*y*(1-y);
9 y(1)=1/5;
10 h=0.001;
11 x=10:h:11;
12 for i=1:length(x)
13     y_exact(i)=1/(1+4*exp(2*(10-x(i))));
14 end
15 % Using explicit Adams Bashforth of second order
16 % y(2)=y(1)+h*f(x(1),y(1)); % two steps method, so evaluating first value using Euler's method.
17 % for i=1:length(x)-2
```



```

13 y_exact(i)=1/(1+exp(2*(10-x(i))));
14 end
15 % Using explicit Adams-Bashforth of second order
16 % y(2)=y(1)+h*f(x(1),y(1)); % two steps method, so evaluating first value using Euler's method.
17 for i=length(x)-2
18     y(i+2)=y(i+1)+h/2*(3*f(x(i+1),y(i+1))-f(x(i),y(i)));
19 end
20 %toc
21 % Using implicit Trapezoidal rule of second order
22 for i=length(x)-1
23     g=@(z) z-y(i)-(h/2)*(f(x(i+1),z)+f(x(i),y(i)));
24     y(i+1)=fsolve(g,0.1);
25 end
26 toc
27

```

$$\begin{bmatrix} 1 & -\frac{h}{2} \\ \frac{h}{2} & 1 \end{bmatrix} = A \quad \text{and} \quad \begin{bmatrix} y_{n+1} \\ y_n \end{bmatrix} = B$$

$$y_{n+1} - y_n = \frac{h}{2} [y_{n+1}' + y_n']$$

$$= \frac{h}{2} [2y_{n+1}(1-y_{n+1}) + 2y_n(1-y_n)]$$

So now the same stuff or we can also do in case of a nonlinear problem. Same stuff I mean to say that we will, here also we will try to solve a nonlinear problem with the help of explicit and implicit methods and in this case we will also conclude that implicit method is taking more time than explicit method. So basically an explicit method is better than an implicit method.

So here the right hand side is  $2y(1-y)$  so, this is the nonlinear differential equations which is you can see from here and we have started initial condition from  $y(10) = 1/5$  because here we have written the domain  $x$  is greater than 10 and the exact solution of course, explicitly also I have

written here  $y(x) = 1/(1+4\exp(2(10-x)))$ . So, in fact I have also implemented this exact solution in this case. So  $y(1) = 1/5$ , again we are working with 0.001,  $x = 10:h:11$ .

So initially we have implemented the exact solution and then this is again a trapezoidal method. So here also you can see how basically in the trapezoidal method we are working because if you remember, the algorithm of the trapezoidal method will be this.

$$y_{n+1} - y_n = \frac{h}{2}(y'_{n+1} + y'_n)$$

$$= \frac{h}{2}(2y_{n+1}(1 - y_{n+1}) + 2y_n(1 - y_n))$$

. So this is contrary to the explicit method where in the right hand side you do not involve any term which contains (n+1)th terms. So you are ending with a nonlinear difference equation, nonlinear difference equations.

(Refer Slide Time: 18:14)

```

13 y_exact(i)=1/(1+4*exp(2*(10-x(i))));
14
15 % Using explicit Adam Bashforth of second order
16 y(2)=y(1)+h*f(x(1),y(1)); % two steps method, so evaluating first value using Euler's method.
17 for i=1:length(x)-2
18     y(i+2)=y(i+1)+h/2*(3*f(x(i+1),y(i+1))-f(x(i),y(i)));
19 end
20
21 %toc
22 % Using implicit Trapezoidal rule of second order
23 for i=1:length(x)-1
24     g=@(z) z-y(i)-(h/2)*(f(x(i+1),z)+f(x(i),y(i)));
25     y(i+1)=fsolve(g,0.1);
26 end
27 toc
28

```

Handwritten note in red:  $+ 2y_n(1-y_n)$

Line 26: col: 8 encoding: SYSTEM eol: LF

NPTEL Build 9200 10:05 AM 16-Oct-20

The screenshot shows the Octave Editor window with a file named `nonlinear_comp.m`. The code defines a function `y_exact` and compares it with numerical solutions using the explicit Adams-Bashforth method and the implicit Trapezoidal rule. The code is as follows:

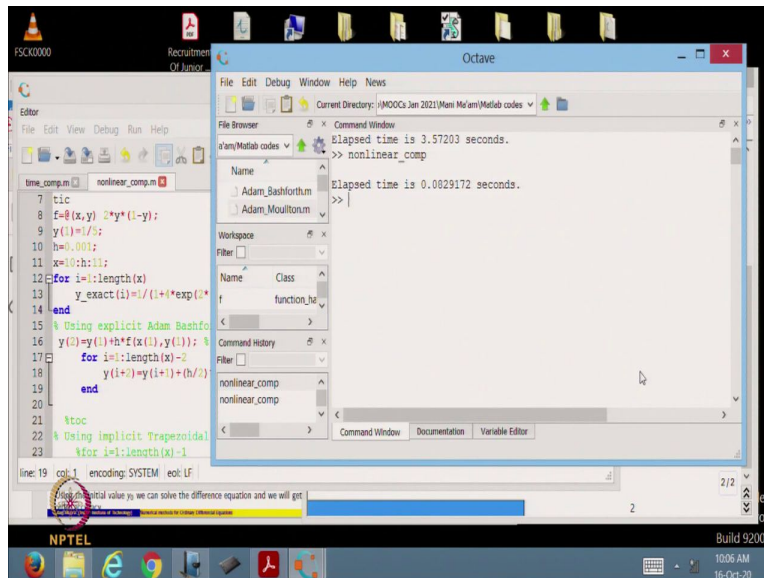
```
13 y_exact(i)=1/(1+exp(2*(10-x(i))));
14
15 % Using explicit Adams-Bashforth of second order
16 y(2)=y(1)+h*f(x(1),y(1)); % two steps method, so evaluating first value using Euler's method.
17 for i=length(x)-2
18     y(i+2)=y(i+1)+(h/2)*(3*f(x(i+1),y(i+1))-f(x(i),y(i)));
19 end
20
21 %toc
22 % Using implicit Trapezoidal rule of second order
23 for i=length(x)-1
24     % y(i+2) = y(i) - (h/2)*(f(x(i+1),y(i+1))+f(x(i),y(i)));
25     % y(i+1)=fsolve(q,0.1);
26 end
27 toc
```

The status bar at the bottom indicates "line: 26 col: 6 encoding: SYSTEM eol: LF". The taskbar at the bottom shows the NPTEL logo and the date "16-Oct-20".

The screenshot shows the Octave Editor window with a file named `nonlinear_comp.m`. The code defines a function `y_exact` and compares it with numerical solutions using the explicit Adams-Bashforth method and the implicit Trapezoidal rule. The code is as follows:

```
7 tic
8 f=@(x,y) 2*y*(1-y);
9 y(1)=1/5;
10 h=0.001;
11 x=0:h:11;
12 for i=length(x)
13     y_exact(i)=1/(1+exp(2*(10-x(i))));
14 end
15 % Using explicit: Adams-Bashforth of second order
16 y(2)=y(1)+h*f(x(1),y(1)); % two steps method, so evaluating first value using Euler's method.
17 for i=length(x)-2
18     y(i+2)=y(i+1)+(h/2)*(3*f(x(i+1),y(i+1))-f(x(i),y(i)));
19 end
20
21 %toc
22 % Using implicit Trapezoidal rule of second order
23 for i=length(x)-1
```

The status bar at the bottom indicates "line: 19 col: 1 encoding: SYSTEM eol: LF". The taskbar at the bottom shows the NPTEL logo and the date "16-Oct-20".

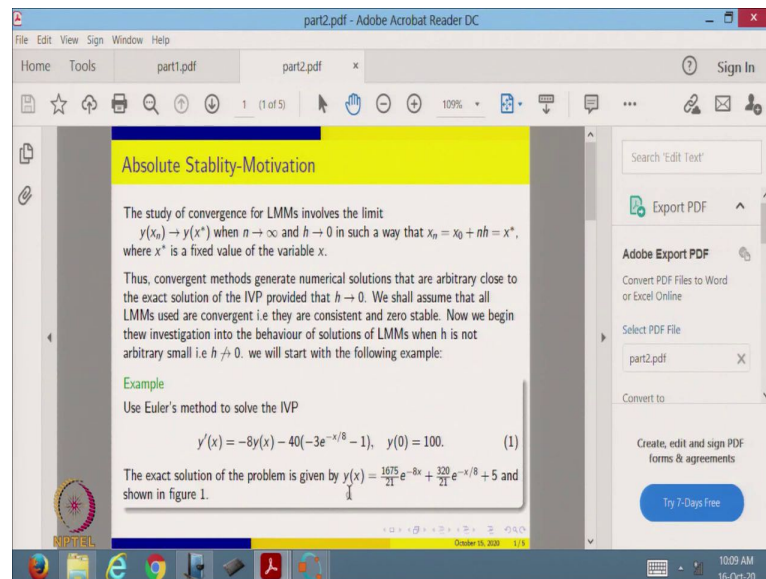


So that nonlinear difference equations we will be solving with the help of inbuilt Matlab function `fsolve`. So of course, whenever you solve any nonlinear equations you have to start with some initial guess and that is the case with Newton-Raphson method also, if you remember from the previous lectures how to solve a nonlinear equation. And basically `fsolve` is an inbuilt function in Octave as well as in MATLAB and implements Newton-Raphson's method.

So I am taking the help of `fsolve` functions initial approximation 0.1 and now, you can see what is the time taken by implicit method in this case, 3 seconds. So again let me comment on this, again I am removing the `clc` command so the previous computer should also be visible to you on the screen. Now let us see how much time is taken by explicit method. So you can yourself appreciate the beauty, the time taken by explicit method is 0.088 seconds and the time taken by implicit method is 3 seconds. And of course, this is the difference in fact, if you are solving only one nonlinear system, one nonlinear system of order 1 or you can say nonlinear equation just 1.

In fact, if you have to solve a system of nonlinear equations by implicit method you can imagine yourself how much time difference there will be. So of course, the explicit method is also better in solving nonlinear equations in terms of computational efficiency. So, so far we all are convinced that explicit methods are better than implicit methods. So it means no one is going to use the implicit method, but it is not the case. There are some disadvantages of explicit methods as well that I am going to cover now.

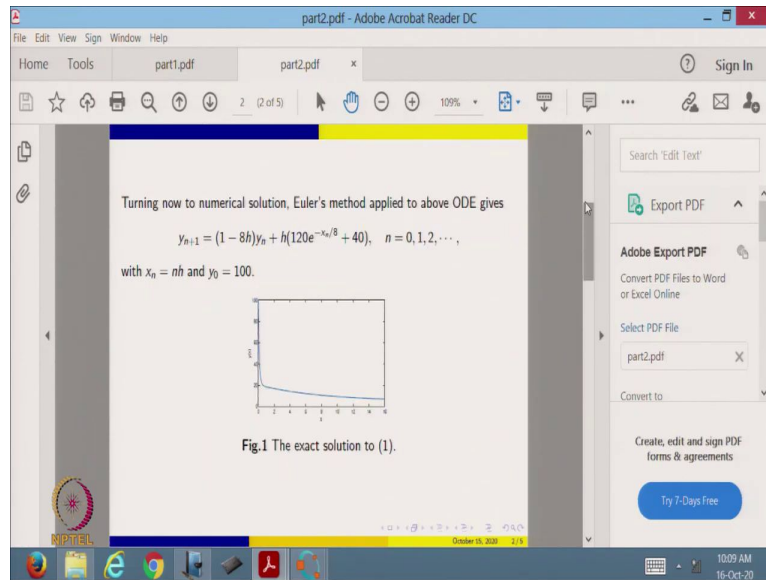
(Refer Slide Time: 21:26)



So here, what we see is the definition of convergence of LMM, LMM stands for Linear Multi-Step Methods. So the study of convergence for LMM involves the following way;  $y(x_n) \rightarrow y(x')$ , when  $n \rightarrow \infty$  and  $h \rightarrow 0$  in such a way that this is the case. Thus the convergence method generates numerical solutions that are arbitrarily close to the exact solution of initial value provided that  $h \rightarrow 0$ .

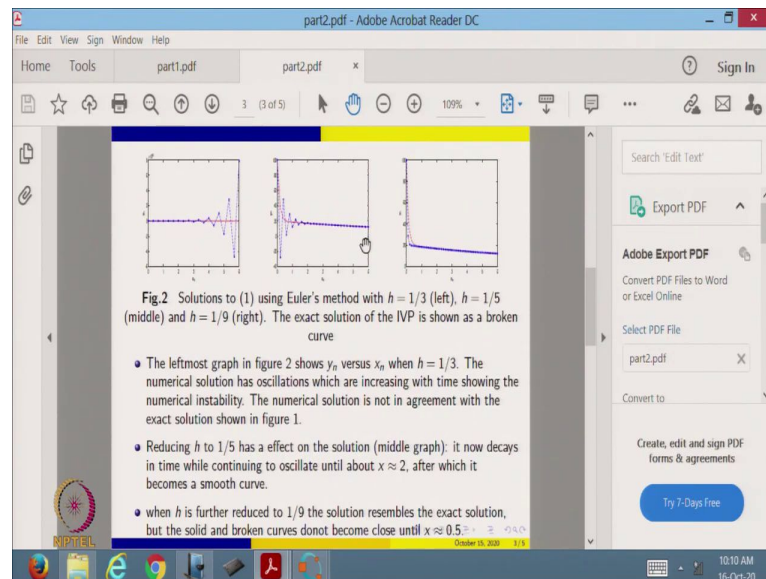
So we shall assume that all Linear Multi-Step Methods which we have seen are convergent, that is they are consistent and zero stable. That is also we have seen in earlier lectures with the help of one theorem, that consistency and zero stability implies convergence. Now we begin the investigations into the behavior of solutions of the Linear Multi-Step Method when  $h$  not arbitrary small. So that is what we will look at with the following example, this is again an example of initial value problem, where initial condition is 100 of course, this is a non homogeneous and the exact solution of the problem is also given by this formula.

(Refer Slide Time: 22:58)



Turning now to the numerical solutions, Euler methods forward Euler, whenever we do not write anything specific in front of Euler method; this is forward Euler. Forward Euler method applied to the above initial value problem will give the following difference equations. So this is how the exact solution of the initial value problem that we have plotted here and I have also implemented these algorithms in Octave, which you will see later on. So of course, the initial condition is 100 that is why  $y(0) = 100$  and forward Euler explicit methods, so we are ending with these difference equations.

(Refer Slide Time: 23:46)



So now, you will see that how for forward Euler behaves for the following example, in case of a different  $h$  because, the leftmost figure represents the solution corresponds to  $h = 1/3$ , in middle we are plotting corresponds to  $h = 1/5$  and in the rightmost figure we are plotting with respect to  $h = 1/9$ .

So the red line shows the exact solution and the blue line shows the numerical solutions. So you can see there is some problem with the forward Euler method when I am choosing  $h = 1/3$  and that problem is not very much severe in case of  $h = 1/5$  and that problem has completely gone in case of  $h = 1/9$ .

So what is the moral of all these three figures? That there is something wrong when I am choosing  $h = 1/3$ . So and as I am choosing a smaller value of  $h$  than  $1/3$ , that problem is not visible. So it means I should work for a very small  $h$ . I should work for a very small  $h$ , that is the conclusion of these three figures. But what that small number should be? What that small number should be? Of course, in this case we have done with the hit and try  $1/3$ ,  $1/5$ ,  $1/9$ .

But otherwise, how will you come to know that for which  $h$  your forward Euler is working fine, for which  $h$  it will not work? So there is a theoretical concept of absolute stability, which you will see in some advanced courses on numerical solutions to initial value problems. Of course, as



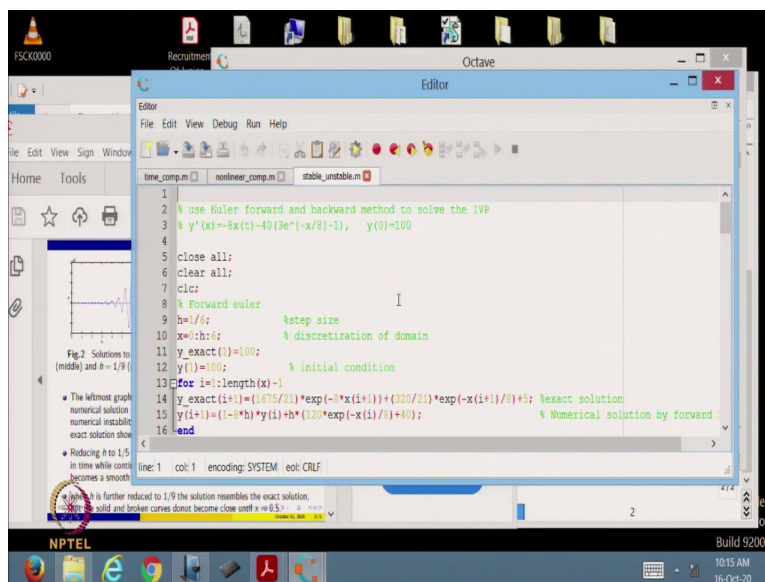
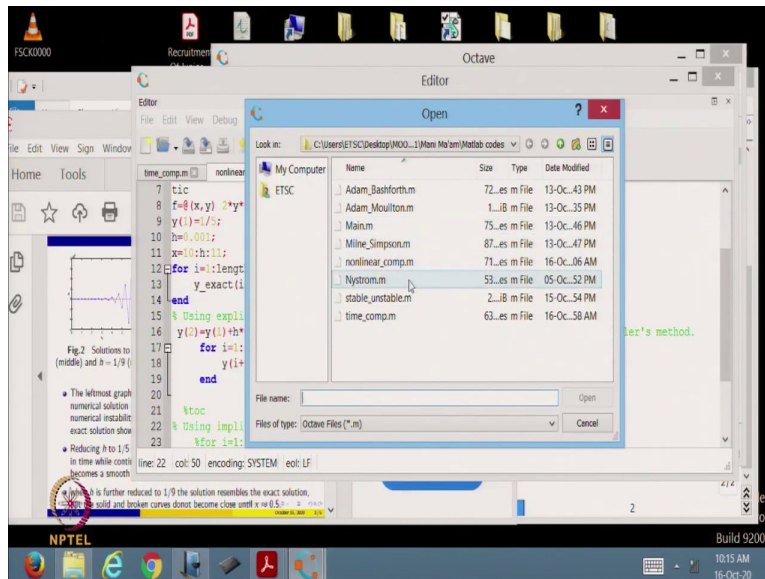
far as this course is concerned, I am just giving you the motivations. I will not cover the entire detail about this absolute stability concept.

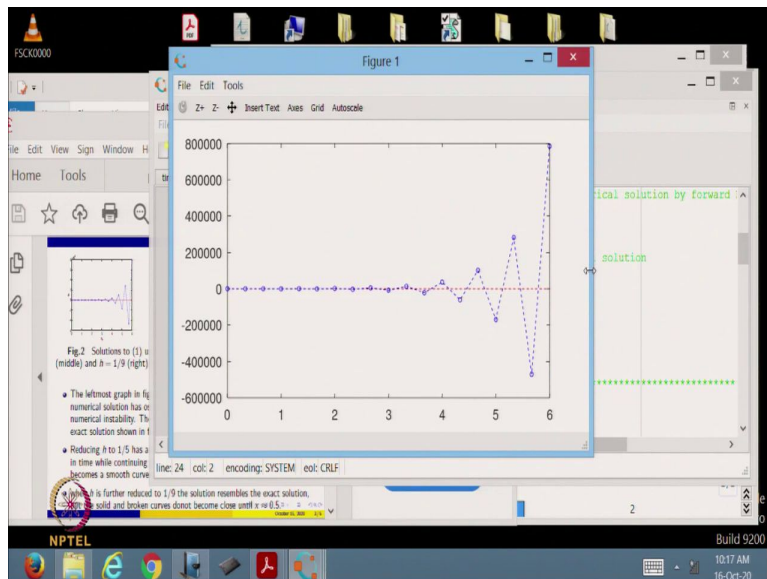
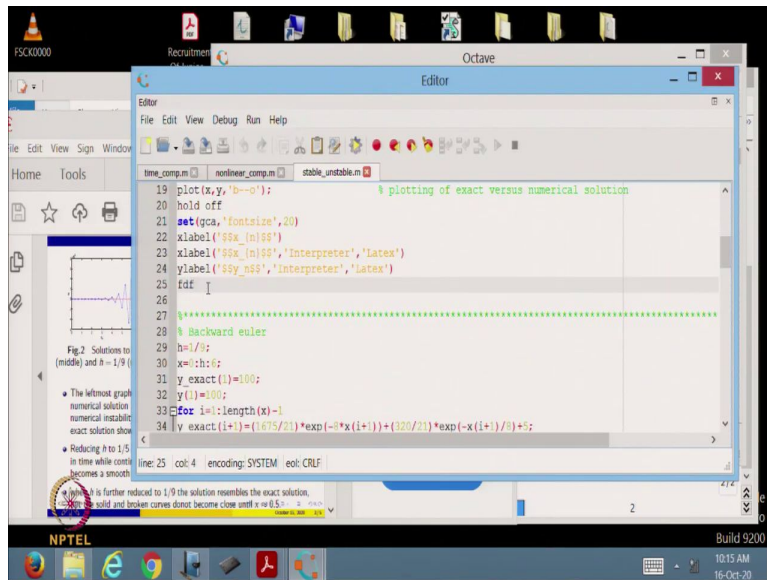
But my aim is to tell you that in some sense, implicit methods are better than explicit methods, because they do not give you any restrictions on  $h$  as the restrictions, which you saw as an explicit method in the following case. So that is the whole idea, because this is the last lecture of this course. So my aim is to tell you one advantage of implicit method as well and that advantage of implicit method is that there is no restriction on  $h$  and how to choose that restriction is a matter of advanced course of numerical solutions to initial value problems, which you may learn in later point of time.

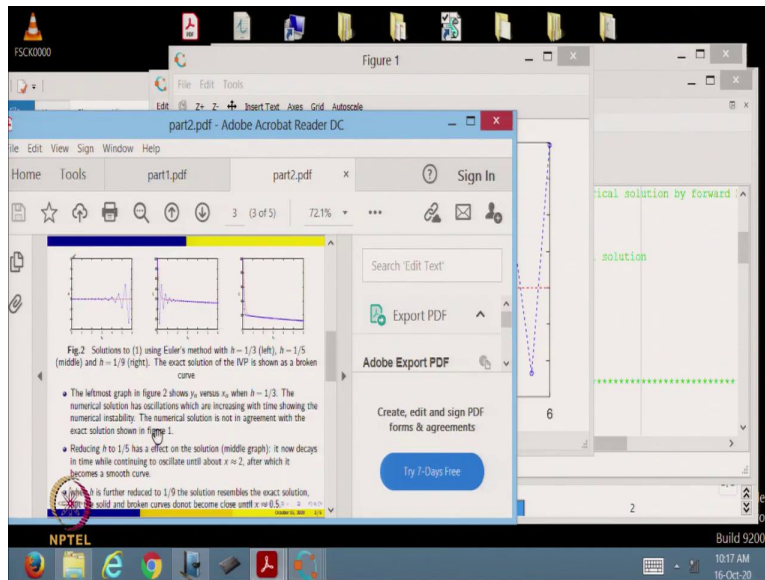
So here again, the same thing, the leftmost graph in figure 2 where  $y_n$  versus  $x_n$  is  $h = 1/3$ , the numerical solution says some oscillation which is visible from the figure as well, which are increasing with time showing the numerical instability of course, this is 0, 1, 2, 3, 4, 6. So in fact, these oscillations grow with the time. The numerical solution is not in agreement, which is very much clear from the figure reducing to  $h = 1/5$  has an effect on the solution.

It now decays in time while continuing to oscillate until about  $x > 2$  and later on it decays that is very much one could observe from the middle figure after which, it becomes a smooth curve. When  $h$  is further reduced to  $1/9$ , the solution resembles the exact solution but the solid and broken curve do not become close until 0.5 which is also visible from the rightmost figure. So in fact if I am, I will work for a smaller value than  $1/9$ , my solution will be closer to the exact solution, that is our intuition and which will be the case, which will be the case.

(Refer Slide Time: 28:41)



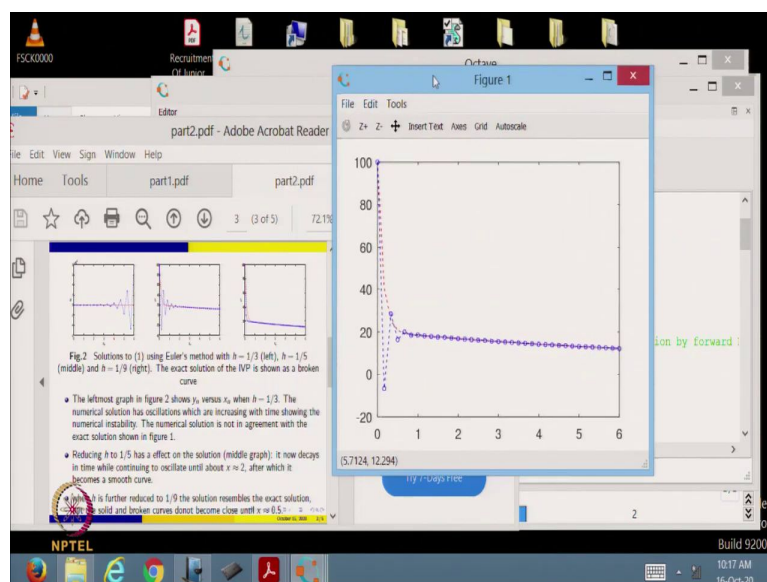
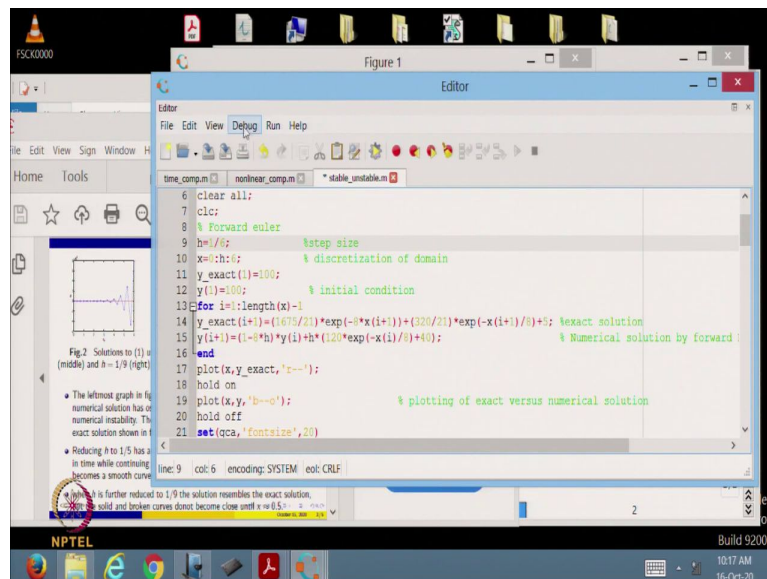




So that also we can see with the help of one code. So here basically, I have tried to implement this code. So again in a comment box, I have written what is the initial value problem we are going to solve and which by which method. So again, I have implemented an initially forward Euler exact solution also, I have implemented and then you will see.

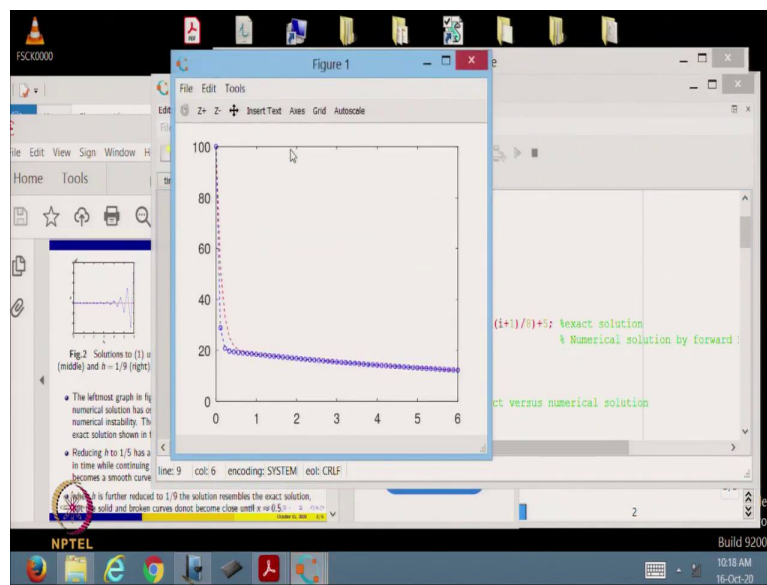
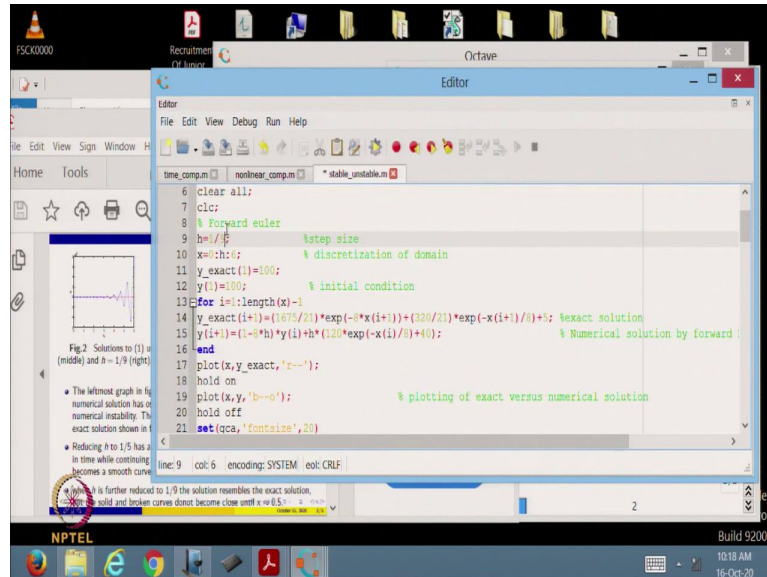
So here just let me I am typing some random thing fdf so that my code stops here, for initially I wanted to see the results of forward Euler  $h = 1/3$ , which is the leftmost figure should come because this corresponds to  $h = 1/3$ , yes. Of course, this is just x label and y label, so that there is some problem with this command in the octave it will run in a similar way with the Matlab. So yes. So you can see some oscillations are there. That is what we are saying. That numerical solution is not in agreement with the exact solution and these oscillations are increasing with respect to time. So just we have implemented a simple forward Euler.

(Refer Slide Time: 31:06)

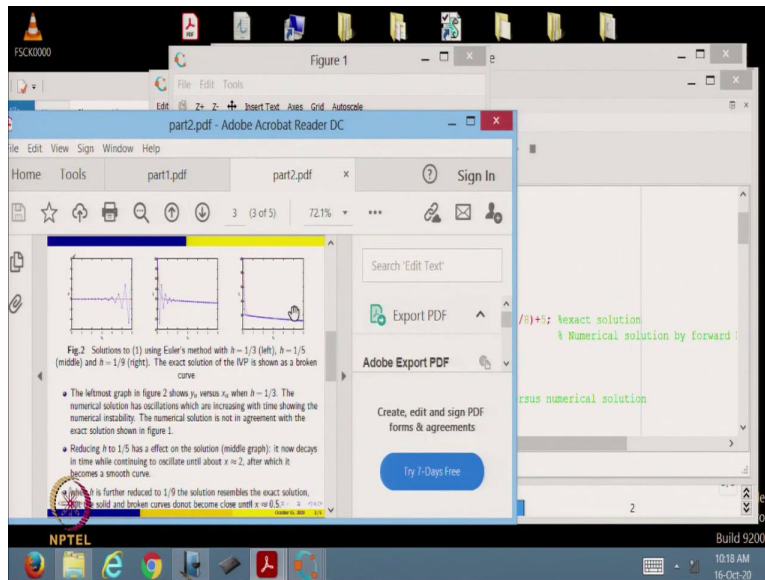


And now if I choose  $h = 1/5$ , now if I run my code, so this is again the same figure, middle one. So of course, oscillations are less in this case, because we have chosen a smaller  $h$ . Of course,  $1/5$  is smaller than  $1/3$ , but still there are some oscillations.

(Refer Slide Time: 31:41)

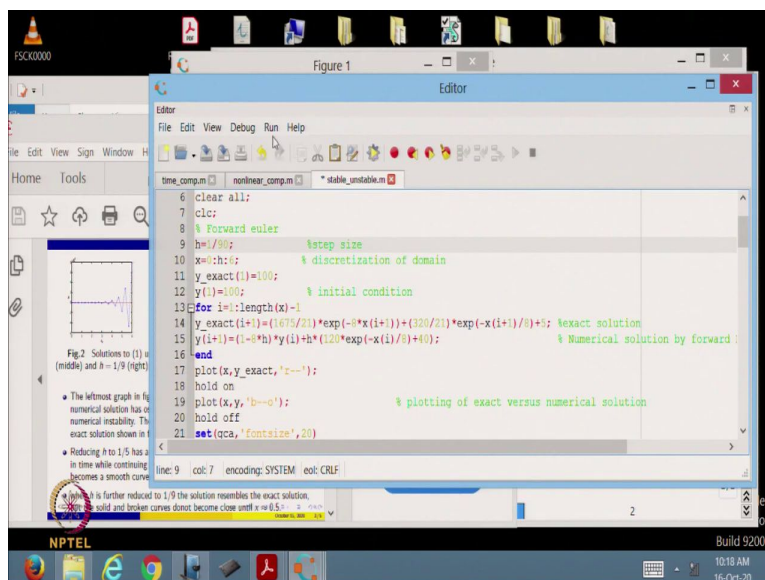




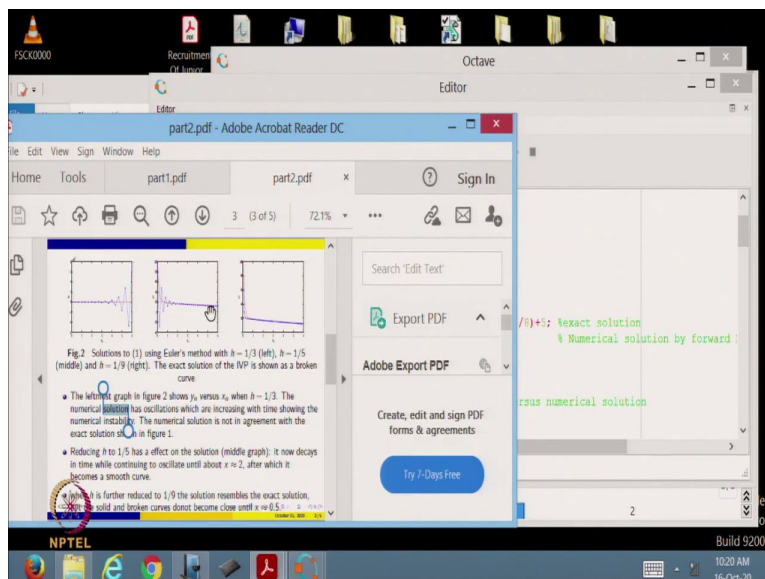
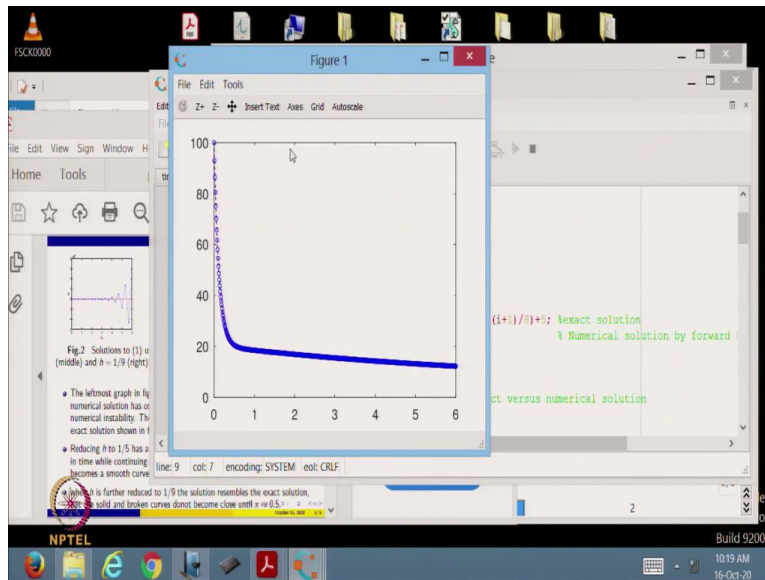


So let me work out with in fact, more a smaller value of  $h$ , let us take  $1/9$ . So yes. So, we thought, that is what we got that rightmost figure. In fact, as we could observe from the figure that we do not see very close agreement with the exact solution. So the blue line indicates the numerical solution and red line indicates the exact solution. So we do not see very close agreement before time 1 or let us say close to 0.5.

(Refer Slide Time: 32:23)





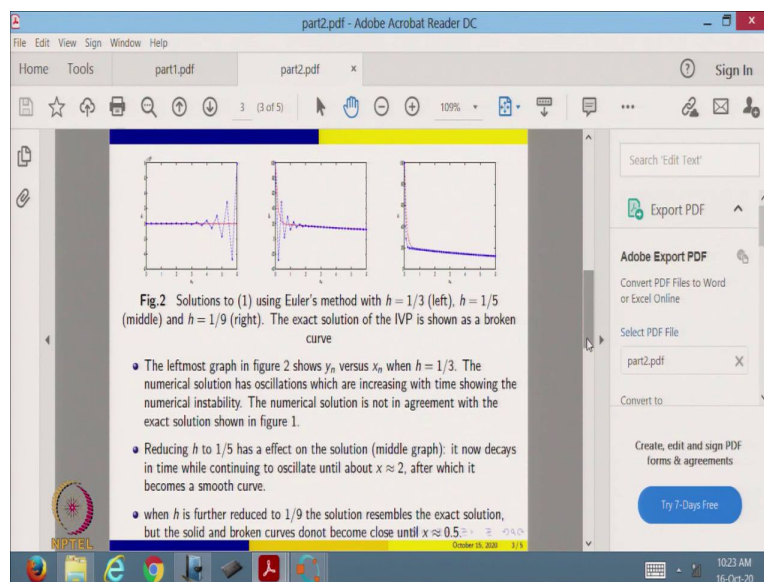
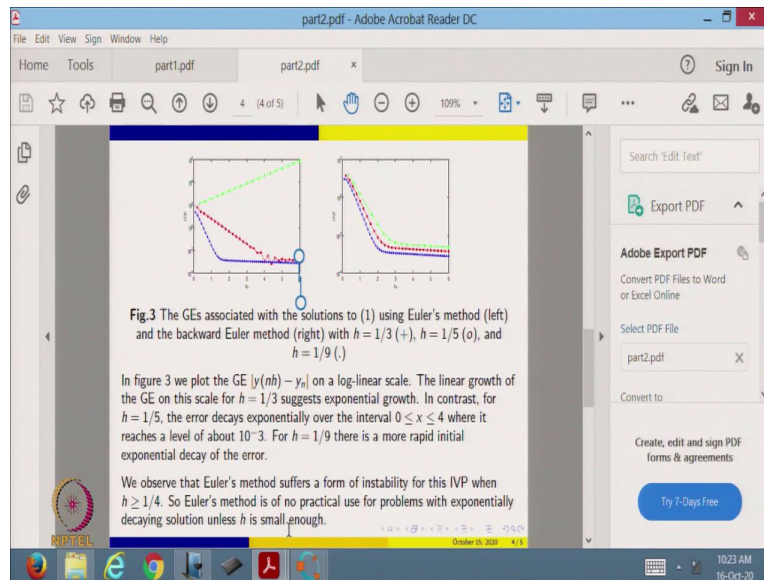


So in fact, that is our intuition that if I really work with a very smallest let, let me arbitrarily choose  $h = 1/90$  then what will be my solution? Yes, so it is not a close agreement. So our intuition has become true that in case of a forward Euler method we have to work for a very small  $h$  for some specific set of initial value problems.

What is that specific set of initial value problem and what is that condition based on what we can choose the  $h$  that is a matter of advanced course of on this topic, which I have already said earlier, but here at least you can get convinced that when you are applying a explicit method for some specific problems, you will have to be extremely careful because you should work out with

a small  $h$ . Because by chance if you work for the  $h = 1/3$  in this case your, you will not get the numerical solutions exactly. So now, that is what I have said about explicit methods.

(Refer Slide Time: 34:06)



Now, let us see how implicit methods work for this example. This is again, if you want, you can also compute the error though I have not implemented those errors formula in the octave code, but by this time you must have learned you can do it yourself. But here, let me show that this is the left, the global error associated with the solution of the initial value problem which we have considered as an example using the Euler method and the backward Euler method. So this is with

the implicit method and this is with the explicit methods. So of course, this green line will correspond to  $h = 1/3$  and then  $h = 1/5$  and then  $h = 1/9$ .

So, in case of  $h = 1/9$  of course error will tend to 0 which we have observed otherwise also. So, in figure 3 basically we plot global error on a log linear scale, the linear growth of global error on this scale for  $h = 1/3$ . So this green line corresponds to  $h = 1/3$  suggests that it is exponential growth which is not acceptable. So in contrast for  $h = 1/5$ , there  $h$  decays exponentially over the interval 0 to 4 where it reaches a level of about  $10^{-3}$  and for  $h = 1/9$  there is a more rapid initial exponential decay of the error.

So,  $h = 1/9$  is acceptable. In fact, I have also shown you the results by choosing  $h = 1/90$  just arbitrarily. You can also plot the error corresponding to that, because the exact solution is in front of you. So what we observe is that Euler methods suffer a form of instability for this initial value problem. Of course, this is also problem dependent, it is not that forward Euler method will when you will choose some  $h$  for a forward Euler method it will not work for all sets of problems.

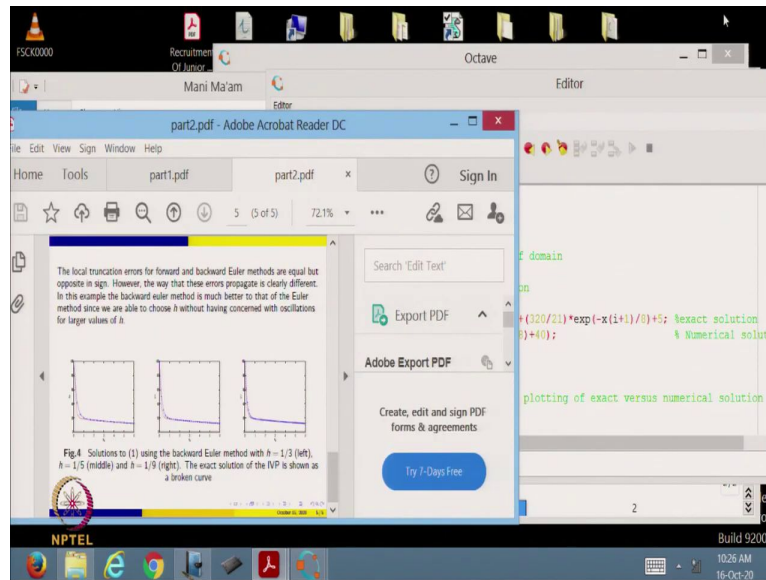
So this is also problem dependent, initial value problem for this particular initial value problem. So. Euler method is of no practical use for problems with exponentially decaying solutions unless  $h$  is small enough. So exponentially decaying because this is the behavior of the exact solution of this initial value problem what we could observe because  $y' = -8y$  term is there.

So because of that if you look at the exact solution of the initial value problem that is in the nature of the exponential; exponentially decaying solution unless  $h$  is small enough. So, it is not that the Euler method will never be used or it is a bad method you have and that is the aim of this part of the course also you should learn which method you should choose for which problem.

Of course, as I also discussed in the previous two examples, when we were solving a system of differential equations as well as more linear initial value problems that implicit methods are taking more time as compared to explicit methods. In this case, what we have seen that explicit method is not performing well when I am choosing large  $h$ .

So, one can question, what is the harm if I choose a very small  $h$ ? What is the harm? The harm is that computation will be more if I choose a small  $h$ ; computation will be more. That is why we should not take excessively small  $h$  as well as computation will be more and once computation will be more your machine error will grow. That also I will explain to you with the help of one diagram.

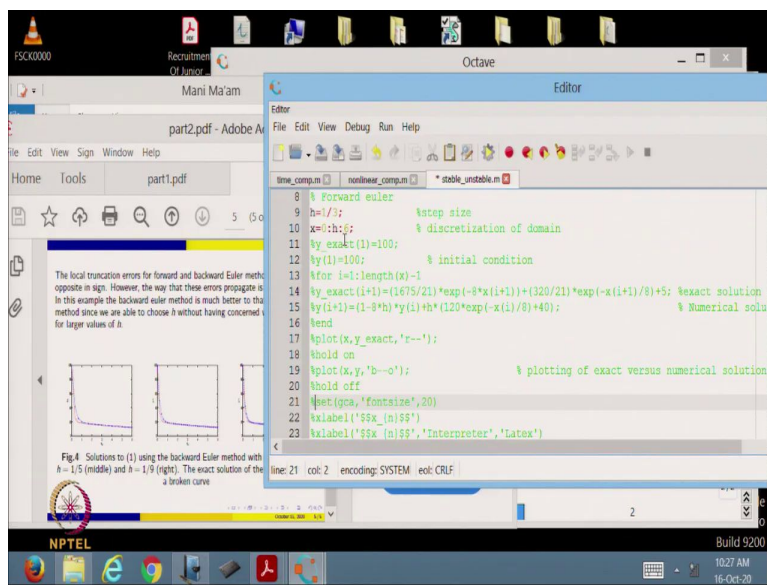
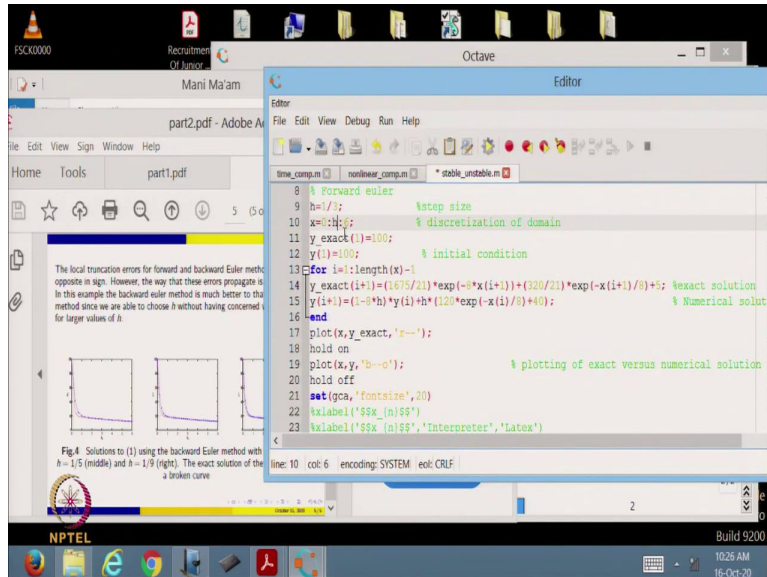
(Refer Slide Time: 38:46)

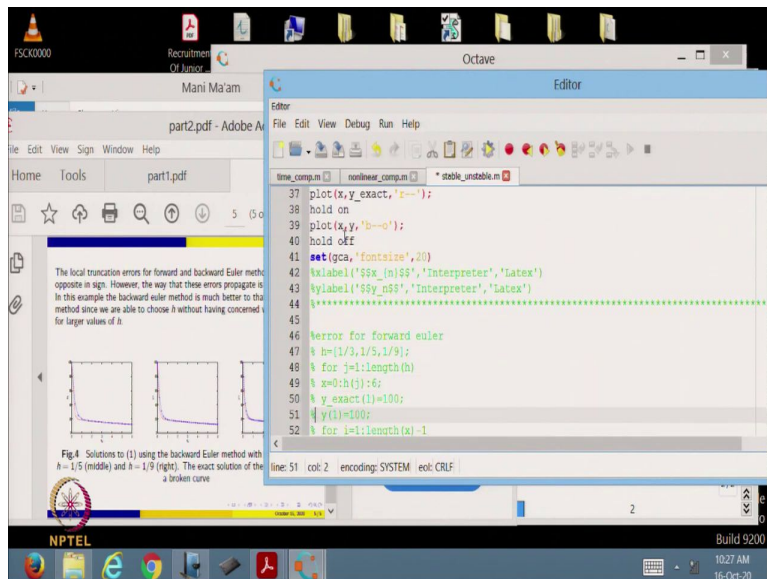
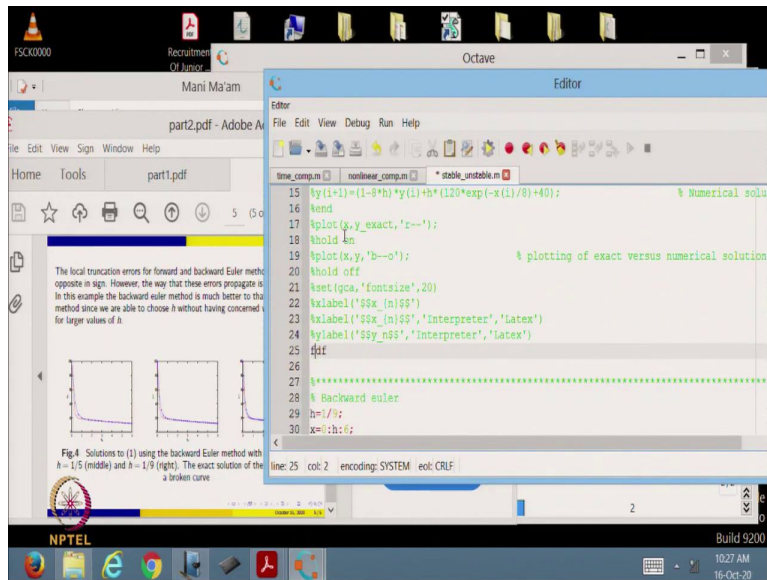


But, this is what I said. Let me plot the solution of the initial value problem as well as the numerical solution of backward Euler method with respect to different  $h$ . So in the left hand side again  $h = 1/3$ , in the middle  $h = 1/5$ , in the rightmost  $h = 1/9$  using backward Euler. So, the local truncation error for forward and backward Euler methods are equal, but opposite in sign.

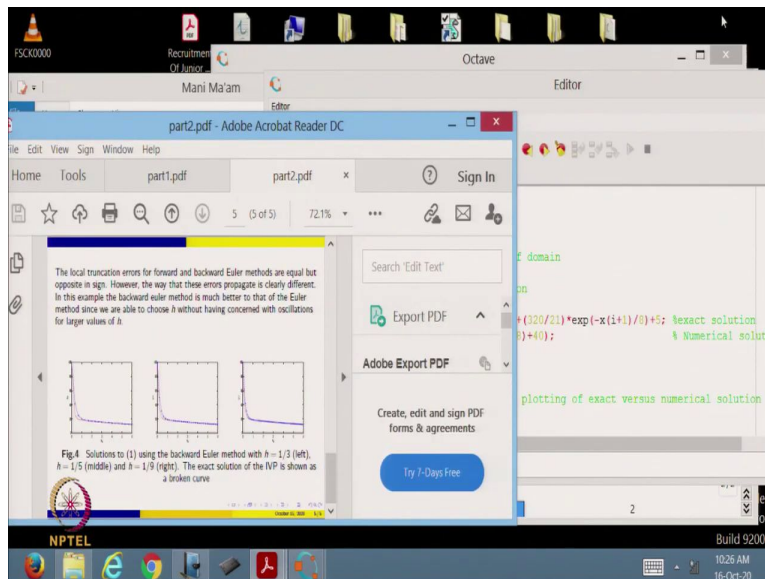
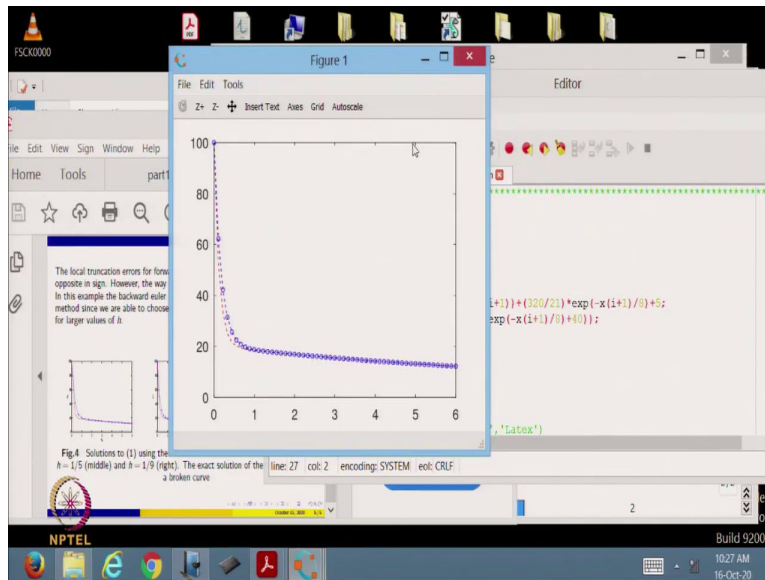
However, the way that these errors propagate is clearly different from what we have already observed from the previous figures. So in the following example, the backward Euler method is much better than that of the Euler method since we are able to choose  $h$  without having to be concerned with oscillations for larger values of  $h$  that is what is the moral of all discussion.

(Refer Slide Time: 39:43)









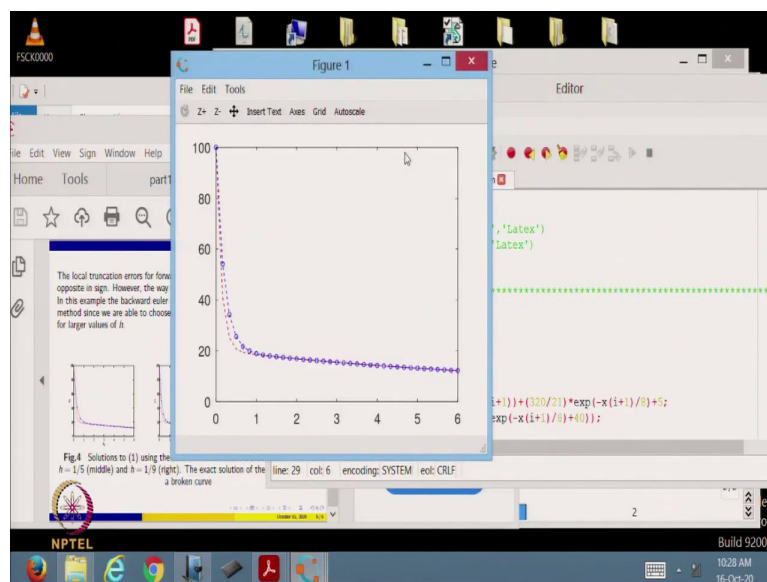
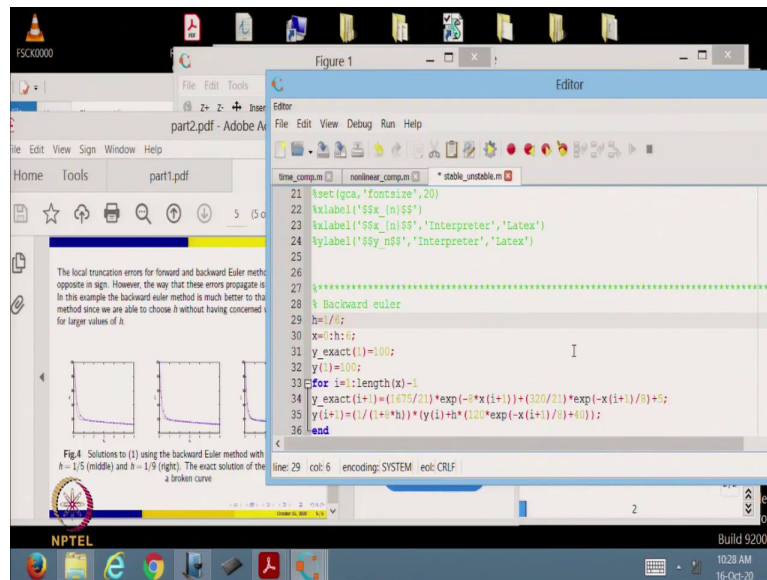
And here, the remaining task is that so far I have not shown you the analysis of backward Euler in Octave. So again let me choose  $h = 1/3$  and comment all the lines which implement the exact solution as well as solution with forward Euler. So again let me remove this arbitrary fdf which I have just added to terminate the code and these commands also because they are not working in a proper way in Octave.

And in fact, so you implement backward Euler algorithms and you will see how the solutions look like in case of  $h = 1/3$  which is basically the leftmost figure which you see in the slides. So now, let me change  $h$ , here I have chosen  $h = 1/9$  because initially I thought I have changed to



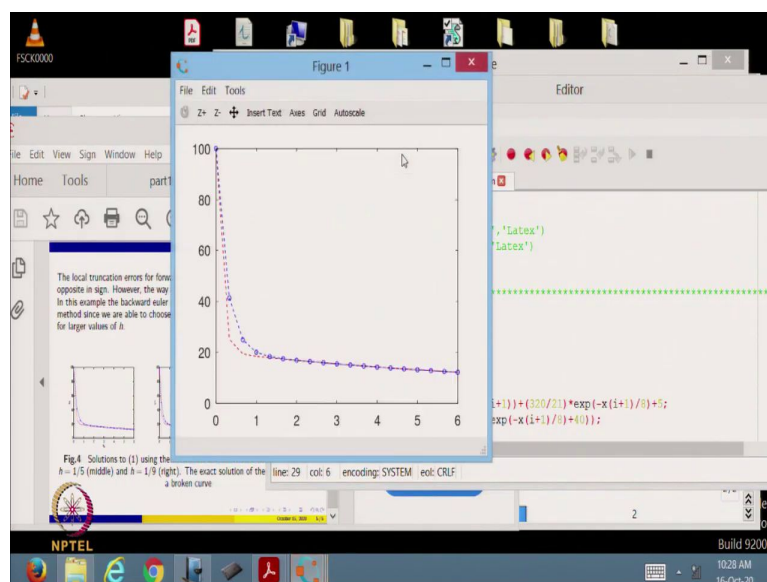
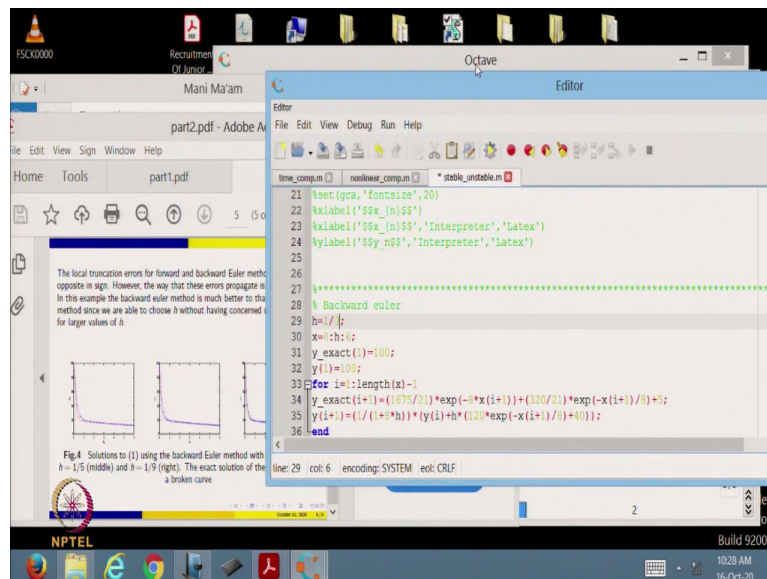
$1/3$ , no but further it has been changed to  $h = 1/9$ , so this corresponds to  $1/9$ . So in that case it is the rightmost figure.

(Refer Slide Time: 41:47)



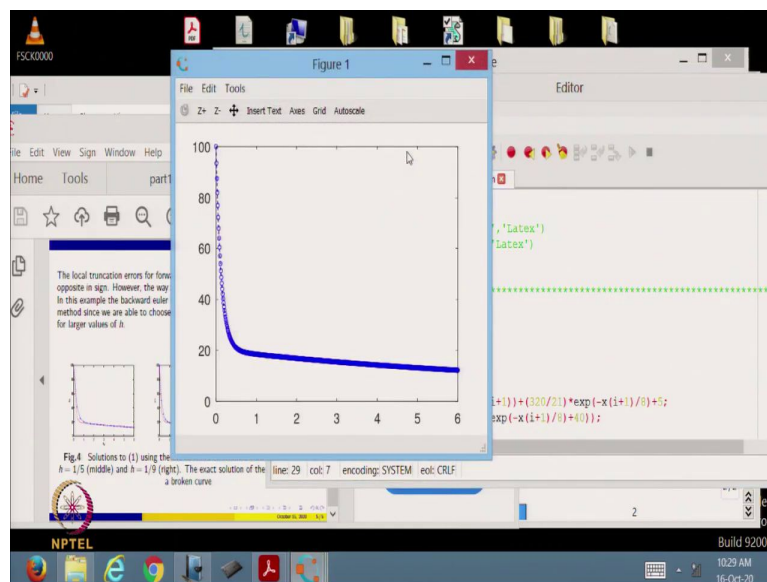
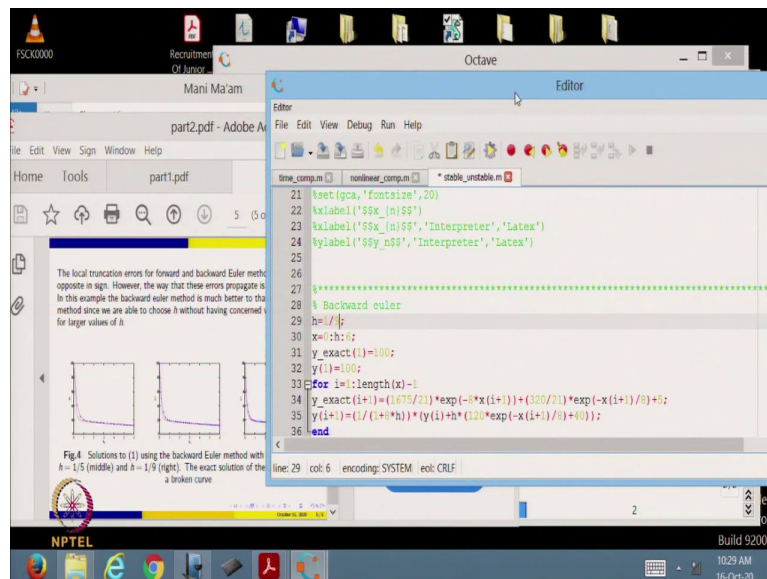
Now let me change to  $h = 1/5$ . So in this case it should, we should end up with the middle figure. That is what we can see.

(Refer Slide Time: 42:00)



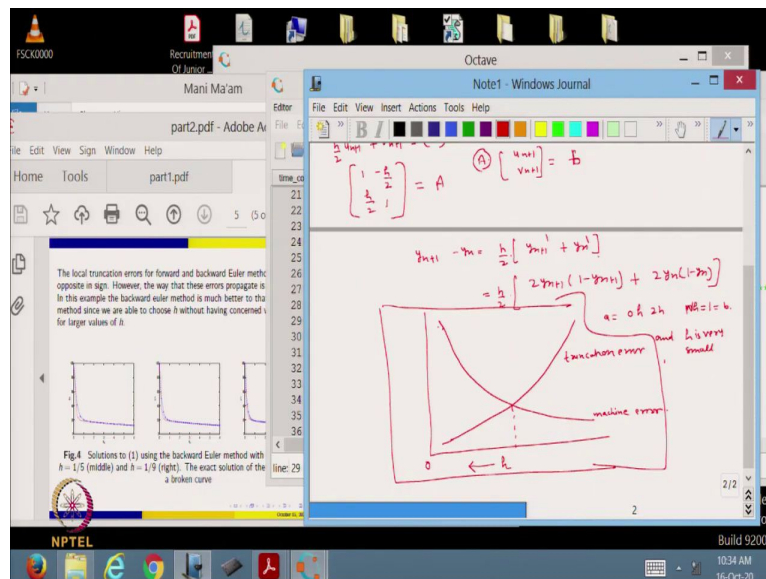
And here if I change it to  $h = 1/3$ , it will be the leftmost figure where disagreement between the exact solution and numerical solution is more visible. And that is because of, because really the method is convergent when  $h \rightarrow 0$ . But it is not the case that error is growing. There will be some disagreement here also but it is not that solution is nowhere close to the exact solution. So this is with respect to backward Euler.

(Refer Slide Time: 42:48)



In fact, if in this case also let me work out with very small  $h$  arbitrarily, so again you will get a very good agreement. That is what you can observe from the following figure.

(Refer Slide Time: 43:16)



So, now as I told you I will explain to you why it is not a good idea to work with a very small  $h$  because let me draw a figure here,  $h$ . So here  $h \rightarrow 0$ . So you know, when you are working for a very small  $h$ , you have to apply more operations or you have to reach a 1 in a more operations because from 0 to  $h$ ,  $h$  to  $2h$  and then  $nh$  which is equal to 1.

That is how we have chosen our grid size. This is  $N$ . So if  $N$  is very large, and  $h$  is very small because  $Nh = 1$ , so  $A$  is 0, this is  $b$  you can say,  $h$  is very small. You have to do more operations. So of course machine error will be more. So machine error will behave like that. So this corresponds to machine error and when and what is the behavior of the truncation error?

Behavior of truncation error is that truncation error tends to 0 as  $h$  tends to 0. So this is a truncation error. So every numerical method is equipped with these two errors; truncation error, machine error. So if we are working for a very small  $h$ , machine error is large, truncation error is small but the total error is the sum of these two. So of course we should work for a value of  $h$  for which is somehow balanced between the truncation error and the machine error because if we work for a large  $h$  truncation error will be more. If we work for a small  $h$ , machine error will be more.

So we have to choose  $h$  in a optimal way and in this case the optimal  $h$  is here where the sum of these two error is minimum. Where the, so basically the conclusion is that we should not work

either with very large  $h$  or with very small  $h$ . we should try to make a balance so that the sum of machine error and truncation error is small and in fact if you say that there is no machine error then of course everyone will choose for a very small  $h$  because only truncation error is a which we are getting in numerical method but any computer machine is not free with machine error.

One way of reducing the machine error is that you should always work on higher precision machines which will give you the most exact computations. If you have any arithmetic so which you must have already, what are machine errors, how it affects when I grow with the number of steps or number of operations that you must have already studied is some basic course of numerical analysis. So with this diagram which is a crux of any numerical methods; in fact, when you apply to initial value problems and when you apply to boundary value problems as well.

Because in boundary value problems also you have to discretize the derivative in the same way and then you end up with a system of algebraic equations. But in boundary value problems also this is the way how truncation error and machine error behaves. In fact, when you apply any numerical methods to PD also the behavior of error, so this is a, basically this diagram which we have seen, you should understand very carefully which is a crux of any numerical method because any numerical method is associated with these two kinds of error; truncation error and machine error.

So with these words let me stop with my discussion and I hope all of you must have enjoyed this course and this, also this part of the course gives you the motivation why you should study some advance course as well on numerical methods and because numerical methods play a very important role for solving a real life problems where exact solution is not available. Thank you very much.