Numerical Analysis
Prof. S. Baskar
Department of Mathematics
Indian Institute of Technology - Bombay

Lecture - 60
Numerical ODEs Stability Analysis

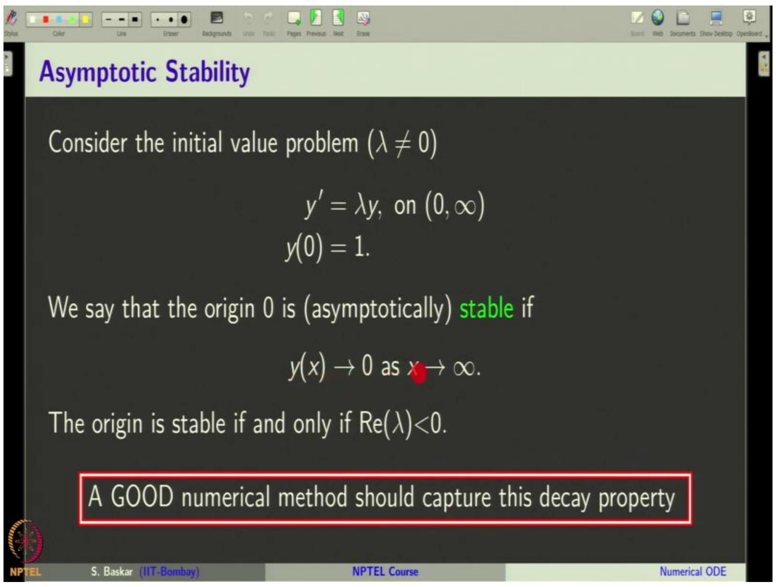
Hi, we are learning numerical methods for initial value problems of first order ordinary differential equations. In this we have learned some explicit methods as well as implicit methods. Explicit methods are easy to implement and they are also efficient in computation because their formulas are written explicitly for the unknowns in terms of all the known quantities. Whereas, implicit methods in general gives us a relation which may be a non-linear equation, if our given ODE is a non-linear ODE.

In which case, you may have to go for non-linear iterative method, in order to get the approximate solution or you may have to go for a predictor corrector approach. This is what we have seen in the last few lectures. You may be wondering when we have explicit methods, why are we struggling with the implicit method because we need to put more efforts in order to get the approximate solution through implicit methods.

Well, there is a purpose for implicit methods. This purpose will be very clear if we understand the stability analysis behind these methods. In this lecture, we will try to understand the stability of these methods. In order to keep our discussion very simple, we will do the stability analysis for only forward Euler method and the trapezoidal method. Recall, forward Euler method is an explicit method and trapezoidal method is an implicit method.

In order to see the importance of the implicit method, we will just do the stability analysis for these two methods. One can also extend the stability analysis for other methods, like Runge-Kutta methods and other multi-step methods that we introduced in our previous lectures. But because of the time constraints and also to keep our discussion simple, we will only cover the stability analysis for these two methods.

(Refer Slide Time: 02:47)



Asymptotic Stability

Consider the initial value problem ($\lambda \neq 0$)

$$y' = \lambda y, \text{ on } (0,\infty)$$
$$y(0) = 1.$$

We say that the origin 0 is (asymptotically) stable if

$$y(x) \to 0 \text{ as } x \to \infty.$$

The origin is stable if and only if Re(λ)<0.

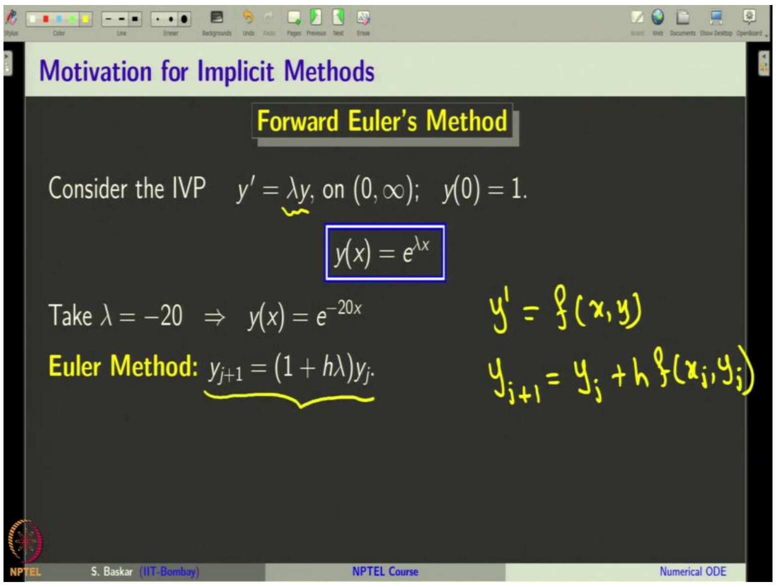
A GOOD numerical method should capture this decay property

S. Baskar (IIT-Bombay) NPTEL Course Numerical ODE

For stability analysis, we will only consider the linear initial value problem of the form $y' = \lambda y$, posed on the interval $(0, \infty)$, with this particular initial condition $y(0) = 1$. We say that the origin 0 is asymptotically stable if $y(x)$, that is the exact solution of this initial value problem, tends to 0 as x tends to ∞. Remember in our initial value problem, we will always take λ to be non-zero. It may be a complex number or just a real number.

You can see that the origin is stable, if and only if, the real part of the λ, that we have on the right hand side of our equation, is strictly less than zero. A good numerical method should capture this decay property of the exact solution.

(Refer Slide Time: 03:58)



Motivation for Implicit Methods

Forward Euler's Method

Consider the IVP $y' = \lambda y$, on $(0,\infty)$; $y(0) = 1$.

$$\boxed{y(x) = e^{\lambda x}}$$

Take $\lambda = -20 \Rightarrow y(x) = e^{-20x}$

Euler Method: $\underbrace{y_{j+1} = (1 + h\lambda)y_j}$.

$y' = f(x,y)$

$y_{j+1} = y_j + h f(x_j, y_j)$

S. Baskar (IIT-Bombay) NPTEL Course Numerical ODE

Now, let us go to the forward Euler method and we will try to apply the forward Euler method to our initial value problem $y' = \lambda y$ with initial condition as $y(0) = 1$ and see whether forward

Euler method captures the decay property of this initial value problem when $\lambda < 0$. Let us keep in mind that the exact solution of this initial value problem is given by $y(x) = e^{\lambda x}$. Since we want to study the stability, we will always take λ to be negative.

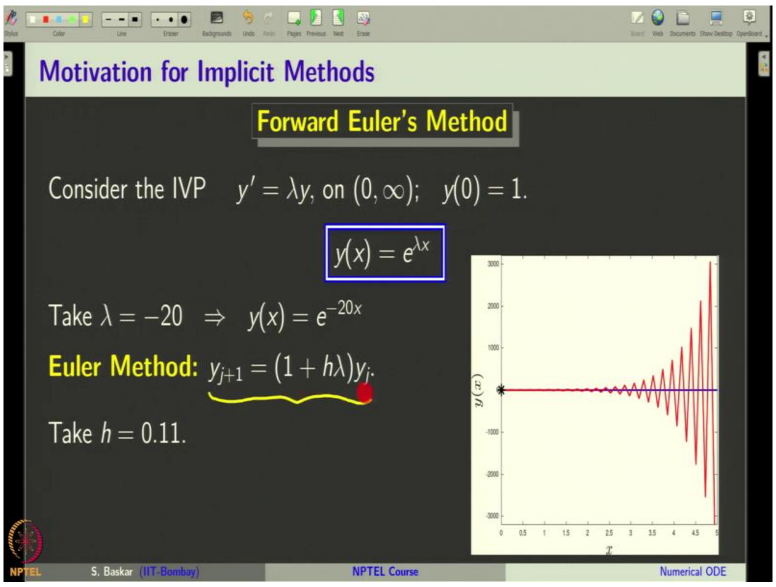
To be more precise, let us take $\lambda = -20$. In which case the solution $y(x)$ is given by e^{-20x}. The graph of the solution is shown in the blue solid line. The star here indicates the point at which we have specified the initial condition. You can see that as *x* increases, the solution drops down rapidly and tends to 0 as x tends to ∞. That shows that for $\lambda = -20$, you have the asymptotic stability for the origin.

Of course, we have seen that for $\lambda < 0$, for any value, you have the asymptotic stability because the solution has the exponential decay nature. Here you can see that the solution drops down from 1 to 0 more rapidly within the interval 0 to almost say 0.3. Therefore, our numerical method should capture this rapid change in the solution. This is one of the aspects of the so-called stiff problems.

In stiff problems, the solution undergoes a rapid change in a small interval of the independent variable and therefore this particular initial value problem exhibits one aspect of the stiff problem. In general, explicit methods struggle to capture the solution when the problem is stiff. Let us see how forward Euler method captures this solution. Recall the forward Euler method for this initial value problem is given by this formula.

If you recall, if you have $y' = f(x, y)$ then the forward Euler method is given by $y_{j+1} = y_j + hf(x_j, y_j)$. Here, we have $f(x, y) = \lambda y$, therefore the forward Euler formula is given by this expression.

(Refer Slide Time: 07:28)



Motivation for Implicit Methods

Forward Euler's Method

Consider the IVP $y' = \lambda y$, on $(0, \infty)$; $y(0) = 1$.

$$\boxed{y(x) = e^{\lambda x}}$$

Take $\lambda = -20 \Rightarrow y(x) = e^{-20x}$

Euler Method: $\underbrace{y_{j+1} = (1 + h\lambda)y_j}$.

Take $h = 0.11$.

S. Baskar (IIT-Bombay) NPTEL Course Numerical ODE

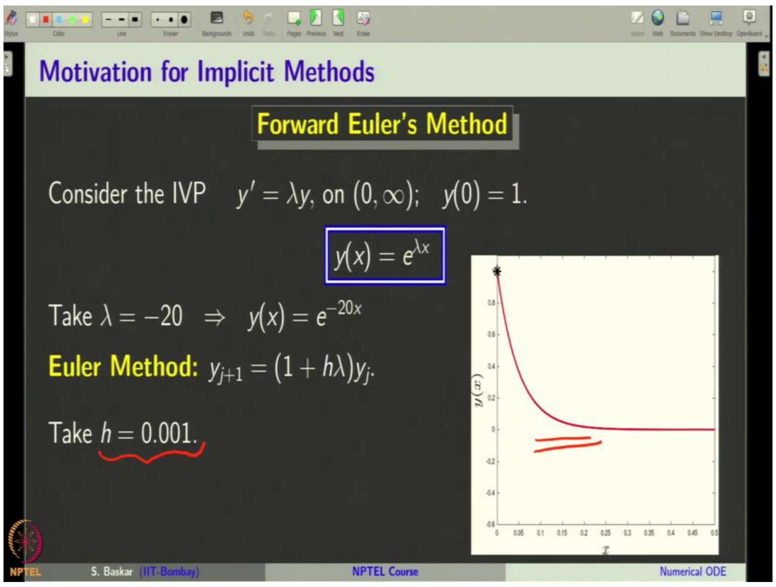
Let us take $h = 0.11$ and see whether the Euler solution captures this decay property well or not. The approximate solution computed from the Euler method is shown in the red solid line. Here blue solid line is the exact solution. Recall, the exact solution starts from 1 and it falls to zero quite rapidly.

However, since the y coordinate is shown with the scaling ranging from -3000 to 3000, therefore in this y scaling the exact solution is seen almost like a straight line just parallel to the x axis. Why it is happening, because the Euler solution is furiously oscillating around $y = 0$ and in fact as it oscillates, the amplitude also blows up to ∞ as $x \to \infty$. You can see that this is no way near to the exact solution because exact solution starts from 1 at $x = 0$ and decays quite rapidly to zero.

Whereas, the Euler solution started correctly from $y = 1$ at $x = 0$ but then it started oscillating and the amplitude of the oscillation keeps on increasing and as $x \to \infty$, amplitude tends to ∞ as it oscillates. It is quite surprising why Euler method gives such a drastic behaviour and such a behaviour is what we refer to as instability in the numerical method.

When the exact solution is bounded, we at least expect the numerical solution to be also bounded. Whereas in this case, the exact solution is nicely decaying to zero but the approximate solution computed from the forward Euler method is blowing up.

(Refer Slide Time: 09:51)



Motivation for Implicit Methods

Forward Euler's Method

Consider the IVP $y' = \lambda y$, on $(0,\infty)$; $y(0) = 1$.

$\boxed{y(x) = e^{\lambda x}}$

Take $\lambda = -20 \Rightarrow y(x) = e^{-20x}$

Euler Method: $y_{j+1} = (1 + h\lambda)y_j$.

Take $\underbrace{h = 0.001}$.

Let us try to understand, why such an instability is observed in the forward Euler method. Let us go back to the Euler method formula. You have $y_{j+1} = (1 + h\lambda)y_j$ and in this particular initial value problem, it can be written as $y_{j+1} = (1 + h\lambda)^{j+1}$. How is this happening? It is very clear, you take y_1, it is given by $(1 + h\lambda)y_0$ but $y_0 = 1$ therefore it is $1 + h\lambda$.

Now y_2 is equal to $(1 + h\lambda)y_1$, which is equal to $(1 + h\lambda)$ into, $y_1 = 1 + h\lambda$, therefore if you multiply y_1 with $1 + h\lambda$, you get $y_2 = (1 + h\lambda)^2$. Similarly, you have $y_3 = (1 + h\lambda)^3$ and so on. So, in general $y_{j+1} = (1 + h\lambda)^{j+1}$. We have already chosen λ to be negative. Now, since we are with forward Euler method and we want the solution on the right side of the initial point x_0, which is equal to 0, we have to take h to be positive.

You have λ negative and h positive. Now with this the formula for Euler method, which is given like this, will show up the decay property only when $|1 + h\lambda| < 1$. Because you have $(1 + h\lambda)^{j+1}$, as j tends to ∞, which is equivalent to saying that $x \to \infty$, y_{j+1} will tend to 0 if and only if $|1 + h\lambda| < 1$.

So, that is very clear. Now you want this condition in order to capture the decay property which is there in your exact solution. Let us see what choice of your h will give this condition. Of course, from this inequality you can immediately see that you cannot achieve this inequality for any choice of $h > 0$ but you have to choose $h < -\frac{2}{\lambda}$. Remember, λ is negative therefore $-\frac{2}{\lambda}$ will be some positive number.

So, you have to choose *h*, which is of course greater than 0, but it should be less than this number. Now when you take $\lambda = -20$, you can see that this is equal to 0.1. Therefore, this small stability analysis says that your forward Euler method will give you the decay property which is there in your exact solution, if and only if, you choose your h to be some number which is less than 0.1.

Of course, it should be greater than 0 also. Whereas in the previous numerical experiment, we have chosen $h > 0.1$. Thereby we had $|1 + h\lambda| > 1$, that is why when you raised that to the power of *j* and as you go on increasing *j*, your *y* tends to ∞. That is what, we can clearly see in the numerical solution here.

Therefore, in order to have stability in the forward Euler method for this particular initial value problem, you have to choose your $h < -\frac{2}{\lambda}$. Let us choose $h < -\frac{2}{\lambda}$, which is in our particular choice of λ happens to be 0.1. We will now choose $h < 0.1$ and let us see how the numerical solution looks like. Well, one good improvement is that the numerical solution is not oscillating wildly and tending to infinity. It is nicely decaying now.

So, that one improvement is there in our numerical solution. However, you can see that for small values of *x* it shows some spurious oscillation which is not there in your exact solution. This is purely a numerical noise, which you do not want to have in the numerical solution but Euler method is exhibiting it. Well, if you decrease the value of *h*, this oscillation will tend to decrease. Let us take *h* = 0.025.

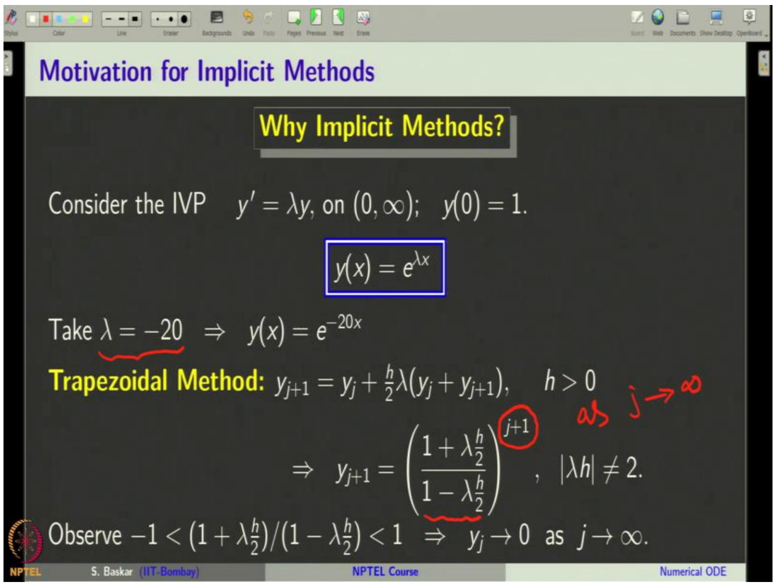
You can see at least at the points where we plotted the graph of the approximate solution, we do not observe any oscillation for this value of *h*. However, you can see that at some points the accuracy is not so satisfactory. You still have a big error, that may be improved by taking smaller values of h. Now, I am taking h to be 0.01. You can see that the approximate solution is now better coinciding with the exact solution, when compared to the approximate solution obtained for h=0.025.

Let us further decrease the value of h and see at least for $h = 0.001$. You can see that the red solid line is coinciding more well with the blue solid line and you can see, at least visually, that the accuracy is now better. So, from here you can see that to get a satisfactory accuracy like this, you have to really choose your h very very small in the forward Euler method. Remember, forward Euler method is an explicit method and it is a first order method.

You can speed up the convergence by going for higher order methods, like Runge-Kutta method and all. In fact, you can go for multi-step methods. In that way you not only gain higher order, you also include more points into your formula. In that way, when you are working with stiff problems, multi-step methods work very well. Because it takes the information from more points and build the approximate solutions at the unknown points.

In that way, multi-step methods are more suitable for such stiff problems and for stability purpose. Explicit methods always come with some extra conditions on h in order to achieve the stability. This is what generally people call as conditionally stable methods. Like Euler method, many other explicit methods are only conditionally stable and also they demand a very small value of h in order to achieve the stability.

(Refer Slide Time: 18:34)



Motivation for Implicit Methods

Why Implicit Methods?

Consider the IVP $y' = \lambda y$, on $(0, \infty)$; $y(0) = 1$.

$$\boxed{y(x) = e^{\lambda x}}$$

Take $\underbrace{\lambda = -20}$ $\Rightarrow$ $y(x) = e^{-20x}$

Trapezoidal Method: $y_{j+1} = y_j + \frac{h}{2}\lambda(y_j + y_{j+1})$, $h > 0$ as $j \to \infty$

$$\Rightarrow y_{j+1} = \left(\frac{1 + \lambda\frac{h}{2}}{\underbrace{1 - \lambda\frac{h}{2}}}\right)^{\boxed{j+1}}, \quad |\lambda h| \neq 2.$$

Observe $-1 < (1 + \lambda\frac{h}{2})/(1 - \lambda\frac{h}{2}) < 1 \Rightarrow y_j \to 0$ as $j \to \infty$.

S. Baskar (IIT-Bombay) NPTEL Course Numerical ODE

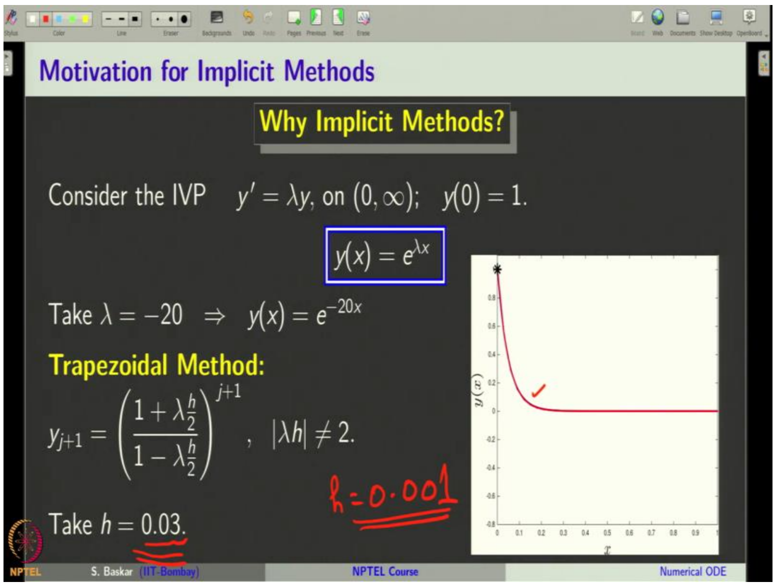
Now let us go on with the implicit methods and see what is this stability situation in the implicit methods. Just for the illustration, let us take the trapezoidal method. Recall, the trapezoidal method is an implicit method because it gives an implicit relation for the unknown y_{j+1} and this implicit relation will be a non-linear equation, if you are working with a non-linear ODE. However, in the present case we are working with a linear ODE therefore trapezoidal method in fact can be written explicitly.

And therefore, we do not need any non-linear iterative method or we do not need any predictor corrector approach. You can see that this expression can also be written like this and it is well defined for any λ and h such that $|\lambda h| \neq 2$. Of course, we are always taking λ to be negative and h to be positive. With this you can in fact see that the expression within the bracket will always lie between -1 and 1.

That shows that whatever may be the choice of h, as long as $h > 0$, when you are working with $\lambda < 0$, you always have the expression within the bracket to be something lying between -1 and 1. Therefore when you raise that to the power of j and as $j \to \infty$, you will always see that $y_j \to 0$, with respective to any choice of h positive. So, that shows that trapezoidal rule does not demand any condition in order to achieve the stability.

That is the good part of trapezoidal method. Not only trapezoidal method, it also holds for many implicit methods. That is why, implicit methods are preferred especially for stiff problems.

(Refer Slide Time: 21:01)



Motivation for Implicit Methods

Why Implicit Methods?

Consider the IVP $y' = \lambda y$, on $(0, \infty)$; $y(0) = 1$.

$\boxed{y(x) = e^{\lambda x}}$

Take $\lambda = -20 \Rightarrow y(x) = e^{-20x}$

Trapezoidal Method:

$$y_{j+1} = \left(\frac{1 + \lambda\frac{h}{2}}{1 - \lambda\frac{h}{2}}\right)^{j+1}, \quad |\lambda h| \neq 2.$$

Take $\underline{h = 0.03}$.

$\underline{h = 0.001}$

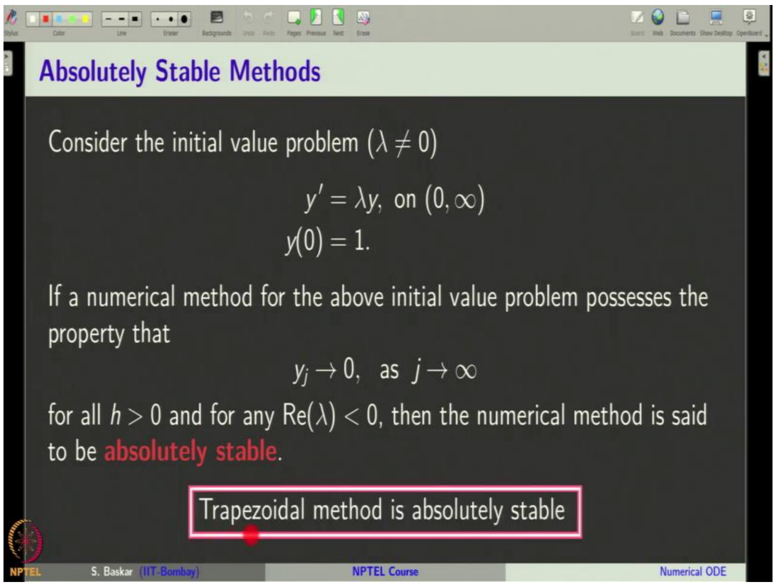
NPTEL S. Baskar (IIT-Bombay) NPTEL Course Numerical ODE

Let us see numerically, how trapezoidal method works. Let us take $h = 0.11$. Recall that for this value of h, the approximate solution computed from forward Euler method actually blows up as $x \to \infty$. Here you can see that for $h > 0.1$, the trapezoidal solution still decays and tends to 0 as $x \to \infty$. However, the accuracy is not that satisfactory, especially for small values of x. But that does not matter, when you decrease h little bit, you can see that trapezoidal method captures the exact solution more accurately.

If you recall, Euler method captured the exact solution with this accuracy almost for h = 0.001, which is very small when compared to the h that is given for the trapezoidal method. Of course, you can prove that trapezoidal method is of order 2 and also it is an implicit method. So, trapezoidal method has two advantages, one is it is an implicit method, it is unconditionally stable.

Second thing is, it is a second order method. Therefore, even for a relatively bigger value of h, you may get more accurate result when compared to the first order methods.

(Refer Slide Time: 22:44)



Absolutely Stable Methods

Consider the initial value problem $(\lambda \neq 0)$

$$y' = \lambda y, \text{ on } (0, \infty)$$
$$y(0) = 1.$$

If a numerical method for the above initial value problem possesses the property that

$$y_j \to 0, \quad \text{as } j \to \infty$$

for all $h > 0$ and for any $\mathrm{Re}(\lambda) < 0$, then the numerical method is said to be absolutely stable.

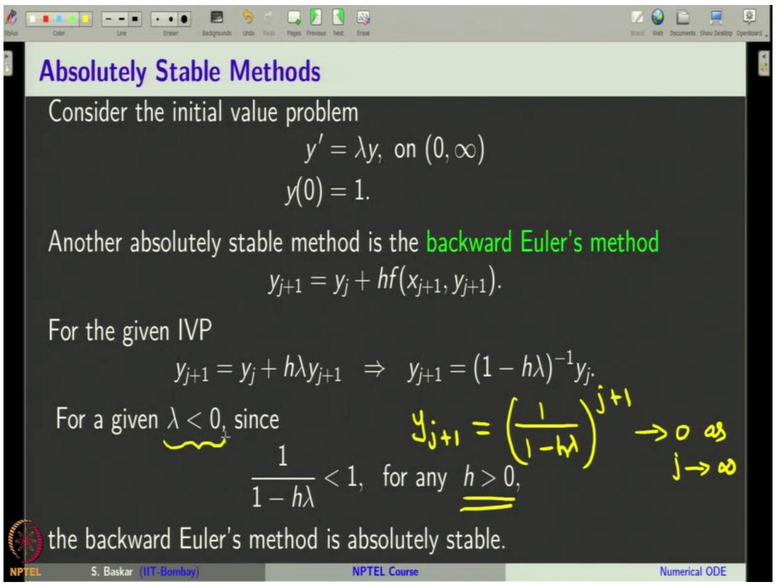
$\boxed{\text{Trapezoidal method is absolutely stable}}$

Let us formally define absolute stable methods, for the initial value problems of the form that we considered here. A numerical method for the above initial value problem possesses the property that $y_j \to 0$ as $j \to \infty$ for all $h > 0$. So, there should not be any restriction on h in order to achieve this property for any real part of $\lambda < 0$. Remember, if you have this property then the exact solution of this problem will have exponential decay.

And a good numerical method should capture that property in its approximate solution and we say that a method is absolutely stable if this property is achieved for any choice of h positive. So, that is what is highly desirable for numerical methods, if you want this property. In fact the message from our previous example is that you have to go for an implicit method.

If you go for an explicit method, like Euler method, they all come up with some condition, under which you have this stability therefore they are not absolutely stable. Whereas, implicit methods are often absolutely stable. Particularly, we have seen that trapezoidal method is absolutely stable.

(Refer Slide Time: 24:23)



Absolutely Stable Methods

Consider the initial value problem

$$y' = \lambda y, \text{ on } (0, \infty)$$
$$y(0) = 1.$$

Another absolutely stable method is the backward Euler's method

$$y_{j+1} = y_j + hf(x_{j+1}, y_{j+1}).$$

For the given IVP

$$y_{j+1} = y_j + h\lambda y_{j+1} \Rightarrow y_{j+1} = (1 - h\lambda)^{-1} y_j.$$

For a given $\underbrace{\lambda < 0}$, since

$$\frac{1}{1-h\lambda} < 1, \text{ for any } \underline{\underline{h > 0}},$$

$$y_{j+1} = \left(\frac{1}{1-h\lambda}\right)^{j+1} \to 0 \text{ as } j \to \infty$$

the backward Euler's method is absolutely stable.

In fact, there is one more important method which is absolutely stable, is the backward Euler's method. If you recall, in the very first lecture of this chapter, we have introduced backward Euler method but there we have used backward Euler method as an explicit method because we used it to compute the solution at the points on the left side of the initial condition. In that case, the backward Euler method becomes an explicit method.

But now what we are doing is, we are using the backward Euler method in a different way. We are using it to compute the solution of the initial value problem, at the points which are lying on the right side of the initial point x_0, which is taken in our problem as 0. And we are interested in obtaining the solution for $x > 0$. Therefore, if you try to apply the backward Euler method for this problem, you will see that the backward Euler method will also be an implicit method.

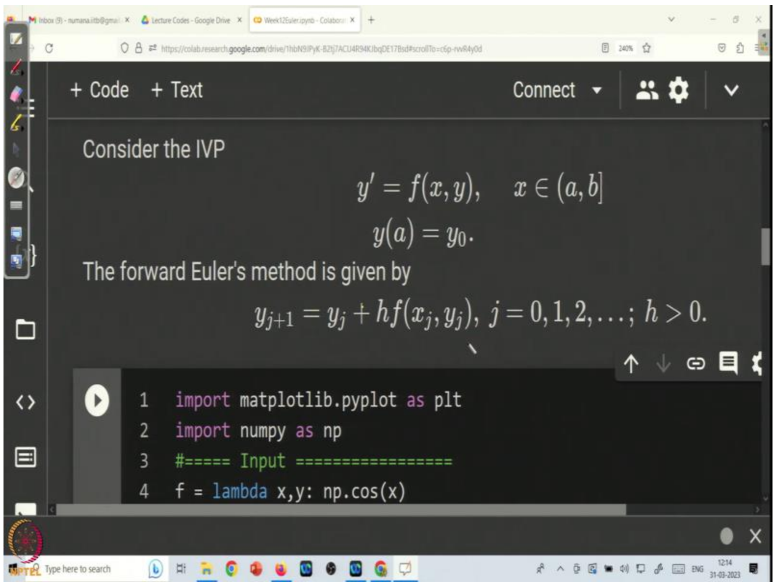
More precisely for this initial value problem, the backward Euler method is given by this expression and from here you can reduce this formula to $y_{j+1} = \frac{y_j}{1-h\lambda}$. You see λ is negative and h is positive therefore $(1 - h\lambda)$ will always be greater than 1 and therefore $\frac{1}{1-h\lambda}$ will be always less than one. Again, you see that backward Euler method is absolutely stable.

Because, you see that backward Euler method for this particular initial value problem is written as $\frac{1}{(1-h\lambda)^{j+1}}$ and that tends to 0 as $j \to \infty$, irrespective to whatever may be the choice of h

positive. Remember, we always discuss the stability under the condition that $\lambda < 0$. Therefore, backward Euler method is also absolutely stable. With this, our discussion on stability analysis for initial value problem is complete.

From here, you can clearly see the importance of using implicit methods when compared to the explicit methods.

(Refer Slide Time: 27:19)



Consider the IVP

$$y' = f(x,y), \quad x \in (a,b]$$
$$y(a) = y_0.$$

The forward Euler's method is given by

$$y_{j+1} = y_j + hf(x_j,y_j),\ j = 0,1,2,\ldots;\ h > 0.$$

```
import matplotlib.pyplot as plt
import numpy as np
#===== Input ================
f = lambda x,y: np.cos(x)
```

Before ending this lecture, we will also spend some time on implementing the Euler's method for an initial value problem. Let us first recall the forward Euler method. Forward Euler method is applied to an initial value problem of the form $y' = f(x,y)$ posed on the interval $(a,b]$ with the initial condition as $y(a) = y_0$. The forward Euler formula is given by $y_{j+1} = y_j + hf(x_j, y_j)$ and it runs for $j = 0, 1, 2, \cdots$.

We have to also give the value of h as the input. Apart from h, we have to also give the expression for the right hand side function f as a function of x and y.

(Video Starts: 28:21)

Let us see how to code this method using python. As usual, the first two lines are to import the libraries for plotting graph and also the numpy. The first line of the input is about taking the expression for the right hand side function f. Here, I want to take the right hand side function as simply cos x. Therefore, I have np·cos x because cos is not a function inbuilt in Python but it is inbuilt in the module numpy, that is why we have np·cos x.

Of course, I have given two parameters here x and y, but in this particular function *f*, I have only x dependency and y dependency is not there. but still I have y, because if you wish to change this function to something like $x^2 + y^2$, you can do that without disturbing the input parameter that is why I just kept here. Otherwise, y has no role for this particular definition of *f*. Next is, the value of h. We are taking the value of h as 0.1 and then we are discretizing the interval $(a, b]$.

Here I am taking *a* as 0 and *b* as 6, therefore our interest in solving the problem is in the interval [0,6]. We are discretizing the interval [0,6] uniformly with step size as 0.1. Therefore, your array x = (0, 0.1, 0.2, ⋯, 5.9, 6). The next input is, the initial condition at $x = 0$. We are taking $y_0 = -1$, with this we have taken all the necessary inputs for our problem.

Now we are ready to apply the forward Euler method. The first line is to get the information about the length of the array x. That will tell us, how many grid points, that is node points, are there in our problem. It is $x_0, x_1, \cdots, x_n$ and then we are initializing the approximate solution array, which has *n* components $y_0, y_1, \cdots, y_n$, its length is equal to the length of the array x.

And just to start with, we are initializing the array y with zero value and then we are putting y[0], which is the first component of the array y, with the initial condition. So, that is how it goes. y0 is the initial condition, that is what we are assigning here, from y[1] onwards, we will compute using the Euler forward formula and that is given by $y_{j+1} = y_j + hf(x_j, y_j)$. That is what precisely we are writing here in the python notation and that will go for each $j = 0,1,2, \cdots, n-1$.

So, that is what is happening here. In fact, I should write it as n then only it will go up to n – 1, that is the way range works. Therefore, it should be 0 to n then it will go from 0 to n - 1 in Python and that will make the vector $y_1, y_2, \cdots, y_n$. So, once we have calculated the array y then we will go to plot the graph of the approximate solution y. You can see that the exact solution of the initial value problem, that we have considered is $y = sin\, x - 1$.

Our interest also is to plot the exact solution and compare it with the approximate solution. Remember approximate solution is stored in the array y and we are printing that in the first part of the plot command x, y and in the same plot command we are also plotting x, sin(*x*0 – 1.

Therefore, in this single command we are actually plotting two graphs. One is the approximate solution and another one is the exact solution and this is just the legend.

And with this the Euler code is complete. Let us run the code and see the output. The code is executed and now we can see the output of the code. Here you can see that the Euler solution is plotted with blue solid line and the red solid line shows the exact solution. You can see that the accuracy is not that good. However, if you reduce the step size here, say you go and reduce it to 0.01, you can see that the approximation will be now better, as you can see here.

Let us also change the function, say for instance, let us take the function as $-20 * y$. If you recall, this is what we have discussed in our stability analysis part and let us take h $= 0.11$. From our discussion, you can see that the Euler forward formula should give unstable solution, which is clearly seen here. Of course, it is not $f(x) = sin\, x - 1$, just because we asked it to print like this it is printing.

But now it is –20 * y and you can clearly see that the Euler method shows furious oscillation and also it blows up as you go on increasing the value of *x*. You can see that the oscillation up to $x = 6$ is between -20000 to 20000, which is really big, whereas the exact solution is nicely decaying from 1 to 0 as $x \to \infty$.

(Video Ends: 36:01)

From this lecture, we can clearly see the importance of the implicit method and we also learn to implement the forward Euler method. Similarly, you should also think how to implement the trapezoidal method and Runge-Kutta methods. With this note, let us end this lecture. Thank you for your attention.