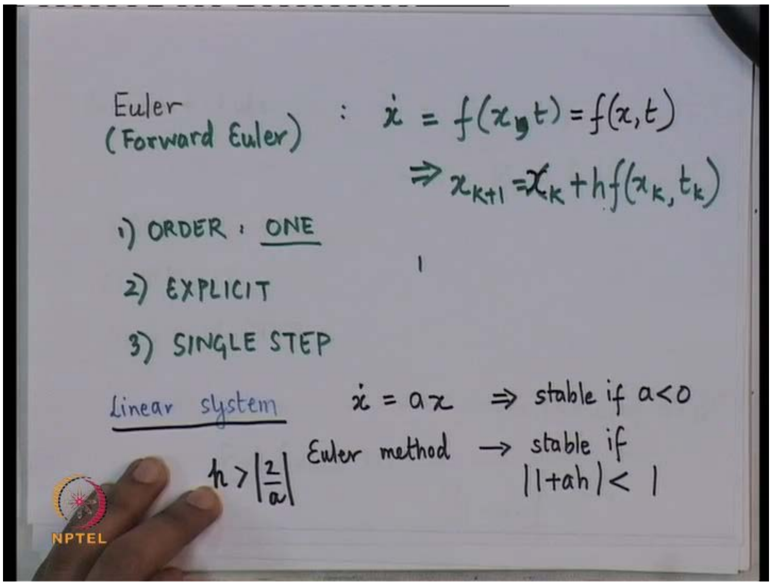
Power System and Dynamic Control
Prof. A M Kulkarni
Department of Electrical Engineering
Indian Institute of Technology Bombay

Module #01
Lecture #08
Numerical Integration (Contd.)

In the previous class, I have introduced you to numerical methods of solving differential equations. In this particular lecture, we will continue that strength and we will do a few examples on psi lab and to illustrate the basic properties of most of the commonly used numerical methods. The examples which I will chose today are examples which we have done before we have tried to analyze them using some intuition as well as linear analysis in some cases.

Today, let us just first revise what we have done in the previous class. In the previous class we learnt a few numerical methods like Euler method.

(Refer Slide Time: 01:12)



Euler
(Forward Euler) : $\dot{x} = f(x, t) = f(x,t)$

$\Rightarrow x_{k+1} = x_k + h f(x_k, t_k)$

1) ORDER : ONE
2) EXPLICIT
3) SINGLE STEP

Linear system $\dot{x} = ax \Rightarrow$ stable if $a<0$

Euler method → stable if $|1+ah| < 1$

$h > \left|\frac{2}{a}\right|$

So, let us just quickly go through this and then begin with our examples. Euler method or is also known as forward Euler method is the first is a first method which is taught to any student studying numerical methods in for integration of differential equations. It is a very simple method. In fact, if you have got a differential equation x dot is equal to x comma t, this comma looks a bit big, so I will just write it down again x of t.

In that case, we evaluate the function using this algorithm at discrete time steps spaced by h. So, we have got x k 1 is equal to x k this is x x k plus h into f into x k and that is the function f evaluated at the point x k and t k. Forward Euler method or simply Euler method is having order 1 that is, if your numerical solution can be expressed simply as a first order polynomial in t. Then, Euler method will you exact answers. Typically of course, you know that our responses are not of first order are not first order polynomials in t, we have in fact come across sinusoids and exponential functions which are not first order polynomial. So, Euler method is likely to have some error whenever you integrate most common differential equations.

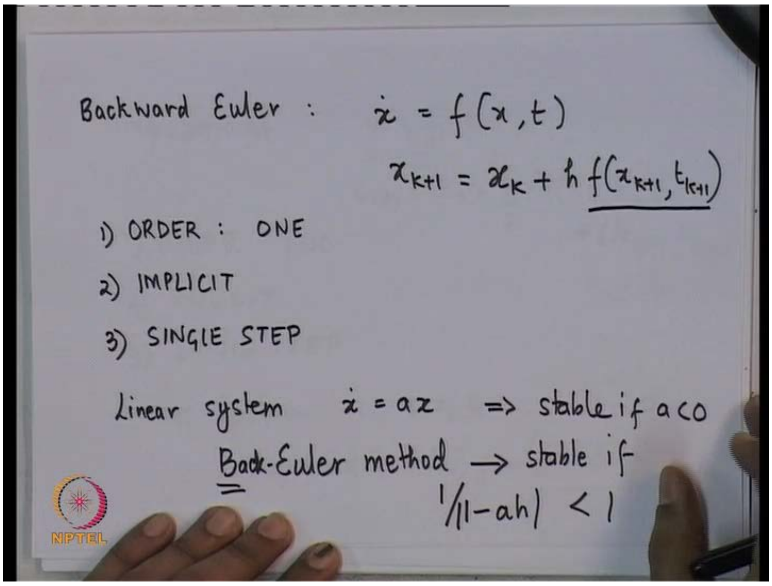
It is also an explicit method in the sense to evaluate x k plus 1; you really do not have to solve any differential equation. You have to just write the, if you know the previous value of x k. In one shot you can get x k plus 1. So, you get essentially in explicit algebraic equation where in x k plus 1 is written explicitly in terms of x k. Forward Euler is also a single step method. So, you just require one previous, the value at the previous time step in order to evaluate the value at the next time step.

If you look at a special class of systems that are linear systems then, one thing we can attempt to do since we know the solution of linear equations, we can compare what the numerical method gives us and what is the actual solution of a linear system. Now, we cannot do this in general for non-linear system because, you cannot write down the response of a non-linear system in terms of simple functions. In contrast, the linear systems for example, x dot is equal to x we know, it is stable when a is less than 0. If a is a real number then, if a is less than 0 then it is a stable.

Euler method, if you apply Euler method, if you apply this algorithm where f of x is nothing but, a of A x. In that case, Euler method tells you that the system is stable if mode of the absolute value of 1 plus A h is less than 1. So obviously, the Euler method may give these two conditions in fact are not equivalent. You know, if you look at, you know, for example, if h is greater than 2 by a, in that case even though a is less than 0 you may still have Euler method showing the system to be unstable.

So, Euler method when you integrate the system may be give you wrong answer. Even, not only wrong answers, it will give you a qualitatively wrong result. It may show a stable to system to be a unstable one. So, this is basically 1 one problem with Euler method that, you know, you will not get a accurate answer. Then, you may even get qualitatively wrong answers.

(Refer Slide Time: 05:14)



Backward Euler : $\dot{x} = f(x,t)$

$$x_{k+1} = x_k + h\,\underline{f(x_{k+1}, t_{k+1})}$$

1) ORDER : ONE

2) IMPLICIT

3) SINGLE STEP

Linear system $\dot{x} = ax$ $\Rightarrow$ stable if $a<0$

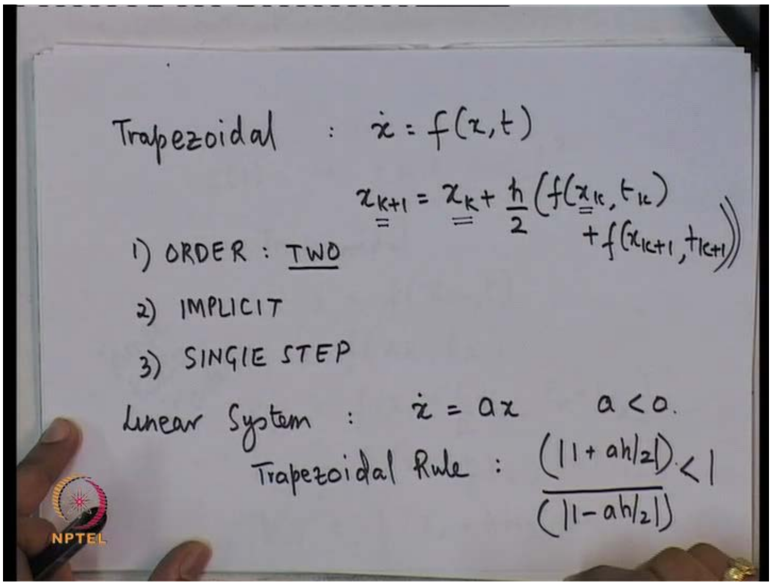
$\underline{\underline{\text{Back-Euler}}}$ method $\rightarrow$ stable if $1/|1-ah| < 1$

Look at backward Euler method, backward Euler method; the only difference is that the function evaluation is done at the new value of x. It is in fact done at the point which you want to evaluate.

So although, backward Euler is also order 1 this is something you can prove. I am not proved this in this class. It is implicit, it is an implicit method in the sense to evaluate x k plus 1. You have to solve this equation x k plus 1 appears in within this function. Therefore, you need a numerical method at every time step in order to get x a plus 1. So, this is; if for, if x is f is a non-linear function, you will have to use some numerical method to solve algebraic equations like Newton Raphson or Godse did.

Backward Euler also is a single step method. For a linear system, Backward Euler is stable if, this condition is satisfied, that is, 1 upon the absolute value of 1 minus A h less than 1. This condition also is not equivalent to this condition. In fact, an interesting point which I will show you sometime later 1 one of those examples is that, you can chose values of h so that, an unstable system is shown as a stable system if you use backward Euler method. So, backward Euler method also can in sometimes, if depending on the choice of the value h, give you qualitatively wrong answers as far as stability is concerned of a linear system of this kind.

(Refer Slide Time: 06:57)



Trapezoidal : $\dot{x} = f(x,t)$

$$\underline{\underline{x_{k+1}}} = \underline{\underline{x_k}} + \frac{h}{2}\left(f(\underline{\underline{x_k}}, t_k) + f(x_{k+1}, t_{k+1})\right)$$

1) ORDER : TWO

2) IMPLICIT

3) SINGLE STEP

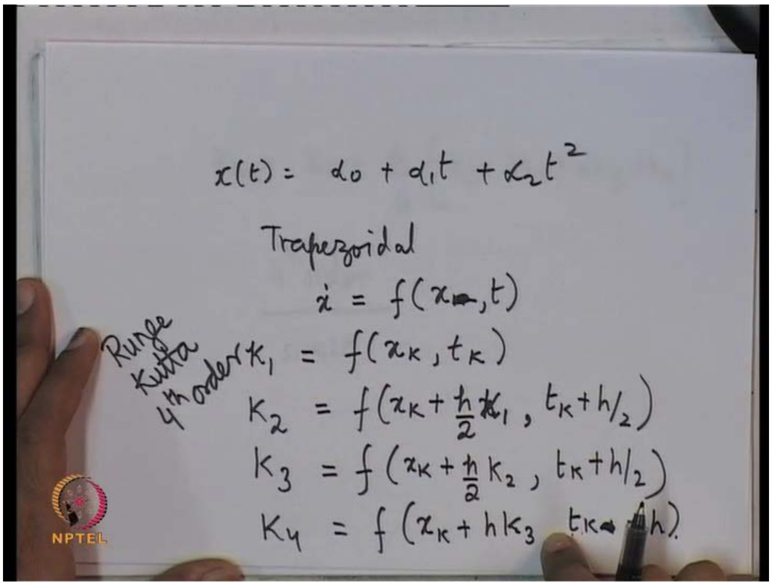
Linear System : $\dot{x} = ax$ $\quad a < 0.$

Trapezoidal Rule : $\dfrac{(|1+ah/2|)}{(|1-ah/2|)} < 1$

Trapezoidal method on the other hand is an order 2 two method. The basic algorithm is this; it takes the average of the function evaluation here, at the previous point and the new point. It is an implicit method again and it is a single step method. It requires only the previous value of x in order to get the new value of x. For linear systems, trapezoidal rule give says that this system is stable if this is less than 1. Whereas, the actual system is stable when a less than 0 but, interestingly this is a very interesting fact which I leave to you to work out; these 2 two conditions are equivalent in the sense that, if a less than 0 it also means that absolute value of 1 plus A h by 2 divided by absolute value of 1 minus A h by 2 is less than 1.

So that is an interesting thing. Trapezoidal rule, whenever your simulating linear systems will never give you a wrong result as far as stability is concerned. So, that is something interesting and useful. But, of course, one important point is, implicit methods like backward Euler and forward Euler are difficult to solve because, especially when you come to non-linear systems, because you will need to use a numerical method to compute x k plus 1 from x k. It is a iterative method. At every time step itself, you will have to iterate and solve a numerical solve a non-linear algebraic equation. When you are talking of linear systems of course, it would mean typically a matrix inversion or a solution of a linear system of equations.

(Refer Slide Time: 08:47)



$$x(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2$$

Trapezoidal

$$\dot{x} = f(x, t)$$

Runge Kutta 4th order

$$k_1 = f(x_k, t_k)$$

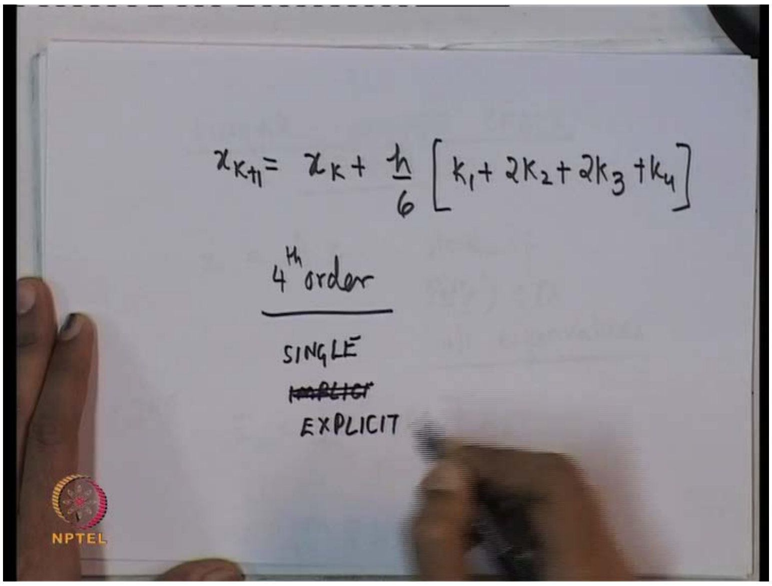
$$k_2 = f\left(x_k + \frac{h}{2}k_1, t_k + h/2\right)$$

$$k_3 = f\left(x_k + \frac{h}{2}k_2, t_k + h/2\right)$$

$$k_4 = f(x_k + hk_3\ t_k + h)$$

We also discussed in the previous class; not trapezoidal, this method which I am talking of is Runge Kutta method. Runge Kutta method of fourth order has the following evaluations; you have to evaluate k 1 at these points. I have done this in the previous lecture, evaluate k 1 first, that is the function evaluation; k 2 is a function evaluation at a slightly different point depending on which depends on the first evaluation. So, f of x k plus h by 2 into k 1 k 1 is evaluated here. Similarly, you have got the evaluation of k 3 which requires the value of k 2 and k 4 which requires the value of k 3. So, what you see here, is sorry this is small error here.

(Refer Slide Time: 09:42)



$$x_{k+1} = x_k + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]$$

4th order

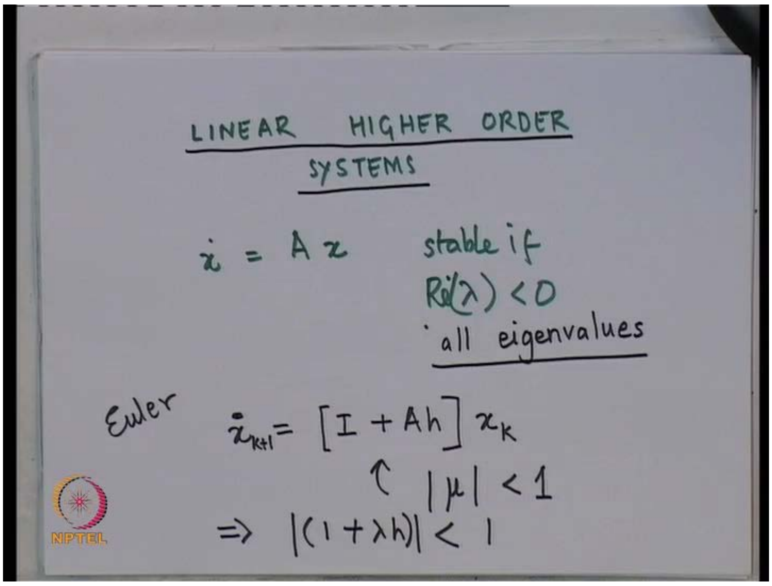
SINGLE

~~IMPLICIT~~

EXPLICIT

What you see here is; the final value of x k is a combination of these. Of course, Runge Kutta requires a lot many more evaluations and can be proven that this is the fourth order method. The 1 one I have shown you is a fourth order method. It is of course, single step. The reason is that we require only the value of x k in order to compute all these functions k 1 k 2 k 3 k 4 and it is an implicit method. At least this one which I have shown you is an implicit sorry it is an explicit method. I am sorry Runge Kutta fourth order which I have shown you here, is an explicit method. Explicit in the sense, that all these functions evaluations are single shot. They are explicit evaluations of functions. You do not have to solve any non-linear algebraic equations to evaluate any any value here.

(Refer Slide Time: 10:39)



LINEAR HIGHER ORDER SYSTEMS

$\dot{x} = Ax$ stable if $Re(\lambda) < 0$

all eigenvalues

Euler $\dot{x}_{k+1} = [I + Ah] x_k$

$\uparrow \; |\mu| < 1$

$\Rightarrow |(1 + \lambda h)| < 1$

Of course, when you are trying to numerically integrate linear higher order systems, again let me point out here, that the reason why I am studying the behavior of numerical methods using linear systems this is not because linear systems cannot be sort solved. We saw in the lectures, previous lectures that, linear systems in fact are amenable to exact solutions. You can write down the solution in terms of very well known functions. So, the only reason why I am using linear systems a numerical integration of linear systems, the study of that is because I would like to know how the numerical methods behave.

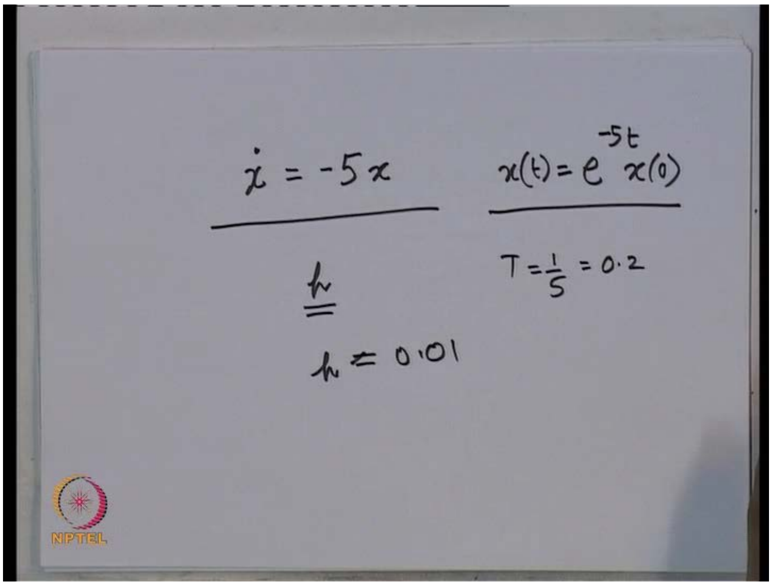
Now, if you have got a linear higher order system like, x dot is equal to x a is a matrix x is vector. It is stable you know, it is stable if real of the Eigen, any of the Eigen, all Eigen values should have the real part less than 0 then we say its stable. But, if I numerically integrate this higher order linear system; in that case what you have is, x k plus 1, this is a vector is 1 plus A h times x k, which is again a vector. This is a matrix. Now, this system is stable, this is a discrete time system. It is stable if, any the Eigen values of 1 plus A h is less than 1. This not something I am explicitly proving but, it is not difficult to prove. You can use the model analysis or linear systems analysis, Eigen values and Eigen vectors to analyze discrete time systems as well.

Now, if the absolute value of the Eigen value of 1 plus A h is less than 1, then it is stable system. That is what Euler method says. The original system is stable if all Eigen values are the real part less than 0. So, for a higher order linear system, Euler method will give will give a result that, if 1 plus lambda h the modulus of lambda h is less than 1, then it is stable. So, lambda is the Eigen value of a. So, let me just repeat; if the Eigen values of 1 plus A h have the modulus or absolute value less than 1 then, Euler method will show that the system is stable. It may be actually a stable system and Euler method may show it to me a unstable system if, this becomes greater than 1. It really depends on the value of h here.

You can show that mu is nothing but, 1 plus lambda h well where lambda are the Eigen value of a. So, the condition, this condition can be rewritten as the condition for this discrete time system to be stable is, 1 plus lambda h, the absolute value should be less than 1. So, again let me just clarify what I am trying to say; I am trying to say that, even if the original system is stable, if this condition is not satisfied then, your numerical method will give you a qualitatively wrong answer about stability. So, you have to be careful when you do this simulations that you do not get wrong answers.

Now, moving on we will do a few examples in psi lab. It would be wonderful if you have got psi lab or mat lab or even if you program and see your program FORTRAN and verify what I am trying to say here.

(Refer Slide Time: 14:23)



$\dot{x} = -5x$ $\qquad$ $x(t) = e^{-5t}x(0)$

h $\qquad$ $T = \frac{1}{5} = 0.2$

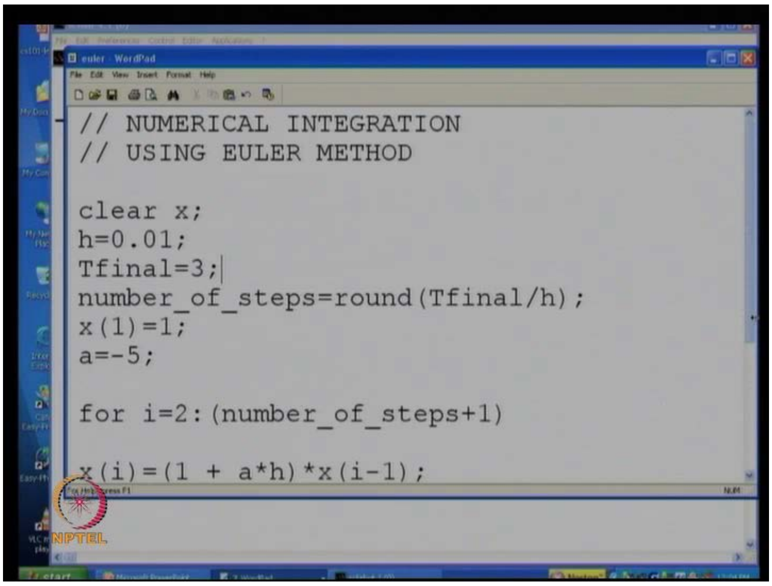
$h = 0.01$

What I will do is, I will kind of make an algorithm for doing the numerical integration first of a system x dot is equal to minus 5 x. This is a stable system. So, I will first numerically integrate it using Euler method. Actually, if Euler method is used remember that it will you know it is likely to give correct results. 1 One of the intuitive is of choosing your h is that, look at this Eigen value this Eigen value will probably will give in fact a solution of this kind.

We will just to try to verify whether we get that solution or not. In fact you have to chose choose your h carefully. If you look at, this particular you know, function it will decay almost to 0 in 5 times the time constant of this system. The time constant of this system is in this particular case 1 up on 5 that is, point tow two. It is a time constant of this system. So, we expect that, it will decay. This particular thing will decay if I give a non equilibrium initial condition. You will go to 0 which is the equilibrium in this case in about 1 about 1 one second.

So, I can just intuitively think that, if I want to get an nice, at least a nice picture of what is going to happen; I should chose h at least you know, something like 0.0 1 or so. We will just try out with various values of h and see what we get. I had done this in the previous class also. But, it will was somewhat in a hurried fashion. So, let us do it bit calmly in this particular lecture.

(Refer Slide Time: 16:19)

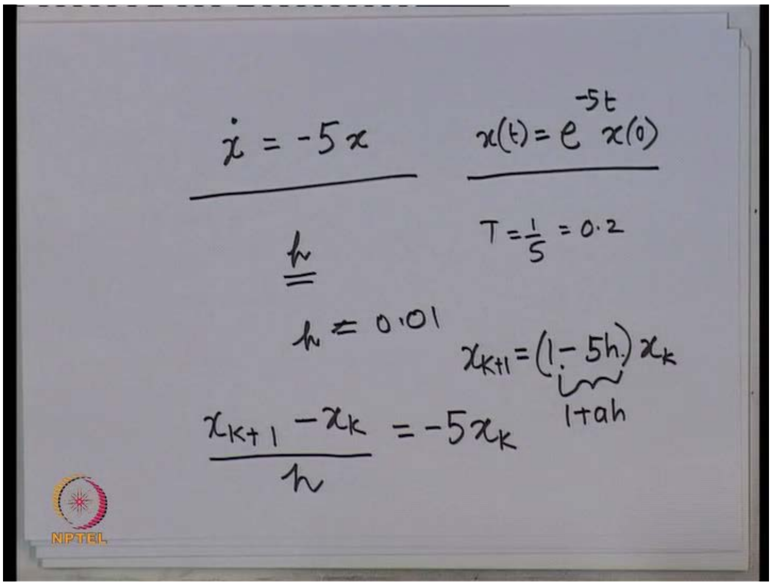


```
// NUMERICAL INTEGRATION
// USING EULER METHOD

clear x;
h=0.01;
Tfinal=3;
number_of_steps=round(Tfinal/h);
x(1)=1;
a=-5;

for i=2:(number_of_steps+1)

x(i)=(1 + a*h)*x(i-1);
```

So this is the basic psi lab window. I have already invoked psi lab. What we will do is, first of all, let us look at the solution of, so we will do a numerical integration using Euler method. First of all, we will clear this variable x. We will chose choose a time constant let us say of a time step of 0.01. Let us simulate or numerically integrate this system till 3 seconds because we know that it is for lightly to decay within that time. So, we have got a number of discrete time steps at which x has to be evaluated. So, let us assume the initial condition on x is 1. In fact, I should have written this x of 0. But, psi lab does not allow 0 as an index. So, that is why I have called it 1.

So, this is the value in fact at time t is equal to 0. So, I have given initial condition of 1. Remember x dot is equal to minus 5 x has got in equilibrium value of 0. So, we are giving an initial condition which is not the equilibrium value. So, you will see a transient, the value of a is 5 as I discussed some time back. So, what I will do is, I will evaluate the behavior of this system using Euler method. So, x of I is equal to 1 plus a into h into x of i minus 1. This is what Euler method will do.

(Refer Slide Time: 17:48)



$$\dot{x} = -5x \qquad x(t) = e^{-5t}x(0)$$

$$T = \frac{1}{5} = 0.2$$

$$\underline{\underline{h}}$$

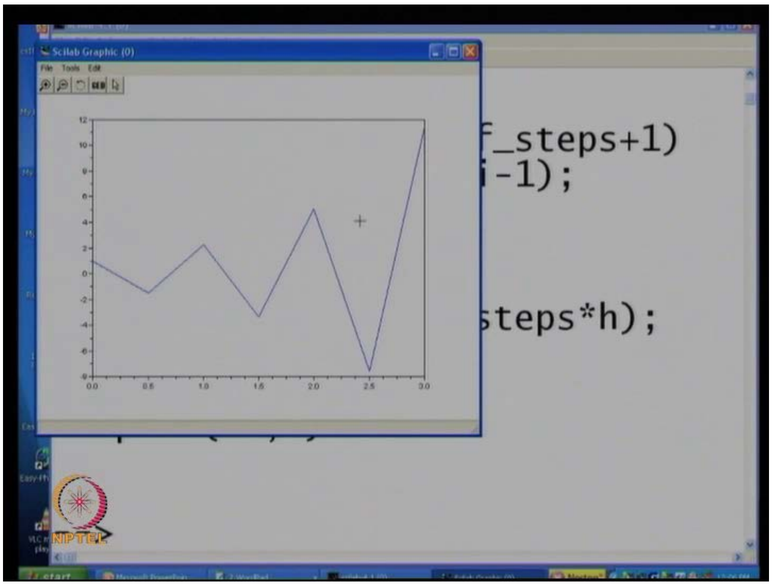
$$h = 0.01$$

$$x_{k+1} = (\underbrace{1 - 5h}_{1+ah})x_k$$

$$\frac{x_{k+1} - x_k}{h} = -5x_k$$

So if you recall, we have got x k plus 1 minus x k up on h is equal to minus 5 of x k. So, this will effectively give us x k plus 1 is equal to 1 minus 5 h into x k. In fact, minus give five h is nothing but, plus A h. This is nothing but, this is 1 plus A h, a itself is negative. That is why we are getting 1 minus 5 h.

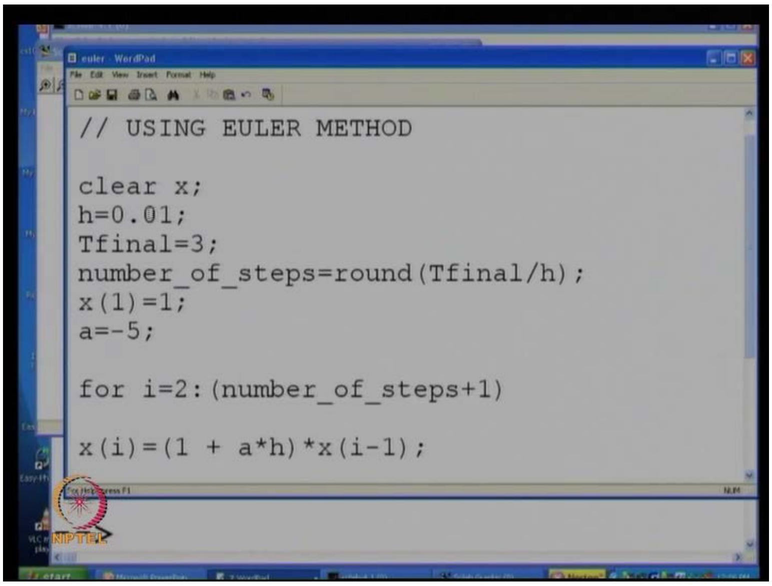
(Refer Slide Time: 18:35)



```
f_steps+1)
i-1);

steps*h);
```

So, what I will do is, I will evaluate this and plot it. So, the way to do it is, so we evaluate this. Hoops Oops! It is giving a growing function. Now, why is this happening? We will just look back. Perhaps, I have done, I have made some error here.

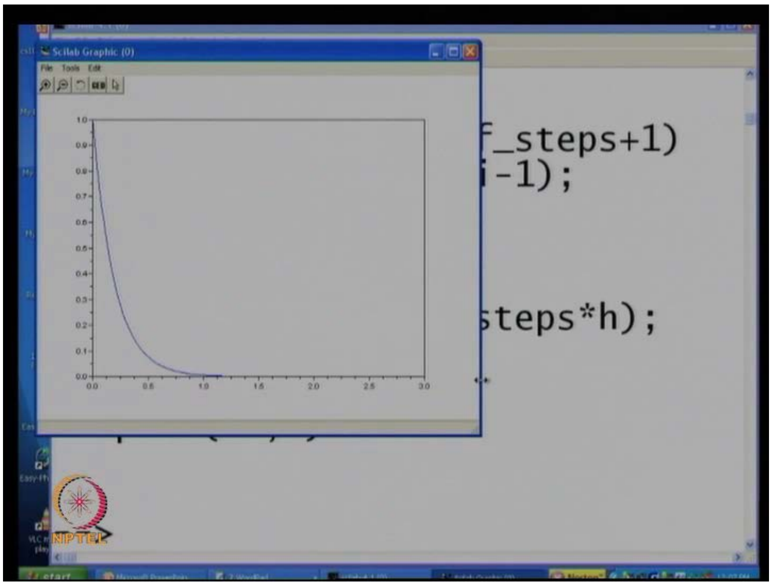
(Refer Slide Time: 18:50)



```
// USING EULER METHOD

clear x;
h=0.01;
Tfinal=3;
number_of_steps=round(Tfinal/h);
x(1)=1;
a=-5;

for i=2:(number_of_steps+1)

x(i)=(1 + a*h)*x(i-1);
```

H is equal to 0.01 that is oh okay. I am have not saved it. So, I need to save it first. yeah I have saved it first. yeah I have saved it. So, I have given the value of h is equal to 0.01 which I have saved changed from the earlier value and now I will run it again. So let me just run this again. s So, I am doing this of course, simulations of 3 seconds.

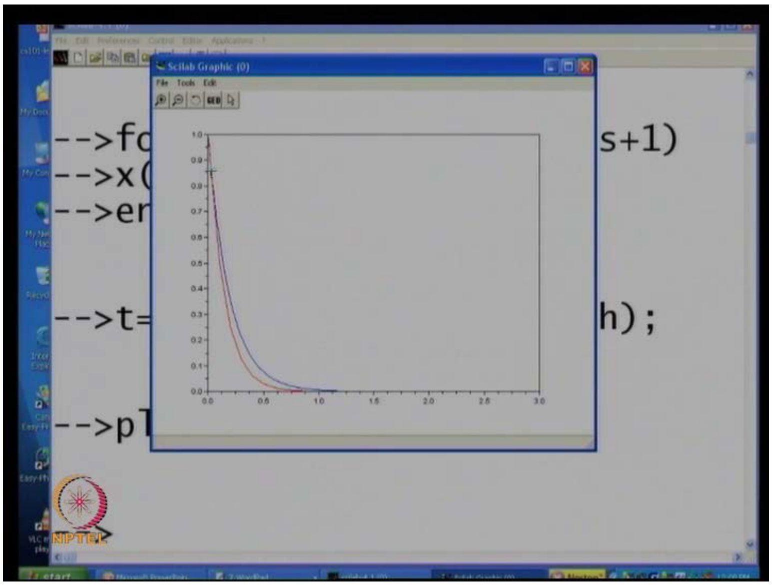
(Refer Slide Time: 19:14)



```
_steps+1)
i-1);

teps*h);
```

Yeah Now, what we are getting is what we kind of expect or exponentially decaying e raise to minus 5 t kind of response. The initial value is one, in case you are not able to see it clearly and this time period is from 0 to 3 seconds. So, what I will do next is, I will change the time step to 0.1 and just to make it a bit easier, to make see, we save it we plot it as a red.

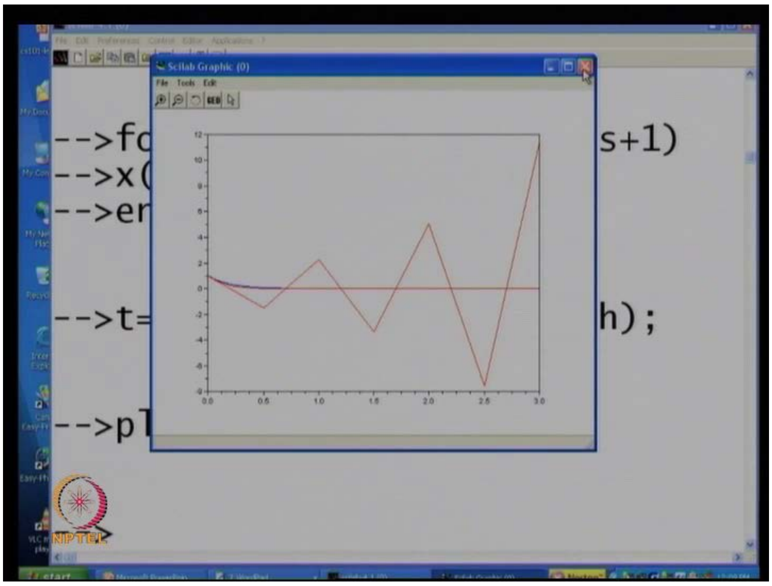
(Refer Slide Time: 20:04)



```
-->fc                                    s+1)
-->x(
-->er

-->t=                                    h);

-->pl
```

So if we look at this, you see that by changing the time step to 0.1, we have slightly diff, the behavior is has slightly changed. You see this red? If you look at this red response it is slightly different.

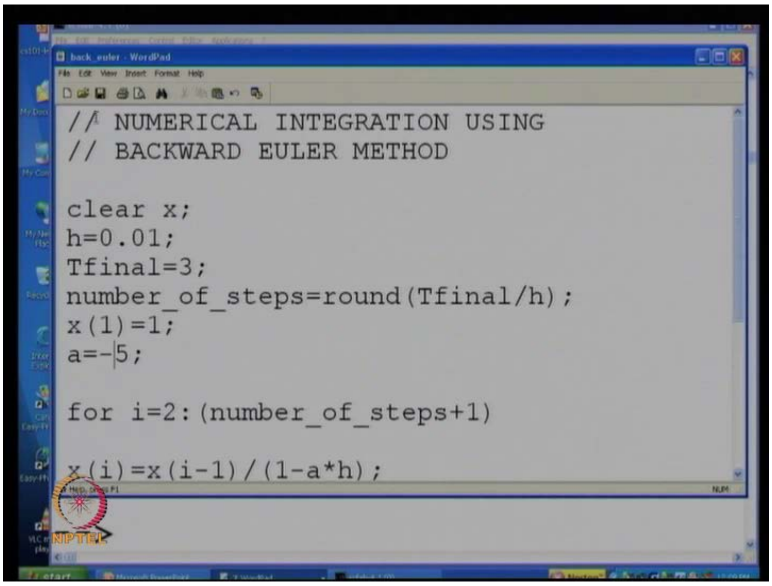
(Refer Slide Time: 20:45)



```
-->fc                                    s+1)
-->x(
-->er

-->t=                                    h);

-->pl
```

If I go ahead and I am sorry let may let me make this 0.5 s and look at the response. Well, you look what has happened. Your response is absolutely different from what is expected. So, if you chose a time step like 0.5 in which is in fact greater than 2 times 1 up on 5, so I had mentioned that, if h is greater than 2 by a the absolute value of 2 by a; in that case your response is altogether different. You really have a situation where you are getting an unstable response. This is not of course, the correct response. We say the numerical integration is blowing up. So, though the actual system is stable, the numerical method gives you a wrong answer. So obviously, you chose your time step carefully especially if you are using Euler method.

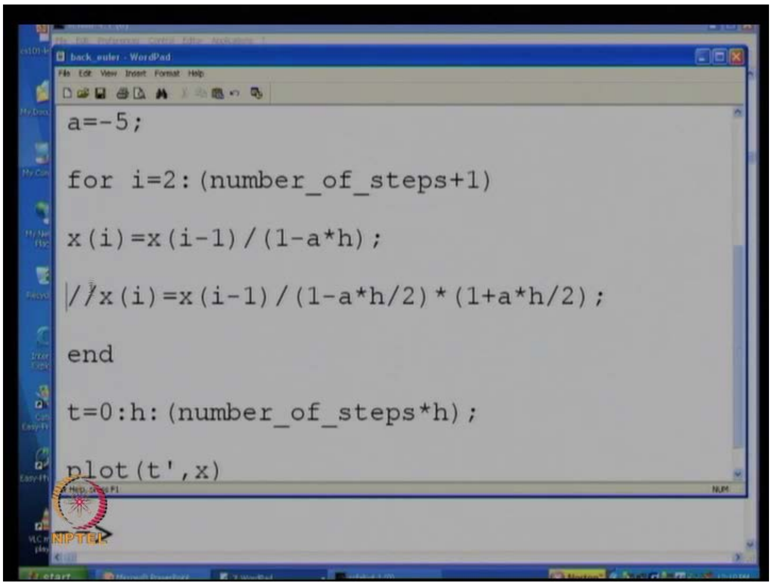
(Refer Slide Time: 21:38)



```
// NUMERICAL INTEGRATION USING
// BACKWARD EULER METHOD

clear x;
h=0.01;
Tfinal=3;
number_of_steps=round(Tfinal/h);
x(1)=1;
a=-5;

for i=2:(number_of_steps+1)

x(i)=x(i-1)/(1-a*h);
```

Let us now go on and try to simulate the same system using backward Euler method. So, let us just have the same thing. This is backward Euler method. h is equal to 0.01, t final is 3, number of steps of course, is the depends on the duration as well as h the initial value of x is 1 a is minus 5 so I will just save this. yeah

(Refer Slide Time: 21:38)

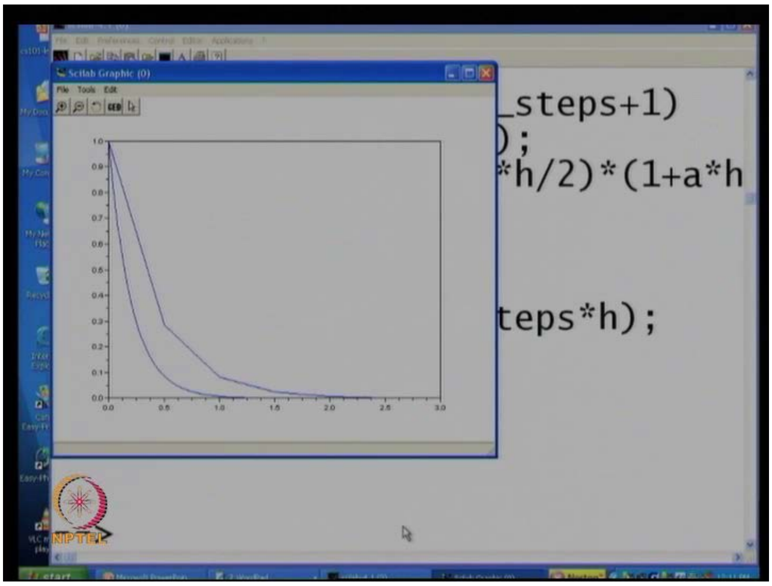


```
a=-5;

for i=2:(number_of_steps+1)

x(i)=x(i-1)/(1-a*h);

//x(i)=x(i-1)/(1-a*h/2)*(1+a*h/2);

end

t=0:h:(number_of_steps*h);

plot(t',x)
```

So as per backward Euler method, you will have X of k or X of I is equal to i minus divided of 1 minus a h. You can easily derive it from the basic formula for discretizing or you know using backward Euler method. Now, this particular sentence has been commented so, I am not actually going to implement this particular you know this particular command. What I am going to implement is this algorithm, X of k is equal to X of k minus 1 divided by 1 plus a h.

So, let us just numerically integrate and see what we get. This is with a time step of, I believe of, so we are getting the correct answer of course, the time step is 0 0.01 and a is minus 0.5. Now, I can make this also 5. We had done that before. Let us see what happens.

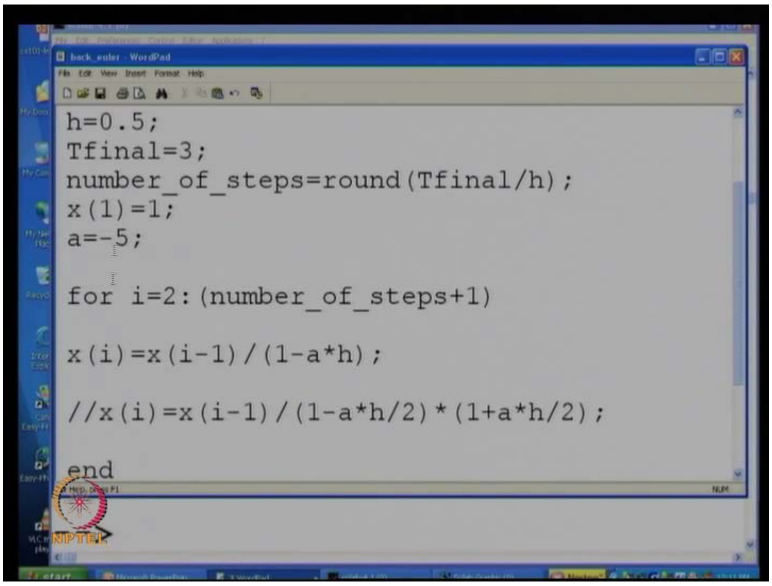
(Refer Slide Time: 23:07)



```
_steps+1)
);
*h/2)*(1+a*h

teps*h);
```

Well, the solution which you get is this. The interesting thing is that of course,there is an a kind of error you know. The earlier with a lower time step we got this response, now we are getting this slightly erroneous but, of course, it is still stable. So what what one one small point which we should remember is that, backward Euler a stable system will in fact be shown as a stable system.

(Refer Slide Time: 23:39)

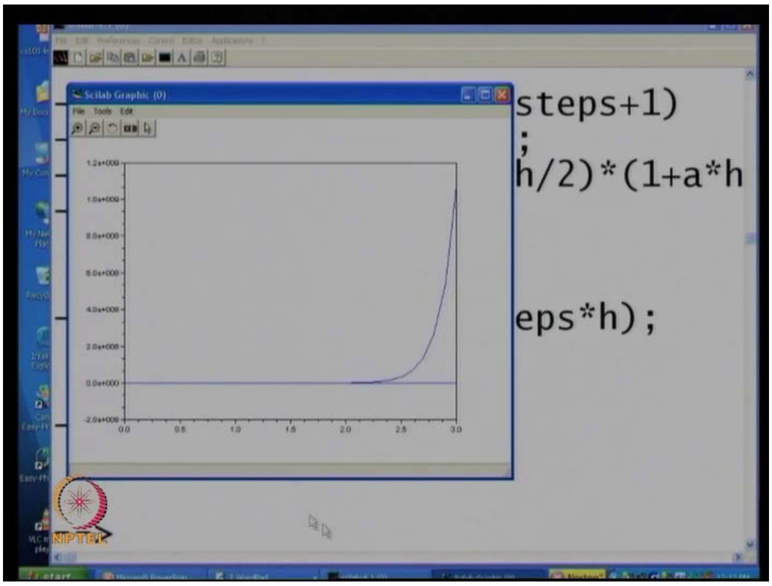


```
h=0.5;
Tfinal=3;
number_of_steps=round(Tfinal/h);
x(1)=1;
a=-5;

for i=2:(number_of_steps+1)

x(i)=x(i-1)/(1-a*h);

//x(i)=x(i-1)/(1-a*h/2)*(1+a*h/2);

end
```

So that is no worry. But, a very interesting thing is what if I have got an unstable system? A is equal to 5 and I chose my time step more than 2 times 1 upon 5. An interesting thing which you will see here is, that this system which ought to be unstable, you know after all a is equal to plus 5. This is the real system is unstable, e raise to 5 t into x of 0 that is the solution.

What you will find is, backward Euler, when you use backward Euler it is actually showing this system to be a stable one. That is very interesting. So, it really this something you should keep in mind that is backward Euler in fact is gives you, can also give you wrong qualitative answers if your time steps stamp are larger this is. Of course, not true if you chose a time step stamp to be small. So for example, if I chose 0.1 backward Euler of course, should give you the correct answer. Correct in the sense that is an unstable system so you see that the unstable system.

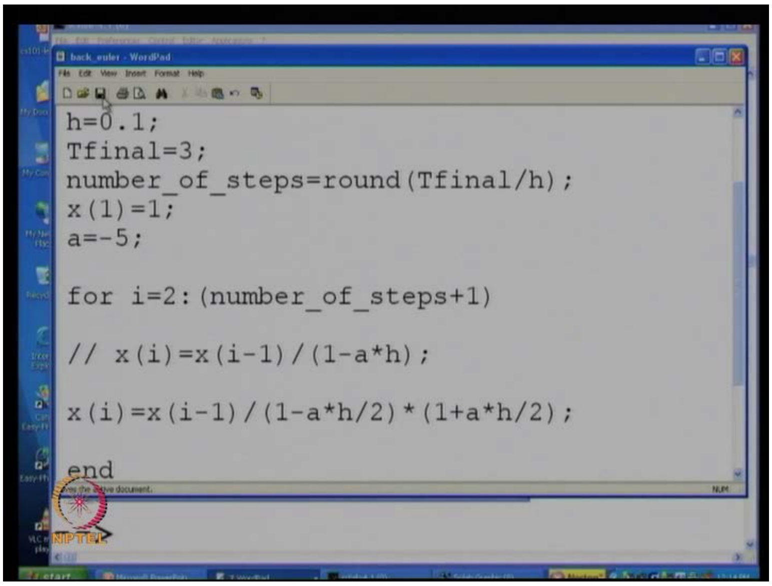
(Refer Slide Time: 24:38)



```
steps+1)
;
h/2)*(1+a*h


eps*h);
```

So it just grows up and if your, it is not visible on your screen, this is almost 1.2 into e raised to nine. So 1.2 into 10 raised to nine so, that really means that it is really blown out in 3 second itself. That is not very surprising.

(Refer Slide Time: 25:06)

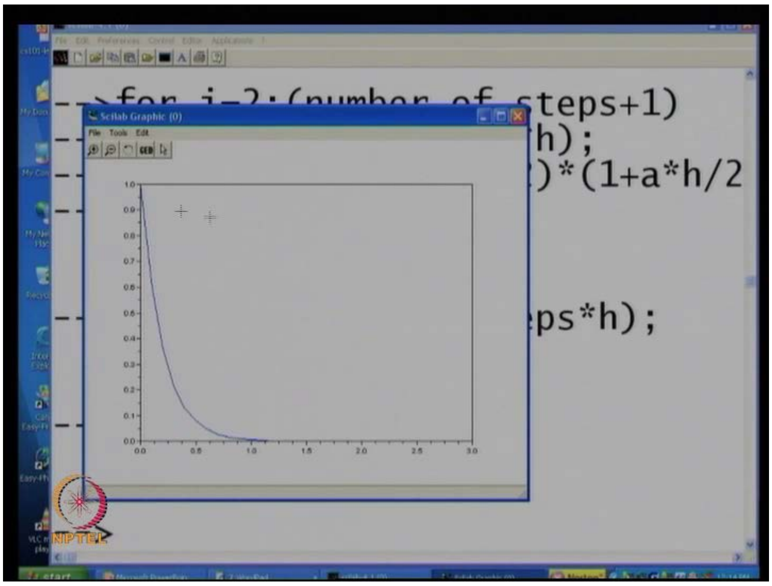


```
h=0.1;
Tfinal=3;
number_of_steps=round(Tfinal/h);
x(1)=1;
a=-5;

for i=2:(number_of_steps+1)

// x(i)=x(i-1)/(1-a*h);

x(i)=x(i-1)/(1-a*h/2)*(1+a*h/2);

end
```

So, this is about backward Euler. What if I do the same system simulation using trapezoidal rule? Trapezoidal rule has an interesting property that, whether your stable if you have a stable system, it will show be shown as stable. So for example, this is 0.01 just a sec I will just check whether I have done the right thing. oh I have to comment this statement and enable this statement which is essentially implementing trapezoidal rule. This is x of k is equal to x of this is x of I is equal to x of 5 minus 1 divided by 1 minus h by 2 into 1 plus h by 2.

So you can actually work out this that you get this particular algorithm if you apply trapezoidal rule to x dot is equal to a x. So, if you have got a is equal to minus 5 and h is equal to 0.1 as we saw just now. I save this. Run it.

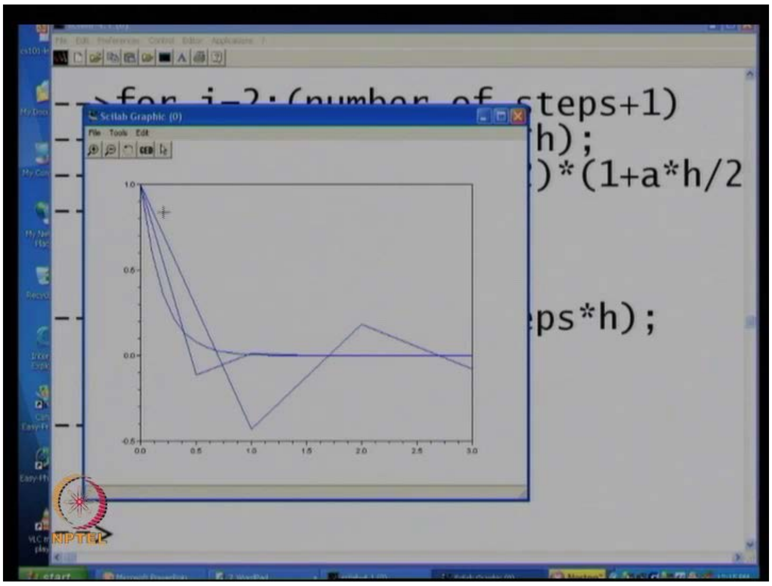
(Refer Slide Time: 26:04)



```
-->for i=2:(number of steps+1)
h);
)*(1+a*h/2
ps*h);
```

The Remember, although I called this file backward Euler; still it is doing trapezoidal. It is using trapezoidal method of integration now. So, what if I increase the time step? So, I am using trapezoidal rule. Just remember that. So, for example, 5 this was unstable for example, Euler method when we try to integrate it with h is equal to 5, it was unstable.

(Refer Slide Time: 26:34)



```
-->for i=2:(number of steps+1)
h);
)*(1+a*h/2
ps*h);
```

What does trapezoidal rule, is also stable of course, the errors in the response, so if I actually go on increasing h, the errors increase. But, as we shall see, I will just do something more sillier. I will make h is equal to 1 second. Even if I make h is equal to 1 second, although I am going to get an error in the response it is not unstable. This is my new response. Similarly, if a is equal to plus 5, if I chose h is equal to 1, this is an unstable system.

We will just do it one on a fresh plot. This is an unstable system and trapezoidal rule is showing it to be an unstable system. So, really trapezoidal rule you know, at least in a linear system it kind of preserves the stability information. So I have of course, if I make this time step smaller, the answer is of course, closer the right answer. yeah So, you see this increasing here like this.

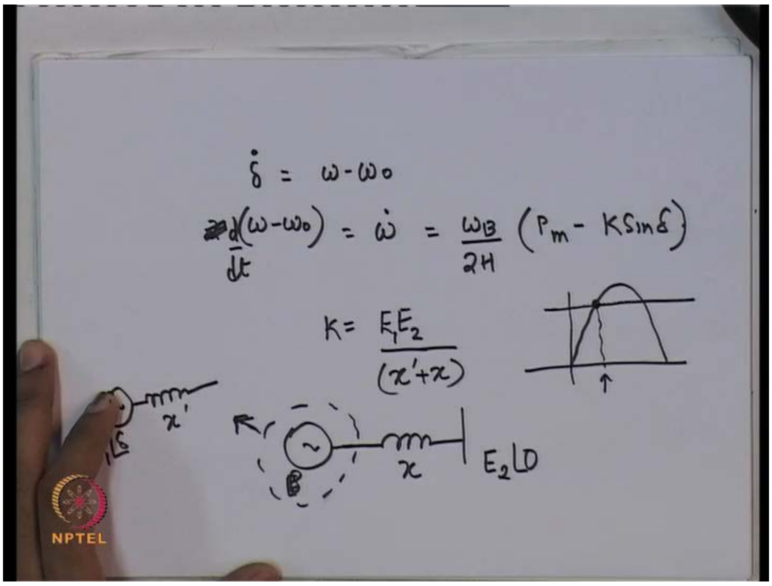
So to summarize whatever I have shown so far, is that when we are trying to numerically integrate a system, x dot is equal to minus 5 x; if you chose a Euler method, if you chose A h to be large, in fact if it is greater than the absolute value of 2 by a, a is equal to minus 5, in that case you in fact get a wrong qualitative picture about the stability. Of course, as you go on increasing the time step, a numerical methods accuracy versions worsens. So, it is not unexpected that you will get a better results if you chose h smaller. But, the important point which I can emphasize is that, Euler method may give you a qualitatively wrong result also. It may show you a stable system in fact to be unstable if your h is larger.

This does not happen with backward Euler or with trapezoidal rule. In fact, a stable system is always shown to be a stable whether one uses backward Euler or trapezoidal rule. But, interestingly with backward Euler method, I am essentially unstable system can be shown as s stable system if your h is very large. So, if h is greater than point 2 upon a, the modulus of it and unstable system for example, a is equal to plus 5 is shown to be stable 1 one whereas, trapezoidal rule preserves the stability information of a linear system.

So, that is one thing a very interesting point which we must note. Now, the basic you know, the 3 three method which I have shown you to get accurate solution in fact for example, x dot is equal to minus 5 x, it makes sense in fact to use methods with time steps of 0.1 0 also. You will get a reasonably accurate picture you know. It is another thing about you know, if you chose h is equal to 0.5 etc etc. Not only you are going to inaccurate picture but, in case of Euler method you will get even a qualitatively wrong picture.

So, you will use a it is a good idea if you have got a system with a time constant of point 2 or as in this case we use a time step of 0.01 and so on. But, you know 1 you know for example, if your situations were a system is marginally stable you know, you have got a system in which the real part of the Eigen value of the continuous time system 1 near 0; Euler method is absolutely hopeless. You will find that it does not give you a good you know, it gives you a qualitatively wrong picture. What I will do now is, try to simulate 1 one of the systems which we have simulated before, try to understand qualitatively, before that this swing equations of a power system.

(Refer Slide Time: 30:41)



$\dot{\delta} = \omega - \omega_0$
$\frac{d}{dt}(\omega-\omega_0) = \dot{\omega} = \frac{\omega_B}{2H}(P_m - K\sin\delta)$
$K = \frac{E_1E_2}{(x'+x)}$
x'
x
δ
$E_2\angle 0$

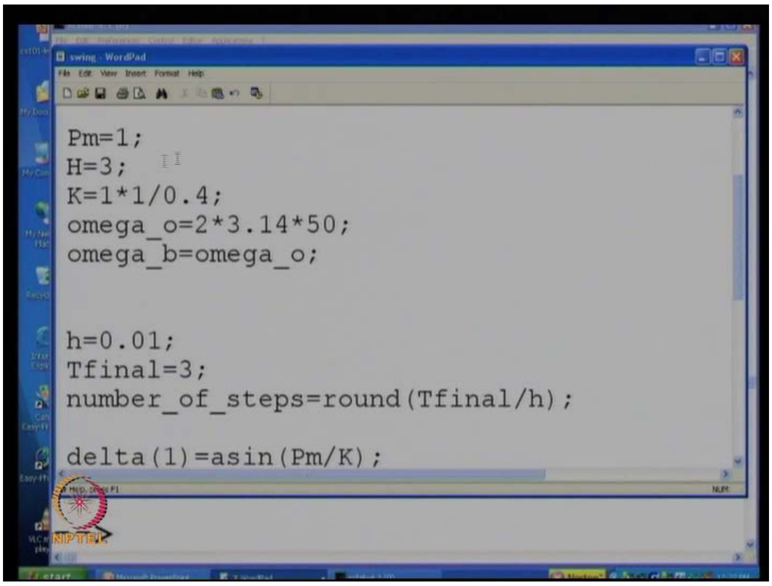
So the swing equations of a single machine infinite bus system or delta dot is equal to omega minus omega dot naught and the derivative of omega minus omega naught which is nothing but, omega dot itself because, omega naught is a constant which is the frequency of the infinite bus is equal to omega B by 2 H p m minus k sin delta.

K is in fact, we assume it to be a constant. This is the internal voltage of the synchronous machine. The infinite bus voltage divided by the reactants. If you have got a single machine connected to an infinite bus E 2 angle 0 and this is E 1. Well, this is a synchronous machine which is modeled as an internal. So, this is our synchronous machine. So this is E 1 angle delta ,this is the internal voltage, this is the rotor angle, x dash is a transient reactant. So, this these are the equations swing equations. This is a non-linear system of equations.

So, if I am trying to solve this system of equations by Euler method, what is the kind of response I will get? We saw for small disturbances, we have done the analysis of this system and we saw that, in fact, it is for the equilibrium corresponding to which is less than ninety degrees this one, for small disturbances around the equilibrium we will get essentially an oscillated response. So, we know that, this if you are at this equilibrium, you will get an oscillated response. If I give of course, a very large disturbance from this then, the system may even go unstable. The synchronous machine may fall out of step or lose synchronism.

So, we know the kind of responses which can be expected at least for small disturbances and qualitatively of course, we know that for large disturbances the system may go unstable. So, with this this is the non-linear system, by the way this is not a linear system because your delta appears within the sin function. So, what I will do is, try to integrate the swing equations by Euler's method.

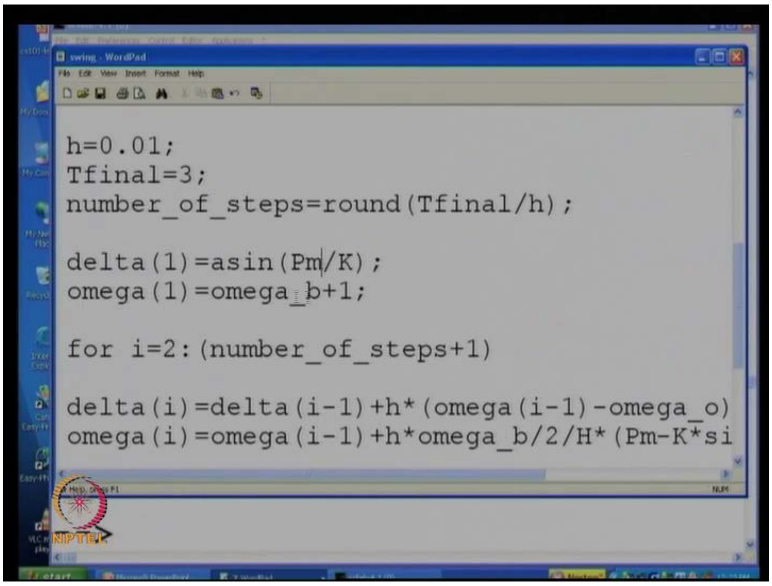
(Refer Slide Time: 33:23)



```
Pm=1;
H=3;
K=1*1/0.4;
omega_o=2*3.14*50;
omega_b=omega_o;


h=0.01;
Tfinal=3;
number_of_steps=round(Tfinal/h);

delta(1)=asin(Pm/K);
```

So, you just take, let us assume P m is 1, h is 3 at this capital h is of course, the inertial constant, k let us assume is e 1 e 2 divided by 0.4, this is omega and omega base we assume is the same as this frequency of the infinite bus.

(Refer Slide Time: 33:38)



```
h=0.01;
Tfinal=3;
number_of_steps=round(Tfinal/h);

delta(1)=asin(Pm/K);
omega(1)=omega_b+1;

for i=2:(number_of_steps+1)

delta(i)=delta(i-1)+h*(omega(i-1)-omega_o)
omega(i)=omega(i-1)+h*omega_b/2/H*(Pm-K*si
```

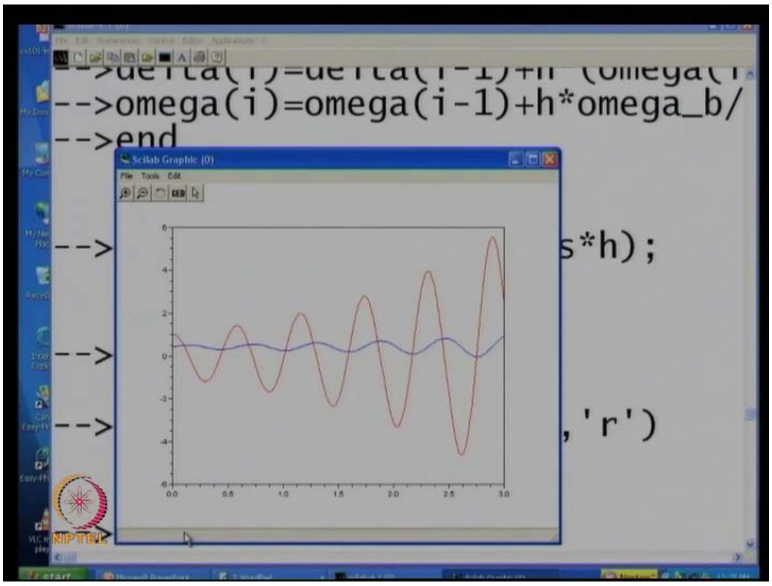
Let us take time step of 0.01, the final time is 3, the number of steps of course, can be calculated from t final n h. Let us assume that delta is equal to sin inverse, this sin inverse p m by k. In fact this is the equilibrium value of delta. Omega is nothing. Omega, let us assume is omega b plus 1. Now, remember one important point is that, although delta is at the equilibrium value of that the value of delta is actually its equilibrium value. The value of omega is not its equilibrium value. So, as a result this we are not actually at the equilibrium because both states have to be at the equilibrium value if we are not to have any transient.

So, by adding this plus 1 we have in fact given an initial condition which is not the equilibrium value of the states. So, we are going to get a transient. So, I have given you initial condition which is not equal to the equilibrium value if I have given you the equilibrium value, the system would simply not move. It would just remain where it is. So, if I apply Euler method, you have got delta is equal to delta of i is equal to delta of i minus 1 plus h. This is in fact simply omega minus, so we are just numerically integrating the system by Euler method. The omega I is equal to omega i minus 1 plus h times omega base divided by 2 times the inertial constant into p m minus k sin delta.

This is easy to evaluate if I know. What the initial values of delta and omega r, I can find out the new values since we are not at equilibrium. Since, I have given in initial condition which is not the equilibrium condition of course, if I made this 0, this 1 is 0.

We would be at equilibrium. So, what I will do is, I will just simulate this system. So, what I will do is again, so I am evaluating this system using Euler method.

(Refer Slide Time: 35:53)

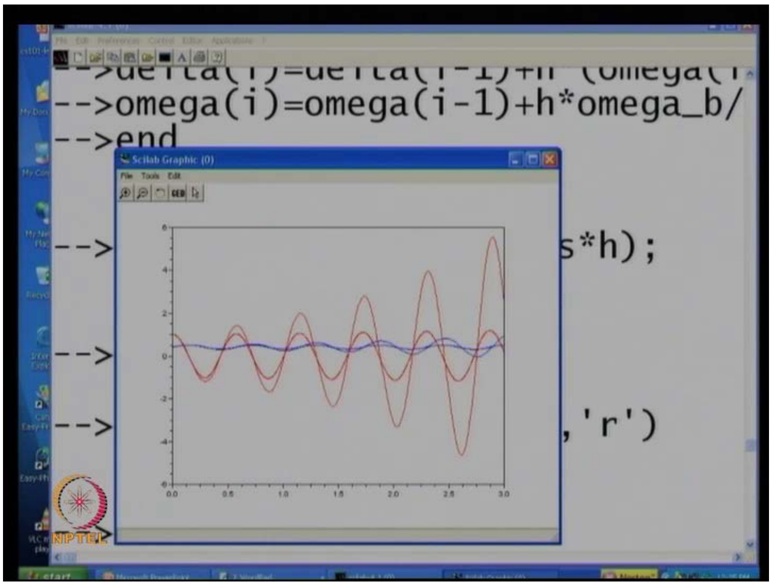


```
-->delta(i)=delta(i-1)+h*(omega(i
-->omega(i)=omega(i-1)+h*omega_b/
-->end
-->                              s*h);
-->
-->                              ,'r')
```

So, if you look at the behavior of a system, it looks as if it is an growing oscillation. In fact, if you look at, we have done the small disturbance stability of this particular system and we know that response if in fact not a growing oscillation but, a steady oscillation if those disturbance is small.

So obviously, there is a problem in the method. Let us just try reducing the time step. What is the problem in reducing the time step? There is no problem. You will have to wait longer for the simulation to get over in some time. That becomes a bit. That may become a limiting factor for doing any kind of study if you if you make the time step too small.

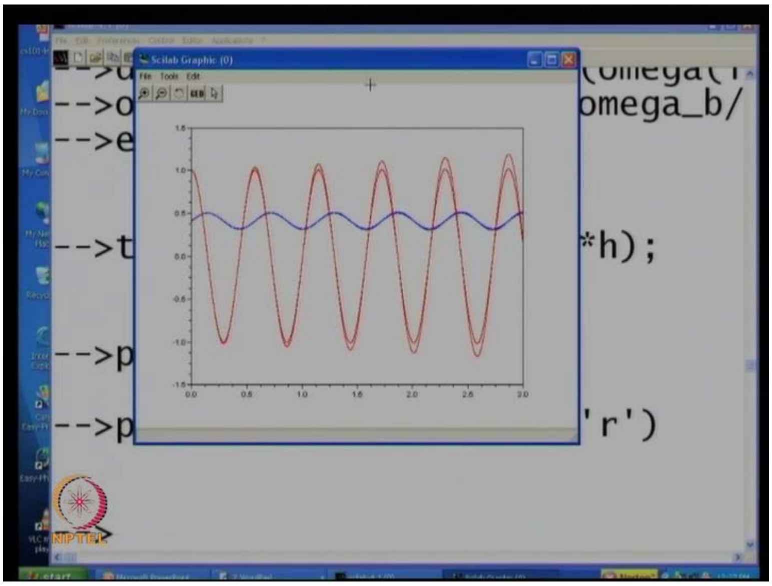
(Refer Slide Time: 36:35)



```
-->delta(i)=delta(i-1)+h*(omega(i-
-->omega(i)=omega(i-1)+h*omega_b/
-->end

-->                                         s*h);

-->

-->                                         ,'r')
```

Now, you are getting you know, a different answer that that shows that you have to actually reduce the time step.

(Refer Slide Time: 36:46)



```
-->o                                        omega_b/
-->e

-->t                                        *h);

-->p

-->p                                        'r')
```

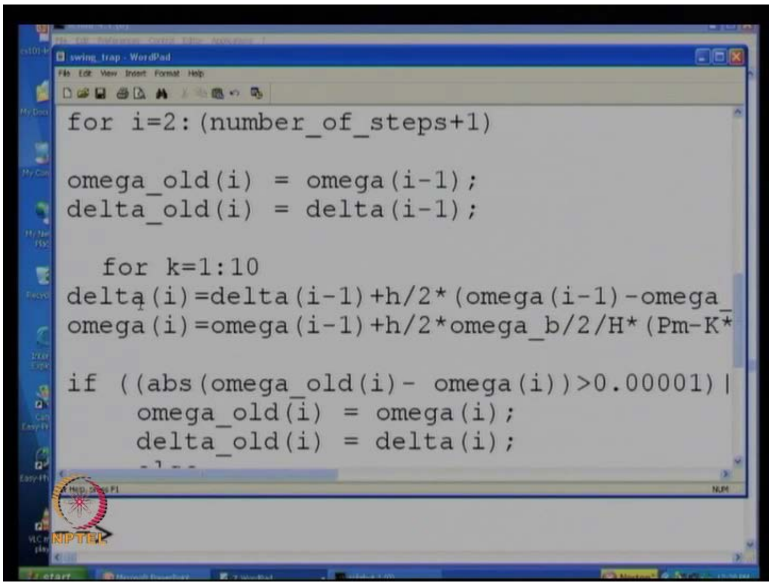
So, I will just redo this. In fact by using a smaller time constant time step, we are in fact getting a better answer. But, you see that this is still, it is not it is not you know, you this value is slightly less than the value you have here.

So what we see is that, if you have got you know, marginally dam systems; Euler method in fact does not seem to be very nice because whatever I reduce, if I reduce the time step

I can do that even further. Now, it is going to be painful because you will have to wait for a longer time. Now, it is still, it will take some time for this to come up. yeah So, you see that there is slight difference in the what I got with the previous time step and the time step you are having now. 1 One thing, we can simulate for longer and longer time with this time step that is 0. 2 0 1. But, the point you can actually try it out whatever time step you take, you will find this eventually starts growing. If you try to simulate beyond 3 seconds, say twenty seconds, even with this time step you will find that it is growing.

So, Euler, forward Euler method is not very nice to simulate systems with (( )) which are marginally damped or have poor damping. So, let us just rerun this system using trapezoidal rule. So I am using exactly the same system with a time step of 0.01 and 1 one problem comes here, now if I am going to try to do integrate this method using trapezoidal rule, what is going to be the problem? The problem which you see here of course, is that the statements in the program seem to have become much larger large here.

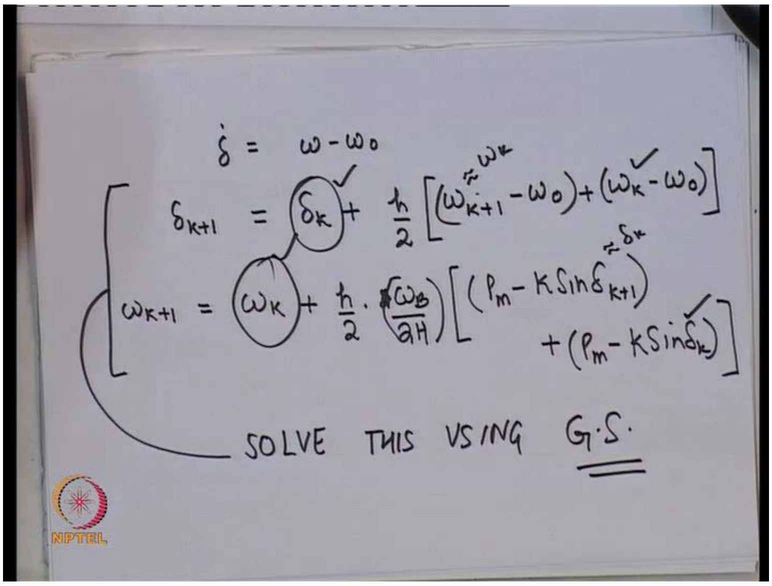
(Refer Slide Time: 38:46)



```
for i=2:(number_of_steps+1)

omega_old(i)  = omega(i-1);
delta_old(i)  = delta(i-1);

  for k=1:10
delta(i)=delta(i-1)+h/2*(omega(i-1)-omega_
omega(i)=omega(i-1)+h/2*omega_b/2/H*(Pm-K*

if ((abs(omega_old(i)- omega(i))>0.00001)|
     omega_old(i)  = omega(i);
     delta_old(i)  = delta(i);
```

One other thing you will notice is the delta. I will not be easily obtained from the previous value that is delta i minus 1. So, let us just pay our attention on what we get if we try to discretize swing equation using trapezoidal rule.

(Refer Slide Time: 39:07)



$$\dot{\delta} = \omega - \omega_0$$

$$\delta_{k+1} = \overset{\checkmark}{\delta_k} + \frac{h}{2}\left[\left(\overset{\approx \omega^k}{\omega_{k+1}} - \omega_0\right) + \left(\overset{\checkmark}{\omega_k} - \omega_0\right)\right]$$

$$\omega_{k+1} = \omega_k + \frac{h}{2}\cdot\left(\frac{\omega_B}{2H}\right)\left[\left(P_m - K\sin \overset{\approx \delta^k}{\delta_{k+1}}\right) + \left(P_m - K\sin \overset{\checkmark}{\delta_k}\right)\right]$$

SOLVE THIS USING G.S.

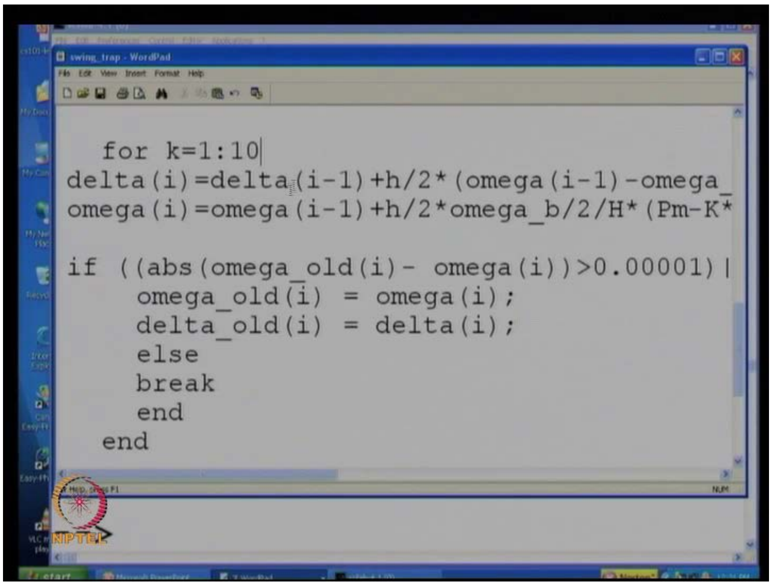
So you have got delta dot is equal to omega minus omega naught. So, we will have delta k plus 1 is equal to delta k plus h by 2 times omega k plus 1 minus omega naught plus omega k minus omega naught. So of course, we require the value of k plus 1 here. Omega k plus 1 will be equal to omega k plus h by 2 into omega b by 2 h this is the into p m minus k sin delta k plus 1 plus p m. we assuming of course, p m is a constant sin delta k.

So, this is the discretization which you will have to use and that is what is really been programmed there. Now, the problem here of course, is that you do not know the value of delta k plus 1, you do not know what omega k plus 1 is. But, you require that in order to evaluate delta k plus 1. So, if I want, if I given have given you delta k and omega k how do I get omega k plus 1 and delta l plus 1? You will have to use an iterative method. So that is what exactly I have done in the program. What I have done is, try to use within every time step within every time step solve this using Gauss Seidel method.

So, what we do is get? Plug in this value of delta k and omega k. You will plug it in here, here, here and here. You plug it in. Omega k plus 1 is something you do not know. So, let us guess that it is equal to omega k in the first instant and this is delta k. So, you run this. You you evaluate this assuming this is omega k and this is delta k. You will get a new value of delta k plus 1 and omega l plus 1. Pug in this here and you plug you plug in this here, you plug in this here and reevaluate delta k plus 1 and omega l plus 1. They basically, we are following Gauss Seidel method of solving this algebraic equations set of algebraic equations.

You go on doing this till, there is no difference between a previously evaluated del delta k plus 1 and the newest value of delta k plus 1 which you are getting. So, that is exactly what I am doing this here. The program is a bit complicated and probably you will find it difficult to follow this these steps but, I will just indicate what I am doing.

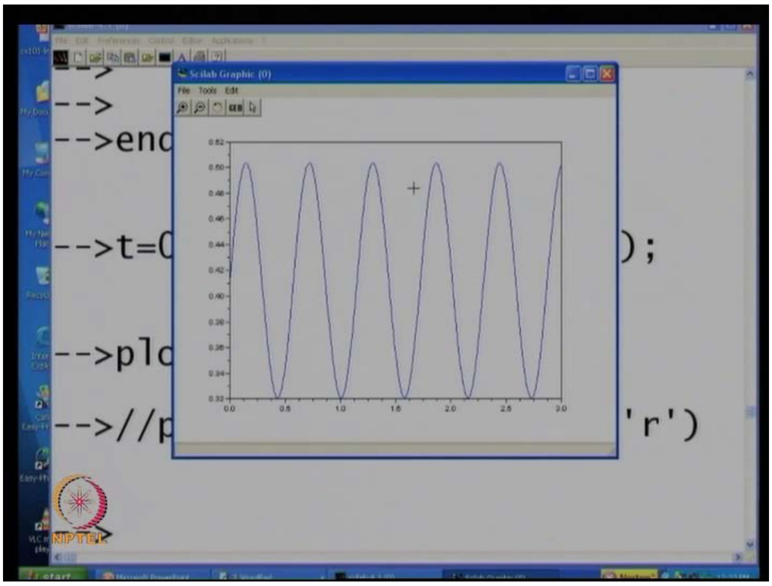
(Refer Slide Time: 42:15)



```
  for k=1:10
delta(i)=delta(i-1)+h/2*(omega(i-1)-omega_
omega(i)=omega(i-1)+h/2*omega_b/2/H*(Pm-K*

if ((abs(omega_old(i)- omega(i))>0.00001)|
     omega_old(i) = omega(i);
     delta_old(i) = delta(i);
     else
     break
     end
  end
```

So, for what I do here is, essentially in these steps. I am in fact using Gauss Seidel method. I have taken a maximum number of iterations per time step as 10 hope with a hope that of course, that will converge. So, in fact I am checking out the convergence criterion as well. So, let just me re go through the whole program slowly. This is the check on the convergence. So, I will just run this of course, one small point will give the same disturbance as before, in the sense that the this was one in the previous. So I have given a small disturbance. We are away from the equilibrium. Save this. yeah So, this is basically what you get. We actually evaluated just to, it this is not the what we get with trapezoidal rule.

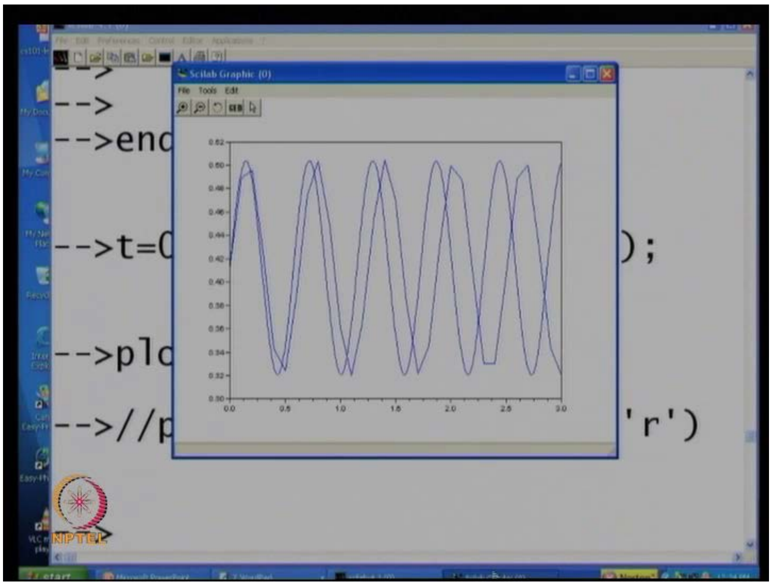
(Refer Slide Time: 44:17)



```
-->
-->
-->enc
-->t=(                                   );
-->plc
-->//p                                   'r')
```

Yeah Now, we will get what we should. So, this is the response and actually this is the sinusoid does not grow with time. So, this is a, this is a true. In fact, reasonably accurate picture of how delta varies if the system is given a small disturbance so even if I increase the time step remember, here this is this will not really be affected. So, I just changed my time step to 0.1 10 times more. Save this. Rerun it.

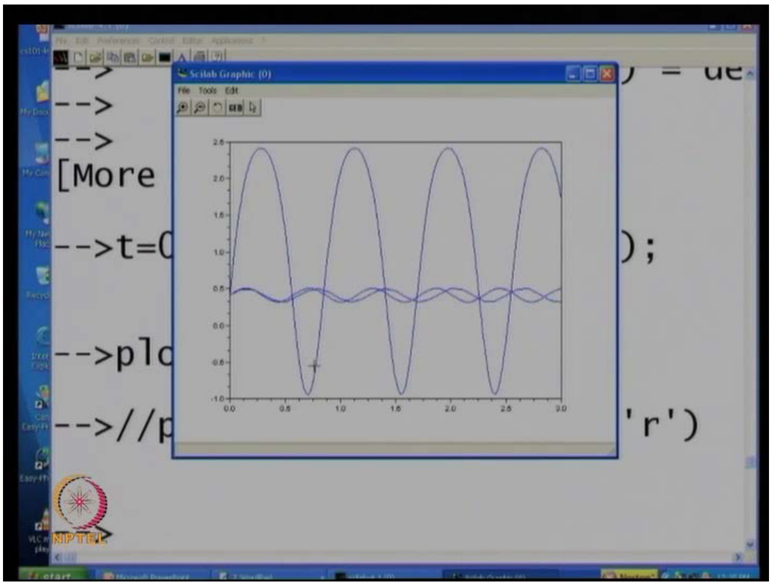
(Refer Slide Time: 44:48)



```
-->
-->
-->enc
-->t=(                                   );
-->plc
-->//p                                   'r')
```

Because of course, I have chosen a larger time step. There is slight you know, kind of poki poki solution here. This particular sinusoid has got rough edges but, on the whole it does not give you a wrong picture of stability. It is a undammed undamped sinusoid. Delta is an undammed undamped sinusoid. So, this is basically what you get with trapezoidal rule. If I increase the disturbance, I increase the disturbance I increase the disturbance in that case you even may go unstable. So, I am now, I am going to give a large disturbance relatively wrong.

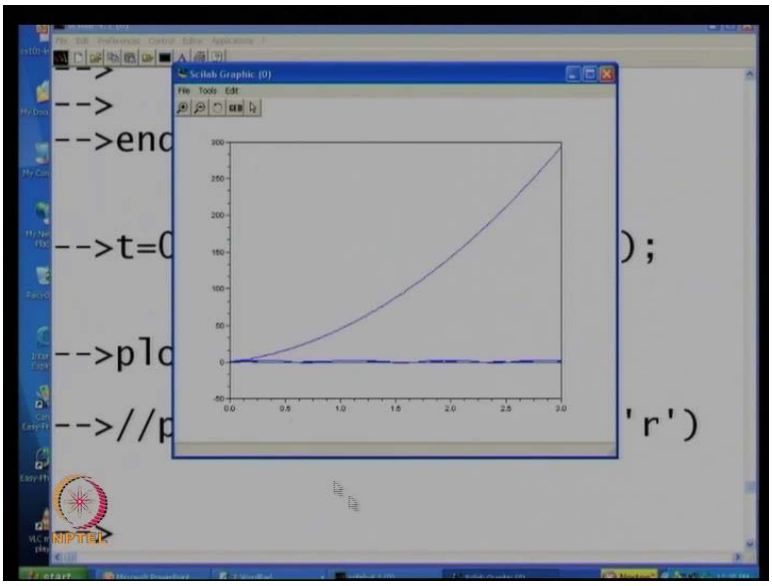
(Refer Slide Time: 45:41)



```
-->
-->
-->
[More
-->t=(                               );
-->plo
-->//p                               'r')
```

You see that, because of this larger disturbance, the delta oscillations are become larger. And one interesting thing that this is no longer sinusoidal looking you see this top part looks much fatter than the bottom part. So, obviously with the if you make the disturbance larger, your behavior starts moving into the non-linear zone. And you do not get the sinusoidal response as predicted by linear analysis.

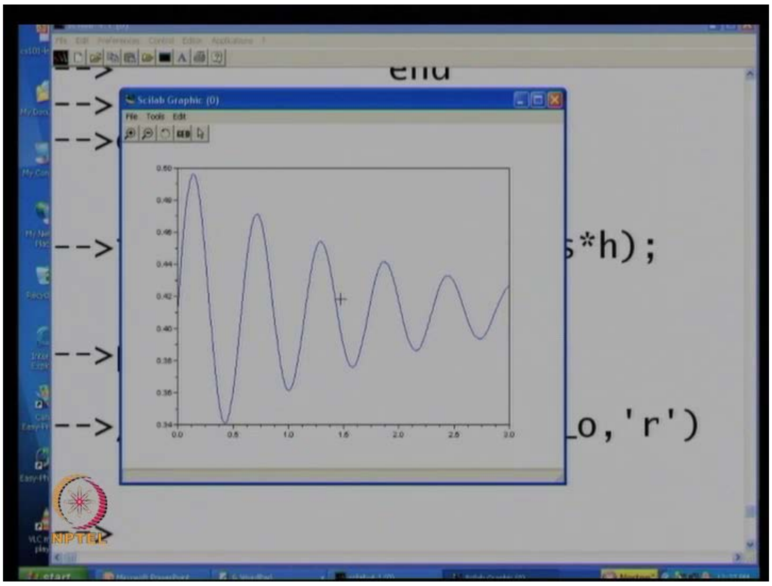
(Refer Slide Time: 46:22)



```
-->
-->
-->enc
-->t=(                          );
-->plo
-->//p                          'r')
```

So, if I actually make this disturbance even larger, you are going to get, the system looses synchronism. You see that, this angle just goes on increasing with time. So, if I go on increasing the, you know deviation from the equilibrium. We go into non-linear behavior and this is off course captured quite well by this numerical integration method. So, this is basically what I wish to tell you about using trapezoidal rule. For small disturbances it gives you the correct response. If I use backward Euler method for the same problem, Backward Euler method is also an implicit method. Remember, so I have to use an iterative solution to get this iterative method to get the values of the states at every time step.

(Refer Slide Time: 47:33)

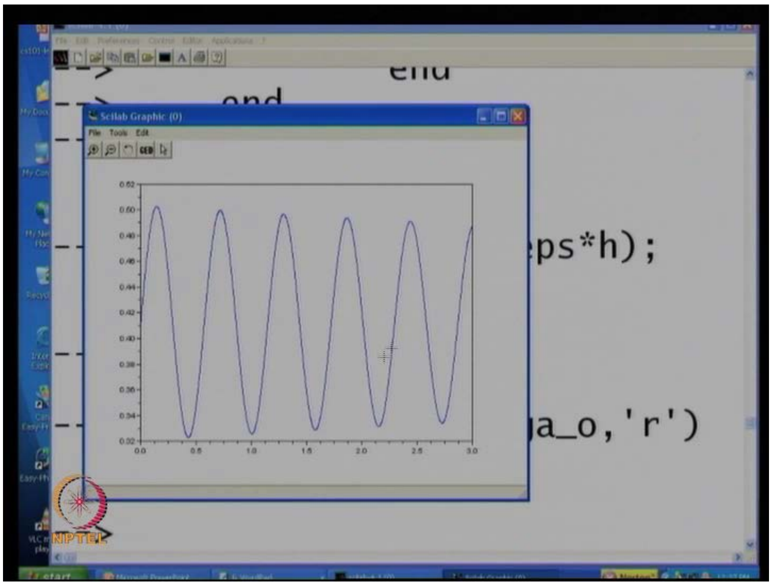


```
-->
   end
-->
-->
-->                s*h);
-->
-->                o,'r')
-->
```

So, let us just do this again with backward Euler. So, an interesting thing to observe here is that, when I use backward Euler on a marginally stable system which is just having 0 damping; Euler, backward Euler shows it as if it is a stable system as a damped oscillation. So, that is an interesting point about backward Euler. It really changes the picture. It s means especially your system is a marginally damped, it is got very little damping. So it is showing you the system to be a stable system whereas, actually it as undammed oscillation. So this system is a showing as if it is dying down.

So that is one interesting observation about backward Euler method. Of course, if I use backward Euler method with a very small time step, you are going to get reasonably accurate solution. So, I just numerically integrated with a small smaller time step.

(Refer Slide Time: 48:30)



```
-->         end
-->     end
-->
ps*h);
a_o,'r')
```

Well, we are coming, the answer is slightly different and we are coming closer to the solution we have in mind. So of course, reducing time step is 1 one of the ways you can get accurate solution or a more accurate solution. So, with this I end my discussion about the numerical integration of swing equations.

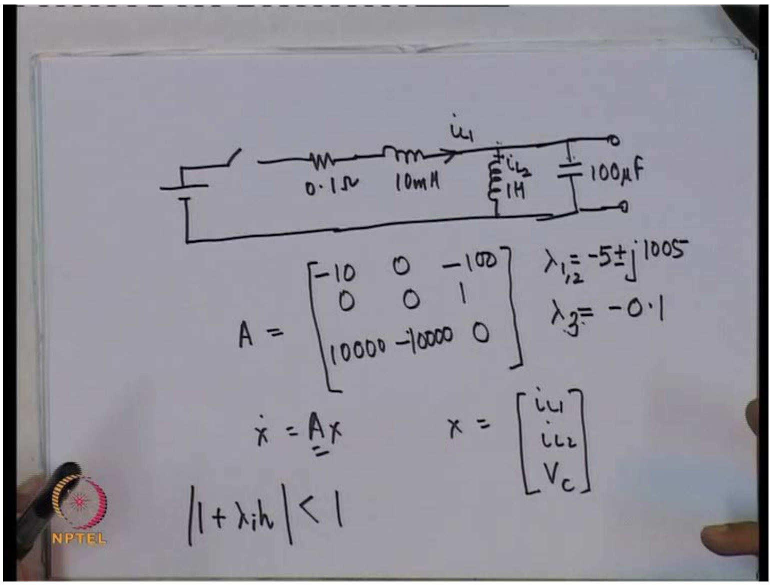
My discussion about numerical integration method would be quite incomplete if I do not consider what are known as stiff systems. We have learnt about stiff systems. Stiff systems are systems in which the Eigen values of the of linear system especially are far apart. In the sense that there were small if there is large Eigen value which translates to the facts that the systems has got both fast and slow transients. And if you are actually interested only in the slow transients, then you really need not modulate in great the fast transients in detail. In fact I showed to you in 1 one of the previous lectures that when you are trying to analyze the system with slow and fast transients the fast transient differential equations corresponding to the fast transients of the states associated with the fast transients, can be converted to algebraic equations and we really do not loose out much as far as the information on the slow transients is concerned.

A similar issue will kind of crops up when you have got even a numerically integrating stiff system. So if you got a stiff system, you got large and small Eigen values and the question comes, what is going to be the choice of time step? So, if you have got of a stiff system and you have got fast transients, you will have to chose a time step which is compatible with the fastest transient of the system in order to accurately get the response. So, if you got a system with slow as well as fast transients inherent then, you have to chose a time step to be compatible with the fastest transient so that, your fast transient are accurately obtained.

But consider situation where you are not really interested in the fast transients. So, how will you chose a time step? In Euler method, it is very clear that if 1 plus lambda h the absolute value is greater than 1 then, the numerical integration method is going to blow up. So even if your are fast transients are stable, if I chose my time step which is not compatible with this particular condition then; you your numerical integration as I said will blow up. But, what if I am interested only in the slow transients unfortunately with Euler method will be forced to take a time step which is small, very very small which is compatible with fast transient.

So, that is one of the problem. So, even if you interested in the slow transients, you have chose to choose very small time steps so that is 1 one of the problems which you will encounter when you try to use Euler method to integrate a stiff system.

(Refer Slide Time: 52:05)



$$A = \begin{bmatrix} -10 & 0 & -100 \\ 0 & 0 & 1 \\ 10000 & -10000 & 0 \end{bmatrix}$$

$\lambda_{1,2} = -5 \pm j1005$

$\lambda_3 = -0.1$

$\dot{x} = \underline{A}x \qquad x = \begin{bmatrix} i_{L1} \\ i_{L2} \\ V_c \end{bmatrix}$

$|1+\lambda_i h| < 1$

So, let us again look at a stiff system. We have done this example before. This is i l 2, this is i l 1, this is v c. We know that this particular system has both fast and slow transients. So this is 10 mille Henry, this is 1 Henry. In fact, the state associated with the slow transient is this and the states associated with the fast transient are these from the participation factors which we had evaluated in a couple of lectures before.

So, if you have got this particular system, this contains slower and faster transients. In fact how do you know that it contains slow and fast transient? affect In fact if you have, if you know the a matrix you have got X dot is equal to A X where X is equal to i l 1 i l 2 and v c then this a is equal to minus 10 0 minus hundred 0 0 1. We have done this in the previous class so, I am not reworking out this, minus 10000 and 0. This is a, the Eigen values of this system are approximately, minus 5 plus or minus j lambda 1 and lambda 2. So they are the two 2 actually complex conjugate pair.

So, this is a stable system but, one of the Eigen values is very small and the other one is very large. This is the example of a stiff system. If I try to numerically integrate the system stiff system using Euler method, I will have to chose my time step so that, 1 plus lambda 1 h is less than 1. So, h should be such such that this is satisfied. In fact, it should be satisfied for all the 3 1 2 and 3, all the 3 three Eigen values. So 1 one thing you will notice that, if you want to satisfy it for all the 3 three Eigen values then, h will have to be very small. In fact, it becomes completely unstable in case you chose your h say as 0.0 0 1 or so.

So, I will just do this numerical integration using Euler method and then we will conclude this particular lecture. So, let us do this 1 one integration. This is a system; this is a matrix I have chosen a very small time step. I will just simulate it for a short while because, I know it is really not going to work out with Euler method. Even with this small time step, even with this small time step we have a problem with Euler method. So, the first thing I will do is integrate it using Euler method. This is a remember, a higher order system. So, instead of 1 plus small case a H you have got identity matrix plus A into H into X plus H into u.

So, the I am implementing Euler method for this system. Of course, there is an input here. So, I have used Euler method, I have generalized it to a situation where you have got an input is as well. So, if I numerically integrate this system; hoops Oops! There is an error here. So, I will just check out what is the problem and we will redo it. The problem probably is, there is some error here. So, we will what we will do is, it do is redo this example again in the next lecture, will just debug the error which is there in the program and rerun this example in the next class.

So, in next class we will continue with this particular example, see how it is in fact a very interesting and important thing. We are going to learn in the next lecture that, for stiff systems of this kind; you have to use specific numerical method. Even you know, explore if you want to capture. In fact, both the fast and slow transients you may even have to explore things like variable time steps and so on. So, let us conclude this lecture at this point. We will redo this example which did not work out in this particular lecture again in the next lecture.