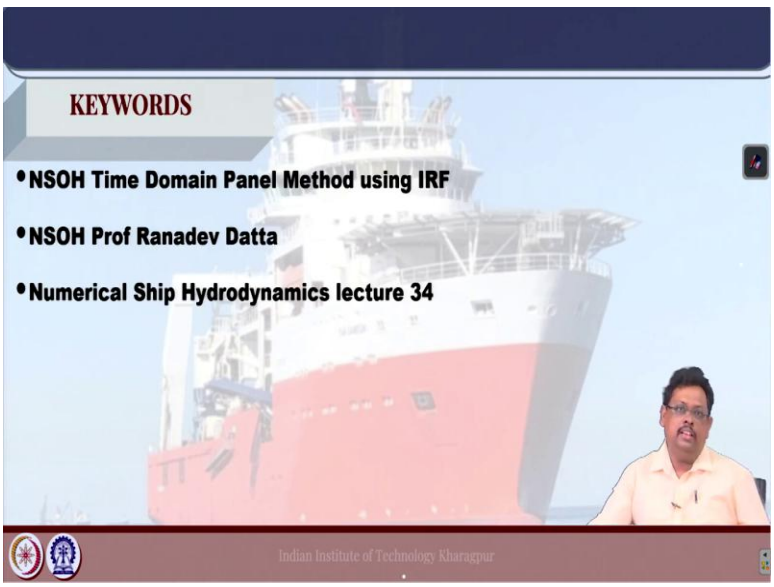
Numerical Ship and Offshore Hydrodynamics
Prof. Ranadev Datta
Department of Ocean Engineering and Naval Architecture
Indian Institute of Technology, Kharagpur

Lecture - 34
Numerical computation of IRF based method

(Refer Slide Time: 00:23)

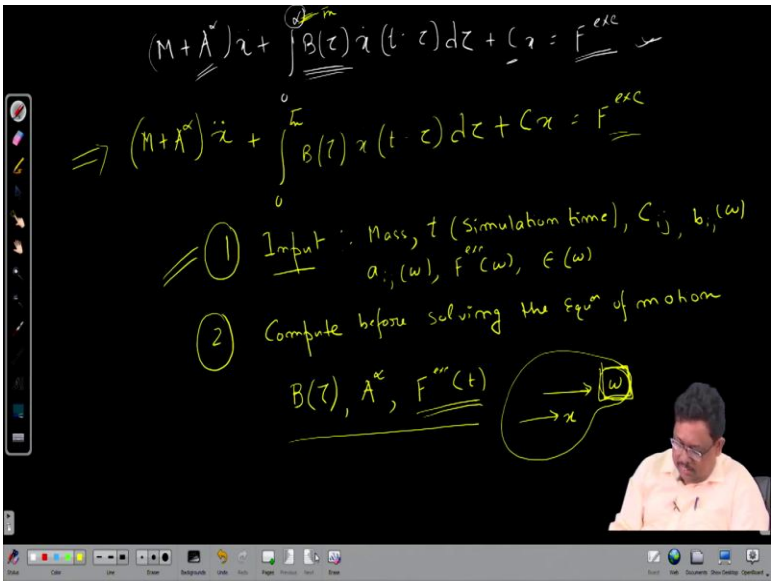
(Refer Slide Time: 00:31)



Hello, welcome to Numerical Ship and Offshore Hydrodynamics, today we have the Lecture-34. Today we are going to discuss the how we can solve numerically the IRF

Based Solution ok and this is the keyword that you have to use to get this lecture ok. So, without further delay let us start the main part. Well, now if you remember in my previous classes, we have discussed ok.

(Refer Slide Time: 00:49)



Let me always first let me write this equation of motion which is $\left(M + A^{\infty}\right)\ddot{x} + \int_0^{\infty} B(\tau)x(t-\tau)d\tau + Cx = F^{exc}$. Now, in this equation we have discussed how we can get this exciting force. We have discussed how we can get this C, we can we discussed that I mean how we can get this $B(\tau)$ and also the added mass infinity.

So, now we know that this parameter that is required even before we start our writing the main code right. So, here we can further modify this equation, because this limit is not infinity right. This limit it should be in some finite value and we know that finite value is τ_n right. So, therefore, we can rewrite this equation as $\left(M + A^{\infty}\right)\ddot{x} + \int_0^{\tau_n} B(\tau)x(t-\tau)d\tau + Cx = F^{exc}$.

So, now how we can do that, right. Now, today we are going to one by one now we have this whole thing with us together, now today actually step by step we try to build up the flow chart. And then we need to see that what is not much critical thing anyway here,

because this the most complex this added mass damping term, we are getting from this frequency domain solver.

But here still we have some kind of numerical instability possible here also, that we have already discussed in my previous class that how to deal with at least for the exciting force ok. Now, let us see, how we can build up the flow chart?

Now, here if you look at this equation we have three parts, the first part is something is required even before start the time domain computation right. So, what the thing is required initially or you can take that is my input. Of course, my input would be the mass and then my input would be that what would be my simulation time. So, you can call it t or you can say it is a simulation time and of course, we need to give you can give as a input because here we really do not need the shift geometry as such. So, we can use this input this c the coefficient C_{ij} let us say that also we can use as the input.

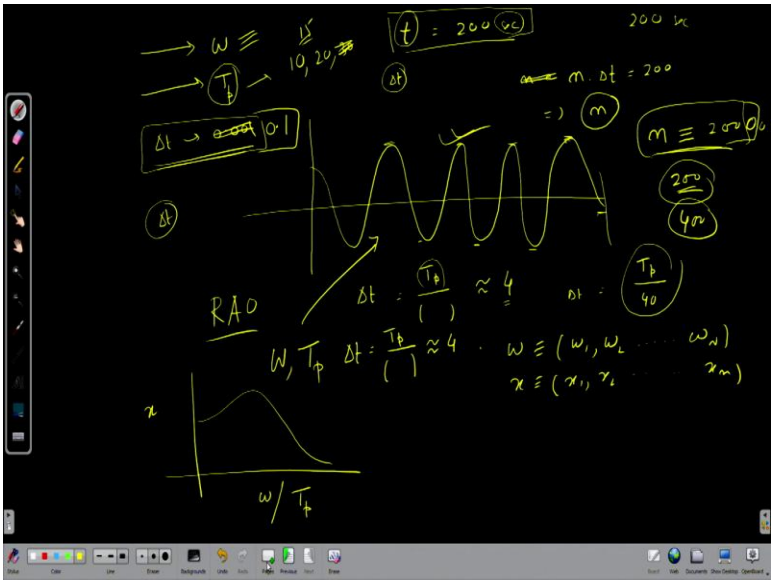
And, what more there would be input here? Ok, let us go with this if something more is required then we can add here, second thing is something that we need to compute before the equation we solve the equation of motion compute before that solving the equation of motion solving the equation of motion; that means, before we start the time matching algorithm ok.

Of course, ok now I remember this is of course, this b_{ij} in frequency domain that is our input. Then a_{ij} in the frequency domain that is our input and then $F(\omega)$ of course, that exciting force that is our input and also the phase that is ours. So, these all are my inputs right it can be more let us see at this point as I remember all these things ok.

Now what you need to compute before this we start the equation of motion is we need to have we need to compute first the $B(\tau)$ right. And then after this we need to compute my A^{∞} and then we need to find out my F^{exc} and which is this prescribed simulating time t. So, all this data is required now here in this equation of motion then everything is that which I required before even start it is here. And then we need to the idea of this I mean this code why we are writing because we need to find out if I gave some wave of frequency ω.

So, what I need is what would be my the corresponding my response that x. So, this is actually it is the main input that is going to give I mean by the user that at which ω I try to find out the response of the vessel. So, definitely this is something that we want from the user apart from this this definitely these are the input to run the code, but once this is ready so the biggest input is that at which frequency, I am going to find out my response of the vessel right.

(Refer Slide Time: 07:37)



Now, the question is there are major question is. So, ω is given to me right and then to run this this equation of motion, let us say I am running for the time step t. So, I give $t = 200$ second. So, I said that I need to run the code for 200 second, now the question arises here that, what would be the Δt?.

That means that how I do the increment right. Because with this respect to this I need to find out the n the parameter because that is important to run our software, is it not? I need because a computer coding you have to give some i =1to 2 the total stress. So, n should be. So, therefore, I need to compute my $n.\Delta t = 200$ seconds. So, from there I need to calculate that what would be my n, fine.

We understand. So, what I am trying to say is this is given to me that at which frequency I am going to do the evaluation or may be at which you know time or which time period T_p ok, I need to find out my response of the vessel ok. Now, here at which time step I

need to calculate it ok. So, here we know that we give that after 200 second I am going to give it.

Then how I set my Δt, because you know you can take random let us say $\Delta t = 0.001$ you can make a fixed value of Δt right. So, then you can see that you're a would be very large in fact, ok and then it might take some not necessarily ok. Normally, the way we represent the delta t actually at least the way I do that in our coding so, I make sure that how many cycles I required.

For example; I need let us say in 200 second or let us say that the time period is you know like after 200 second then I need let us say I need some 1 2 3 some 4 cycle I need. So, with this I need some kind of the 4 cycle I need ok. So normally what I do in our code, I really do not give the second, what I give is that what would be actually directly I am asking the n ok.

Now, suppose the user gives me this it is 200. So, then I need to decide that I need to fix my Δt to make sure that I must have 1 2 3 4 peaks when actually I have $n = 200$. So, what I did normally that, this is the time period right. So, I fix my Δt I divided this time period by number of some integer. So, that this gives approximately 1 2 3 4 that number of cycles that I need ok.

So, that means, I need to divide my time period in that way. So, that when it comes around 200, I can get roughly the 4-cycle let us see in that case if I fix my Δt equal to time period let us say divided by 40 then when actually n become 200 actually, I can get you know 5 cycles. So, I make sure that. So, in my case the Δt is not a fixed time 0.001 it is based on the fact that how many cycles I need.

So, maybe I need some 10 so I have a control on this. So, here what is happening now what I am going to do with this code finally, I need to get the RAO, what is the RAO? RAO is nothing, but it is the x versus you know ω right or in some way x versus the time period T p. So, then I give that let us say list of ω.

So, I am giving some $\omega_1, \omega_2 \ldots\ldots\ldots\ldots \omega_n$ and I try to calculate what is the corresponding response which is x 1, x 2 and then x n and then I need to plot this graph right. Now, thing is that here when we plot this graph so, I need to give this code so many value of omega

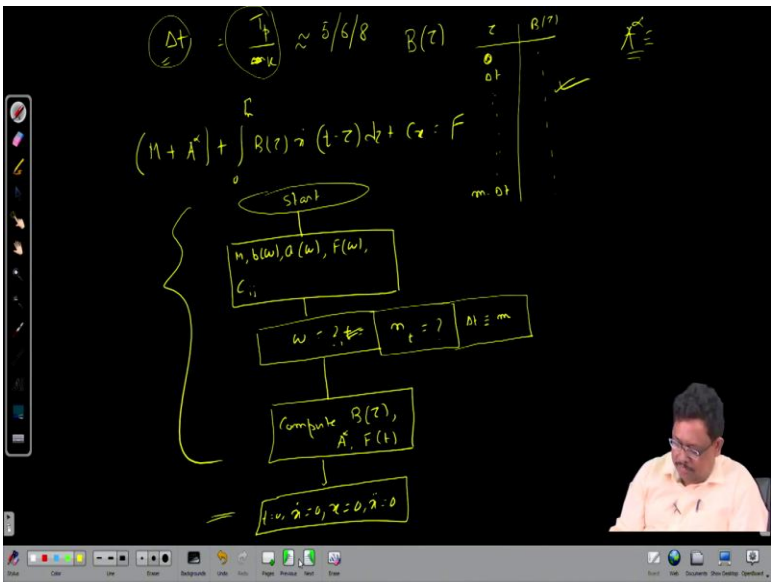
and in now here if I make it is delta t equal to 0.001 and if I fix not the second, if I fix the number of time step n is my input then you can see that then this become really small because if I take 200 and it is 0.001. So, roughly you know you do not get much.

Let us even if I take 2000 still it is or I make sure even 20000. So, you see it is very difficult even if you make it 0.11 let us say to be precise is fairly large ok of course. Then also even if you make some 2000 then only you can get at least for this you have 200 times there. I mean 200 seconds something like this, but then if your time.

Now, you can see if your time period something around 10 second, sometimes 20 second, sometimes 30, not 30 is very high, may be some 15 second intermediate, now when you do that then this number of cycles this one will change right. But what normally that what I want in my code what irrespective of your ω or irrespective of time period I should get the same number of cycles with the stipulated time or you make 200 or 400.

If I make 400, I can get 8 cycles. So, that I make sure so, in that way I fix my Δt equal to that the time period divided by the factor and that will give me the desired cycle that I want sometime it is 4 or 5 whatever ok.

(Refer Slide Time: 15:17)



So, now I know that how to I fix my Δt. So, here the thing is that after getting all this input I fixed my $\Delta t = \frac{T_p}{k} \approx 5/6/8$ whatever ok. So, this is how normally we do and also, we need to make sure this with this Δt at least you can get the result little bit you know convergent, it should not be very large ok. So, then we might not get very good results.

So, that also we need to make sure, but all this comes with the experience when you work this IRF code for 2 3 I mean many times then you have got some idea how you can fix your Δt ok.

So, now, once I fix the Δt so, then when you compute this $B(\tau)$ which is again the list of here you have the τ and if the value of $B(\tau)$. So, we are using the same Δt it is easy to you can use different, but normally we do not we do not use this different Δt or use the same Δt and you can get this corresponding $B(\tau)$ right. Fine and then you know that A^{∞} is a constant term. So, you do not have to bother much about this. So, now, here we do very important thing how to fix Δt and then with respect to that Δt only we can get the value for β.

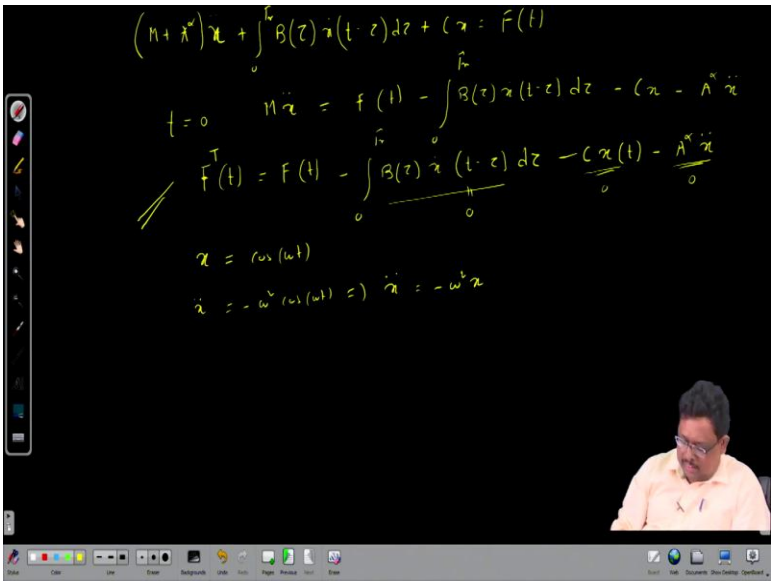
So, now if I write this code the flow charts this flow chart is now, I just write again $\left(M + A^{\infty}\right)\ddot{x} + \int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau + Cx = F$ so, when I writing this, so now we how I set the flow chart. So, now, I just start this code and then initially we need to get the data which is M the mass, then the added mass damping then F then this one and then your C_{ij} all these things we need to get from your input right.

And then I need a input about at which ω we need to calculate the response and then at till which actually I give the which number of time step I need right. So, not t normally what I get this n_t that what is that the total number of time step we need to give. And also, I give I ask another input that what about the Δt; that means, that how many I mean that time period I need to split in how many parts ok. So, it is basically the m. So, this is the three thing that we are going to ask ok.

And then after having this we can compute this $B(\tau)$ and A^{∞} ok and ok. So, yeah then I am done and all also this $F(t)$ right. So, these three things actually I can get once I have.

So, this is the initial part of the code right and then we start the time matching algorithm right.

(Refer Slide Time: 19:44)



So, now here ok. So, always I just write this equation in top, so that you can understand that what actually I am doing $\left(M + A^{\infty}\right)\ddot{x} + \int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau + Cx = F(t)$. So, now, the time matching algorithm now we need to start right. So, how I do that?

First, we need to calculate this I need to arrange this way $M\ddot{x} = F(t) - \int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau - Cx - A^{\infty}\ddot{x}$. So, then we need to compute initially my we can call this total force $F^T(t) = F(t)$. So, now, actually I start my code let us say from $t = 0$ ok.

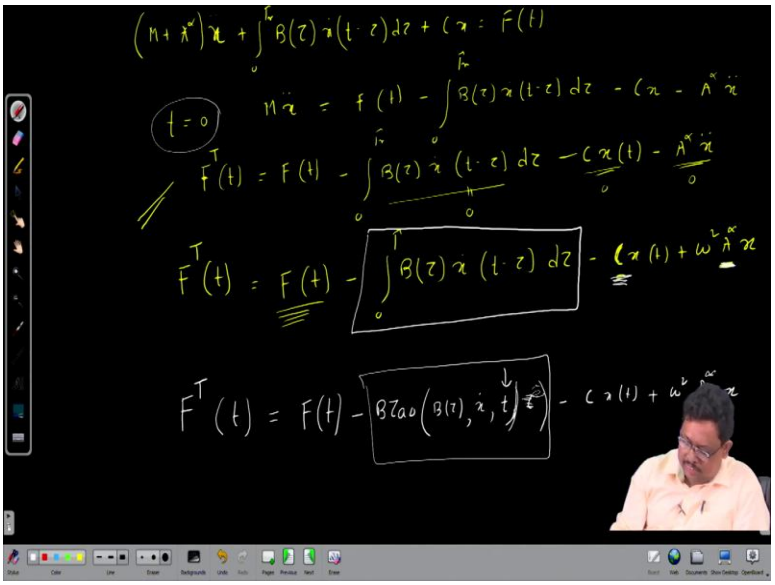
And then I need to calculate the whole thing. So, $F^T(t) = F(t) - \int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau - Cx(t) - A^{\infty}\ddot{x}$. Now, here I need again that I missed here that initialization of the whole thing. So, here we need to initialize also that the next step that at $t = 0$ you have $\dot{x} = 0$ to $x = 0$.

These two things are enough ok and you do not need actually; however, you can make the acceleration also 0 at $t = 0$ this also you need to define before we start the time

matching algorithm ok. So, now, here I have this thing. So, at $t = 0$ you understand that this will gives me no continuity it would be 0 and this should be 0 and this should be 0.

Now, here I need to do once a small modification that actually we really do not compute $\ddot{x}$ why because actually since it is under the linearity. So, if you assume let us say $x = \cos(\omega t)$ and then definitely you can get $\ddot{x} = -\omega^2 \cos(\omega t)$. So, you can assume $\ddot{x} = -\omega^2 x$.

(Refer Slide Time: 23:24)

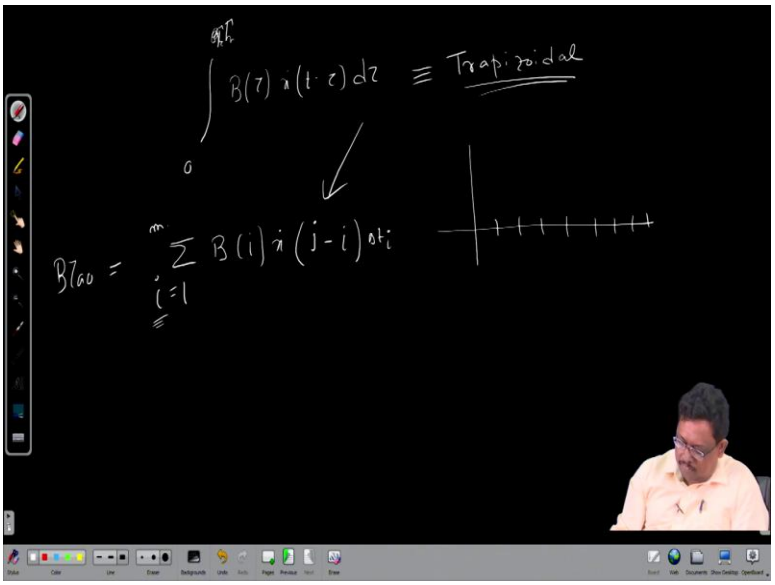


So, that I understand right. So, I use this property I use this property and then what I do is I rewrite this again this $F^T(t) = F(t) - \int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau - Cx(t) + \omega^2 A^\infty x$. So, this modification I do. Now, here you see here I mean you do not, you already calculated this it is with you, you know what is your C. So, you do not have to worry about this, you have calculated this part also beforehand.

Now, only the one thing that actually you have to dynamically each time you need to calculate is this tau. See, this is already you have already stored it in your memory computer memory, this is your geometric property and that is your input. So, you know that this also you have computed before we start the time matching algorithm at t equal to 0.

The only thing you did not do before this one. So, this actually you need to calculate dynamically. So, then actually if I write a MATLAB code. So, probably I can write at this moment $F^T(t) = F(t) - B\tau_{a0}(B(\tau), \dot{x}, t, \tau) - Cx(t) + \omega^2 A^\infty x$. So, this is the thing. So, only thing that we need to call the function at each time step.

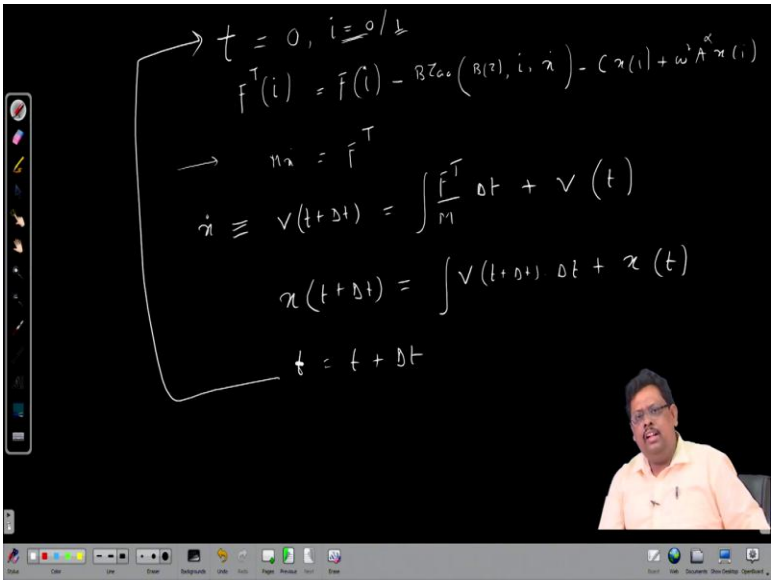
(Refer Slide Time: 26:07)



Now, how I do that? So, now, it is so trivial that you can do that now suppose here this is basically $\int_0^{\tau_n} B(\tau)\dot{x}(t-\tau)d\tau = Trapezoidal$ ok. So, when I show you the code that time we can go there. So, it is very easy. So, you have this, the graph which is the $B(\tau)$ and then the velocity this is the term and so therefore, at each time step now this you can write in the summation form

So, $\sum_{i=1}^{m} B(i)\dot{x}(j-i)\Delta t_i$ this is gives you the value of your $B(\tau)$ right. So, I know that how do we calculate, this is so trivial like a trapezoidal will do. So, we are using non by the trapezoidal rule to do that right fine.

(Refer Slide Time: 28:08)



So, now let me again finally, rearrange everything. So, at $t = 0$ first we calculate the total time step F T i let us say it is; that means, that means, let us say i equal to 0 or i equal to 1 if you are using the C code or MATLAB code, if C it is i starts with 0, if the MATLAB i starts with 1 anything.

So, $F^T(i)$ which is nothing, but $F^T(i) = F(t) - B\tau_{a0}(B(\tau), i, \dot{x}) - Cx(i) + \omega^2 A^{\infty} x(i)$. So, this is your total.

So, then in the next time step you have to solve this equation $M\ddot{x} = F^T$ right and this actually we can do very easily that Euler integration scheme if you remember we have solved in your in the first in the first lectures actually we solved this equation right. So, I know very well that you have to do this $V(t+\Delta t)$ where V is nothing but your $\dot{x}(t+\Delta t)$ is nothing but $\int \frac{F^T}{M}\Delta t + V(t)$.

This is how we can do this and once I get this. So, then I have do that $x(t+\Delta t) = \int v(t+\Delta t).\Delta t + x(t)$ and then I increase my $t = t + \Delta t$ and again I go back here right. So, this is how actually we write the code ok. So, now, I think after all this discussion probably now here you people can write this code neatly.

Because I discuss each aspect of this code what is the what the problem we can face, how to overcome this problem and how to write this time matching algorithm which is really easy, once you have all this other thing then time matching algorithm is really easy and then after doing all this exercise you should be able to get the equation of motion of course, there is lot of simplification is involved over here, but at the end mostly we can get very realistic result out of it ok.

So, today with this let us finish this discussion on equation motion using impulse response function. Remember that this code is written for zero speed for the forward speed you have some additional term that we are not discussing over here, but if you are interested always, we are happy to help you how to include the forward speed dependent term in this IRF base code ok.

So, thank you very much.