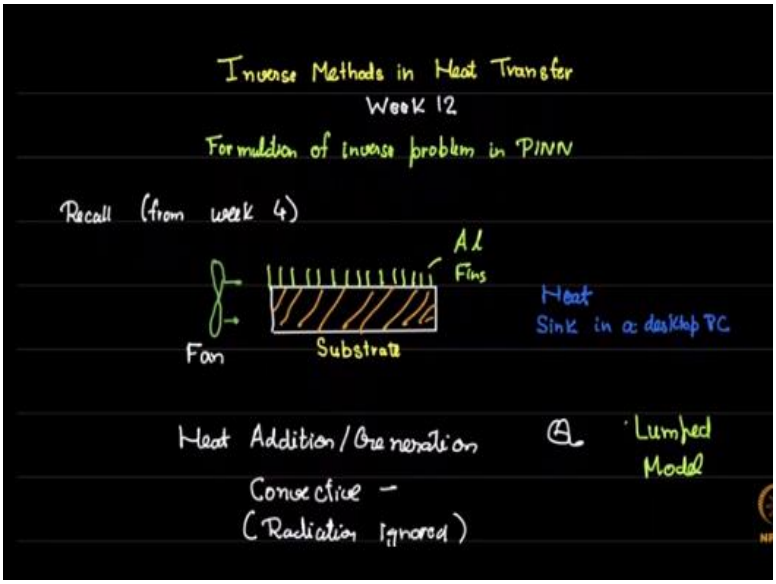
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology-Madras

Lecture - 65
Formulation of a PINN Based Inverse Problem in Unsteady Conduction

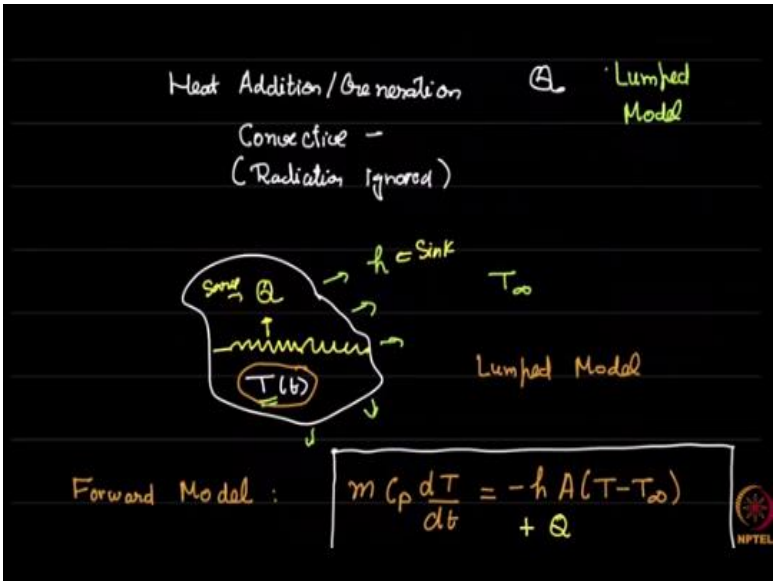
(Refer Slide Time: 00:19)



Welcome back. We are in week 12 of inverse methods in heat transfer. In the previous few videos, you had seen first some utilizations of neural networks for classification. Then we saw neural networks being applied to physics informed neural networks. But that was a forward problem. Now what I want to show you in this video is simply a formulation for an inverse problem in PINN. I am not going to show you the full solution.

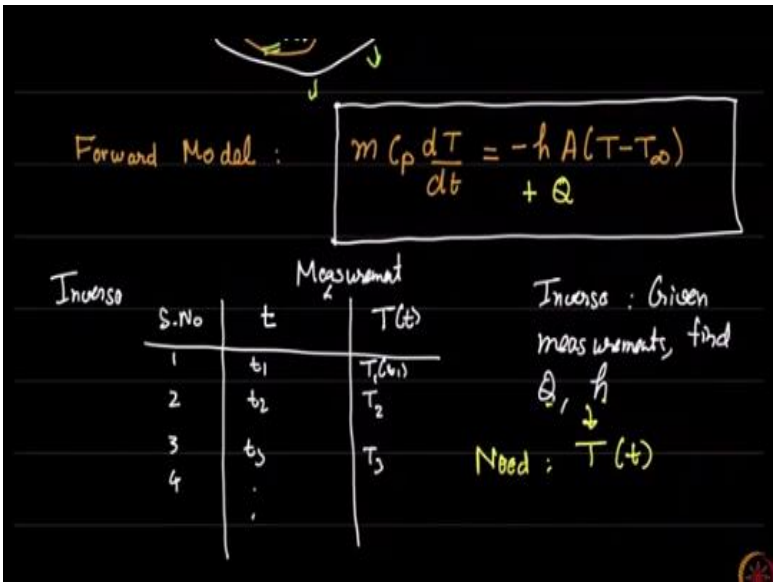
I am just going to show you the formulation. I had already talked about this the last week. But I want to show you with one specific network, just like we did with the XOR gate case, just to get some intuition for what happens. The computation and the coding are a little bit messy. If I use a framework, like we did with the Burgers equation code, you simply see the framework and you do not understand too much of what goes within. So, this is somewhere in between. This is not explicitly a code, but this is just the formulation.

(Refer Slide Time: 01:29)



So just for an example, I am going to take an example that we took in week 4 of something like heat generation and addition, which is basically convective. So, remember we had this case, where we had some body, we have some heater kept in there and it is losing some heat to the surroundings, which is like the sink.

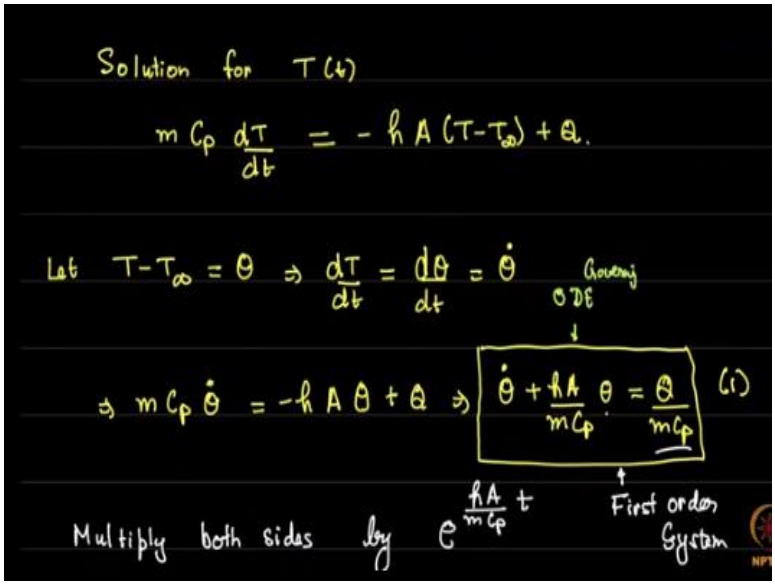
(Refer Slide Time: 01:46)



And we used the lumped-capacitance model here and this was an unsteady problem. So, we actually to an unsteady problem. Of course, again, this is an idealization and a very simplification. And in fact, we had some exercise problems where at various time steps, we measured the temperature and we wanted to find out these parameters, what is Q, what is $\frac{hA}{mC_p}$ etc., okay?

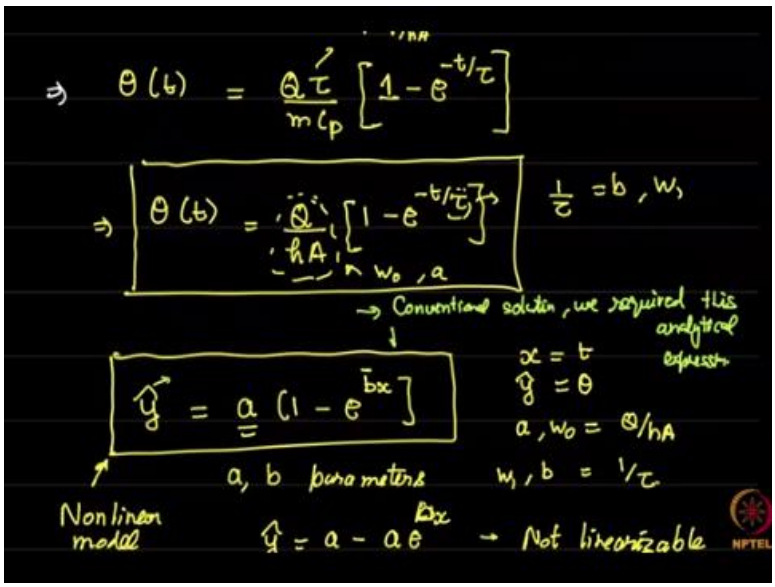
So given the measurements find Q or h for which we of course require the temperature. So, I had, let us assume that this data is given. You are given this measurement of temperature and time, okay?

(Refer Slide Time: 02:25)



Now when we solved it, the way we worked at it was this was the governing differential equation. So, this was the ODE which govern the process. So, this basically is what we mean by physics of the problem. And this physics itself, we will use this data in order to solve the problem.

(Refer Slide Time: 02:55)

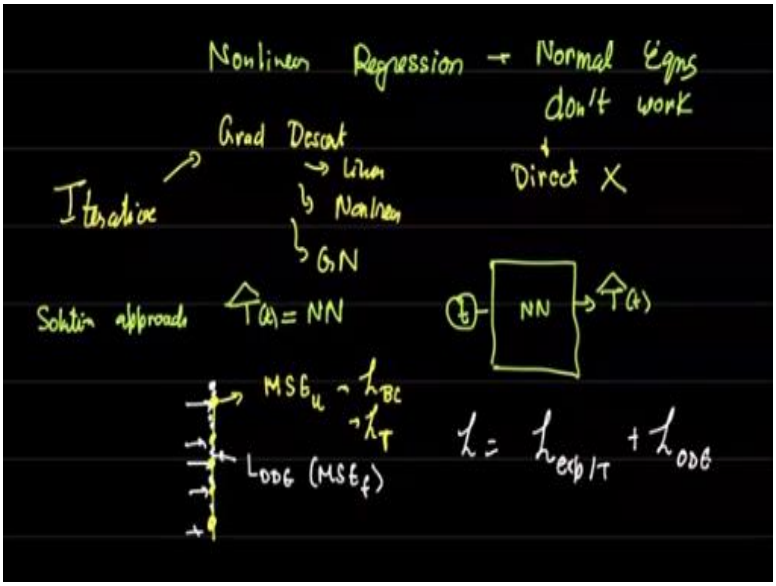


Now recollect that for the conventional solution for the problem, we required this analytical expression. So, this is what we required in order to solve the forward model PINN, so the forward model based inverse problem. So, this of course came from the

physics but we actually had a solution. And where did the solution come from, because we had a solution to this ODE.

Now imagine, we do not know how to solve this ODE, which would be the case let us say we have something like Burger's equation, which we just saw or we have a Navier-Stokes equation. So, the question is, can we solve the inverse problem without explicitly solving the ODE? So that is the basic question. And the answer of course, is we can solve.

(Refer Slide Time: 04:12)



So, the solution method is once again assumed that $\hat{T}$ is a neural network, some neural network. So, it takes in in this case just time, because $\hat{T}$ is a function of t, we just take in time. Just like the previous case, we put in some neural network here and we get $\hat{T}$. So, what we are doing is instead of using this expression, we are simply saying $\hat{y}$ is a neural network of x so to speak.

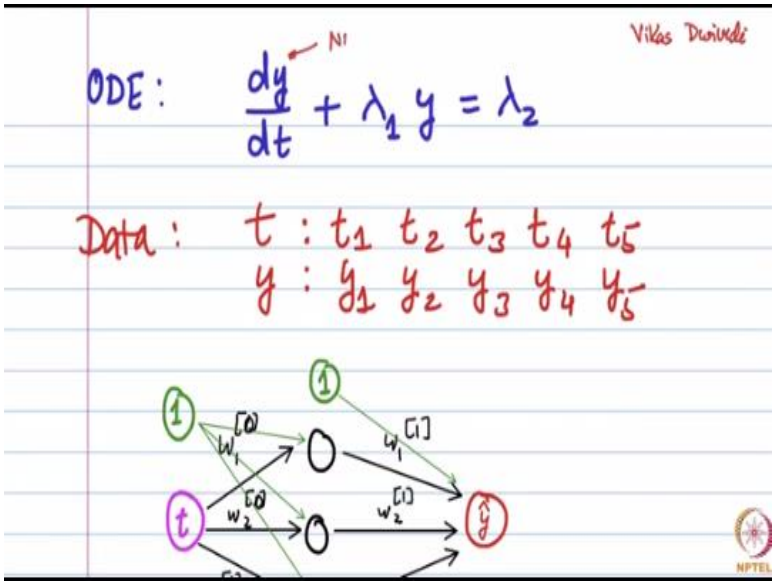
We are using theta of course, instead of t, it does not matter. We use t sorry $\hat{T}$ is a function of t. Now we have let us say in time we make these measurements. So let us say this is the time axis and we make measurements at 5, 6 points or 100 points, it does not matter. So, these yellow points now are where we look at our least square function in terms of what we called MSE_u or I called loss of the BC or the loss of given functions, okay?

So, function value is given here L_T whereas, we can also put a lot of other points in the middle. At these points we satisfy the ODE. What was called in the previous code as MSE_f. So, the total loss is going to be loss due to the experimental or the BC or t plus loss due to ODE which is at arbitrary points. So, these points, the yellow points are fixed by where we take these measurements.

Whereas the white points are free to do any value in the middle, okay? Now when we said the ODE, of course the error is given by this equation, the governing equation. So, this governing equation I am going to write as if this is some $\dot{y}$ plus some constant equal to some other constant λ_2, okay? So, our inverse problem becomes suppose I measure some 6, 7 values here, can you find out λ_1 and λ_2.

Now notice the difficulty of the problem. We drew we are not even giving you λ_1 and λ_2, but just based on these measurements, you want to find out λ_1 and λ_2 as well as find out y basically in the middle. And that is where the genius of the Raissi and Karniadakis approach is.

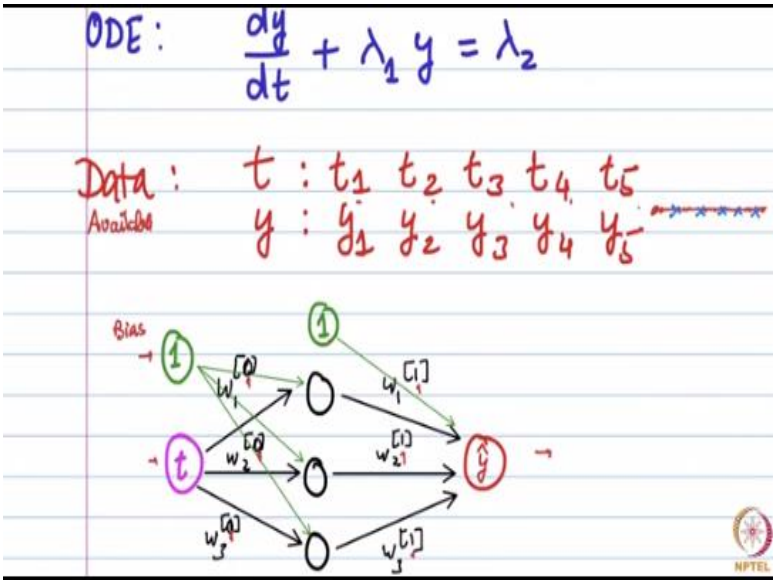
(Refer Slide Time: 07:13)



So just to show you what expressions look like, all we are doing here of course is assuming now that y is a neural network. So proper acknowledgement these nice notes were written by one of the authors of the paper that I had shown you, one of my ex-PhD students, Vikas Dwivedi has written these nice notes. And you I would welcome you to read some of his papers in order to understand the pielm method, which is somewhat simplified method.

But we are not covering pielm here. We are simply looking at some sort of PINN in order to solve this inverse problem. So, this y now here is a neural network. What neural network, that is given here.

(Refer Slide Time: 07:59)

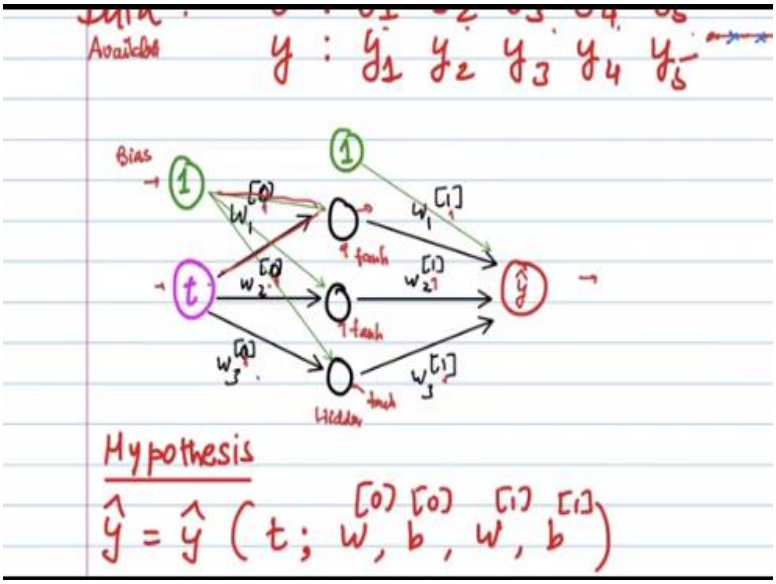


So, we have chosen to show you an example, where it is a simple neural network with just one input as t. This of course is the bias unit. You have the hidden layer here with three neurons. We have taken an extra bias here too, just to make it look reasonably um complicated. So Vikas has chosen to label this as 0. I would have chosen this as 1 and 2, but he has chosen to label it 0 and 1.

Now let us say this data is available at some 5 points t_1, t_2, t_3, t_4, t_5 like I said, we can draw it let me draw it in a horizontal line. So let us say it is available at these five points t_1, t_2, t_3, t_4, t_5 and in the middle, where I have given nothing okay, so that is where we impose, we can actually impose anything we want using the ODE, the governing ODE.

So, at all these other points, these will be ODE points where we can choose as many as we want. For example, we saw on the Burgers example that we had taken 30,000. There is nothing that stops us from taking as many ODE points as we want, okay.

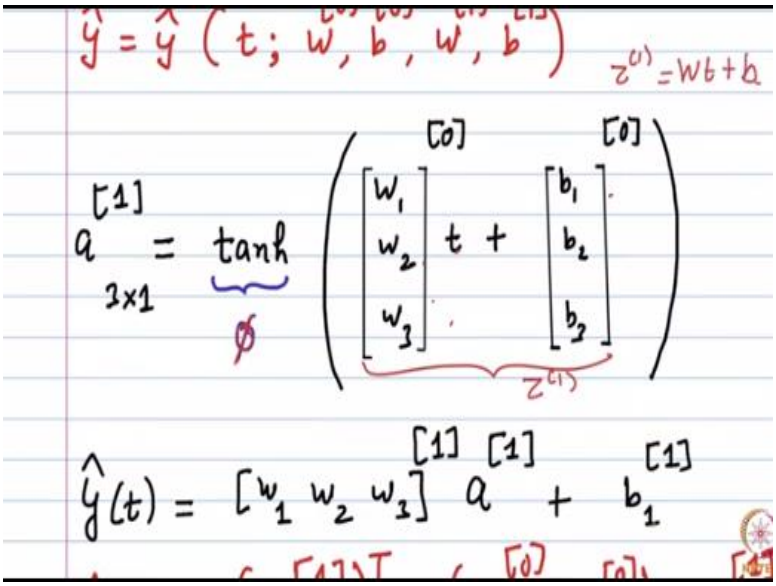
(Refer Slide Time: 09:22)



So, we come here. That data is basically free, the experimental data is what is restricted, okay. So here is the hypothesis function. The hypothesis function as you can see, depends on $w^{(0)}$, it depends on $b^{(0)}$, it depends on $w^{(1)}$, it depends on $b^{(1)}$ etc. Now what happens if we write it explicitly? Vikas has chosen to use $tan\ h$ here, just like we saw in the Burgers example case, we use tan h here.

So now you can write the output. The output here is, you can see this weight plus this weight. So, $w_1{}^{(0)}$ plus this weight, sorry so $w_1{}^{(0)}$, $w_2{}^{(0)}$, $w_3{}^{(0)}$ are all the biases that he has written out.

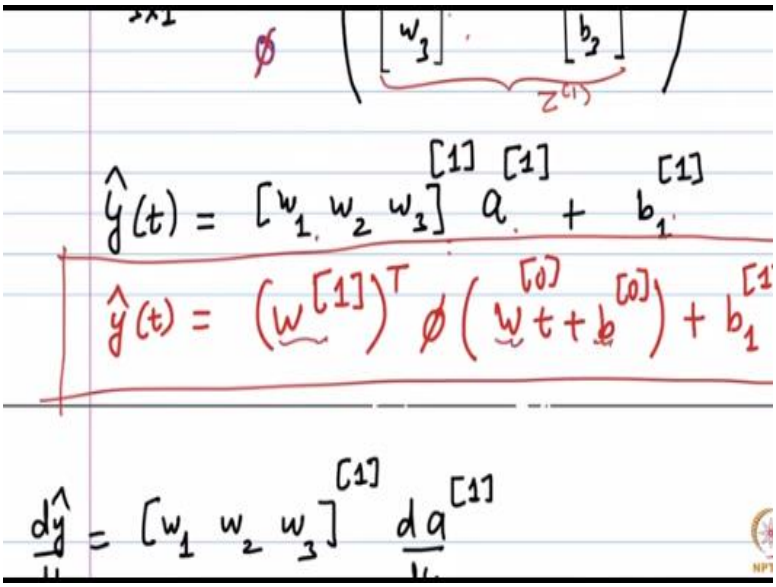
(Refer Slide Time: 10:12)



That you can see here. So, $w_1{}^{(0)}t + w_2{}^{(0)}t + w_3{}^{(0)}t +$ biases. So, this function he is calling ϕ. So, this is the standard thing. This of course is $z^{(1)}$. So, $z^{(1)} = wt + b$. So

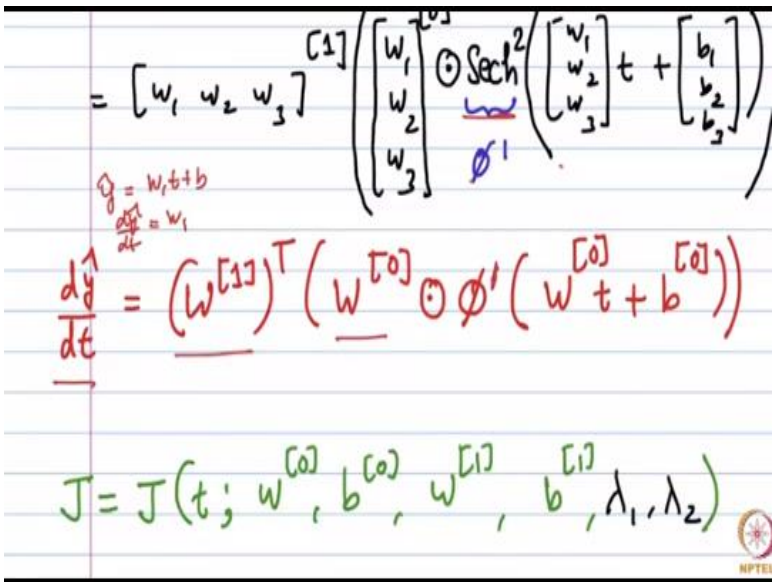
now you can write out y here with respect to the second weight. So, the second weight you can see is explicitly written here.

(Refer Slide Time: 10:43)



So, some other weight multiplied by a plus b. So now you can see this is actually the final hypothesis function that we have, complicated looking function. Notice how many unknowns are there. This is unknown, this is unknown, this is unknown, this is unknown. Now because our governing equation involves $\frac{dy}{dt}$, we differentiate this with respect to t.

(Refer Slide Time: 11:10)



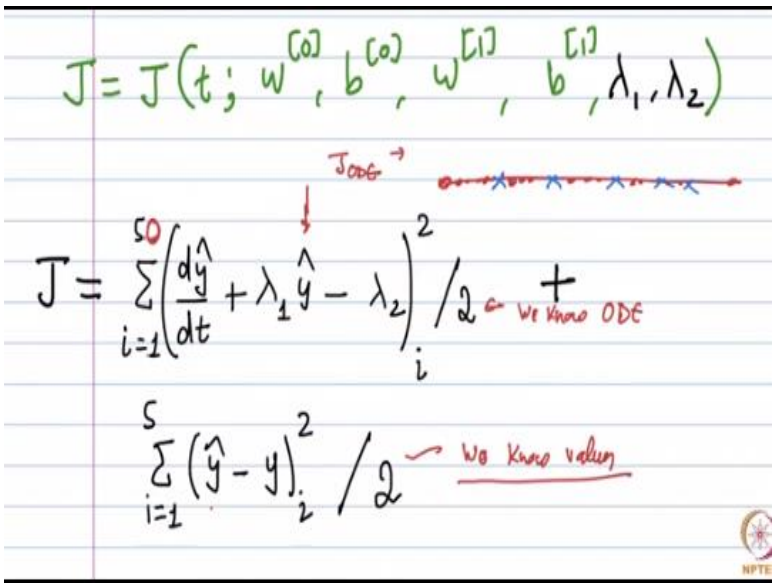
So, when you differentiate this with respect to t, you get a sec h^2 or $\text{sec}^2 h$, sec h^2 term here. Why because $tan\ h$ when differentiated gives sec h^2, so that term is there. So,

since the notes will be uploaded, for those of you taking course for credit, you can actually look through this and look to see if you can identify the differentiation here.

So, you see, $\frac{dy}{dt}$ can be written as some w times some w and this is now ϕ'. That is because $\sec h^2$ is sitting here as ϕ'. If for example, $\hat{y}$ was simply something like w 1 t plus b, then, $\partial\hat{y}$, or $\frac{d,\hat{y}}{dt}$ would simply be w_1. But it is a more complicated expression. So, we have written out the full expression here for whatever corresponds to this.

Obviously, this is going to be messy if I have more units than this, which is why we use automatic differentiation as you can see. If I had put 2 layers here, you really would not have been able to differentiate this by hand. Okay, if this is the case, then how do we proceed further? So, we write $\frac{dy}{dt}$. Now you write J.

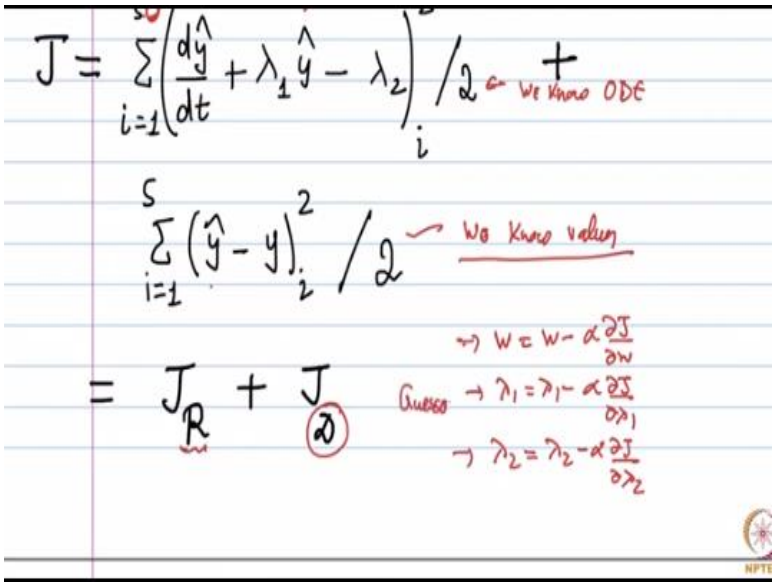
(Refer Slide Time: 12:43)



So now what is J? J as is shown here is J_{ODE}. How did we calculate it? So, J_{ODE}, Vikas has summed it only from 1 to 5, but you can have different ODE points. So let us say you can make these 50 points where you calculate the ODE solution. You can put a lot of points here. Whereas the function points are just 5. So, lots of points here, but just 5 function points and these are the function points here.

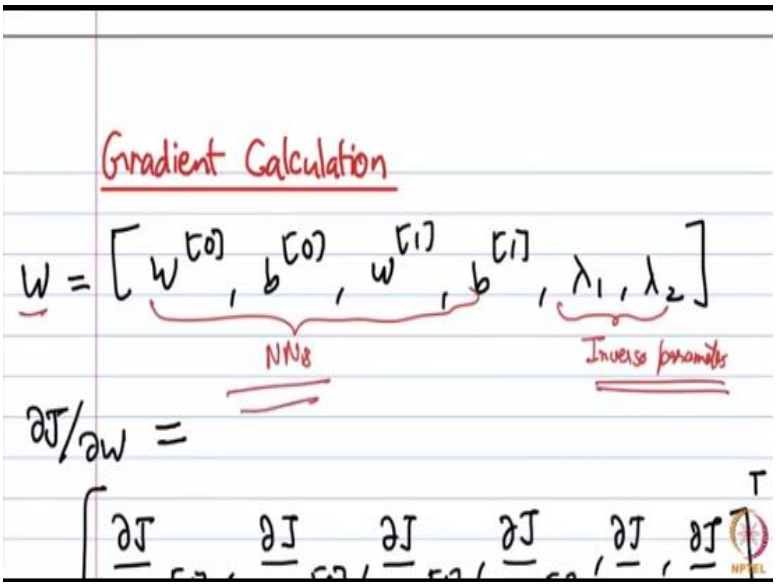
Now at these points, we know the values. At this point, we know the differential equation. So, we can use the differential equation at these points without bothering about the values here, okay?

(Refer Slide Time: 13:47)



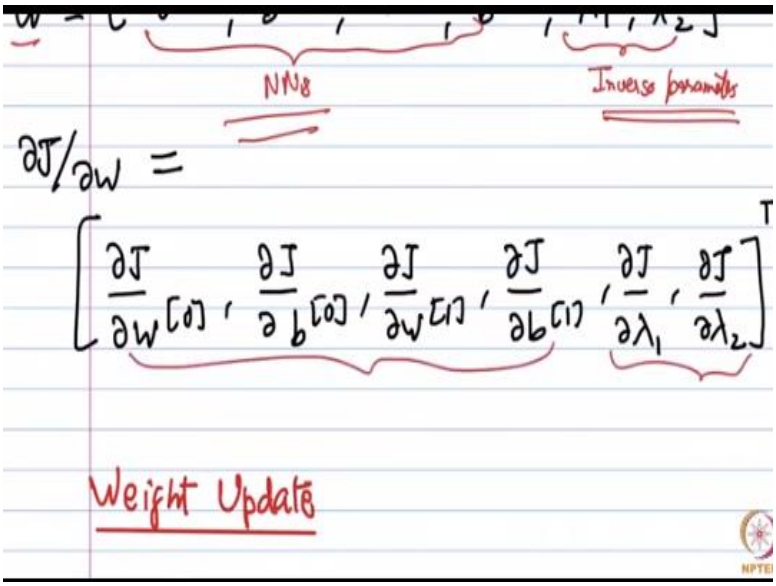
So, this he has written as J_R, which is J residue plus J sum function value. Now this is not where we stop. Remember we have to do $w = w\text{-}\alpha \frac{\partial J}{\partial w}$. But $\lambda_1 = \lambda_1\text{-}\alpha \frac{\partial J}{\partial \lambda_1}$ And $\lambda_2 = \lambda_2\text{-}\alpha \frac{\partial J}{\partial \lambda_2}$. So, all these are updates. We give guesses for these all three values, and then we update those guesses here.

(Refer Slide Time: 14:26)



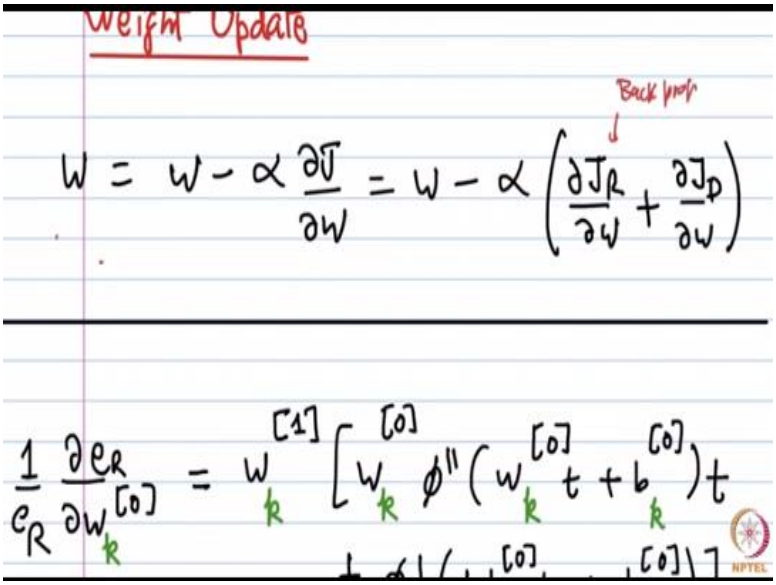
So, notice the W set or the new parameter set is these parameters which are from neural networks. And these are the inverse parameters, the parameters that we are solving from. So very unique idea to put everything together in the neural network. Already if you are solving for let us say 1000 weights, you might as well add a few parameters which are very parallel. It is extremely neat idea to do this.

(Refer Slide Time: 14:55)



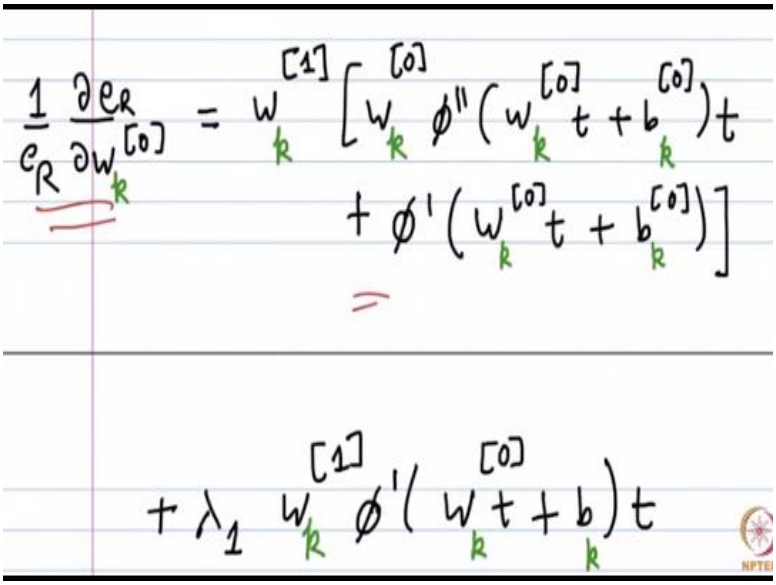
So, you can see you require these four from neural networks and these two from the inverse parameters.

(Refer Slide Time: 15:04)



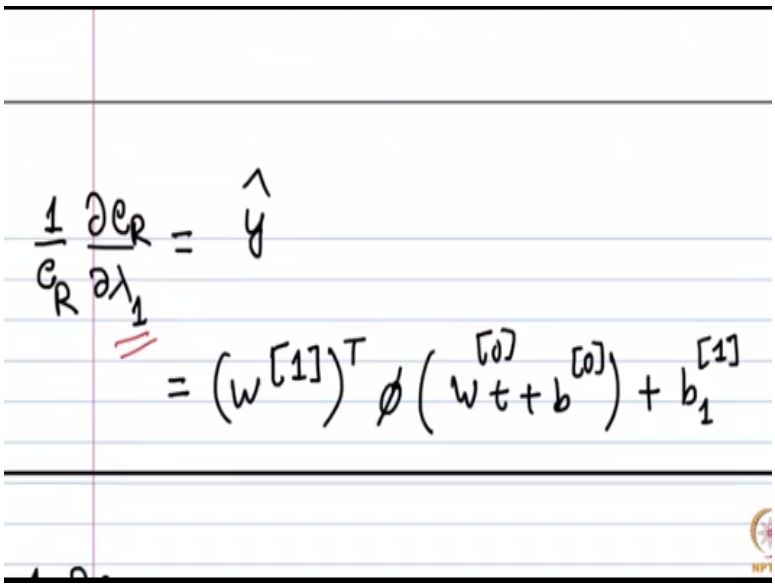
So as is written here, do w = w-$\alpha \frac{\partial J}{\partial w}$. These can be calculated simply by backprop, okay? So Vikas has written these out explicitly, you can notice the parameters that are given here. So, v here is simply the error. I will show you that at the end.

(Refer Slide Time: 15:29)



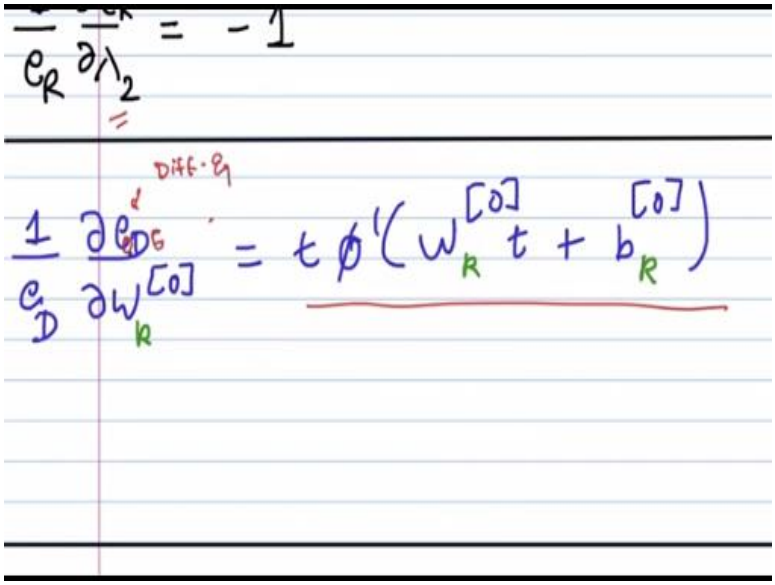
So, you can see, again you can see the differentiation. I am not going to look at these or go through these expressions in detail. I welcome you to go through this. The whole point of this is to show how messy this process can get just like we calculated ϕ', you will have a ϕ'' from a second derivative that we take. And you proceed in the same way. So, you calculate for the bias term, you calculate for the next w and you calculate this.

(Refer Slide Time: 15:59)



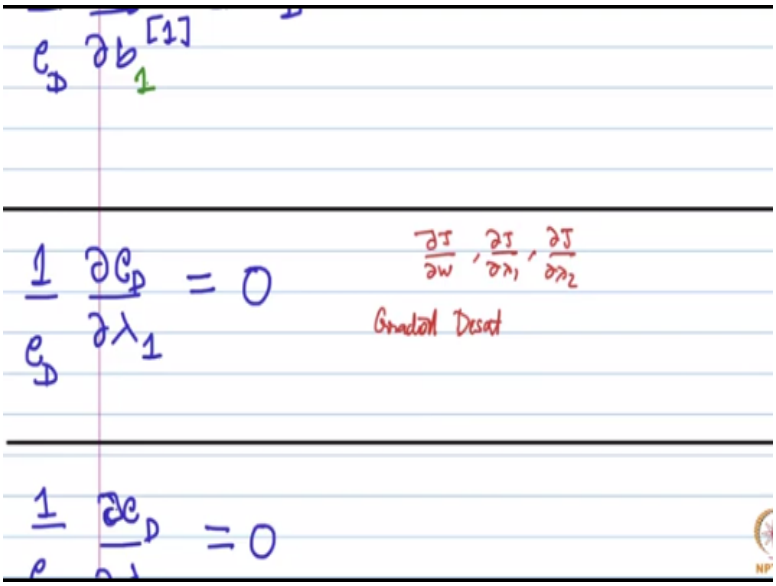
But most importantly, here are these simple terms, when you take a differentiation with respect to λ_1, and differentiation with respect to λ_2.

(Refer Slide Time: 16:11)



So here, the error can then be put together in this term, $t\phi'$. This is just the term that comes from the differential equation. We can call this error ODE, okay. So, when you put these terms together, every single thing is written explicitly here.

(Refer Slide Time: 16:38)



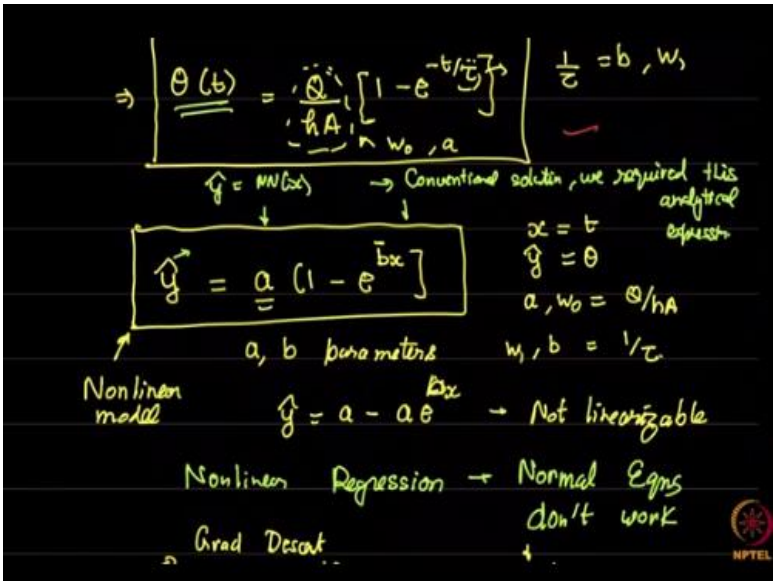
So, when you put these terms together, you get expressions for del J with respect to each one of these terms, δ_1 and δ_2. And then all we need to do is simply do gradient descent, okay? So, we you can as an exercise maybe if you are really interested solve the problem that we solved using Gauss-Newton using this approach. Of course, it is really, really messy to find out all these derivatives.

If you know Python or if you know MATLAB and the deep learning toolbox well, you can put it so that it differentiates the whole thing automatically. And you can solve this

entire problem as an inverse problem. So, what I want to point out here is the complete difference in approach between our original approach which used the Gauss-Newton nonlinear regression approach and this approach.

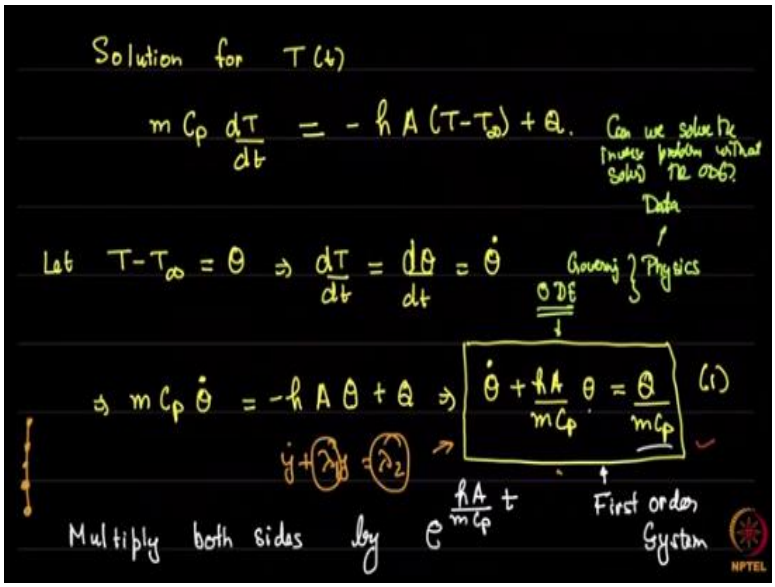
The Gauss-Newton approach did not start from the differential equation, but it started from the solution of the differential equation.

,**(Refer Slide Time: 17:39)**



So, it started from the solution. And since we knew the form, we could basically regress to it.

(Refer Slide Time: 17:53)



Whereas PINN cleverly starts with the differential equation itself, treats that as data. And of course, wherever we make measurements are taken as additional data and you

regress to it. And indeed, if you solve it this way, you will see that you get the same solution as the Gauss-Newton algorithm.

So, I hope at least this one threw a little bit of light on how exactly this problem could have been solved using a physics informed neural network approach. In the next video, I will similarly give you an overview of how the same idea could have been solved using a surrogate model approach.

Again, I am not going to show you a code because we have seen sufficient codes this week, and I will just give you an overall overview of how it could have been solved using surrogate models, the same nonlinear problem. So, I will see you in the next video. Thank you.