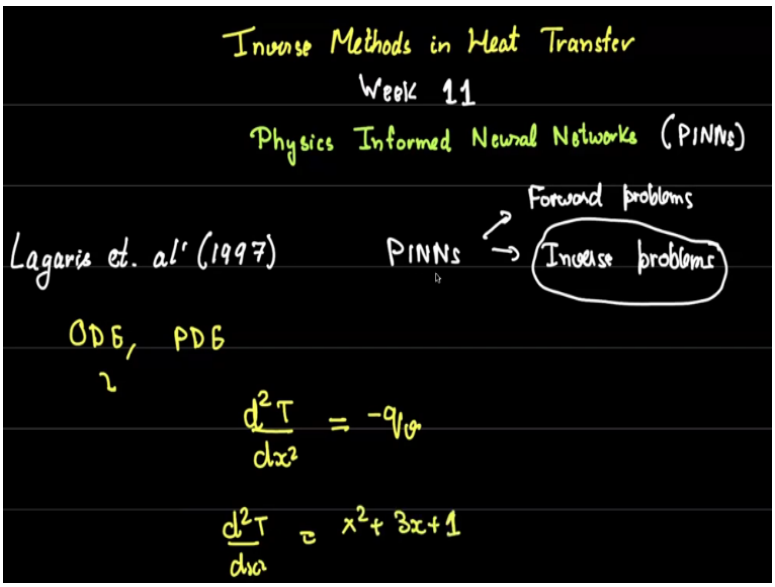
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology – Madras

Lecture – 57
Physics Informed Neural Networks - Introduction

(Refer Slide Time: 00:19)

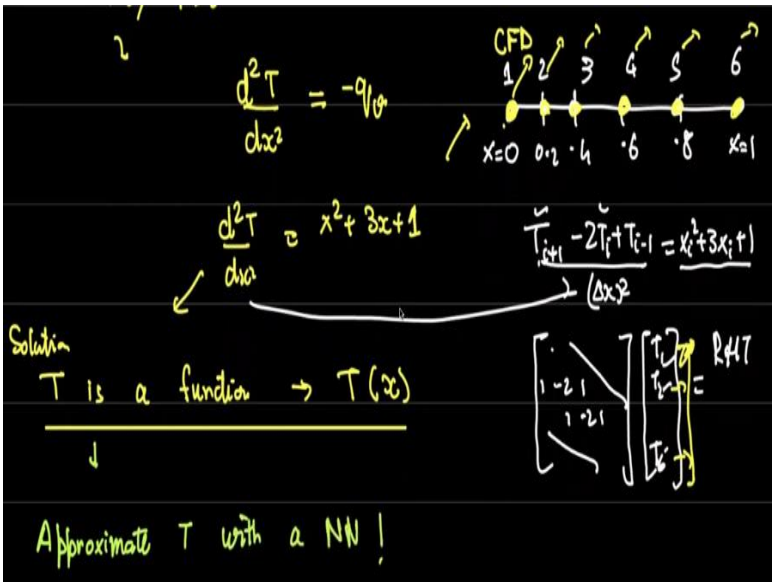


Welcome back to week eleven of inverse methods in heat transfer. In this video we will be looking at an important modern technique called physics informed neural networks. Pioneering work in physics informed neural networks has been primarily done by the crunch group in brown headed by professor George Karniadakis. But before this there was actually a very similar in my opinion paper by Lagaris et al this is sometime in nineteen nineties. I think the paper that I am going to show is Lagaris 1997. so, what I would like to do is to think of physics informed neural networks also called now called PINNs in two steps. first is PINNs for forward problems and then PINNs for inverse problems.

In my opinion it is in the solution for inverse problems that PINNs really shine in forward problems you will see that they offer a very nice alternative to traditional techniques the CFD technique that I introduced you to in the last week. But really speaking their major power comes in inverse problems where you do not need any large-scale simulations as I had shown you for the case of surrogate models in the last video. So, this is a very interesting idea like I said first introduced by Lagaris et al I will show you the paper. But let me first briefly introduce you to the idea okay so the idea is this suppose you have any ODE or PDE okay. So, I will take an

example ODE or PDE, so let us take some ordinary differential equation let us say we have the equation $\frac{d^2T}{dx^2}$ equal to some source some let us say some volumetric heat generation, in this case let me give you an example let us say volumetric generation looks like $x^2 + 3x + 1$.

(Refer Slide Time: 02:38)



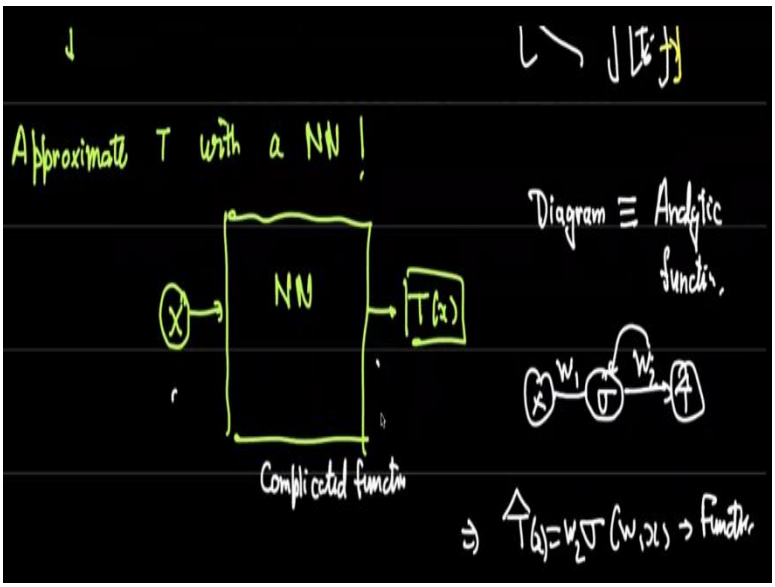
So, the way you would solve it using CFD would be you would make a mesh and you will take these points in the mesh some point some random points. so let us say in this case Δx let us say this goes from $x = 0$ to $x = 1$ and this is $x = 0.2, 0.4, 0.7$ let me make it uniform and add one more point here so point two, point four, point six, point eight and one so then you say something of the sort that this is i equal to one two three these are grid points five six then you will have a relation such that $\frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2} = x_i^2 + 3x_i + 1$.

So, this if you see the right hand side is known the left hand side basically becomes a matrix and you will have something like T_1, T_2 up till T_6 equal to some RHS so we will have some matrices which will look like one minus two one one minus two one and to get right diagonal matrix in case this looks too fasting please do take a look at a basic numerical methods course my point here is traditional CFD solves for these points separately. So, each of these solutions is a separate solution and you solve for that. now here is the PINN idea the PINN idea is this I know that T the solution T is a function particularly T here in this ODE is a function of x okay,

Now the way we solve CFD was not to assume that it is a function of x or at least not explicitly there are implicit assumptions here in terms of piecewise linear piecewise quadratic etcetera. So, you take these six as separate unknowns okay you have to of course give boundary

conditions which I am skipping briefly for now I will come back to the boundary conditions issue very shortly. but for now, let us just ignore the boundary conditions of the problem, but if we just look at these separate solutions you actually get separate solutions for CFD at least six patches. now there is a different method once I know that T is a function T(x), I can say approximate T with a neural network what does that mean?

(Refer Slide Time: 05:38)

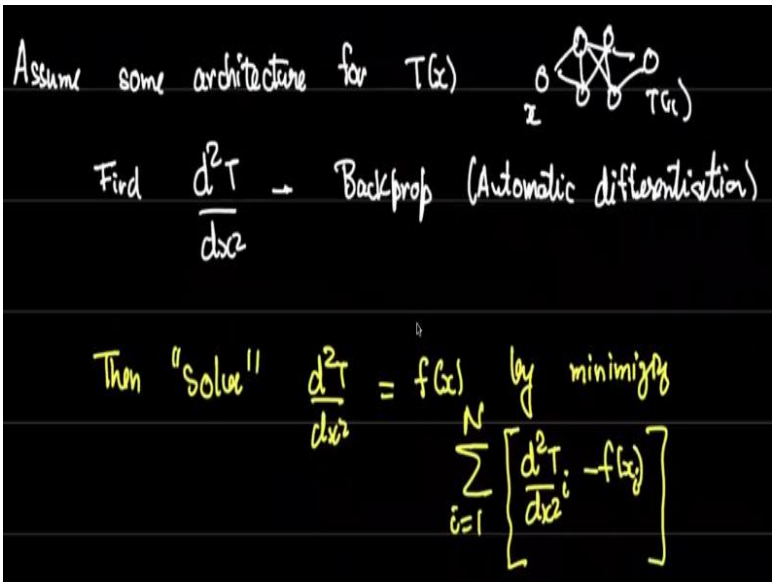


It means here is a diagram I take x, this is the input. There is some neural network here for example there could be one hidden layer multiple hidden layers, we have already seen these examples and the output here is T(x). Now notice even though as I have said it multiple times even though I draw this as a diagram every diagram you would have seen this in the exercise also is equivalent to an analytical function.

For example, if I take x just have one neuron here which is sigmoid and this is w1 and I have a linear output here y or let us call this T just to be consistent $\hat{T}$ and this is w_2 then this will say that $\hat{T} = \sigma(w_1 x)$, now this layer will multiply, this multiplied by w_2 so this then is a function okay so $\hat{T}(x)$ is also a function.

This also is a function except it is a complicated function. Now how does that help us. once I know that it is a function, the way I do it is as follows.

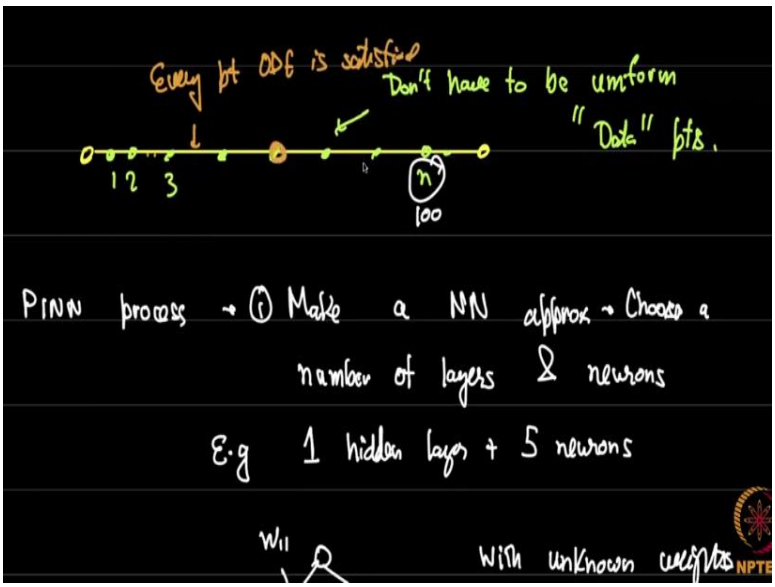
(Refer Slide Time: 07:06)



You assume some architecture means number of neurons, number of layers etcetera, assume some architecture for T(x) so it could be like something of this sort x, T(x) okay etcetera okay here there should not be any layers but anyway. so then use or find $\frac{d^2T}{dx^2}$, how do I find out $\frac{d^2T}{dx^2}$? once I know this this can be done through backprop or what is known as automatic differentiation which we saw in the last video. So, for a given T(x) you can find out what $\frac{d^2T}{dx^2}$ is and then solve let me show this a little bit better.

Then effectively solve $\frac{d^2T}{dx^2}$ equal to some RHS let us call this f(x), by minimizing $\sum \frac{d^2T}{dx^2}$ -f(x) all of these calculated at i over lots of points.

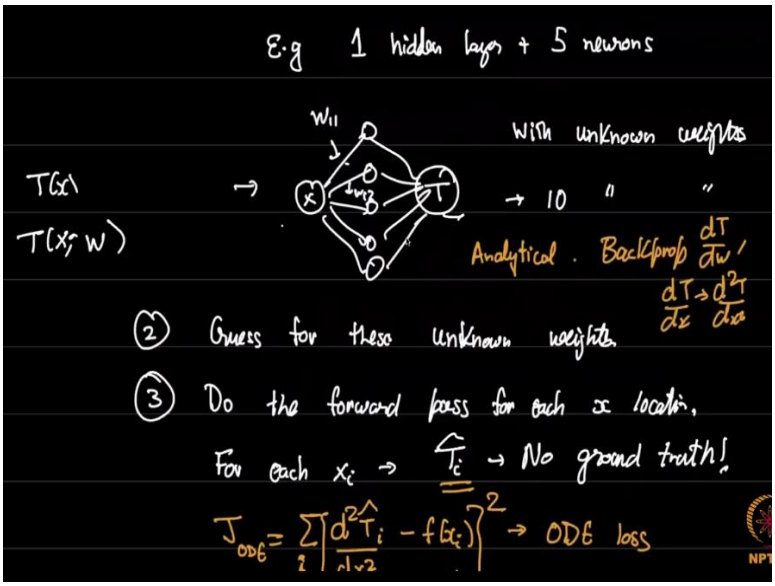
(Refer Slide Time: 09:07)



So, the way this works is as follows the same domain I take lots of points now unlike CFD these points have a different purpose. They do not have to be uniform. These are treated basically as data points, where I will show you the brilliance of this when it is applied to inverse problems later on. But these do not have to be uniform you can basically treat these as data points. so let us say we have one two three n data points okay. now the general process is as follows, the process is making a NN approximation what do I mean by making NN approximation?

Choose So let us say I can choose number of layers and neurons so let us say for the differential equation that I have shown I make a simple choice.

(Refer Slide Time: 10:26)

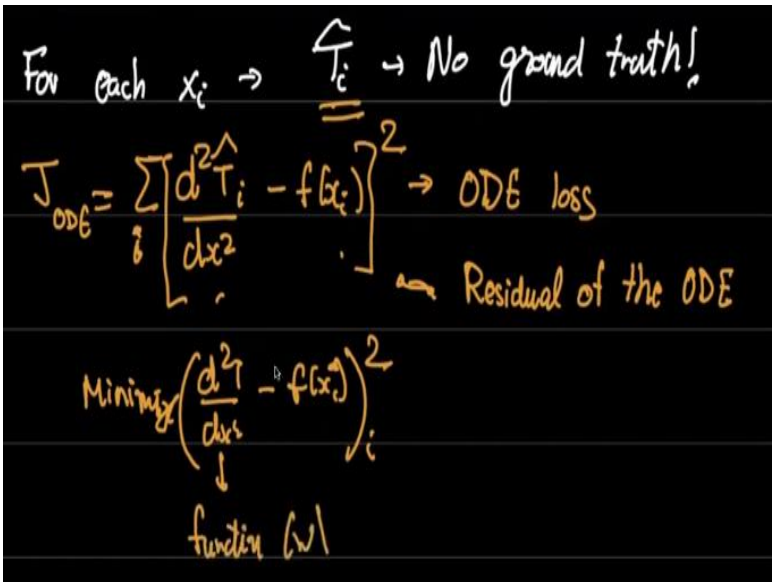


So, for example I choose one 1 hidden layer plus five neurons, if I make that choice, so this is what it will look like x then one two three four five and then T okay. now what is the problem here? the problem here is we do not have these weights so w one one w one two etcetera with unknown weights okay. so let us say our weights are also unknown okay so we have a whole bunch of weights here let us say we have weights and biases, but let us say just weight so let us say in this case we have ten unknown weights okay. So, what do we do we do the usual thing that we do with neural networks guess for these unknown weights.

This also looks simply. So, we just guess for these weights alright. So, we have an initial guess. now we need to improve, do the forward pass okay. So, we do the forward pass how many times n times, let us say I have hundred data points so for each x location okay, so if you have a domain going from zero to one and you have hundred locations randomly thrown, you throw these hundred locations, do a forward pass, now what will happen is, for each x_i, I obtain a T_i

that is fine. but notice there is no ground truth I do not know what temperature is supposed to be here. so how do I do a backprop how do I find out a how do I find out a loss function here.

(Refer Slide Time: 12:48)

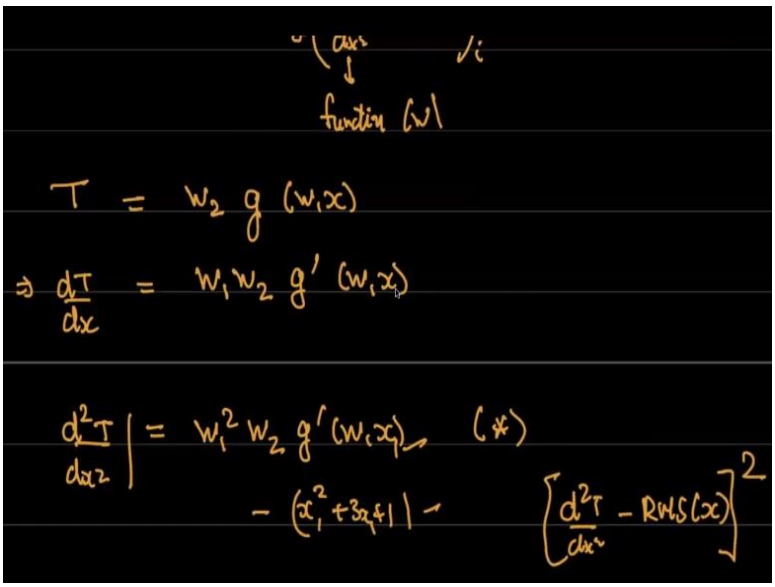


So, here is where the genius of Lagaris has shown you define J as I showed earlier you define J not on T_i but on derivative of T_i. So, $J = \frac{d^2\hat{T}_i}{dx^2} - f(x_i)$, so the loss is what I would call the ODE loss. So, we will call this J_{ODE} notice if you had the data there you could have used it but you do not have the data, but you do know you know one secret in this domain it is not as if nothing is known about what happens here. we know that at every point the ODE is satisfied. so regardless of where you sample even if you do not know the temperature you do know that the temperature satisfies a certain relationship which is $\frac{d^2\hat{T}}{dx^2} - f(x_i)$ is supposed to be zero. now instead of setting that to zero we want now we cannot satisfy that every point but we can minimize it.

So, instead of doing this you try to minimize which would be the same as trying to force it to zero if we can do so okay. So, what we do is we simply such create a loss function which is the sum of the individual residuals of the ODE okay. So, this is a very simple but powerful idea but you will say okay I had only T during the forward pass where do I get $\frac{d^2T}{dx^2}$ from. Now notice x led to T and it is an analytical expression which means that through backprop just like I can find $\frac{dT}{dw}$ the process that I showed you I can find out $\frac{dT}{dx}$ also and in fact I can repeat this back prop twice and I can get $\frac{d^2T}{dx^2}$ so when you calculate $\frac{d^2T}{dx^2}$ this will be a function of the w for example let us take the simple case here $\hat{T} = w_2\sigma(w_1 x)$.

This will mean $\frac{dT}{dx} = w_2\sigma'(w_1x)*w_1$, let us let me write it here just to be clear.

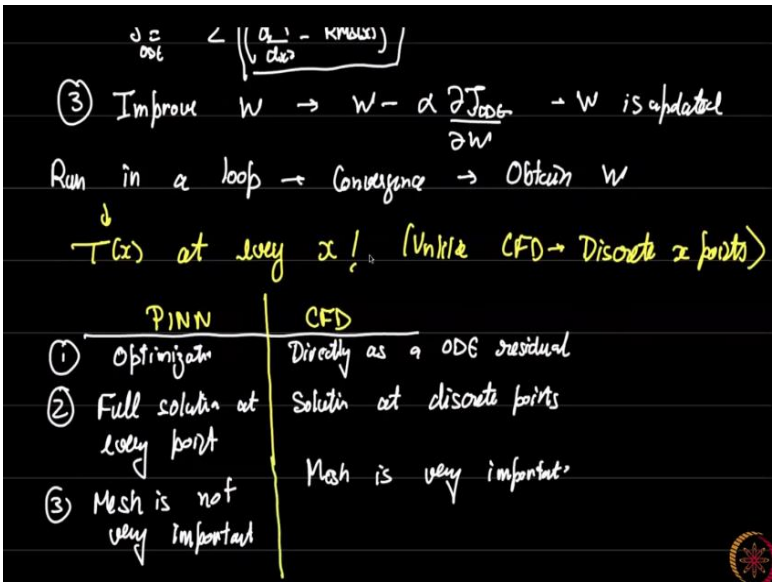
(Refer Slide Time: 15:42)



So, in case T equal to let us say $w_2\sigma(w_1x)$ or let me replace the sigmoid with some general function $w_2g(w_1x)$ this means $\frac{dT}{dx} = w_1w_2g'(w_1x)$, which is if you notice another function of x as well as w_1 and w_2. Next $\frac{d^2T}{dx^2} = w_1^2w_2g'(w_1x)$. So, notice this so when I give you a new x I have let us say hundred x so if I give you x_1 you can calculate this at x_1, you also have the right-hand side which was some $x^2 + 3x + 1$ let us say so you do x one square calculate this okay so this for.

For a given x one we can calculate this then you can write $\frac{d^2T}{dx^2}$-RHS(x), this will be the loss function you can calculate that.

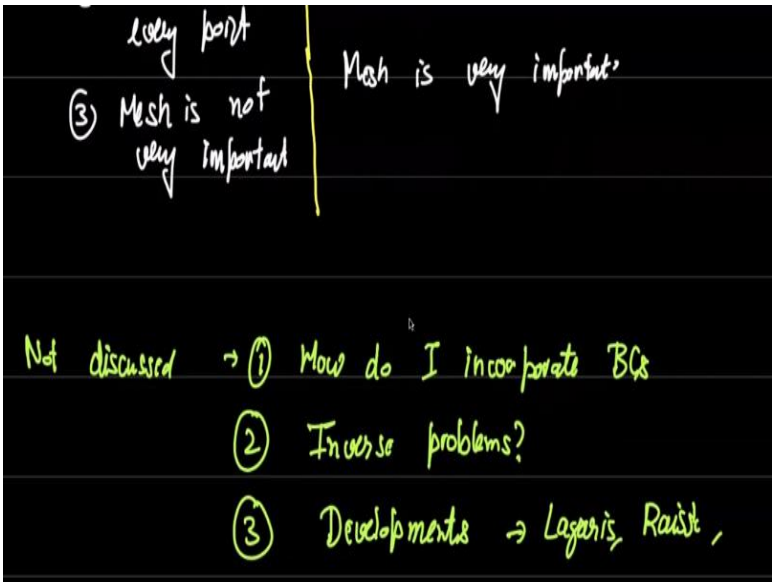
(Refer Slide Time: 16:58)



Now you have the last step so basically J has been calculated is $\sum(\frac{d^2T}{dx^2}\text{-RHS(x)})^2$ this is just the ODE loss. The final thing is you want to find out improve w, so $w = w\text{-}\alpha\frac{\partial J_{ODE}}{\partial w}$, so you do this for all the ten w, then w is updated. so now run in a loop at convergence you obtain w. now once you obtain w it means you have T(x) why because T(x) was simply a function of x and w which means unlike c of d which gave you solutions only at six points this gives you T(x) at every x unlike CFD. CFD gives you only at discrete x points

So, the difference between PINN and CFD is PINN treats the problem as an optimization problem. what are you trying to optimize? you are trying to minimize this function let whatever be the function that you are trying to solve you are trying to minimize that function I will make this a little bit cleaner in the next video. CFD on the other hand is solving this directly as a ode residual ODE or PDE residual. second is you get the full solution at every point. here you get solution at discrete points. Finally, this mesh is not very important okay we will see how that affects things shortly. whereas here mesh is very important, because depending on how I give the mesh my difference equation or my difference approximation actually changes.

(Refer Slide Time: 20:09)



Now there are a few things that I have not discussed here. The things that I have not discussed at this point are how do I incorporate BCs second how do I solve inverse problems and of course some developments in this field okay. So, for example I want to talk about what the general idea of Lagaris was how that was improved by Raissi and Karniadakis and then there is some work that we did even at IIT Madras regarding this to improve upon this method so these developments I will discuss in the future videos. This was just a very brief introduction to the idea of physics informed neural networks I will also talk about why they are called physics informed as is as against something else which it becomes obvious once we talk about inverse problems.

But before I go further, I just want to you to remember one very key point here. There is no data okay unlike let us say a normal neural network, where you give some ground truth, the ground truth here comes from the differential equation. So, the truth that you are trying to talk about is the fact that at every point here you can choose any point in the domain we know that the differential equation is satisfactory. that is the genius of the method which was first introduced by Lagaris. so the idea is this every point here you can apply this loss function without any doubt, even though you do not know what T is you do know what $\frac{d^2\hat{T}}{dx^2} - f(x_i)$, so I will show you a simple example of this you know without the confusion of the neural networks in the next video and then we will proceed with what exactly Lagaris did, Raissi did and other developments in the next video so I will see you in the next video thank you.