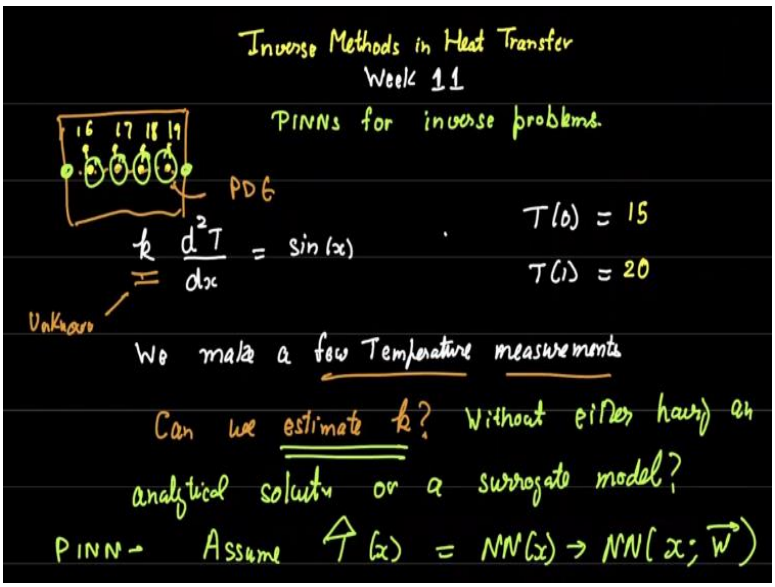
Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology – Madras

Lecture - 60
PINNs for Inverse Problems

(Refer Slide Time: 00:19)



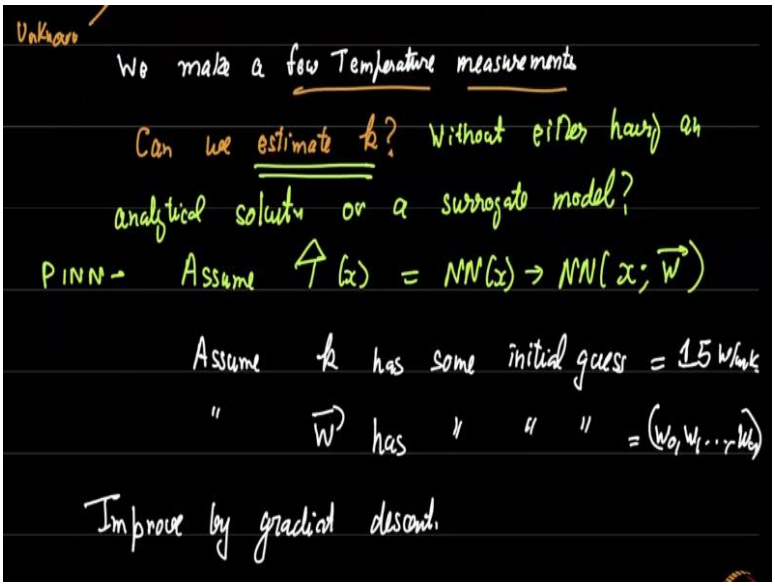
In the last few videos, we saw how to use PINNs for direct problems. In the previous video I in fact showed you how to incorporate boundary conditions but let us now consider PINNs for inverse problems which is where in my personal opinion the true power and genius of the Raissi approach as well as the PINN approach lies okay. so now let us take an equation such as this okay some heat addition or some other equation you just treat it as a PDE or an ODE for now $k\frac{\mathrm{d}^2\mathrm{T}}{\mathrm{dx}^2} = \sin(x), \mathrm{T}(0) = 15, \mathrm{T}(1) = 20$, there is no problem in solving this equation provided you know k. but let us say we do not know k, so k is unknown if k is unknown.

And let us say you are within a slab and you make a few temperature measurements let us say at four or five occasions you make temperature measurements and you do get values. So let us say it is sixteen here seventeen here eighteen here nineteen here just to give you an example I mean I am just making up numbers. So let us say we make these measurements we have also measured the boundary condition and we want to make an estimate for k. So, the question is can we estimate k now without here is the catch either solving have either having an analytical solution or a surrogate model? Now what do I mean by this? suppose I have an analytical

solution let us say the actual analytic analytical solution for this happens to be T equal to let us say $-sin(x)$ or something else okay.

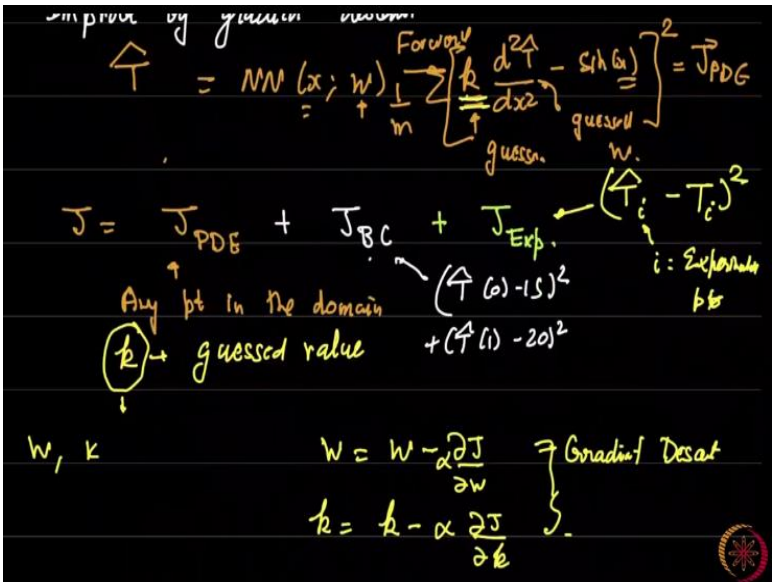
Then you do the standard thing you put that analytical model put it in your forward model you never bother about the differential equation now. you have a $\hat{T}$ and then you take these measurements and then it is a standard inverse problem with some models of the temperature you know that $T = w_0 + w_1 sin(x)$, then you can just solve it for w_0 and w_1, but we do not have an analytical solution dependent on k. Now we do not have a surrogate model for the forward solution also and you are not able to solve using CFD here is where PINN is very helpful so the PINN solution works as follows.

(Refer Slide Time: 03:00)



The PINN solution is assumed that $\hat{T}(x)$ is a neural network and this neural network has some weights okay. I am going to call it w vector okay so assume that is the neural network assume that k has some initial value. what do I mean by initial value? I am taking an initial guess let us say k initial guess is fifteen watts per metre kelvin, again assume w vector has some initial guesses this is as usual so w_0 and w_1 etcetera, you make some guesses for their values okay. Now we have to improve by gradient descent. we have no other go other than to improve by gradient descent, but how are we going to improve by gradient descent?

(Refer Slide Time: 04:16)



We already have $\hat{T}$ is neural network of (x; w), do a forward pro with the given value of w this gives you some $\hat{T}$ it also gives you some $\frac{d^2\hat{T}}{dx^2}$ as we saw earlier -k$\frac{d^2\hat{T}}{dx^2}$ - sin(x) okay. So, this is of course our J_{PDE} or J_{ODE} square it adds it for all the points okay one by m as usual. Now notice this k is our guessed value of k and within this $\frac{d^2\hat{T}}{dx^2}$ lie our guessed values of w okay x is of course known because we are guessing at specific x's okay.

So, which x should we guess at we guess everywhere in between for PDE loss but this J has three components now J_{PDE} can be any point in the domain, why because PDE is satisfied at every point in the domain now at these points we still need the value of k for this loss. k is from the guessed value okay so really nothing has changed from our usual PINN except that you have a new parameter k which you are guessing for then you also have your boundary condition losses which of course is the same as before this will be $\left(\hat{T}(0) - 15\right)^2 + \left(\hat{T}(1) - 20\right)^2$, so that is also straight forward.

Now the final part is J data or J experiment. now what is this we had these four points where we measured the temperature. Now the measured temperature is something and our hypothesis function will say that the temperature is something else. So, we need to make sure that the difference between these two is reduced to as little as possible. So, for example we would add this would be sorry temperature at all the experimental points minus the actual by some called experimental points I just call it $\left(\hat{T}_i - T_i\right)^2$ where i are all the experimental points. So let us look at this loss function once more J is made up of three parts, first part is the pure physics part.

The pure physics part can be any point in the domain you can just choose as many points as you want you can choose thousand ten thousand a million the choice is yours, the more you add obviously the more computation you spend in finding out J_{PDE}. but you can spend you can find out any point in the domain but the catch here which is different from the initial forward pass that we did in the last three videos is this scale is part of the iteration. So, you have to start with an initial guess for k that is fine. calculate this through the neural network this is of course known, boundary conditions are given in the problem whether it is a function boundary condition or a derivative boundary condition you can easily apply it $\left(\hat{T}(0) - 15\right)^2 + \left(\hat{T}(1) - 20\right)^2$ and add.

Finally, you have to look at the fact that some extra information has been given which is the experimental values. This is what determine the inverse problem. So, you ensure that the prediction you are making with your neural network matches the experimental as close as possible. So, we have these three losses that is fine now what do you need to update we need to update w but we also need to update k. we can update w easily, $w = w - \alpha \frac{\partial J}{\partial w}$, gradient descent. Now what do we do about k turns out k is very straightforward to $k = k - \alpha \frac{\partial J}{\partial k}$ this is also gradient descent. So, now notice the beauty of this system which is that here w, were the parameters of the neural network and k is the parameter which sits in the PDE but as far as gradient descent is concerned these two are not different, they are just parameters.

So, you update these w, as $w - \alpha \frac{\partial J}{\partial w}$ and k you update as, $k - \alpha \frac{\partial J}{\partial k}$ really nothing really is different as far as the neural network is concerned in fact Raissi and Karniadakis which I show you shortly they just put the same usual neural network algorithms and it just works like a charm okay. So, the entire process is such that the inverse solution is baked in into the forward solution. what do I mean by that? notice that when you want to solve this problem if I want to solve this problem traditionally and somebody gave you this problem and said solve it and I will not tell you k there is no way you can solve it okay you cannot solve it here too you cannot solve it but you solve it as if you knew a value of k you solve it the next guess for k depends on what the solution was okay for this guessed value of k so.

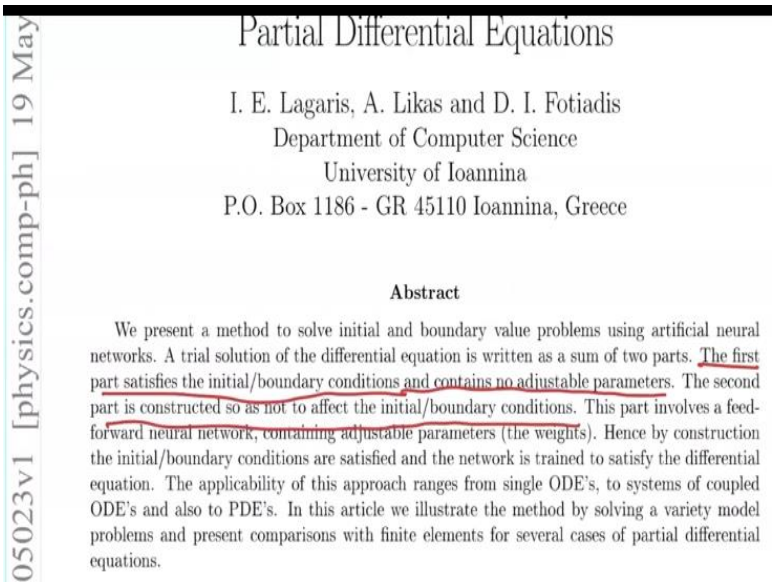
Everything is incorporated simultaneously which is really elegant I mean it is a really a pretty way of solving the whole problem. So, what you do is unlike the surrogate model approach the

surrogate model approach will be okay I will guess one value of k solve this problem, guess another value of k solve this problem, guess a third value of k solve this problem solve this for two hundred values as I showed you in the first video in this series then basically have a correlation between temperature distribution and k make a neural network for that. After that when I give you new temperature measurements, then you see which k fits amongst all these simulations that you did you solve that using a genetic algorithm.

This one is much simpler guess a k guesses a temperature distribution next iteration improve temperature distribution improve k also based on what based on three pieces of data. I know the boundary condition, I know so that is one piece of data I know that the physics should satisfy this equation everywhere in the middle, that is this loss J_{PDE} loss J_{BC} loss and the third and final part is the experimental data that we collected. notice that all three feeds towards this gradient, so, when k is updated k is updated based on for this k was the PDE satisfied for this k was the BC satisfied and for this k was the experimental value satisfied.

Similarly, w looks at is the PDE satisfied is the BC satisfied is the experimental value satisfied. now all three are updated simultaneously there is no other method like this prior to this prior to the PINN method in the literature. So, what I would like to show you as I promised in the previous video is to show some of these seminal papers shortly. so please just look at the papers that I will be showing in a minute or so.

(Refer Slide Time: 12:14)



So, the first paper I wish to show you was this seminal paper called artificial neural networks ANN for solving ODE and PDE Lagaris et al. He is a professor in Greece the paper was

published in nineteen ninety-seven and it took a good twenty-one twenty-two years for it to sort of get incorporated into the modern framework by Raissi and Karniadakis. So, you can see these talks about how it incorporates initial and boundary value problems using architectural neural networks. you can also see this trial function is written it is a function of two parts.

This is not the same as the Raissi approach, these two parts one part satisfies the boundary condition exactly whereas in the Raissi thing everything is satisfied in the neural sorry in least square sense and this contains no adjustable parameters the second part is just the PDE okay it only contains the PDEs okay. **(Video Starts: 13:17)**

So, I will just show you their approach their notation is a little bit different you can see this is the equivalent of our n here they just write it and psi is the equivalent of our t or our u whichever function we are trying to solve you write the whole equation as something equal to zero and they write it as two parts. this part satisfies the BCs exactly okay so that is different from what happens in Raissi. this part is for the PDE or the ODE okay while not affecting this okay.

So, this is a very clever construction it sits there but overall, the Raissi approach is easier to apply for general problems even though it is less adjustable for various equations. So, you can see this it looks a little bit complicated but the idea is very simple. if you have this equation, you would turn it into you can see d psi dx minus f the whole square okay. so that is basically the loss function okay.

This on the other hand the boundary condition is ensured to be satisfied exactly here x is an initial value problem. So, this is like time. So, they satisfy it exactly. So, you can see they do this construction of as I told you x(1-x) etcetera so this is in order to satisfy a boundary value problem where you have a Dirichlet boundary condition and Neumann boundary condition on the right-hand side and they have several special constructions. now they deal with as you can see Laplace equation an example which I also gave you once again you will see a special function construction in order to make the boundary conditions satisfied exactly. So, BCs are exactly satisfying. incidentally this paper you can search for on archive and google and you will obtain it regardless of whether you are within an educational institution or not.

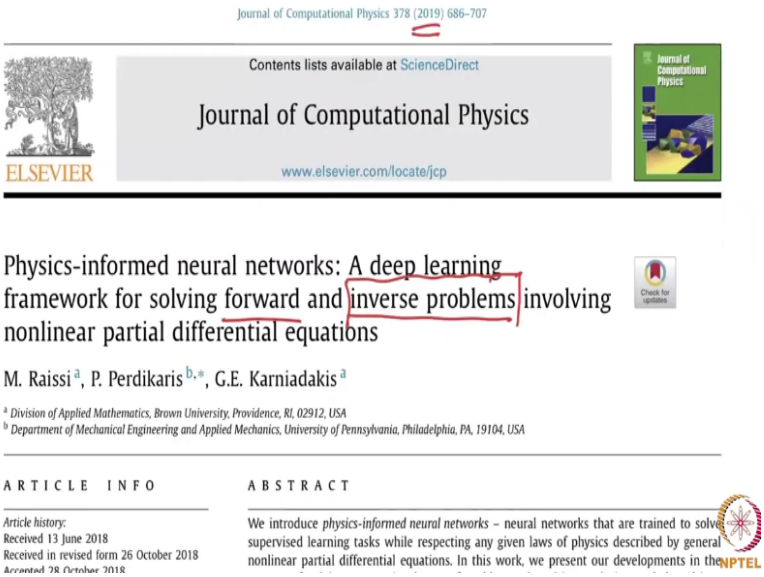
Now again notice the function the function is since it was originally $\partial^2\psi = $ f(x), f(x,y) now you can see $(\partial^2\psi - f(x,y))^2$ is minimized and you take some points, these are the PDE loss

points within the domain. So, they satisfy this and they have taken a sigmoid neuron, you can see that the Lagaris paper is a shallow network just one hidden layer with ten hidden units and they get very good results with that this is of course please remember nineteen ninety-seven computational power etcetera was not there at that time. now with this they satisfied they solved an impressive series of problems we can see quite a complicated ode you cannot they have an analytical solution here and they compare with it another ODE here.

These are first order odes then you have a second order ode then they of course satisfied coupled first order odes which is very impressive and then they look at PDEs okay. So, all these are solved here a whole bunch of odes. so exact solutions here I want to show you something which helps you visualize what is happening. So, look at the final solution of the problem okay so this is solution accuracy let us say the solution of the problem is this when we say that $\hat{T}(x, y)$ is this what we mean is $\hat{T}$ is a neural network that when you take in x and y that outputs this function this entire function is outputted by this. So, this is $\hat{T}(x, y)$ let us say this is x and this is y the output is $\hat{T}$ of x of y regardless of whether it is a neural network or not.

It is simply an analytical function I have probably said this hundred times but despite me saying it usually in classes I find that people miss this simple point okay and the what was the data there was no data other than the boundary conditions. the boundary conditions were used and the other information piece that was used is PDE is satisfied everywhere within the domain okay. So, whether you look at this function so this is again a solution accuracy function all of this function this is again $\hat{T}(x, y)$ what enables our neural network do this is that if I change w one of the shapes will look like this if I change w the other shape will look like this. **(Video Ends: 18:01)** So it is the universality of our neural networks that allows us to do this with Lagaris approach.

(Refer Slide Time: 18:15)



Now I what I want to show you also is the other seminal paper you can see this is twenty nineteen where you can see now that they are solving both forward as well as inverse problems as I said this apart from being the important point of our course also is a really seminal contribution as far as this paper is concerned. just a small change about Lagaris idea but sometimes small changes can lead to very profound effects.

(Video Starts: 18:41) So, that is what is being done here you can see that they talk about supervised learning tasks but they are adding physics. the way they look at it is some experimented data is given as well as a certain amount of physics is known. So, they define the governing equation this way.

Let us say it is an unsteady problem or a time dependent problem so t, here is prime so just to give you an example you could have something like u_t you could have an advection equation u_t plus let us say au_x equal to zero in that case if you look at $\mathcal{N}$, $\mathcal{N}$ simply is an operator which is au_x okay. So, $\mathcal{N}$ could in general be a nonlinear operator for example as we will see later you can have burgers equation where you have a nonlinear type. So, for example here is burgers equation where you have uu_x-u_{xx} I will show you that equation shortly. So, all they have is a simple method just like the Lagaris method which satisfies this at one shot. what we are doing of course is, we are saying that you have the operator f this is f here is not a function it is an operator.

Operator meaning it takes a function and returns another sort of function so it takes u and it returns $\frac{\partial u}{\partial t} + \mathcal{N}(u)$ and here is what they have they have the same idea that I told you the least

square loss J equal to J at the experimental points or J at what they call the u points or the data points plus J where we impose the physics which is through the PDE. So, wherever you know the experimental measurements or the computational measurements already you can see this is the ground truth and your network returns this okay. So, similarly, this is simply calculated from the network by differentiation yeah so as they mentioned here MSE_u corresponds to initial data boundary data and any experimental data that you have MSE_f is basically where you have physics.

So, these points where we impose them are sometimes called collocation points. here they have like we have said that even though this has been observed in previous studies, they are now using modern computational tools such as TensorFlow and other things. again, if time permits, I would like to show you an entire code for this using MATLAB in the next week okay. So, they have several examples here one of these examples let me show you the results they actually showed the Schrodinger equation also. but looks like a complicated equation so now one beautiful thing about this entire process is you do not need to put much effort to create an entire mesh or anything of that sort we just throw points here and there ensure the boundary conditions and initial conditions are satisfied.

for example, if you want to satisfy this all you would say is $(h(t,-5)-h(t,5))^2$ you add this to one J boundary condition loss okay. you can add this as $(h(0,x)-2\,\mathrm{sec}\,h(x))^2$ you see how simple it is all you need to do is this square this will be added to the J_{PDE} loss okay. None of us know or in case you do not know Schrodinger's equation you need not worry it works exactly in the same way. So, here it is we have some solutions we have given these data points which are basically add the boundary and initial conditions. So, these are just like the usual thing and you get a full solution here of Schrodinger equation which is apparently really good okay.

So, I am not going to show you the discrete time models here I want to show you the burgers equation solutions okay so here is one interesting example of an inverse problem that they did okay. So, here is an inverse problem they have a cylinder vortex shading cylinder at a particular Reynolds number re equal to hundred. So, now what they have is they made a lot of measurements in the entire domain. They just have points let us say they are coming from piv or some such experimental measurements. Now unlike what we did with the surrogate model approach which I showed you as the first video in the series all they did was you measure u you

measure v and you put in the Navier stokes equations and you put in the Navier stokes equations and you know you serve it this way.

Suppose the Navier stokes equations let me just write it here is $u_t + uu_x + vu_v = -p_x + \mu\nabla^2 u$ some of you might remember this if you do not it does not matter you have a complicated equation, so this is just the u momentum equation all you need to do is say $u_t + uu_x + vu_v + p_x - \mu\nabla^2 u$, calculate this square it and this should be minimized okay very simple idea you take a guess for u you take a guess for v they do this using the streamline vorticity formulation which it will take us too far away to discuss that. but you do anything of this sort this is a simple measurement and then ensure that J equal to J_{PDE} plus J boundary condition plus J experimental data, minimize this.

Now what was the inverse problem they were solving in some cases you can try and find out what this mu is okay you can actually solve for mu by looking at a fielding okay. all these inverse problems can be solved with this nice approach. they also had some nice burgers equation solutions which I come yeah here it is. **(Video Ends: 25:32)**

(Refer Slide Time: 25:34)

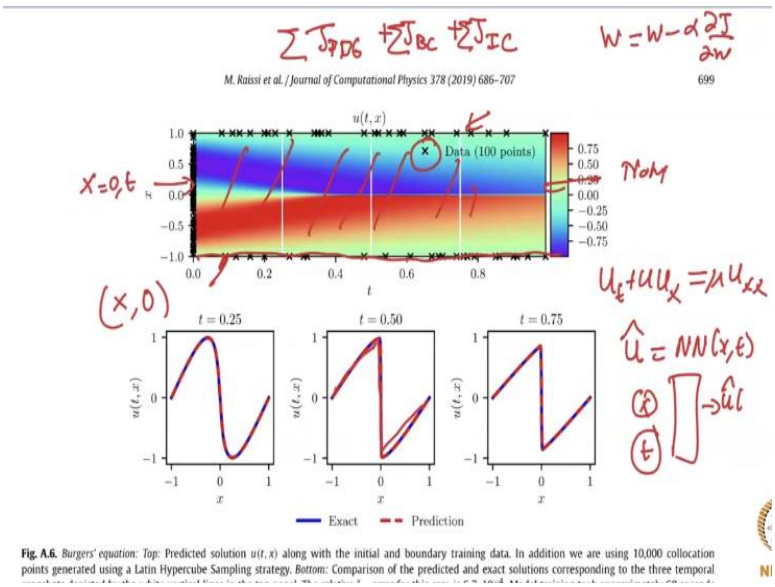
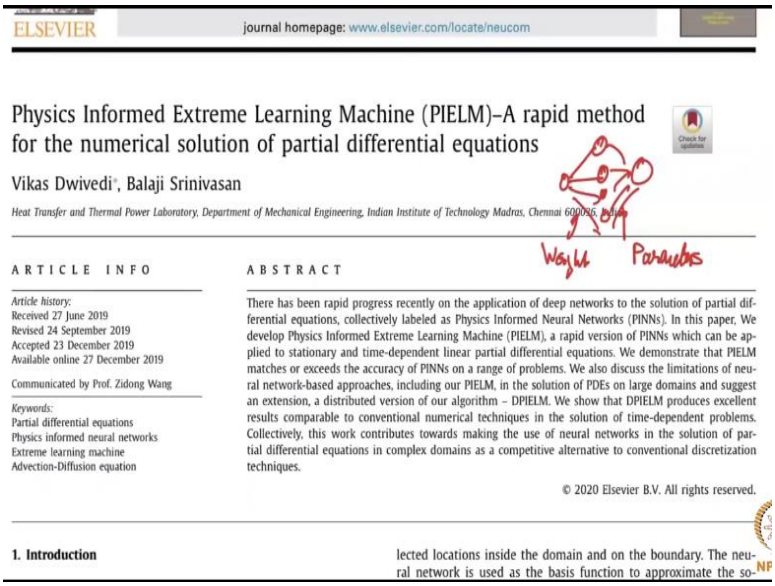


Fig. A.6. *Burgers' equation: Top:* Predicted solution $u(t, x)$ along with the initial and boundary training data. In addition we are using 10,000 collocation points generated using a Latin Hypercube Sampling strategy. *Bottom:* Comparison of the predicted and exact solutions corresponding to the three temporal

So, burgers equation this equation is u_t it is a one-dimensional equation equal to sorry μ times u_{xx} or μ times u_{xx} so you can solve both forward and inverse problems here you can see this nice sharp solution. again, the same thing you basically just say u is a neural network of x and t so take x_t put a neural network here it gives out u or you can say $\hat{u}$ and after that all you need to do is say okay minimize J_{PDE} plus J_{BC} plus J initial conditions

So, notice there is data given here initial condition is for all x for t equal to zero. this is left side x, x equal to zero for all t right hand side nothing is given and, on the top, you have some other conditions. So, you can give all sorts of data points and just ensure that this summation is minimized. The weights of the neural network w are updated as usual $w = w\text{-}\alpha \frac{\partial J}{\partial w}$, keep on updating it, you get the final set of w you just plot this here and you get a very nice elegant solution for this problem. Now this tended to have certain problems.

(Refer Slide Time: 27:14)



ELSEVIER journal homepage: www.elsevier.com/locate/neucom

Physics Informed Extreme Learning Machine (PIELM)–A rapid method for the numerical solution of partial differential equations

Vikas Dwivedi*, Balaji Srinivasan

Heat Transfer and Thermal Power Laboratory, Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai 600036, India

ARTICLE INFO

Article history:
Received 27 June 2019
Revised 24 September 2019
Accepted 23 December 2019
Available online 27 December 2019

Communicated by Prof. Zidong Wang

Keywords:
Partial differential equations
Physics informed neural networks
Extreme learning machine
Advection-Diffusion equation

ABSTRACT

There has been rapid progress recently on the application of deep networks to the solution of partial differential equations, collectively labeled as Physics Informed Neural Networks (PINNs). In this paper, We develop Physics Informed Extreme Learning Machine (PIELM), a rapid version of PINNs which can be applied to stationary and time-dependent linear partial differential equations. We demonstrate that PIELM matches or exceeds the accuracy of PINNs on a range of problems. We also discuss the limitations of neural network-based approaches, including our PIELM, in the solution of PDEs on large domains and suggest an extension, a distributed version of our algorithm – DPIELM. We show that DPIELM produces excellent results comparable to conventional numerical techniques in the solution of time-dependent problems. Collectively, this work contributes towards making the use of neural networks in the solution of partial differential equations in complex domains as a competitive alternative to conventional discretization techniques.

1. Introduction

lected locations inside the domain and on the boundary. The neural network is used as the basis function to approximate the so-

And I will just briefly end this section of how inverse problems as well as forward problems are solved using a couple of papers that were published in by a PhD Student at IIT Madras Vikas Dwivedi. this is a good paper called physics informed extreme learning machine. the idea was using a shallow network and this is called an PIELM structure you fix the weights here but change the weights at the output and it turns out that this works really well for linear equations.

(Video Starts: 27:46) So we were able to solve for fairly complex domains very quickly PINN tends to be a little bit slow otherwise compared to conventional methods. so even in points like this you can simply put points around as you can see this is a map of Australia and this is a star usually if you use if any of you are familiar with finite difference or finite volume you actually need good machine here whereas PINN does not require any machine.

Another idea by the same student who has now of course finished this PhD was to use distributed learning machines which is somewhat similar to using a mesh that is you use different neural networks in different domains okay. So, you use one neural network here one other neural network here so on and so forth and they added an extra interface term. So, you will see this

here if this is a linear equation, we have one such equation sitting within your assignment also. So, you will see this is distributed. you can also do time marching step by step from one step to the other again this is not going to come in the exam or in the assignment. this is just to show you some advances that have happened here or maybe some of these advances might be obvious to you.

So, typically, as is shown here J sum usually the PDE loss plus some regularization loss and we added at the interfaces to ensure some continuity. **(Video Ends: 29:27)**

So overall, the PINN approach unlike the surrogate model approach or unlike the usual forward model approach that we have seen so far, lets you solve inverse problems in one shot. it solves inverse problem as well as forward problem in one shot as I showed you earlier on in this video. So, the method is extremely powerful and I expect it is going to become even more powerful in the years to come. So, I hope you found this discussion on physics informed neural networks over these last three four videos useful. please do let me know in the forum in case you have some questions since it is a research topic some of what I said might not be particularly or always clear. So, I will be happy to answer those questions. otherwise, I will see you in the next week. Thank you.