MA 202

Maths Project: Partial Differential Equations

Solving PDE using Neural Network

Gaurav Joshi	Adit Kaushik	Aditya Deshmukh
21110065	21110010	21110014

Abstract:

Deep learning has exploded as a field in recent years and has revolutionized collection and analysis of data.Neural networks are an integral port of deep learning models are used in various domains of computer and data science. In this project we aim to use neural networks and train them to solve partial differential equations (PDE's) using supervised learning.

Methodology:-

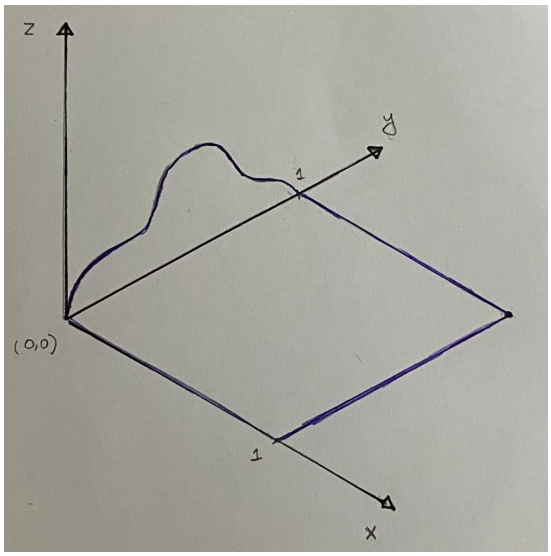
Neural networks that are deep enough can be approximated by any function . We will use this fact and train the model for boundary conditions. For making the model follow the partial differential equation. We will also include the tie square of the PDE as a loss. This work will come under the broad framework of PININ (Physics-Inspired Neural Networks method). We will compare our solution with some common methods of solving PDE's i.e Analytic and Numerical methods.

I) Solving 2-D Laplace Equation :- Analytic Method

Laplace Equation:- $\nabla^2 z=0$

$$\frac{\partial^2 z}{\partial y^2}+\frac{\partial^2 z}{\partial y^2}=0 \qquad \text{_____(1)}$$

. Therefore, it is a very useful equation applied mathematics occurring in potential theory, waves, acoustics quantum physics.



We will be using drichlet boundary conditions:-

membrane equation : Z(x,y)

Z(0,y)=f(y) ____(1)

Z(x,0)=0 ____(2)

Z(1,y)=0 ____(3)

Z(x,1)=0 ____(4)

Using method of separation of variables:-

$$Z = X(x)Y(y)$$

$$\frac{Y\partial^2 X}{\partial x^2} + \frac{X\partial^2 Y}{\partial y^2} = 0 \qquad \text{and} \qquad \frac{1}{X}\frac{\partial^2 X}{\partial x^2} + \frac{1}{Y}\frac{\partial^2 Y}{\partial y^2} = 0$$

As both parts are functions of separate variables, we can take them as constants , let

$$\frac{1}{X}\frac{\partial^2 X}{\partial x^2} = +\lambda \qquad \frac{1}{Y}\frac{\partial^2 Y}{\partial y^2} = -\lambda$$

Which on solving gives

$$Y(y) = A\cos\sqrt{\lambda}y + B\sin\sqrt{\lambda}y$$

and

$$X(x) = c_1 e^{\sqrt{\lambda}x} + c_2 e^{-\sqrt{\lambda}x}$$

net z is equal to

$$Z(x,y) = (A\cos\sqrt{\lambda}y + B\sin\sqrt{\lambda}y)\left(c_1 e^{\sqrt{\lambda}x} + c_2 e^{-\sqrt{\lambda}x}\right)$$

By BC(2) (boundary condition)

$$Z(x,0) = 0$$

$$(A + B^*0)\,X(x) = 0 \Rightarrow A = 0$$

By BC(4)

$$Z(x,1) = 0$$

$$(B\sin\sqrt{\lambda})\left(c_1 e^{\sqrt{\lambda}x} + c_2 e^{-\sqrt{\lambda}x}\right) = 0$$

B cannot be equal to zero or else, we get a trivial solution.

$B \neq 0$ $\sqrt{\lambda} = n\pi$ $\lambda = \pi^2 n^2$

Eigenvalues

By BC(3)

$$z(1, y) = B\sin(n\pi y)\left[c_1 e^{n\pi} + c_2 e^{-n\pi}\right] = 0$$

$c_1 e^{n\pi} + c_2 e^{-n\pi} = 0$ which on solving $\frac{c_2}{c_1} = -e^{2n\pi}$ gives

$$Z(x, y) = B\sin(n\pi y)\left[c_1 e^{n\pi x} + c_2 e^{-n\pi x}\right]$$

$$= B\sin(n\pi y)\left[c_1 e^{n\pi} e^{n\pi(x-1)} + c_2 e^{n\pi} e^{n\pi(1-x)}\right]$$

$$= -B\sin(n\pi y)\left[2c_1 e^{n\pi}\right](\sinh(n\pi(1-x)))$$

$$Z(x, y) = B(\sin(n\pi y))(\sinh(n\pi(1-x)))$$

By BC(1) (boundary condition)

$Z(0, y) = f(y)$ so we can represent the boundary value function as

$$\sum B_r \sin(ry)(\sinh(n\pi(1-x))) = f(y)$$

$$f(y)\sin(n\pi y) = \sum_{r=1}^{N} (B_n \sinh r\pi) \cdot (\sin r\pi y)(\sin n\pi y)$$

Using fourier series expansion:

$$\int_0^{+1} f(y)(\sin n\pi y)\, dy = B_n \sinh(\pi n) \int_0^{+1} \sin^2(n\pi y) dy$$

Putting $n\Pi y = t$

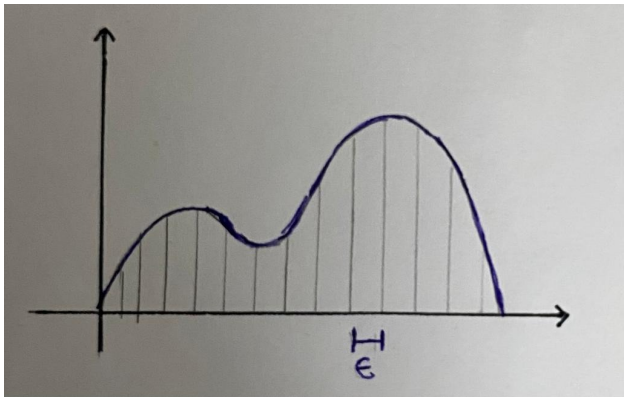
$$\frac{1}{n\pi}\int_0^{n\pi} \sin^2(t)dt = \frac{1}{n\pi}\int_0^{n\pi}\left(\frac{1-\cos 2t}{2}\right)tdt \quad = \frac{1}{n\pi}\left[\frac{1}{2}(n\pi - 0)\right] = 1/2$$

Which finally leads us to

$$B_n = \frac{2}{\sinh(n\pi)}\int_0^{+1} f(y)\sin(n\pi y)dy$$

There are two methods obtaining B_n

1]Analytic method :- This method involves numerically calculating the integral of function As we know the function, we keep an array with the values of $f(y)$ separated by a small value of $\epsilon = 10^{-3}$, This array can be made only once and can be repeatedly used to get the integral. For each $B(n)$, we can keep a similar array say $sin(n\pi y)$ and multiply the elements of both, add them and finally multiply the sum with our parameter ϵ.



This is an approximate method, but gets us our integral for any function.

2]Exact method - Calculate the integral for the function

In this problem; we have use the functions $f(y) = \sin(10\pi y)$ and $f(y) = y(1 - y)(4y + 0.1)$

Numerical Method :-

To solve such partial derivatives, we can also use many tools of numerical analysis. In this particular paper, we will use the finite differences method.

Consider any function u:

$$u_{x+\epsilon} = u_x + \epsilon u'_x + \frac{\epsilon^2}{L^2} u''_x + \cdots \quad \text{-(1)}$$

$$u_{x-\epsilon} = u_x - \epsilon u'_x + \frac{e^2}{2} u''_x + \cdots \quad \text{-(2)}$$

Neglecting powers of ϵ higher than 2 .

(1) –(2)

$$u_{x+\epsilon} - u_{x-\epsilon} = \partial \epsilon u'_x$$

$$u'_x = \frac{u_{x+\epsilon} - u_{x-\epsilon}}{2\epsilon} \text{ (valid up to } \epsilon^2 \text{)}$$

(1) + (2)

$$u_{x+\epsilon} + u_{x-\epsilon} = 2u_x + \epsilon^2 u''_x$$

$$u''_x = \frac{u_{x+\epsilon} + u_{x-\epsilon} - 2u_x}{\epsilon^2}$$

for $u(x, y)$

$$\frac{\partial u^2}{\partial x^2} = \frac{u_{x-\epsilon,y} + u_{x-\epsilon,y} - \partial u_{x,y}}{\epsilon^2}$$

$$\frac{\partial u}{\partial x} = \frac{u_{x+\epsilon,y} - u_{x-\epsilon,y}}{2t}$$

$$\frac{\partial u^2}{\partial y^2} = \frac{u_{x,y+\epsilon} + u_{x,y+\epsilon} - 2u_{x,y}}{\epsilon^2}$$

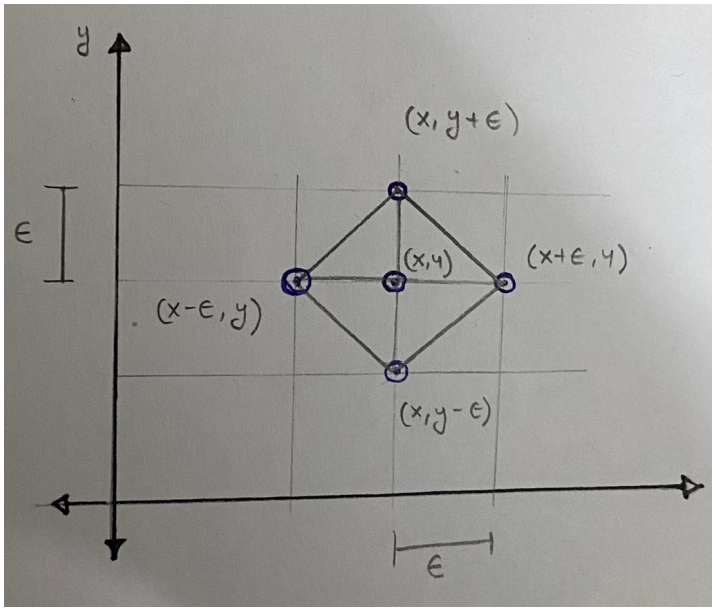
$$\frac{\partial u}{\partial y} = \frac{u_{x,y+\epsilon} - u_{x,y-\epsilon}}{2\epsilon}$$

These can be used to get approximate solutions for PDE's

$$\varepsilon^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$$

$$u_{x,y} = \frac{1}{4}\left(u_{x+\varepsilon,y} + u_{x-\varepsilon,y} + u_{x,y+\varepsilon} + u_{x,y-\varepsilon}\right)$$

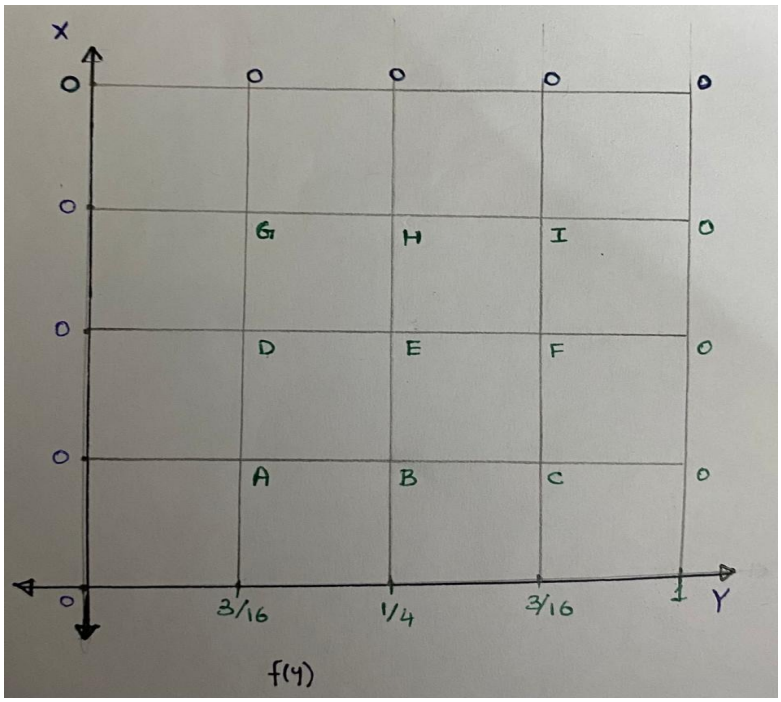
In other words,



Any point is equal to the average of all the points around it in the 2 directions.

We can form an n x m grid (Here, we show the 3X3 grid and approximate the solutions)

Let us take $f(y) = y(1 - y)$



$$U_A = \frac{1}{4}(U_D + 0 + U_B + 3/10)$$
$$U_B = \frac{1}{4}(U_A + U_C + U_E + V_4)$$
$$U_C = \frac{1}{4}(U_B + U_F + 0 + 3/16)$$
$$U_D = \frac{1}{4}(0 + U_C + U_E + U_P)$$
$$U_F = \frac{1}{4}(U_H + U_b + U_F + U_B$$
$$U_F = \frac{1}{4}(U_F + U_I + 0 + U_C)$$
$$U_A = \frac{1}{4}(0 + 0 + U_1 + U_0)$$
$$U_H = \frac{1}{4}(0 + U_C + U_2 + U_E)$$
$$U_I = \frac{1}{4}(0 + 0 + U_H + U_F)$$

This will form n*m linear equations with n*m variables. This is a very sparse matrix and there are specific methods to solve it, but we will use an iterative method to solve this equation.

Algorithm for solving the sparse matrix :

1) We will give value 0 to all points of the matrix except the boundary points.
2) Then we will move from point matrix[1][1] or A() and change its value to ¼*(sum of neighbors) and will continue this till point matrix[n-1][m-1].
3) We will then iteratively repeat this cycle N times till sufficient accuracy is achieved. (Therefore, accuracy can be checked by ensuring that all the differences in value are below a threshold)

Using PINNS :

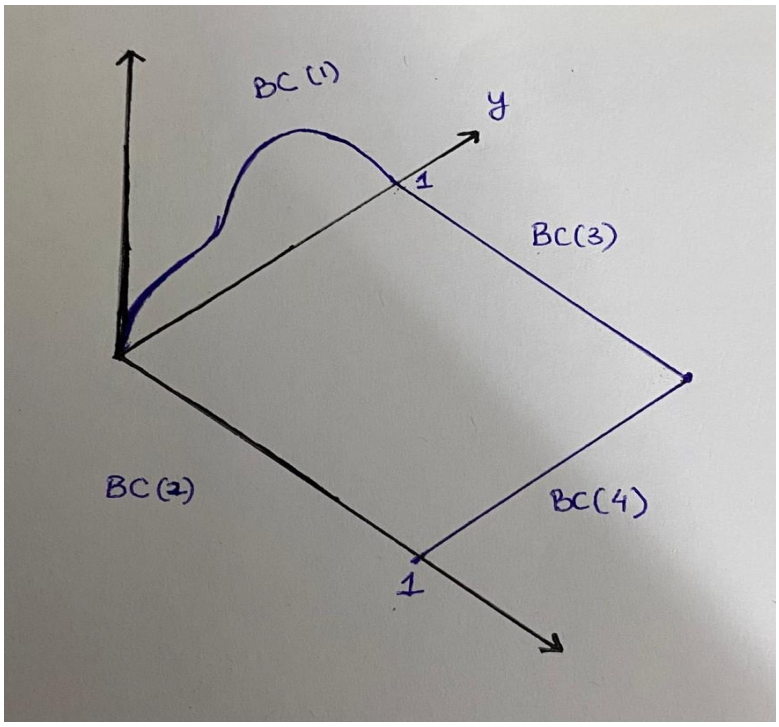
Physics Inspired Neural Networks or (PINN'S) are networks which respect the physics of the system by incorporating the partial differential equation into the net.
For example:

If the problem has a PDE $\frac{dz}{\partial y} = c'\left(\frac{\partial' x}{\partial x^2} + \frac{\partial^2 z}{\partial y'}\right)$, then we take —

$$[L(x,y)]^2 = [\frac{\partial z}{\partial y} - c'(\frac{\partial' z}{\partial y'} + \frac{\partial' z}{\partial x'})]^2$$

as a loss feed it to function on multiple points and the neural network and feed it to the neural network.

We hove 3 types of inputs for any equation:
1) The residual : The loss of the PDE
2) Initial loss : - Loss due to initial value function
3) Boundary loss: Loss due to boundary condition



As the laplace equation has a dirichlet boundary condition ; we only have the residual and boundary loss.

We take N_b points for each of the four boundaries
BC(1) has Z(x,y) = f(y). So we take N_b points and form loss function
$f = 2[f(y) - N(0,y)]^2$

Hence, boundary loses = $\frac{1}{N\_b}\sum [B(x,y) - N(x,y)]^2$

where, B(x,y) is the boundary value and N(x,y) is the output from network

Residual loss = $\frac{1}{N\_r}\sum [L(x,y)]^2$

Important Points

1) To get the residual we used the technique of auto differentiation which available in tensorflow tape.

The tape calculates the value of z in the neural network with respect to x and y by making a graph. Then it uses backpropagating auto differentiation to get the gradient of z with respect to x and y.

Next we get the double derivatives z_xx and z_yy by calculating the gradient of z_x with respect to x and similarly for y.

We send the derivatives to the residual function and the calculate the gradient of loss with respect to the trainable variables of the network (that is the biases and the weights)

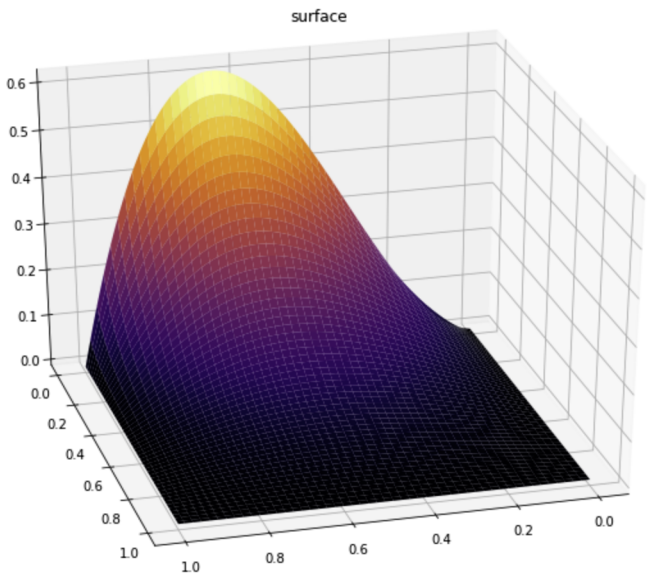
2) We put the $total\ loss\ = \lambda(Residual\ Loss)\ +\ (1\ -\ \lambda)(Boundary\ Loss)$

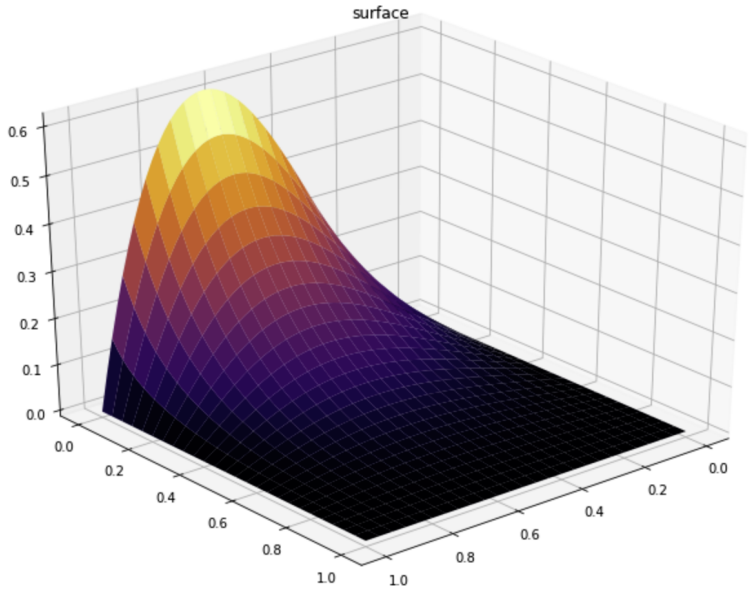
Where $\lambda\ =\ 10^{-5}$. This was necessary to make the boundary loss more weighted so that the boundary conditions are fit properly.

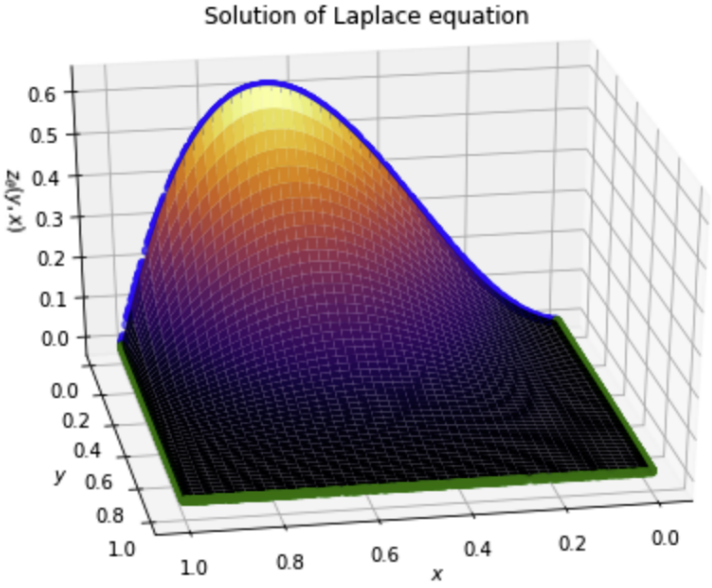
Comparisons :-

Here we make comparisons of solutions of the two laplace conditions obtained from the three methods

1) $Boundry\ Condition$: $x * (1 - x) * (4 * x + 0.1)$

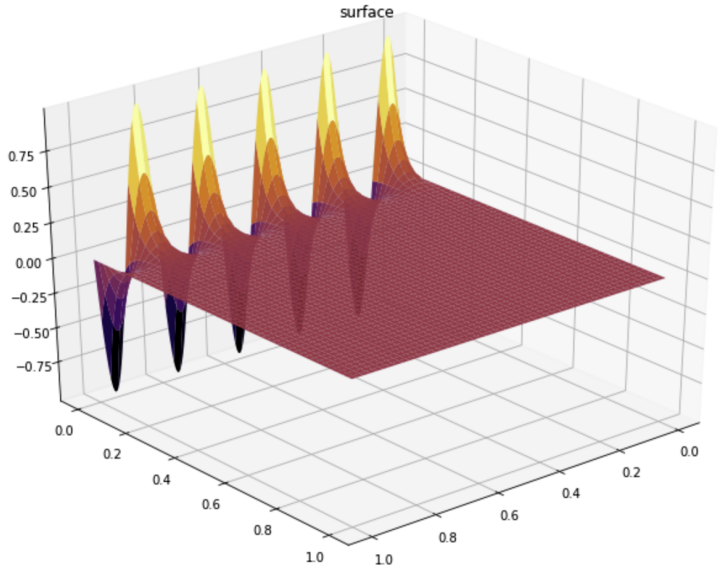


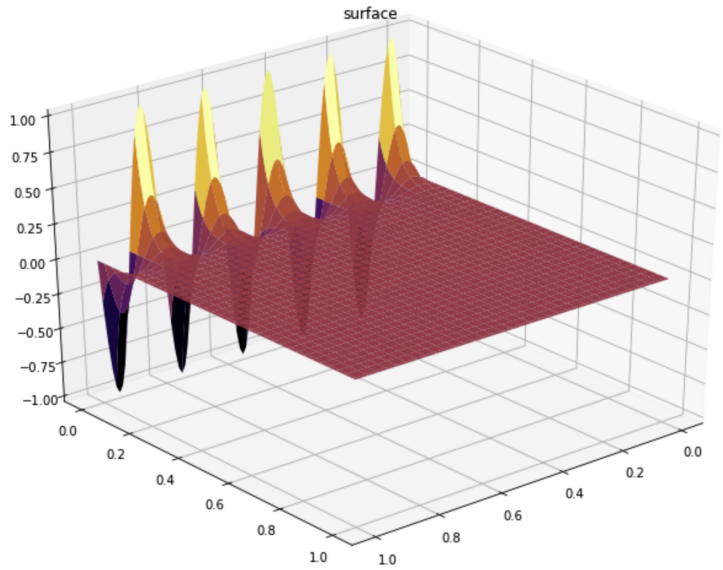


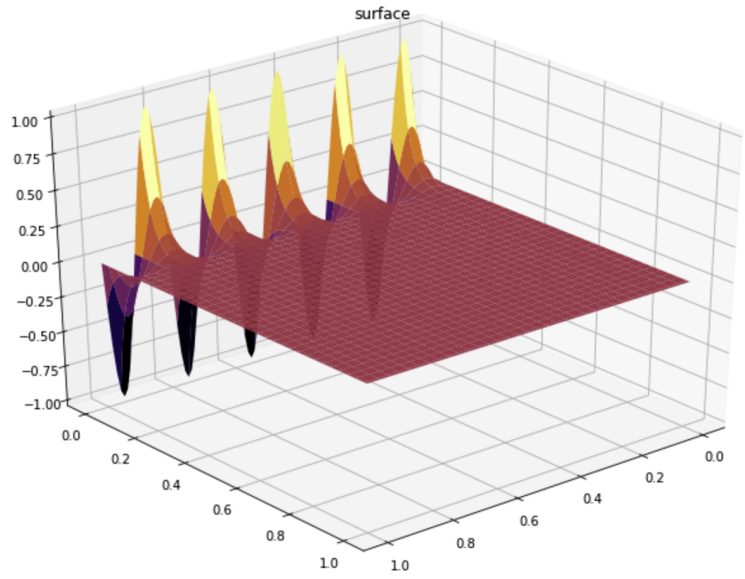


The first image is the solution using the **analytic method.**
The second image is the solution using the **numerical method**.
The third image is the solution using the **neural network method.**

2) $Boundry\ Condition$: $sin(10\pi x)$







The first image is the solution using the **analytic method**.
The second image is the solution using the **numerical method.**
The third image is the solution using the **neural network method.**

Conclusion :-

Using neural networks to approximate functions and boundaries is a central theme of machine learning and pattern recognition, so here we use them to simulate parietal differential equations by feeding the equation itself as a loss function to minimize.

This is a very useful technique which can be applied in all field where we have to solve partial differential equations, which is basically all fields of applied mathematics.

Using changes in parameters, depth of the neural network and type of architecture, the PINN method will show excellent results for any approximation problem.

Applications:-

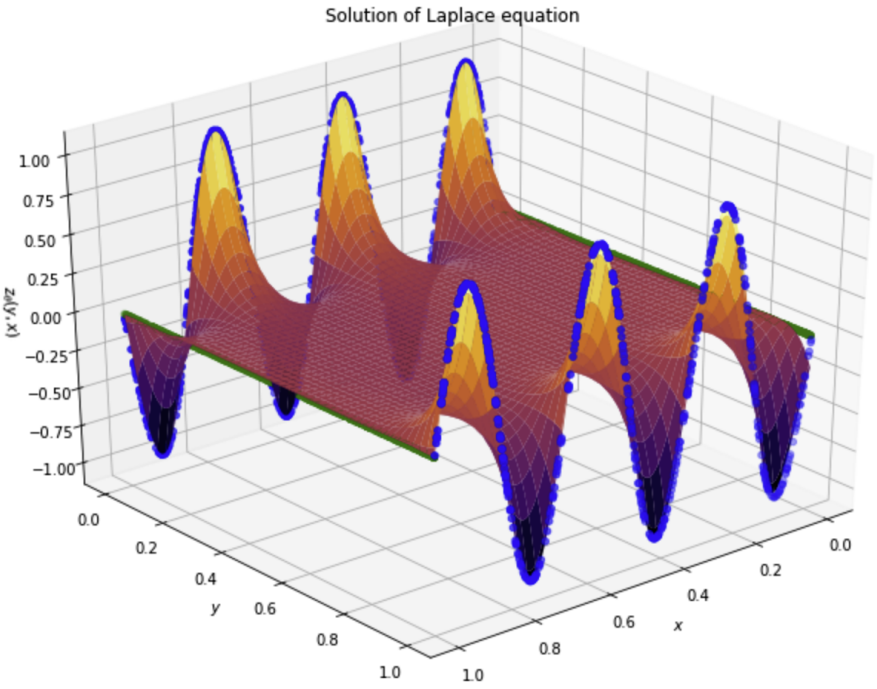
Here, we are applying the neural network to solve the laplace equation for boundary conditions which are more complicated to solve by using analytic or numberical methods.

Solving Retangular Laplace Equation:- With different boundary conditions

First Example:-

$Boundry\ Condition1:\ sin(6\pi x)$

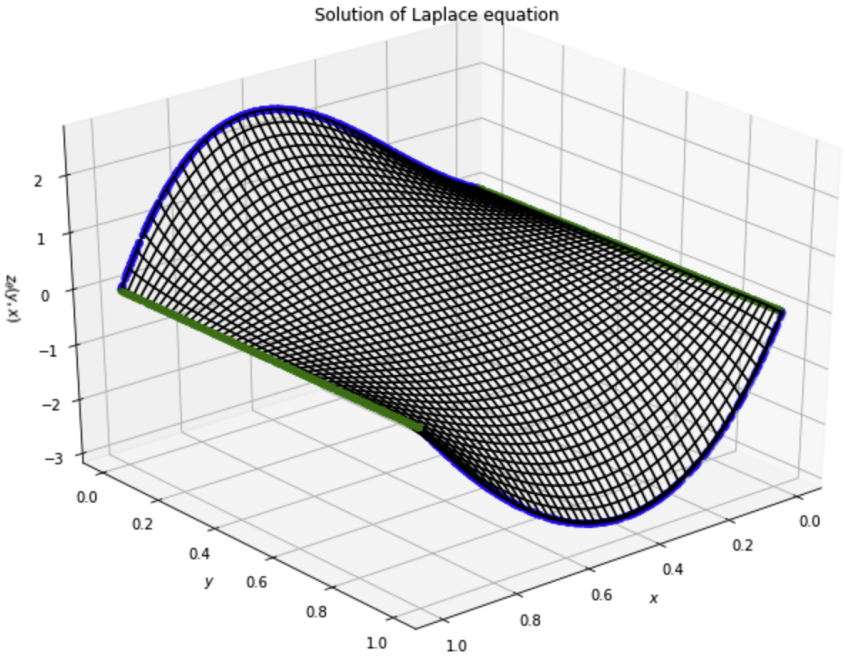
$Boundary\ Condition2:\ sin(-\ 6\pi x)$



Secound Example:-

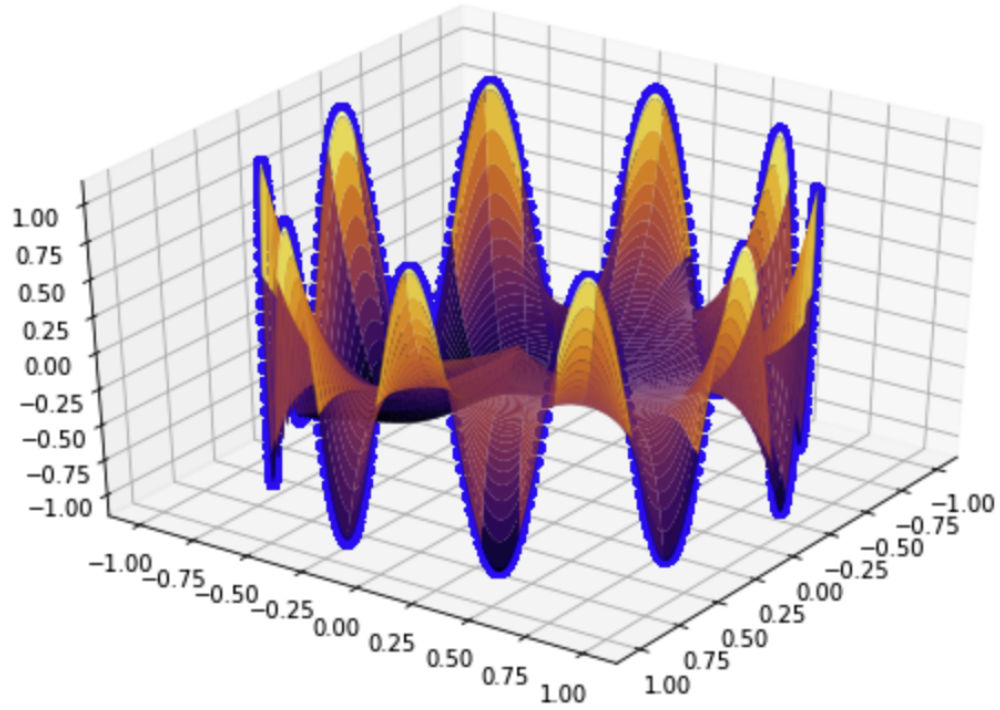
$Boundry\ Condition1:\quad 3\ *\ y\ *\ (1-y)\ *\ (4\ *\ y+1)\quad)$

$Boundry\ Condition2:\quad -\ 3\ *\ y\ *\ (1-y)\ *\ (4\ *\ y+1)$



Solving Polar Laplace Equation:- A drichlet boundary value problem
Here we have put a drichlet boundary condition as used put the boundary as a sin function which makes a complete, non discontinous boundary for value of phi.

$$Boundry\ Condition1: \quad sin(10\pi\phi)$$



GitHub Repository Link:

We have open sourced the source code filed on GitHub and it can be accessed from the following link:

https://github.com/Gaurav17Joshi/PDEs

Notes and References

1) Jan Blechshmidt, Oliver G Ernest, Three Ways to Solve Partial Differential Equations with Neural Networks

2) Zongyi Li*, Hongkai Zheng*, Anima Anandkumar, Physics-Informed Neural Operator for Learning Partial Differential Equations

3) Yue Lu and Gang Mei , A Deep Learning Approach for Predicting Two-Dimensional Soil Consolidation Using Physics-Informed Neural Networks (PINN)