上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

中科院暑期科研实习答辩

PINNs for Helmholtz Equations' Forward/Inverse Problems in Multiple Propagation Mediums

李云艾

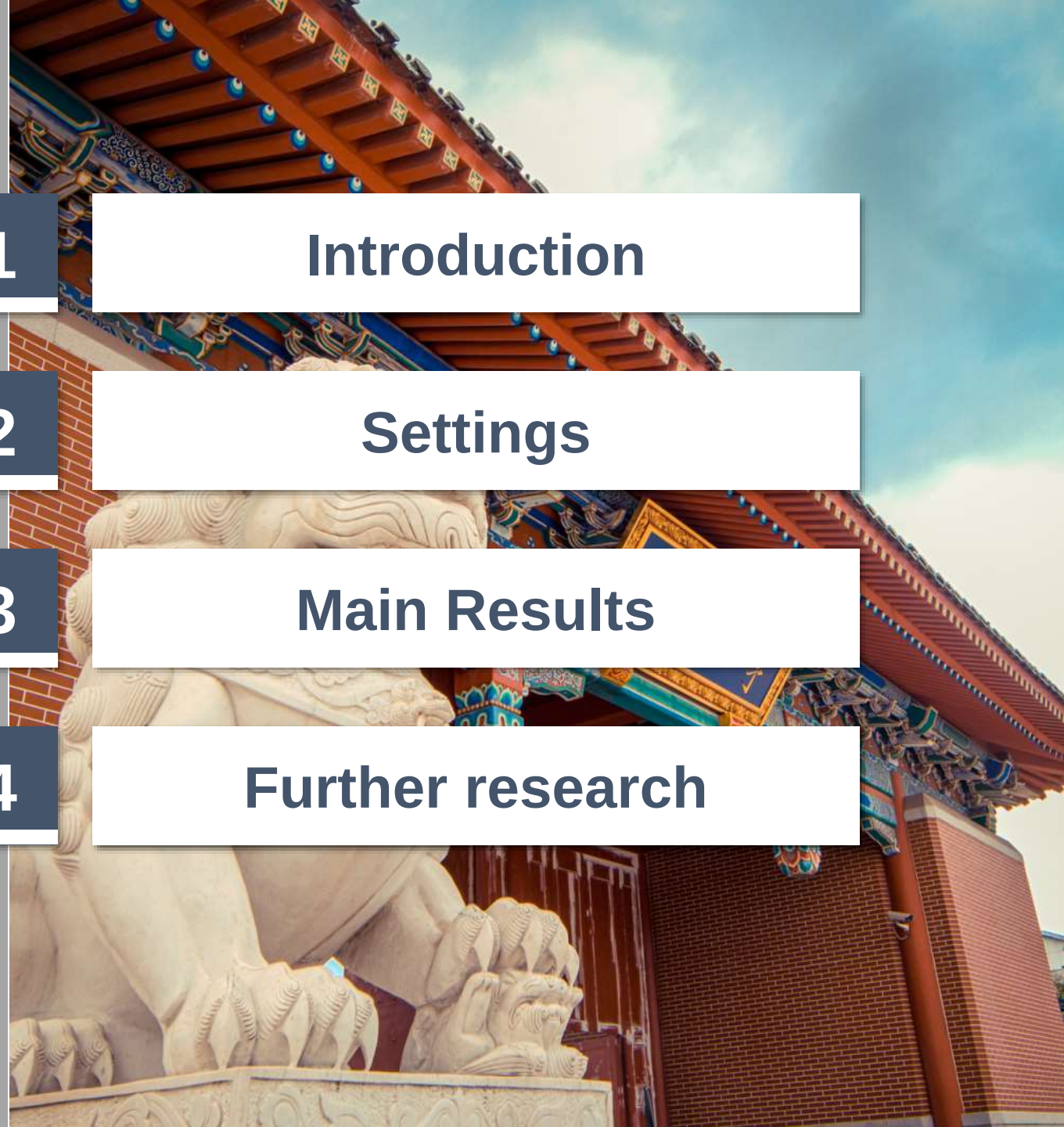
饮水思源 • 爱国荣校

Tips: This was originally a slide in Chinese for presentation, and I used auto translation to translate it into English. Sorry if there's any confusion.

SJTU

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录



01

Introduction

Research on the forward and inverse problems of Helmholtz's equation based on PINN.

PINN & Helmoholtz equations

PINN (Physics-Informed Neural Network):

Physical Informed Neural Networks (PINNs) are an approach that combines the principles of physics and neural networks for the task of solving physics problems and partial differential equations (PDEs).
While traditional methods for solving partial differential equations usually require discretized grids and the application of numerical methods, PINN is capable of sampling randomly over an arbitrary range. It utilizes the powerful fitting ability of neural networks and the back-propagation autoderivative function to approximate the solution of partial differential equations by adding a loss function that includes the form of the physical equations, boundary conditions, initial conditions, and so on.

Helmholtz equation:

The Helmholtz equation is an important partial differential equation that occurs frequently in physics and engineering. Named after the German physicist Hermann von Helmholtz, the equation describes the phenomenon of fluctuations and mathematical modeling of wave propagation.
The Helmholtz equation can be written:
$\nabla^2 u + k^2 u = 0$
Various types of fluctuation phenomena can be described, such as acoustic, electromagnetic, and water waves, etc. In this study, we replace the fixed parameter k^ 2with functions of different properties to solve the equations and invert the parameters.

02

Settings

Problem Setting

For function u defined on (0,1)×(0,1) $\mathrm{u} = (x + y)e^{-(x+y)}$, Consider:

$$u \times g(x, y) + \Delta u = f_g$$

Solve the above equation using PINNs and compare it with the true solution u. Also, if u is known, we can reverse-engineer g at the sampling points in the entire domain. It's straightforward to compute g for each point in the domain, but we can add Gaussian noise manually to observe the impact of noise.

Additionally, we can include precise values of u at (x,y)=(1,0) and (1/2,0), and reverse-engineer u and g across the sampling points in the entire domain (with f g known). Observe the error in the solutions for uu and g in this case.

For the function g, we conducted the following three trials:$g\equiv1$; g is a Gaussian function centered at (0.5,0.5) (with three different σvalues); g is a piecewise constant function that is discontinuous in the domain (with three different values for each segment).

PINN model setup

- Input: sampling points (x, y) (the closer to (0,0), the steeper the function, so the more intensive the sampling)
- Output: predicted value u (positive problem); g or (u,g) (negative problem) The specific PINN model is shown on the left below.

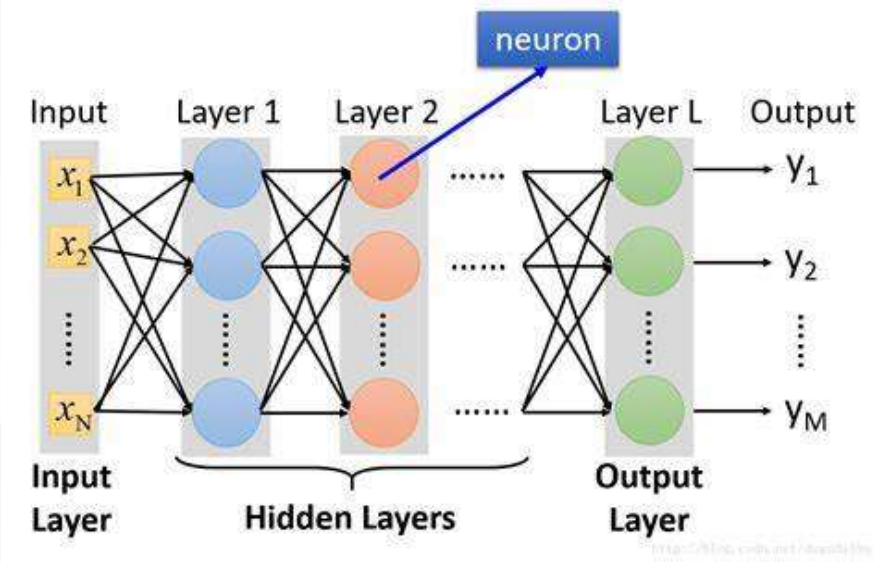
```python
# 定义PINN模型
class PINN(nn.Module):
    def __init__(self, layers):
        super().__init__()
        self.layers = nn.ModuleList(layers)

        '''对权重进行Xavier初始化,使每一层的训练过程在一个相对合理的范围内进行,
        防止梯度发散难收敛。 Xavier初始化会选择一个符合取决于输入和输出节点的个数
        的正态分布的随机数来初始化权重矩阵中的每个元素'''

        for layer in self.layers:
            if isinstance(layer, nn.Linear):
                init.xavier_uniform_(layer.weight)

    def forward(self, x):
        for i, layer in enumerate(self.layers):
            x = layer(x)
            if isinstance(layer, nn.Linear):
                # 将第1、2和3个线性层的激活函数设置为sinusoidal
                if i == 1 or i == 2 or i == 3:
                    x = sinusodial_activation(x)
        return x

# PINN模型  层数及神经元个数
layers = [nn.Linear(2, 40),
          nn.Linear(40, 40),
          nn.Linear(40, 40),
          nn.Linear(40, 1)
          ]
```



Loss function: $\left|\left|\mathrm{u}\cdot g(x,y)+\Delta u-f_g\right|\right|_2+$ $boundary\ condition$ (from real solution)

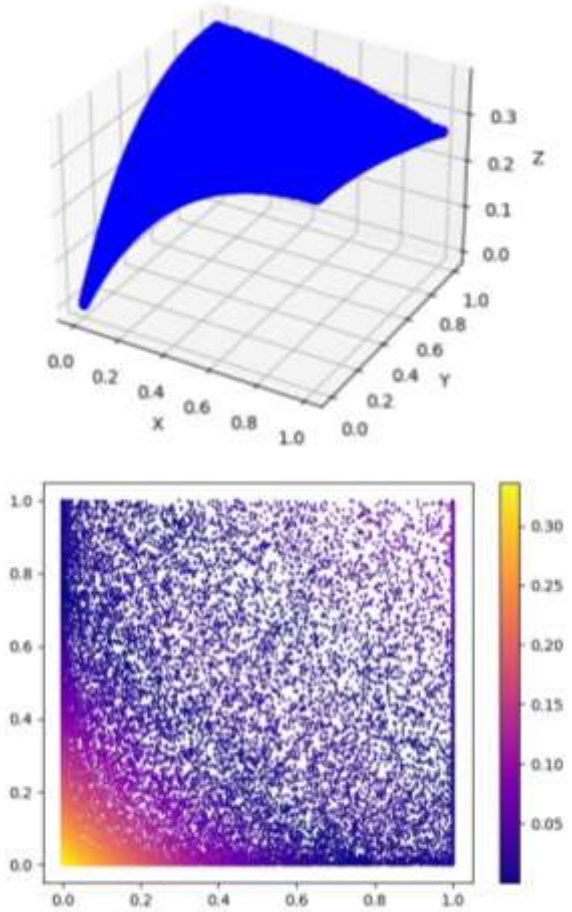
Optimizer: Adam+SGD

03

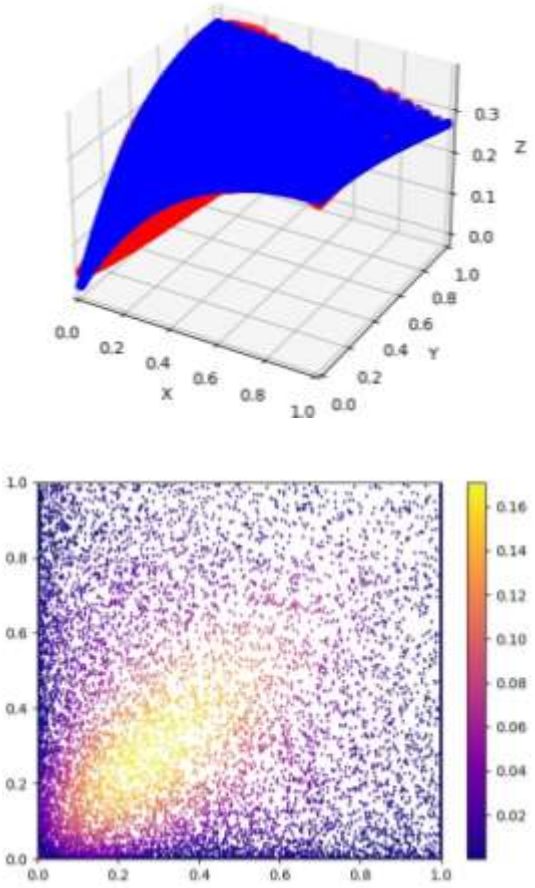
Main Results

Performance of PINN solving the positive problem with different g

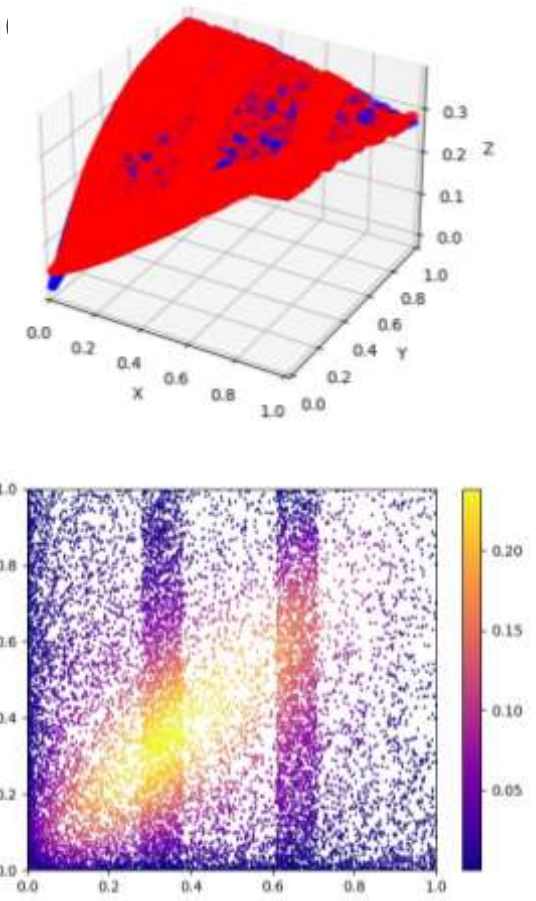
$g \equiv 1$:



Gaussian functions:



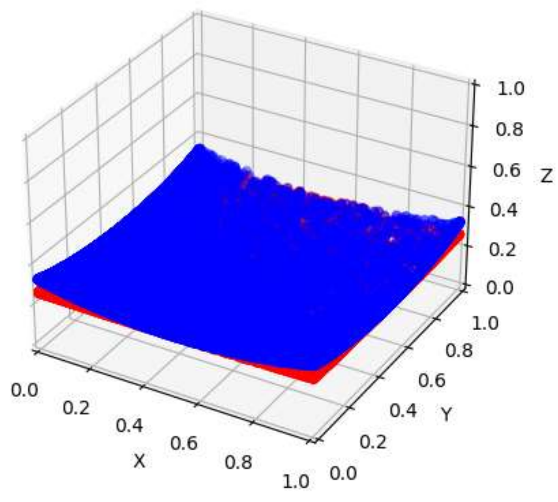
Segmentation fun

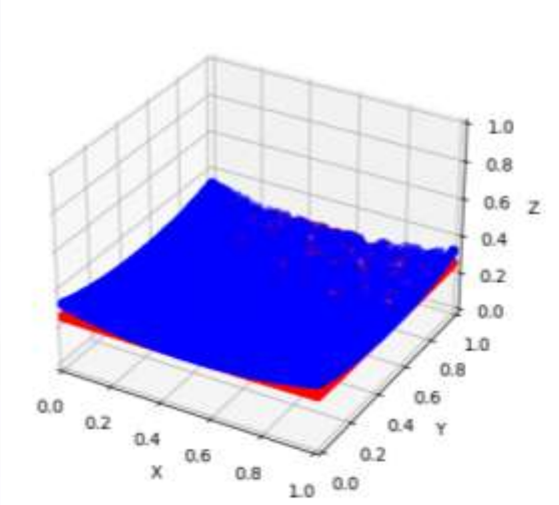


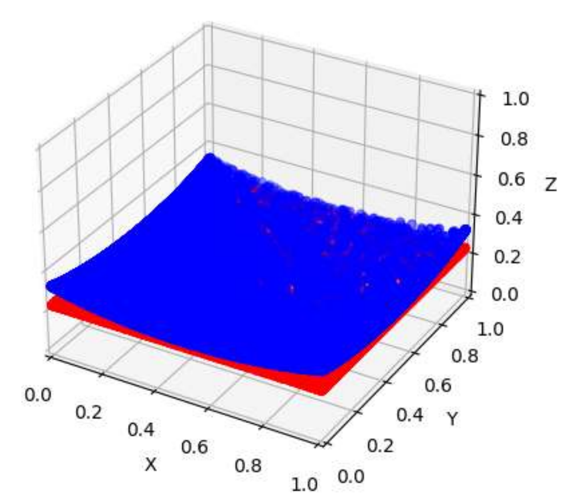
The top image shows the true values in blue and the fitted values in red; the bottom image shows a heat map of the error values at the sampling points.

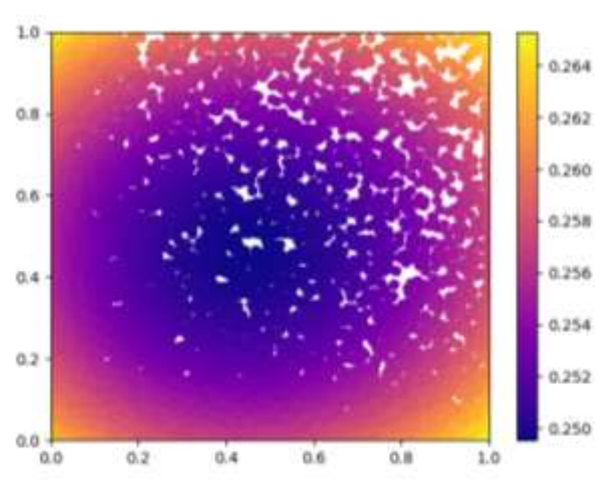
Inverse problem: invert only the g-value and take into account the effect of noise.

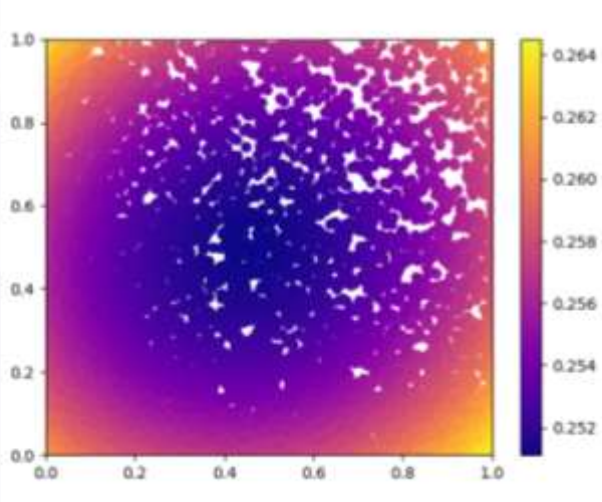
- Example: g= $\frac{20\times\exp\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)}{\sqrt{2\pi}}$ The lower image shows the heat map at the sampled point values.

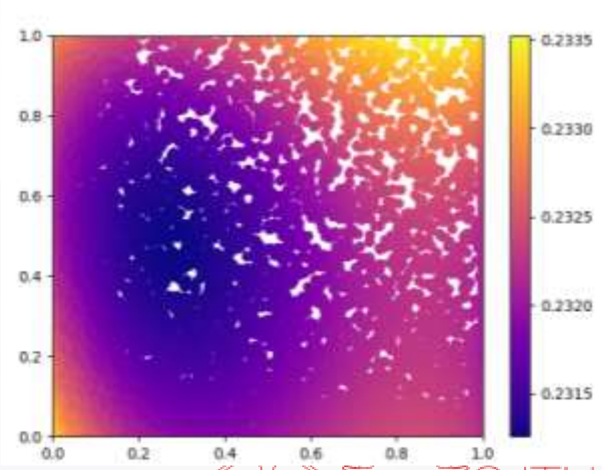












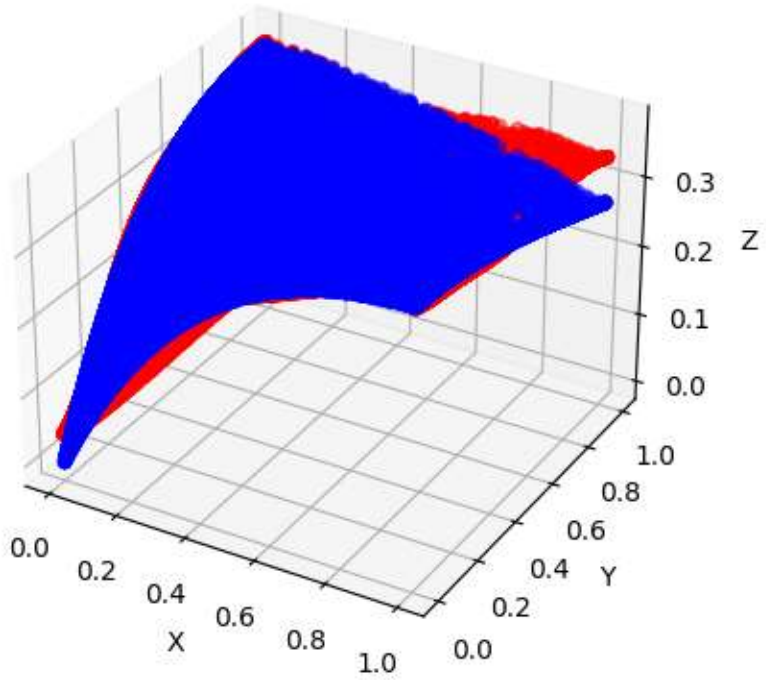
No noise: MSE= 0.000786

σ ^ 2= 1e^(- 3) MSE= 0.000821

σ ^ 2= 1e^(- 1) MSE= 0.001933

Simultaneous inversion of u, g: constant field

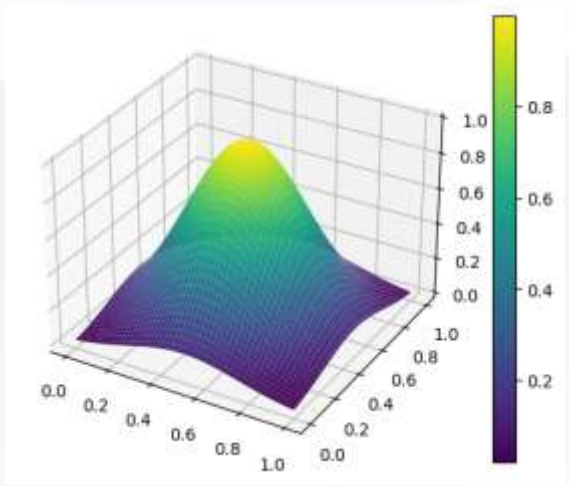
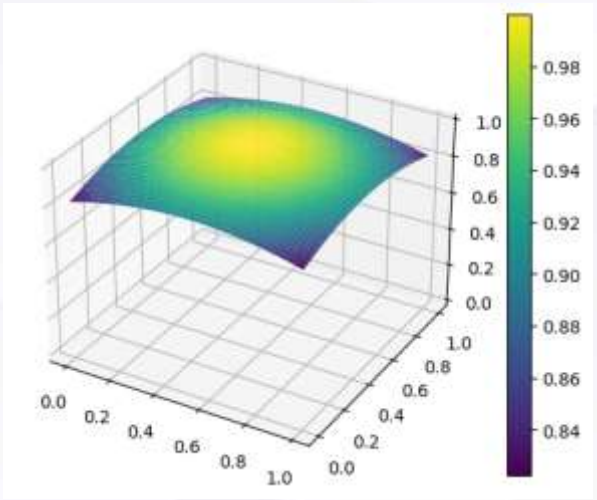
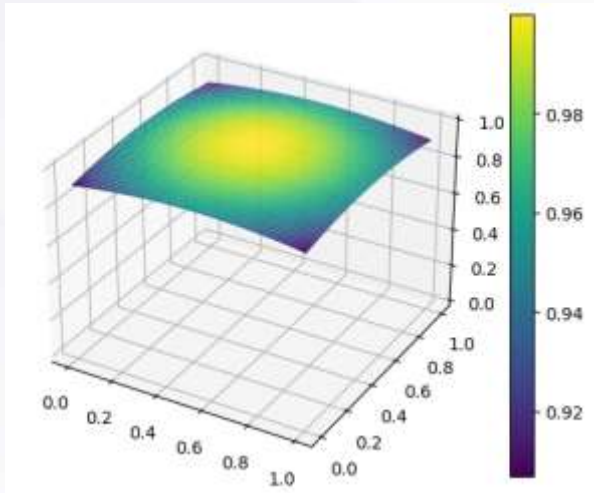
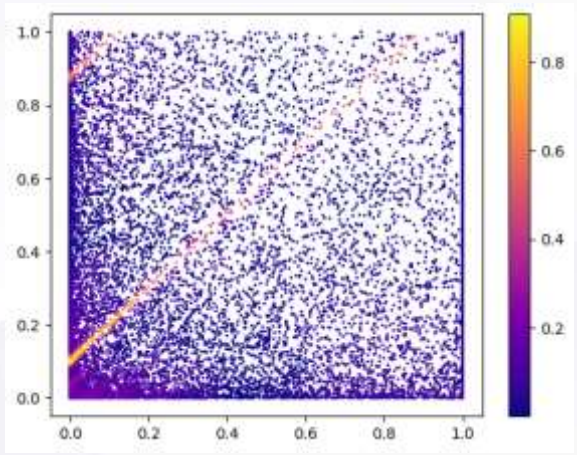
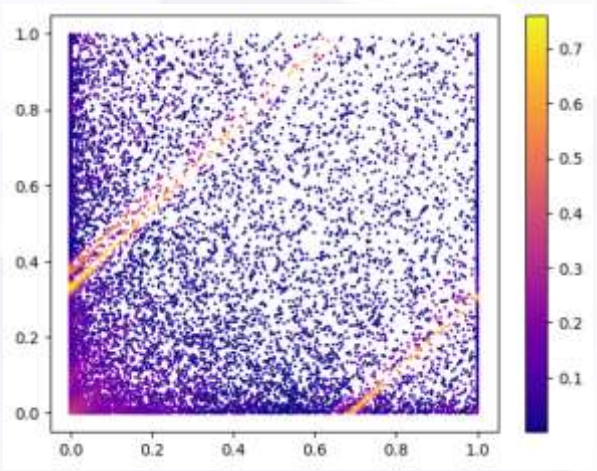
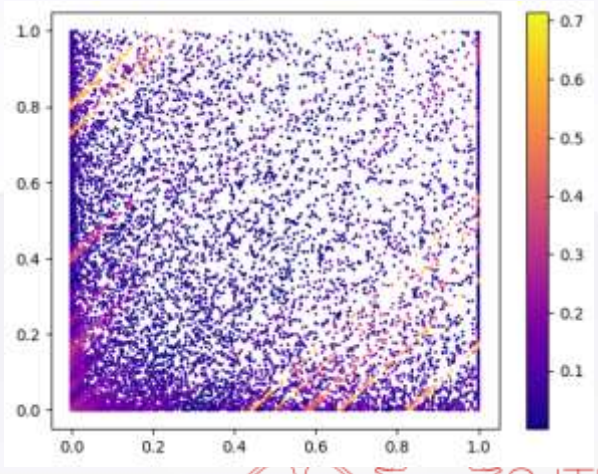
- Here, g is artificially set to a tensor of 1× 1for training, instead of predicting g values at all sampled points.




```
Epoch: 9990 loss:  22.35382843017578
Epoch: 9991 loss:  22.35382843017578
Epoch: 9992 loss:  22.35382843017578
Epoch: 9993 loss:  22.35382843017578
Epoch: 9994 loss:  22.35382843017578
Epoch: 9995 loss:  22.35382843017578
Epoch: 9996 loss:  22.35382843017578
Epoch: 9997 loss:  22.35382843017578
Epoch: 9998 loss:  22.35382843017578
Epoch: 9999 loss:  22.35382843017578
Predicted values of v:
0.9999884963035583
```

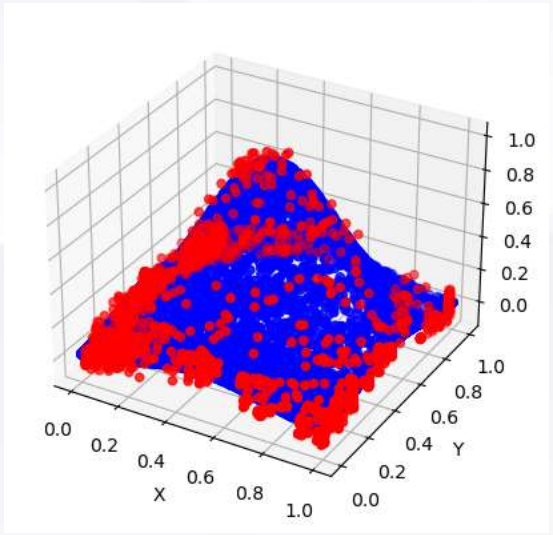
Simultaneous inversion of u, g: Gaussian function field

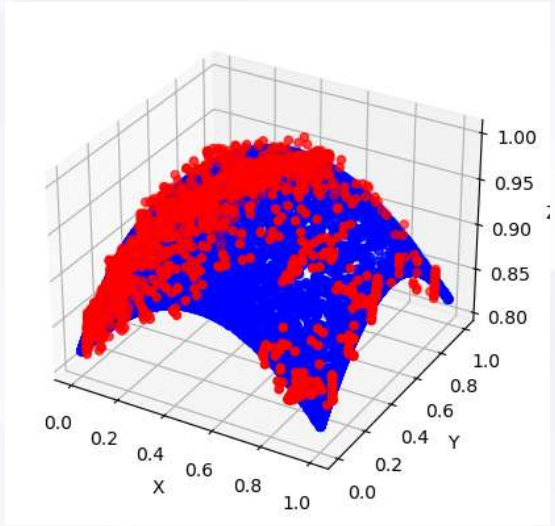
- The g-value is set to a tensor of the same size as the sampled points for training.

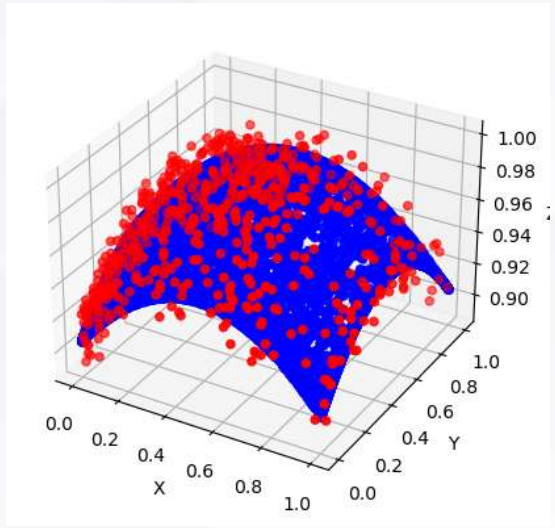







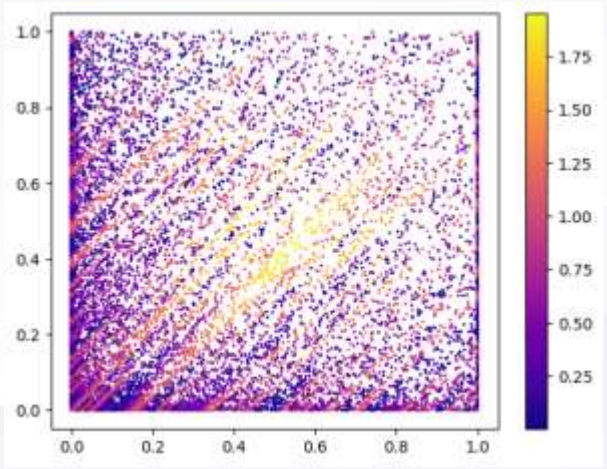
Below is a heat map of the error in the predicted value of u

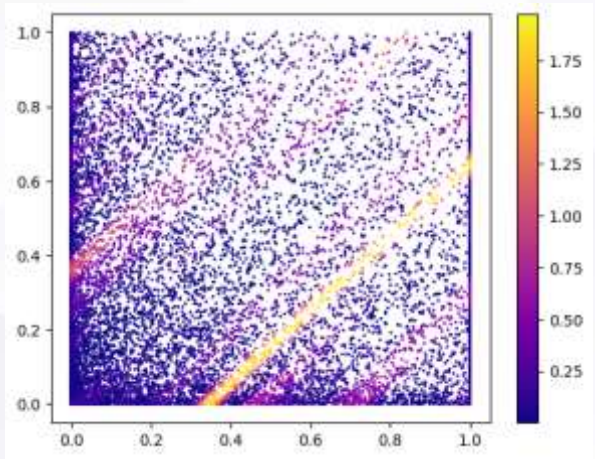
A closer look at g: where do we have better solution accuracy?

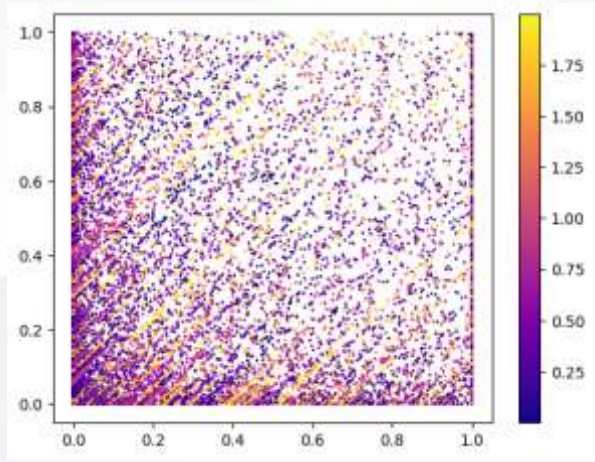










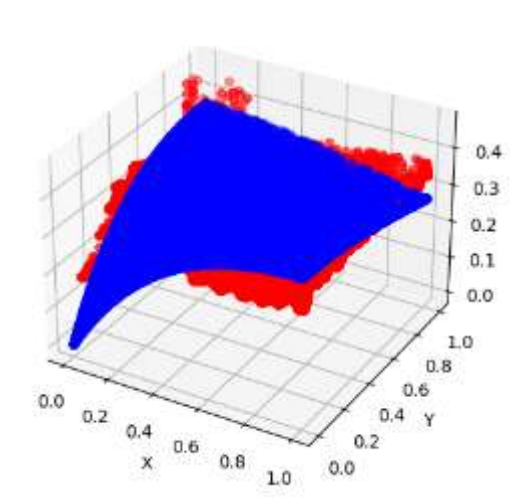


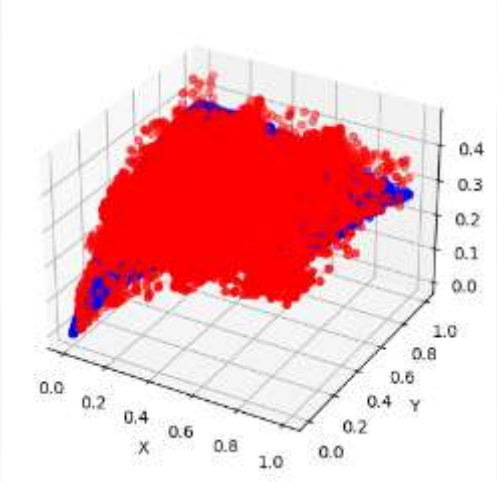
From left to right, the values of σ^ 2are 0.05, 1, 2 The red points in the upper image are the sampling points with fitting errors less than 0.02, and the blue ones are the exact values.

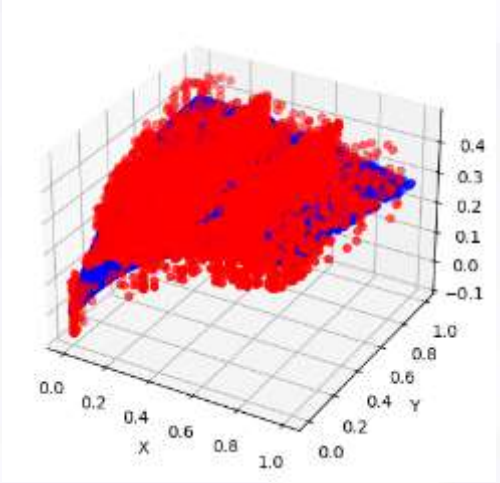
Below is a heat map of the error values of g. (Tips: the axes of the first row of graphs have been scaled down for a better view of the fit, and are not to the original scale).

Simultaneous inversion of u, g: discontinuous fields

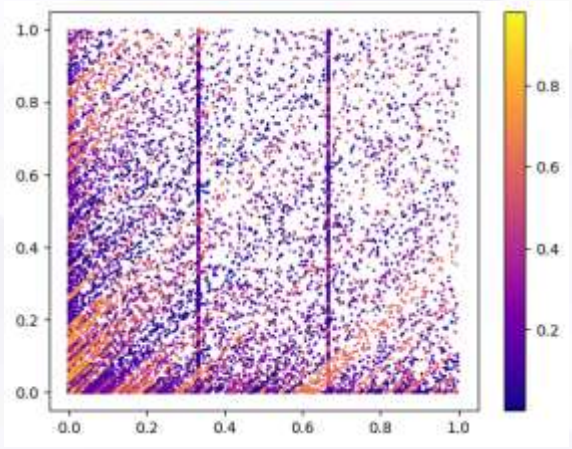
- In many papers, the discontinuities are convolved for the smoothness of the function, but what if they are not? In order to better learn the values around the discontinuities, the number of sampling points around them is increased.

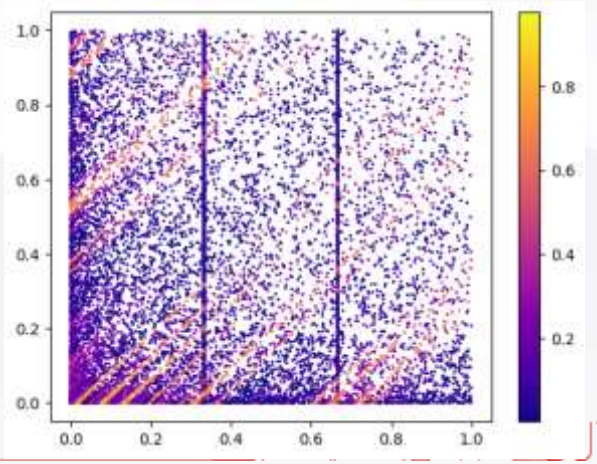






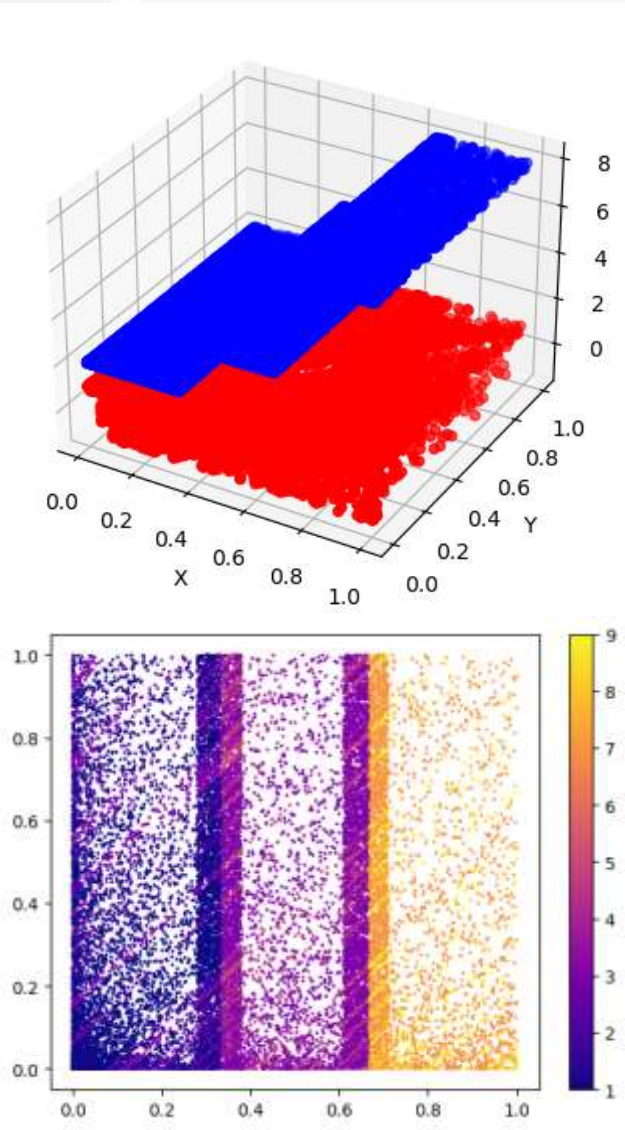
$$g=\begin{cases} 2\ (0.2;\ 0.1) & x \in (0,1/3) \\ 4\ (0.4;\ 0.2) & x \in (1/3,2/3) \\ 8\ (0.8;\ 0.3) & x \in (2/3,1) \end{cases}$$

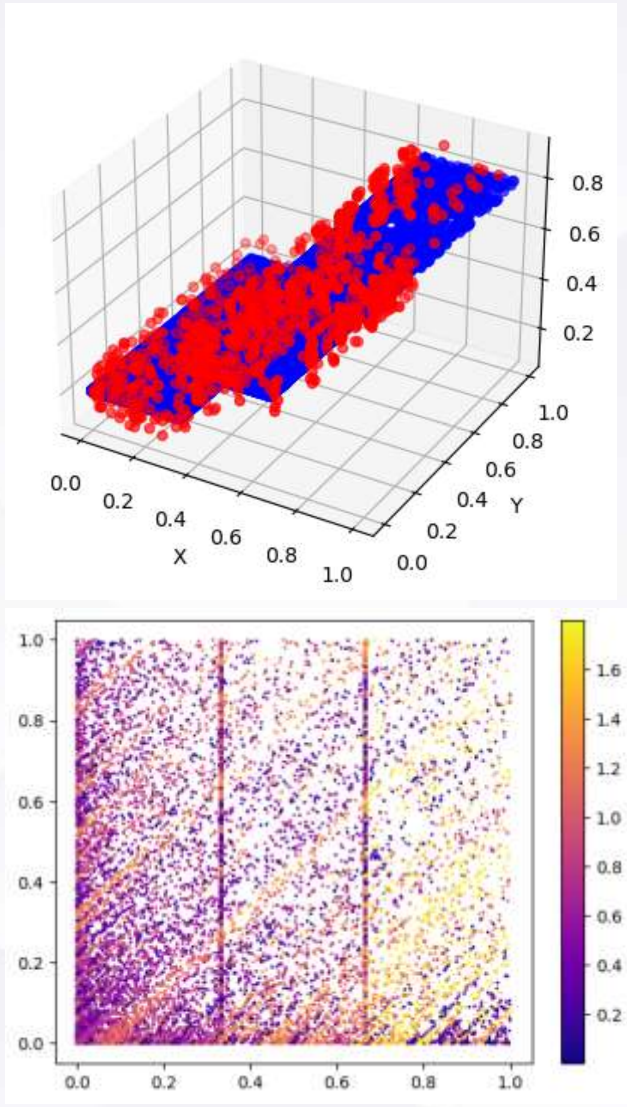


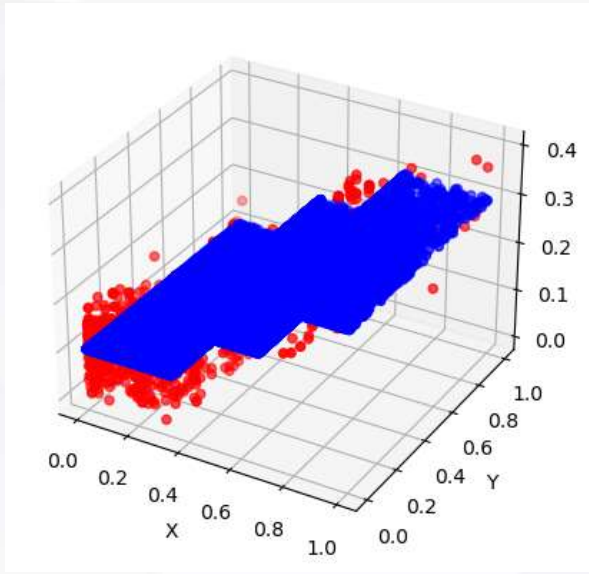


The top graph shows the predicted value of u in red, the actual value in blue, and the bottom graph shows a heat map of the error value of u.

Inverse Problem Results







(Tips: in order to better observe the fitting situation, the axes of the first row of graphs have been scaled down, not the original scale) In the three images of the first row, the sampling points of the first graph are not filtered, the sampling points of the second graph have an error between plus and minus 0.02, and the sampling points of the third graph have an error between plus and minus 0.1.

Summary

- For forward problems:

Since only a one-dimensional tensor needs to be predicted, the choice of parameters generally has good convergence. Increasing sampling at steep or discontinuous areas of the parameters or solutions can lead to better convergence.

- For inverse problems:

The difficulty of inversion is inversely proportional to the dimensionality of the tensor (inverting gg alone is easier than inverting both g and u; when inverting both g and u, training g as a 1×1 tensor is easier compared to training uu and $g$simultaneously with the same shape as the sampling points).The difficulty of inversion is related to the steepness, continuity, and the scale of the function's value range relative to the sampling range. However, it is not simply that smoother functions are easier to fit. Since some u values are used as observations, the overall prediction accuracy of uu is higher than g, and less affected by changes in g.

04

Further Research

Some Questions

Why is it not always better for parameter inversion to be smoother? What kinds of parameters are easier to learn? How much observational data (u) is needed to solve the inversion parameter problem? (Existing observational data: x or y = 1/2, 0, 1 of u (boundary conditions + extra sampling)). Can the book on numerical solutions to inverse problems you read over the summer provide some explanation for these inversion-related issues?

Are PINNs (Physics-Informed Neural Networks) advantageous compared to FEM (Finite Element Method) for solving PDE forward and inverse problems, particularly in high dimensions? (Currently, you are using equations with known exact solutions, which makes it easier to assess the accuracy of the solutions. If there are no explicit analytical solutions, how does one compare with numerical solutions?)

For trivial inversion (where you know u and calculate g), if a more general inversion problem has a good fitting result, can it also have some noise resistance?

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thanks for listening

饮 水 思 源　爱 国 荣 校