1. 方程形式：

二维耦合burgers方程如上式所示，其中方程中值为，初始条件如上，边界条件设置为0。取值范围：

1. 精确解：由github：[diakodarian/2DBurgersMATLAB (github.com)](https://github.com/diakodarian/2DBurgersMATLAB)

解二维耦合burgers方程matlab,可得到精确解文件：\burger正二维\reference\_solution.mat，该文件可用于作图，验证最终结果，于反问题中当作精确的实验数据。

文件中数据：

#有五个double数据，分别是

#x :101\*1 是0到1的均分

#y :101\*1 0-1

    #t :101\*1 0-1

#uref精确解 101\*101\*101

#vref精确解 101\*101\*101

1. 文件说明：
   1. \data\_generation 文件是matlab用于计算reference\_solution.mat的.m函数文件
   2. \models文件是用于存放得到的一些模型结果
   3. \pinn\neural\_net.py 是网络的一些基础构建设置，定义模型,train函数
   4. \pinn\get\_points.py 是获取精确解的一些函数
   5. \pinn\utilt.py 是log函数,更稳定的输出日志函数
   6. \plot\_history 是存储不同结果的一些图像
   7. \result\_plot 是存储结果图像的一个文件
   8. reference\_solution.mat 是通过matlab计算出的精确解
   9. train\_model.py 是求解二维burger方程正问题的主函数
   10. .ipynb 是Jupyter Notebook 来查看数据文件和写代码时留下的一些思路代码，属于调试文件，对结果无直接影响
   11. 实验记录.txt是在调试过程中修改一些参数时获得的一些输出日志
2. 求解思路说明：
   1. 定义方程：上面已给出
   2. 构建神经网络：在此我们设计了一个全连接的神经网络

网络结构为：[3, 20, 20, 20, 20, 2]

激活函数为：tanh

学习率为:0.001 k的学习率为0.00001

优化器为:adam法

残差计算：MSE 均方误差

Loss函数：loss\_PDE+loss\_solution 将PDE方程（u,v）和与精确解误差加入到Loss函数以达到控制神经网络拟合方程的效果，

其中边界条件过多我们直接定义到网络中：实现方法是通过U = d(t, x, y)U˜ + (1 − d(t, x, y))Up d是到边界距离，是我们设计的一个函数

这个距离函数是在靠近边界距离很近时接近1，一旦距离稍大一点直接迅速下降为0，形式：d = 16x(1 − x)y(1 − y) tanh(αt).

* 1. 具体学习：（思路）

for epoch in range(epochs):

optimizer.zero\_grad()

首先设置PDE方程内部的点：目的是求loss\_pde

随机取点 取10000（不取边界上的）

U\_f = PINN(t,x,y) uf是通过网络pinn预测出的解

v\_f = PINN(t,x,y) uf是通过网络pinn预测出的解

PDE条件即U\_f v\_f，x,t,y三个尽量满足方程：PDE方程尽量趋近于0

U\_f,v\_f相加

Loss\_pde即求出

其次取reference\_solution.mat中的坐标求预测解与真实解的mse误差

Loss\_solu即可求出

self.optimizer.add\_param\_group({'params': self.nu, 'lr': 0.00001 })

增加k变量此参数可以影响方程，以至于影响最终loss，我们要得到loss最小时对应的nu，此时为最佳参数 1

# 因为我们要求nu精度要求比较高，所以用小的学习率

Loss=loss\_ped+loss\_solu

记录损失函数及画图

Loss.backward()

optimizer.step()

* 1. 验证网络及作图：
     1. 画热力图显示方程精确解形式：\result\_plot\ Burger2d\_exact\_u.png Burger2d\_exact\_v.png
     2. 预测解的热力图：\result\_plot\ Burger2d\_pretict\_u.png Burger2d\_pretict\_v.png
     3. Loss值历史图:\result\_plot\Burger2d\_loss.png
     4. K(nu)值历史图：\result\_plot\nu\_loss(40000).png
     5. 预测解与精确解之间的误差图\result\_plot\ Burger2d\_point\_error\_u.png Burger2d\_point\_error\_v.png
     6. 六幅子图，分别是t=0.5时 y=0.15 y=0.33 y=0.4 预测解与真实解

t=0.5 x=0.53 \ 0.76 \ 0.87

以上即为实验四的基本思路