Numerical Methods and Deep Learning Frameworks for Solving Monge-Ampere Equation

Jialin Chen
518071910001
sjtuchenjl@sjtu.edu.cn

June 10, 2021

Abstract

The Monge-Ampere equation is a fully nonlinear elliptic partial differential equation that arises in optimal mass transportation (OMT). Due to its non-linearity, singularity, and convexity constraints, there are substantial numerical difficulties in solving such kind of equations. In this report, we introduce some related numerical methods and then use the monotonicity properties to approximate the operator. We use multigrid with Gauss-Seidel operations on it and do some numerical experiments. For neural network frameworks, we compare the performances of two classical networks in solving Monge Ampere equations.

Contents

1 Introduction

1.1 Optimal Mass Transport (OMT) Problem

For d dimension, consider two convex domains (i.e. open and bounded subsets) on $\mathbb{R}^d$, Υ and Ω, their mass densities are $\rho : \Omega \to \mathbb{R}$ and $\sigma : \Upsilon \to \mathbb{R}$ respectively. The two domains have equal total mass (i.e. $\int_\Omega \rho \mathrm{d}x = \int_\Upsilon \sigma \mathrm{d}x$). We aim to minimize the cost of transporting a density σ to a density ρ using a transport map T, which leads to the so-called Monge Optimal Transport Problem. By a change of coordinates, one can have the following equation,

$$\rho(x) = \sigma(T(x))\mathrm{det}DT(x). \tag{1}$$

where T satisfies that $T(\Omega) = \Upsilon$. By Brenier's theoremBrenier [1991], $T = \nabla u$, i.e. T is the gradient of a convex function u that map the mass density ρ to σ. Consequently one can have Monge-Ampere Equation in the following form,

$$\rho(x) = \sigma(\nabla u(x))\mathrm{det}D^2u(x) \qquad \forall x \in \Omega, \tag{2}$$

where ∇u and $D^2(x)$ respectively denote the gradient vector and Hessian matrix of u at x. Due to the convexity of the domains Ω and υ, T satisfies that $T(\Omega) = \Upsilon$ is equivalent to the boundary condition

$$\nabla u(\partial\Omega) = \partial\Upsilon \tag{3}$$

For $d = 1$, the Monge Ampere equation reduces to the Poisson equation and the boundary condition reduces to a nonlinear Neumann boundary condition. In this report, we consider the case $d = 2$, then equation (2) is a fully nonlinear elliptic equation when u is a convex function.

1.2 The Setting for Equation

In this report, the PDE we study is

$$\mathrm{det}D^2u(x) = f(x),$$

Since we restrict to domain $\Omega \subset \mathbb{R}^2$, then we consider the following form

$$\left(\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} - \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2\right)(x, y) = f(x, y) \text{ in } \Omega \subset \mathbb{R}^2$$

The PDE comes with Dirichlet boundary conditions $u = g$ and the constraint that u must be convex. Most of the work have been done on trying to solve this equation numerically. The equation is difficult to solve numerically due to following reasons as mentioned in Benamou et al. [2010].

1. **Non-linearity:** The equation is fully nonlinear and therefore the weak solutions are either geometric solutions or viscosity solutions Benamou et al. [2010]. Even if the two-dimensional equation can be written in the divergence form we would still have the Hessian of the solution. This places restrictions on using the Finite Element Method (FEM) or general numerical methods.

2. **Weak Solutions:** For singular solutions, the appropriate notion of weak solution must be used. Oliker and Prussner [1989] presented a method that converges to the Aleksandrov solution. Oberman [2008] introduced a wide stencil finite difference method which converges to the viscosity solution. Both of these methods were restricted to two dimensions.

3. **Convexity:** The convexity constraint is necessary for both uniqueness and stability. In particular, the equation fails to be elliptic if u is non-convex. Some tricks can be built into the discretization, as in Oberman [2008] and into the design of network structure, as in Amos et al. [2017].

4. **Accuracy:** The convergent monotone scheme of Oberman [2008] uses a wide stencil and the accuracy of the scheme depends on the width of the stencil, as we demonstrate below. Existing work has proved that singular solutions using standard finite differences, which is formally $\mathcal{O}\left(h^2\right)$, produce results which are only accurate to $\mathcal{O}\left(\sqrt{h}\right)$Benamou et al. [2010].

In this report, we take $u(x,y) = e^{\frac{1}{2}(x^2+y^2)}$ and $f(x,y) = (1+x^2+y^2)e^{\frac{1}{2}(x^2+y^2)}$ as a smooth example and explore the performance of traditional numerical methods and deep learning framework on this equation. Figure 1 shows the true solution on $[0,1] \times [0,1]$.

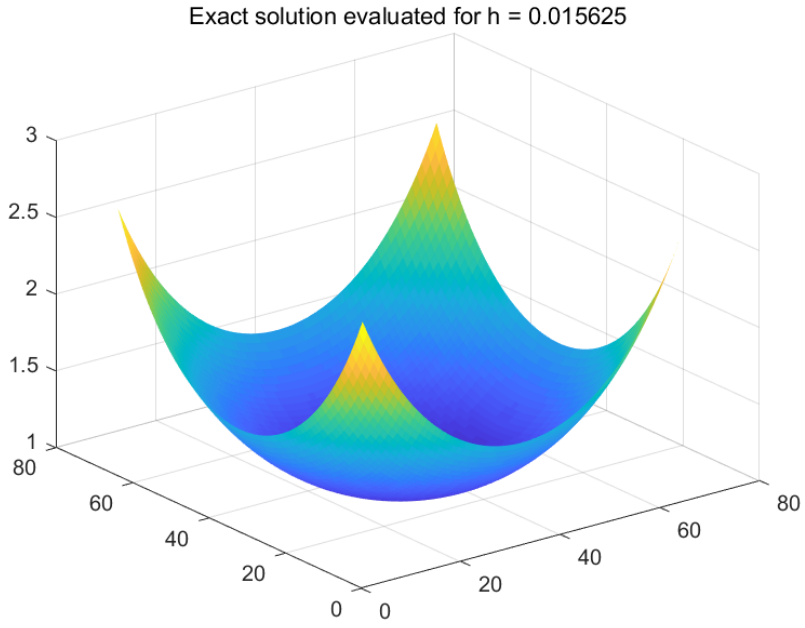


Figure 1: True solution on $[0,1] \times [0,1]$, $N = 65$

2 Traditional Numerical Methods

2.1 Related Numerical Works

For the Dirichlet condition problem, a series of papers have recently appeared by two groups. Firstly, by Oliker and Prussner [1989], uses a discretization based on the geometric interpretation of the solutions. In two dimensions this method converges to the geometric solution. Secondly, Benamou et al. [2010], proposed methods that perform best in the regular case and can break down in the singular case. The works Loeper and Rapetti [2005] solves the problem in the periodic case and Delzanno et al. [2008] solves the problem based on the applications of mappings.

2.2 Multi-grid Method with Wide Stencil Scheme

We say an update scheme is monotone if for all the discrete solutions U and V, if $U_j^n \geqslant V_j^n$, then $U_j^{n+1} \geqslant V_j^{n+1}$. A simple numerical discretization of the determinant operator on 9-points stencil is

$$\begin{aligned} det\mathrm{D}^2 u(i,j) = & \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j}}{h^2} \times \frac{u_{i,j+1} + u_{i,j-1} - 2u_{i,j}}{h^2} \\ & - \left(\frac{u_{i+1,j+1} + u_{i-1,j-1} - u_{i+1,j-1} - u_{i-1,j+1}}{4h^2} \right)^2 \end{aligned} \tag{4}$$

We denote the root-finding algorithm by P, i.e.,

$$U_{i,j}^{n+1} = P\left(U_{i+1,j+1}^n, U_{i+1,j}^n, U_{i+1,j-1}^n, U_{i,j+1}^n, U_{i,j-1}^n, U_{i-1,j+1}^n, U_{i-1,j}^n, U_{i-1,j-1}^n, f_{i,j}\right)$$

We can see that P is not monotonically increasing with respect to the four items $U_{i-1,j-1}^n$, $U_{i-1,j+1}^n$, $U_{i+1,j-1}^n$, $U_{i+1,j+1}^n$ at the same time. Consequently, this simple discretization scheme is not monotone, which contradicts with the monotonicity of determinant operator $\det(D^2)$.

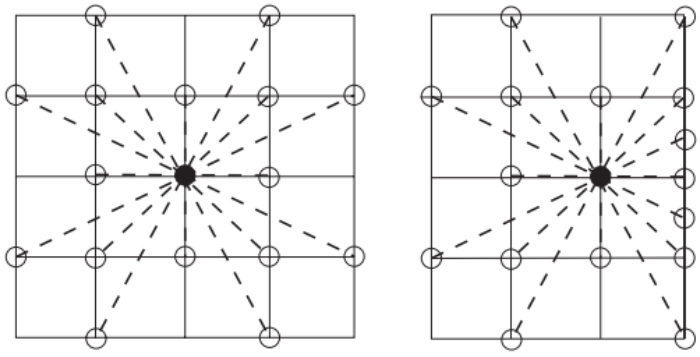
Benamou et al. [2016] introduces a new discretization scheme of the determinant operator on $(2L+1) \times (2L+1)(L \in \mathbb{N})$ stencil:

$$\det\left(D^2 u_{ij}\right) \approx \max\left\{0, \left(\mathrm{MA}_h^+ u\right)_{ij}\right\}$$

where

$$\left(\mathrm{MA}_h u\right)_{i,j} = \min_{\substack{0<m\leqslant L, 0\leqslant l\leqslant L \\ \gcd(m,l)=1}} \frac{u_{i+m,j+l} - 2u_{i,j} + u_{i-m,j-l}}{m^2h^2 + l^2h^2} \times \frac{u_{i+l,j-m} - 2u_{i,j} + u_{i-l,j+m}}{l^2h^2 + m^2h^2}$$

Since u is convex, we know $\det(D^2 u) \geqslant 0$, so we take the maximum value of 0 and the minimization result. Denote the wide stencil scheme as D_V^{WS} Benamou et al. [2016]. Take $L = 2$, the corresponding 5×5 wide stencil is shown in Figure 2, where four different colors represent four orthogonal bases of different orientations, 0°, 27°, 45°, 63° respectively.



(a) In the interior. (b) Near the boundary.

Figure 2: 5×5 wide stencil Froese and Oberman [2011]

Benamou et al. [2016] proves the monotonicity of D_V^{WS}. In this part, we use the wide stencil discretization to solve the Monge Ampere equation on multigrid. The algorithm is shown as follows:

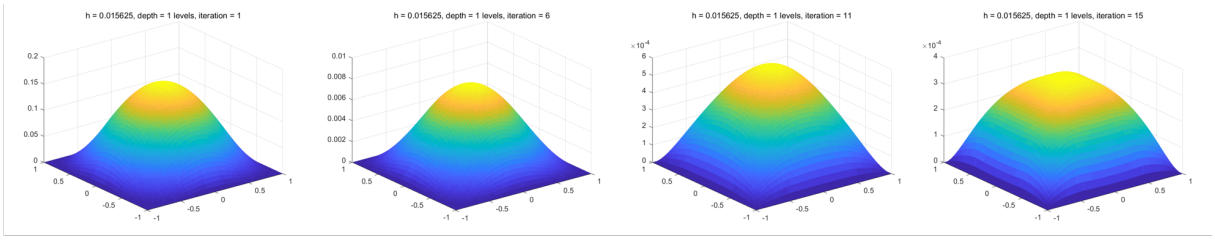
Algorithm 1 Numerical Method with D_V^{WS} on multigrid

Require: equation to solve: $MA_h(u^h) = f^h$, grid size h, threshold ε
Ensure: the numerical solution v^h
Initialize v^h
Compute the residual on the fine grid: $r^h = f^h - MA_h(v^h)$
if residual $>$ threshold **then**
    Restrict r^h to the coarse grid: $r^{2h} = I_{h\to 2h} r^h$
    Restrict the approximation solution v^h to the coarse grid: $v^{2h} = I_{h\to 2h} v^h$
    Solve the restricted problem on the coarse grid: $MA_{2h}(u^{2h}) = MA_{2h}(v^{2h}) + r^{2h}$
    Compute the error on the coarse grid: $e^{2h} = u^{2h} - v^{2h}$
    Interpolate e^{2h} to the fine grid: $e^h = I_{2h\to} e^{2h}$
    Use e^h to update the current approximation on the fine grid: $v^h = v^h + e^h$
    Compute the residual on the fine grid: $r^h = f^h - MA_h(v^h)$
end if
return numerical solution v^h

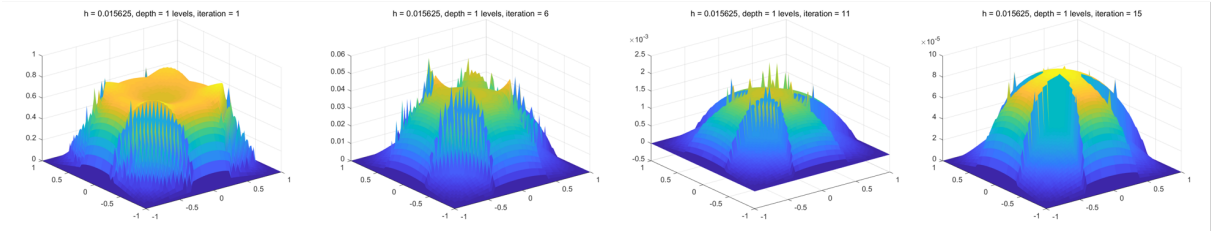
2.3 Numerical Results

We implement the two-layer version of multi-grid algorithm to solve this equation. The number of iterations represents one V-cycle in the multigrid which can be from 1-4 levels in depth. The error goes down and we control the error by setting the residual tolerance in the 'RES' variable which is set to $h^2/10$ where h is the grid size in our Matlab codes. Take $h = \frac{1}{2^6}$, depth $\in \{1, 2, 3, 4\}$, Figure 3, 4, 5, 6 show the error and residual as iteration number changes with depth $= 1, 2, 3, 4$ respectively. The top row represents the change in errors, and the next row represents the change in residuals. Zoom in to see the change of value. The error and residual are defined as follows:

$$\begin{aligned} error(x) &= u(x) - \hat{u}(x) \\ residual(x) &= f(x) - \det D^2 \hat{u}(x) \end{aligned} \tag{5}$$



Error v.s. Iteration numbers



Residual v.s. Iteration numbers

Figure 3: error and residual changes with the iteration number (depth=1)

We can see clearly that the errors keep positive and decay to zero as iteration increases. Residuals oscillate intensely with iterations but show a decreasing and smoothing trend. At the same

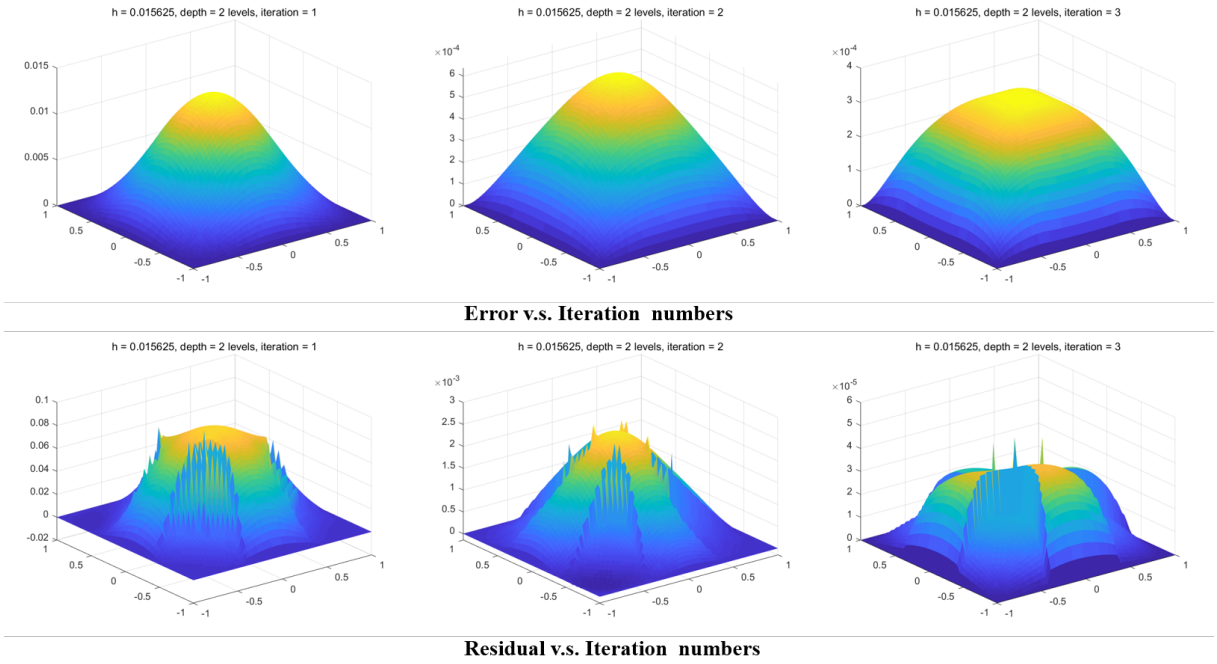


Figure 4: error and residual changes with the iteration number (depth=2)

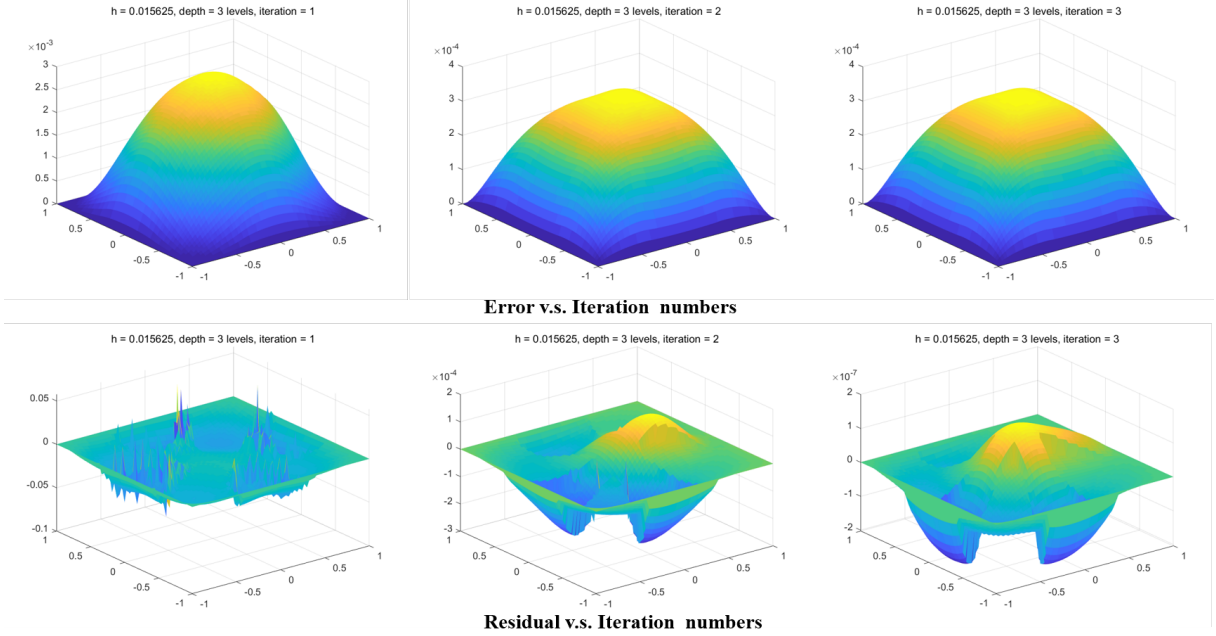


Figure 5: error and residual changes with the iteration number (depth=3)

time, we can see that the errors and residuals are close to zero at the boundary and that is a good check with our problem setup.

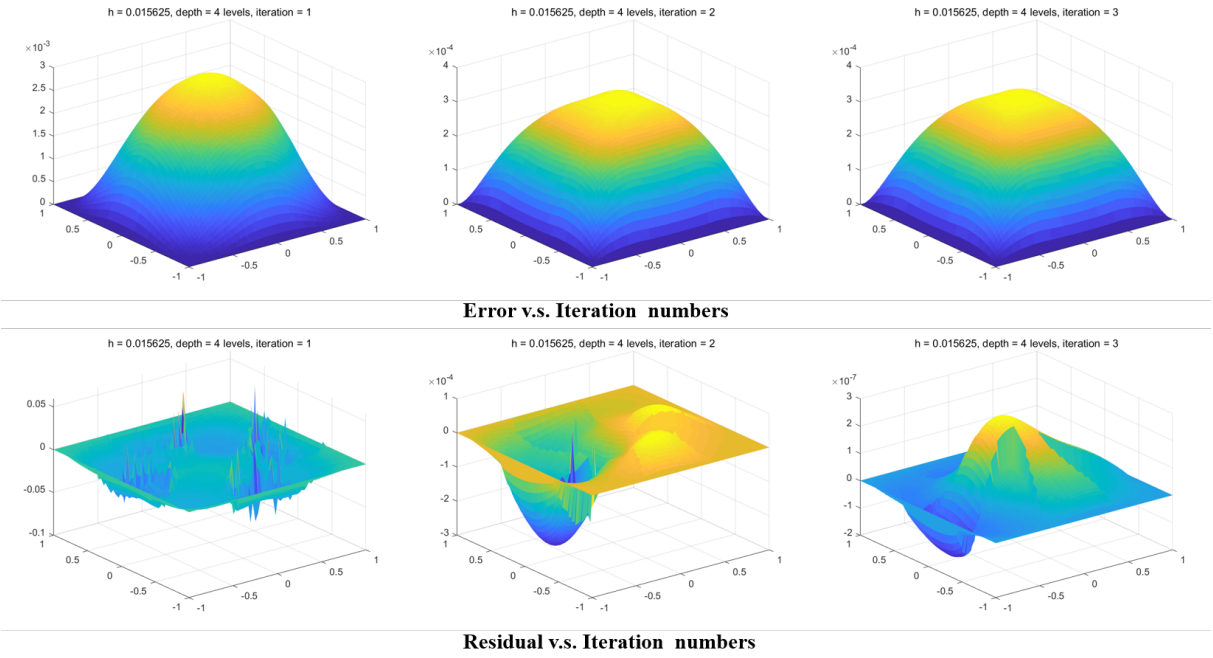


Figure 6: error and residual changes with the iteration number (depth=4)

3 Deep Learning Framework

3.1 Deep Learning Methods for PDE

Neural networks introduce nonlinear activation functions like ReLU or Softmax functions into the structure. As a consequence, the hypothesis space is enlarged, and the nonlinear expression ability of the neural network is stronger. The basic concept of deep learning networks is to use the universal approximation property of neural networks to represent the solution of a PDE. The boundary and initial conditions are usually encoded in the loss function which we want to minimize. Deep learning models need a certain amount of training sets to learn the parameters of the network, and then use these parameters to predict new data points.

3.2 Physics-informed Neural Networks (PINN)

Physics Informed Neural Networks (PINNs) are proposed by Raissi et al. [2019] to solve general nonlinear partial differential equations. Compared with the normal deep neural networks, PINNs design the loss functions by combining the boundary and initial conditions of the equations. The structure of PINNs is shown in Figure 7. Consider Monge Ampere equation $\det D^2 u(x) = f(x)$ with the boundary condition $u(x) = g(x)$, $x \in \partial\Omega$.

We need to restrict the neural network $\hat{u}$ to satisfy the physics imposed by the PDE and boundary conditions. In practice, we restrict $\hat{u}$ on some scattered points (e.g., randomly distributed points). For dataset, $\mathcal{T}$ comprised of two sets $\mathcal{T}_f \subseteq \Omega$ and $\mathcal{T}_b \subseteq \partial\Omega$, which are the points in the domain and on the boundary, respectively.

The loss function contains two main parts: $\mathcal{L}_f$ and $\mathcal{L}_b$ which represent the loss of points in the domain and points on the boundary respectively. For the points in the domain, we use the predicted function value $\hat{u}(x,t)$ to calculate the partial derivatives. For the boundary points, we

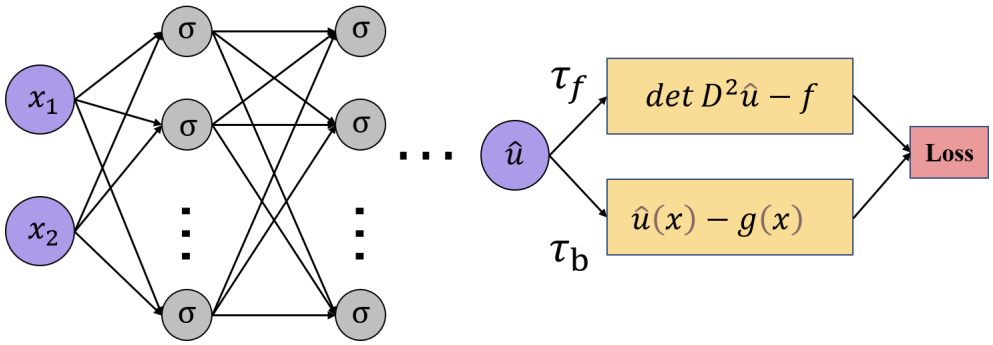


Figure 7: Structure of PINNs

compare them with the boundary / initial conditions.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b)$$

where w_f and w_b are the weights (in the experiment, we take $w_f : w_b = 1 : 1000$) and

$$\begin{aligned} \mathcal{L}_f(\boldsymbol{\theta}, \mathcal{T}_f) &= \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| \det D^2 u(x) - f(x) \right\|_2^2 \\ \mathcal{L}_b(\boldsymbol{\theta}, \mathcal{T}_b) &= \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \| u(x) - g(x) \|_2^2 \end{aligned} \tag{6}$$

We use a network with 5 hidden layers with 128 neurons in each layer. 1024 boundary points and 1024 internal points were generated randomly. We use gradient descent to optimize the loss function and Adam Kingma and Ba [2015] is used as our optimizer. The initial learning rate is $1e-3$. We run 20 epochs in total and select the model that achieves the best result in test. Figure 8 shows the absolute error as there are more epochs in the process. The definition of error at point x is the same as equation 5. We find that the error becomes smaller and less smooth as the number

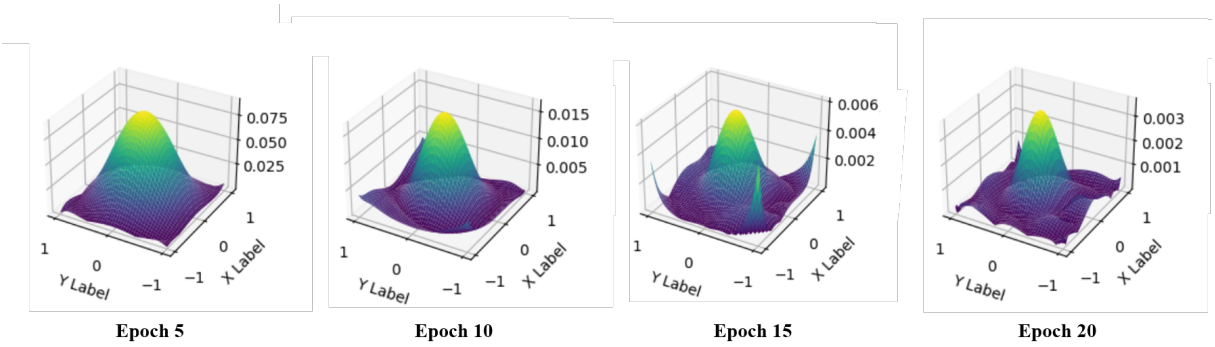


Figure 8: Absolute error changes with model PINNs (epoch=5, 10, 15, 20)

of epochs increases At the same time, the boundary is fitted well first and the error in the interior domain is larger.

3.3 Input Convex Neural Network (ICNN)

Recently, Amos et al. [2017] proposed a new architecture of neural netwokrs, Input Convex Neural Networks (ICNNs), that explicitly constraints the function approximated by the network to be

convex. By Breniers' theorem mentioned before, the transport map T is the gradient of a convex function u which is exactly what we're looking for. Consequently this architecture naturally lends itself to our proposed application. The architecture of ICNNs is shown in Figure 9. The update

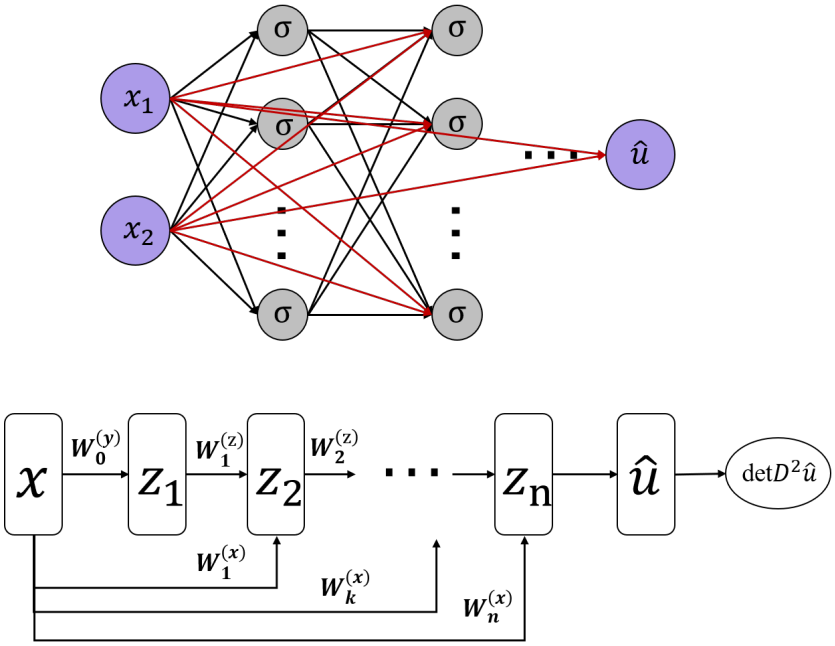


Figure 9: Structure of ICNNs

rule and final output is

$$z_{i+1} = \sigma\left(W_i^{(z)} z_i + W_i^{(x)} x + b_i\right), \quad \hat{u}(x) = z_n$$

In the ICNN, the activation function σ is a non-decreasing convex function and the weights $W_k^{(z)}$ are constrained to be non-negative. There is no constraints for weights $W_k^{(x)}$. It is easy to prove that the approximated function value $u(x)$ is a convex function of the input x under these conditions. We choose the Softplus function to the power of α as our non-decreasing activation function, which is defined as

$$\text{Softplus}^{\alpha}(x) = (\log(1 + \exp x))^{\alpha}$$

Similar to the previous simulation parameters, we use a network with 5 hidden layers with 128 neurons in each layer and Adam Kingma and Ba [2015] as our optimizer. 1024 boundary points and 1024 internal points were generated randomly. The initial learning rate is $1e-2$. We run 20 epochs in total and select the model that achieves the best result in test. Figure 10 shows the absolute error plots as there are more epochs in the process. We find that the error decreases and becomes less smooth as the epoch increases. Unlike PINN, the error of ICNN does not fit well at the four vertices (which may also be a problem of my experimental design), but the error in the middle region is more flat than PINN. Comparing the error magnitude of the two models, we find that the error of ICNN is smaller than that of PINN. More details of the experiments are in section 4.

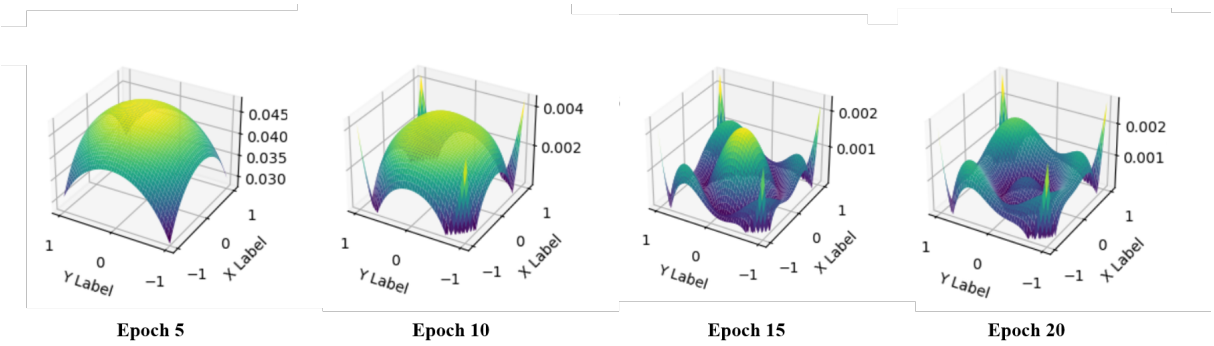


Figure 10: Absolute error changes with model PINNs (epoch=5, 10, 15, 20)

4 Additional Experiments and Results Analysis

4.1 Traditional Numerical Simulation

In this part, we show the experimental results and compare the performance of the numerical methods and the neural network frameworks in solving Monge Ampere equation. In the part of traditional numerical methods, we explore the magnitude of error at different mesh sizes h. Figure 11 shows how the error behaves which is pretty linear as we have set it to be and therefore we do not do more in-depth studies on this matter.

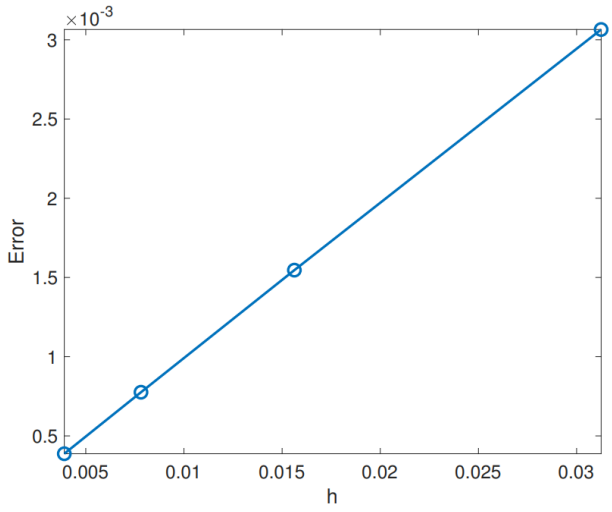


Figure 11: Average Error behavior at different mesh sizes

4.2 Experimental Results of Neural Network

Table 1 shows the average error of different models on the test set in the case of different training set sizes. The test set is generated by taking points uniformly in the domain $[0,1] \times [0,1]$ with a specific step size $h = \frac{1}{2^7}$. The training set is generated randomly. $2^{10} + 2^{10}$ in Table 1 means we take 2^{10} points in the domain and 2^{10} points on the boundary. Experiments show that when we select $2^9 + 2^9$ points, the network cannot converge to an optimal solution. Since the dataset used in neural network is generated randomly, i.e. neural network is a mesh-free method for solving PDEs,

we have no concept of h here. However, we can roughly see that the fitting accuracy of the neural network greatly depends on the number of points.

Model	$2^{10}+2^{10}$	$2^{11}+2^{11}$
PINN	0.0066	0.0014
ICNN	0.0036	0.0012

Table 1: the average error of different models on the test set in the case of different training set sizes after 20 epochs

4.3 Comparison

In the task of solving Monge Ampere equation, the accuracy of neural network is lower than that of numerical scheme, and the computation cost is higher. It takes more time to get an acceptable numerical solution. But neural networks have natural advantages, they can fit nonlinear functions, and can be effective in solving higher-dimensional PDEs. Compared with PINN, ICNN embeds the prior condition that u is a convex function through special network structure. The convergence is faster and the accuracy is higher than PINN under the same training time.

5 Conclusion

In this report, we introduce the mathematical background of Monge Ampere equation which is extracted from the optimal mass transport problem. There are substantial numerical difficulties when it comes to the non-linearity, singularity, and convexity constraints which are a major burden to making solvers which are robust but also fast. In traditional part, we go over in shallow details of some related numerical methods and then describe our multigrid approach with wide stencil proposed by Benamou et al. [2016]. We use the monotonicity properties to approximate the operator and then use multigrid with Gauss-Seidel operations on it. In neural network framework part, we introduce two classical networks and compare the compare the fitting performance of the two models numerically.

References

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Jean-David Benamou, Brittany D Froese, and Adam M Oberman. Two numerical methods for the elliptic monge-ampere equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44 (4):737–758, 2010.

Vladimir I Oliker and Laird D Prussner. On the numerical solution of the equation $\partial\hat{}\ 2\ z\partial\ x\hat{}\ 2\ \partial\hat{}\ 2\ z\ \partial\ y\hat{}\ 2$-\left ($\partial\hat{}\ 2\ z\ \partial\ x\ \partial\ y$\right)$\hat{}\ 2$= f and its discretizations, i. *Numerische Mathematik*, 54(3):271–293, 1989.

Adam M Oberman. Wide stencil finite difference schemes for the elliptic monge-ampere equation and functions of the eigenvalues of the hessian. *Discrete & Continuous Dynamical Systems-B*, 10(1):221, 2008.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

Grégoire Loeper and Francesca Rapetti. Numerical solution of the monge–ampère equation by a newton's algorithm. *Comptes Rendus Mathematique*, 340(4):319–324, 2005.

Gian Luca Delzanno, Luis Chacón, John M Finn, Y Chung, and Giovanni Lapenta. An optimal robust equidistribution method for two-dimensional grid adaptation based on monge–kantorovich optimization. *Journal of Computational Physics*, 227(23):9841–9864, 2008.

Jean-David Benamou, Francis Collino, and Jean-Marie Mirebeau. Monotone and consistent discretization of the monge-ampere operator. *Mathematics of computation*, 85(302):2743–2775, 2016.

Brittany D Froese and Adam M Oberman. Convergent finite difference solvers for viscosity solutions of the elliptic monge–ampère equation in dimensions two and higher. *SIAM Journal on Numerical Analysis*, 49(4):1692–1714, 2011.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.