Analyzing beam deformation problem using Deep Energy Method with various Neural Networks

Chan-Eui Song[1]

Ⅰ. INTRODUCTION

On this research, we analyze mechanical problems utilizing PINN(Physics-Informed Neural Networks) to mechanical problems. To be specific, this research focuses on solving deformation problems in beam.

Effected by the exterior forces and moments, deformation occurs inside the beam, associating with stresses and strains at each point. For proper cases with simple exterior conditions, this problem could be simplified into simple linear ODE. However, for arbitrary external moments and forces, it could be difficult to simplify the problem. Also, with beam structures that we cannot apply assumptions to simplify the problem, such as thin thickness or symmetry, also know as Euler-Bernoulli assumptions, this problem cannot be solved with ordinary differential equations. For general cases, this problem is solved using iteration algorithms, dividing structures with finite small elements, and applying principals to the small partition. We initialize the properties of elements with reliable values at first, than start iteration using data of neighboring partitions that are connected to the small partition.

However, this methods have some problems. It's simulation rate is not fast enough. Moreover, small change in exterior conditions can be critical for this method, since we have to initialize the condition from the beginning, and start the simulation one-by-one. It also lacks robustness and convergence. So, there exists some demand for other methods to solve this problem.

In this context, Physics-Informed Neural Networks, in short called PINN, can be an alternative to solve this problem. Training with neural network had showed its ability to solve various problems in diversity of fields. However, it requires numerous data to train robustly. Since normal mechanical systems cannot provide enough data, utilizing neural network in traditional fashion isn't desirable. However, using the principles of mechanical problems, this strategies can be modified. By applying the differential equations to the neural network many research have shown that problems could be solved with relatively less data.

This research solve the deformation problem in beam with energy method, applying neural network to it. Unlike previous research, we apply dynamic neural network. Then we compare this results with previous research using other types of neural networks. As a result, we compare performance of various neural network structures. At large, this research is planning to contribute to the PINN research field, and provide some significant data and proof of convergence.

Ⅱ. RELATED WORKS

A. PINN

Previous research had already showed that using deep neural network can be used to solve nonlinear problems with more robustness and for larger domain than previous Gaussian process regression. And using PINN, a lot of mechanical problems that was hard to solve with tradition methods had been solved. Most of these were problems comprised with partial differential equations, with the property of slight change in conditions can bring vast differentiation at the results. For example, Burger's Equation, Schrödinger Equation, Allen-Cahn Equation were solved with PINN with continuous time models and discrete time models, using automatic differentiation and Runge-Kutta methods. The

[1]C.E. Song is with the Department of Mechanical Engineering, Seoul National University, Seoul 08826, Korea (e-mail: cksdml1014@snu.ac.kr)

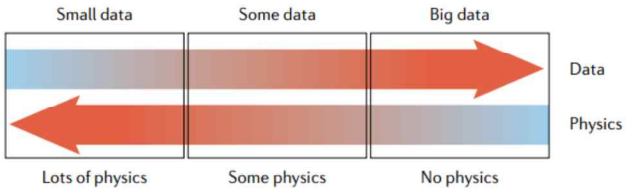

Figure 1. Data and physics scenarios

result showed that it came with better results than previous numerous methods of solving.

Most of the problems of real applications stay in the middle range of Figure 1, with some data and some physics coexist. The physics is partially know, but several measurements are available to infer parameters and infer functional terms of PDE. It can also be used to solve stochastic PDEs. Also, problems involving long-range spatiotemporal interactions which is expressed in complex fractional PDEs.

B. Deep Energy Method

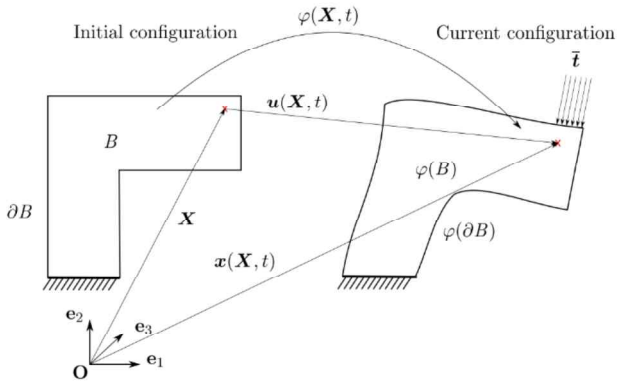

Figure 2. Motion of body B

Deep energy method is analyzing the difference of a body due to external force using potential energy. To illustrate, we consider a situation of a body B deforming to a body $\phi(B)$ with each point X of body B mapping to $\phi(X)$ as shown in Figure 2. The boundary of the body ∂B maps to $\phi(\partial B)$. The difference of each point is defined as $u(X,t)$ where t is the force acting on the point X.

With P the first Piola-Kirchhoff stress, f_b the body force, and N the outward unit vector, the equilibrium condition and the boundary conditions are written as (1)~(3).

$$\nabla_X \cdot P + f_b = 0 \tag{1}$$

$$u = \bar{u} \quad \text{on} \quad \partial B \tag{2}$$

$$P \cdot N = \bar{t} \quad \text{on} \quad \partial B \tag{3}$$

Here, $\bar{u}$ and $\bar{t}$ denotes the wanted boundary condition of deformation and external force.

The deformation gradient F is defined as (4).

$$F = \nabla_X \phi(X) \tag{4}$$

The strain energy Ψ satisfies the constitutive law as (5).

$$P = \frac{\partial \Psi}{\partial F} \tag{5}$$

In this situation, the potential energy Π can be calculated as (6).

$$\Pi(\phi) = \int_B \Psi dV - \int_B f_b \cdot \phi dV - \int_{\partial B} \bar{t} \cdot \phi dA \tag{6}$$

For given situation, we want to find the deformation ϕ that minimizes the potential energy Π. Deep energy method utilizes neural network to find ϕ that minimizes Π. Schematic diagram of Deep Energy Method is shown as Figure 3.

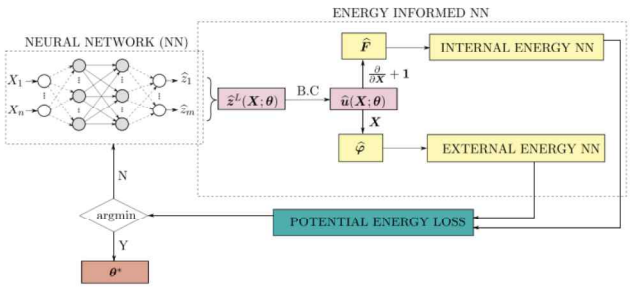

Figure 3. Schematic diagram of Deep Energy Method

Predicted deformation $\hat{\phi}$ in trial process of neural network, and its gradient $\hat{F}$ are function of the neural network parameter θ. The potential energy Π needs to be minimized, so it could be used as a loss function for neural network, which we denote $\mathcal{L}(\theta)$ since it relies on θ. Equation (6) deforms to (7) in the trail process.

$$\mathcal{L}(\theta) = \int_B \Psi(\hat{F}) dV - \int_B f_b \cdot \hat{\phi}(X;\theta) dV - \int_{\partial B} \bar{t} \cdot \hat{\phi}(X;\theta) dA \tag{7}$$

Using collocation points $X_1, \ldots, X_N$ as inputs for neural network, we can get output $\hat{z}_1, \ldots, \hat{z}_m$. These outputs are then calculated through proper equation to achieve deformation $\hat{u}$ that satisfies boundary condition of deformation. Since $\phi(X,t) = u(X,t) + X$ and using equation (4), we can get $\hat{\phi}$, $\hat{F}$ as equation (8), (9).

$$\hat{\phi}(X,t) = \hat{u}(X,t) + X \tag{8}$$

$$\hat{F}(X,t) = \frac{\partial \hat{u}(X,t)}{\partial X} + 1 \tag{9}$$

These values are then used to calculate the loss function (7). Using various optimization methods, such as Limited-memory quasi-Newton methods or Adam, we modify the network parameters and conduct iterations to get the minimum potential energy.

C. Numerical Integration Method

While conducting iterations on deep energy method, we need to calculate the loss function (7), which contains integration. However, we cannot express integrands in (7) to analytic forms. Also, since we put finite number of inputs to the neural network, the integrals in (7) are actually needs to be computed with data from finite number of points. We utilize couple of numerical integration method on this research, and compare the results.

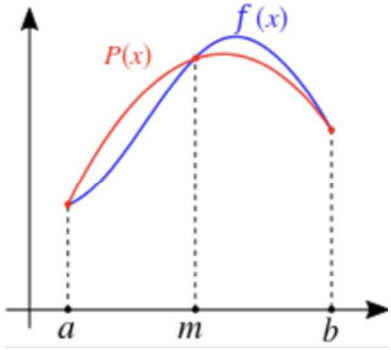

Figure 4. Idea of Simpson's Rule

The first numerical integration method is Simpson's rule. Simpson's rule divides the domain and approximates the function as quadratic functions in each division.

To calculate the integral $\int_a^b f(x)dx$ applying Simpson's rule, we need $2N+1$ data of $(x_i, f(x_i))$ where $i = 0, \ldots, 2N$, $a = x_0$, and $b = x_{2N}$. which is divided into constant distance. Then, we divide the interval $[a,b]$ into N intervals $[x_{2j-2}, x_{2j}]$ where $j = 1, \ldots, N$. For each interval, we approximate $f(x)$ as a quadratic polynomial $P_j(x) = a_j x^2 + b_j x + c_j$ that passes through $(x_{2j-2}, f(x_{2j-2}))$, $(x_{2j-1}, f(x_{2j-1}))$, $(x_{2j}, f(x_{2j}))$. Using Lagrange interpolation, this quadratic can be expressed as (10).

$$\begin{aligned} P_j(x) = {} & \frac{(x - x_{2j-1})(x - x_{2j})}{(x_{2j-2} - x_{2j-1})(x_{2j-2} - x_{2j})} f(x_{2j-2}) \\ & + \frac{(x - x_{2j-2})(x - x_{2j})}{(x_{2j-1} - x_{2j-2})(x_{2j-1} - x_{2j})} f(x_{2j-1}) \\ & + \frac{(x - x_{2j-2})(x - x_{2j-1})}{(x_{2j} - x_{2j-2})(x_{2j} - x_{2j-1})} f(x_{2j}) \end{aligned} \tag{10}$$

Using this interpolation, the integral in the j^{th} interval $[x_{2j-2}, x_{2j}]$ can be approximated as (11).

$$\begin{aligned} & \int_{x_{2j-2}}^{x_{2j}} f(x)dx \approx \int_{x_{2j-2}}^{x_{2j}} P_j(x)dx \\ & = \frac{x_{2j} - x_{2j-2}}{6} [f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})] \end{aligned} \tag{11}$$

Then, the total integral of the whole region is calculated as (12) where $h = \frac{b-a}{2N}$.

$$\begin{aligned} \int_a^b f(x)dx \approx \frac{h}{3} [& f(x_0) + 4 \sum_{k=1}^{N} f(x_{2k-1}) \\ & + 2 \sum_{k=1}^{N-1} f(x_{2k}) + f(x_{2N})] \end{aligned} \tag{12}$$

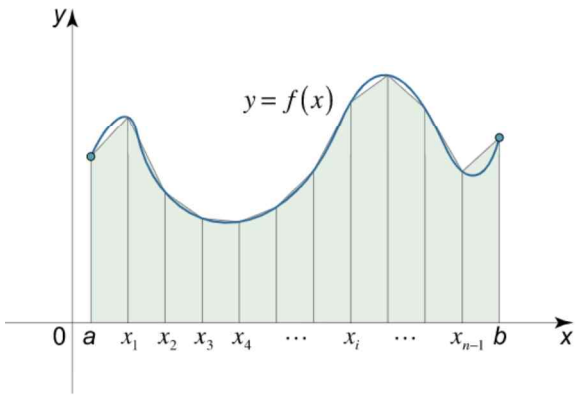

Figure 5. Idea of trapezoidal rule

The second numerical integration method is trapezoidal

rule. This method divides the domain and approximates the function as linear function in each division.

To calculate the integral $\int_a^b f(x)dx$ applying Simpson's rule, we need $N+1$ data of $(x_i, f(x_i))$ where $i = 0, \ldots, N$, $a = x_0$, and $b = x_N$. which is divided into constant distance. Then, we divide the interval $[a, b]$ into N intervals $[x_{j-1}, x_j]$ where $j = 1, \ldots, N$. For each interval, we approximate $f(x)$ as a quadratic polynomial $Q_j(x) = a_j x + b_j$ that passes through $(x_{j-1}, f(x_{j-1}))$, $(x_j, f(x_j))$. Using Lagrange interpolation, this quadratic can be expressed as (13).

$$Q_j(x) = \frac{(x - x_j)}{(x_{j-1} - x_j)} f(x_{j-1}) + \frac{(x - x_{j-1})}{(x_j - x_{j-1})} f(x_j) \tag{13}$$

Using this interpolation, the integral in the j^{th} interval $[x_{j-1}, x_j]$ can be approximated as (14).

$$\int_{x_{j-1}}^{x_j} f(x)dx \approx \int_{x_{j-1}}^{x_j} Q_j(x)dx = \frac{x_j - x_{j-1}}{2}[f(x_{j-1}) + f(x_j)] \tag{14}$$

Then, the total integral of the whole region is calculated as (15) where $h = \frac{b-a}{N}$.

$$\int_a^b f(x)dx \approx \frac{h}{2}[f(x_0) + 2\sum_{k=1}^{N-1} f(x_k) + f(x_N)] \tag{15}$$

D. ODE Network

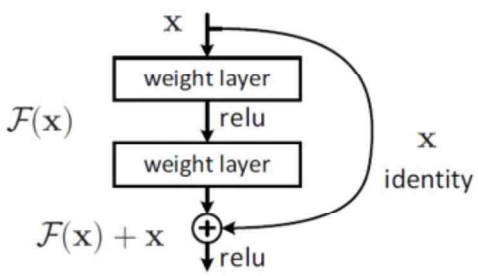

Figure 6. Idea of ResNet

There are various methods to create neural network and setup optimization algorithm. One of the innovative method that recently came out is Residual neural network, also know as ResNet. This network utilizes residual, which was only used to evaluate the performance of the network, for learning. As Fig.6 illustrates, the residual x that was calculated from layer above, are again added and used for learning. Using this method had made it easier to design more deep layers of networks, which led to better performance.

This ResNet algorithm can be briefly expressed into equation (16).

$$h_t = h_{t-1} + F(h_{t-1}) \tag{16}$$

h_t denotes the output of t^{th} layer. However, equation (16) has the similar form with Euler method, which is an numerical method to solve ODEs. The equation (16) is equivalent to solving ODE expressed in equation (17) with Euler method.

$$\frac{dh}{dt} = F(h) \tag{17}$$

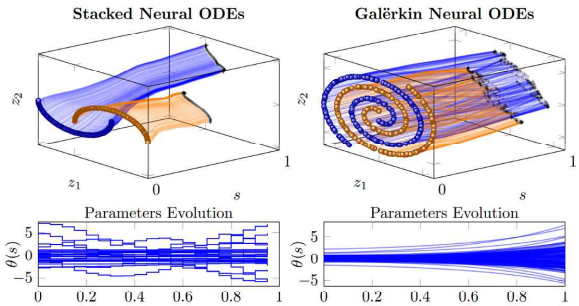

Figure 7. Example of parameters of neural ODEs

This gives the idea of neural ODEs. We can regard the network as solution of proper ODEs. With these kinds of contraints attached to the neural network, the parameters of the neural network would behave in a predictable way, and it is probable to make a better performance. Especially on PINN, which is governed by the differential equations, it is probable to predict that optimal neural network might have some physical meaning. For example, Figure 7 shows parameters of an example of utilizing neural ODEs. We can observe that the parameters changes with some pattern similar to normal ODE solution. As a result, this research utilized neural ODEs on deep energy method.

Ⅲ. 1D beam deformation problem

A. Exact Solution

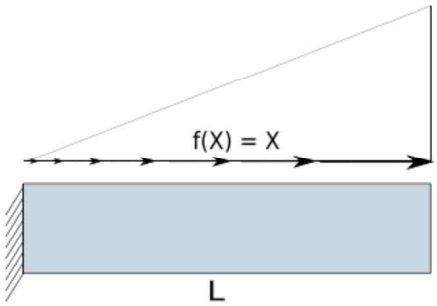

Figure 8. 1D beam deformation problem

For simple understanding of the deep energy method and PINN, this research started by solving a 1D beam deformation problem with simple external force $F(X)=X$ as can be seen in Figure 8. And the potential energy is given as equation (18).

$$\Psi(\epsilon)=(1+\epsilon)^{\frac{3}{2}}-\frac{3}{2}\epsilon-1 \tag{18}$$

The equilibrium constraint and the boundary conditions of (1)~(3) can be expressed as (19)~(21).

$$\frac{d}{dX}\frac{\partial\Psi}{\partial\epsilon}+f(X)=0, \quad -1\leqq X\leqq 1 \tag{19}$$

$$u(-1)=0 \tag{20}$$

$$\frac{\partial\Psi(\epsilon)}{\partial\epsilon}\bigg|_{X=1}=\bar{t} \tag{21}$$

The exact solution is expressed as (22).

$$u(X)=\frac{1}{135}(68+105X-40X^3+3X^5) \tag{22}$$

B. Neural Network

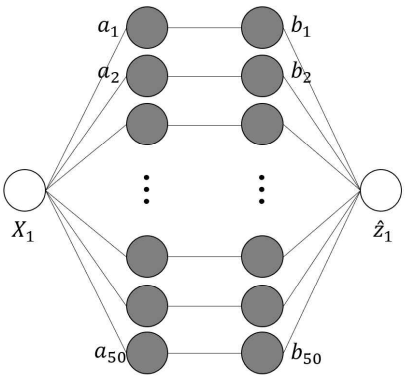

Figure 9. Neural Network

Before applying neural ODEs, first we applied simple form of neural network. This network consists of a single hidden layer, with 50 neurons on it. The input and output's dimension is both 1. The relationship between the input layer and the hidden layer had set to be linear, as well as the relationship between the hidden layer and the output layer. We used tanh activation function at the hidden layer. The schematic drawing of the network is shown in FIgure 9.

C. Algorithm

The basic algorithm is same as the deep energy method. It could be shown as Table 1.

Table 1. Algorithm for deep energy method

Input: netParams-network parameters
intType-type of numerical integration
X-coordinates of all data
X_f-coordinates of data subjected to loading
X_u-coordinates of data subjected to displacement constrain
X_{Test}-coordinates of data which will be predicted
Output: U-Displacement of the predicted data
F-Deformation gradient of the predicted data
Procedure
1. Initialize the material and deep energy method model
2. train model
3. Evaluate model

D. Result

Numerical integration using Simpson's rule had less error than numerical integration with trapezoidal rule. With 100 training data, the errors were below 10^{-3} and with 1000 training data, the errors were below 10^{-4}.

Ⅳ. 2D beam deformation problem

A. Problem Setup

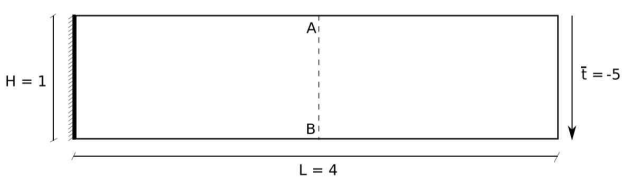

Figure 10. 2D beam deformation problem

Now we develop the 1D problem into 2D problem. We consider a 2D beam with its length $L = 4\mathrm{m}$, height $H = 1\mathrm{m}$, and depth $D = 1\mathrm{m}$. We assume the beam as a 2D plane, with plane stress condition. For boundary condition, we fix the left end at the wall, and we apply traction load of $-5\mathrm{N}$ at the right end. i.e., the boundary condition follows as equation (23).

$$u(0,y) = 0,\ \bar{t}(4,y) = -5\mathrm{N}\ \ (0 \leq y \leq 1) \tag{23}$$

We assume the beam with the material with following physical properties, Young's moudulus $E = 1000\mathrm{GPa}$, and Poisson's ratio $\nu = 0.3$. We also assumed the beam made of a homogeneous, istropic, Neo-Hookean material.

For 2D Neo-Hookean material, the strain energy density function follows as equation (24).

$$\Psi(I_1, J) = \frac{1}{2}\lambda[\log(J)]^2 - \mu log(J) + \frac{1}{2}\mu(I_1 - 2) \tag{24}$$

As the same way with the 1D beam problem, we minimize the potential energy loss $\mathcal{L}$ defined in equation (7), using the strain energy Ψ defined in (24).

B. Neural Network

We applied 4 layer neural network, with each neighboring layers connected with fully connected layer. The first and last layer are consisted of 2 neurons, each corresponding to the coordinates and the deformation. The 2nd and 3rd layer are consisted of 30 neurons. After each layer, we applied $\tanh$ activation function. We applied the L-BFGS optimizer for training, with the learning rate of 0.5, and 20 epochs. Also, for numerical integration regarding the potential energy loss, we applied the Simpson's rule.

C. Result

The L2 norm is 0.200 and the H10 norm is 0.1466. For the deformation in the middle of the bar, the deformation at line AB in the Figure 10 follows as the Figure 11. Also, the deformation of the whole bar is plotted as Figure 12.

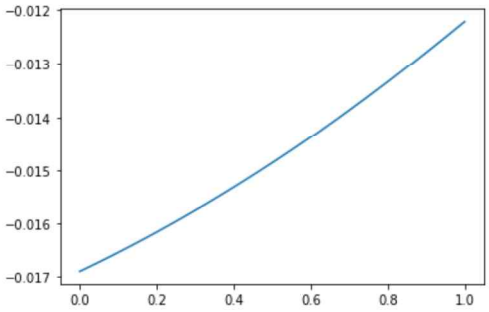

Figure 11. Deformation of the line AB

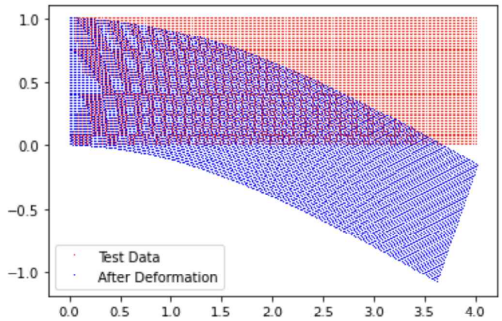

Figure 12. Deformation of the 2D beam

V. 3D beam deformation problem

A. Problem Setup

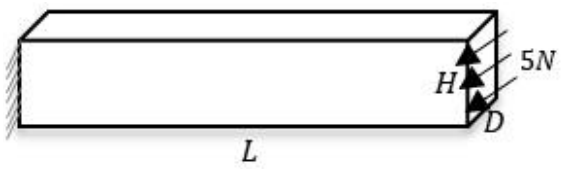

Figure 13. 3D beam deformation problem

Fianlly we analyze the 3D beam deformation problem. We consider a 3D beam with its length $L = 4\mathrm{m}$, height $H = 1\mathrm{m}$, and depth $D = 1\mathrm{m}$ as same as the 2D beam deformation problem. Also, we use the same boundary condition as before. i.e., the boundary condition follows as equation (25).

$$u(0,y,z) = 0,\ \bar{t}(4,y,z) = -5\mathrm{N}\ \ (0 \leq y, z \leq 1) \tag{25}$$

We assume the same physical properties.

For 3D Neo-Hookean material, the strain energy density function follows as equation (26).

$$\Psi(I_1, J) = \frac{1}{2}\lambda[\log(J)]^2 - \mu log(J) + \frac{1}{2}\mu(I_1 - 3) \tag{26}$$

As the same way with the 1D beam problem, we minimize the potential energy loss $\mathcal{L}$ defined in equation (7), using the strain energy Ψ defined in (26).

B. Neural Network

We applied 4 layer neural network, with each neighboring layers connected with fully connected layer. The first and last layer are consisted of 3 neurons, each corresponding to the coordinates and the deformation. The 2nd and 3rd layer are consisted of 30 neurons. After each layer, we applied tanh activation function. We applied the L-BFGS optimizer for training, with the learning rate of 0.5, and 40 epochs. Also, for numerical integration regarding the potential energy loss, we applied the Simpson's rule.

C. Result

The L2 norm is 1.208 and the H10 norm is 0.9259. Also, the deformation of the whole bar is plotted as Figure 14.

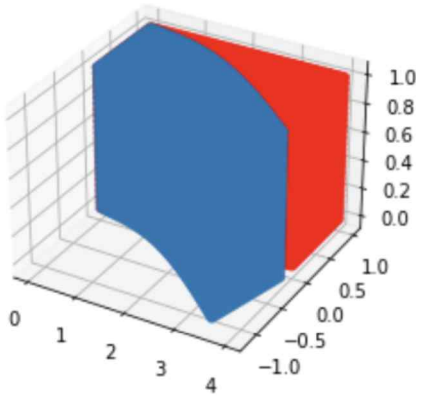

Figure 14. Deformation of the 3D beam

VI. T-structure deformation problem

A. Problem Setup

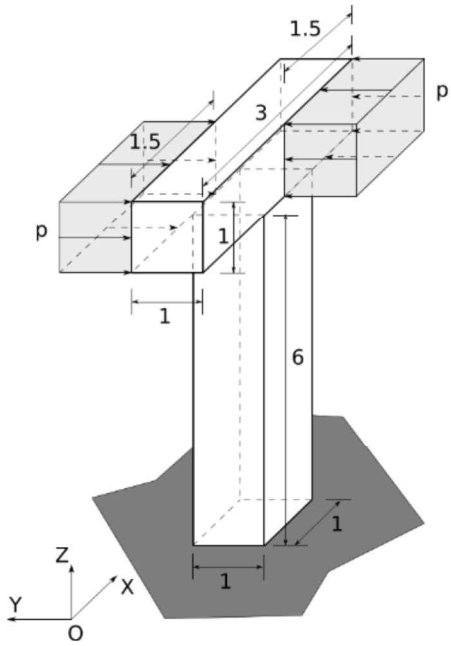

Figure 15. T-structure deformation problem

We move further to analyze deformation in a more complicated structrue. We apply forces to the top part of a T-structure described as Figure 15. We fixed the bottom part of the T-structure to the ground. We use the same stain energy denstiy function (26) as before, and we used the Simpson's rule for numerical integration..

B. Neural Network

We applied the same neural network structure of 4 layer neural network, with each neighboring layers connected with fully connected layer. The first and last layer are consisted of 3 neurons, each corresponding to the coordinates and the deformation. The 2nd and 3rd layer are consisted of 30 neurons. After each layer, we applied tanh activation function. We applied the L-BFGS optimizer for training, with the learning rate of 0.5, and 40 epochs.

C. Result

To clearly show the deformation results, we plotted the points of T-structures in two different grids. Figure

16 shows the points before the deformation, and Figure 17 shows the results after the deformation, calculated through trained model. We can find out that the deep energy method fits well with complicated models, such as a T-structure with twisting moments applied.

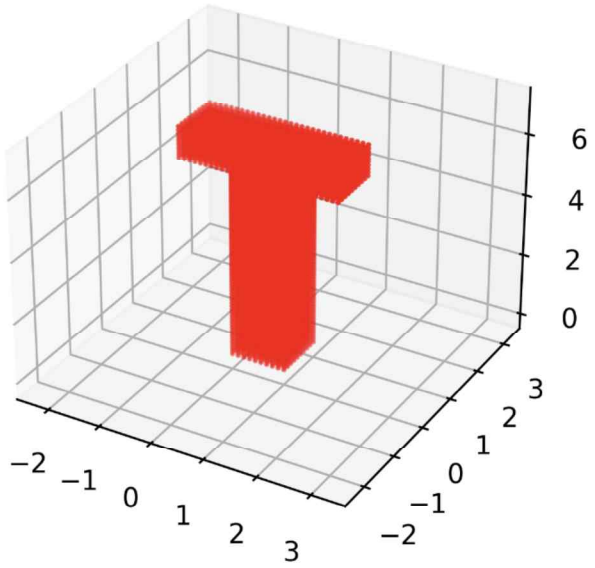

Figure 16. Test points of T-structure before deformation

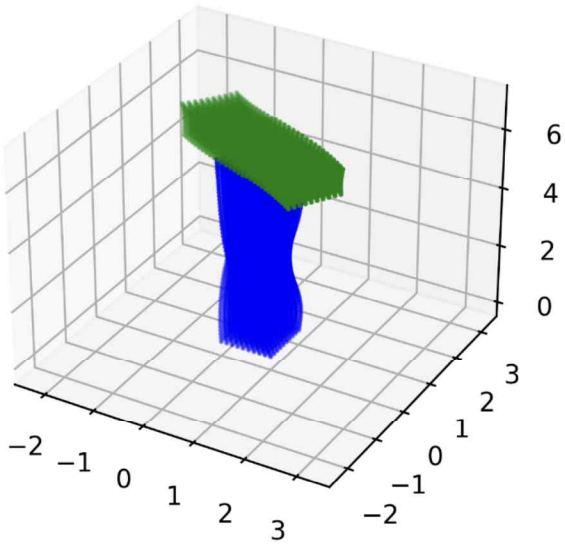

Figure 17. Deformation of the T-structure

Ⅶ. Solving Double Moon Classification Problem with Neural ODE

A. Defining problem

Double Moon classification problem is classifying points which are distributed in shape of two crescents, which are placed symmetrically as can be seen in Figure 18. Since the distribution of the points in same crescent are continuous and having continuous, but not linear structures, these problems can be solved by implementing neural ODE.

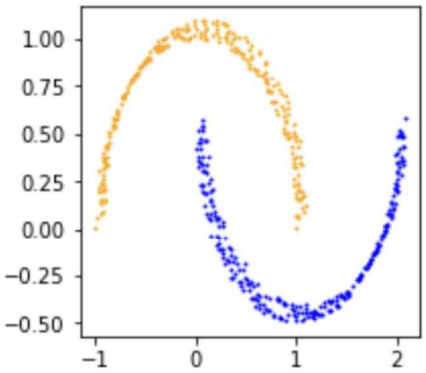

Figure 18. Double Moon Classification

B. Neural Network

The goal of the neural network is to find a mapping that gets the coordinate of the points and input, and giving out the classification of the points as output. For traditional neural networks, we connected many nodes with affine functions along with activation functions. However for applying neural ODE, the method changes.

Using neural ODE, we ought to find a function that connects the coordinates and classification as outputs for two different inputs. And we try to find the function by assuming that the function can be found by solving ODE, which we tune the parameters of ODE by neural networks. It can be written as the equations below.

$$\dot{z}(t) = f(z(t), \theta) \tag{27}$$
$$z(0) = x \tag{28}$$
$$\hat{y} = z(1) \tag{29}$$

We're trying to find the function z which connects the coordinate x and the predicted classification $\hat{y}$ by

two output of different inputs. And we're assuimg z can be solved through the ODE (27) with the parameter θ will be trained through the neural network.

For this problem, we assumed the function f, which equals to the $\dot{z}$ in ODE as 2-layer neural networks. We made a layer of 16 nodes between the input layer and the output layer. Each layers are connected with affine functions, along with the activation function $\tanh$ implemented on the middle layer.

We select some points for train data and used Adam optimizer to find the parameters that best classifies the double moon.

C. Result

After optimizing the function z, we can plot the results respect to each components of input and output.

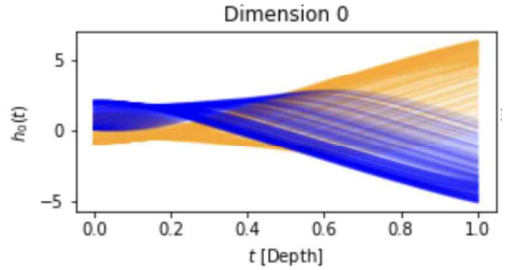

Figure 19. First components of z

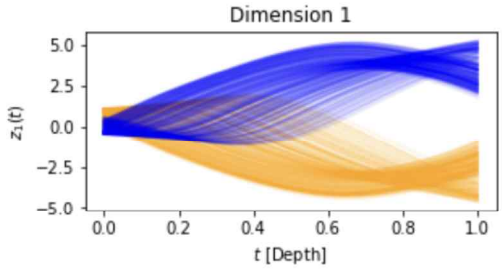

Figure 20. Second components of z

Figure 19 shows the first component of z varying from $t=0$ to $t=1$. It can be shown that at $t=0$, the value is connected to the x coordinate of the points, and the value at $t=1$ is connected to the prediction of classification. We can find that the classification is well done. Same results can be found throught Figure 20, which the second components of z connectes the y coordinate of points to classification.

Ⅷ. Discussions & Future works

Through this research, we made a approach for applying neural network to engineering using PINN. Specifically, we implemented the deep energy method to various examples. By using the principal that the deformation occurs in the direction that minimize the strain energy, we can train neural network to minimize the energy as a loss function.

With various examples such as 1D, 2D, 3D beam and 3D T-structure, we can verify that the deep energy method works well with these structures. By the simplicity of the model and the training speed, we can further replace FEM as deep energy method, although there are some more researches to be done.

For example, the optimal neural network structure for the deep energy method haven't been verified. In the examples above, we utilizes the simplest neural with shallow MLP, and it led plausible results. Since shallow MLP can be trained fastly thanks to the little amount of calculation, the current structure can be viewed as a good option if the results are guranteed.

Still, it's worthy to test various types of models and compare the results. Through this research, we intended to apply the Neural ODE into the deep energy method. However, there are some issues to overcome to implement this process. Recall that loss function in deep energy method has the shape as (18), which requires the gradient of the deformation, which is the output of the neural network. However, as the equation (27)~(29) suggests, the Neural ODE method makes the output as the result of a function, which is a solution for solving the parametrized ODE. since the numerically solving ODE requires calculating gradients in it, we need to calculate the 2^{nd} gradient to calculate the loss. Since autograd in pytorch library yet support only the 1^{st} gradient, and since the 2^{nd} gradient is difficult to calculate via chain rule, it's not easy to apply neural ODE to deep energy method. We leave this part for furure works, waiting for development of new method for calculating 2^{nd} gradients.

REFERENCES

[1] C. E. Rasmussen, C. K. Williams, Gaussian processes for machine learning, volume 1, MIT press Cambridge, 2006.

[2] M. Raissi, P. Perdikaris, G. E. Karniadakis, Numerical Gaussian processes for time-dependent and non-linear partial differential equations, arXiv preprint arXiv:1703.10230 (2017).

[3] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, arXiv preprintarXiv:1708.00588 (2017).

[4] H. Owhadi, C. Scovel, T. Sullivan, et al., Brittleness of Bayesian inference under finite information in a continuous world, Electronic Journal of Statistics 9 (2015) 1–79.

[5] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (1989) 359–366.

[6] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions of the Burgers equation, Computers & fluids 14 (1986) 23–41.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprintarXiv:1603.04467 (2016)

[8] Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 422–440 (2021).

[9] Cuomo, Salvatore, et al. "Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next." arXiv preprint arXiv:2201.05624 (2022).

[10] Nguyen-Thanh, Vien Minh, Xiaoying Zhuang, and Timon Rabczuk. "A deep energy method for finite deformation hyperelasticity." European Journal of Mechanics-A/Solids 80 (2020): 103874.

[11] Samaniego, Esteban, et al. "An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications." Computer Methods in Applied Mechanics and Engineering 362 (2020): 112790.

[12] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[13] Chen, Ricky TQ, et al. "Neural ordinary differential equations." Advances in neural information processing systems 31 (2018).

[14] Massaroli, Stefano, et al. "Dissecting neural odes." Advances in Neural Information Processing Systems 33 (2020): 3952-3963.