Final Report: Neural network finite element in solid mechanics

Wei Chen
weichen6@stanford.edu

Xinwei He
xhe17@stanford.edu

Jessica Zou
jzou@stanford.edu

Abstract

In traditional computational mechanics, constitutive model, which reflects the mechanical properties of material, plays a crucial role. In this project, we propose a new method to use neural network to replace the traditional constitutive model. A neural network uses experimental data directly, allowing non-linear solid mechanics problems, such as plasticity and viscosity, to be solved more efficiently.

1 Introduction

Finite Element Analysis (FEA) is widely used to solve for numerical solutions of partial differential equations. Many famous finite element software, such as Ansys, Abaqus, and Comsol, are used in industry to analyze the stress (a physical quantity to describe the force), strain (a dimensionless physical quantity to describe the deformation), pressure or other physical state under certain mechanical, thermal, electrical and chemical conditions. For example, FEA is the basis in the design of tiny deformable electric units and spectacular dams and bridges. It is important to propose a specific method to solve partial differential equations efficiently. FEA is difficult, primarily because of nonlinear geometry and material properties. Different material properties, elasticity, plasticity and viscoplasticity influence the complexity of FEA programs greatly such that some analyses can take a long time and do not converge due to Newton-Raphson iteration required in the process. Our goal is to avoid Newton-Raphson iteration by introducing the neural network.

2 Related Work

Neural networks have been used to model the constitutive relations in different materials including sand[1], nonlinear elastic composites[2], viscoelastic material[3] and multiscale porous media[4]. These works try to find neural networks to simulate the stress-strain or stress-strain increment relation. The data are from experimental data or another sub-scale numerical simulation[4]. Nevertheless, the quantity of experimental data is not sufficient for training a highly accurate network for computational use. Sub-scale simulation is usually based on ideal condition which couldn't reflect the true cases. The existing constitutive model could describe most of material effectively which has been proven by its usage in industry for decades. We want to find the proper neural network structure to match each constitutive model and implement it in finite element program so as to reduce the computational cost. Theoretically, we could get enough data to train a neural network to achieve a desired high accuracy.

3 Dataset and Features

Constitutive models have been researched by thousands of scientists over past decades. Many models have been proposed and applied effectively in industry. However, many constitutive relations couldn't be expressed with analytical math formula. The finite element program needs to do a lot of Newton-Raphson iteration to implement the constitutive model. By using neural network here, we want to set up the phenomenological model for ordinary material. A phenomenological model could be meaningless in Physics but useful in numerical calculation. Once the model could be established, we could transform a non-linear problem into a linear problem.

We will apply supervised learning to train the neural network. The input is strain and the output is stress. To further demonstrate the problem we need to solve. The constitutive equation between stress and strain are introduced below. The governing equation is:

$$\nabla\sigma + f_s = 0 \tag{1}$$

where σ is stress and f_s is body force. FEA is the boundary value partial differential equation solution of Eq.(1). Strain and displacement are related by:

$$\varepsilon = sym(\nabla u) \tag{2}$$

where ε refers to strain and u refers to displacement. The constitutive equation we mentioned above is the relation between σ and ε.

Constitutive model could be categorized generally time dependent and time independent. Below, We will study them respectively. We will study perfect plasticity and viscoelasticity, two widely used constitutive models.

3.1 Time Independent Model- Perfect Plasticity

For 1D plasticity problem, the known equation is

$$\sigma = E\varepsilon^e = E(\epsilon - \epsilon^p)$$

and

$$\dot{\varepsilon}^p = \begin{cases} 0, & \text{if} |\sigma| < y_0 \\ \lambda \frac{\sigma}{|\sigma|}, & \text{if} |\sigma| = y_0. \end{cases}$$

To generate data, we process it as:

$$\Delta\sigma = \begin{cases} E\Delta\varepsilon, & \text{if} |\sigma| < y_0 \\ E\Delta\varepsilon(1 - \frac{\sigma}{|\sigma|}), & \text{if} |\sigma| = y_0. \end{cases}$$

Using predictor-corrector algorithm, we could summarize the algorithm as following:

```
Step 1: Compute $\sigma_{n+1}^{tr} = \sigma_n + E\Delta\epsilon$
Step 2: $|\sigma_{n+1}^{tr}| - \sigma_{Y,n} > 0$ ?
        No, set $\sigma_{n+1} = \sigma_{n+1}^{tr}$, $\sigma_{Y,n+1} = \sigma_{Y,n}$ and exit.
Step 3: Yes, compute $\Delta\lambda = \frac{|\sigma_{n+1}^{tr}| - \sigma_{Y,n}}{E+H'}$
Step 4: Set $\sigma_{n+1} = \sigma_{n+1}^{tr} - E\Delta\lambda \,\mathrm{sign}\,(\sigma_{n+1}^{tr})$ and
        $\sigma_{Y,n+1} = \sigma_{Y,n} + H'\Delta\lambda$ and exit.
```

For 3D problem, we need to express stress and strain with symmetric tensor(3x3). Therefore, we have a supervised learning problem with six inputs and six outputs. We want to generate enough data to train in advance so that the new stress-strain relation could be implemented in finite element program directly like a linear elasticity problem. Usually, the given condition is:

$$\varepsilon = \varepsilon^e + \varepsilon^p$$

We could write the elastic constitutive equation as:

$$\sigma = \mathbb{C}^e : \varepsilon^e = \mathbb{C}^e : (\varepsilon - \varepsilon^p)$$

or express using index notation.

$$\sigma_{ij} = c^e_{ijkl}\varepsilon^e_{kl} = c^e_{ijkl}(\varepsilon_{kl} - \varepsilon^p_{kl})$$

The yield condition now is:

$$f(\sigma, \kappa) = \sqrt{2J_2} - \kappa = ||s|| - \kappa$$

where $s = \sigma - \frac{1}{3}tr(\sigma)1$, the deviatoric stress. $||s|| = \sqrt{s_{ij}s_{ij}}$, $J_2 = \frac{1}{2}s : s = \frac{1}{2}s_{ij}s_{ij}$. As for the plastic strain,

$$\dot{\varepsilon}^p = \dot{\lambda}\frac{\partial f}{\partial \sigma} = \dot{\lambda}\hat{n}$$

, where $\hat{n} = s/||s|| = s/\kappa$. Starting from $\varepsilon = 0$ and $\sigma = 0$. To generate data for one specific cases, we use the following steps:

Step 1: Compute $\sigma^{trial}_{n+1} = \sigma_n + c^e : \Delta\varepsilon$, $p = \frac{1}{3}tr(\sigma^{trial}_{n+1})$, $s^{trial}_{n+1} = \sigma^{trial}_{n+1} - p1$

Step 2: Check $a = ||s^{tr}_{n+1} > \kappa$? No, set $\sigma_{n+1} = \sigma^{tr}_{n+1}$ and exit.

Step 3: Yes, set $\sigma_{n+1} = p1 + (\kappa/a)s^{tr}_{n+1}$ and exit.

For simplicity, consider linear elasticity. Which mean that

$$\sigma = (K - \frac{2}{3}\mu)1tr(\varepsilon) + 2\mu\varepsilon$$

In vigot expression, stress is expressed as

$$\sigma = \begin{pmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{12} & \sigma_{23} & \sigma_{13}\end{pmatrix}$$

$$\varepsilon = \begin{pmatrix}\varepsilon_{11} & \varepsilon_{22} & \varepsilon_{33} & \varepsilon_{12} & \varepsilon_{23} & \varepsilon_{13}\end{pmatrix}$$

$$\mathbb{C}^e = \begin{pmatrix} K + \frac{4\mu}{3} & K - \frac{2\mu}{3} & K - \frac{2\mu}{3} & 0 & 0 & 0 \\ K - \frac{2\mu}{3} & K + \frac{4\mu}{3} & K - \frac{2\mu}{3} & 0 & 0 & 0 \\ K - \frac{2\mu}{3} & K - \frac{2\mu}{3} & K + \frac{4\mu}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}$$

3.2 Time Dependent Model- Viscoelasticity

The problem we deal with is 3D linear viscoelasiticity. For the 3D case, isotropic viscoelasticity is characterized by two elastic moduli (resp. two viscosities, or equivalently two relaxation times) for each spring (resp. dashpot) element of the 1D model. Here, we will restrict to a simpler case in which one modulus is common to all elements (similar Poisson ratio for all elements), that is:

$$\boldsymbol{\sigma} = E_0\mathbb{C} : \boldsymbol{\varepsilon} + E_1\mathbb{C} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^v)$$

$$\dot{\boldsymbol{\varepsilon}}^v = \frac{E_1}{\eta_1}\mathbb{C} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^v)$$

where $\mathbb{C} = \frac{\nu}{(1+\nu)(1-2\nu)}\mathbf{1} \otimes \mathbf{1} + \frac{1}{1+\nu}\mathbb{I}$ with $\mathbf{1}$ and $\mathbb{I}$ being respectively the 2nd and 4th order identity tensors and ν being the Poisson ratio.

We use the formulas below to generate data.Formulating the time-discretized equations consists in approximating the viscous strain evolution equation using an implicit backward-Euler scheme at a given time t_{n+1}:

$$\frac{\varepsilon^{v,n+1} - \varepsilon^{v,n}}{\Delta t} \approx \dot{\varepsilon}^{v,n+1} = \frac{E_1}{\eta_1}\mathbb{C} : (\varepsilon^{n+1} - \varepsilon^{v,n+1})$$

which can be rewritten as:

$$\varepsilon^{v,n+1} = (\mathbb{I} + \frac{\Delta t E_1}{\eta_1}\mathbb{C})^{-1}(\varepsilon^{v,n} + \frac{\Delta t E_1}{\eta_1}\mathbb{C} : \varepsilon^{n+1})$$

Then:

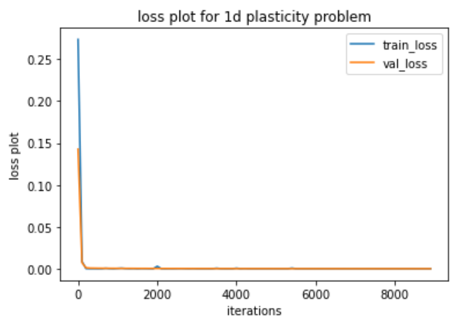
$$\boldsymbol{\sigma}_{n+1} = E_0\mathbb{C} : \boldsymbol{\varepsilon}_{n+1} + E_1\mathbb{C} : (\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^v_{n+1})$$

The algorithm above is used to generate data artificially. We want our network to perform a good match between stress and strain so that we could forget the complex equations here to avoid Newton-Raphson iteration during computation.

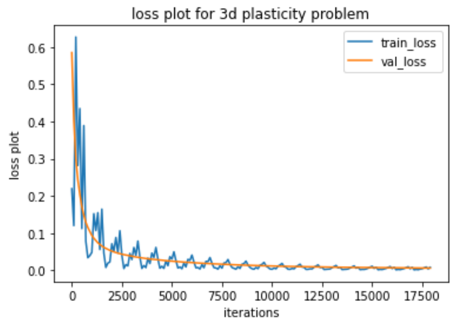
4 Method and Experiment

4.1 Time Independent Model

For perfect plasticity discussed above, we built neural networks with fully-connected layers to efficiently learn the underlying function from strain to stress. We set learning rate to be 0.01, optimizer to be stochastic gradient descent optimizer and loss function to be mean square error. Since we are working on regression problems, we will use the mean MSE across our validation set to select the best model and mean MSE for test set to determine the model performance.
For 1D perfect plasticity model, we found fully connected neural network with 1 hidden layer with 10 neurons and tanh activation function to be the best performing model, with validation loss and test loss to be 7.578e-06 and 7.455e-06 respectively.
For 3D perfect plasticity model. We experimented with 78 total variations, with activation function to be relu or tanh, number of hidden layer ranging from 1 to 3 and number of neurons to be 3, 5 or 10. For each of these models we set the epoch number to be 20. And based on the validation loss, the model with 3 hidden layer, with 10 neurons, 3 neurons and 10 neurons and activation function tanh, gives the best performance, with validation loss and test loss to be 0.003 and 0.004 respectively.

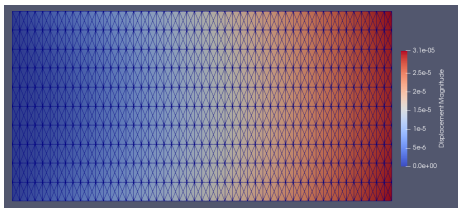


(a) 1d plasticity problem

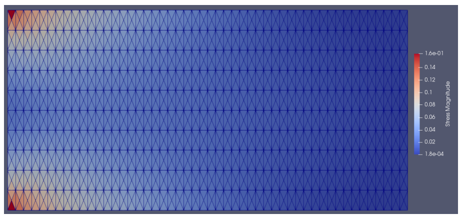


(b) 3d plasticity problem

The model trained then was implemented in a linear finite element program and was able to successfully calculate the real problems. The cantilever beam under the gravity load was simulated by the perfect plasticity model trained above and the result shows good correspondence to the exact solution.



(a) displacement problem

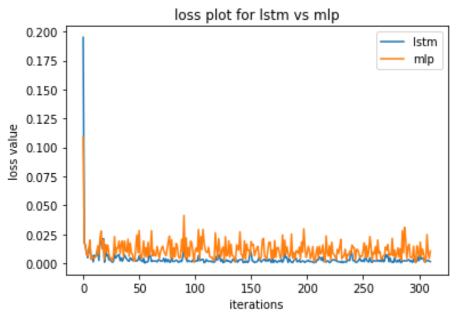


(b) stress problem

This actually provided a template for implementing the neural network in finite element program which is efficient and concise. The successful implementation of our model in finite element program also shows that our model is well trained.

4.2 Time Dependent Model

For time dependent model, we explored neural networks with full-connected layers and LSTMs on time series prediction tasks, aiming to predict the numerical values for stress given its values in previous time stamps. For our task, we choose sliding window in training and testing to be 10. We set learning rate to be 0.001, optimizer to be stochastic gradient descent optimizer and loss function to be mean square error.
In Full-connected neural network, we choose 2 hidden layers with size 10 and relu activation function. In LSTM model, we choose LSTM with 1 layer with size 5.
Below are the comparison of loss plot for both models.



We achieved 0.027 test loss for full-connected neural networks and 0.005 test loss for LSTMs, thus we can conclude that LSTM performed much better compared to vanilla ANN for our time series prediction task. This result falls into our expectation, as in the training process, LSTM model is able to capture the time dependency property of sequential training data, while vanilla ANN fails to do so.

5 Discussion

In this project, we are training regards different kinds of constitutive models, time dependent or time independent model. From our result, LSTM is suitable for time dependent model and fully connected model is suitable for time independent model. We only explore plasticity and viscoelasticity for brevity, but the conclusion could be used in other more complex models as well like the viscoplasticity. Given the trained data, we find a way to apply it in a finite element program so that the non-linearity is avoided which brings high efficiency and convergence.

6 Contributions

Wei Chen: Write finite element related program, train fully connected models and work on writeup
Xinwei He: Write finite element related program, train LSTM and work on writeup.
Jessica Zou: Do fully connected model training and work on writeup.
The work of this project is evenly assigned to the three members.

References

[1]G.W. Ellis, C. Yao, Rui Zhao, Df. Penumadu, Stress-strain modeling of sands using artificial neural networks, J. Geotech. Eng. 121 (5) (1995) 429–435.

[2]B.A. Le, Julien Yvonnet, Q.-C. He, Computational homogenization of nonlinear elastic materials using neural networks, Int. J. Numer. Methods Eng. 104 (12) (2015) 1061–1084.

[3]Tomonari Furukawa, Genki Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, Int. J. Numer. Methods Eng. 43 (2) (1998) 195–219.

[4]Kun Wang, WaiChing Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Eng. 334 (2018) 337–380.