Using Physics-Informed Neural Networks for Solving Forward and Inverse Problems in Solid and Fluid Mechanics

Presented by: Alexandros Papados (AMSC)
Advisor: Professor Balakumar Balachandran (ENME)

University of Maryland, College Park:
Applied Mathematics, Applied Statistics, & Scientific Computing

May 11, 2021

Table of Contents

Table of Contents

Project Proposal Recap

- Investigate PINNs and their ability to solve forward and inverse problems in solid and fluid mechanics
- Compare to classical numerical methods such FVM, FEM, and NLS
- **Problems in question:**
  - Conservation Laws - Burgers equation, Euler equations for compressible flow [1] – Fluid Mechanics
  - Plane stress linear elasticity boundary value problem [2] – Solid Mechanics

Why PINNs?

Advantages:

- Simplistic implementation to solve PDEs compared to FVM and FEM
- Parameter estimation requires less data and is faster than standard parameter estimation methods
- Meshless method
- Purpose is to "solve supervised learning tasks while respecting any given law of physics described by a general nonlinear partial differential equation" (Karniadakis et al.)

Drawbacks:

- Forward problem is slower than classical PDE solvers at times
- Weak theoretical grounding

PINNs Universal Approximation Theorem

Theorem (Pinkus, 1999):

Let $\boldsymbol{m}^i \in \mathbb{Z}_+^d$, $i = 1, \dots, s$,and set $m = \max_{i=1,\dots,s} |\boldsymbol{m}^i|$. Assume $\sigma \in C^m(\mathbb{R})$ and is not a polynomial. Then the space of single hidden layer neural nets:

$$\mathcal{M}(\sigma) = span\{\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

is dense in $C^{\boldsymbol{m}^1,\dots,\boldsymbol{m}^s}(\mathbb{R}^d)$. In other words, for any $f \in C^{\boldsymbol{m}^1,\dots,\boldsymbol{m}^s}(\mathbb{R}^d)$, any compact $K \subset \mathbb{R}^d$, and any $\epsilon > 0$, there exists a $g \in \mathcal{M}(\sigma)$ satisfying

$$\max_{\boldsymbol{x} \in K} \left| D^{\boldsymbol{k}} f(\boldsymbol{x}) - D^{\boldsymbol{k}} g(\boldsymbol{x}) \right| < \epsilon$$

for all $\boldsymbol{k} \in \mathbb{Z}_+^d$ for which $\boldsymbol{k} \leq \boldsymbol{m}^i$.

Introduction
W-PINNs-DE
Solid Mechanics
Linear Elasticity Boundary Value Problems
Remaining Work
Software and Coding Languages

- Euler Equations for compressible flow

1. Single Contact Discontinuity
   - Conserved form
   - Weighted loss function (W-PINNs)
2. Sod Shock Tube Problem
   - Characteristic Form
   - Weighted loss function and domain extended (W-PINNs-DE)
   - Major contribution to the study of PINNs
3. Inverse Problem

Table of Contents

W-PINNs-DE Validation

- Recall the characteristic form of the Euler equations

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{U}}{\partial x} = 0$$

where,

$$\boldsymbol{U} = \left(\rho, u, p\right)^T, \ \boldsymbol{A} = \begin{pmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{pmatrix}$$

where $a = \sqrt{\gamma p / \rho}$

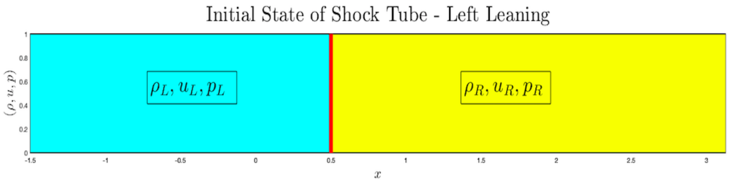
Problem	ρ_L	u_L	p_L	ρ_R	u_R	p_R
Single Contact Discontinuity	1.4	0.1	1.0	1.0	0.1	1.0
Double Expansion Fan	1.0	-1.0	0.4	1.0	1.0	0.4
Sod Shock Tube Problem	1.0	0.0	1.0	0.125	0.0	0.1
Reverse Shock Tube Problem	0.125	0.0	0.1	1.0	0.0	1.0
High Speed Shock Tube 1	0.125	0.0	0.1	1.0	0.75	1.0
High Speed Shock Tube 2	0.445	0.698	0.70	0.5	0.0	0.571

Domain Extension

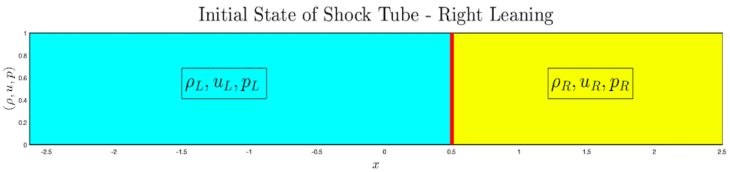
Given

$$\boldsymbol{u} = \begin{cases} \boldsymbol{u_L}, & x < x^* \\ \boldsymbol{u_R}, & x > x^* \end{cases}$$

$\boldsymbol{u_L} > \boldsymbol{u_R} \implies$ Left Leaning Extension, ex) $[0, 1] \to [-1.5, 3.125]$



$\boldsymbol{u_L} < \boldsymbol{u_R} \implies$ Right Leaning Extension, ex) $[0, 1] \to [-2.6175, 2.5]$



W-PINNs-DE Architecture

Weighted Loss Function:

$$G(\theta) = \frac{\omega_f}{N_f} \left\| \frac{\partial \tilde{\boldsymbol{U}}}{\partial t}(x,t,\theta) + \boldsymbol{A}\frac{\partial \tilde{\boldsymbol{U}}}{\partial x}(x,t,\theta) \right\|^2_{\Omega\times(0,T],\nu_1} + \frac{\omega_{IC}}{N_{IC}} \left\| \tilde{\boldsymbol{U}}(x,0,\theta) - \boldsymbol{U}(x,0) \right\|^2_{\Omega,\nu_2}$$

where $\omega_f = 0.1$ and $\omega_{IC} = 10$

- For each problem we sample points from the computational domain, $N_{x,t} = \{1000, 1000\}$.
- The learning rate is taken to be 0.0005
- Each neural network has 7 layers with 30 neurons per layer, with tanh() activation function for non-linear layers

Problem	**Epochs**	**Original Domain**, (x,t)	**Extended Domain**, (x,t)
Single Contact Discont.	44,350	$[0,1]\times[0,0.2]$	$[-1.5,3.125]\times[0,0.2]$
Double Expansion Fan	40,165	$[0,1]\times[0,0.2]$	$[-2.6175,2.5]\times[0,0.2]$
Sod Shock Tube	76,140	$[0,1]\times[0,0.2]$	$[-1.5,3.125]\times[0,0.2]$
Reverse Sod Shock Tube	76,140	$[0,1]\times[0,0.2]$	$[-2.625,2.5]\times[0,0.2]$
High-speed Shock Tube 1	55,765	$[-0.5,1.5]\times[0,0.2]$	$[-2.625,2.5]\times[0,0.2]$
High-speed Shock Tube 2	67,200	$[0,1]\times[0,0.2]$	$[-2.625,3.125]\times[0,0.2]$

Single Contact Discontinuity

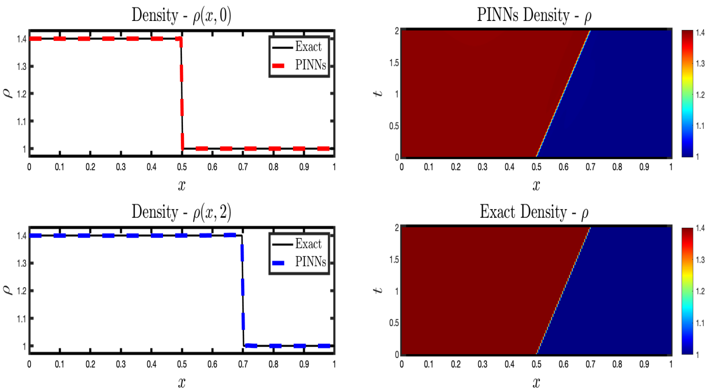


Figure 1: $N_f = 20,000$, $N_{IC} = 1000$

$\frac{\|\|\rho_{approx} - \rho_{exact}\|\|_2}{\|\|\rho_{exact}\|\|_2}$	$\frac{\|\|u_{approx} - u_{exact}\|\|_2}{\|\|u_{exact}\|\|_2}$	$\frac{\|\|p_{approx} - p_{exact}\|\|_2}{\|\|p_{exact}\|\|_2}$
$1.622e-04$	$1.2e-03$	$1.608e-04$

Sod Shock Tube Problem

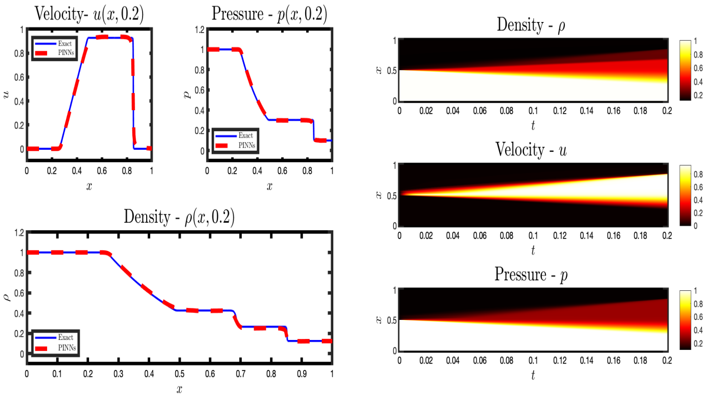


Figure 2: $N_f = 11,000$, $N_{IC} = 1000$

$\frac{\lvert\lvert \rho_{approx} - \rho_{exact} \rvert\rvert_2}{\lvert\lvert \rho_{exact} \rvert\rvert_2}$	$\frac{\lvert\lvert u_{approx} - u_{exact} \rvert\rvert_2}{\lvert\lvert u_{exact} \rvert\rvert_2}$	$\frac{\lvert\lvert p_{approx} - p_{exact} \rvert\rvert_2}{\lvert\lvert p_{exact} \rvert\rvert_2}$
$8.6e-03$	$6.29e-02$	$8.2e-03$

Reverse Sod Shock Tube Problem

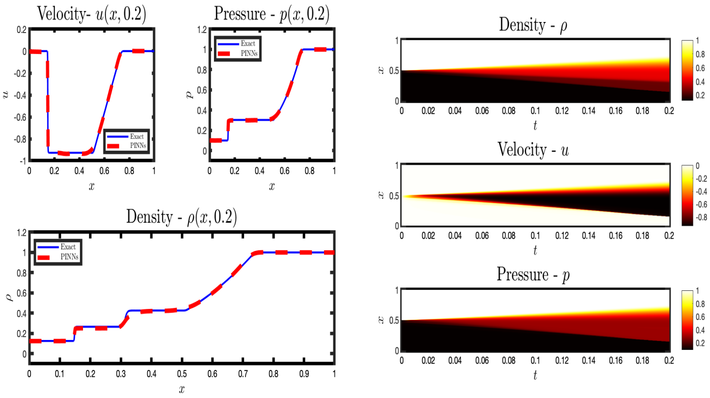


Figure 3: $N_f = 10,500$, $N_{IC} = 1000$

$\frac{\|\|\rho_{approx} - \rho_{exact}\|\|_2}{\|\|\rho_{exact}\|\|_2}$	$\frac{\|\|u_{approx} - u_{exact}\|\|_2}{\|\|u_{exact}\|\|_2}$	$\frac{\|\|p_{approx} - p_{exact}\|\|_2}{\|\|p_{exact}\|\|_2}$
$8.5e-03$	$3.04e-02$	$6.6e-03$

Double Expansion Fan

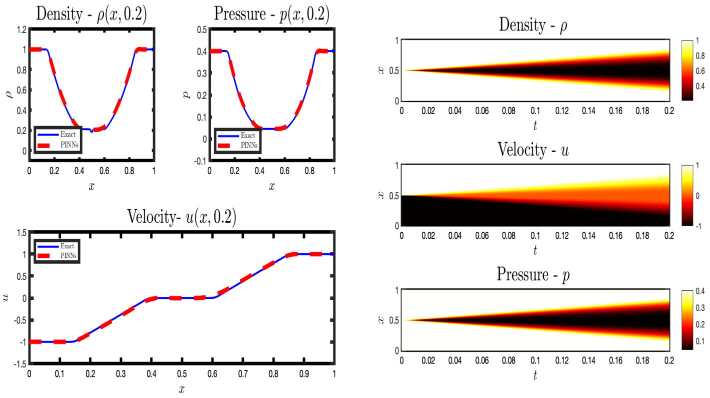


Figure 4: $N_f = 10,500$, $N_{IC} = 1000$

$\frac{\lVert \rho_{approx} - \rho_{exact} \rVert_2}{\lVert \rho_{exact} \rVert_2}$	$\frac{\lVert u_{approx} - u_{exact} \rVert_2}{\lVert u_{exact} \rVert_2}$	$\frac{\lVert p_{approx} - p_{exact} \rVert_2}{\lVert p_{exact} \rVert_2}$
$2.0e-03$	$2.9e-03$	$2.0e-03$

High Speed Shock Tube 1

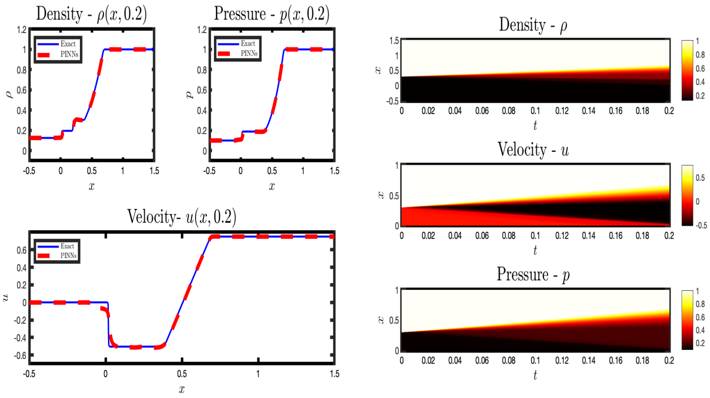


Figure 5: $N_f = 10,500$, $N_{IC} = 1000$

$\frac{\|\|\rho_{approx} - \rho_{exact}\|\|_2}{\|\|\rho_{exact}\|\|_2}$	$\frac{\|\|u_{approx} - u_{exact}\|\|_2}{\|\|u_{exact}\|\|_2}$	$\frac{p_{approx} - p_{exact}\|\|_2}{\|\|p_{exact}\|\|_2}$
$6.7e-03$	$3.29e-02$	$5.0e-03$

High Speed Shock Tube 2

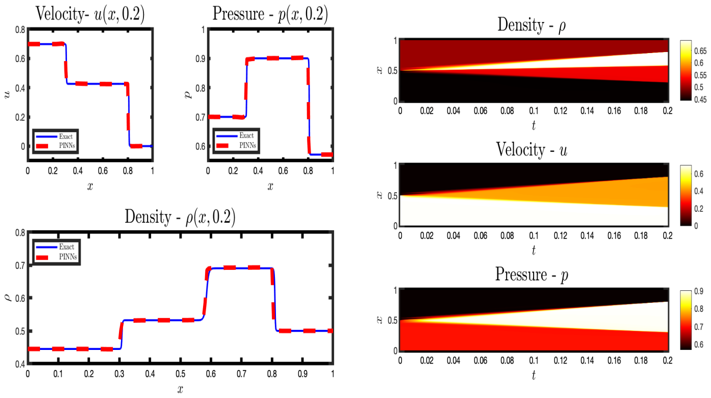


Figure 6: $N_f = 11,000$, $N_{IC} = 1000$

$\frac{\|\|\rho_{approx} - \rho_{exact}\|\|_2}{\|\|\rho_{exact}\|\|_2}$	$\frac{\|\|u_{approx} - u_{exact}\|\|_2}{\|\|u_{exact}\|\|_2}$	$\frac{\|\|p_{approx} - p_{exact}\|\|_2}{\|\|p_{exact}\|\|_2}$
$7.7e-03$	$1.4e-02$	$6.9e-03$

Loss Plots

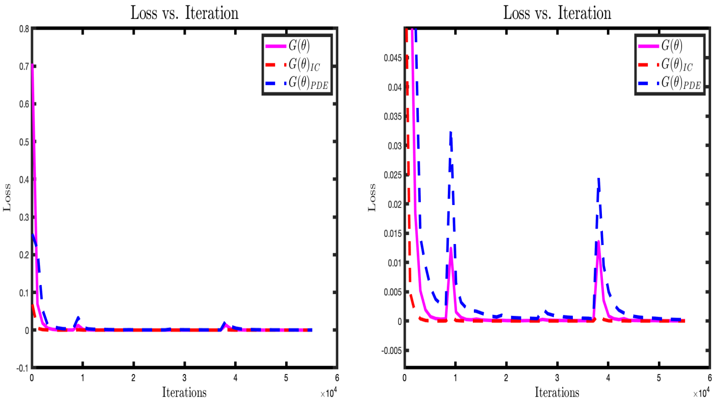


Figure 7: The plot on the left is a magnification of the plot on the right

Loss Plots

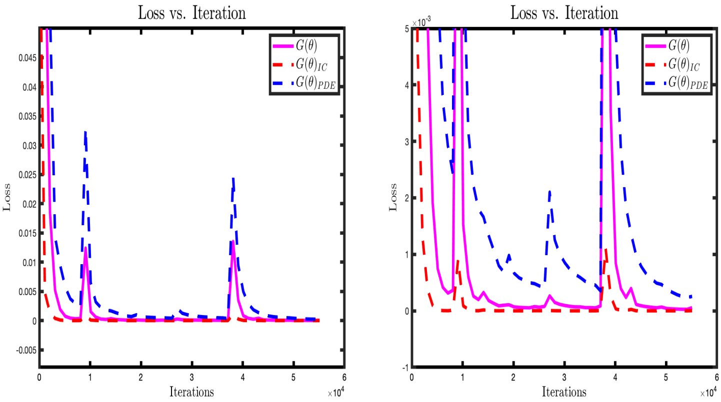


Figure 8: Magnification of the plot from previous slide

Conclusions W-PINNs-DE

- W-PINNs-DE can solve complicated shock tube problems to impressive degree of accuracy
- Competitive numerical tool to solve conservation laws
- Method may be used to solve isothermal Euler equations, and advection dominated compressible Navier-Stokes equations

Table of Contents

Table of Contents

LEBVP

- **Motivation:** Solid and Structural Mechanics
- The material matrix for an isotropic material in an elasticity boundary value problem consisting of two parameters, E - Young's Modulus, and ν - Poisson Ratio.
- Let $M_{E\nu} = \frac{E}{(1+\nu)(1-2\nu)}$. Then the material matrix is defined by:

$$C = M_{E\nu} \begin{pmatrix} 1-\nu & 0 & 0 & 0 & \nu & 0 & 0 & 0 & \nu \\ 0 & 1-2\nu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-2\nu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 & 0 & 0 & 0 \\ \nu & 0 & 0 & 0 & 1-\nu & 0 & 0 & 0 & \nu \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ \nu & 0 & 0 & 0 & \nu & 0 & 0 & 0 & 1-\nu \end{pmatrix}$$

- Solve for the amount of deformation a material undergoes under prescribed body loading, $\boldsymbol{f}$, and surface loading, $\boldsymbol{g}$

LEBVP

- The deformation tensor is defined as

$$\boldsymbol{u} = \left(u_1, u_2, u_3\right)^T$$

- u_i corresponds to the deformation in the x, y, and z direction, and $u_i : \mathbb{R}^3 \rightarrow \mathbb{R}$.

- We solve for the deformation of a material undergoing loading by solving the equilibrium equation:

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f}, & x \in \Omega \subset \mathbb{R}^3 \\ \boldsymbol{u} = 0, & x \in \Gamma_D \\ \boldsymbol{\sigma} \cdot \boldsymbol{\nu} = \boldsymbol{g}, & x \in \Gamma_N \end{cases} \tag{1}$$

where,

$$\boldsymbol{\sigma} = C\boldsymbol{\epsilon}, \quad \epsilon_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right] + \frac{1}{2}\sum_{k=1}^{3} \frac{\partial u_i}{\partial x_k}\frac{\partial u_j}{\partial x_k}, \quad i, j = 1, 2, 3$$

LEBVP

Since we are considering a LEBVP, the parabolic terms vanish, hence

$$
\begin{aligned}
\boldsymbol{\epsilon} &= \frac{1}{2}\left[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right] \\
&= A \nabla \boldsymbol{u} \\
&= \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\frac{\partial u_1}{\partial x_1} \\
\frac{\partial u_1}{\partial x_2} \\
\frac{\partial u_1}{\partial x_3} \\
\frac{\partial u_2}{\partial x_1} \\
\frac{\partial u_2}{\partial x_2} \\
\frac{\partial u_2}{\partial x_3} \\
\frac{\partial u_3}{\partial x_1} \\
\frac{\partial u_3}{\partial x_2} \\
\frac{\partial u_3}{\partial x_3}
\end{pmatrix}
\end{aligned}
$$

Plane Stress

A material undergoes plane stress provided the stress vector is zero in a specific plane. Here we chose to have zero stresses in the $z - direction$, hence,

$$\sigma_{3j} = \sigma_{i3} = 0, \quad \text{for } i, j = 1, 2, 3$$

Then the stress tensor in the $xy - direction$ is defined by:

$$\begin{aligned} \boldsymbol{\sigma} &= C_{E\nu}\boldsymbol{\epsilon} \\ &= \frac{E}{(1-\nu^2)} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{pmatrix} \begin{pmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{pmatrix} \end{aligned}$$

where $\gamma_{12} = \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right)$

Forward Problem

LEBVP

$$G\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right]+G\left(\frac{1+\nu}{1-\nu}\right)\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 v}{\partial y \partial x}\right]=\sin(2\pi x)\sin(2\pi y)$$

$$G\left[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right]+G\left(\frac{1+\nu}{1-\nu}\right)\left[\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 u}{\partial x \partial y}\right]=\sin(\pi x)+\sin(2\pi y)$$

where $G=\frac{E}{2(1+\nu)}$, $E=1$ is the Young's modulus, and $\nu=0.3$ is the Poisson ratio of the material. The problem has fixed boundary conditions.

Domains

We will solve this problem on two domains, $[0, 1]^2$ and $[0, 1]^2 \backslash [0, 0.5] \times [0.5, 1]$.

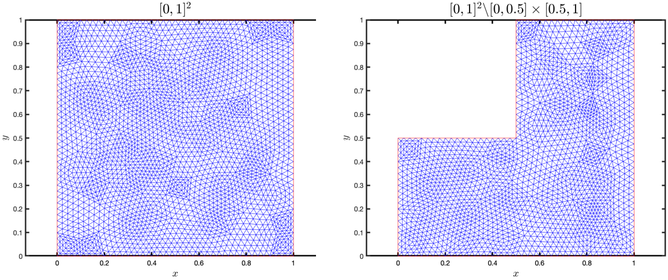


Figure 9: Domains for Plane Stress Problem

Issues Using PINNs

- PINNs have much difficulty approximating simple boundary conditions
- Immense error at the boundary

Preliminary Work - Square Domain - $[0, 1]^2$

Weighted Loss Function

Introduce a serve penalty for the BC loss term:

$$G(\theta) = G_{pde}(\theta) + \lambda_{bc} G_{bc}(\theta)$$

where $\lambda_{bc} = 10000$.

- We sample points from the computational domain, $N_{x,y} = \{500, 500\}$.
- The learning rate is taken to be 0.001
- Each neural network has 6 layers with 50 neurons per layer, with tanh() activation function for non-linear layers
- $N_f = 4000$, $N_{BC} = 500$, $epochs = 200,000$

Displacement in x

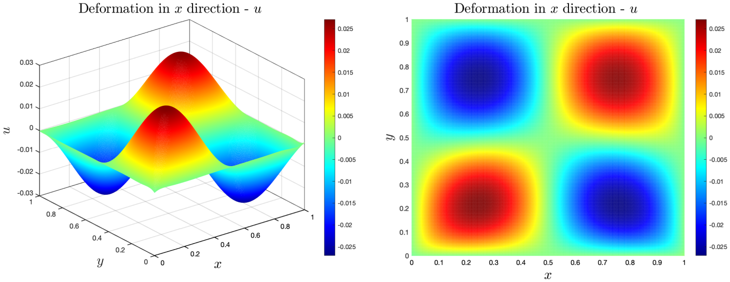


Figure 10: Displacement in x direction (PINNs)

Displacement in y

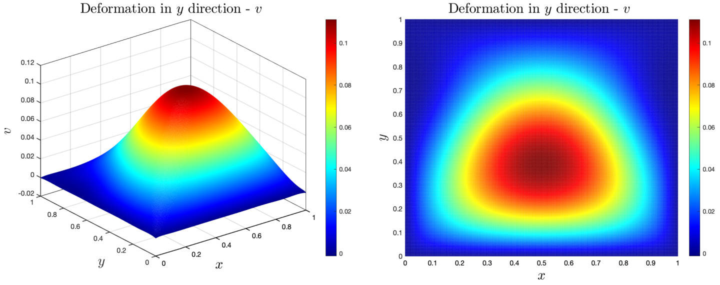


Figure 11: Displacement in y direction (PINNs)

FEM Comparison

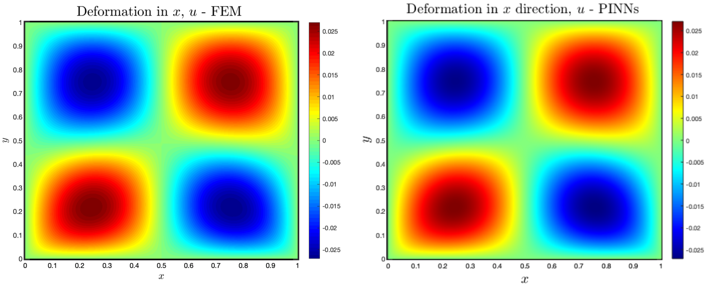


Figure 12: The left figure is the FEM solution, right is the PINNs solution of the displacement in the x-direction

FEM Comparison

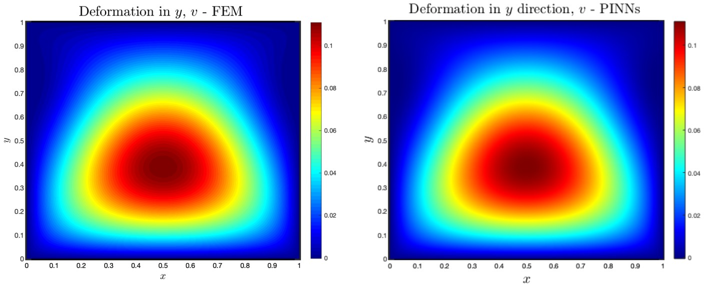


Figure 13: The left is the FEM solution, right is the PINNs solution of the displacement in the y-direction

Table of Contents

Remaining Work

- Solve LEBVP for both domains
- Compare to FEM
- Inverse Problem

Table of Contents

Software and Coding Languages

Coding Languages

- Python
- MATLAB

Libraries

- PyTorch

Reference

[1] Roesner, K. G., Leutloff, D., Srivastava, R. C. (1995). *Computational fluid dynamics: Selected topics*. Berlin: Springer.

[2] Chen, Y., Press, H. H. (2013). *Computational Solid Mechanics Structural Analysis and Algorithms*. Berlin: De Gruyter.

[3] Thomas, J. W. (1999). *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. New York: Springer.

[4] Golsorkhi, N. A., Tehrani, H. A. (2014). *Levenberg-marquardt Method For Solving The Inverse Heat Transfer Problems*. Journal of Mathematics and Computer Science, 13(04), 300-310. doi:10.22436/jmcs.013.04.03

[5] Chen, Z. (2010). *Finite Element Methods and their Applications*. Berlin: Springer.

[6] Mao, Z., Jagtap, A. D., Karniadakis, G. E. (2020). *Physics-informed neural networks for high-speed flows*. Computer Methods in Applied Mechanics and Engineering, 360, 112789. doi:10.1016/j.cma.2019.112789

Reference

[7] Raissi, M., Perdikaris, P., Karniadakis, G. (2019). *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. doi:10.1016/j.jcp.2018.10.045

[8] Sirignano, J., Spiliopoulos, K. (2018). *DGM: A deep learning algorithm for solving partial differential equations*. Journal of Computational Physics, 375, 1339-1364. doi:10.1016/j.jcp.2018.08.029

[9] Lu, L.,Jagtap, A. D., Karniadakis, G. E. (2019). *DeepXDE: A Deep Learning Library for Solving Differential Equations*. ArXiv.org,arxiv.org/abs/1907.04502.

[10] Cybenko, G. (1989). *Approximation by superpositions of a sigmoidal function*. Mathematics of Control, Signals, and Systems, 2(4), 303-314. doi:10.1007/bf02551274

[11] Mishra, S., amp; Molinaro, R. (2020). *Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs II: A class of inverse problems*. https://arxiv.org/abs/2007.01138

The End

Thank You! Questions?