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

September 29, 2020

Table of Contents

Table of Contents

Background

- Since the late 90s researchers have been studying how to solve PDEs using neural networks
- Scientific machine learning has become a buzzing area of research since 2018
- PINNs are first introduced in the paper *Physics-Informed Neural Networks: A deep learning framework for solving forward and inverse problems involving partial differential equations* published in the *Journal of Computational Physics*
- **Authors:**
  - Dr. George Karniadakis, Brown University
  - Dr. Maziar Raissi, Colorado University, Boulder
  - Dr. Paris Perdiakaris of the University of Pennsylvania
- This paper inspired me to research PINNs and their application to problems that arise in solid and fluid mechanics, particularly for parameter estimation

Why PINNs?

- Provide an easy-to-use framework to solve forward and inverse problems successfully, with impressive degrees of accuracy
- Meshless method
- Research on PINNs is in high demand
- Purpose are to "solve supervised learning tasks while respecting any given law of physics described by a general nonlinear partial differential equation" (Karniadakis et al.)

Project Proposal

- Investigate PINNs and their ability to solve forward and inverse problems in solid and fluid mechanics
- Compare to classical numerical methods
- **Three types of equations:**
  - Euler equations [1] for an incompressible and compressible fluid in 1/2-D – Fluid Mechanics
  - Navier-Stokes Equations [1] for a compressible fluid in 1/2-D – Fluid Mechanics
  - Plane stress linear elasticity boundary value problem [2] in 3-D (nonlinear if there is time) – Solid Mechanics

Table of Contents

PINNs Algorithm

- The PINNs algorithm stems from the Deep Galerkin Method (Sirignano et al., 2018).
- Difference between the two is that PINNs have incorporated inversion capabilities where DGM has not

Problem Statement

Consider a general nonlinear PDE:

$$\begin{cases} \frac{\partial u}{\partial t} + \mathcal{L}(u, a) = 0, & (x, t) \in \Omega \times \mathbb{R}^+ \\ u(x, t) = g(x, t), & (x, t) \in \Gamma \times \mathbb{R}^+ \\ u(x, 0) = h(x), & x \in \Omega \end{cases} \tag{1}$$

where $\mathcal{L}$ is a nonlinear differential operator, $\Omega \subset \mathbb{R}^n$, $u(x, t)$ be the exact solution to (1), and a is an arbitrary physical parameter for the governing PDE.

PINNs Algorithm - Forward Problem

1. Generate weights $\theta \in \mathbb{R}^k$ and DNN, $f(t, x, \theta)$, where number of layers, neurons per layer, and activation functions for each layer are prescribed by the user
2. Generate random points (x_n, t_n) from $\Omega \times \mathbb{R}^+$, (z_n, τ_n) from $\Gamma \times \mathbb{R}^+$, and w_n from Ω, according to respective probability densities ν_1, ν_2, and ν_3
3. Calculate the squared error $G(\theta_n, s_n)$ at randomly sampled points $s_n = \{(x_n, t_n), (z_n, \tau_n), w_n\}$ where:

$$\begin{aligned} G(\theta_n, s_n) = &\left(\frac{\partial f}{\partial t}(x_n, t_n, \theta_n) + \mathcal{L}(f(x_n, t_n, \theta_n), a) \right)^2 \\ &+ (f(z_n, \tau_n, \theta_n) - g(z_n, \tau_n))^2 \\ &+ (f(w_n, 0, \theta_n) - h(w_n))^2 \end{aligned}$$

4. Perform stochastic gradient decent at random points s_n :

$$\theta_{n+1} = \theta_n - \eta_n \nabla_\theta G(\theta_n, s_n)$$

PINNs - Inverse Problem

1. Generate weights $\theta \in \mathbb{R}^k$ and DNN, $f(t, x, \theta)$, where number of layers, neurons per layer, and activation functions for each layer are prescribed by the user
2. Generate random points (x_n, t_n) from $\Omega \times \mathbb{R}^+$, (z_n, τ_n) from $\Gamma \times \mathbb{R}^+$, and w_n from Ω, according to respective probability densities ν_1, ν_2, and ν_3, and generate initial guess for the physical parameter, a
3. Calculate the squared error $G(\theta_n, s_n)$ at randomly sampled points $s_n = \{(x_n, t_n), (z_n, \tau_n), w_n\}$ where:

$$G(\theta_n, s_n) = \left(\frac{\partial f}{\partial t}(x_n, t_n, \theta_n) + \mathcal{L}(f(x_n, t_n, \theta_n), a) \right)^2 + (f(z_n, \tau_n, \theta_n) - g(z_n, \tau_n))^2 + (f(w_n, 0, \theta_n) - h(w_n))^2 + (f(x_n, t_n, \theta_n) - u(x_n, t_n))^2$$

4. Perform stochastic gradient decent at random points s_n :

$$\theta_{n+1} = \theta_n - \eta_n \nabla_\theta G(\theta_n, s_n)$$

Table of Contents

Euler Equations - Compressible Flow

Consider the Euler equations in conserved form for compressible flow:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = 0 \qquad (2)$$

In 1-D the conserved variables and fluxes are represented by:

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \quad \boldsymbol{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u\left(\rho E + p\right) \end{pmatrix} \qquad (3)$$

And in 2-D

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad F_1 = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ p + \rho uv \\ pu + \rho uE \end{pmatrix}, \quad F_2 = \begin{pmatrix} \rho v \\ p + \rho uv \\ p + \rho v^2 \\ pv + \rho vE \end{pmatrix} \qquad (4)$$

Euler Equations - Compressible Flow

- ρ is the density
- p is the pressure
- u, v correspond to the velocity in the x, y-directions respectively
- E is the total energy of the fluid.

Equation of State

Consider the equation of state based on the ideal gas-law. The equation of state relates the pressure and energy of the fluid by:

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho||\boldsymbol{u}||^2\right], \quad \boldsymbol{u} = (u, v) \tag{5}$$

where γ is the heat capacity ratio.

Forward Problem

- Use PINNs to solve for the physical quantities $\rho, u,$ and p in 1-D, and additionally v in 2-D.
- We consider the shock tube problem in both dimensions.
- For the 1-D problem, we will reproduce the results from reference [1] as a starting point for the project.
- All remaining problems are original work

Forward Problem

1-D (Mao, et all)

$$\begin{cases} \dfrac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = 0, & (x,t) \in (0,1) \times (0,2] \\ \Big(\rho, u, p\Big)_{t=0} = \begin{cases} (1.4, 0.1, 1.0), & 0 \le x < 0.5 \\ (1.0, 0.1, 1.0), & 0.5 \le x \le 1 \end{cases} \end{cases} \tag{6}$$

with Dirichlet boundary conditions which take the values of the initial condition at each boundary point.

2-D

$$\begin{cases} \dfrac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = 0, & (x,y,t) \in (0,1)^2 \times (0,0.3] \\ \Big(\rho, u, v, p\Big)_{t=0} = \begin{cases} (0.5323, 1.206, 0.0, 0.3), & 0 \le x,y < 0.5 \\ (0.138, 1.206, 1.206, 0.029), & 0.5 \le x,y \le 1 \end{cases} \end{cases}$$

with Dirichlet boundary conditions which take on the values of the initial condition at each boundary point.

We take $\gamma = 1.4$ in both problems.

Forward Problem

- After obtaining the PINNs solutions, we will compare them to a finite volume approximation with flux limiting [3]
- **Finite Volume Method with Flux Limiting:**
  - RK-2 in time
  - Lax-Friedrichs flux solver
  - Osher flux limiter
- Comparative study will measure run-time, shock capturing, and accuracy

Inverse Problem

- The physical parameter for the Euler equations is the heat capacity ratio, γ
- For the inverse problem, we wish to use PINNs to obtain the original parameter $\gamma = 1.4$, based on three sets of target data and inputs to the network
- The three cases in consideration are:
  1. A full analytic solution data set is used as target data, with initial and boundary conditions prescribed to the network
  2. A full analytic solution data set is used as target data, with no initial and boundary conditions prescribed to the network
  3. A partial analytic solution data set with 1% noise is used as target data, with initial and boundary conditions prescribed to the network

Inverse Problem

- For each case, we will measure how well PINNs can calculate γ comparatively to the nonlinear least square method, using the Levenberg–Marquardt algorithm [4]
- The Levenberg–Marquardt method is a damped nonlinear least-squares algorithm
- MATLABs built-in function ***fmincon*()** provides a framework for solving NLS problems using the Levenberg–Marquardt algorithm

Euler Equations - Incompressible Flow

An incompressible fluid is a fluid whose density remains constant as pressure changes. Hence each derivative of ρ in (2) will be zero. This results in Euler equations for incompressible flow to be of the form:

$$\begin{cases} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} & + \frac{\nabla p}{\rho} = 0 \\ & \nabla \cdot \boldsymbol{u} = 0 \end{cases} \tag{7}$$

- Define the density to be $\rho = 1g/cm^3$, which corresponds to the density of water
- Solve for the physical quantities u and p in 1-D, as well as for v in 2-D.
- Prescribe continuous initial conditions and Neumann boundary conditions.

Forward Problem

1-D

$$\begin{cases} \frac{\partial u}{\partial t} + (u \cdot \nabla) u + \frac{\nabla p}{\rho} = 0, \quad \nabla \cdot u = 0, \ (x,t) \in (0,\pi) \times (0,1] \\ u(x,0) = 1.0 - 0.25 \sin(4\pi x) - 0.25 \sin(8\pi x), \quad x \in (0,\pi) \\ p(x,0) = 1.0, \quad x \in (0,\pi) \\ \frac{\partial u}{\partial \nu} = \frac{\partial p}{\partial \nu} = 0, \quad t \in (0,1] \end{cases}$$

2-D

$$\begin{cases} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} + \frac{\nabla p}{\rho} = 0, \quad \nabla \cdot \boldsymbol{u} = 0, \quad (x,y,t) \in (0,\pi)^2 \times (0,1] \\ u_{t=0} = 1.0 - 0.25 \sin(4\pi x) - 0.25 \sin(8\pi x), x \in (0,\pi) \\ v_{t=0} = 1.0 - 0.25 \cos(4\pi y) - 0.25 \cos(8\pi y), \ y \in (0,\pi) \\ p_{t=0} = 1.0, \quad (x,y) \in (0,\pi)^2 \\ \frac{\partial \boldsymbol{u}}{\partial \nu} = \frac{\partial p}{\partial \nu} = 0, \quad t \in (0,1] \end{cases}$$

- A similar framework is used for that of compressible flow

Table of Contents

Navier-Stokes Equation - Compressible Flow

Consider the Navier-Stokes equations for compressible flow:

$$\begin{cases} \rho \boldsymbol{u}_t + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \mu \nabla^2 \boldsymbol{u} - (\lambda + \mu) \nabla \nabla \cdot \boldsymbol{u} + \nabla p = 0 \\ \rho_t + \nabla \cdot (\rho \boldsymbol{u}) = 0 \end{cases} \tag{8}$$

- $\boldsymbol{u} = u$ in 1-D and $\boldsymbol{u} = (u, v)$ in 2-D, is the velocity in the x, y direction respectively
- p is the pressure
- ρ is the density
- λ is the bulk viscosity
- μ is the dynamic viscosity.

Remark: The physical parameters λ and μ are referred to as the Lamé parameters.

Forward Problem

- In 1-D, impose discontinuous initial conditions for each physical quantity with periodic boundary condition
- In 2-D, impose continuous initial conditions, with periodic boundary conditions.
- The bulk viscosity, $\lambda = 0.001$, and the dynamic viscosity, $\mu = 0.014$

1-D Hydrodynamic Sod Problem

$$\begin{cases} \rho u_t + \rho u \cdot \nabla u - \mu \nabla^2 u - (\lambda + \mu) \nabla \nabla \cdot u + \nabla p = 0, & \in (0,1) \times (0, 0.2] \\ \rho_t + \nabla \cdot (\rho u) = 0, \quad (x,t) \in (0,1) \times (0, 0.2] \\ \Big(\rho, u, p\Big)_{t=0} = \begin{cases} (1.0, 0.0, 1.0), & 0 \leq x < 0.5 \\ (0.125, 0.0, 0.1), & 0.5 \leq x \leq 1 \end{cases} \\ \Big(\rho, u, p\Big)_{x=0} = \Big(\rho, u, p\Big)_{x=1}, \quad t \in (0, 0.2) \end{cases}$$

- **Finite Volume Method with Flux Limiting (1-D):**
  - RK-3 in time
  - Lax-Friedrichs flux and second order viscous solver
  - Osher flux limiter

Forward Problem

2-D

$$\begin{cases} \rho \boldsymbol{u}_t + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \mu \nabla^2 \boldsymbol{u} - (\lambda + \mu) \nabla \nabla \cdot \boldsymbol{u} + \nabla p = 0, \quad (0,\pi)^2 \times (0,1] \\ \rho_t + \nabla \cdot (\rho \boldsymbol{u}) = 0, \quad (0,\pi)^2 \times (0,1] \\ \rho_{t=0} = (1.0 - 0.25\cos(8\pi x))(1 - 0.20\sin(4y)), \ (x,y) \in (0,\pi)^2 \\ p_{t=0} = 1.0, \quad (x,y) \in (0,\pi)^2 \\ u_{t=0} = 1.0 - 0.25\sin(4\pi x) - 0.25\sin(8\pi x), \ x \in (0,\pi) \\ v_{t=0} = 1.0 - 0.25\cos(4\pi y) - 0.25\cos(8\pi y), \ y \in (0,\pi) \\ \Big(\rho, u, p\Big)_{x=0} = \Big(\rho, u, p\Big)_{x=\pi}, \quad (x,t) \in (0,\pi) \times (0,1] \\ \Big(\rho, v, p\Big)_{y=0} = \Big(\rho, v, p\Big)_{y=\pi}, \quad (y,t) \in (0,\pi) \times (0,1] \end{cases}$$

- **Finite Volume Method without Flux Limiting (2-D):**
  - RK-3 in time
  - Lax-Friedrichs flux and second order viscous solver

Table of Contents

- **Motivation:** Solid and Structural Mechanics
- The material matrix for an isotropic material in an elasticity boundary value problem consists 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 define 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 $u_i : \mathbb{R}^3 \to \mathbb{R}$.
- We solve for the deformation of a material undergoing loading by solving the equilibrium equation:

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f}, & x \in \Omega \subset \mathbb{R}^3 \\ \boldsymbol{u} = \boldsymbol{0}, & x \in \Gamma_D \\ \boldsymbol{\sigma} \cdot \boldsymbol{\nu} = \boldsymbol{g}, & x \in \Gamma_N \end{cases} \tag{9}$$

where,

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

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

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

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

- Let the Young's Modulus, $E = 35\ GPa$, and the Poisson Ratio, $\nu = 0.16$
- Define the body force vector to be:

$$\boldsymbol{f} = \begin{pmatrix} \sin(\pi x) + \sin(2\pi y) \\ \sin^2(\pi y) \\ \sin(\pi x)\sin(2\pi y) \end{pmatrix}, \quad (x, y) \in (0, \pi)$$

- Take the surface force $\boldsymbol{g} = \boldsymbol{0}$.

LEBVP - Plane Stress

Let $\Omega = [0, \pi]^2$. Then, the LEBVP in consideration is of the form:

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f}, & x \in \Omega \\ \boldsymbol{u} = \boldsymbol{0}, & x \in \Gamma_D \\ \boldsymbol{\sigma} \cdot \boldsymbol{\nu} = \boldsymbol{0}, & x \in \Gamma_N \end{cases}$$

- We compare the PINNs approximated solution of $\boldsymbol{u}, \boldsymbol{\epsilon}$, and $\boldsymbol{\sigma}$ to an finite element approximation [5] of the physical quantities.

Table of Contents

Software and Coding Languages

Coding Languages

- Python
- MATLAB
- C++

Libraries

- DEEPXDE
- Tensorflow
- PyTorch

Table of Contents

Validation

To ensure a proper PINNs solution is obtained:

- Measure the relative L_2 error between the PINNs solution and an analytic solution, a classical numerical solution, or experimental data

$$\mathcal{E} = \frac{||\hat{u}_{PINNs} - u_{Exact}||_2}{||u_{Exact}||_2}$$

Universal Approximation Theorem (Cybenko, 1989)

Let $x \in I^n \equiv [0,1]^n$ and define

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma\left(y_j^T x + \theta_j\right)$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous discriminatory function, $y_j \in \mathbb{R}^n$, and $\alpha_j, \theta \in \mathbb{R}$ are fixed .Then $G(x)$ is dense in $C(I^n)$. In other words, for any $f \in \mathcal{C}(I^n)$, and $\epsilon > 0$, there exists a finite sum of the form above, $G(x)$, such that

$$|G(x) - f(x)| < \epsilon, \quad \forall x \in I^n$$

- Under the assumptions of the UAT, solutions of PDEs can be approximated by neural networks
- (Mishra and Molinaro, 2020) relate number of layers, neurons per layer, and activation functions used, to obtain error estimates

Table of Contents

Milestones

- **November 1st, 2020:** Comparative study for the Euler equations for compressible flow
- **December 15th, 2020:** Comparative study for the Euler equations for incompressible flow
- **February 24th, 2021:** Comparative study for the Navier-Stokes equations for compressible flow
- **April 1st, 2021:** Comparative study for the LEBVP
- **April 20th 2021:** Comparative study for the NLEBVP

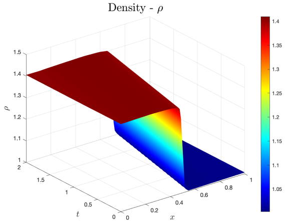
Table of Contents

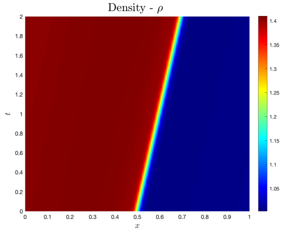
Preliminary Results

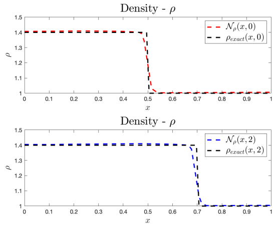
Mao et al. (2020), the authors solve a 1-D Riemann Problem for the Euler equations of a compressible fluid, of the form of the (6)

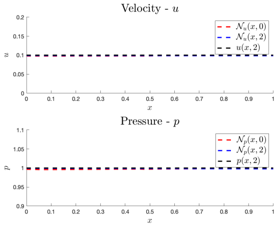
1. A deep neural network (DNN) with 6 layers, each layer with 40 neurons and consider two cases:
   - $N_{x,t} = \{100, 100\}$, $N_f = 10000$, $N_{BC} = 50$, $N_{IC} = 100$
   - $N_{x,t} = \{1000, 1000\}$, $N_f = 10000$, $N_{BC} = 50$, $N_{IC} = 1000$
2. We compare to FVM with flux limiting where:
   - $N_{x,t} = \{100, 514\}$
   - $N_{x,t} = \{1000, 5133\}$
3. To validate each method the relative L_2 norm of each physical quantity is taken with respect to the analytic solution

PINNs Solution $-N_{x,t} = \{100, 100\}$

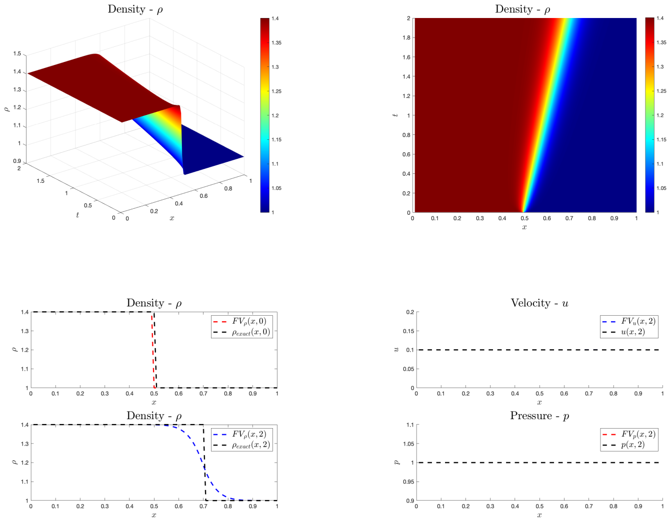




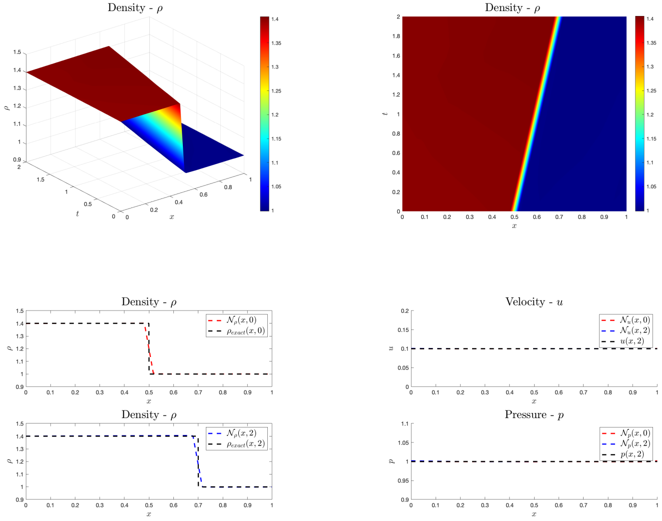




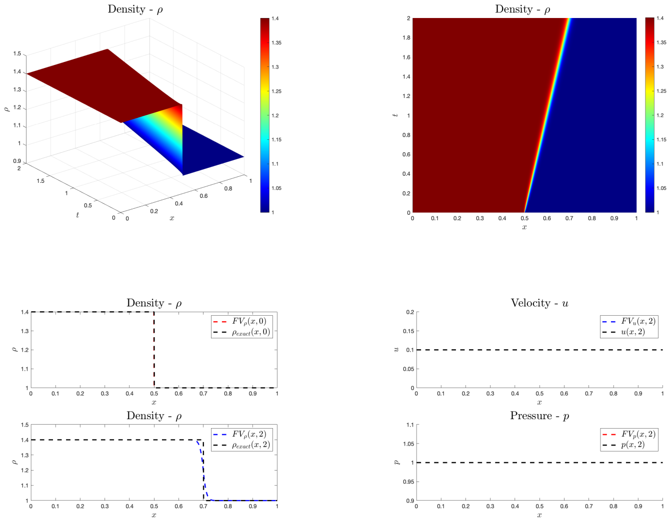
Finite Volume Solution - $N_{x,t} = \{100, 514\}$



PINNs Solution - $N_{x,t} = \{1000, 1000\}$



Finite Volume Solution - $N_{x,t} = \{1000, 5133\}$



Results

Method	PINNs$_{\{100,100\}}$	PINNs$_{\{1000,1000\}}$	FVM$_{\{100,514\}}$	FVM$_{\{1000,5133\}}$
$\frac{\|\|\rho_{approx} - \rho_{exact}\|\|_2}{\|\|\rho_{exact}\|\|_2}$	5.0e-03	4.2e-03	1.35e-02	3.5e-03
$\frac{\|\|u_{approx} - u_{exact}\|\|_2}{\|\|u_{exact}\|\|_2}$	1.11e-02	5.1e-03	5.308e-04	5.335e-05
$\frac{\|\|p_{approx} - p_{exact}\|\|_2}{\|\|p_{exact}\|\|_2}$	3.0e-04	1.0e-04	3.649e-05	3.652e-06
CPU	923 sec	323 secs	1.41 secs	361.63 secs

Observations

- PINNs experience some form of dissipation much like the FVM solution, except, less points are need to capture shock
- Less points required more training, hence higher CPU, whereas more points required less training and hence smaller CPU
- FVM required more points and effort
- PINNs are easier to implement

Inverse Problem and Results

Two out of three cases for the inversion problem are presented:

- A full analytic solution data set is used as target data, with initial and boundary conditions prescribed to the network
- A full analytic solution data set is used as target data, with no initial and boundary conditions prescribed to the network

A DNN with 4 layers, each layer with 60 neurons is constructed.

- $N_{x,t} = \{1000, 100\}$, $N_f = 2000$, $N_{BC} = 100$, and $N_{IC} = 1000$
- For each layer we use the tanh activation function

Method	BC/ICs Prescribed	Full Solution Dataset	γ_{exact}	γ_{approx}	CPU
PINNs	Yes	Yes	1.40	1.405	107.71 secs
PINNs	No	Yes	1.40	1.404	40.31 sec
NLS	Yes	Yes	1.40	1.410	173 secs

Observations

- BCs/ICs and full solution data sets must be prescribed for parameter estimation using NLS
- PINNs do not require BC/ICs and full solution data prescribed to the network
- PINNs have predictive capabilities, whereas NLS does not
- NLS only obtained γ when total energy, E, solution data was provided, and could not approximate γ when pressure, p, solution data was provided

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