

Physics-informed neural networks to solve the compressible Euler equations

Dario Rodriguez (AE)

Overview

- Approximate solution of the Euler equations using a neural network (forward method)

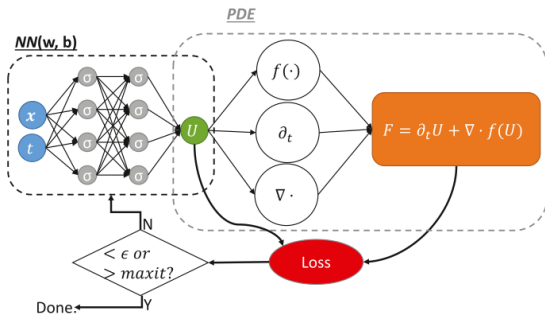
1-D Compressible Euler equations

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

where,

$$U = (\rho, u, p)^T$$
$$A = \begin{bmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{bmatrix}$$
$$a = \sqrt{\gamma p / \rho}$$

- 2 inputs (x, t)
- 3 states outputs (ρ, u, p)



Loss function

- In general, the loss function is defined as follows:

$$L(\theta) = \frac{1}{N_f} \left| \frac{\partial \tilde{U}}{\partial t}(x, t, \theta) + \tilde{A} \frac{\partial \tilde{U}}{\partial x}(x, t, \theta) \right|_{\Omega \times (0, T]}^2 + \frac{1}{N_{IC}} \left| \tilde{U}(x, 0, \theta) - U(x, 0) \right|_{\Omega}^2 + \frac{1}{N_{BC}} \left| \tilde{U}(x, t, \omega) - U(x, t) \right|_{\partial \Omega \times (0, T]}^2$$

- Problem-specific → Sod shock tube problem

$$\begin{bmatrix} \rho_L \\ u_L \\ P_L \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

$$\begin{bmatrix} \rho_L \\ u_L \\ P_L \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.0 \\ 0.1 \end{bmatrix}$$

Advances

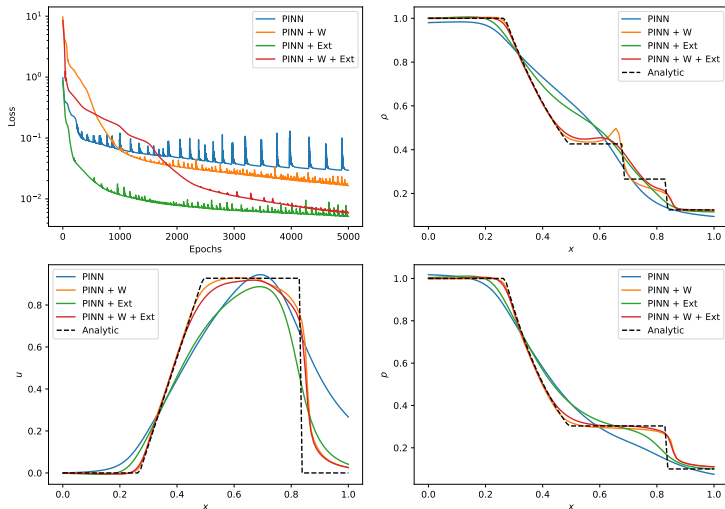
A public repository  [link](#) was setup to host the models implemented

Four different models were trained by using a neural network (30-neuron width & 7 hidden layers), a learning rate of 0.0005 and the classical Adam optimizer. The epochs for these cases were set to 5000

- ▶ Space domain unaltered ($x \in [0, 1]$), no weights for losses
- ▶ Space domain extended ($x \in [-1.5, 3.125]$), no weights for losses
- ▶ Space domain unaltered ($x \in [0, 1]$), loss weights added (0.1 for PDE loss and 10 for IC loss)
- ▶ Space domain extended ($x \in [-1.5, 3.125]$), loss weights added (0.1 for PDE loss and 10 for IC loss)

Preliminary results

- ▶ The outcomes from the mentioned trained models are shown below.
- ▶ A comparison has been made only the exact solution at the desired time ($t = 0.2$)



Difficulties & Next steps

► Difficulties

- None of the models trained is able to completely capture the physics at the discontinuity (shock)
- Some models are presenting oscillations, while others present dissipation
- So far, training the models using cpu-only machines requires significant time (for 5000 epoch $\approx 15[\text{min}]$)

► Next steps

- Train the model using any GPU-accelerated device (HAL, local) and compare training times
- Implement a formal comparison with the exact solution and a high-order numerical method (WENO scheme)
- Define a set of neural networks parameters (width, hidden layers, optimizers, mini batches) and train the model with their combination.
- For the previous goal, a shell script will be created to set the aforesaid parameters and run the jobs automatically in a remote computer
- Implement either a 2-D compressible flow case or solve the inverse problem for a given set of density gradients (mimicking a Schlieren image)