Adaptive-Activation Residual Physics-Informed Neural Network for Cavity Flow

Alireza Samari

Introduction

This framework implements a *Physics-Informed Neural Network (PINN)* for solving partial differential equations (PDEs), specifically the incompressible Navier–Stokes equations for cavity flow. The method leverages deep learning to approximate the solution fields $\psi(x, y)$ (streamfunction) and $p(x, y)$ (pressure) across the spatial domain. By incorporating physical laws directly into the loss function, the network can learn solutions that satisfy governing equations and boundary conditions without relying solely on labeled data.

The code uses a novel neural network structure that includes:

1. Adaptive Activation Function: A trainable parameter scales the argument of a $tanh$ activation.
2. Residual Blocks: Specialized network layers that improve gradient flow and help the model train more effectively.
3. AdaptiveResNet Architecture: A neural network composed of multiple residual blocks followed by an output layer.

Governing Equations: The Incompressible Navier–Stokes System

We consider the 2D incompressible Navier-Stokes equations in the domain $\Omega \subseteq \mathbb{R}^2$ given by:

$$\nabla \cdot \mathbf{u} = 0$$
$$\rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u}$$

where $\mathbf{u}(x, y) = (u(x, y), v(x, y))$ is the velocity field, $p(x, y)$ is the pressure, ρ is the density, and μ is the dynamic viscosity. In the code, the kinematic viscosity $\nu = \mu/\rho$ is used, and ρ and ν are given constants.
The code adopts a streamfunction $\psi(x, y)$ to ensure incompressibility:

$$u = \frac{\partial \psi}{\partial y},\ v = -\frac{\partial \psi}{\partial x}$$

This automatically satisfies $\nabla \cdot \mathbf{u} = 0$. The network thus outputs:

This automatically satisfies $\nabla \cdot \mathbf{u} = 0$. The network thus outputs:

$$\mathcal{N}_\theta(x, y) = [\psi_\theta(x, y), p_\theta(x, y)]$$

Then:
$$u_\theta(x, y) = \frac{\partial \psi_\theta}{\partial y},\ v_\theta(x, y) = -\frac{\partial \psi_\theta}{\partial x}$$

PDE Residuals

We define the PDE residuals for the momentum equations as:

$$f_u = uu_x + vu_y + \frac{1}{\rho}p_x - \nu\left(u_{xx} + u_{yy}\right)$$
$$f_v = uv_x + vv_y + \frac{1}{\rho}p_y - \nu\left(v_{xx} + v_{yy}\right)$$

where subscripts denote partial derivatives with respect to x or y.

Adaptive Activation Function

The code defines an AdaptiveActivation class. A parameter α is learned to scale the input to the tanh function:

$$\text{AdaptiveActivation}\ (z) = \tanh\ (\alpha z)$$

where $\alpha \in \mathbb{R}$ is a trainable parameter. Initially, α is set to 0.9 . By allowing α to vary, the network can adjust the "sharpness" or slope of the activation function dynamically during training.

Residual Blocks

The ResidualBlock class implements a block of layers inspired by the concept of ResNet architecture.
Each residual block maps an input vector **x** to an output **y** as follows:

1. Apply a fully-connected layer and activation:

$$\mathbf{z}_1 = \text{AdaptiveActivation}\ (W_1\mathbf{x} + \mathbf{b}_1)$$

2. Apply a second fully-connected layer (without activation):

$$\mathbf{z}_2 = W_2\mathbf{z}_1 + \mathbf{b}_2$$

3. Compute a residual connection via another linear mapping:

$$\mathbf{r} = W_r\mathbf{x} + \mathbf{b}_r$$

4. Combine with a learnable parameter β that balances the residual and the transformed input:

$$\mathbf{y} = \beta\mathbf{z}_2 + (1 - \beta)\mathbf{r}$$

5. Finally, apply the adaptive activation again:

$$\mathbf{y} = \text{AdaptiveActivation}(\ \mathbf{y})$$

The parameter β is constrained between $[0.01,1.0]$ to ensure a stable balance. This structure helps with better gradient propagation during backpropagation and allows the network to learn more complex representations effectively.

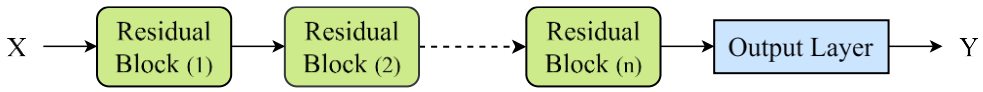
AdaptiveResBlockNet

The AdaptiveResBlockNet class stacks multiple residual blocks. Let the input dimension be $\boldsymbol{d}_{\text{in}}$, the output dimension be d_{out}, and we have a list of hidden-layer sizes (e.g.. $[20,20,\ldots,20]$). For each layer l :

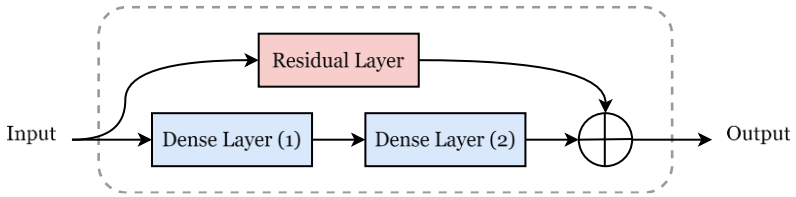
$$\mathbf{x}_{l+1} = \text{ ResidualBlock }_{l}(\mathbf{x}_{l})$$

where $\mathbf{x}_0$ is the input (x, y). After the final block, a linear output layer maps the features to the desired output (ψ, p).

A.



B.



1. A) AdaptiveResBlockNet B) A Residual Block

Loss Function

The training aims to minimize a loss function that consists of three main contributions:

1) Interior PDE Residual Loss: Ensures that the predicted fields satisfy the governing PDEs in the interior of the domain. If $\mathcal{D}$ is the set of interior training points:

$$\mathcal{L}_{\text{interior}} = \text{MSE}\left(f_u(\mathbf{x}),0\right) + \text{MSE}\left(f_v(\mathbf{x}),0\right) + \text{MSE}\left(u_\theta(\mathbf{x}), u_{\text{data}}(\mathbf{x})\right) + \text{MSE}\left(v_\theta(\mathbf{x}), v_{\text{data}}(\mathbf{x})\right)$$

Here, MSE is the mean squared error, and $u_{\text{data}}, v_{\text{data}}$ are given training data for velocity in the interior.

2) Boundary Condition Loss: Enforces that the solution matches known values (Dirichlet boundary conditions) on the boundary $\partial\mathcal{D}$:

$$\mathcal{L}_{\text{boundary}} = \text{MSE}\,(u_\theta(\mathbf{x}_b), u_b) + \text{MSE}\,(v_\theta(\mathbf{x}_b), v_b),$$

where (u_b, v_b) are known boundary velocities at boundary points $\mathbf{x}_b$.

3) Physics (Unlabeled) Points Loss: Additional "physics points" (x_p, y_p) are sampled in the domain where no direct measurement data is available, ensuring the PDE residuals are minimized throughout the domain:

$$\mathcal{L}_{\text{physics}} = \text{MSE}\,(f_u(\mathrm{x}_p), 0) + \text{MSE}\,(f_v(\mathrm{x}_p), 0).$$

The total loss is:

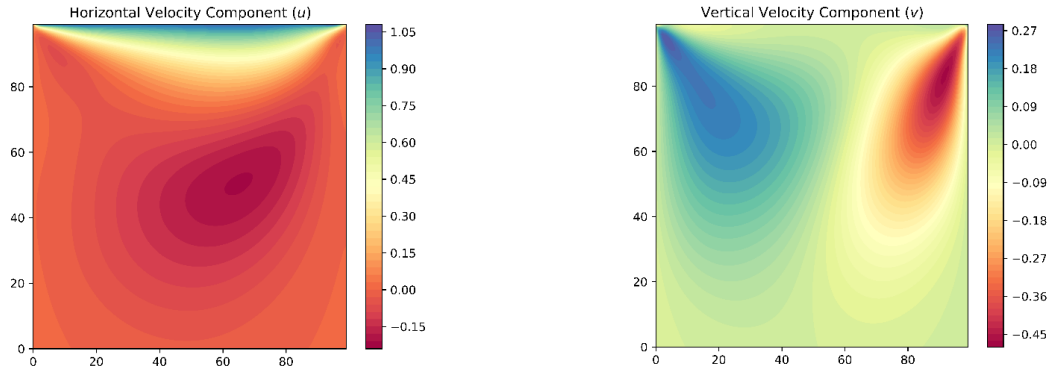
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{interior}} + \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{physics}}.$$

This combined loss enforces both data-fitting (where data is available) and physics constraints (where no direct data is provided).

Result

Results demonstrate the effectiveness of the Adaptive-Activation Residual PINN framework in solving the incompressible Navier-Stokes equations for cavity flow. The expected pressure field (p) can capture subtle gradients across the domain, especially near the upper corners, which agrees with the anticipated physical behavior. The horizontal (u) and vertical (v) velocity components clearly display separated primary and secondary flow features such as high-velocity zones located close to the moving lid, and recirculation regions along the walls. What's more, the magnitude of the velocity also shows a smooth gradient, with the largest values near the lid, decaying to zero on the fixed walls, exactly agreeing with the applied boundary conditions.

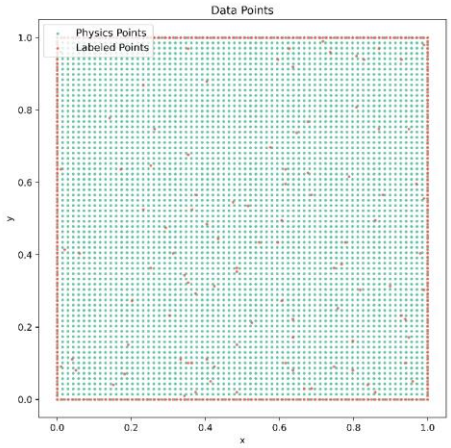
A. B.



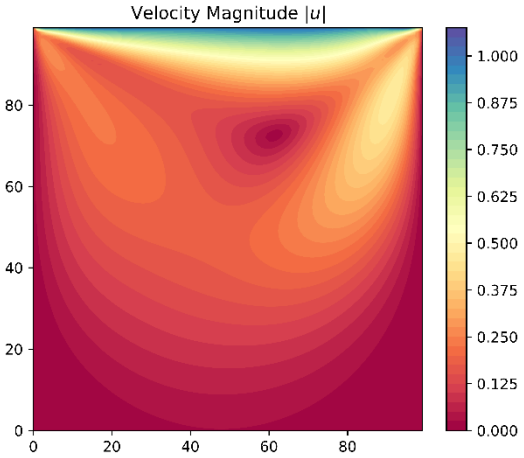
2. A) Horizontal Velocity Field B) Vertical Velocity Field.

The distribution of the training points, with a predominance of physics points in the interior and labeled points at boundaries ensures the governing equations and boundary conditions are adhered to. Combined with the new adaptive activation function and residual network architecture, this allows the model to approximate solutions to a high fidelity even when labeled data is scarce. The results validate the ability of the framework to capture complex flow dynamics while keeping physical consistency, and they presage its potential in solving PDEs in difficult domains.

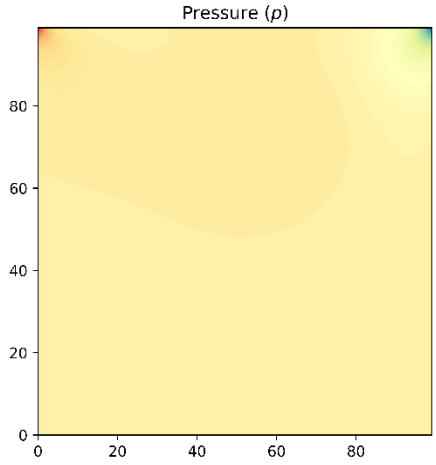
A.



B.



C.



3. A) Data Points B) Velocity Magnitude C) Pressure Field