Michael D. Walker (mw6136)
MAE557: Simulation and Modeling of Fluid Flows
January 9, 2026

Final Project Report: RT-PINNs
Physics-Informed Neural Networks for Compressible Turbulent Mixing

Disclaimer: I received assistance from Trevor Fush and Israel Bonilla. I used Google AI Studio (Gemini 3.0 Pro) for debugging.

Abstract

This project investigates the capability of Physics-Informed Neural Networks (PINNs) to simulate the two-dimensional, compressible Rayleigh-Taylor instability. A deep feed-forward neural network is trained to solve the compressible Navier-Stokes equations without labeled data, relying solely on a composite loss function derived from the governing partial differential equations (PDEs), and the associated initial (ICs) and boundary conditions (BCs). The PINN results are benchmarked against `Miranda`, a high-order finite-difference solver. While the PINN successfully captures the linear growth phase and the large-scale mean flow features, it fails to resolve the fine-scale turbulent structures characteristic of high Reynolds number mixing. This limitation is attributed to the "spectral bias" of fully connected neural networks, which preferentially learn low-frequency components of the solution. Consequently, the PINN behaves analogously to a Reynolds-Averaged Navier-Stokes model or a flow with artificially high viscosity. This report details the method, implementation, and a comparative analysis of computational cost and accuracy.

A. Introduction

Buoyancy-driven Rayleigh-Taylor Instability (RTI) occurs when a less dense fluid is continuously accelerated towards a more dense fluid, a result of the conversion of gravitational potential energy into kinetic energy. The subsequent mixing is important in a wide range of processes across engineering and the environment: stellar nebulae, Type 1a supernovae explosions, salt domes and geological flows, mushroom clouds from detonations and volcanic eruptions, temperature inversion in the atmosphere and extreme weather, combustion instabilities limiting engine performance, and capsule implosion in inertial confinement fusion. Accurately predicting the growth rate of the mixing layer and the transition to turbulence is a classical challenge in Computational Fluid Dynamics (CFD).

Traditional numerical methods for CFD, such as high-order finite-difference or finite-volume schemes, rely on discretizing the domain into a mesh. While highly accurate, these methods can be computationally expensive, particularly for high-resolution turbulent flows, and are constrained by grid quality. Physics-Informed Neural Networks (PINNs) [13, 14, 15, 21] offer a novel, mesh-free paradigm. By encoding the governing equations directly into the loss function of a neural network, PINNs can theoretically solve forward and inverse problems without the constraints of a grid by leveraging the approximation power of deep learning while enforcing physical laws.

However, resolving multi-scale phenomena like turbulence presents a significant challenge for PINNs. This project aims to assess whether a standard PINN architecture can capture the physics of compressible RTI and to verify its performance against a benchmark solver. The PINN-generated flow fields will be quantitatively and qualitatively compared against high-fidelity data from `Miranda` [3, 10, 7, 11], a high-order finite-difference implicit Large Eddy Simulation (iLES) code.

A key hypothesis of this work is that the PINNs will face challenges in accurately resolving the turbulent spectrum. Neural networks are known to exhibit "spectral bias" [12, 9], preferentially learning low-frequency features when trained with gradient descent and effectively filtering high-frequency components. Turbulence is an inherently broadband, high-frequency phenomenon, characterized by the "energy cascade," where large-scale fluid structures (like the main plumes of RTI) become unstable and break down into smaller and smaller eddies. This cascade creates a solution that is rich in features across a

wide spectrum of spatial and temporal frequencies. Therefore, it is anticipated that the PINNs solution may fail to capture the fine-scale turbulent structures characteristic of late-stage Rayleigh-Taylor mixing, potentially yielding a solution that resembles a "smoothed" RANS (Reynolds-Averaged Navier-Stokes) solution, effectively filtering the turbulent fluctuations and capturing only the mean flow behavior.

The primary objective is to critically assess the PINN's capabilities and limitations. This work will investigate whether the PINNs can at least capture the early-stage linear growth and subsequent mean flow characteristics, such as the mixing layer growth rate and mixture fraction profiles, even if it fails to resolve the turbulence. This study will provide valuable insights into the practical applicability of PINNs for complex, unstable, and turbulent flow phenomena.

B. Literature Review

The theoretical basis for RTI was established by Lord Rayleigh [17] and Taylor [19, 8]. This case is of significant scientific interest [5] as the flow transitions from a quiescent (but unstable) initial state, through a linear growth phase, and into a self-similar turbulent mixing regime. In the self-similar growth regime, the mixing width h is expected to grow quadratically as $h = \alpha \mathrm{A_t} g t^2$, where $\mathrm{A_t}$ is the Atwood number and g is gravity. Experimental and numerical studies [5, 22, 23, 24] typically place the growth constant α between 0.02 and 0.07.

The validity of high-order compact finite-difference methods for such flows is well-established [2, 11]. The `Miranda` [3, 7] code utilizes 10th-order compact schemes to minimize numerical dissipation, making it ideal for implicit Large Eddy Simulation where the grid cutoff acts as the subgrid filter.

In the realm of deep learning, Raissi et al. [13, 14, 15] introduced the PINN framework. While successful for laminar flows and some reacting flows [21], recent literature highlights the difficulty of applying PINNs to turbulence. Rahaman et al. [12] and Luo et al. [9] identified "spectral bias," showing that neural networks with smooth activations (like `tanh`) converge extremely slowly on high-frequency solution components during gradient descent. Tancik et al. [18] and Wang et al. [20] proposed Fourier Feature Mapping as a remedy, though standard PINNs often yield overly smoothed, "laminarized" solutions in turbulent regimes [21]. Other mitigation strategies include adaptive activation functions [6], adaptive sampling, curriculum learning, and dynamic learning rates. More classical methods like network hyperparameter tuning (via Bayesian optimization) and specialized network architectures may aid in capturing flow features.

C. Method Details and Analysis

Governing Equations: This analysis considers the 2-D compressible multi-phase Navier-Stokes equations. The state variables are density ρ, velocity $\mathbf{u} = (u, v)$, pressure p, and the mass fraction Y_k of species k. The conservation equations are:

$$
\begin{aligned}
\text{Mass:} \quad & \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\
\text{Species:} \quad & \frac{\partial \rho Y_k}{\partial t} + \nabla \cdot (\rho Y_k \mathbf{u} + \boldsymbol{J}_k) = 0, \quad k = 1, 2, \ldots, N_s \\
\text{Momentum:} \quad & \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + p\boldsymbol{\delta} - \boldsymbol{\tau}) = \rho \mathbf{g} \\
\text{Total Energy:} \quad & \frac{\partial E_t}{\partial t} + \nabla \cdot \left[(E_t + p)\, \mathbf{u} - \boldsymbol{\tau} \cdot \mathbf{u} + \boldsymbol{q} \right] = \rho \mathbf{g} \cdot \mathbf{u} \,,
\end{aligned}
$$

where E_t is the total energy, $\mathbf{g}$ is the gravitational acceleration, $\boldsymbol{\delta}$ is the Kronecker delta function, $\boldsymbol{\tau}$ is the viscous stress tensor defined by dynamic viscosity μ, and $\mathbf{q}$ is the heat flux defined by thermal conductivity κ. Gamma-law ($\gamma = c_p/c_v = 5/3$) gases are considered, such that a compressible equation of

state is $E_t = p/(\gamma - 1) + \rho\mathbf{u}\mathbf{u}/2$. Only inviscid, adiabatic flow with negligible species diffusion is considered ($\boldsymbol{\tau}, \boldsymbol{q}, \boldsymbol{J}_k = 0$, although small artificial viscosities and diffusivities are used for numerical stability in the benchmark comparison). A viscous form of these equations was also implemented and shown in the Appendix.

For the application of Rayleigh-Taylor instability, only two species ($N_s = 2$), a heavy fluid Y_h and a light fluid Y_ℓ, are considered. Thus the system is closed by $\sum_k Y_k = 1 = Y_h + Y_\ell$. The fluids are assumed to be fully miscible (no surface tension at the interface).

PINN Framework: We approximate the solution using a fully connected feed-forward neural network $\mathcal{N}(t, x, y; w_i)$ parameterized by network weights w_i. The network takes spatio-temporal coordinates (t, x, y) as inputs and outputs the primitive flow variables $\varphi \in (\rho, u, v, p, Y_h)$. It has been shown [21] that predicting primitives is often more stable than conserved variables. Further to enforce positivity, some outputs are transformed to latent variables: $\ln \rho$, $\ln p$, $\texttt{sigmoid}(Y_h)$.

A squared error loss function $\mathcal{L}$ is computed according to the governing equations, initial conditions, and boundary conditions at selected training (collocation) points $\varphi_{m,n}$:

$$\mathcal{L}\left(w_i\right) = \sum_{m,n=1}^{N} \left(\varphi_{m,n} - \hat{\varphi}_{m,n}\right)^2 \ ,$$

where $\hat{\varphi}$ is the known or computed value. This "physics-informed" loss function represents the residual of the PDEs and the mismatch at the boundaries, requiring no *a priori* labeled training data. Gradient descent is then used to find optimal weights (and biases). The squared error loss function is minimized when the partial derivatives with respect to

$$\frac{\partial}{\partial w_i} \sum_{m,n=1}^{N} \left(\varphi_{m,n} - \hat{\varphi}_{m,n}\right)^2 = 0 \ .$$

Backpropagation (backward pass) applies the chain rule to calculate derivatives of the loss with respect to weights (and biases). This process is iterated at a selected learning rate σ:

$$w_i^{(\psi+1)} = w_i^{(\psi)} - \sigma \frac{\partial \mathcal{L}}{\partial w_i}\Big|_{w_i^{(\psi)}} \ .$$

The PDE residual $f(\mathbf{x}, t) = \text{LHS} - \text{RHS}$ is computed using automatic differentiation `torch.autograd.grad` from the conservation laws[1], providing the exact derivatives of the network's output with respect to its inputs. Ideally, if $\mathcal{N}$ is the exact solution, $f \equiv 0$. The training process minimizes the PDE loss function, which is the squared summation of the residuals:

$$\mathcal{L}_{\text{PDE}} = \overline{f_{\text{mass}}^2} + \overline{f_{\text{species}}^2} + \overline{f_{\text{mom},x}^2} + \overline{f_{\text{mom},y}^2} + \overline{f_{\text{energy}}^2} \ .$$

[1] In explicit 2-D notation, as it appears in code, the residuals are calculated for the 5 conservation laws:

$$\begin{aligned}
\text{Mass}: \quad & \frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \ , && f_{\text{mass}} = \left[\tfrac{\partial \rho}{\partial t} + \tfrac{\partial \rho u}{\partial x} + \tfrac{\partial \rho v}{\partial y}\right] \\
\text{Species}: \quad & \frac{\partial \rho Y_h}{\partial t} + \frac{\partial \rho u Y_h}{\partial x} + \frac{\partial \rho v Y_h}{\partial y} = 0 \ , && f_{\text{species}} = \left[\tfrac{\partial \rho Y_h}{\partial t} + \tfrac{\partial \rho u Y_h}{\partial x} + \tfrac{\partial \rho v Y_h}{\partial y}\right] \\
\text{X}-\text{Momentum}: \quad & \frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u u + p)}{\partial x} + \frac{\partial \rho v u}{\partial y} = \rho g_x \ , && f_{\text{mom},x} = \left[\tfrac{\partial \rho u}{\partial t} + \tfrac{\partial (\rho u u + p)}{\partial x} + \tfrac{\partial \rho v u}{\partial y}\right] - [\rho g_x] \\
\text{Y}-\text{Momentum}: \quad & \frac{\partial \rho v}{\partial t} + \frac{\partial \rho u v}{\partial x} + \frac{\partial (\rho v v + p)}{\partial y} = 0 \ , && f_{\text{mom},y} = \left[\tfrac{\partial \rho v}{\partial t} + \tfrac{\partial \rho u v}{\partial x} + \tfrac{\partial (\rho v v + p)}{\partial y}\right] \\
\text{Energy}: \quad & \frac{\partial E_t}{\partial t} + \frac{\partial [(E_t + p) \cdot u]}{\partial x} + \frac{\partial [(E_t + p) \cdot v]}{\partial y} = \rho g_x u \ , && f_{\text{energy}} = \left[\tfrac{\partial E_t}{\partial t} + \tfrac{\partial [(E_t + p) \cdot u]}{\partial x} + \tfrac{\partial [(E_t + p) \cdot v]}{\partial y}\right] - [\rho g_x u]
\end{aligned}$$

Benchmark Method: For verification, `pyranda` [10], a Python-based Eulerian flow solver utilizing the `Miranda` numerics [3, 11, 7], is used. It employs a 10th-order compact finite-difference scheme (penta-diagonal, semi-implicit stencil) for spatial differentiation and a 5-stage, 4th-order Runge-Kutta scheme for time integration (with an adaptive timestep). It uses a structured, uniform grid and a low-pass filter for de-aliasing.

`Miranda` is designed for accurate and efficient simulations of turbulent mixing problems and is computationally efficient on massively parallel architectures. It is globally conservative with good dispersion properties and has shown exceptional ability to resolve the small scales of turbulence. However, stabilization is needed for sharp features and nonphysical numerical oscillations, such as material interfaces, since no slope limiters are used. `Miranda` employs an implicit Large Eddy Simulation framework by adding highly-localized artificial fluid properties to ensure stability. Localized Artificial Diffusivity (LAD) targets diffusivity at oscillations in mass fraction Y_k and balances the species fluxes to prevent diffusion of density. The artificial diffusion processes (denoted by *) are structured like physical diffusion (viscosity, thermal conductivity, Fickian diffusion):

$$\begin{aligned}
\text{Viscous Stress:} \quad & \boldsymbol{\tau}^\star = \mu^\star \left(2\boldsymbol{S}\right) + \left(\beta^\star - \tfrac{2}{3}\mu^\star\right)\left(\nabla\cdot\mathbf{u}\right)\boldsymbol{\delta} \\
\text{Thermal and Enthalpy Diffusion:} \quad & \boldsymbol{q}^\star = -\kappa^\star \nabla T + \textstyle\sum_{k=1}^{N_s} h_k \boldsymbol{J}_k^\star \\
\text{Species Flux:} \quad & \boldsymbol{J}_k^\star = -\rho\left(D_k \nabla\left(Y_k\right) - Y_k \textstyle\sum_i^{N_s} D_i \nabla\left(Y_i\right)\right) \ ,
\end{aligned}$$

where $\boldsymbol{S}$ is the strain rate tensor, $c_s = \sqrt{p\,\gamma/\rho}$ is the speed of sound, and the volume fraction is defined as $\rho_k V_k = \rho Y_k$. The artificial fluid properties are:

$$\begin{aligned}
\mu^\star &= C_\mu \, \overline{|\nabla^r \boldsymbol{S}|\,\Delta^{r+2}} \\
\beta^\star &= C_\beta \, \overline{\rho\,|\nabla^r(\nabla\cdot\mathbf{u})|\,\Delta^{r+2}} \\
\kappa^\star &= C_\kappa \, \overline{\rho\, c_s^3\,|\nabla^r(1/T)|\,\Delta^{r+2}} \\
D_k^\star &= \max\left[C_D\,\overline{|\nabla^r Y_k|\,c_s\Delta^{r+1}}, C_Y\,\overline{\left(|Y_k| - 1 + |1-Y_k|\right)c_s\Delta}\right] \ .
\end{aligned}$$

$r = 8$-th order derivatives are non-zero only if there are sharp features, so the artificial viscosity is highly localized. $\overline{(\cdot)}$ is the ensemble mean. The constants C are tuned to add as little diffusion as possible to avoid affecting the physics. In the mixed zones, an iterative method for pressure-temperature equilibrium is solved (4 equation model in p, T, V_k, ρ_k) between the materials.

D. Implementation

Domain and Fluid Properties: The instability is initialized in a rectangular domain $L_x = 2.8\pi, L_y = 2\pi$ with gravity $g_x = -0.01$ acting from right to left. The initial condition places a heavy fluid ($\rho_h = 3$) over a light fluid ($\rho_\ell = 1$), such that the Atwood number $\mathrm{A_t} \equiv \frac{\rho_h - \rho_\ell}{\rho_h + \rho_\ell} = 0.5$. Initial non-dimensional temperature and pressure (at the free surface) are $T_0, p_0 = 1$. For both fluids, $c_p = 1.4$, $c_v = 1$. The flow is inviscid (Reynolds number $\mathrm{Re} = \infty$).

Boundary Conditions: The boundary conditions are no-slip/adiabatic walls in x and periodic in y, so the loss function is decomposed $\mathcal{L}_{\mathrm{BC}} = \mathcal{L}_{\mathrm{periodic}} + \mathcal{L}_{\mathrm{wall}}$. The periodic boundary condition in y enforces that the flow variables $\varphi|_{y=0} = \varphi|_{y=L_y}$, so the loss function is:

$$\mathcal{L}_{\mathrm{periodic}} = \overline{(\rho|_{y=0} - \rho|_{L_y})^2} + \overline{(u|_{y=0} - u|_{L_y})^2} + \overline{(v|_{y=0} - v|_{L_y})^2} + \overline{(p|_{y=0} - p|_{L_y})^2} + \overline{(Y_h|_{y=0} - Y_h|_{L_y})^2} \ .$$

The fixed walls in x enforce a zero-slip/zero-penetration condition $\mathbf{u}|_{x=0,L_x} = 0$ and fixed species concentrations $Y_\ell|_{x=0} = 1, Y_h|_{x=L_x} = 1$:

$$\mathcal{L}_{\mathrm{wall}} = \overline{(u|_{x=0} - 0)^2} + \overline{(v|_{x=0} - 0)^2} + \overline{(u|_{x=L_x} - 0)^2} + \overline{(v|_{x=L_x} - 0)^2} + \overline{(Y_h|_{x=0} - 0)^2} + \overline{(Y_h|_{x=L_x} - 1)^2} \ .$$

Initial Conditions: The flow is initially quiescent $\mathbf{u}_0 = 0$. A diffuse interface approximation is used to spread the interface over multiple grid points with a sigmoid function as the level-set function. It is of second-order accuracy and satisfies mass conservation in the region bounded by the interface. At the interface,

$$Y_{h,0}(x,y) = \frac{1}{2}\left[1 - \tanh\left(\frac{x-\eta(y)}{\Delta}\right)\right] ,$$

where Δ is some parameter to control the thickness of smoothing. Here it is chosen in reference to the size of the domain $\Delta = 4 \cdot L_x/N_{\text{pts}}$. The free surface is perturbed along the y-axis by a sinusoidal function $\eta = A\sin(k\,y) + L_x/2$, where $k = 4$ and $A = 0.5$. $\lambda_0 \equiv 1/k$ is the wavelength of the initial perturbation, so a characteristic Rayleigh-Taylor time constant $\tau_0 = \sqrt{\lambda_0/g_x \mathrm{A_t}}$ can be obtained.

The initial condition for pressure and density may be derived analytically through the ideal gas law $p_0 = \rho_{\text{mix}} R_{\text{mix}} T_0$ and the hydrostatic equilibrium equation:

$$\frac{\partial p}{\partial x} = \rho\, g_x = \frac{p_0}{R_{\text{mix}} T_0}\, g_x \, .$$

For the heavy and light fluids, $\rho_{\text{mix}} = \rho_h Y_h + \rho_\ell Y_\ell$. Therefore, the mixture gas constant R_{mix} is defined by the harmonic mean of the species gas constants, weighted by mass fraction (Dalton's Law of Partial Pressures),

$$R_{\text{mix}} = \frac{1}{Y_{h,0}/R_h + Y_{\ell,0}/R_\ell} ,$$

where $\hat{R}$ is the universal gas constant and the species gas constants are related through their molecular weights, $R_h \equiv \hat{R}/M_h$, $R_\ell \equiv \hat{R}/M_\ell$. Substituting the smoothed level-set function for the heavy mass fraction and $Y_\ell = 1 - Y_h$:

$$\frac{1}{R_{\text{mix}}(x)} = \frac{Y_h(x)}{R_h} + \frac{Y_\ell(x)}{R_\ell} = \frac{1}{R_h}\left[\frac{1+\tanh(\frac{x-\eta(y)}{\Delta})}{2}\right] + \frac{1}{R_\ell}\left[\frac{1-\tanh(\frac{x-\eta(y)}{\Delta})}{2}\right] .$$

The hydrostatic equation is now a separable first-order linear ODE:

$$\frac{\mathrm{d}p}{p_0} = \frac{g_x\,\mathrm{d}x}{T_0} \Big/ \left[\frac{1+\tanh\left(\frac{x-\eta(y)}{\Delta}\right)}{2R_h} + \frac{1-\tanh\left(\frac{x-\eta(y)}{\Delta}\right)}{2R_\ell}\right] .$$

The closed form solution can be found via integration and written as:

$$p_0(x) = p_0 \exp(\mathcal{A}x)\cosh^{\mathcal{B}}\left(\frac{x-\eta}{\Delta}\right) \quad \text{where} \quad \mathcal{A} \equiv \frac{g_x\,(R_h+R_\ell)}{2T_0 R_h R_\ell} , \quad \mathcal{B} \equiv \frac{g_x\,\Delta\,(R_\ell - R_h)}{2T_0 R_h R_\ell},$$

and the initial location of the free surface of the fluid interface is $p(x = x_0) = p_0$. Similarly, the density profile $\rho_{\text{mix}} = \rho(x)$ is given by rearranging the ideal gas law for density $\rho_0(x) = p_0(x)/[R_{\text{mix}}(x)\,T_0]$:

$$\rho_0(x) = \frac{p_0(x)}{g_x}\left[\mathcal{A} + \frac{\mathcal{B}}{\Delta}\tanh\left(\frac{x-\eta(y)}{\Delta}\right)\right] .$$

The initial condition loss function considers the quiescent state, initial perturbation, and the analytical solutions to pressure and density:

$$\mathcal{L}_{\text{IC}} = \overline{[\rho(x) - \rho_0(x)]^2} + \overline{(u-u_0)^2} + \overline{(v-v_0)^2} + \overline{[p(x) - p_0(x)]^2} + \overline{[Y_h(x,y) - Y_{h,0}(x,y)]^2} .$$

Neural Network Architecture: The PINN was implemented in `PyTorch 2.5.1` [1]. A visualization of the network and a full backward pass are shown in the Appendix (Figures 8 and 9).

- **Architecture:** 8 hidden layers with 128 neurons each. Totaling 116,741 trainable parameters at an estimated total size of 0.48 MB.
- **Activation:** Hyperbolic tangent (tanh). This activation function is chosen for performance, because its derivative $1 - \tanh^2$ can be calculated arithmetically from the original function and can be written in terms of exponentials.

- **Inputs/Outputs:** 3 input (t, x, y) domain variables $\rightarrow$ 5 output (ρ, u, v, p, Y_h) primitive variables.
- **Positivity Enforcement:** To prevent non-physical negative densities or pressures during training, the logarithms of some values $(\ln \rho,\ \ln p)$ were predicted and an exponential transform was applied before computing the PDE loss. Similarly $\texttt{sigmoid}(Y_h)$ was used to enforce its existence on the domain $[0, 1]$.

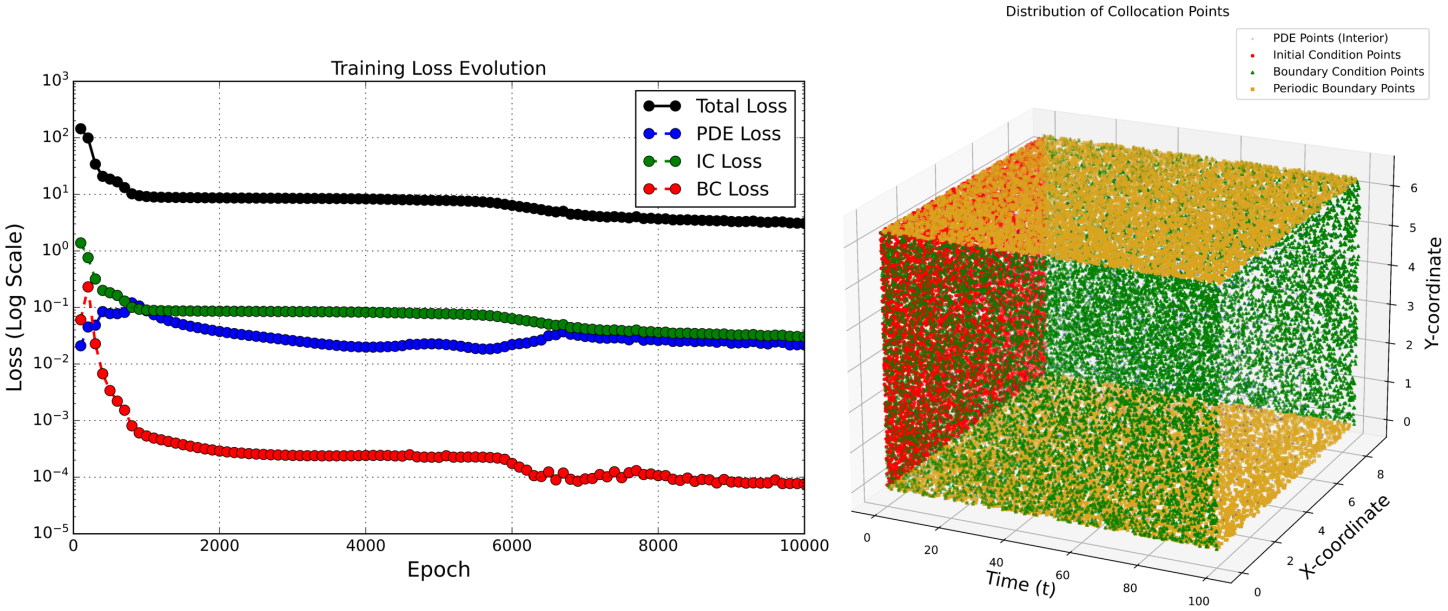

FIGURE 1. (left) Evolution of the constituent loss terms (PDE, IC, BC) and Total Loss for the first 10,000 epochs (weights: $\boldsymbol{\omega}_{\text{PDE}} = 1.0$, $\boldsymbol{\omega}_{\text{IC}}, \boldsymbol{\omega}_{\text{BC}} = 100.0$); (right) Initial generation of training (collocation) points, then resampled every 1000 epochs ($N_{\text{PDE}} = 25,000$: interior points, $N_{\text{IC}} = 10,000$, $N_{\text{BC}} = 7500$: both wall and periodic boundaries)

E. Verification

Training and Optimization: The total loss function $\mathcal{L}$ is the weighted sum:

$$\mathcal{L} = \boldsymbol{\omega}_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \boldsymbol{\omega}_{\text{IC}}\mathcal{L}_{\text{IC}} + \boldsymbol{\omega}_{\text{BC}}\mathcal{L}_{\text{BC}} \ .$$

Gradient descent was controlled with the `Adam` algorithm (Adaptive Moment Estimation) optimizer with an initial learning rate $\sigma = 1 \times 10^{-4}$. The learning rate was then adaptively adjusted for each individual parameter automatically based on the first and second moments of the gradients. A decay factor of 0.9 was applied every 2000 epochs. Initial training (collocation) points, shown in Figure 1, were generated with Latin Hypercube sampling (interior points: $N_{\text{PDE}} = 25,000$: initial condition points: $N_{\text{IC}} = 10,000$, wall and periodic boundary points: $N_{\text{BC}} = 7500$). These points were resampled every 1000 epochs to improve generalization and prevent overtraining. Hyper-parameter tuning of the network(number of hidden layers, neurons, activation functions), optimizer (algorithm, initial learning rate), and scheduler (epoch step size, decay factor) was beyond the scope of this project.

A significant challenge encountered was the optimization landscape. As shown in Figure 1, the IC and BC losses converge rapidly, but the PDE loss plateaus. This indicates the network is "lazy," satisfying the easy boundary constraints while struggling to minimize the complex PDE residuals in the domain interior. To combat this, the loss weights were manually adjusted, reducing $\boldsymbol{\omega}_{\text{IC}}$ and $\boldsymbol{\omega}_{\text{BC}}$ to encourage the optimizer to focus on the physics. For the first 10,000 epochs: $\boldsymbol{\omega}_{\text{PDE}} = 1.0$, $\boldsymbol{\omega}_{\text{IC}}, \boldsymbol{\omega}_{\text{BC}} = 100.0$. Training weights were adjusted to favor PDE loss $(\boldsymbol{\omega}_{\text{PDE}}, \boldsymbol{\omega}_{\text{IC}}, \boldsymbol{\omega}_{\text{BC}} = 10.0)$ after 10,000 epochs. Verification of the PINN was performed by monitoring the loss convergence. Figure 2 displays the full loss history. The rapid drop in BC loss confirms the network satisfies the periodicity and wall conditions. The slow decay of the PDE loss suggests the network is resolving the mean flow but struggling with smaller scales.

Computational Cost: Both the `PyTorch` and `Miranda` codes are written for CPU and GPU (`Cuda`) portability. CPU runs were completed on `CZ DANE` with Intel Sapphire Rapids (Xeon(R) Platinum 8480+) 112 cores/node architecture. GPU acceleration was achieved on `Adroit` with NVIDIA A100 GPUs (4 per node), in addition to the same Intel Sapphire Rapids CPU node.

- **PINN:** Training required approximately 100,000 epochs, as shown in Figure 2. On CPU, this corresponds to roughly 60-80 hours of wall-time. On GPU an $\approx 10\times$ speedup was achieved, with training in 6-8 hours. The cost is dominated by the automatic differentiation graph traversal for derivatives. Inference time for figure generation was negligible.
- `Miranda:` The finite-difference solver ran on the same domain with a coarse 256×256 grid. While explicit time-stepping requires many steps, the per-step cost is low. For this 2-D problem, `Miranda` is orders of magnitude faster (minutes vs. hours) than training the PINN.

The mesh-free paradigm of PINNs allows for a computational cost independent of resolution once training is complete. Inference time for figure generation will increase with resolution, but still on the order of minutes. Scaling of compressible finite-difference codes is prohibitive ($\propto N_{\text{pts}}^3$ in 2-D, inclusive of timestep stability; $\propto N_{\text{pts}}^4$ in 3-D).

F. Application

Comparison with `Miranda` (iLES): Results are shown for the trained PINN applied to the evolution of RTI for $t \in [0, 100]$ [sec]. This is sufficient time for a well developed mixing layer such that $t_{\max}/\tau_0 > 10$. The specified resolution for both the PINN and `Miranda` benchmark was $N_{\text{pts}} = 256$.

Figure 3 shows time evolution of the density field computed from `Miranda`. The fine scale turbulent structures can be seen clearly. This includes the secondary Kelvin-Helmholtz instabilities that create the "roll-up" on the sides of the mushroom cap, the secondary vortices, and the eventual chaotic mixing at the smallest scale with highly localized mixed regions at the interface. The progression from early linear growth, non-linear saturation (bubble and spike formation), and late-time mixing can be closely delineated.

A comparison of the initial conditions from the PINN and `Miranda` is shown in Figure 4 for the density field and flow parameters vertically averaged in y. The analytical solution and diffuse interface for pressure, density, and mass fraction implemented in `Miranda` is also shown. A comparison of the density field shows a close match of the interface, though there is significant dispersion in the diffuse region. This can also be seen in the mixing ratio ($\theta \in [0, 1]$, $\theta = 0$ fully segregated, $\theta = 1$ fully mixed), which is initially not sharp.

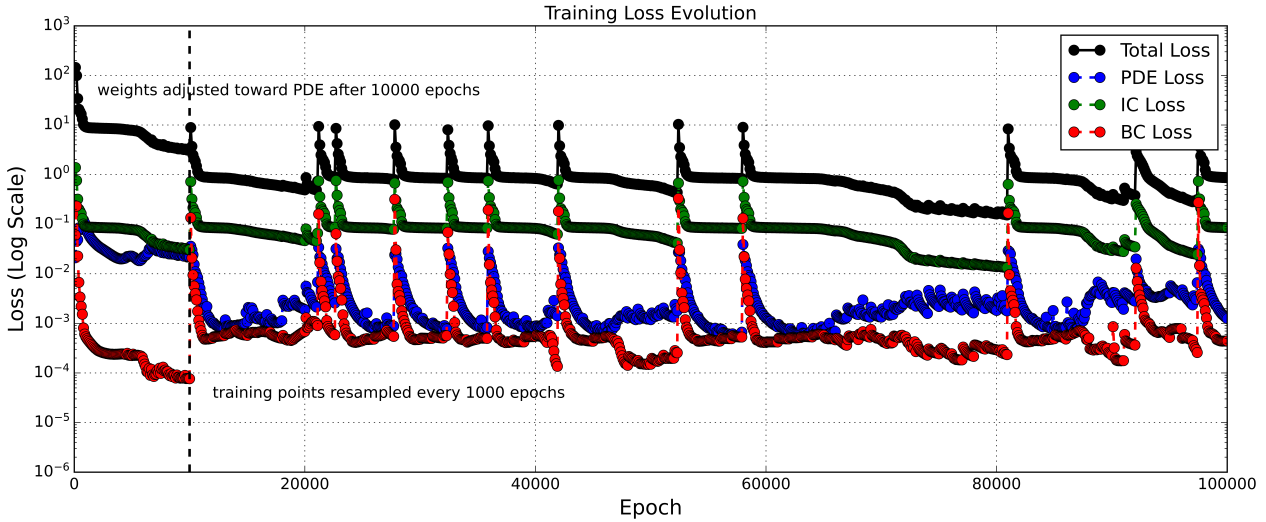

FIGURE 2. Evolution of the constituent loss terms (PDE, IC, BC) and Total Loss over 100,000 epochs. Note the rapid convergence of BCs contrasted with the slow, plateauing decay of the PDE loss, indicative of the optimization difficulty in the domain interior. Training points were resampled every 1000 epochs for generalization; training weights were adjusted to favor PDE loss ($\omega_{\text{PDE}}, \omega_{\text{IC}}, \omega_{\text{BC}} = 10.0$) after 10,000 epochs.

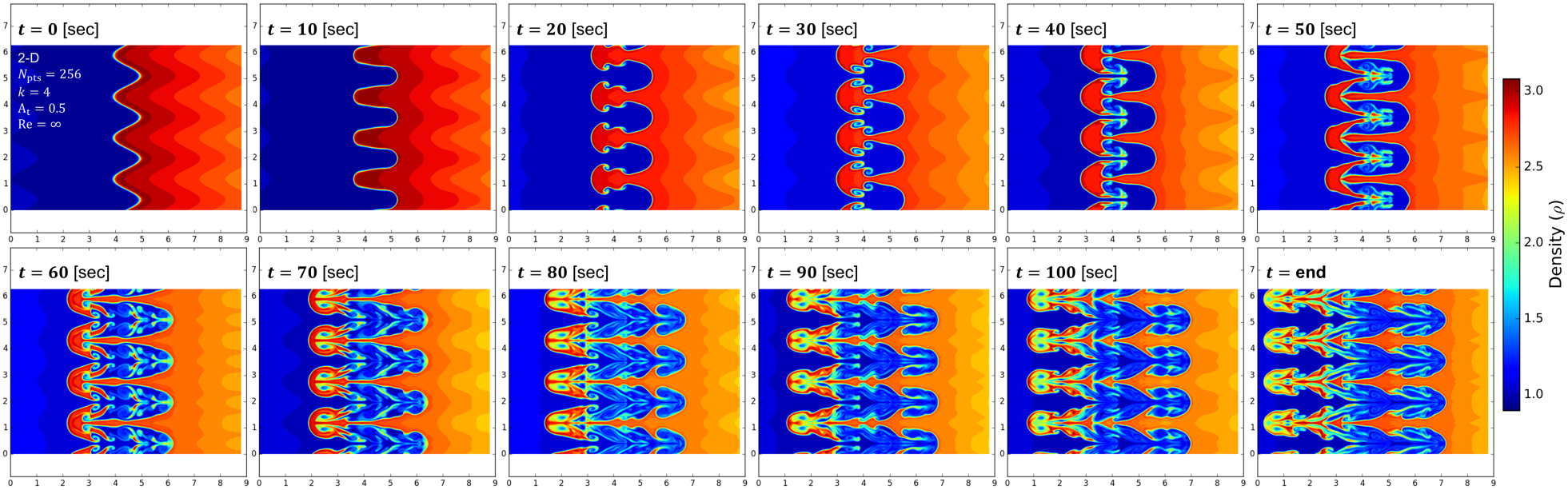

FIGURE 3. Time evolution of the density field computed from `Miranda`, showing the progression from early linear growth, non-linear saturation (bubble and spike formation), and late-time mixing.

The mixing ratio in time $\theta\,(t)$ can be expected to grow as:

$$\theta(t) = \frac{\int \overline{Y_h}\left(1-\overline{Y_h}\right)\,\mathrm{d}x}{\int \overline{Y_h(1-Y_h)}\,\mathrm{d}x} = 1 - \frac{4\int \overline{Y_h Y_\ell}\,\mathrm{d}x}{h(t)}\ .$$

$\overline{(\cdot)}$ is the ensemble mean (2-D average in height). The mixing layer width $h = C_h \int_0^{L_x} \overline{Y_h}\left(1-\overline{Y_h}\right)\mathrm{d}x$ is defined up to a multiplicative normalization factor $C_h = 4$ for a diffuse interface. Figure 5 shows the instantaneous profiles of the flow variables every $t = 10$ [sec] from `Miranda` and the final value captured by the PINN. Figure 6 shows the predicted pressure field with velocity vectors and vorticity field ($\omega = \nabla \times \mathbf{u}$) at PINN initialization and end. In summary:

- **Mean Flow:** The PINN accurately captures the position of the bubble and spike tips. The global symmetry and stratification are preserved.
- **Turbulence and Spectral Bias:** A stark difference is visible in the resolution of fine structures. The `Miranda` simulation (Figure 3) shows Kelvin-Helmholtz roll-up and secondary vortices. The PINN solution (Figure 6) appears highly diffused. The vorticity field highlights the shear layers at the bubble-spike interface. The lack of small-scale vortices indicates the spectral bias filtering high-frequency dynamics. This confirms the hypothesis of spectral bias; the neural network acts as a low-pass filter, effectively simulating a flow with a much higher effective viscosity.

Characteristics of the Bulk Flow: While the PINN struggles to capture small-scales structures, macroscopic parameters like the mixing layer thickness and growth rate are also of interest. The mixing layer width $h = 4\int \overline{Y_h}\left(1-\overline{Y_h}\right)\mathrm{d}x$ has been shown [16, 4] to theoretically grow,

$$h(t) = \alpha \mathrm{A_t} g t^2 + 2t\sqrt{\alpha \mathrm{A_t} g h_0} + h_0\ ,$$

with a growth rate $\dot{h}(t) = \sqrt{4\alpha \mathrm{A_t} g t^2}$. α [-] is a "self-similarity" parameter that settles to a constant in fully developed turbulence. The mixing width $h(t)$ is shown in Figure 7. The PINN prediction roughly follows the theoretical quadratic growth $h \approx \alpha \mathrm{A_t} g t^2$, confirming that the solver is capturing the fundamental driving force of the instability.

Discussion of Failure Modes: The "smoothness" of the PINN solution is not merely a resolution issue but a fundamental property of the training dynamics. The high-frequency components of the solution (turbulence) correspond to small eigenvalues in the Neural Tangent Kernel (NTK) [12, 9], meaning they are learned exponentially slower than low-frequency components. The standard fully-connected architecture used here inherently behaves like a Reynolds-Averaged model, capturing the mean profile while filtering fluctuations.

Several effective techniques have been developed to help PINNs learn high frequencies. *Fourier Feature Mapping* [18, 20] is a feature engineering approach that transforms the raw inputs into a higher-dimensional feature vector containing sine and cosine terms of increasing frequencies $[\sin(k\,x), \sin(2k\,x), \dots]$. This essentially converts a high-frequency problem into a low-frequency problem that the network is much better at learning. Some specialized network architectures are inherently better at representing high-frequency details. *SIRENs (Sinusoidal Representation Networks)*, which use $\sin(x)$ as their activation function, are a prime example. *Adaptive Sampling* periodically checks where the PDE error is highest and adds more training points to those "difficult" regions, rather than sampling collocation points randomly. This forces the network to pay more attention to the areas where high-frequency structures are forming. Similarly *Adaptive Activation Functions* [6] may select the network activation function in high error regions. *Curriculum Learning* trains the network in stages, starting with an "easy" version of the problem (e.g., low Reynolds number or a short time duration where the flow is still smooth). Then, it uses those trained weights as a starting point to continue training on the harder, more turbulent version of the problem. *Cyclical Learning Rates* use a scheduler (`torch.optim.lr_scheduler.CyclicLR` in `PyTorch`) that varies the learning rate up and down between a minimum and maximum value over a set number of epochs. The "uphill" part of the cycle is designed specifically to escape local minima.

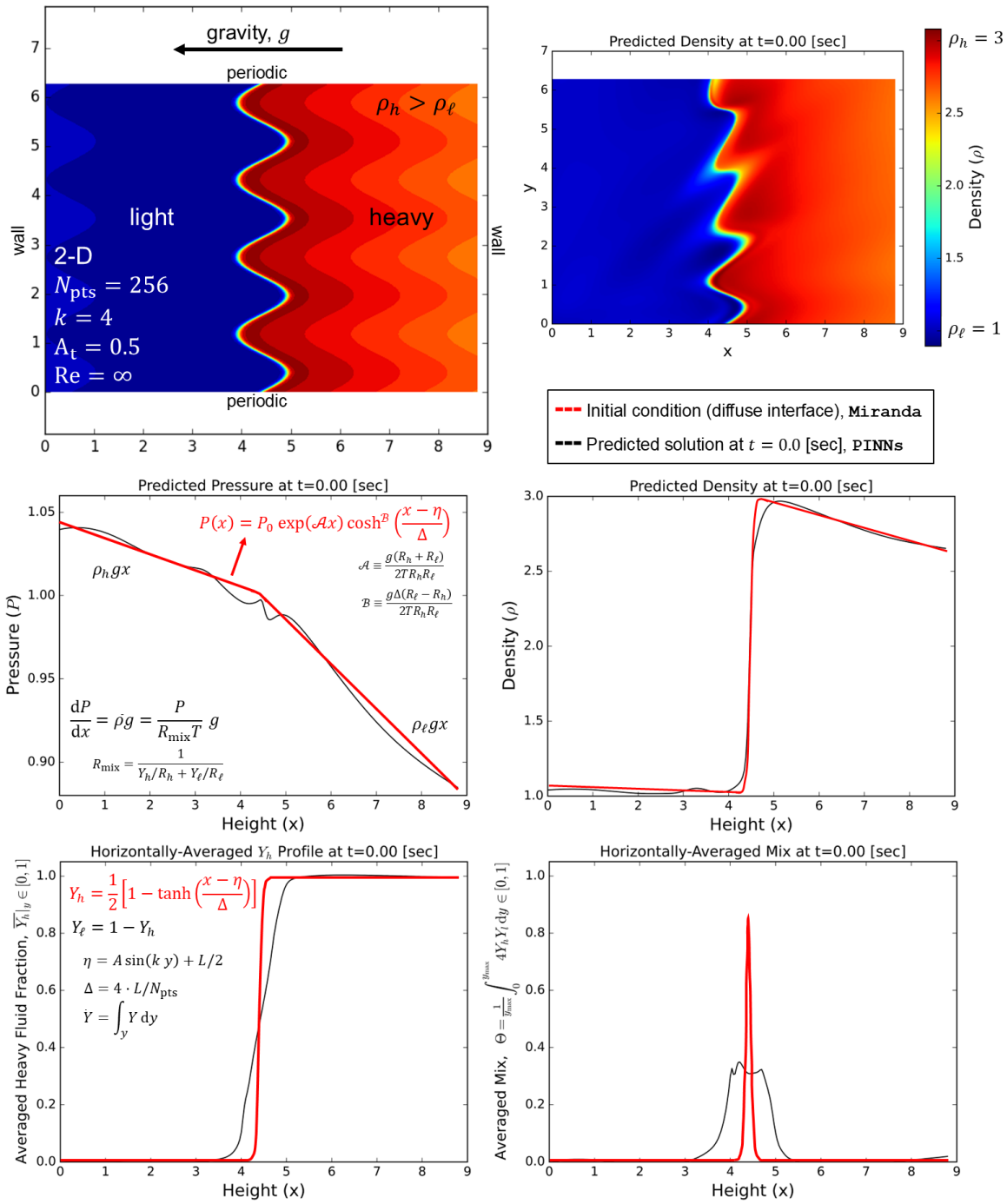

FIGURE 4. Initial conditions for the Rayleigh-Taylor instability simulation. Initial `Miranda` values (red lines) for pressure and fluid mass fraction are determined from the ideal gas law and hydrostatic equilibrium, shown in the figures as a function of height x. Predicted PINNs solution at $t = 0$ [sec] is shown with black lines. The density field shows the heavy fluid (red, left) and light fluid (blue, right) separated by a perturbed interface.

Hyper-parameter tuning (via Bayesian optimization) of the network architecture (number of hidden layers, neurons, activation functions), optimizer (algorithm, initial learning rate), and scheduler (epoch step size, decay factor) was beyond the scope of this project. If the network is too shallow or narrow, it may not have the capacity to represent the complex solution to the Navier-Stokes equations. It may be advantageous to make the network deeper (e.g., 10 layers instead of 8) or wider (e.g., 256 neurons instead of 128). While loss function weights were changed manually to favor PDE loss later in training, optimizations can be further implemented to change the loss weights dynamically.

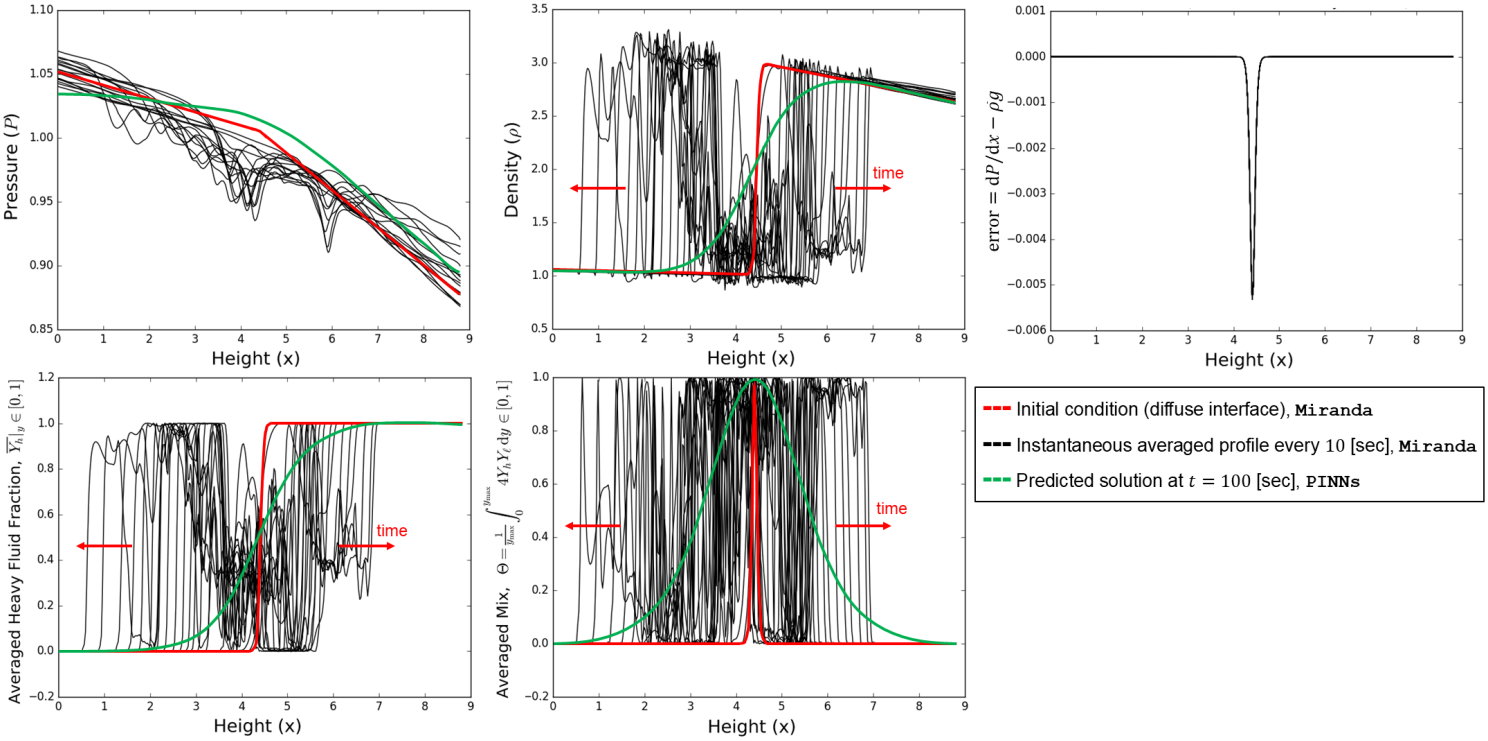

FIGURE 5. Instantaneous profiles of mixture and flow variables vertically averaged over the domain. Black lines show the broadening of the mixing layer in time. With the PINN's prediction (green) at late time, the interface remains coherent but highly diffusive, characteristic of a low-Reynolds number flow or a RANS-like approximation.

Conclusion

This project successfully developed a PINN solver for the 2-D compressible Navier-Stokes equations and applied it to the Rayleigh-Taylor instability. Comparison with the high-order `Miranda` code revealed that while PINNs can accurately enforce boundary conditions and capture macro-scale dynamics (linear growth rates, mixing width), they struggle significantly with the high-frequency spectrum required for turbulence. The PINN solution resembles a highly viscous or ensemble-averaged flow. Future work to make PINNs viable for DNS/LES of turbulence must incorporate spectral bias mitigation strategies, such as Fourier features or multi-scale network architectures.

Supplement

The core code libraries for the results presented are available at the following repository, including instructions for running and compiling (primarily in `Jupyter` notebook).

Github: https://github.com/mw6136/RT-PINNs

References

[1] R. A. Bafghi and M. Raissi. PINNs-Torch: Enhancing speed and usability of physics-informed neural networks with PyTorch. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.

[2] W. H. Cabot and A. W. Cook. Reynolds number effects on Rayleigh-Taylor instability with possible implications for type Ia supernovae. *Nature Physics*, 2(8):562–568, Aug 2006.

[3] A. W. Cook. Artificial fluid properties for large-eddy simulation of compressible turbulent mixing. *Physics of Fluids*, 19(5):055103, 05 2007.

[4] A. W. Cook, W. Cabot, and P. L. Miller. The mixing transition in Rayleigh-Taylor instability. *Journal of Fluid Mechanics*, 511:333–362, 2004.

[5] G. Dimonte, D. L. Youngs, A. Dimits, S. Weber, M. Marinak, S. Wunsch, C. Garasi, A. Robinson, M. J. Andrews, P. Ramaprabhu, A. C. Calder, B. Fryxell, J. Biello, L. Dursi, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. Timmes,

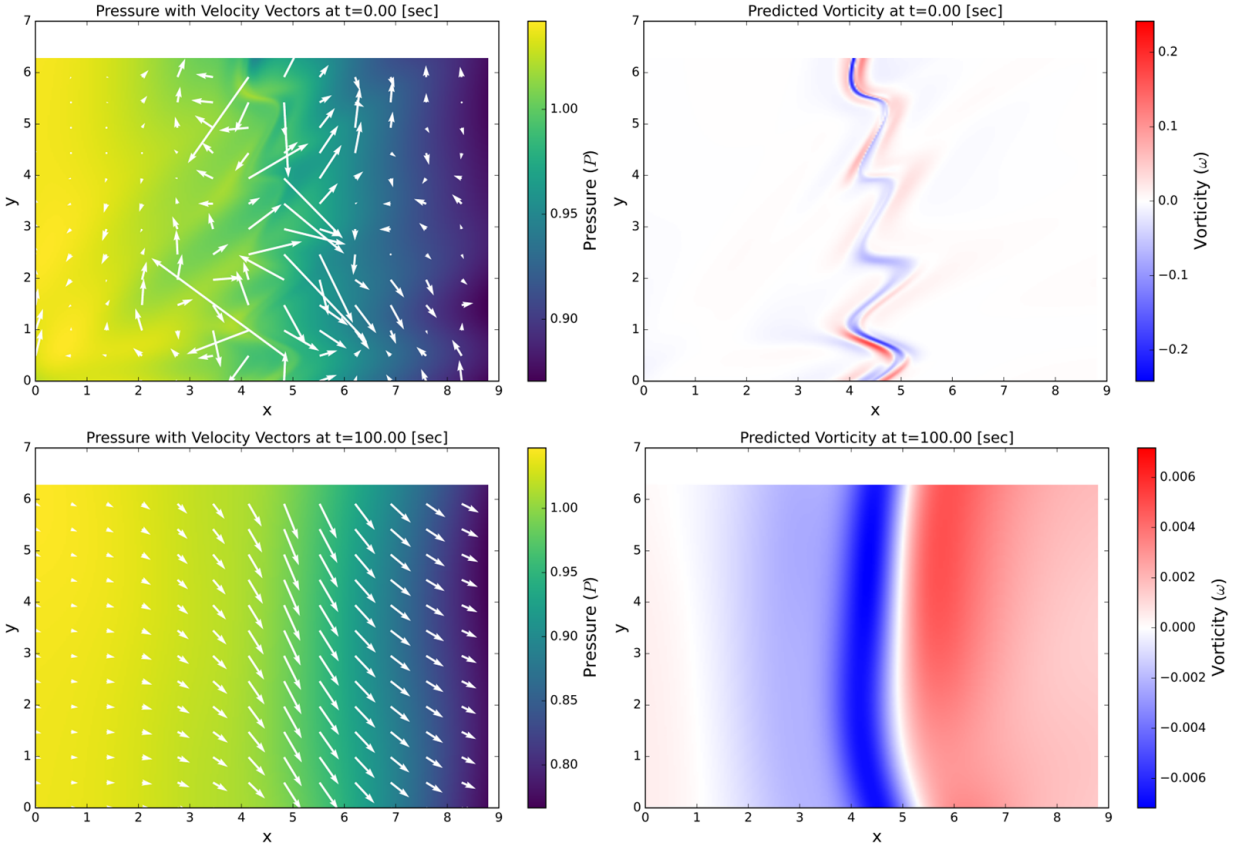

FIGURE 6. Predicted Pressure field with velocity vectors (left) and Vorticity field (right) at PINN initialization ($t = 0$ [sec]) and end ($t = 100$ [sec]). The vorticity field highlights the shear layers at the bubble-spike interface. The lack of small-scale vortices indicates the spectral bias filtering high-frequency dynamics.

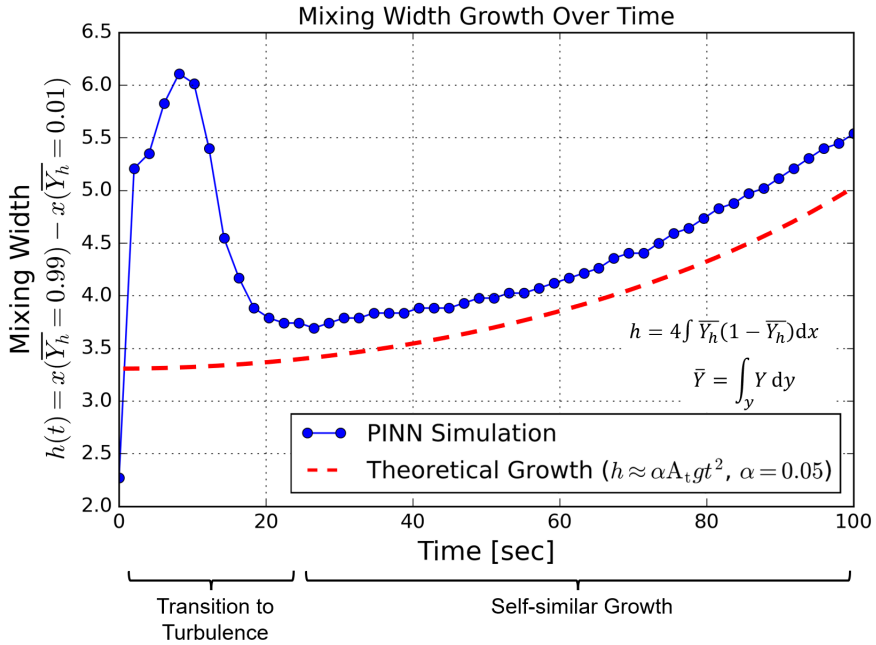

FIGURE 7. Evolution of the mixing width $h(t)$ predicted by the PINN compared to the theoretical quadratic growth $h \sim t^2$. The agreement in the growth trend validates the PINN's ability to capture the bulk mixing rate.

H. Tufo, Y.-N. Young, and M. Zingale. A comparative study of the turbulent Rayleigh-Taylor instability using high-resolution three-dimensional numerical simulations: The Alpha-Group collaboration. *Physics of Fluids*, 16(5):1668–1693, 05 2004.

[6] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.
[7] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.
[8] D. J. Lewis and G. I. Taylor. The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. II. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 202(1068):81–96, 1950.
[9] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang. Theory of the frequency principle for general deep neural networks. *CSIAM Transactions on Applied Mathematics*, 2(3):484–507, Aug 2021.
[10] B. J. Olson. PYRANDA: A Python driven, Fortran powered Finite Difference solver for arbitrary hyperbolic PDE systems and mini-app for the LLNL Miranda code, 2023.
[11] B. J. Olson and S. K. Lele. Directional artificial fluid properties for compressible large-eddy simulation. *Journal of Computational Physics*, 246:207–220, 2013.
[12] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, Jun 2019.
[13] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations, 2017.
[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations, 2017.
[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis. 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, 2019.
[16] J. R. Ristorcelli and T. T. Clark. Rayleigh-Taylor turbulence: self-similar analysis and direct numerical simulations. *Journal of Fluid Mechanics*, 507:213–253, 2004.
[17] Strutt, J. W., the 3rd Lord Rayleigh. Investigation of the Character of the Equilibrium of an Incompressible Heavy Fluid of Variable Density. *Proceedings of the London Mathematical Society*, s1-14(1):170–177, 1883.
[18] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2020.
[19] G. I. Taylor. The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 201(1065):192–196, 1950.
[20] S. Wang, H. Wang, and P. Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.
[21] V. Yadav, M. Casel, and A. Ghani. RF-PINNs: Reactive flow physics-informed neural networks for field reconstruction of laminar and turbulent flames using sparse data. *Journal of Computational Physics*, 524:113698, 2025.
[22] Y. Zhou. Rayleigh-Taylor and Richtmyer-Meshkov instability induced flow, turbulence, and mixing. I. *Physics Reports*, 720–722:1–136, 2017.
[23] Y. Zhou. Rayleigh-Taylor and Richtmyer-Meshkov instability induced flow, turbulence, and mixing. II. *Physics Reports*, 723-725:1–160, 2017.
[24] Y. Zhou, R. J. R. Williams, P. Ramaprabhu, M. Groom, B. Thornber, A. Hillier, W. Mostert, B. Rollin, S. Balachandar, P. D. Powell, A. Mahalov, and N. Attal. Rayleigh-Taylor and Richtmyer-Meshkov instabilities: A journey through scales. *Physica D: Nonlinear Phenomena*, 423:132838, 2021.

Appendix

A. Network Structure Visualization

Figure 8 shows the deep feed-forward neural network structure. Figure 9 shows a full backward pass through the neural network. Hyper-parameter tuning of the network (number of hidden layers, neurons, activation functions), optimizer (algorithm, initial learning rate), and scheduler (epoch step size, decay factor) was beyond the scope of this project.

B. Inclusion of Viscosity

A PINNs inclusive of viscosity (Re = finite) and heat flux was developed. Due to the already significant spurious, artificial viscosity present in the inviscid case, the results are not shown. A Reynolds number ($\mathrm{Re} \equiv \rho\lambda_0\sqrt{g\lambda_0}/\mu$) and Prandtl number ($\mathrm{Pr} \equiv \mu c_p/\kappa \approx 0.71$ for air-like gas) must now be specified to

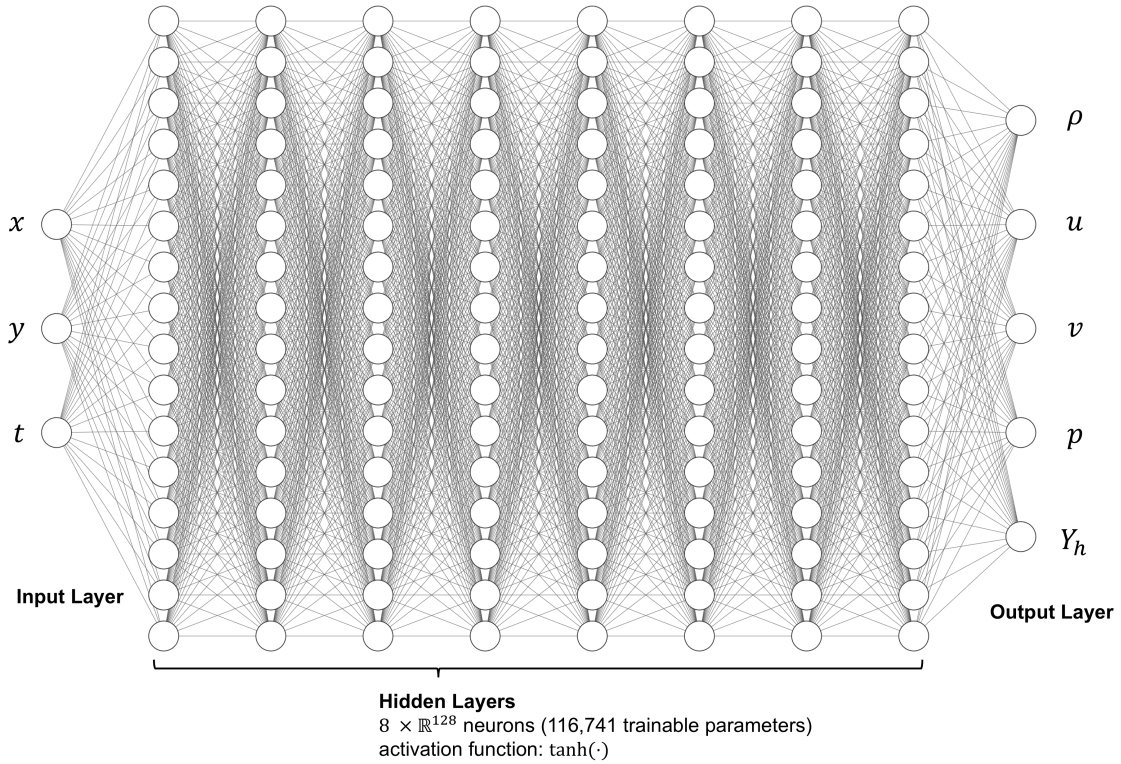

FIGURE 8. Neural Network structure, showing the hidden layers transforming inputs (t, x, y) to flow variables.

define the dynamic viscosity μ and thermal conductivity κ. There is no species diffusion in this model. First, the shear stress and heat flux terms are calculated:

$$\begin{aligned} \tau_{xx} &= \tfrac{2}{3}\mu(2\tfrac{\partial u}{\partial x} - \tfrac{\partial v}{\partial y}) \qquad q_x = -\kappa\,\partial T/\partial x \\ \tau_{yy} &= \tfrac{2}{3}\mu(2\tfrac{\partial v}{\partial y} - \tfrac{\partial u}{\partial x}) \qquad q_y = -\kappa\,\partial T/\partial y \\ \tau_{xy} &= \mu(\tfrac{\partial u}{\partial y} + \tfrac{\partial v}{\partial x}) \end{aligned}$$

Then the residuals are calculated for the 5 conservation laws:

$$\begin{aligned} \text{Mass}: \quad & \frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \\ f_{\text{mass}} &= \left[\tfrac{\partial \rho}{\partial t} + \tfrac{\partial \rho u}{\partial x} + \tfrac{\partial \rho v}{\partial y}\right] \\ \text{Species}: \quad & \frac{\partial \rho Y_h}{\partial t} + \frac{\partial \rho u Y_h}{\partial x} + \frac{\partial \rho v Y_h}{\partial y} = 0 \\ f_{\text{species}} &= \left[\tfrac{\partial \rho Y_h}{\partial t} + \tfrac{\partial \rho u Y_h}{\partial x} + \tfrac{\partial \rho v Y_h}{\partial y}\right] \\ \text{X}-\text{Momentum}: \quad & \frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u u + p)}{\partial x} + \frac{\partial \rho v u}{\partial y} = \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \rho g_x \\ f_{\text{mom},x} &= \left[\tfrac{\partial \rho u}{\partial t} + \tfrac{\partial (\rho u u + p)}{\partial x} + \tfrac{\partial \rho v u}{\partial y}\right] - \left[\tfrac{\partial \tau_{xx}}{\partial x} + \tfrac{\partial \tau_{xy}}{\partial y} + \rho g_x\right] \\ \text{Y}-\text{Momentum}: \quad & \frac{\partial \rho v}{\partial t} + \frac{\partial \rho u v}{\partial x} + \frac{\partial (\rho v v + p)}{\partial y} = \frac{\partial \tau_{yy}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} \\ f_{\text{mom},y} &= \left[\tfrac{\partial \rho v}{\partial t} + \tfrac{\partial \rho u v}{\partial x} + \tfrac{\partial (\rho v v + p)}{\partial y}\right] - \left[\tfrac{\partial \tau_{yy}}{\partial x} + \tfrac{\partial \tau_{xy}}{\partial y}\right] \\ \text{Energy}: \quad & \frac{\partial E_t}{\partial t} + \frac{\partial [(E_t + p)\cdot u]}{\partial x} + \frac{\partial [(E_t + p)\cdot v]}{\partial y} \\ & = \frac{\partial (u\tau_{xx} + v\tau_{xy} - q_x)}{\partial x} + \frac{\partial (u\tau_{xy} + v\tau_{yy} - q_y)}{\partial y} + \rho g_x u \\ f_{\text{energy}} &= \left[\tfrac{\partial E_t}{\partial t} + \tfrac{\partial [(E_t + p)\cdot u]}{\partial x} + \tfrac{\partial [(E_t + p)\cdot v]}{\partial y}\right] - \left[\tfrac{\partial (u\tau_{xx} + v\tau_{xy} - q_x)}{\partial x} + \tfrac{\partial (u\tau_{xy} + v\tau_{yy} - q_y)}{\partial y} + \rho g_x u\right] \end{aligned}$$

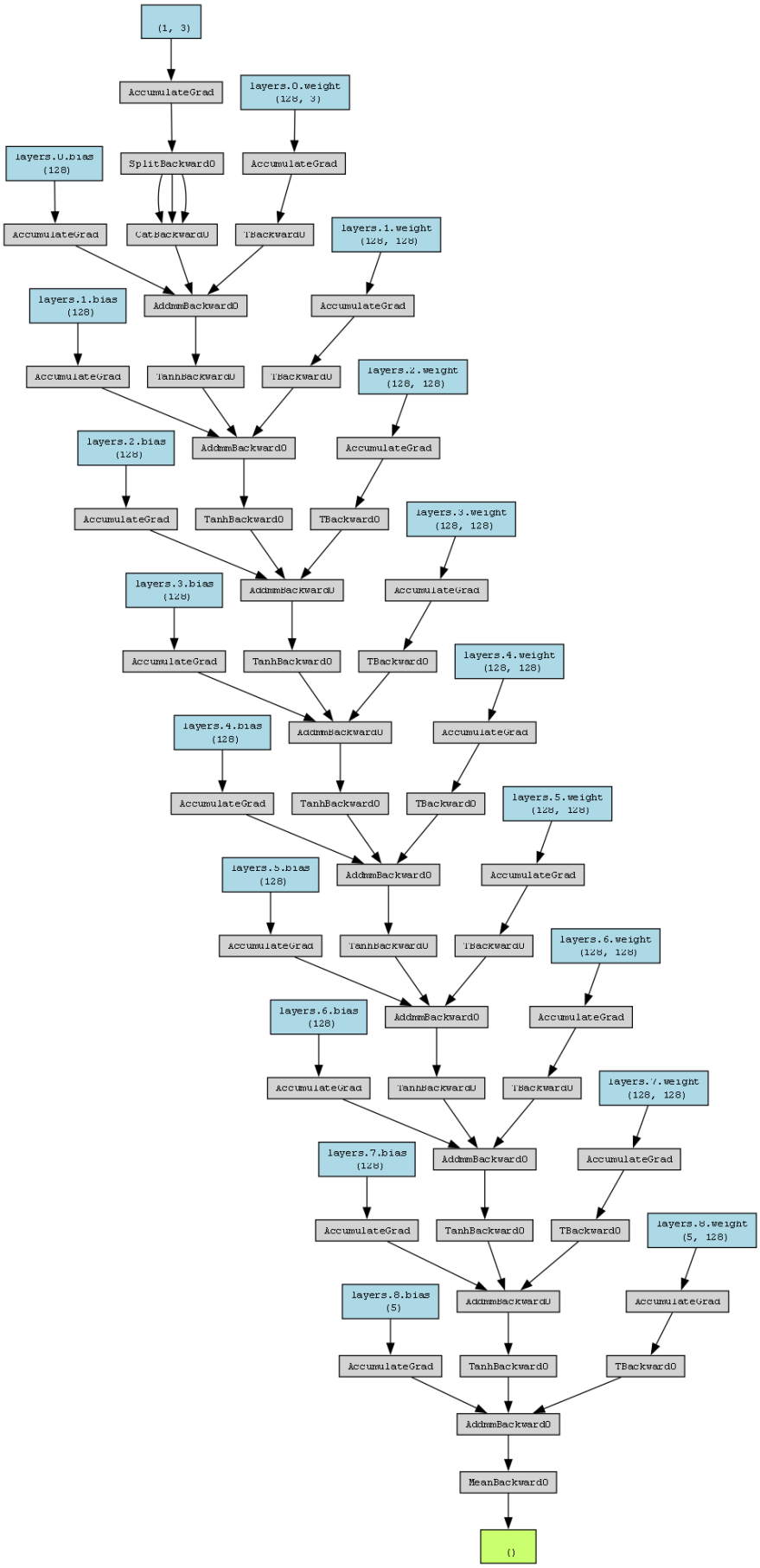

FIGURE 9. Graph visualization of the Physics-Informed Neural Network architecture, showing the fully connected layers transforming inputs (t, x, y) to flow variables.