MSBM Project Documentation

Michael K. Davis

September 11, 2025

1 PINN Architecture

We are solving the 2D MSBM equations at steady state along the x-direction, so the solution varies only along the y-direction. Furthermore, the y-direction velocity is zero at steady state. Thus, our only input is y, and our solution is u and ϕ, where u is the x-direction velocity and ϕ is the particle volume fraction.

We use two separate PINNs for u and ϕ, respectively, which are denoted in the style of [1] as

$$u(\theta_u, y) = \Gamma_u(\mathcal{M}_u(\theta_u, I_u(y))),$$

$$\phi(\theta_\phi, y) = \Gamma_\phi(\mathcal{M}_\phi(\theta_\phi, I_\phi(y))).$$

The input transformations are $I_u(y)$ and $I_\phi(y)$ for each PINN, where

$$I_u(y) = \frac{2y}{H} - 1$$

normalizes the input such that $I_u(y) \in [-1, 1]$; this range is beneficial for the choice of activation function, which are discussed later.

As for $I_\phi(y)$, each script will specify using either a Fourier expansion or a Gaussian expansion:

$$I_\phi(y) = a \cdot \left[\sin(2\pi b_i^\top |\frac{2y}{H} - 1|), \;\; \cos(2\pi b_i^\top |\frac{2y}{H} - 1|)\right]$$

where a is a scalar hyperparameter and b_i is a learnable parameter, or

$$I_\phi(y) = \alpha_i \cdot \exp\left(-\beta_i \left((\kappa_i + 0.1)|\frac{2y}{H} - 1|\right)^{(\gamma_i + 1)}\right),$$

where the input is still similarly normalized, but then passed through a modified Gaussian expansion for learnable parameters α_i, β_i, $\gamma_i, \kappa_i > 0$, $i \in [1, N_\phi]$, where N_ϕ is the number of neurons for ϕ's PINN.

The modified Gaussian expansion was chosen because its solution for y is similar in shape to the expected solution of ϕ, and it is modified by including $(\gamma_i + 1)$ (rather than a simple square) and $(\kappa_i + 0.1)$; this allows for a high-frequency solution for ϕ, but only where high-frequencies are expected. The $(\cdot + 1)$ and $(\cdot + 0.1)$ are included to prevent singularities; the chosen values 1 and 0.1 seem to be arbitrary, and the model will perform well so long as they are non-zero and positive. The modified Gaussian expansion is more problem-specific and stable than a Fourier expansion.

As for the core neural networks themselves, $\mathcal{M}_u(\theta_u, I_u(y))$ and $\mathcal{M}_\phi(\theta_\phi, I_\phi(y))$ are

$$\mathcal{M}(\theta_u, I_u(y)) = \sigma\big(W_u^{(L_u)}\,\sigma\big(W_u^{(L_u-1)} \cdots \sigma\big(W_u^{(1)} \cdot I_u(y) + b_u^{(1)}\big) \cdots + b_u^{(L_u-1)}\big) + b_u^{(L_u)}\big),$$

$$\mathcal{M}(\theta_\phi, I_\phi(y)) = \sigma\big(W_\phi^{(L_\phi)}\,\sigma\big(W_\phi^{(L_\phi-1)} \cdots \sigma\big(W_\phi^{(1)} \cdot I_\phi(y) + b_\phi^{(1)}\big) \cdots + b_\phi^{(L_\phi-1)}\big) + b_\phi^{(L_\phi)}\big).$$

where the number of layers for each PINN is denoted as L_u and L_ϕ, and the weights and biases are collectively denoted as $\theta_u = \{W_u^{(l)}, b_u^{(l)}\}_{l=1}^{L_u}$ and $\theta_\phi = \{W_\phi^{(l)}, b_\phi^{(l)}\}_{l=1}^{L_\phi}$. The weights and biases are in this case matrices with sizes that depend on the number of neurons N_u and N_ϕ, respectively. Neurons contribute to PINN width, while layers contribute to PINN depth. As for the activation function, σ, it is the same hyperbolic tangent function for both PINNs.

$$\sigma = \tanh(\cdot) = \frac{e^{(\cdot)} - e^{-(\cdot)}}{e^{(\cdot)} + e^{-(\cdot)}}.$$

Lastly, the output transformations are $\Gamma_u(\cdot)$ and $\Gamma_\phi(\cdot)$, where

$$\Gamma_u(\cdot) = (\cdot) \cdot (1 + I_\phi(y)) \cdot (1 - I_\phi(y)),$$

$$\Gamma_\phi(\cdot) = \frac{\phi_m}{1 + e^{-(\cdot)}} \cdot (1 + I_\phi(y)) \cdot (1 - I_\phi(y)),$$

which serve to enforce the zero Dirichlet boundary conditions as hard constraints for both u and ϕ, and also bound the solution for ϕ such that $\phi \in [0, \phi_m]$.

2 Loss Handling

The inverse problems involve solving for the velocity u and particle volume fraction ϕ, and in some cases the lift force exponent β, using known synthetic or experimental data. First, u is computed using data points $u_{\text{data}}(y_i)$ at specific spatial locations y_i. Subsequently, ϕ, or both ϕ and β, are determined using a combination of data and physics-based loss terms.

The loss function for each component is generalized as:

$$\mathcal{L}_j = m_j \sum_{i=1}^{M} \text{mask}_j(\lambda_{j,i}) \cdot f_j(y_i),$$

where:

- $j = 1, \ldots, L$ indexes the loss component, with $L = 1$ for scripts solving solely for u using $u_{\text{data}}(y_i)$, and $L = 5$ for scripts solving for ϕ or both ϕ and β.
- M is the number of collocation or data points y_i, where $i = 1, \ldots, M$.
- m_j is a global scalar weight for the j-th loss term, balancing its contribution to the total loss.
- $\lambda_{j,i}$ is a local, self-adaptive weight for the i-th point and j-th loss term, enabling the model to focus on regions with higher residuals.
- $\text{mask}_j(\lambda_{j,i}) = \text{softplus}(\lambda_{j,i}) = \ln(1 + e^{\lambda_{j,i}})$ ensures non-negative weights, providing numerical stability. In some implementations, $\text{mask}_j(\lambda_{j,i}) = \lambda_{j,i}^2$ is used for stronger emphasis on high-error points.
- $f_j(y_i)$ is the residual or error term for the j-th loss component, evaluated at point y_i.

The specific loss terms $f_j(y_i)$ depend on the problem and are defined as follows:

- **Data loss for** u (used when $L = 1$):

$$f_{\text{data},u}(y_i) = \left(u(\theta_u, y_i) - u_{\text{data}}(y_i)\right)^2,$$

  where $u(\theta_u, y_i)$ is the PINN prediction for velocity, and $u_{\text{data}}(y_i)$ is the known data.

- **Physics and constraint losses** (used when solving for ϕ or ϕ and β, with $L = 5$):

  1. **Particle conservation**:

  $$f_{\text{particle}}(y_i) = \left(\nabla \cdot \mathbf{J}(y_i)\right)^2,$$

  enforcing the divergence-free condition for the particle migration flux $\mathbf{J}$.

  2. **Momentum balance (xy-component)**:

  $$f_{\text{mom},xy}(y_i) = \left(\left(\nabla \cdot \mathbf{\Sigma}\right)_{xy}(y_i)\right)^2,$$

  where $\mathbf{\Sigma}$ is the total stress tensor.

  3. **Momentum balance (yy-component)**:

  $$f_{\text{mom},yy}(y_i) = \left(\left(\nabla \cdot \mathbf{\Sigma}\right)_{yy}(y_i)\right)^2.$$

4. **Mass conservation**:

$$f_{\text{mass}}(y_i) = \left(\frac{1}{M} \sum_{i=1}^{M} \phi(\theta_\phi, y_i) - \phi_{\text{avg}} \right)^2 ,$$

ensuring the average particle volume fraction matches the known ϕ_{avg}.

5. **Symmetry constraint**:

$$f_{\text{sym},\phi}(y_i) = \left(\phi(\theta_\phi, y_i) - \phi(\theta_\phi, -y_i) \right)^2 ,$$

enforcing symmetry of ϕ across the channel centerline.