Physic’s Informed Reservoir Model

Arofenitra Rarivonjy
Akmuhammet Gurbangeldiyev

Physic’s Informed Reservoir Model

Course: Technology of Science Informed Machine Learning

Term 5, 2025

Presented by

Arofenitra Rarivonjy

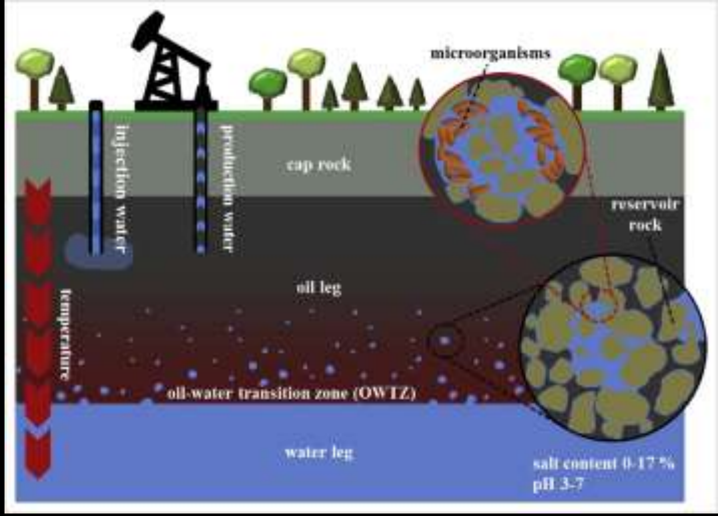
Akmuhammet Gurbangeldiyev

Arofenitra Rarivonjy

Akmuhammet Gurbangeldiyev

Meet the team

1. Introduction : Background



Reservoir simulation is critical for optimizing oil and gas production. PINN offers

- Embedding physical laws directly into the loss function
- Providing continuous solutions in space and time
- Enabling efficient forward and inverse problem solving
- Reducing computational cost for parametric studies

1. Develop a PINN framework for reservoir pressure prediction
2. Implement physics-informed loss functions incorporating PDEs and boundary conditions
3. Train and validate models for single-well and multi-well scenarios
4. Compare performance and analyze pressure field characteristics

Scope of the Research :
This study focuses on 2D horizontal reservoir flow with incompressible fluid assumptions. The governing equation is the diffusivity equation, which describes pressure propagation in porous media.

2. Methodology

We model pressure dynamics $p(x, y, t)$ in a 2D oil reservoir defined over the domain $[0, 1] \times [0, 1]$ and time $[0, T]$. The governing PDE, a time-dependent diffusivity equation, is:

$$\frac{\partial p}{\partial \mathrm{t}}(x, y, t) = \nabla \cdot (k(x, y)\nabla p) + q(x, y, t)$$

Where:

- $p(x, y, t)$: Pressure at position (x, y) and time t .
- $k(x, y)$: Permeability map, describing the ease of fluid flow through the reservoir rock.
- $q(x, y, t)$: Source term, representing oil production (negative for extraction) at well locations.
- Boundary conditions: No-flow (Neumann, $\frac{\partial p}{} = 0$ on the domain boundaries.
- Initial condition: Uniform pressure $p(x, y, 0) = p_0$ (e.g., $p_0 = 1$).

For constant permeability k:

$$\frac{\partial p}{\partial t} = k\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) + q$$

Loss Function

$$\mathcal{L}_{data} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(p_{\text{pred}}(x_i, y_i, t_i) - p_{\text{data},i}\right)^2$$

$$r = \frac{\partial p}{\partial t} - \nabla \cdot (k \nabla p) - q = \frac{\partial p}{\partial t} - k\Delta p - q \Rightarrow \mathcal{L}_{pde} = \frac{1}{N_{\text{col}}} \sum_{i=1}^{N_{\text{col}}} r(x_i, y_i, t_i)^2$$

$$t = 0 : p(x, y, 0) = 1 \Rightarrow \mathcal{L}_{ic} = \frac{1}{N_{\text{ic}}} \sum_{t=0} \left(p_{\text{pred}}(x_i, y_i, 0) - 1\right)^2$$

$$\forall x \in \{0,1\}, y \in \{0,1\} : \frac{\partial p}{\partial n} = 0$$

$$\mathcal{L}_{bc} = \frac{1}{N_{\text{bc}}} \sum_{\text{boundary points}} \left(\left.\frac{\partial p}{\partial x}\right|_{x=0,1}\right)^2 + \left(\left.\frac{\partial p}{\partial y}\right|_{y=0,1}\right)^2$$

$$\mathcal{L}_{total} = \lambda_{\text{data}} \mathcal{L}_{data} + \lambda_{\text{pde}} \mathcal{L}_{pde} + \lambda_{\text{bc}} \mathcal{L}_{bc} + \lambda_{\text{ic}} \mathcal{L}_{ic}$$

Implementation

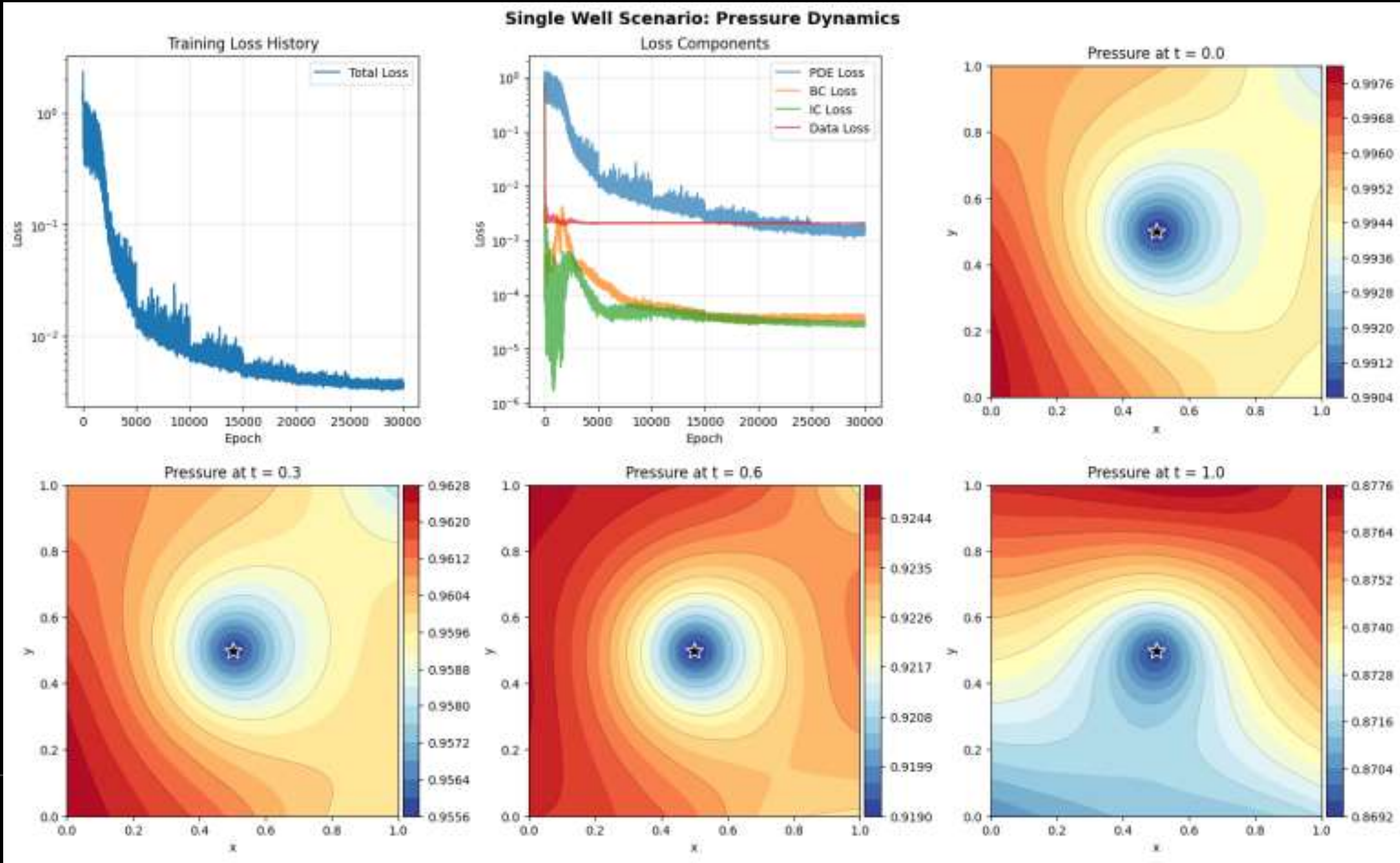
- Pytorch implementation with automatic differentiation
- Neural network : MLP [3] → [40] → [40] → [40] → [1] , with tanh activation functions
- Collocation Points: Randomly sampled points $(x, y, t) \in [0,1]^2 \times [0,1]$ (e.g., 1000 points) to evaluate the PDE residual.
- Permeability: Currently a constant $k = 10$, but designed to handle a grid-based map via interpolation (e.g., using scipy.interpolate.griddata).
- Well Source: A Gaussian source term $q(x, y, t) = -10e^{-\frac{(x-0.5)^2+(y-0.5)^2}{2\cdot 0.05^2}}$ represents a production well at (0.5, 0.5).
- Boundary Sampling: Neumann conditions are enforced by penalizing normal derivatives at boundary points.
- Data Points: Placeholder data (e.g., $p = 1$ at $(x, y, t) = (0.5,0.5,0)$ can be replaced with real measurements.

Implementation

•**Optimizer**: AdamW with initial lr 1e-3 and weight_decay of 1e-4
•**Learning rate schedule**: Step decay (γ = 0.5 every 5000 epochs)
•**Collocation points**: 1000 (single well) to 1500 (multiple wells) randomly sampled per epoch
•**Training epochs**: 15,000 for both scenarios
•**Data augmentation**: Synthetic pressure measurements (20-30 points) with noise
•Sparse pressure measurements are generated using:

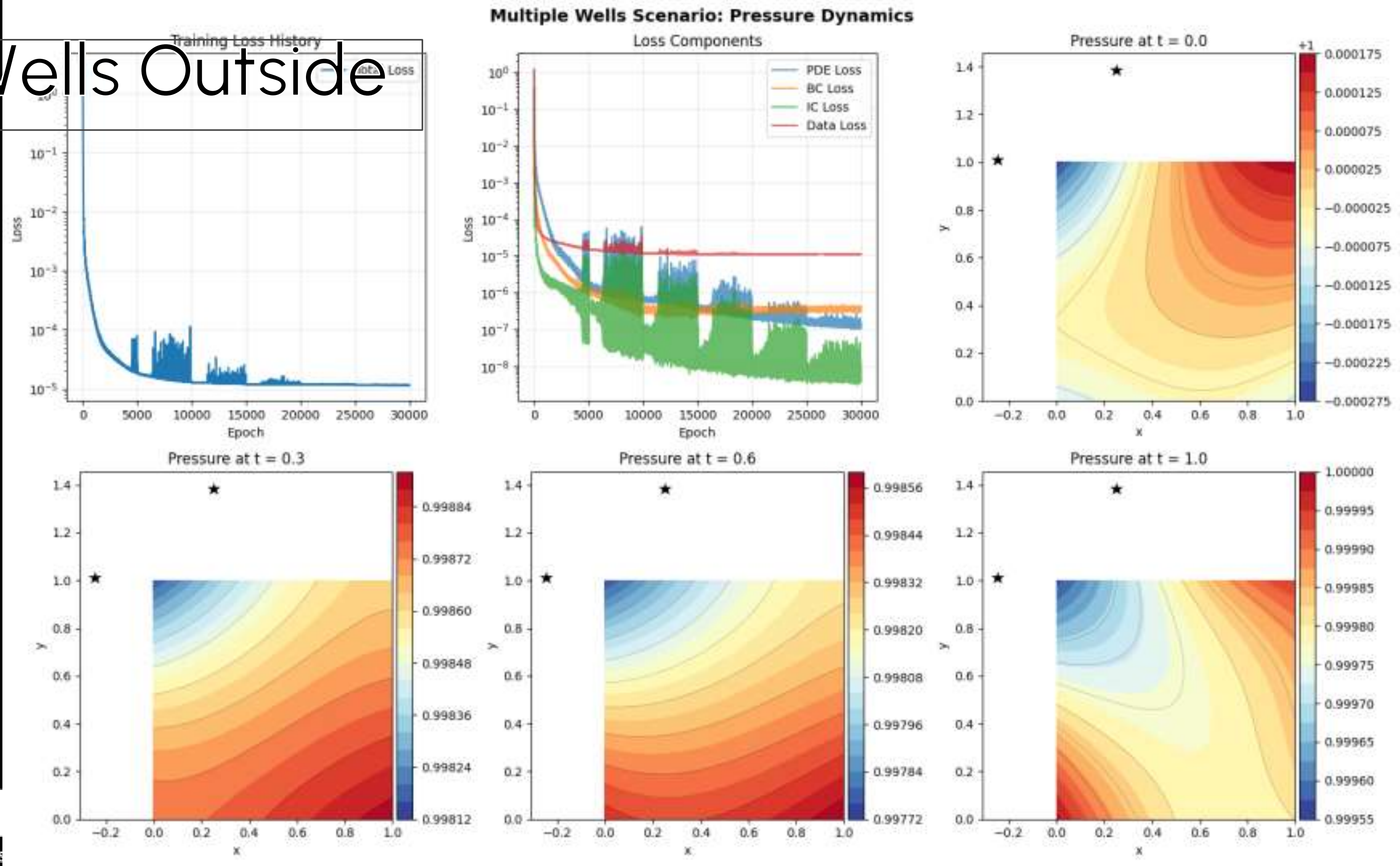
$$p_{\text{data}} = p_0 - \sum_{\text{wells}} \alpha \cdot t \cdot ex\,p\left(\frac{(x - x_w)^2 + (y - y_w)^2}{2\sigma_{\text{data}}^2}\right))$$

3. Results : Single Well Scenario

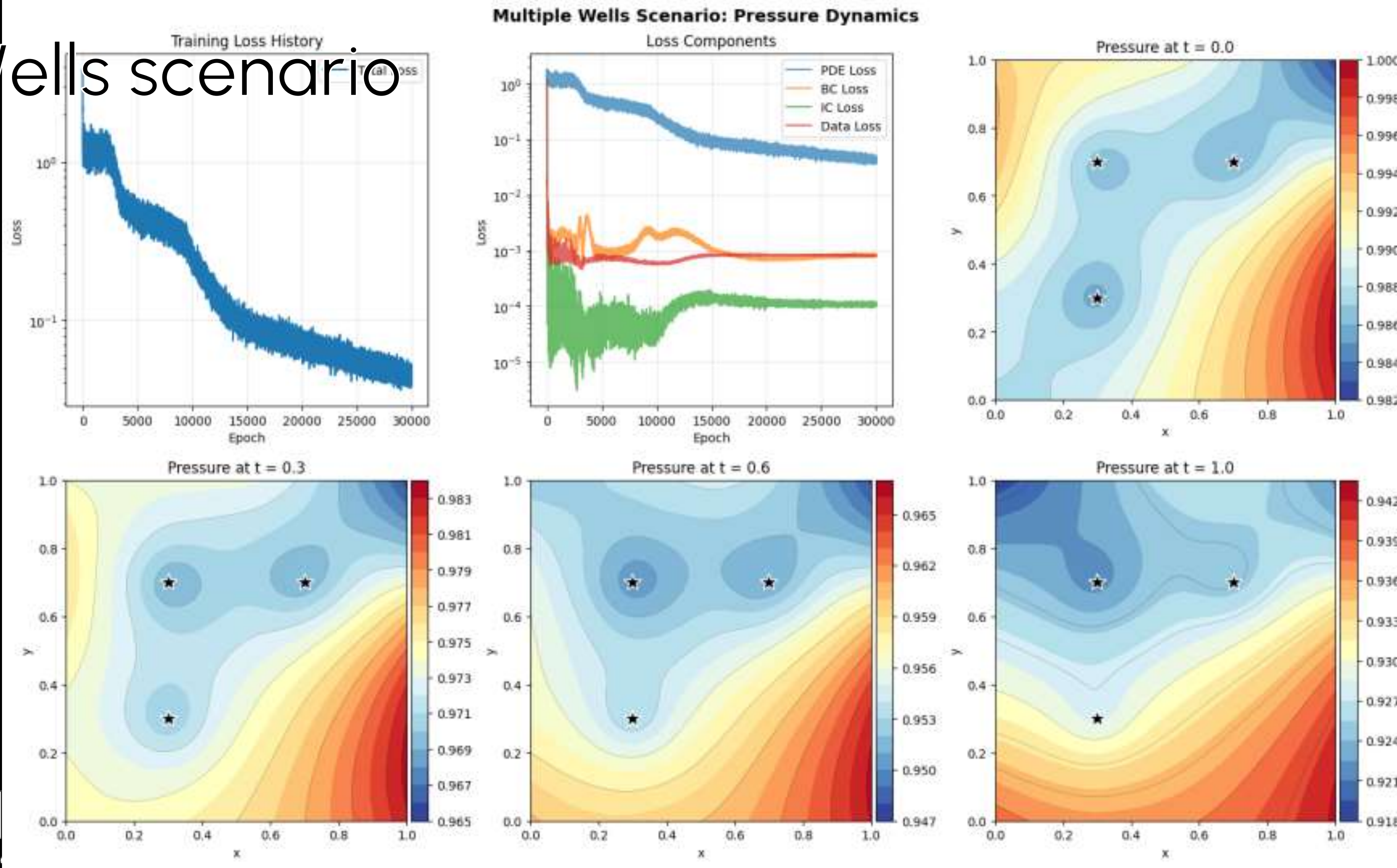


Skoltech
Skolkovo Institute of Science and Technology

2 Wells Outside

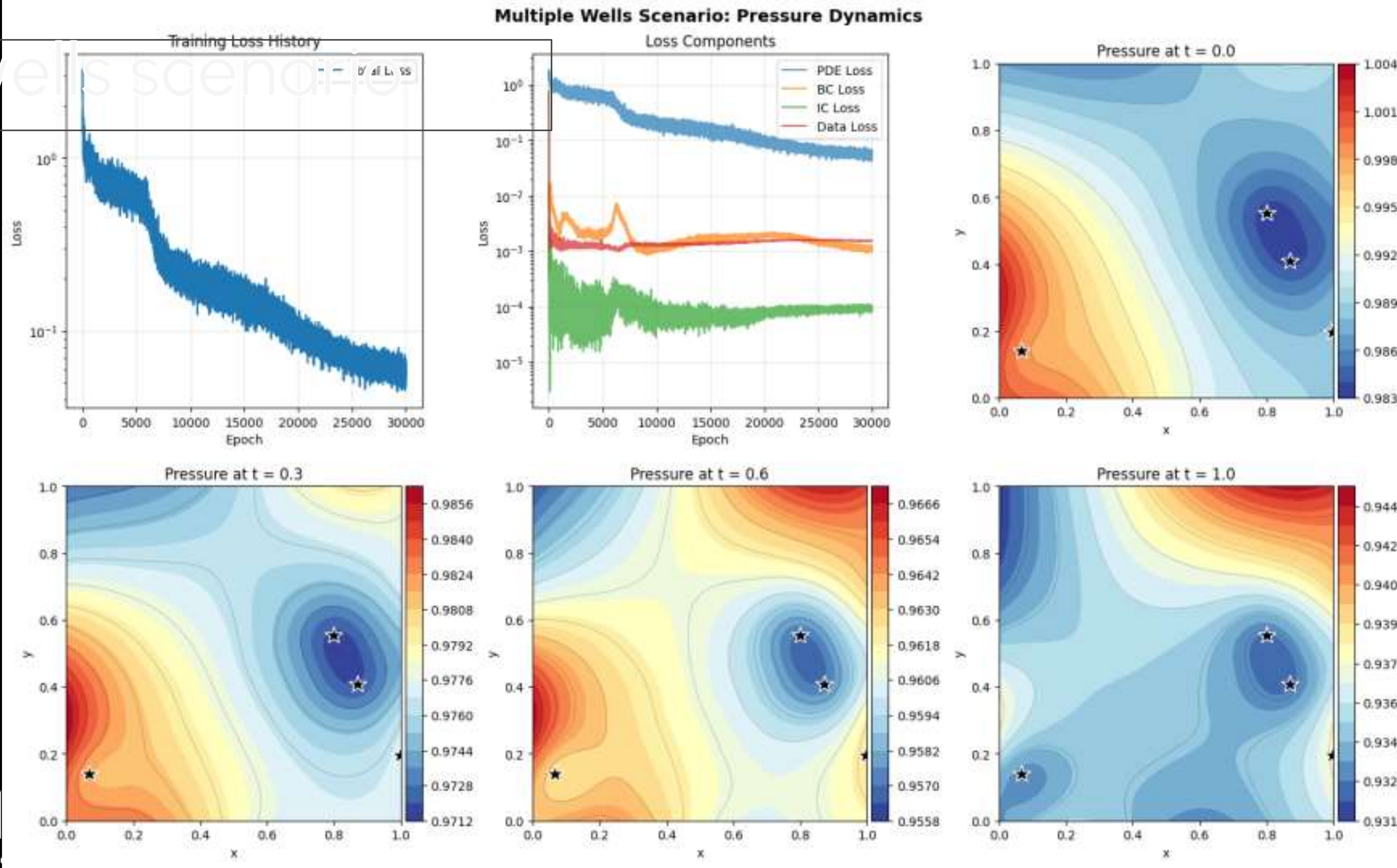


3 Wells scenario



Skolkovo Institute of S

4 wells scenario



Skol

Skolkovo Institute of S

Comparison of scenarios

Metric	1 well	2 wells	3 wells	4 wells
Total Loss	3.748e-3	1.2e-5	4.25e-2	0.055145
Training time	15 min	16 min	15 min	14 min
Min Pressure (t=1.0)	0.8692	0.999955	0.918	0.9312
Pressure Gradient	Uniform Radial	Complex	Complex	Complex
Data Points Used	20	30	30	30
Collocation Points	1000	1500	1500	1500

4. Discussion and Analysis : Strength of PINN

- Physics Integration : Embedding the diffusivity equation into the learning process
- Mesh-Free Solution (unlike finite element method)
- Handling complex well configurations
- Inverse Problem capability (Estimating k(x,y) from pressure)

4. Discussion and Analysis : Limitations of PINN

- Computational Cost
- Loss balancing (hyperparameters finetuning)
- Can stuck in local minima
- Sharp gradient near well

PINN VS Traditional Methods

Aspect	PINN	Finite Difference
Setup Time	Low (no meshing)	Medium (grid setup)
Accuracy	Good (with sufficient training)	High (stable schemes)
Computational Cost	Medium (training)	Low (single solve)
Inverse Problems	Natural	Difficult (adjoint)
Adaptivity	Easy (resampling)	Hard (regridding)
Data Integration	Seamless	Post-processing

Connection to PINO Framework

This work implements PINN (point-wise evaluation) but shares philosophy with Physics-Informed Neural Operators (PINO):

PINN (this work):

- Learns mapping: $(x, y, t) \rightarrow p$
- Requires retraining for new permeability k or well configuration

PINO (future extension):

- Learns operator: $k, q \rightarrow p$ (function → function)
- Generalizes to new permeability maps without retraining
- Combines Fourier Neural Operators (FNO) with physics loss

Extension to more complex cases

- Sine activation networks (improved spectral propriety)
- Heterogeneous permeability fields
- Mixed well configurations
- Curriculum learning (begin from 1 well to several wells)

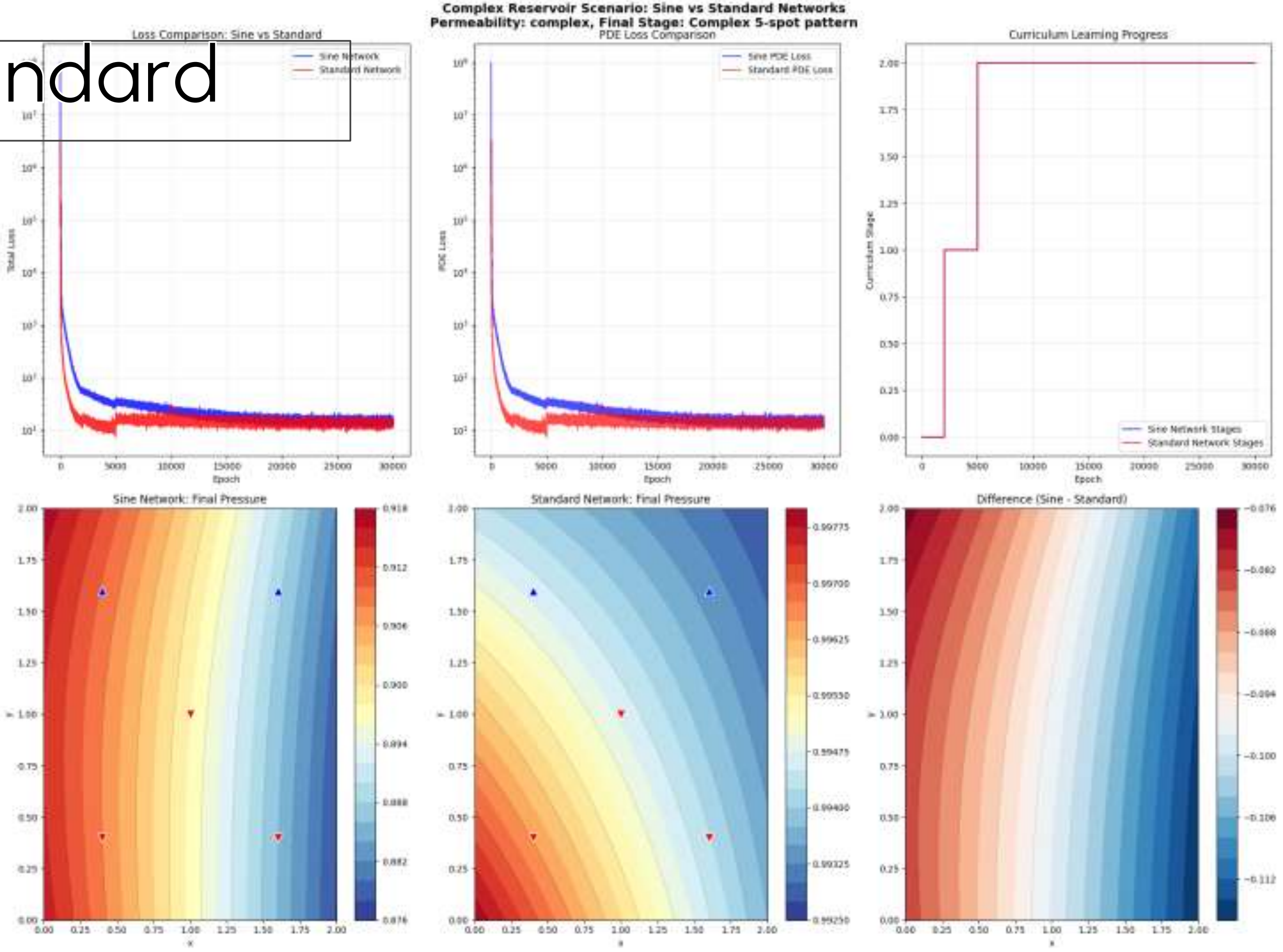
$$\frac{\partial \mathrm{p}}{\partial \mathrm{t}} = \nabla \cdot \left(\frac{k(x,y)}{\phi \mu c_t} \nabla p \right) + \frac{q(x,y,t)}{\phi}$$

where:

φ = 0.2: Porosity (constant)
μ = 1.0: Fluid viscosity
c_t = 10^{-3}: Total compressibility
k(x,y): Spatially-varying permeability
q(x,y,t): Time-dependent source term

Sine vs standard

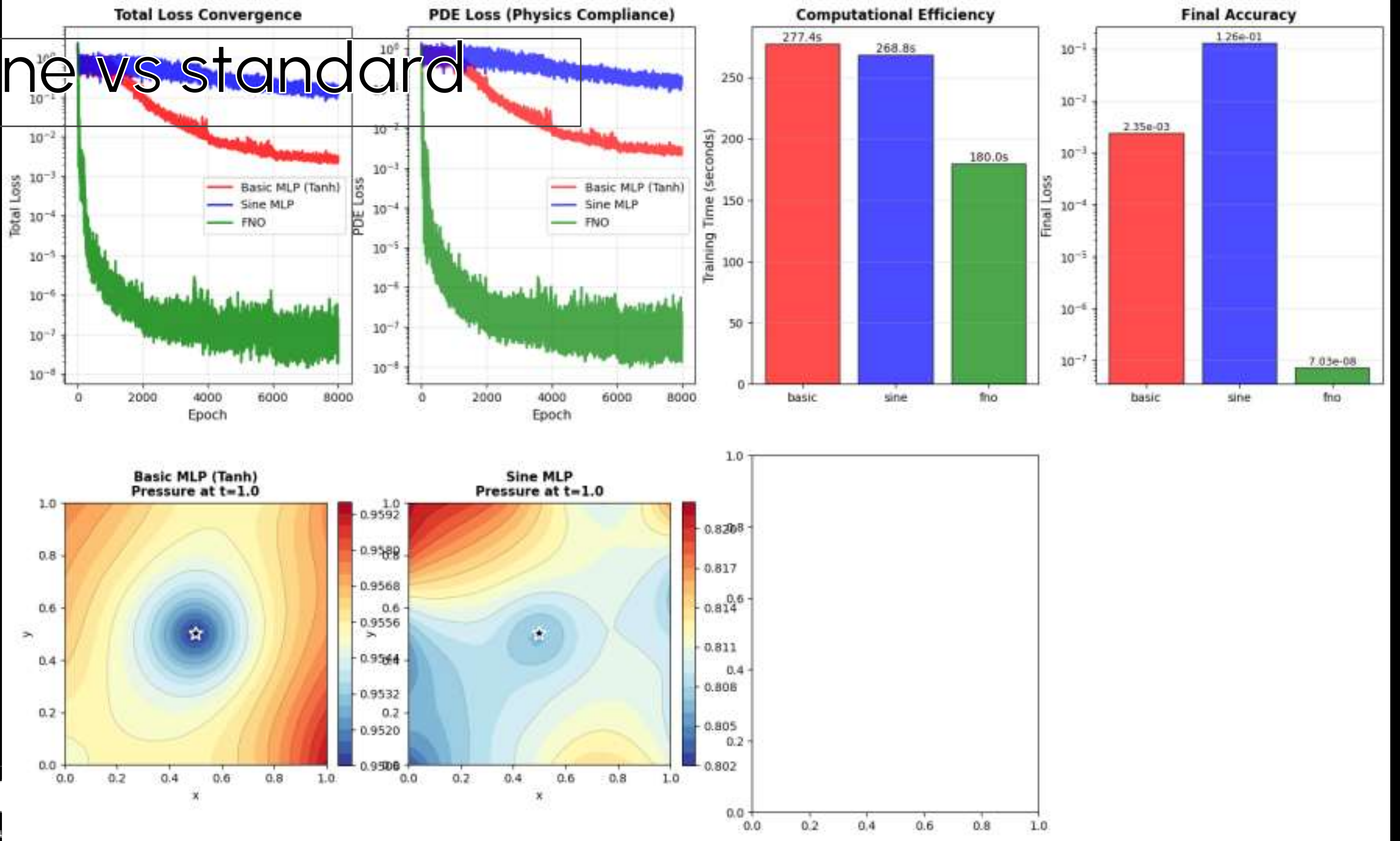
Total Loss: 12.456
PDE: 12.453490,
BC: 0.002154,
IC: 0.000729
LR: 3.78e-07,
Network: sine



Total Loss: 13.327844
PDE: 13.327836,
BC: 0.000005,
IC: 0.000002
LR: 3.78e-07,
Network: standard

Skoltech
Skolkovo Institute of Science and Technology

Sine vs standard



Skolkovo Institu

Summary

This Project presents the development and application of Physics-Informed Neural Networks (PINNs) for predicting pressure dynamics in oil reservoirs. The approach combines neural network learning with physical laws (partial differential equations) to model fluid flow in porous media. Two scenarios are investigated: single-well and multi-well configurations. Results demonstrate that PINNs can effectively capture complex pressure distributions while respecting underlying physics.

5. Conclusion

- We successfully provide mathematical formulation, PINN Implementation, Physics informed loss, single well and multiple well success and visualization.
- Physics integration enable accurate predictions, mesh-free approach, scalable from single to multiple well, converge in 30 000 epochs, interpretable.
- Practical Implications : Real time forecasting, uncertainty quantifications, optimizations, history matching, etc .

5. Future Research Directions

- Model enhancements in several cases
- Use of advanced architectures NN or NO
- Use in real case study


Thank you!

References

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv:2010.08895*.

- Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., & Anandkumar, A. (2023). Physics-Informed Neural Operator for Learning Partial Differential Equations. *arXiv:2111.03794*.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (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.
- Aziz, K., & Settari, A. (1979). *Petroleum Reservoir Simulation*. Applied Science Publishers.
- Wang, S., Teng, Y., & Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5), A3055-A3081.