Towards Fast GNN Surrogates for CO2 Migration in Complex Geological Formations

R. S. Luna[1], T. H. Coelho[1], R. M. Velho[1], A. M. Cortes[1], R. N. Elias[1], A. G. Evsukoff[1], F. A. Rochinha[1], M. Araya-Polo[2], H. Gross[2], A. Coutinho[1]

[1]High Performance Computing Center, COPPE/Federal University of Rio de Janeiro, Rio de Janeiro, Brazil.

[2]TotalEnergies EP Research and Technology, US.

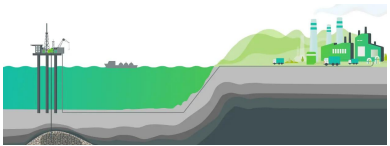
OUTLINE

1. Motivation
2. The SPE11A Benchmark
3. The GNN Engine
4. Results
5. Conclusions and Discussion

Motivation

Carbon Capture and Storage

- Carbon Capture and Storage (CCS) in faulted reservoirs is a key strategy to reduce CO_2 emissions, but fault heterogeneity increases uncertainty in plume migration and trapping;



- Accurate CO_2 plume prediction is crucial for storage integrity, leakage assessment, and injection optimization;
- High-fidelity multiphase flow simulations capture these processes but are too expensive for real-time digital twin applications;
- Surrogate models based on scientific machine learning (SciML) enable fast and differentiable approximations for uncertainty quantification and control.

Objectives

- **Main Objective:** Develop high-fidelity surrogate models using Scientific Machine Learning (SciML) to accelerate multiphase flow simulations in CO_2 storage reservoirs.
- **Research Focus:** Advance Graph Neural Networks (GNNs)[1] for 3D spatio-temporal modeling of complex flow and transport phenomena in faulted reservoirs.
- **Long-term Goal:** Integrate and validate surrogate models using open-source simulators enhancing their predictive and real-time capabilities.

[1] Ju, Xin, et al. "Learning CO2 plume migration in faulted reservoirs with Graph Neural Networks." Computers & Geosciences 193 (2024): 105711.

GEOS Multiphysics Simulator

The goal of GEOS is to open up new horizons in modeling carbon storage and other subsurface energy systems including taking advantage of the ongoing revolution in high-performance computing hardware, see [1], [2]

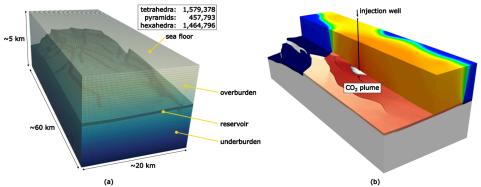


Figure 1: Real world CO_2 storage site: (a) discrete mesh, transparency is used for the overburden region to reveal the complex faulted structure of the storage reservoir; (b) results of a compositional flow simulation after 25 years of CO_2 injection. The CO_2 plume is shown in white near the bottom of the well. Colors in the reservoir layer indicate changes in fluid pressure, and the colors in the overburden indicate vertical displacement resulting from the injection. Note that color scales have been removed intentionally.

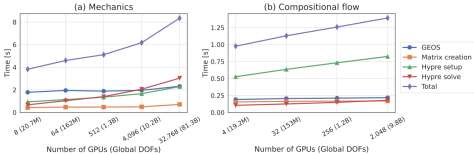
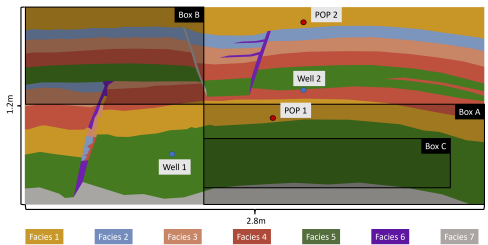


Figure 2: Weak scaling results on ORNL/Frontier: average execution time per newton iteration vs number of GPUs for a mechanics (a) and a compositional flow (b) simulation, respectively.

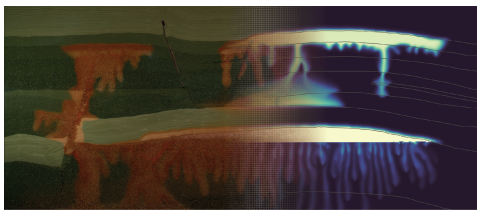
[1] Settgast et al. "GEOS: A performance portable multi-physics simulation framework for subsurface applications." Journal of Open Source Software 9.LLNL–JRNL-864747, 2024.
[2] Trebotich et al. "A multiphysics coupling framework for exascale simulation of fracture evolution in subsurface energy applications." Front. High Perform. Comput. 2:1416727, 2024.

The SPE11A Benchmark

Based on laboratory scale problem described in Nordbotten et al. "The 11th Society of Petroleum Engineers Comparative Solution Project: Problem Definition." SPE J. 29 (2024): 25072524.



Problem description showing faults and facies.

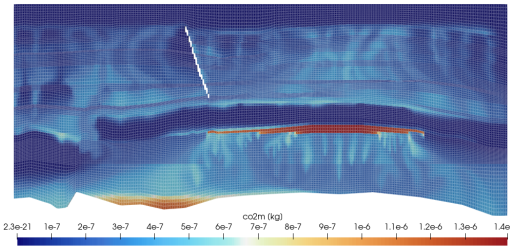


(left half) Experiment x (right half) simulation.

Model: Isothermal two-phase, two-component flow

Generating Training Data for the Surrogate

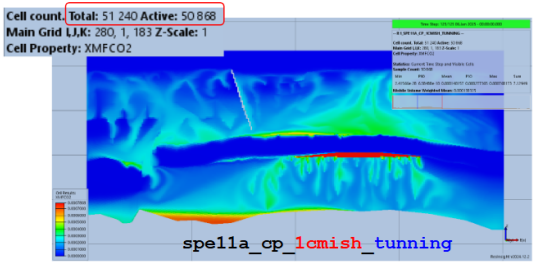
- GEOS[1] and OPMFlow[2] can be used to produce training data for our surrogate model;
- For OPMFlow, input files are prepared using pyopmspe11[3] tool.

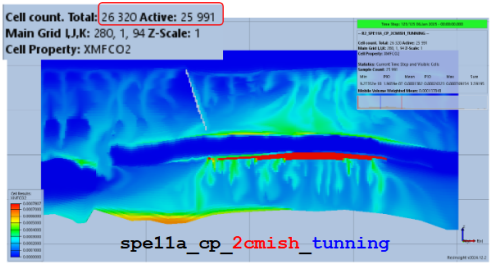


SPE11A corner-point grid with “2cmish” size.

[1] Settgast et al. "GEOS: A performance portable multi-physics simulation framework for subsurface applications." Journal of Open Source Software 9.LLNL–JRNL-864747, 2024.
[2] Rasmussen, et al. "The open porous media flow reservoir simulator." Computers & mathematics with applications 81: 159-185, 2021.
[3] Landa-Marbán, D., & Sandve, T. H. "pyopmspe11: A Python framework using OPM Flow for the SPE11 benchmark project." Journal of Open Source Software, 10(105), 7357, 2025

SPE11A Grids: Grids with 1cm and 2cm resolution






```
================      End of simulation      ================

Number of MPI processes:          33
Threads per MPI process:           2
Setup time:                     1.45 s
  Deck input:                   0.29 s
Number of timesteps:           47041
Simulation time:             8153.97 s
  Assembly time:             1939.76 s (Wasted: 422.5 s; 21.8%)
    Well assembly:              0.00 s (Wasted: 0.0 s; 0.0%)
  Linear solve time:         2842.80 s (Wasted: 1004.9 s; 35.4%)
    Linear setup:             596.85 s (Wasted: 194.2 s; 32.5%)
  Props/update time:         1877.97 s (Wasted: 398.6 s; 21.2%)
  Pre/post step:             1383.50 s (Wasted: 39.7 s; 2.9%)
  Output write time:           66.58 s
Overall Linearizations:      110802      (Wasted: 24601; 22.2%)
Overall Newton Iterations:    75502      (Wasted: 24564; 32.5%)
Overall Linear Iterations:   819562      (Wasted: 302451; 36.9%)
```

135 min

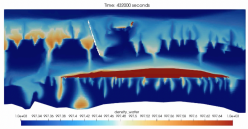
Storage:
1cmish: 413MB/simulation
2cmish: 214MB/simulation

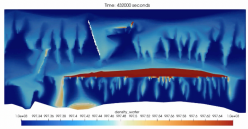
```
================      End of simulation      ================

Number of MPI processes:          33
Threads per MPI process:           2
Setup time:                     1.18 s
  Deck input:                   0.24 s
Number of timesteps:           24469
Simulation time:             6669.99 s
  Assembly time:             1003.71 s (Wasted: 242.2 s; 24.1%)
    Well assembly:              0.00 s (Wasted: 0.0 s; 0.0%)
  Linear solve time:         2773.86 s (Wasted: 957.6 s; 34.5%)
    Linear setup:             378.47 s (Wasted: 133.9 s; 35.4%)
  Props/update time:         1335.29 s (Wasted: 322.3 s; 24.1%)
  Pre/post step:             1380.68 s (Wasted: 54.5 s; 3.9%)
  Output write time:          114.70 s
Overall Linearizations:       61221      (Wasted: 14673; 24.0%)
Overall Newton Iterations:    42824      (Wasted: 14656; 34.2%)
Overall Linear Iterations:   453767      (Wasted: 169754; 37.4%)
```

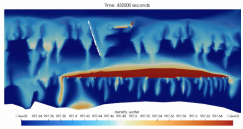
111 min

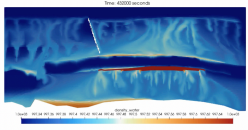
SPE11A Trajectories Generation

- Ten trajectories generated based on randomly varying the facies' permeability and porosity values given in Flemisch et al.[1].
- System states are sampled every $\Delta t = 90\,\mathrm{s}$, yielding trajectories of $N = 4800$ time steps – a total simulated time of five days (120 hours).









[1] Flemisch et al., "The FluidFlower Validation Benchmark Study for the Storage of CO2", Transport in Porous Media, 2024.

MeshGraphNet: Isotropic vs Anisotropic (direction-aware)

Isotropic (standard MeshGraphNet)

No directional gating: neighbors contribute uniformly through aggregation.

$$m_{ij}^{t,l} = \phi_e^l\Big([z_{ij}^{t,l-1}, z_i^{t,l-1}, z_j^{t,l-1}]\Big)$$

$$z_i^{t,l} = \phi_v^l\Big([z_i^{t,l-1}, \underbrace{\sum_{j\in\mathcal{N}(i)} m_{ij}^{t,l}}_{\textit{isotropic}}]\Big)$$

Anisotropic (ours, direction-aware, inspired in Thurlemann and Rineker [2])

Directional edge features $\mathbf{z}_{ij}$ modulates interactions via α_{ij}.

$$z_{ij}^{t,l} = \phi_e^l\Big([z_{ij}^{t,l-1}, z_i^{t,l-1}, z_j^{t,l-1}]\Big)$$

$$\alpha_{ij}^{t,l} = \mathrm{softmax}_{k\in\mathcal{N}(i)}\Big(\phi_\alpha^l([z_{ij}^{t,l-1}, z_i^{t,l-1}, z_j^{t,l-1})\Big)$$

$$z_i^{t,l} = \phi_v^l\Big([z_i^{t,l-1}, \underbrace{\sum_{j\in\mathcal{N}(i)} \alpha_{ij}{}^{t,l} z_{ij}^{t,l}}_{\textit{anisotropic}}]\Big)$$

[2] Thurlemann and Riniker, "Anisotropic message passing: Graph neural networks with directional and long-range interactions." 11th ICLR, 2023.

Dynamic State Incremental Update

The dynamic state advances via an explicit Euler update, as proposed in [3].

$$X_{\text{dyn}}^{t+1} = X_{\text{dyn}}^{t} + \Delta t \; \phi_{\text{dec}}(\mathbf{H}^t), \qquad \mathbf{H}^t = \text{GraphConvLSTM}(\mathbf{Z}^t).$$

The model is trained using autoregressive multi-step supervision over horizons of length T. The loss is:

$$\mathcal{L}(\theta) = \frac{1}{T - T_{train}} \sum_{t=1}^{T-T_{train}} \frac{1}{T_{train}} \sum_{j=1}^{T_{train}} \frac{1}{n} \left\| \hat{X}^{t+j} - X^{t+j} \right\|_2^2.$$

All variables are normalized using Z-score statistics computed on the training set.

[3] Eliasof, Haber, Treister, and Schonlieb, "Data-driven higher order differential equations inspired graph neural networks." ICLR Workshop on AI4 for Differential Equations, 2024.

Multiple Target Fields

Dynamic fields interfere with each other.

Predicting multiple fields simultaneously may be easier than predicting them individually.

Currently, we use five target fields:

1. Liquid-phase density;
2. Gas saturation;
3. Gas pressure;
4. Liquid pressure;
5. Gas density.

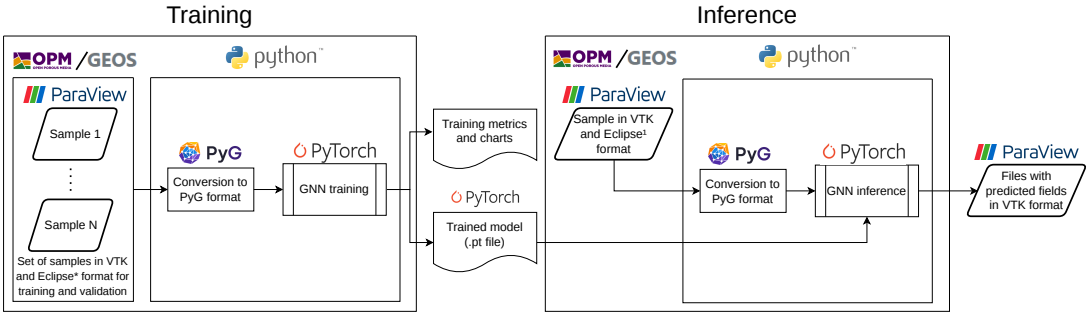
Node and Edge Features

	Feature	Description	Dimension
Node Features	V_i	Cell volume	1
	k_i	Isotropic scalar permeability	1
	p_i	Porosity	1
	n_i	Cell type	4
	c_i	Cell center coordinates	3
	s_i^t	Current target values*	5
Edge Features	$c_i - c_j$	Displacement vector	3
	$\|\|c_i - c_j\|\|$	Euclidean distance	1
Output	s_i^{t+1}	Next step target values*	5

Graph features for each cell i, each pair of cells i, j, and the node output of the model. Feature n_i is a one-hot encoded vector denoting the type of cell i among four categories: normal, well, fault, and boundary. Vector c_i denotes the centroid of cell i.

* Liquid-phase density, gas saturation, gas pressure, liquid pressure, and gas density.

AnisoMeshGraph-LSTM Machine Learning Pipeline



*Eclipse files (.INIT and .EGRID) are used to extract static and structural mesh information for OPM samples only.

Machine Learning Pipeline: developed in Python using the PyTorch framework and can be run via a command-line interface. Executed by Slurm jobs on clusters with GPUs. All computations on a NVIDIA H100 with 94GB VRAM.

Experiments' Hyperparameters

Hyperparameter	Value
Latent node dimension (d_v)	32
Latent edge dimension (d_e)	32
MLP hidden size	128
Number of message-passing layers (L)	20[4]
GraphConv–LSTM hidden size	32
Chebyshev order (K)	8
Optimizer	Adam
Learning rate	1×10^{-3}
Batch size	8
Training epochs	200
Time step (Δt)	$90\,\mathrm{s}$

[4] Tesán, et al. "On the under-reaching phenomenon in message passing neural PDE solvers: Revisiting the CFL condition." Computer Methods in Applied Mechanics and Engineering 449, 2026.

Results

We tested three different models, each trained using the 10 trajectories, on different time intervals:

1. **Injection regime:** Trained predominantly during the injection period ($t \leq 27{,}000\,\mathrm{s}$);
2. **Post-injection regime:** Trained only after the end of injection ($t > 27{,}000\,\mathrm{s}$);
3. **Both regimes:** Trained on the entire time interval;
4. The SPE11A geology specified in pyopmspe11 is used as ground truth in the validation.

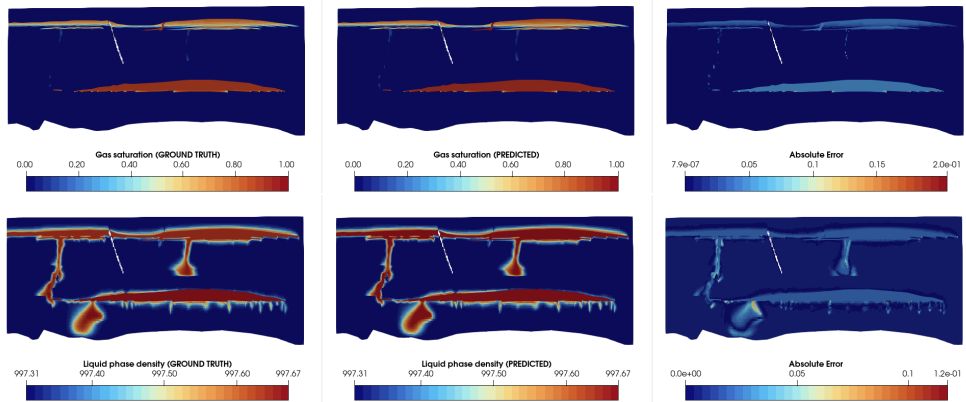
Error Analysis

$$\mathrm{RMSE} = \sqrt{\frac{1}{|V|} \left\| \mathbf{X}_t - \hat{\mathbf{X}}_t \right\|_2^2}$$

Table: RMSE under different training regimes and forecasting horizons. Errors are reported for gas saturation (S_g, dimensionless) and liquid-phase density (ρ_ℓ, kg m^{-3}). All table entries are scaled by $\times 10^{-3}$.

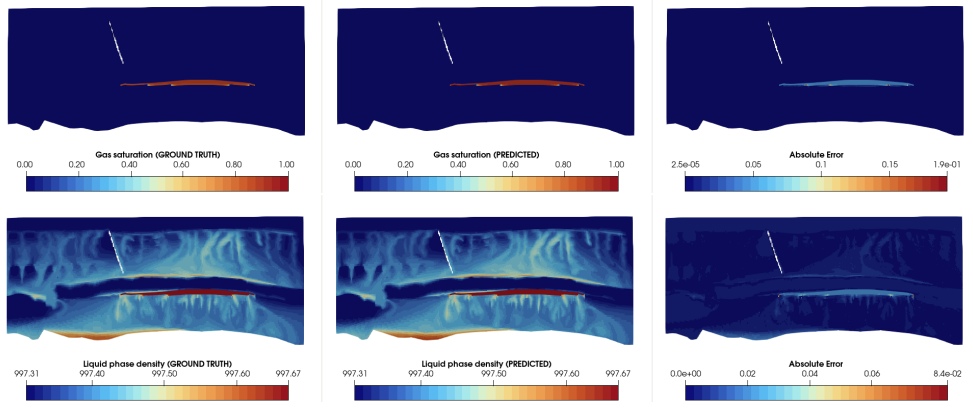
Forecast Horizon	**Injection regime only**		**Post-injection regime**		**Both regimes**	
	S_g	ρ_ℓ	S_g	ρ_ℓ	S_g	ρ_ℓ
RMSE (1-step)	1.164 ± 1.064	0.793 ± 0.920	0.553 ± 1.190	0.42 ± 0.82	0.562 ± 1.152	0.476 ± 0.963
RMSE (10-steps)	15.22 ± 4.75	10.743 ± 2.86	4.80 ± 4.71	3.247 ± 2.67	4.946 ± 4.622	3.434 ± 2.965
RMSE (50-steps)	85.35 ± 21.41	71.47 ± 10.34	23.91 ± 10.44	16.40 ± 6.263	24.734 ± 10.673	16.914 ± 6.457

End of Injection – Training under both regimes



Prediction at time $t = 23{,}400$s (starting inference from $t = 22{,}500$s) using the model trained during both regimes. Top, gas saturation; bottom, liquid phase density ($kg\,m^{-3}$).

End of Dissolution – Training under both regimes



Prediction at time $t = 405{,}900$s (starting inference from $t = 405{,}000$s), using the model trained during both regimes. Top, gas saturation; bottom, liquid phase density ($kg\,m^{-3}$).

Conclusions and Discussion

- Plume migration dynamics are captured without spurious oscillations or numerical drift under unseen geological configurations
- Complex fingering in liquid-phase density is also well represented
- Incorporating **geometry-informed anisotropic message passing**, the proposed model captures the strongly directional transport mechanisms that govern plume migration, including sharp gas–water interfaces and density-driven fingering
- AnisoMeshGraph-LSTM generalizes robustly beyond the training data while error remains acceptable.
- Findings highlight importance of anisotropic spatial modeling and point to promising directions for physically informed surrogate models for carbon storage and other subsurface energy systems at scale.

ACKNOWLEDGEMENTS

This study was partially financed by:

- Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brazil (CAPES)–Finance Code 001;

- CNPq/Brazil.

This research was also sponsored by TotalEnergies E&P Brazil under the 1% ANP obligation (GNN CO_2 Project 24630-6), and TotalEnergies EP R&T US.

COPPE
UFRJ

TotalEnergies