Advancing Graph Neural Networks for CO_2 Plume Migration in Complex Geological Formations

Rodrigo S. Luna [1,2], Thiago H. N. Coelho[1,2], Roberto M. Velho[1,2], Adriano M. A. Cortes[1,2], Renato N. Elias[2,4], Alexandre G. Evsukoff[2,4], Fernando A. Rochinha[2,3], Herve Gross[5], Mauricio Araya-Polo[5], Alvaro L. G. A. Coutinho[2,4]

[1]*Systems and Computer Engineering, COPPE/Federal University of Rio de Janeiro*
[2]*High Performance Computing Center, NACAD-COPPE/Federal University of Rio de Janeiro*
[3]*Mechanical Engineering, COPPE/Federal University of Rio de Janeiro*
[4]*Civil Engineering, COPPE/Federal University of Rio de Janeiro*
[5] TotalEnergies

(luna, tcoelho, robertovelho)@cos.ufrj.br, (adriano, rnelias)@nacad.ufrj.br, alexandre.evsukoff@coc.ufrj.br, faro@mecanica.coppe.ufrj.br,(herve.gross, mauricio.araya)@totalenergies.com, alvaro@nacad.ufrj.br

Abstract. Carbon Capture and Storage (CCS) is essential for mitigating climate change by permanently storing industrial CO_2 emissions in subsurface formations. Simulating the migration of injected CO_2 plumes presents significant computational challenges. Deep learning surrogate models offer orders-of-magnitude speedups compared to conventional numerical simulators while maintaining reasonable accuracy. Graph Neural Networks (GNNs) approaches such as MeshGraphNet can naturally handle unstructured meshes, thereby representing heterogeneous subsurface environments and geological boundaries that are critical to CO_2 migration. We present our MGN-LSTM implementation for the temporal prediction of CO_2 plume migration in faulted reservoirs, incorporating algorithmic improvements to enhance prediction stability and efficiency, and augmenting the loss function with physics-aware regularization. By incorporating CO_2 mass conservation, we achieve physically consistent predictions even with limited training data, improving the model's generalization to unseen geological configurations.

Keywords: Graph neural networks, Carbon Storage, Two-phase flow, Surrogate model, Deep Learning.

1 Introduction

Accurate and efficient prediction of CO_2 plume migration in complex geological formations is critical for the safe deployment of carbon capture and storage (CCS) technologies. Traditional numerical simulators, while reliable, incur prohibitively high computational costs when simulating long-term storage scenarios or performing uncertainty quantification studies. Recent surrogate modeling approaches, such as Fourier Neural Operators and MeshGraphNet (MGN), have demonstrated substantial speedups, with MGN naturally accommodating the unstructured meshes typical of geological reservoirs [1, 2]. However, purely data-driven surrogates may violate fundamental physical constraints, notably mass conservation, leading to unstable long-term forecasts. In this work, we enhance the MGN framework with a Long Short-Term Memory (LSTM) recurrent neural network and integrate a CO_2 mass-conservation loss term, similar to that of Chandra et al. [3]. This approach enforces physical consistency without sacrificing computational efficiency. We validate our approach on 500 synthetic reservoir realizations and benchmark it on the same five test meshes reported by Ju *et al.* [2]. This paper is organized as follows: Section 2 formalizes the problem and graph representation; Section 3 details the surrogate architecture and physics-informed loss; Section 4 presents the experimental setup and results; and finally, Section 5 concludes with the final remarks.

2 Problem Statement

We consider two components (H_2O and CO_2) that can form two fluid phases (supercritical CO_2 and the aqueous-brine phase) in a compressible porous medium. We employ a 2D domain in the x–y plane where pure supercritical CO_2 is injected through a single well at a rate of 0.058 kg/s for 950 days, assuming a storage reservoir with unit meter thickness. The domain is discretized with unstructured polygonal perpendicular bisector (PEBI) meshes on a 1 km $\times$ 1 km $\times$ 1 m domain containing an injector well and two straight impermeable faults conformal to the mesh, following Ju et al. [2].

We generated 500 synthetic geological models. The domain shape and the positions of two impermeable faults remain fixed across all realizations. Fault line 1 endpoints are at (100 m, 300 m) and (400 m, 600 m), while the fault line 2 endpoints are at (400 m, 500 m) and (800 m, 800 m). Each synthetic model varies in geological parameters (specifically permeability), mesh configuration, and well location. For each model, the injection well is randomly placed within a 200 m x 200 m box centered at the origin. A PEBI mesh, conforming to the faults and is refined around the well, is constructed using the Matlab Reservoir Simulation Toolbox (MRST) [4]. Permeability values are assigned using SGeMS[1], with a mean of 3.912 log(mD) and a standard deviation of 0.5 log(mD). Porosity is held constant at $\varphi = 0.2$ for all cells. Simulations are conducted for each model using GEOS[2].

3 Surrogate Model

This section describes the graph representation of the PEBI mesh, the MGN–LSTM surrogate for predicting CO_2 saturation in faulted reservoirs, and the incorporation of a physics-informed CO_2 mass-conservation loss following Chandra et al. [3].

3.1 Graph Representation

To prepare input data for the model, the mesh is represented by an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ and $E \subseteq V \times V$ denote, the set of nodes and edges, respectively, with $n = |V|$ and $m = |E|$. Each cell i is mapped to a node $v_i \in V$ and each pair of connected cells i and j with nonzero transmissibility is mapped to an edge $(v_i, v_j) \in E$. The superscript t denotes discrete time.

	Feature	**Description**	**Dimension**
Node Features	s_i^t	Current CO_2 saturation	1
	V_i	Cell volume	1
	k_i	Isotropic scalar permeability	1
	n_i	Cell type	4
	$k_{r,i}(s_i^{t-1})$	Last relative permeability	1
Edge Features	$c_i - c_j$	Displacement vector	2
	$\|\|c_i - c_j\|\|$	Euclidean distance	1
Output	s_i^{t+1}	Next step CO_2 saturation	1

Table 1. Graph features for each cell i, each pair of cells i, j, and the node output of the model.

The node and edge features are presented in Table 1. Feature n_i is a one-hot encoded vector denoting the type of cell i among four categories: normal, well, fault, and boundary. The vector c_i denotes the centroid of cell i.

During training, the model predicts the CO_2 flow through the porous medium. At each step, the model outputs the predicted gas saturation at the next time step, s^{t+1}, using as input the gas saturation from the current step, s^t, which was itself predicted at the previous step $t-1$. This results in an autoregressive prediction process, where each step depends on the model's past outputs. Note that the relative permeability, k_r, is a function of the gas saturation from the preceding time step, s^{t-1}.

3.2 MeshGraphNet-LSTM

MeshGraphNet [1] is a graph neural network architecture designed for mesh-based physical simulations. It represents the simulation domain as a graph, where nodes correspond to mesh vertices and edges encode the

[1]https://sourceforge.net/projects/sgems/
[2]https://www.geos.dev

relationships between adjacent vertices. The architecture comprises three main components: encoder, processor, and decoder that work together to learn the underlying dynamics. During training, the model is optimized to perform one-step-ahead prediction: given the current state X^t, it predicts the next state X^{t+1}. At inference time, the model generates predictions via autoregressive rollouts of variable horizon. However, this procedure is inherently prone to error propagation: minor inaccuracies in one-step predictions amplify over successive steps, leading to a progressive degradation of long-term forecasting accuracy. To address this issue, a Long Short-Term Memory (LSTM) module is incorporated into the training procedure.

Let the initial-state graph be defined as a directed attributed graph $G^0 = (V, E, X^0, E_{\text{attr}})$, where $X^0 \in \mathbb{R}^{n \times F_v}$ is the node-feature matrix whose row $x_i \in \mathbb{R}^{F_v}$ is the feature vector for node v_i, representing both static and dynamic attributes associated with node and $E_{\text{attr}} \in \mathbb{R}^{m \times F_e}$ is the edge feature matrix whose row $e_{ij} \in \mathbb{R}^{F_e}$ is the feature vector for edge (v_i, v_j), where F_v and F_e denote the dimensionality of the node and edge feature vectors, respectively. Our objective is to recover the dynamic variable s encoded in X after T autoregressive rollouts, e.g., to predict the time-evolved feature matrix X^T as follows:

$$\begin{aligned} \hat{X}^0 &= X^0, \\ \hat{X}^{t+1} &= F_{\text{MGN\_LSTM},\theta}\big(G,\, \hat{X}^t\big), \quad \text{for } t \in \{0, \dots, T-1\}\,. \end{aligned} \tag{1}$$

where $F_{\text{MGN\_LSTM},\theta}$ denotes the MGN–LSTM surrogate model, parameterized by model weights θ, which integrates MGN's spatial message-passing with a LSTM unit to capture full spatiotemporal dynamics.

Encoder, Processor, Graph-LSTM and Decoder

Encoder. The encoder maps the initial node feature matrix $X^0 \in \mathbb{R}^{n \times F_v}$ and the static edge feature matrix $E_{\text{attr}} \in \mathbb{R}^{M \times F_e}$ into latent embeddings. Each node v_i with initial feature vector $x_i^0 \in \mathbb{R}^{F_v}$ — where x_i^0 is a row of X^0 — is embedded individually via a node MLP ϕ_v (Multi-Layer Perceptron).

$$z_i^{t,0} = \phi_v^0(x_i^{t,0}) \in \mathbb{R}^{d_v},$$

where $z_i^{t,0}$ denotes the latent embedding of node v_i in a d_v-dimensional latent space in layer 0 at time t. Similarly, for each edge $(v_i, v_j) \in E$ with initial feature vector $e_{ij}^0 \in \mathbb{R}^{F_e}$, we compute the latent edge embedding via an edge MLP ϕ_e:

$$z_{ij}^{t,0} = \phi_e^0(e_{ij}^{t,0}) \in \mathbb{R}^{d_e},$$

where $z_{ij}^{t,0}$ denotes the latent embedding of edge (v_i, v_j) in a d_e-dimensional latent space.

Processor. The processor module consists of L sequential message-passing steps and it is initialized with the latent node embeddings z_i^0 and latent edge embeddings z_{ij}^0 computed by the encoder. At each message-passing step $l \in \{1, \dots, L\}$, the latent edge embeddings are updated according to:

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

Subsequently, each node embedding is updated by aggregating incident edge embeddings and combining them with the previous node embedding:

$$z_i^{t,l} = \phi_v^l\left(\left[z_i^{t,l-1},\, \sum_{v_j \in \mathcal{N}(v_i)} z_{ij}^{t,l}\right]\right),$$

where ϕ_e^l and ϕ_v^l represent edge- and node-specific MLP at l-th step, implemented with residual connections and ReLU activation; $\mathcal{N}(v_i)$ denotes the neighborhood of node v_i, and $[\cdot]$ indicates concatenation along the feature dimension.

GraphConv–LSTM. To mitigate error accumulation over multiple rollout steps, an LSTM cell is integrated after each message-passing block, allowing the model to capture and propagate temporal dependencies in the latent node embeddings. Specifically, we follow the Graph Convolutional LSTM formulation proposed by Seo et al. [5], where the standard convolution in ConvLSTM is replaced by a spectral graph convolution operator $*_G$.

Let $\mathbf{Z}^{t,L} \in \mathbb{R}^{n \times d_v}$ denote the matrix of node embeddings obtained from the message-passing module at last step L and time t, and let $\mathbf{H}^{t-1}$, $\mathbf{C}^{t-1}$ be the hidden and cell states of the LSTM at the previous time-step. The GraphConv-LSTM cell updates are defined as:

$$\begin{aligned}
\mathbf{i}^t &= \sigma(\mathbf{W}_{xi} *_G \mathbf{Z}^{t,L} + \mathbf{W}_{hi} *_G \mathbf{H}^{t-1} + \mathbf{b}_i),\\
\mathbf{f}^t &= \sigma(\mathbf{W}_{xf} *_G \mathbf{Z}^{t,L} + \mathbf{W}_{hf} *_G \mathbf{H}^{t-1} + \mathbf{b}_f),\\
\mathbf{C}^t &= \mathbf{f}^t \odot \mathbf{C}^{t-1} + \mathbf{i}^t \odot \tanh(\mathbf{W}_{xc} *_G \mathbf{Z}^{t,L} + \mathbf{W}_{hc} *_G \mathbf{H}^{t-1} + \mathbf{b}_c),\\
\mathbf{o}^t &= \sigma(\mathbf{W}_{xo} *_G \mathbf{Z}^{t,L} + \mathbf{W}_{ho} *_G \mathbf{H}^{t-1} + \mathbf{W}_c \odot \mathbf{C}^t + \mathbf{b}_o),\\
\mathbf{H}^t &= \mathbf{o}^t \odot \tanh(\mathbf{C}^t)
\end{aligned} \tag{2}$$

Here, $\mathbf{i}^t$, $\mathbf{f}^t$, and $\mathbf{o}^t$ denote the input, forget, and output gates at time t, respectively; $\odot$ is the element-wise (Hadamard) product. Each gate applies a sigmoid activation $\sigma(\cdot)$ to its affine transform, parameterized by learnable weight matrix $\mathbf{W}_*$ and bias vector $\mathbf{b}_*$. The graph convolution operator $*_G$ is implemented using Chebyshev polynomials [6].

Decoder. The decoder maps the latent node embeddings at time $t+1$, obtained from the GraphConv–LSTM module, back to the physical variable of interest. Let $\mathbf{Z}^{t,L} \in \mathbb{R}^{n\times d_v}$ be the matrix of latent node representations. The decoder applies a shared MLP ϕ_{dec} to each node embedding to recover the predicted state $X^{t+1} \in \mathbb{R}^{n\times F_v}$:

$$\hat{x}_i^{t+1} = \phi_{\mathrm{dec}}(z_i^t), \quad \text{for each } v_i \in V,$$

so that

$$\hat{X}^{t+1} = [\hat{x}_1^{t+1}, \dots, \hat{x}_n^{t+1}]^\top \in \mathbb{R}^{n\times F_v}.$$

This output represents the predicted physical state (e.g., CO$_2$ saturation) at the next time step.

3.3 Loss and Mass Conservation

We extend the methodology by Ju et al. [2], by incorporating a physics-informed loss function [3, 7], to enhance physical consistency and exert greater control over the dynamics during training:

$$L(S,\hat{S}) = L_s + \alpha L_m = \frac{\|S-\hat{S}\|_2}{\|S\|_2} + \alpha \frac{\left\|\int_V m_{\mathrm{CO_2}} - \int_V \hat{m}_{\mathrm{CO_2}}\right\|_2}{\left\|\int_V m_{\mathrm{CO_2}}\right\|_2}, \tag{3}$$

where $S = \{s_i^t\}$ denotes the set of CO_2 saturation values for all cells i and all time steps t. The loss function in Equation 3 comprises two terms: the normalized L_2 data loss between the ground truth variable S and its prediction $\hat{S}$, L_s, augmented by a physical constraint, the mass-conservation term based on the mass of CO$_2$ in the domain, L_m. The regularization weight is denoted by α. The CO$_2$ mass in each cell is computed as:

$$m_{\mathrm{CO_2}}^{(i)} = \rho_g\, \varphi\, s_i\, V^{(i)}\, c_{g,\mathrm{CO_2}}^i \tag{4}$$

where ρ_g is the gas density, s_i is the CO$_2$ saturation, $V^{(i)}$ is the volume of cell i, and $c_{g,\mathrm{CO_2}}^i$ is the mass fraction of CO$_2$ in the gas phase. Note that for simplicity, we do not consider the mass fraction of CO$_2$ in the liquid phase. The total mass is derived from a domain integral, discretized over all cells in the mesh:

$$m_{\mathrm{CO_2}} = \sum_{i=1}^{N} m_{\mathrm{CO_2}}^{(i)}, \tag{5}$$

where N is the total number of mesh cells. The parameter α weights the enforcement of the mass conservation constraint.

4 Evaluation

This section presents the experimental configuration and reports the results obtained from the MGN–LSTM model.

4.1 Experimental Setup

Model Setup. We adopt the MGN architecture with 10 message-passing layers in the processor module. The model is trained for 1000 epochs using the Adam optimizer, with a learning rate of 1×10^{-3}, a batch size of

16, and a latent embedding size of 128. To capture temporal dependencies, LSTM cells are integrated into the model, using Chebyshev polynomial graph convolutions with 8 terms. The number of LSTM cells corresponds to the full rollout length, which is set to 11. We evaluate the effect of the weighting coefficient α across the values $[0, 1, 10, 100, 1000]$. For each value of α, the model is trained independently and the version yielding the best validation performance is selected as the final model.

Training. The dataset comprises 500 PEBI mesh samples. Of these, 450 (90%) were allocated to the training set, 45 (9%) to the validation set, and the remaining five (1%) to the test set. To enable a direct comparison with Ju *et al.* [2], the test set consists of the same five samples they evaluated The validation set was drawn at random (using a fixed seed for reproducibility) from the training pool, with no overlap between validation and test samples.

4.2 Results

We present the results obtained with the MGN–LSTM for meshes 469, 484,490, 494 and 497, the same ones reported by Ju et al.[2]. Observe that we compute the mass conservation term for $\alpha = 0$, but this value does not enter the optimization procedure. It represents mass conservation for the unregularized solution. Figure 2 displays, for each value of α, the predicted mesh state alongside the ground truth in the top, and the absolute relative error at the final prediction time in the bottom. Note that nodes located closer to the injection cell exhibit large relative errors than those farther from the injection cell for all values of α, and the relative error increases uniformly across all cells as α grows. This behavior is captured by the Complementary Cumulative Distributions Functions (CCDFs) of the absolute and relative errors, shown in Figure 1. CCDFs indicate that, for any given error threshold, the probability $P(\text{Error} \geq x)$ increases with α. As α increases, both CCDF curves shift steadily to the right, indicating that extreme errors are increasing. Specifically, the median absolute error grows from order of 10^{-3} with $\alpha = 0$ to 10^{-2} at $\alpha = 100$, and the fraction of cells exceeding any fixed absolute-error threshold similarly increases. The pattern also holds for relative errors: $\alpha = 0$ yields with fewer high percentage outliers, while larger α raises the median relative error from 2% at $\alpha = 0$ to bigger than 10% at $\alpha = 100$. For example, only 20% of pixels exceed a 1% relative error at $\alpha = 0$, while more than 50% with $\alpha = 100$. These distributions demonstrate that increasing α uniformly degrades predictive accuracy across the meshes.

α	Data Loss - L_s (RMSE)	Physical Loss - L_m	Total Loss
0	0.0073 ± 0.0015	0.00036 ± 0.00004	0.00770 ± 0.00148
1	0.0089 ± 0.0006	0.00051 ± 0.00008	0.00939 ± 0.00065
10	0.0301 ± 0.0012	0.00056 ± 0.00008	0.03068 ± 0.00119
100	0.0200 ± 0.0010	0.00068 ± 0.00008	0.02068 ± 0.00095
1000	0.0325 ± 0.0069	0.00070 ± 0.00010	0.03316 ± 0.00687

Table 2. Mean losses over the test set ($\pm$ one standard deviation across the five meshes) for different values of α.

Table 2 reports the loss values computed on the test set for different values of the physics-informed weighting coefficient α. The reported values correspond to the final model selected based on the best validation performance. The loss is decomposed into data loss, physics loss, and total loss components, as defined in Equation 3.

5 Conclusions

We have shown that a surrogate based on MGN-LSTM can reproduce well a CO_2 plume migration in a faulted reservoir. Incorporating a CO_2 mass conservation term in the loss function contributes to the surrogate interpretability. However, our numerical experiments suggest that, for the same network parameters, excessive penalization via a physics-informed regularization term does not improve gas saturation predictions. Further studies should investigate the influence of other hyperparameters combined with a physics-informed loss regularization.

Acknowledgements. This study was financed in part by CAPES, Brazil - Finance Code 001. This work is also partially supported by FAPERJ, CNPq, and TotalEnergies.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the au-

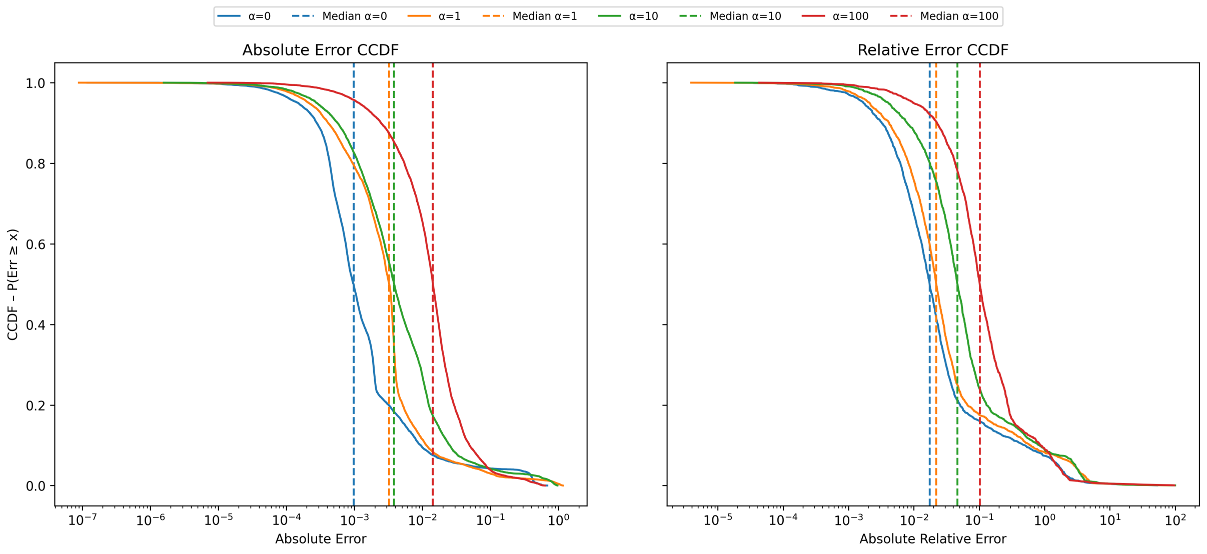

Figure 1. Complementary cumulative distribution functions (CCDFs) of the absolute error (left) and the absolute relative error (right). Dashed lines indicate the median value of each error distribution.

thorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.
[2] X. Ju, F. P. Hamon, G. Wen, R. Kanfar, M. Araya-Polo, and H. A. Tchelepi. Learning co2 plume migration in faulted reservoirs with graph neural networks. *Computers & Geosciences*, vol. 193, pp. 105711, 2024.
[3] A. Chandra, M. Koch, S. Pawar, A. Panda, K. Azizzadenesheli, J. Snippe, F. O. Alpak, F. Hariri, C. Etienam, P. Devarakota, and others. Fourier neural operator based surrogates for *co_2* storage in realistic geologies. *arXiv preprint arXiv:2503.11031*, 2025.
[4] K.-A. Lie. *An introduction to reservoir simulation using MATLAB/GNU Octave: User guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press, 2019.
[5] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pp. 362–373. Springer, 2018.
[6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, vol. 29, 2016.
[7] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, vol. 1, n. 3, pp. 1–27, 2024.

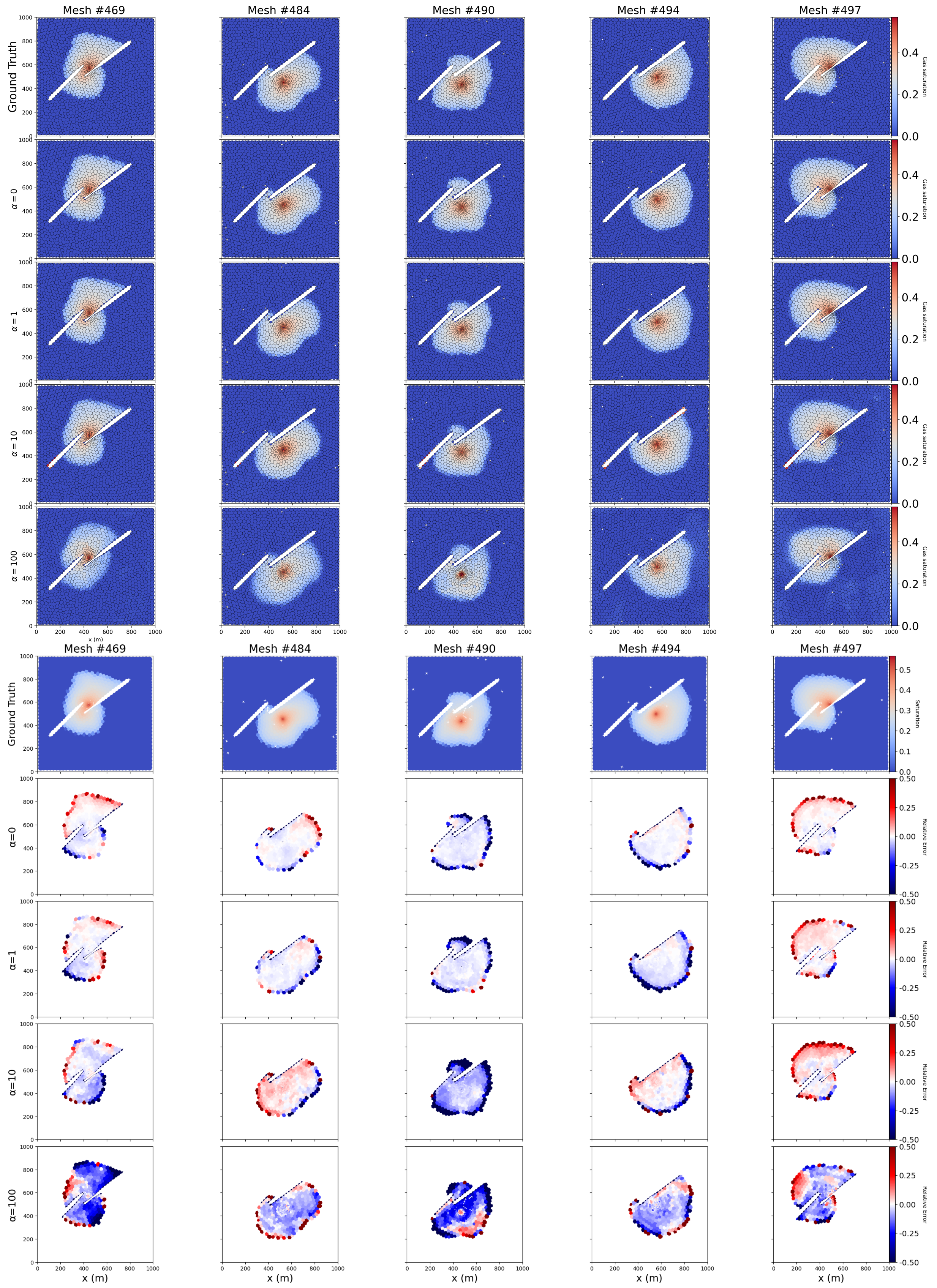

Figure 2. Spatial gas-saturation fields (top) and corresponding residual fields (Δ relative saturation = (prediction–ground truth)/ground-truth) (bottom) for five PEBI test meshes. Columns indicate individual meshes; rows show ground truth, and our method at $\alpha = 0, 1, 10, 100$.