Fourier Neural Operator based surrogates for CO_2 storage in realistic geologies

Anirban Chandra [1] **Marius Koch** [2] **Suraj Pawar** [1] **Aniruddha Panda** [3] **Kamyar Azizzadenesheli** [2] **Jeroen Snippe** [4] **Faruk O. Alpak** [5] **Farah Hariri** [2] **Clement Etienam** [2] **Pandu Devarakota** [1] **Anima Anandkumar** [6 2] **Detlef Hohl** [1]

Abstract

This study aims to develop surrogate models for accelerating decision making processes associated with carbon capture and storage (CCS) technologies. Selection of sub-surface CO_2 storage sites often necessitates expensive and involved simulations of CO_2 flow fields. Here, we develop a Fourier Neural Operator (FNO) based model for real-time, high-resolution simulation of CO_2 plume migration. The model is trained on a comprehensive dataset generated from realistic subsurface parameters and offers $\mathcal{O}(10^5)$ computational acceleration with minimal sacrifice in prediction accuracy. We also explore super-resolution experiments to improve the computational cost of training the FNO based models. Additionally, we present various strategies for improving the reliability of predictions from the model, which is crucial while assessing actual geological sites. This novel framework, based on NVIDIA's Modulus library, will allow rapid screening of sites for CCS. The discussed workflows and strategies can be applied to other energy solutions like geothermal reservoir modeling and hydrogen storage. Our work scales scientific machine learning models to realistic 3D systems that are more consistent with real-life subsurface aquifers/reservoirs, paving the way for next-generation digital twins for subsurface CCS applications.

[1]Shell Information Technology International Inc., Houston, TX, USA [2]NVIDIA, Santa Clara, CA, USA. [3]Shell India Markets Pvt. Ltd., Bangalore, India [4]Shell Global Solutions International B.V., Amsterdam, Netherlands [5]Shell International Exploration and Production Inc., Houston, TX, USA [6]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA. Correspondence to: Anirban Chandra <anirban.chandra@shell.com>, Pandu Devarakota <pandu.devarakota@shell.com>.

Proceedings of the 41st International Conference on Machine Learning (ICML) 2024 AI for Science workshop, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Carbon dioxide (CO_2) capture and storage (CCS) is a critical technology for decarbonization and achieving net zero emissions and climate goals. The first step of CCS involves the separation of CO_2 from the gases produced by large power stations or industrial plants. It is then transported to locations where it can be stored in geological formations below the surface. The process involves integration of CO_2 capture, compression, transport, and storage in the subsurface. The theoretical capacity of CO_2 storing in deep geological formations globally is vast and far exceeds that required to reach net-zero emissions (Rogelj et al., 2018; de Coninck & Benson, 2014; Kelemen et al., 2019). Currently only a small fraction of the potential global storage capacity is used. A major challenge in identifying the most suitable sites for storage is proving the containment of CO_2 within a reservoir post-injection. This is because CO_2 can migrate over time and potentially escape from the reservoir, posing a risk to the environment. In addition, the reservoir pressure propagation caused by CO_2 injection needs to be carefully managed to avoid surface seismicity hazards. We need technological advancements and breakthroughs to rapidly screen potential sites for CO_2 storage in a cost-effective and timely manner.

Historically, numerical simulations have been used to model fluid flow and other physical phenomena in subsurface reservoirs and aquifers (Aziz, 1979; Ertekin et al., 2001). Existing reservoir simulators have been extended to model CO_2 storage scenarios, but they can be computationally expensive, as they require solving non-linearly coupled partial differential equations (PDEs) (Nghiem et al., 2004; Wei & Saaf, 2009). This can limit their use for studying the behavior of individual sites, especially when the subsurface properties are uncertain. The challenge of computational efficiency becomes even more pronounced for CCS site screening applications. This is because a substantial number of simulations are typically required to quantify the effects of subsurface uncertainties. Uncertain subsurface variables can include the dip of the formation, rock and fluid properties, and a wide range of reservoir engineering parameters that govern the trapping of CO_2 in the subsurface. To accurately quantify the uncertainty surrounding the migration of the plume and the buildup and dissipation of pressure, it is

necessary to carry out a large number of simulations.

Several surrogate models have been developed in the recent past for accurately capturing the dynamics for CO_2 plume migration in the subsurface (Wen et al., 2021; Witte et al., 2022; Grady et al., 2023; Falola et al., 2023; Witte et al., 2023). In this study, we build on successes of FNO-based models (Azizzadenesheli et al., 2024) and extend it to larger and more practical subsurface representations using NVIDIA's Modulus package. We train our models on an extensive dataset of numerical simulations performed using a realistic set of subsurface parameters. Along with surrogate models that are $\mathcal{O}(10^5)$ faster than numerical simulations, we present and evaluate several physics based accuracy metrics that are relevant for assessing and monitoring CCS sites, such as CO_2 plume migration and total mass in the reservoir. In an attempt to improve the computational efficiency of our models, we explore super-resolution, a niche of operator methods (Li et al., 2020b;c; Kovachki et al., 2023). Additionally, we propose several strategies to enhance the reliability of the models' predictions, which is vital when evaluating actual geological sites for CO_2 storage such as outlier detection and enforcing mass conservation using physics informed loss functions (Li et al., 2021).

This paper is structured as follows: In Section 2 we discuss the problem setup along with details of the numerical flow simulations results that are used as training/testing data. Section 3 describes our machine learning methodology. We present and discuss results in Section 4.

2. Problem setup and dataset creation

To emulate realistic geological scenarios, we consider a three-dimensional dipping box model with a layered stratigraphic architecture. A schematic of the setup is shown in Figure 1. For simplicity, a 2D representation of the box is used in the schematic visualization; X, Y, and Z dimensions are 100km, 3km, 0.3km respectively. The set of nine input parameters considered in this study are reported in Table 1, which are selected based on advice from domain experts. Outputs are CO_2 mass accumulation (m_{CO_2}), gas saturation (S_g), and change in pressure (δp). Two-phase Darcy flow equations govern the dynamics of fluid flow within the reservoir (Aziz, 1979).

After a grid sensitivity study we arrive at a final resolution that has $241 \times 10 \times 200$ grid points in the X, Y, and Z direction, respectively. The chosen grid is the lowest resolution that captures the plume characteristics with the accuracy necessary for site assessment. While creating different samples (training dataset), the active region of the geometry changes based on the input parameters. In certain cases, the CO_2 plume reaches the edge of the active region and accumulates because of the aquifer boundary condition

Table 1. Scalar parameters used for parameterizing CO_2 storage model and surrogate model development.

Parameters	**Type**
Permeability, number of geological layers, heterogeneity scale, reservoir height	Geology (layering)
Reservoir dip angle	Geology
Porosity	Geology
Vertical-horizontal permeability ratio	Geology
Reservoir Pressure	Geology, Fluid
Reservoir Temperature	Fluid

employed in the simulation.

The number of grid points is constant across all samples leading to a simple data ingestion pipeline. We train the surrogate model to predict the solution at eight time snapshots (including the end-of-injection time) in the post-injection period. These time snapshots are temporally spread out in a logarithmic manner until the end-of-simulation time, which is assumed to be a sufficiently long time at which migration of the CO_2 plume approximately comes to a halt. Our spatial grid is also non-uniformly spaced with finer resolution near the injection location. For modeling of CO_2 storage, we use 20 years for injection period and 1000 years post-injection. Our ML models are trained to predict CO_2 plume migration after injection period. Such long time forecasting has not been carried out in previous machine-learning based CO_2 plume migration studies that have predominantly focused on the injection period and forecasting up to only $\mathcal{O}(10)$ years (Wen et al., 2021; 2022; 2023; Jiang et al., 2023).

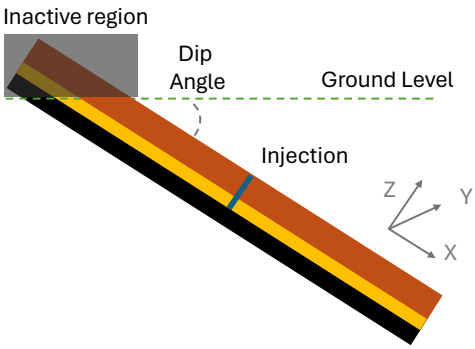

Figure 1. Schematic of the layered reservoir geometries. Different colors correspond to varying permeability. The parts of the reservoir protruding above ground level (reservoir ceiling), shaded in black, represent void-blocks (inactive regions), and changes based on reservoir dip angle and other input parameters (first row of Table 1).

Using a proprietary reservoir simulator for modeling multi-

phase multicomponent flow in porous media (Alpak & Vink, 2018; Por et al., 1989; Regtien et al., 1995) we generate $10,330$ samples for training and $2,593$ for testing. A separate validation set is not considered in this study as model performance was fairly consistent across the chosen hyperparameter combinations. Our choice of hyperparameters was based on previous studies of similar systems in simplified geometries (Pawar et al., 2023; Wen et al., 2021). Combinations of the nine input parameters are chosen using Latin hypercube sampling. Ranges of these parameters are documented in the Appendix A.1 (Table 3). Output variables of interest are: (a) CO_2 gas saturation - S_g (b) CO_2 mass accumulation - m_{CO_2} (c) Change in pressure from the initial state - δp.

3. Machine learning methods

3.1. Neural operators

Neural operators are capable of learning the relationships between function spaces. They can map any parametric functional input to its corresponding output function. Operator learning here involves predicting one function based on other functions. In a physical system characterized by partial differential equations (PDEs), the input functions could be the initial condition u_0 defined on the physical domain x, the boundary condition u_b and the forcing term f, defined on physical temporal domain (x, t). The output in this case would be a PDE solution $u(x, t)$ at all x's and t's, where x and t are the space and time coordinates, respectively. For our problem, we define the domain $D \in R^d$ be a bounded and open set; V be the input function space defined on D that takes values in R^{d_v}; and U be the output function space on D that takes values in R^{d_u}. The mapping between the input function v and output function u is denoted by an operator

$$\mathcal{G} : \mathcal{V} \ni v \to u \in \mathcal{U}. \tag{1}$$

We aim to approximate $\mathcal{G}$ with an operator $\mathcal{G}_\theta$ from training data $\mathcal{T} = \{(v^{(1)}, u^{(1)}), (v^{(1)}, u^{(2)}), \dots (v^{(m)}, u^{(m)})\}$, where $v^{(i)}$ is drawn from a probability measure $\mu \in \mathcal{V}$ and $u^{(i)} = G(v^{(i)})$. Here θ refers to the trainable parameters of the neural operator and we learn this operator $\mathcal{G}_\theta$ by minimizing the following problem with a cost functional $\mathcal{C}$

$$\arg\min_\theta \mathbb{E}_{v\in\mu}\big[\mathcal{C}\big(\mathcal{G}_\theta(v), \mathcal{G}(v)\big)\big]. \tag{2}$$

The Fourier neural operator (FNO) is an iterative architecture that employs kernel integral operations to learn mapping between two function spaces (Li et al., 2020a). In FNO, the kernel integral operator is substituted with a convolution operation defined in the Fourier space, i.e., the coefficients of the Fourier series of the output functions are learned from the data. The FNO consists of a lifting layer, a point wise operator that maps the input co-dimension to higher dimensional representation using a fully connected neural network, then several Fourier layers, and finally the projection layer that maps the high-dimensional output co-dimension of the last Fourier layer to the output function. Several modifications have been proposed in the literature to apply FNOs to input and output functions defined on various domains and to handle complex geometry (Lu et al., 2022).

For the output of the l'th Fourier layer z_l with d_v channels, the Fourier layer can be written as follows:

$$\mathcal{F}^{-1}\big(\mathcal{R}_l \cdot \mathcal{F}(z_l)\big) + W_l z_l \tag{3}$$

where

- $\mathcal{F}$ is the Fourier transform and is applied to each channel of z_l separately. The higher modes of $\mathcal{F}(z_l)$ are truncated to retain only k modes for computational efficiency. Thus, $\mathcal{F}(z_l)$ has the shape $k \times d_v$.
- The W_l is yet another pointwise operator acting as a residual connection.
- We then multiply each mode index of $\mathcal{F}(z_l)$ with a learnable weight matrix (complex number) of shape $d_v \times d_v$ which form the weight matrix $\mathcal{R}_l \in \mathbb{C}^{(d_v \times d_v \times k)}$.
- Finally, we perform inverse Fourier transforms and apply a point-wise nonlinear activation function of choice. When the discretization grids are regular, the approximate integral of Fourier transforms are carried out using fast Fourier Transform (FFT). In the inverse FFT$\mathcal{R} \cdot \mathcal{F}(z_l)$ zeros fill in the truncated modes.

The residual layer term is added to the output of each of the Fourier layers before it is passed through an activation function. This bias term allows FNOs to handle data with non-periodic boundary conditions as well as high frequency components. We use the relative l_p loss to train the neural operator models. The loss function can be written as,

$$L(u, \hat{u}) = \frac{\|(u - \hat{u})\|_p}{\|u\|_p} + \alpha \frac{\| \int_V m_{CO_2} - \int_V \widehat{m}_{CO_2}\|_p}{\| \int_V m_{CO_2}\|_p}, \tag{4}$$

where u is the ground truth, $\hat{u}$ is the predicted output, $\int_V m_{CO_2}$ is the ground truth for total CO_2 mass, $\int_V \widehat{m}_{CO_2}$ is the predicted total CO_2 mass, p is the order of norm, and α is the hyperparameter to assign a weight to the mass conservation penalty term.

3.2. Software framework

Our workflow is based on NVIDIA's open-source package Modulus (NVIDIA Modulus Team). NVIDIA Modulus

is a pytorch-based framework for scientific machine learning applications that provides an extensive collection of network architectures and convenience functions for setting up training and inference pipelines. Several hardware and software optimizations are also implemented implicitly. In our work, we leverage the 4D FNO models that are available in Modulus' architectural library and heavily use `DistributedManager` for scaling to multi-GPU and multi-node settings. More details are provided in A.2.

4. Results and Discussion

Spatiotemporal distributions of mass of CO_2 (m_{CO_2}) and gas saturation (S_g) are two variables of interest, that are correlated to each other via phase densities. Another variable of interest is the pressure change compared to the initial pressure state, δp. Figure 2 shows 3D and 2D snapshots of a specific sample. For spatiotemporal visualization, we use $\overline{m}_{CO_2}$, which is m_{CO_2} scaled by grid volume and porosity, to obtain a better visual representation of the plume. In all cases, a good match between ground truth (numerical simulation) and ML prediction is observed. We evaluate several global error metrics, both numerical and physical, to assess the accuracy of our predictions.

4.1. Accuracy metrics

Figure 3(a) shows the mean absolute error (MAE) for the two solution variables of interest; MAE is generally a good metric for assessing the accuracy of the variables over the entire domain. In addition, we monitor physical metrics such as, $p_{90}[m_{CO_2}]$, which is the migration distance of the 90% plume mass contour from the injection location. In Figure 3 (b) we show the R^2 correlation plots of this metric at different points in time. With increasing time, the accuracy decreases slightly, as the plume migrates. Metrics for pressure (δp) need more careful consideration as global values are not informative. Instead, we explore a few point-wise (local) metrics for assessing the accuracy of the pressure predictions of the FNO based model. Histograms of maximum point-wise error in δp across samples are shown in Figure 3 (c). While most cases have a low error, there exists a $\mathcal{O}(10)$ outliers. Although this metric provides us with an upper bound, it is not a metric that cannot be easily evaluated in realistic settings. While monitoring CCS sites, we are often interested in the location where the maximum pressure occurs – typically near the injection well. Thus, we evaluate the prediction from the model at the location where the true pressure is maximum. Red circles in Figure 3(d) demonstrate the accuracy of the FNO model with respect to this metric. Furthermore, for monitoring of CCS sites, the pressure buildup at additional (site-specific) locations can be important. To emulate this, while avoiding a location bias, we randomly select a few additional locations from our test set and assess predictions from our model at the same location, as represented by the blue circles in Figure 3(d). In both metrics, we observe a R^2 score greater than 0.97, suggesting that the model would provide reasonable predictions in most cases.

4.2. Effect of weak imposition of mass conservation

While our models exhibit good accuracy in all the metrics discussed in the previous subsection, an additional important metric to assess model fidelity is CO_2 mass conservation, $\int_V m_{CO_2}$. As we generally train our models with the relative L_2 loss as shown in Equation 4 (with α=0), there is no explicit constraint on the total mass conservation. Therefore, to enforce better mass conservation, we explore loss functions with finite values of α. Figure 4(a) shows the distribution of relative errors in mass conservation for four settings of α. The histograms are constructed considering all time instances. In the base case, i.e. $\alpha = 0$, not only is the mean relative error (MRE) the highest, but several outliers exist. Here, MRE is used instead of MAE in this case to provide a more intuitive measure of the mismatch in mass conservation. Amongst all the considered values of α, MRE reaches a minimum $\alpha = 0.5$. Other metrics are shown in Figure 4(b). $MAE(m_{CO_2})$ increases with increasing α which is intuitive as the additional soft constraint indirectly modifies this metric. At $\alpha = 0.5$, an empirical optimum is achieved, where $MAE(S_g)$ and $p_{90}[m_{CO_2}]$ is comparable to the base case. At the highest value of α, these error metrics become worse, which is indicative of a harder optimization problem. Other engineered loss functions, akin to those explored by (Wen et al., 2022), could further improve accuracy of predictions.

4.3. Super resolution

A key difference between neural operators and neural networks is the ability to learn functional mappings rather than finite-dimensional approximations. Thus, a model trained on a coarse-resolution data can yield predictions at a finer resolution - this is regarded as super resolution. For investigating the accuracy of super-resolution experiments, the ML models are trained and tested on a variety of resolutions. $ML(r_t, r_i)$ denotes that the model was trained on resolution r_t and inferred on r_i, where r_t and r_i are selected from the set of resolutions $\{c_v, c_d, f\}$. c_v and c_d are volume-averaged and discrete representations of the variables of interest, respectively. Both types of down-sampling reduces the number of grid points by a factor of two in the y-direction. f is the original simulated (ground truth) data on the finer grid. For comparison, the original simulated data, $Sim(f)$, is also down-sampled in the aforementioned ways, and is represented by $Sim(c_v)$ and $Sim(c_d)$. It should be noted that only the variables of interest (input and outputs) undergo two types down-sampling operations. The spatial

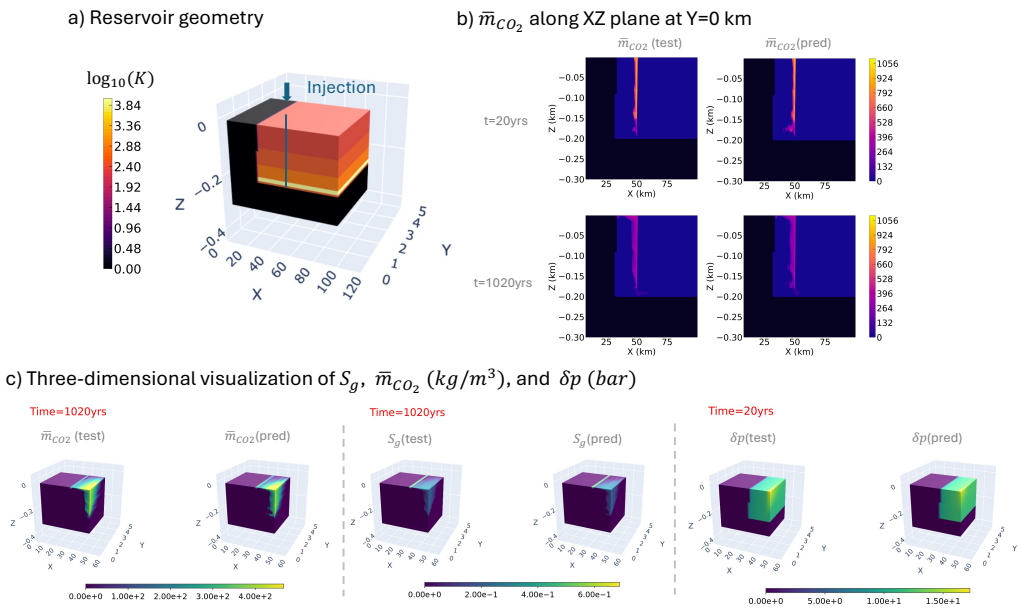

Figure 2. Visualization of fields. (a) 3D permeability map of the reservoir along with injection location. Distance is measured in km and permeability (K) in milliDarcy (mD). (b) $\overline{m}_{\mathrm{CO_2}}$, measured in kg/m^3, as viewed on the XZ plane (c) Cross sectional views of various fields at the injection location. $m_{\mathrm{CO_2}}$ and S_g are shown at final time whereas δp at initial time because of their respective importance in decision making.

grid is always discretely downsampled. More information regarding downsampling is provided in Appendix A.6.

Figure 5(a) shows the performance of models trained and evaluated at various resolutions. All the metrics - MAE for the $MAE(S_g)$, $MAE(m_{\mathrm{CO_2}})$, and $R^2(p_{90}[m_{\mathrm{CO_2}}])$ - are evaluated with respect to numerical simulations. $[Sim(f), ML(f, f)]$, represented by black circles, is the base case where the model was trained and evaluated on the finer grid, and compared with simulated data on the finer grid. Similarly, $[Sim(f), ML(c_v, f)]$, denoted by orange squares, signify that the model was trained on a volume-averaged coarse grid and inference was performed on a fine grid, and compared against finer-grid numerical simulations to obtain the different error metrics. In all cases and through all time-instances, base case performs the best while $[Sim(c_d), ML(c_d, c_d)]$ performs the worst. The advantage of neural operators is predominantly demonstrated by the 'orange squares'. Even though these cases were trained on a volume-averaged representation of variables, the error metrics closely follow the base case. The $[Sim(f), ML(c_d, f)]$ comparison is slightly worse than the volume-averaged case as our grid is non-uniform and volume-averaging embeds more physical information. Snapshots $m_{\mathrm{CO_2}}$ on two planes spanning the y-direction at the injection location are shown in Figures 5(c) and (d). In all cases, $ML(c_v, f)$ represents $ML(f, f)$ more closely as compared to $ML(c_d, f)$. Errors are more prominent closer to the geometrical boundaries ($Y = 30$km) due to coarser grid spacing.

4.4. Outlier detection

While the developed surrogate models allow rapid prediction of $p_{90}[m_{\mathrm{CO_2}}]$ for various geologic scenarios, in certain conditions, the predictions can be misleading. Generally, a slight reduction in overall accuracy is expected and acceptable due to geologic uncertainties, but highly deviating predictions can be misleading. In, Figure 6, we present a methodology for detecting outliers based on model uncertainties. Slightly changing the hyperparameters (decoder width, Fourier modes, number of FNO layers) of the model leads to similar training and validation performance, as shown in Figure 6(a). Details of the considered models are presented in Appendix A.3. The migration distance of the 90 percent of the (vertically integrated) CO_2 plume mass, $p_{90}[m_{\mathrm{CO_2}}]$, is one of the most important metrics for CCS applications, thus we use this variable to tailor our outlier detection technique. In Figure 6(b) we show that the standard deviation of the prediction of plume mass migration distance, $\sigma(p_{90}[m_{\mathrm{CO_2}}])$, has a strong positive correlation with the prediction error of this variable. While deploying the model (during inference), the error in $p_{90}[m_{\mathrm{CO_2}}]$ cannot be evaluated; therefore, this correlating behavior can be exploited to point out high error samples, i.e., the outliers. Figure 6(c) shows R^2 correlations of plume mass

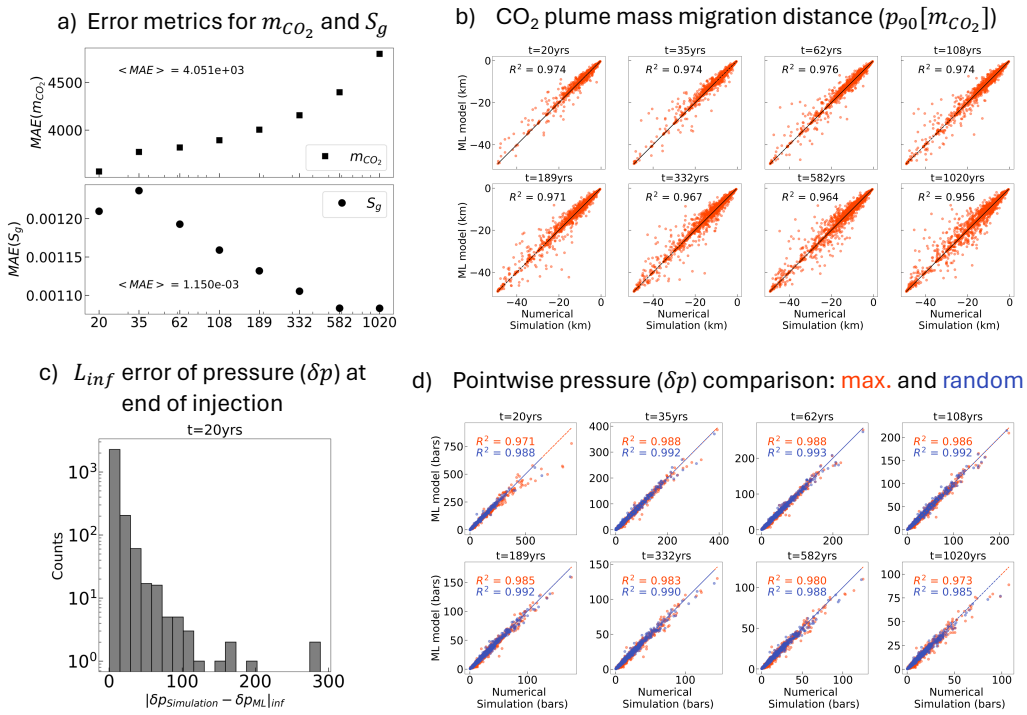

Figure 3. Error metrics (a) MAE for m_{CO_2} and S_g, with $< MAE >$ representing average in time (b) R^2 correlation plots of 90% plume mass migration distance. (c) Maximum pointwise error in δp (d) [red circles] R^2 correlation plots of δp at maximum pressure location corresponding to the test sample. [blue circles] Correlation plots δp for randomly selected locations within the active domain of the test sample.

migration distance of samples selected based on different $\sigma(p_{90}[m_{\text{CO}_2}])$ values. No cutoff represents the base case; increasing value of σ_{cutoff} represents a stricter removal of outlying samples and hence a reduction in the R^2 values. This methodology can be used to identify unreliable predictions from the surrogate model and run additional numerical simulations to enrich the training dataset as needed.

4.5. Model performance

Lastly, we discuss the computational speedup of our model compared to numerical simulation. Training data is generated using an in-house proprietary simulation package for subsurface flows. The hardware used to produce a training sample consists of 4 physical cores of Intel Cascade Lake CPUs. The two ML models, $ML(m_{\text{CO}_2}, S_g)$ and $ML(\delta p)$, are trained on 8 NVIDIA A100 GPUs and inferred on 1 NVIDIA A100 GPU. Typically, our models are trained for 100 epochs. Table 2 provides computational times for generating a sample of training data, training a model, and inference. Inference times on a AMD EPYC 7763 CPU, which is $\mathcal{O}(20)$ slower than GPU inferencing, is presented as a reference; in all our cases, we perform inference on GPU. The training and inference times for the ML model predicting δp are similar to $ML(m_{\text{CO}_2}, S_g)$. Hyperparameters and architectural details are presented in Appendix A.3. During GPU inference, using our ML model provides a speed-up of $\mathcal{O}(10^5)$ with respect to numerical simulations.

Table 2. Run times for training data generation, ML model (S_g, m_{CO_2}) training and inference. Time is reported in either GPU or CPU (Intel Cascade Lake‡, AMD EPYC 7763†) seconds.

Case type	Run time
Simulation (per sample)	3.037×10^4 CPU‡ sec
ML infer. (per sample)	1.780×10^{-1} GPU sec
ML infer. (per sample)	3.329 CPU† sec
ML train (per epoch)	7.205×10^3 GPU sec

A fair and exact comparison between numerical simulation and ML models based on runtime is not possible, as they were run on different hardware. However, we show a simple break-even analysis using *CPU hr* as computational unit for numerical simulation and *GPU hr* as unit for ML models. Total time for generating the entire training dataset accounts to $t_{gen} = 3.037 \times 10^4$ CPU sec $\times$ 12 923 samples $= 1.090 \times 10^5$ CPU hr. The total times for training and validation of a single ML model are, $t_{train} = 100$ epochs $\times$

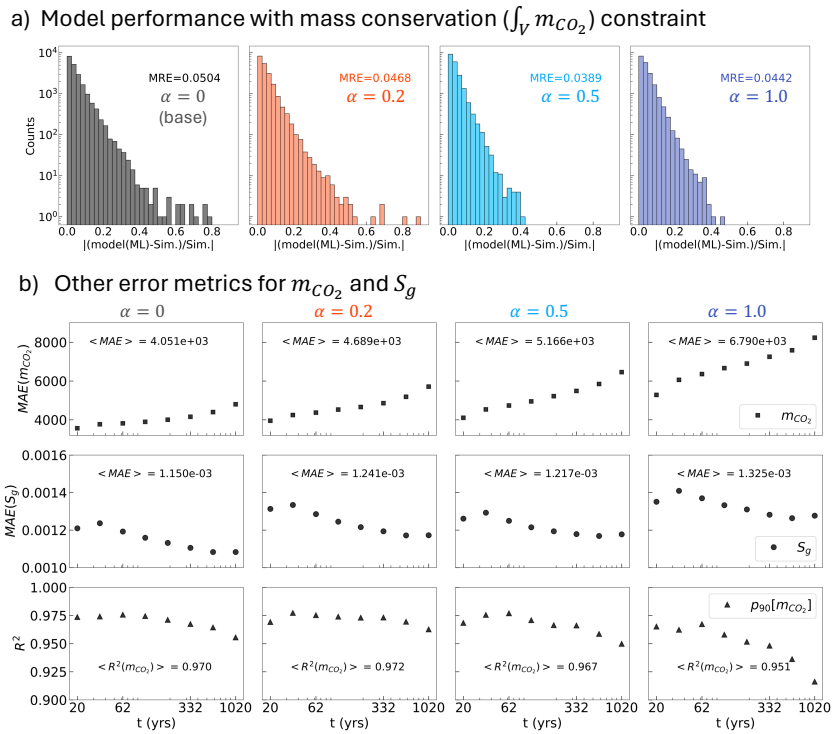

Figure 4. Performance metrics when a soft constraint on total mass in the system is imposed (a) Mean relative error (MRE) of total mass, $\int_V m_{\mathrm{CO}_2}$, over all time instances as weighing factor in the loss function (α) is varied. (b) MAE for m_{CO_2} and S_g along with R^2 of $p_{90}[m_{\mathrm{CO}_2}]$

7.205×10^3 GPU sec $=$ 200.138 GPU hr and $t_{valid} = 1.780 \times 10^{-1}$ GPU sec $\times$ 2593 samples $=$ 0.128 GPU hr respectively. Therefore total time spent building the two ML models is, $t_{ML-develop} = t_{gen} + 2(t_{train} + t_{valid}) = 1.094 \times 10^5$ hr. The factor of two is due to the fact that $ML(\delta p)$ is developed as a separate model, and both models require similar training and validation times.

In a typical screening setting, around $20,000$ Monte Carlo simulations are required for generating a diverse range of scenarios. The total computational time for developing the ML models and inferencing $20,000$ samples sums up to, $t_{ML-develop} + (20,000 \times 2 \times 1.780 \times 10^{-1}$ sec$) = 1.094 \times 10^5 + 1.978$ hr $= 1.094 \times 10^5$hr. Producing $20,000$ samples through numerical simulation requires $20,000 \times 3.037 \times 10^4$ sec $= 1.687 \times 10^5$ hr. Therefore, just one site assessment justifies training an ML model. The fully-trained model can be applied to numerous screening tasks, amplifying the benefits of our approach.

5. Summary and Conclusions

In this study, we have developed a model based on the Fourier Neural Operators (FNO) for real-time, high-resolution simulation of CO_2 plume migration. This model is trained on an extensive dataset derived from realistic subsurface parameters. During inference, we observe a speedup of $\mathcal{O}(10^5)$ over traditional numerical simulators of CO_2 flow fields with minimal reduction in accuracy. Along with fast surrogate models, we present and assess several physics based accuracy metrics that are relevant for assessing and monitoring CCS sites. In an attempt to improve the computational efficiency of our models, we explore training on coarser grid, that is created by downsampling a simulation created on a fine grid, and finally evaluating on a finer grid. Our experiments suggest that training on a coarse grid and evaluating on a finer grid has better accuracy than a case where the model is trained and evaluated on a coarse grid. This is a niche capability of a neural operator. Using uniform grids would further improve the accuracy of the models. Additionally, we propose several strategies – outlier detection, enforcing mass conservation – to enhance the reliability of the model's predictions, which is vital when evaluating ac-

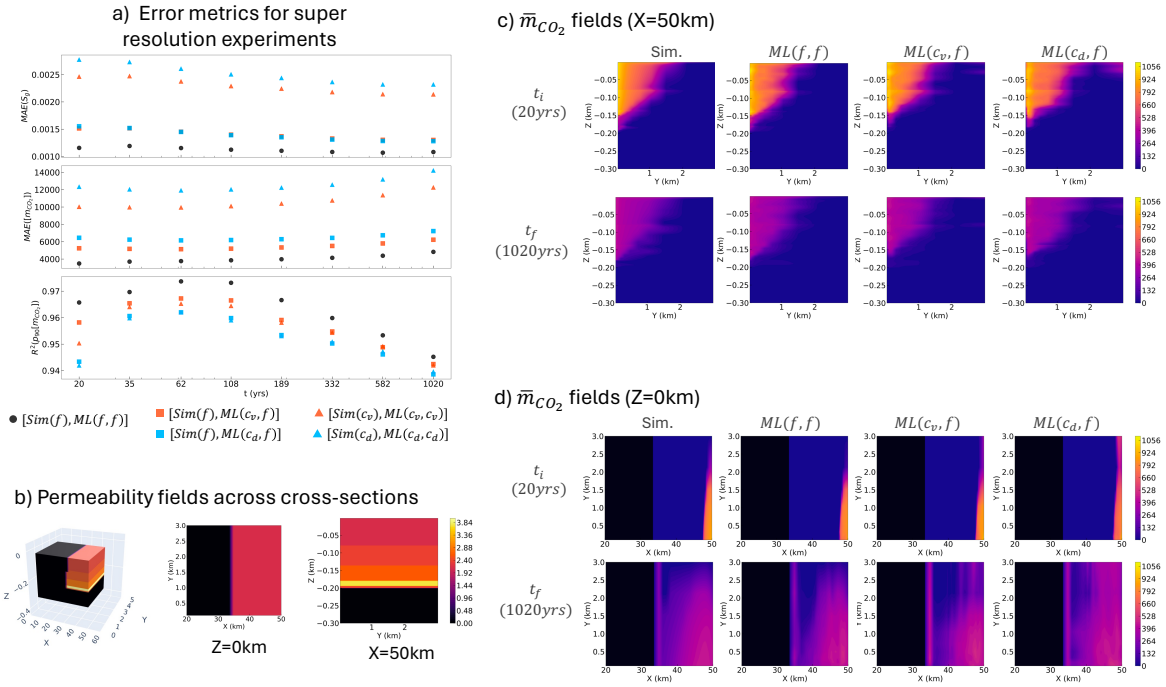

Figure 5. Super resolution experiments (a) Error metrics for different cases – base case (black circles), model trained on vol. avg coarse grid and evaluated on fine grid (orange square), model trained on vol. avg coarse grid and evaluated on vol. avg coarse grid (orange triangle), model trained on discretely down-sampled coarse grid and evaluated on fine grid (blue square), model trained on discretely down-sampled coarse grid and evaluated on discretely down-sampled coarse grid (blue triangle), (b, c, d) 2D visualization of fields along the Y axis.

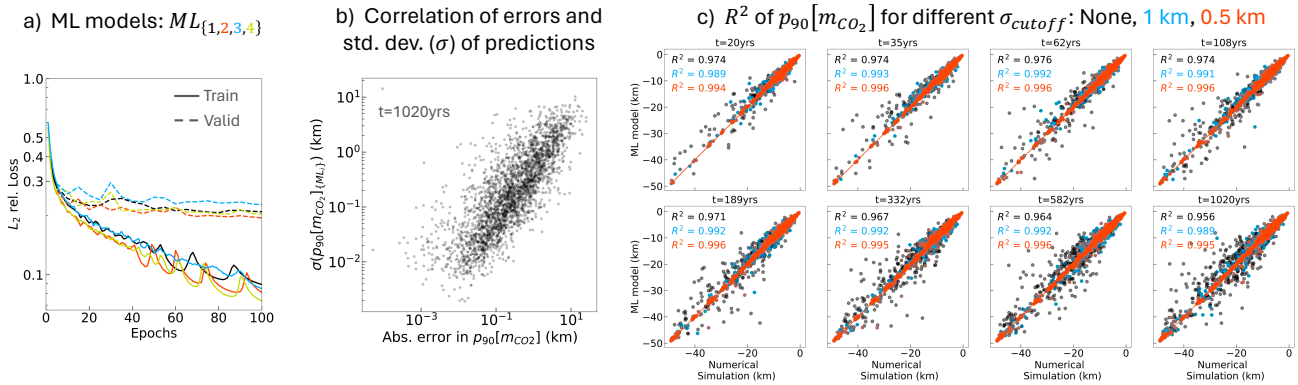

Figure 6. Detecting outliers based on ensemble models. (a) Training statistics of four ML models with slightly varying hyperparameters. All four models have similar performance. (b) Variation of standard deviation of 90% plume mass plume migration distance prediction from four ML models with absolute error in plume mass plume migration distance. (c) R^2 correlation of samples down-selected based on standard deviation of 90% plume mass plume migration distance prediction

tual geological sites. Our methodologies and strategies can be potentially extended with some domain-specific adaptations to other energy solutions such as geothermal reservoir modeling and hydrogen storage. Our work scales up scientific machine learning models to realistic 3D systems that more consistent with real-life subsurface aquifers/reservoirs, and builds a foundation for advanced screening tools for subsurface CCS applications.

Acknowledgements

The authors thank Shell International Exploration and Production Inc., Shell Information Technology International Inc., Shell India Markets Pvt. Ltd., and Shell Global Solutions International B.V. for permission to publish this work.

Impact Statement

The goal of this work is to advance the field of AI for Science. The ethical impacts and expected societal implications are those that are well established when advancing the field of Machine Learning, none which we feel must be specifically highlighted here.

References

Alpak, F. O. and Vink, J. C. A variable-switching method for mass-variable-based reservoir simulators. *SPE Journal*, 23(05):1469–1495, 2018.

Aziz, K. Petroleum reservoir simulation. *Applied Science Publishers*, 476, 1979.

Azizzadenesheli, K., Kovachki, N., Li, Z., Liu-Schiaffini, M., Kossaifi, J., and Anandkumar, A. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pp. 1–9, 2024.

de Coninck, H. and Benson, S. M. Carbon dioxide capture and storage: issues and prospects. *Annual review of environment and resources*, 39:243–270, 2014.

Ertekin, T., Abou-Kassem, J. H., and King, G. R. Basic applied reservoir simulation. *(No Title)*, 2001.

Falola, Y., Misra, S., and Nunez, A. C. Rapid high-fidelity forecasting for geological carbon storage using neural operator and transfer learning. In *Abu Dhabi International Petroleum Exhibition and Conference*, pp. D011S020R003. SPE, 2023.

Grady, T. J., Khan, R., Louboutin, M., Yin, Z., Witte, P. A., Chandra, R., Hewett, R. J., and Herrmann, F. J. Model-parallel fourier neural operators as learned surrogates for large-scale parametric pdes. *Computers & Geosciences*, 178:105402, 2023.

Jiang, Z., Zhu, M., Li, D., Li, Q., Yuan, Y. O., and Lu, L. Fourier-mionet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration. *arXiv preprint arXiv:2303.04778*, 2023.

Kelemen, P., Benson, S. M., Pilorgé, H., Psarras, P., and Wilcox, J. An overview of the status and challenges of co2 storage in minerals and geological formations. *Frontiers in Climate*, 1:482595, 2019.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020c.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Nghiem, L., Sammon, P., Grabenstetter, J., and Ohkuma, H. Modeling co2 storage in aquifers with a fully-coupled geochemical eos compositional simulator. In *SPE Improved Oil Recovery Conference?*, pp. SPE–89474. SPE, 2004.

NVIDIA Modulus Team. NVIDIA Modulus. URL https://github.com/NVIDIA/modulus/tree/main.

Pawar, S., Devarakota, P., Alpak, F. O., Snippe, J., and Hohl, D. Efficient subsurface carbon storage modeling with fourier neural operator. In *SEG International Exposition and Annual Meeting*, pp. SEG–2023. SEG, 2023.

Por, G., Boerrigter, P., Maas, J., and De Vries, A. A fractured reservoir simulator capable of modeling block-block interaction. In *SPE Annual Technical Conference and Exhibition?*, pp. SPE–19807. SPE, 1989.

Regtien, J., Por, G., van Stiphout, M., and van der Vlugt, F. Interactive reservoir simulation. In *SPE Reservoir Simulation Conference?*, pp. SPE–29146. SPE, 1995.

Rogelj, J. et al. Mitigation pathways compatible with 1.5 c in the context of sustainable development ed v masson-delmotte et al glob. warm. 15 c ipcc spec. *Rep. Impacts Glob. Warm. 15 C Pre-Ind. Levels Relat. Glob. Greenh. Gas Emiss. Pathw. Context Strength. Glob. Response Threat Clim. Change Sustain. Dev. Efforts Eradicate Poverty*, 2018.

Wei, L. and Saaf, F. Estimate co 2 storage capacity of the johansen formation: numerical investigations beyond the benchmarking exercise. *Computational Geosciences*, 13: 451–467, 2009.

Wen, G., Hay, C., and Benson, S. M. Ccsnet: a deep learning modeling suite for co2 storage. *Advances in Water Resources*, 155:104009, 2021.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Wen, G., Li, Z., Long, Q., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. Real-time high-resolution co 2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, 16 (4):1732–1741, 2023.

Witte, P. A., Redmond, W., Hewett, R. J., and Chandra, R. Industry-scale co2 flow simulations with model-parallel fourier neural operators. In *NeurIPS 2022 workshop tackling climate change with machine learning*, 2022.

Witte, P. A., Konuk, T., Skjetne, E., and Chandra, R. Fast co2 saturation simulations on large-scale geomodels with artificial intelligence-based wavelet neural operators. *International Journal of Greenhouse Gas Control*, 126: 103880, 2023.

A. Appendix

A.1. Input variables

Ranges of various input variables are shown in Table 3. All the reported values are scalars - permeability fields are generated using a proprietary algorithm scripted in our in-house flow simulator.

Table 3. Ranges of scalar parameters used for parameterizing CO_2 storage model and surrogate model development

Parameters	**Range**
Base permeability ($\log_{10}$, base 1mD)	(1, 3.30)
Number of geological layers	$(2, 20)_d$
Heterogeneity scale	(0, 1)
Reservoir height (m)	(15, 300)
Reservoir dip angle (degrees)	(0, 6)
Porosity	(0.05, 0.35)
Vertical-horizontal permeability ratio ($\log_{10}$)	(-6, 0)
Reservoir Pressure (bar)	(100, 300)
Reservoir Temperature (^oC)	(25, 135)

A.2. Architectural modifications

In NVIDIA Modulus, FNO models can be constructed by a simple function call providing configurations such as the number of FNO layers, number of modes and activation functions. NVIDIA Modulus being open-source in combination with its modular design enables additional customisation of the FNO architectures. Listing 1 provides pseudo code for customising the lift layer akin to the models used in this work.

Listing 1. Pseudocode for changing the lift layer of FNO to a linear transormation.

```
class CustomFNO4DEncoder(modulus.models.fno.FNO4DEncoder):
    def build_lift_network(self):
        self.lift_network = torch.nn.Sequential()
        self.lift_network.append(
            Linear(self.in_channels, self.fno_modes)
        )
```

Other methods used from NVIDIA Modulus are logging routines and the `DistributedManager`. The `DistributedManager` provides a singleton class for setting up the parallel environment by assigning available devices, carrying out optimisations like clearing device cache and automatically choosing the communication backend based on the availability of NCCL.

A.3. Model architecture and hyperparameters

In all base cases domain size is nx, ny, nz, nt = 214, 10, 200, 8 and hyperparameters are shown in Table 4.

It should be noted that the number of input dimensions are 10 – 6 field variables and 4 coordinates corresponding to x,y,z, and t. For super resolution experiments, dimensions are (nx, ny, nz, nt = 214, 5, 200, 8) and 2 modes are retained in the y direction. For outlier detection, the three additional models (in addition to the base case) are created by changing the following hyperparameters independently w.r.t the base case – `padding=2, latent_channels=48, fno_layers=8`.

Parameter	Value
Decoder	
layers	1
layer_size	128
FNO	
dimension	4
latent_channels	36
fno_layers	6
fno_modes	[10,5,10,5]
padding	0
Scheduler	
initial_lr	1.E-3
decay_rate	.95
decay_epochs	2

Table 4. Deep Learning Model Configuration

A.4. Visualization of various output variables

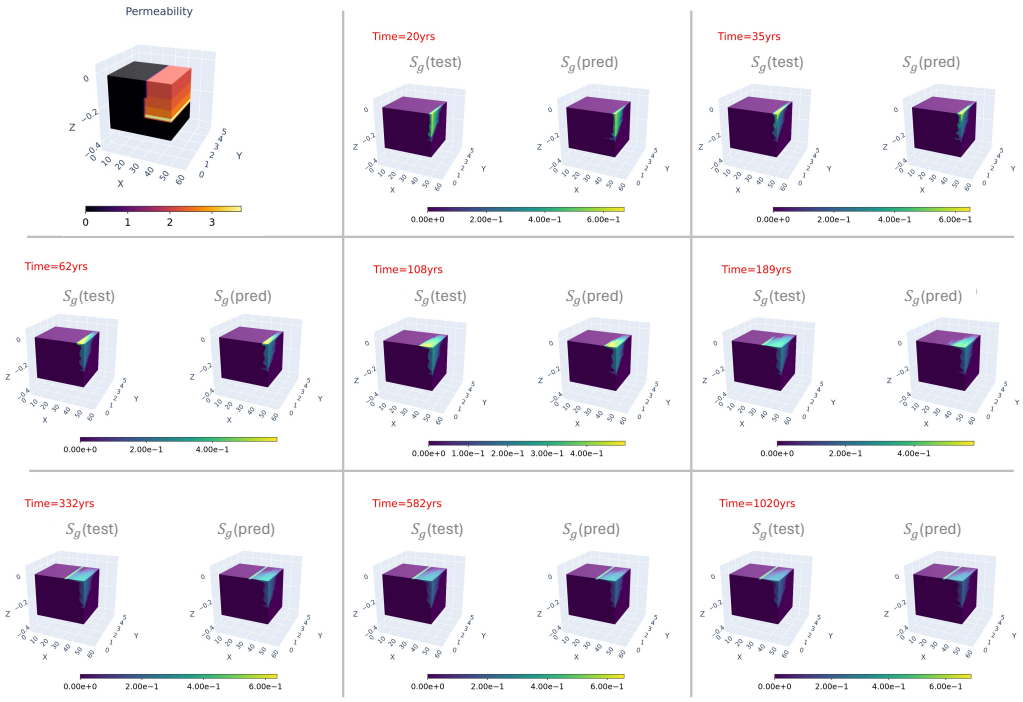

Figure 7. Comparison of S_g-test and S_g-pred of a specific sample at different time instances

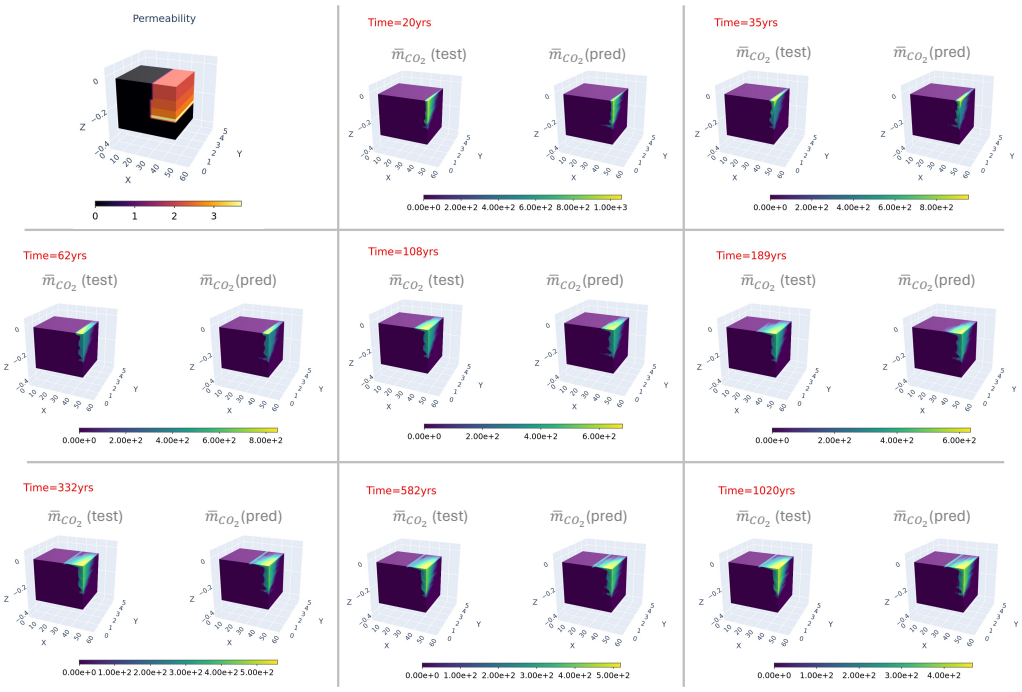

Figure 8. Comparison of $\overline{m}_{\mathrm{CO_2}}$-test and $\overline{m}_{\mathrm{CO_2}}$-pred of a specific sample at different time instances

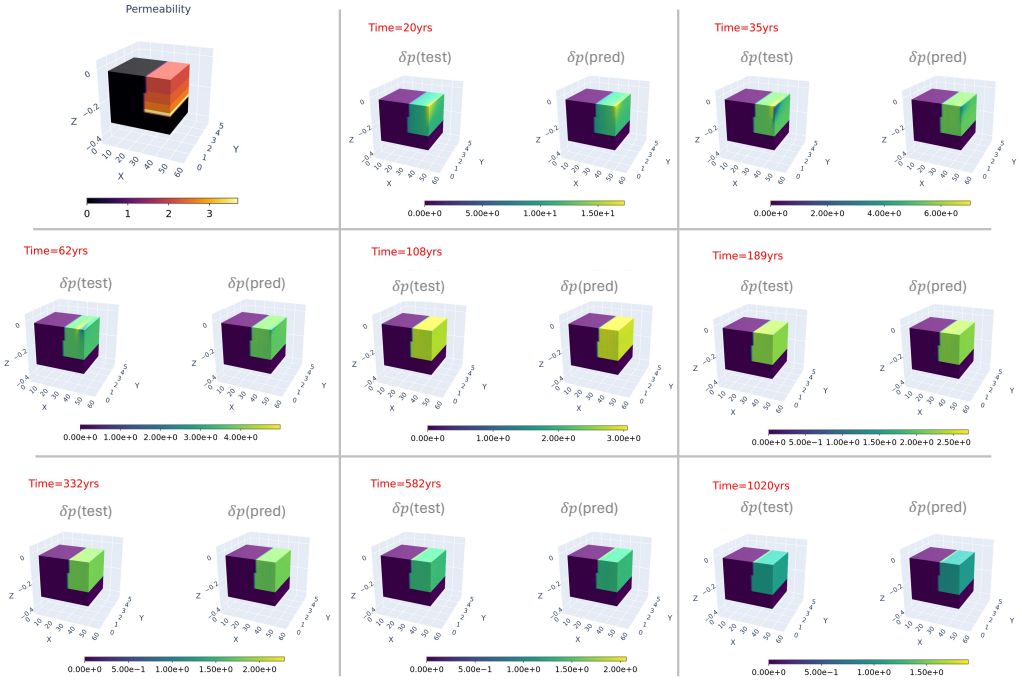

Figure 9. Comparison ofδp-test and δp-pred of a specific sample at different time instances

A.5. Mass Conservation

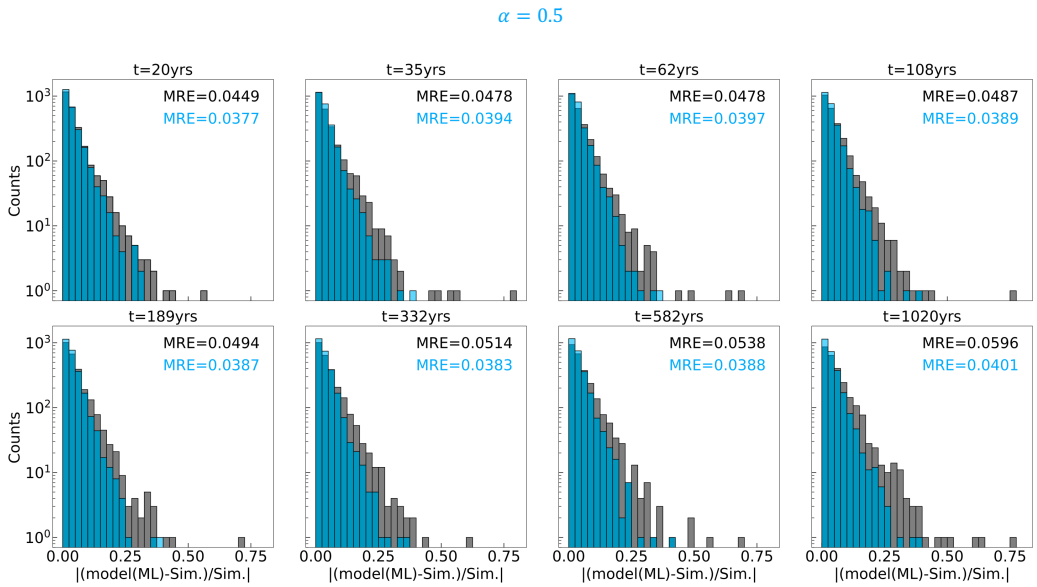

Figure 10. Mean relative error in total mass conservation at different times when $\alpha = 0.5$

A.6. Downsampling strategies

We choose to create our coarsened representation in two ways: (a) Volume averaged downsampling - The field variables are volume averaged (b) Discrete downsampling - every other point from the original grid is selected. A schematic of our methodology is shown in Figure 11. Note that in all our cases, downsampling is only performed in the y-direction. The grid (X, Y, Z) coordinates is always discretely picked - no volume averaging is performed.

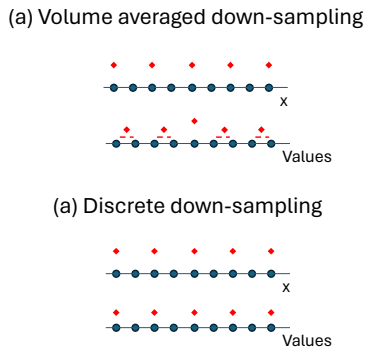

Figure 11. Different downsampling strategies