Physics-informed Localized Learning for Advection-Diffusion-Reaction Systems

Surya T. Sathujoda [1] **Soham M. Sheth** [2]

Abstract

The global push to advance Carbon Capture and Sequestration initiatives and green energy solutions, such as geothermal, have thrust new demands upon the current state-of-the-art subsurface fluid simulators. The requirement to be able to simulate a large order of reservoir states simultaneously, in a short period of time, has opened the door of opportunity for the application of machine learning techniques for surrogate modelling. We propose a novel physics-informed and boundary condition-aware Localized Learning method which extends the Embed-to-Control (E2C) and Embed-to-Control and Observe (E2CO) models to learn local representations of global state variables in an Advection-Diffusion Reaction system. Trained on reservoir simulation data, we show that our model is able to predict future states of the system, for a given set of controls, to a great deal of accuracy with only a fraction of the available information. It hence reduces training times significantly compared to the original E2C and E2CO models, lending to its benefit in application to optimal control problems.

1. Introduction

Subsurface fluid simulators play an important role in modelling various modern geological processes. They enable the efficient and sustainable utilization of geothermal energy by modelling heat and mass transfer processes; they play a vital role in modern Carbon Capture and Sequestration (CCS) initiatives by assessing optimal locations to inject CO_2 into geological formations, and they facilitate the efficient extraction of natural resources to reduce the risks of exploration. Developments in these areas have produced novel challenges to the production workflow of subsurface fluid modelling. The task of optimizing the location and volume of CO_2 injection into a porous medium, for instance, requires vast amounts of simulation data and compute power for each individual control state and time step of the system to determine the maximal storage case. This is especially problematic for applications in which we require real-time decision making. This has hence given birth to a new class of subsurface surrogate modelling techniques using machine learning.

The governing equation of subsurface flow, derived from the law of mass-conservation and Darcy's law, relates the fluid flow and flow potential gradients of a multi-phase porous-medium as given below

$$\nabla \cdot \left[\alpha \mathbf{k} \frac{k_{r,m}(S_m)}{\mu_m B_m} (\nabla p_m - \gamma_m \nabla z) \right] - \beta \frac{\partial}{\partial t} \left(\frac{\phi S_m}{B_m} \right) + \sum_w \frac{q^w_{sc,m}}{V_b} = 0$$

where $\mathbf{k}$ denotes the permeability tensor, k_r the relative permeability, μ the viscosity, B the formation volume factor, p the pressure, S the saturation, ϕ the porosity, t the time, γ the specific weight, q the source/sink terms, z the depth and V_b the bulk volume. The subscript sc represents standard conditions and α and β are unit field constants. More generally, this equation is an instance of an Advection-Diffusion-Reaction (ADR) partial differential equation (PDE) which describes the transport and transformation of scalar quantities (such as pressure and saturation in this case) in a medium due to advection, diffusion and reaction, as the name suggests. The ADR equation arises in many other areas of science and engineering such as environmental engineering, where it can be applied to help understand the spread of contaminants in groundwater, biomedical engineering, where it can be applied to model the transport of drugs in biological tissues, and in chemical engineering, where it can be used to help optimize reactor design and operation. Although our proposed model focuses on subsurface fluid dynamics, it easily generalises to all the above areas with the utilisation of the appropriate domain specific knowledge.

The above equation of subsurface flow describes a highly non-linear system that even state-of-the-art numerical solvers using Newton's method require significant compute power to converge to a solution. For this reason, various

[1]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK [2]SLB, Abingdon, UK. Correspondence to: Surya T. Sathujoda <sts40@cam.ac.uk>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

reduced-order-models (ROMs) have been developed which approximate the evolution of state variables in time to be linear. The most common such ROM is the Proper Orthogonal Decomposition Trajectory Piece-wise Linearization Reduced Order Model (POD-TPWL) (Cardoso & Durlofsky, 2010; He et al., 2011; He & Durlofsky, 2013; Rousset et al., 2014; TPW, 2015; He & Durlofsky, 2015; Jin & Durlofsky, 2018) which reduces the size of the system-state variables via the POD step before approximating their evolution linearly for each time step of the control. Although this method has played an important role in speeding up calculations, in recent years machine learning methods have been developed which, with a one time overhead training cost, promise an even faster inference of the next system state.

Early on, some general machine learning models were first applied to this problem by treating the initial field values at each point as the input to the model and the numerical solution to the PDE as the output for time-independent cases and the next step field values as the output for time dependent cases. These were purely data-driven approaches which were agnostic to the form or the physics of the underlying PDEs governing the system. Some of these early methods, which now serve as a benchmark for which to compare more advanced methods to, are: **NN**: a simple point-wise feed-forward neural network. **RBM**: the classical Reduced Basis Method (using POD basis). **FCN**: a state-of-the-art neural network architecture based on Fully Convolution Networks (Zhu & Zabaras, 2018). **PCANN**: an operator method using PCA as an auto-encoder on both the input and output data and interpolating the latent spaces with a neural network (Bhattacharya et al., 2021). For time dependent tasks, popular models in other time dependent regression tasks were applied such as: **ResNet**: a residual learning framework to ease the training of networks that are substantially deeper than those used previously (He et al., 2015). **U-Net**: A popular choice for image-to-image regression tasks consisting of four blocks with 2-d convolutions and deconvolutions (Ronneberger et al., 2015).

As developments continued, one of the main directions of success to model non-linear control dynamics came from the Embed-to-Control (**E2C**) model (Watter et al., 2015). The E2C model consists of a variational auto-encoder and a transitional block which learns to generate image trajectories from a latent space in which the dynamics are constrained to be locally linear. This work was subsequently modified by replacing the variational auto-encoder with an ordinary encoder-decoder structure and then applied to the problem of 2D reservoir surrogate modelling by (Jin et al., 2020). The results produced by this model were extremely accurate compared to previous methods and formed the bed rock for the expansion of the model to predict well outputs also for resource extraction in (Coutinho et al., 2021). The thus

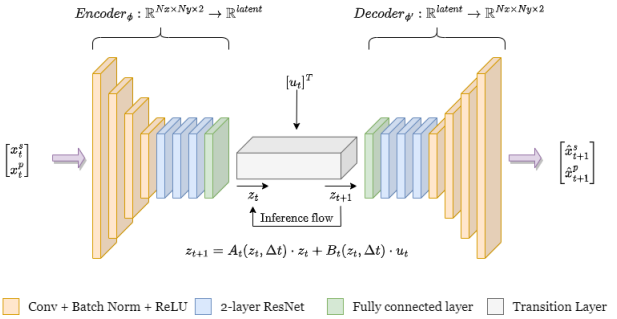


Figure 1. 2D architecture for Embed-to-control applied to the reservoir simulation problem where x_t^s and x_t^p represent the saturation and pressure values respectively and u_t the well controls.

proposed Embed to Control and Observe (**E2CO**) model added an additional parameterized network flow in the transitional block to predict well outputs such as flow rates at source/sink points. Both the E2C and the E2CO models were naturally expanded to 3D grids by (Atadeger et al., 2022) by remodelling the architecture to involve 3D convolution blocks and transposes in the encoder and decoder respectively to more reflect real-world systems in the area. The main draw back of these current models however is that, although they produce promising predictions for future time steps, the training times grow rapidly as grid sizes increase. Training time for a 3D case of 5×10^5 cells, which is on the lower end of a real-world case, takes upwards of 47 hours on a NVIDIA Tesla V100 GPU. Various new architectures have been proposed to tackle this problem, such as **Neural Operators** (Li et al., 2020b;a; 2021; 2022; Kovachki et al., 2023) and **DeepONet** (Lu et al., 2021), but in their current state, they do not rival the accuracy of convolution-based methods.

We hence propose Localized Learning for Embed to Control (**LL-E2C**) and Localized Learning for Embed to Control and Observe (**LL-E2CO**), a method that learns on a random set of sub-grids of the original full grid of simulation, using physics-informed losses, and reconstructs the next state using the locally learned model. The intuition behind this being that physics at a local level is constant everywhere when boundary conditions are accounted for. We show that we achieve similar levels of accuracy to E2C and E2CO with our model while drastically reducing the training times.

2. Preliminaries

The task that we apply our model to is that of a reservoir simulation. The reservoir is discretized into a regular grid and different geological structures are modelled by spatially varying but temporally static permeabilities across the grid. At each point in the grid we have a certain pressure and saturation value and at various points in the reservoir we

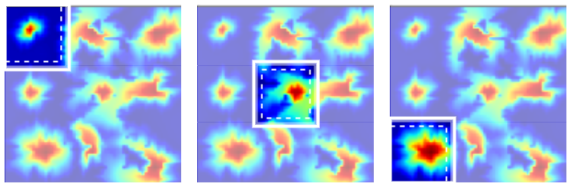


Figure 2. Three different cases of inducing boundary condition awareness in the model. The solid line indicates the sector which is used in training and inference and the cells inside the dotted lines indicate the part of the sector which are 'stitched' together to recreate the full next state. **Left.** The stitched sub-sector shares hard boundaries on the top and left with the actual grid, teaching the model that flow cannot continue in that direction. **Middle.** The stitched sub-sector shares no hard boundaries allowing the model to learn that fluid can flow out of the sub-sector in all directions. **Right.** The sub-sector shares the same hard boundaries on the left and bottom as the grid, imposing Dirichlet boundary conditions, and Von Neumann on the right and top.

have sources/sinks in the form of injection/production wells. We can control these wells to affect the source an sink terms in the governing equation which in turn aids an important application of reservoir simulation, that is the deduction of optimal control and well placement for a given subsurface which maximises well production and storage rates.

2.1. Problem Setting

Let a fluid dynamical system be discretized on a regular grid with dimensions $n = N_x \times N_y \times N_z$. The state of the system at time t is represented by $\mathbf{x}_t \in \mathcal{G}$, where $\mathcal{G} \subset \mathbb{R}^{n \times d_x}$. Given an applied control $\mathbf{u}_t \in \mathcal{U}$, where $\mathcal{U} \subset \mathbb{R}^{d_u}$, we aim to model the arbitrary, smooth, system dynamics function f in the time evolution equation $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ and the function g determining the well outputs $\mathbf{y}_{t+1} \in \mathcal{Y}$ where $\mathcal{Y} \subset \mathbb{R}^{d_u}$, given by $\mathbf{y}_{t+1} = \mathbf{g}(\mathbf{x_t}, \mathbf{u_t})$.

2.2. Data

Training and testing data is generated by a proprietary high-fidelity reservoir simulator (HFS). We consider two separate cases to evaluate our model, a 2D (60×60) case and a larger 3D ($60 \times 220 \times 40$) case. In the 2D case, each train/test sample consists of pressure $\mathbf{p}_t \in \mathbb{R}^n$, saturation $\mathbf{S}_t \in \mathbb{R}^n$ and control $\mathbf{u}_t$ which corresponds to the source/sink terms for $T = 24$ times steps. We also have well output values for the next time step $\mathbf{y}_{t+1}$ in the case of E2CO. The input to the model is constructed such that $\mathbf{x}_t = [\mathbf{p}_t, \mathbf{S}_t]$. Permeability and source/sink location information, which is static with time, are available where required. The same data is simulated for the 3D case but with $T = 20$. We generate 400 such samples and utilise a 3:1 split for training and testing.

3. Methodology

3.1. E2C Reduced Order Model

The E2C model architecture, figure 1, consists of a parameterized encoder $\phi_\theta : \mathcal{G} \to \mathcal{Z}$, where $\mathcal{Z} \in \mathbb{R}^{n_z}$, transition layer $\tau_\theta : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$ and decoder $\psi_\theta : \mathcal{Z} \to \mathcal{G}$. The composition of these three gives the overall model $\mathcal{M}_\theta = \{\psi \circ \tau \circ \phi : \mathcal{G} \to \mathcal{G}\}$. The encoder ϕ consists of 4 convolution layers followed by 3 2-layer ResNets and a final dense layer to map from the state space $\mathcal{G}$ to the locally linear latent space $\mathcal{Z}$. The decoder ψ conversely maps back from latent space to state space using a dense layer followed by 3 2-layer transposed ResNets and 4 transposed convolution layers. The transition layer τ in between aims to model the latent transition function f^{lat} of the dynamics given by $\mathbf{z}_{t+1} = f^{\text{lat}}(\mathbf{z}_t, \mathbf{u}_t)$. Given the state and action sequences $\mathbf{z}_{1:T} = \{\mathbf{z}_1, ..., \mathbf{z}_T\}$ and $\mathbf{u}_{1:T} = \{\mathbf{u}_1, ..., \mathbf{u}_T\}$, optimal controls which give rise to the trajectory $\mathbf{z}_{1:T}$ for a given f^{lat} can be determined using traditional optimal control algorithms. These algorithms approximate global non-linear dynamics with local linear dynamics and it can be shown (section 2.2 of Watter, M. et al. (Watter et al., 2015)) that for a reference trajectory $\bar{\mathbf{z}}_{1:T}$ and controls $\bar{\mathbf{u}}_{1:T}$, the system is linearized as

$$\mathbf{z}_{t+1} = \mathbf{A}(\bar{\mathbf{z}}_t)\mathbf{z}_t + \mathbf{B}(\bar{\mathbf{z}}_t)\mathbf{u}_t + \mathbf{o}(\bar{\mathbf{z}}_t),$$

where $\mathbf{A}(\bar{\mathbf{z}}_t) = \frac{\delta f^{\text{lat}}(\mathbf{z}_t, \mathbf{u}_t)}{\delta \bar{\mathbf{z}}_t}$, $\mathbf{B}(\bar{\mathbf{z}}_t) = \frac{\delta f^{\text{lat}}(\mathbf{z}_t, \mathbf{u}_t)}{\delta \bar{\mathbf{u}}_t}$ are local Jacobians and $\mathbf{o}(\bar{\mathbf{z}}_t)$ is an offset. This equation is the inspiration for the transition layer and the local Jacobians and offsets are parameterized as trainable weights in the model for calculating $\mathbf{z}_{t+1}$.

The loss function of the model is a composition of traditional auto-encoder loss functions and physics-informed loss functions as follows,

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{pred} + \theta_{trans} \times \mathcal{L}_{trans} + \theta_{flux} \times \mathcal{L}_{flux}$$

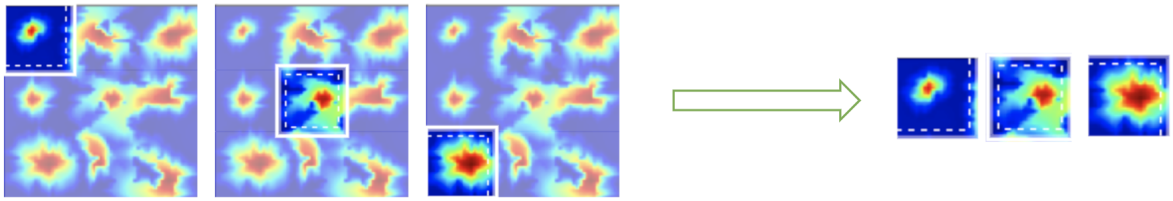
$$\mathcal{L}_{rec} = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2, \quad \mathcal{L}_{pred} = \|\mathbf{x}_t - \hat{\mathbf{x}}_{t+1}\|_2, \quad \mathcal{L}_{trans} = \|\mathbf{z}_{t+1} - \hat{\mathbf{z}}_{t+1}\|_2$$

where $\hat{\mathbf{x}}_t = \phi(\psi(\mathbf{x}_t))$ is the reconstruction of $\mathbf{x}_t$, $\hat{\mathbf{x}}_{t+1} = \mathcal{M}(\mathbf{x}_t, \mathbf{u}_t)$ is the next state prediction, $\hat{\mathbf{z}}_{t+1} = \phi(\tau(\mathbf{x}_t, \mathbf{u}_t))$ is the next state prediction in latent space and $\mathbf{z}_{t+1} = \phi(\mathbf{x}_{t+1})$ is the latent representation of true state $\mathbf{x}_{t+1}$. The physics-informed losses for the model are more specific to the reservoir simulation task, where $\mathcal{L}_{flux}$ aims to minimize the difference in pressure flux passing in and out of a given cell. The flux term is split up into two components, one to calculate the flux loss for reconstruction and one for prediction as follows,

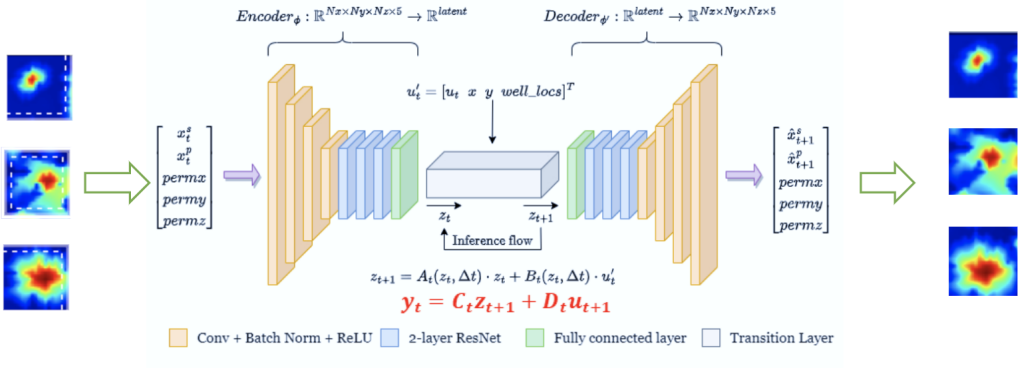
$$\mathcal{L}_{flux} = \|\mathbf{k}\mathcal{F}^{rec}\|_2 + \|\mathbf{k}\mathcal{F}^{pred}\|_2$$

$$\mathcal{F}^{rec} = [k_{ro}(\mathbf{S}_t^w)\Delta\mathbf{p}_t - k_{ro}(\hat{\mathbf{S}}_t^w)\Delta\hat{\mathbf{p}}_t] + [k_{rw}(\mathbf{S}_t^w)\Delta\mathbf{p}_t - k_{rw}(\hat{\mathbf{S}}_t^w)\Delta\hat{\mathbf{p}}_t]$$

1. Generate sector training data



2. Train LL-E2CO model using the sector data



3. Parallel predictions and assemble final solution

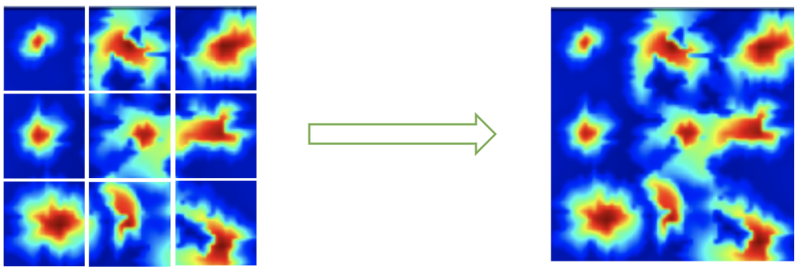


Figure 3. **Top.** Boundary condition-aware mechanism to train and infer on the sub-grid within the solid white line but stitch up using the sub-grid within the dotted white line. **Middle.** Model architecture of the 3D LL-E2C with the equation in red showing the additional flow for LL-E2CO model. **Bottom.** Representative next state prediction using individual stitching of sectors to reconstruct the full image.

$$\mathcal{F}^{pred} = [k_{ro}(\mathbf{S}^w_{t+1})\Delta\mathbf{p}_{t+1} - k_{ro}(\hat{\mathbf{S}}^w_{t+1})\Delta\hat{\mathbf{p}}_{t+1}] + [k_{rw}(\mathbf{S}^w_{t+1})\Delta\mathbf{p}_{t+1} - k_{rw}(\hat{\mathbf{S}}^w_{t+1})\Delta\hat{\mathbf{p}}_{t+1}]$$

where $\mathbf{k}$ is the permeability tensor, k_{ro} and k_{rw} are relative permeability fields of the respective components given the saturation field $\mathbf{S}_t$ and $\Delta\mathbf{p}$ represents pressure drop across adjacent grid cells. Following conservation laws, we require the $\mathcal{L}_{flux}$ to be minimal in order for the model to represent a physical system.

3.2. E2CO Reduced Order Model

The E2CO model expands on this by modifying the transition layer to prediction well outputs $\mathbf{y}_{t+1}$. It does so by approximating global non-linear dynamics with local linear dynamics similar to the latent representation $\mathbf{z}_t$. Here we parametrize two more local Jacobians $\mathbf{C}(\bar{\mathbf{z}}_t)$ and $\mathbf{D}(\bar{\mathbf{z}}_t)$ and calculate the well outputs as following

$$\hat{\mathbf{y}}_{t+1} = \mathbf{C}(\bar{\mathbf{z}}_t)\hat{\mathbf{z}}_{t+1} + \mathbf{D}(\bar{\mathbf{z}}_t)\mathbf{u}_t.$$

We thus introduce an additional loss term to train for the well outputs simultaneously given below.

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{pred} + \theta_{trans} \times \mathcal{L}_{trans} + \theta_{flux} \times \mathcal{L}_{flux} + \theta_{well} \times \mathcal{L}_{well}$$

$$\mathcal{L}_{well} = \|\mathbf{y}_{t+1} - \hat{\mathbf{y}}_{t+1}\|_2,$$

where $\hat{\mathbf{y}}_{t+1}$ is the predicted well output for time $t+1$ and $\mathbf{y}_{t+1}$ is the actual well output. Notice here that we are adding additional complexity to the model by trying to predict another quantity so we expect to have a deterioration in accuracy when trying to predict $\mathbf{x}_t$. Hence, we use E2CO only when we explicitly require well outputs and employ E2C otherwise for comparing results.

3.3. Localized Learning for E2C(O)

The E2C reduced order model, when scaled to larger grids, was found to grow significantly in training time. For a $60 \times 220 \times 40$ grid, training time exceeds 47.2 hours on an NVIDIA Tesla V100 GPU. To tackle this problem we propose Localized Learning for Embed to Control (LL-E2C) (see figure 3), a method where, instead of training on a full grid, we train on a random subset of structured sub-grids (sectors).

The initial formulation is as follows. We chose at random input sectors $\mathbf{s}_t$ from $\mathcal{S} = \{\mathbf{s} : \mathbf{s} \in \mathbb{R}^{n_s} \supset \mathcal{G},\ n_s < n\}$ and train the model to predict sector state $\mathbf{s}_{t+1}$. At inference time, we pass the elements of the set of all non-overlapping sectors which are required to fully reconstruct the grid for state $\mathbf{s}_t$, $\mathcal{I} = \{\mathbf{s}_t^i : ||_i^{n/n_s} \mathbf{s}_t^i = \mathbf{x}_t,\ \cap_i^{n/n_s} \mathbf{s}_t^i = \emptyset\}$, and stitch up the predicted outputs $\mathbf{s}_{t+1}^i$ to produce the full next state image $\mathbf{x}_{t+1}$.

The key intuition behind sector training is that the underlying physics governing the time evolution of the subsurface properties is constant. This implies that whichever part of the grid we train on, the physics the neural network learns to emulate will remain the same (while respecting boundary conditions), hence sector-wise training loses no generality in solution.

In reality however, when sectors are chosen for training, we are artificially imposing Dirichlet boundary conditions at the edges where they might otherwise be Von Neumann. For example, when we train using a sector taken from the middle of a grid, all the edges of the sector exhibit Von Neumann boundary conditions but there is no way that the model would be able to capture this without further information. Furthermore, when a sector is chosen from a corner or side, the boundary conditions are a combination of both Dirichlet and Von Neumann. To tackle this problem, we employ a mechanism in which we train on sectors of size $n_s + n_p$ (where $n_p < n_s$), infer on sectors of size $n_s + n_p$ but only use sub-sectors of size n_s to stitch up to the final $\mathbf{x}_{t+1}$ state, as shown in figure 2. The location of the sub-sector is chosen such that its edges line up with the edges of the full image when the inference sector is around the periphery (to capture Dirichlet boundary conditions) and its edges are within the edge of the inference sector when it is taken from the middle of the image (to capture Von Neumann boundary conditions by allowing there to be 'flow' out of the sub-sector). We hence modify our formulation to take as inputs expanded sectors from $\tilde{\mathcal{S}} = \{\tilde{\mathbf{s}} : \tilde{\mathbf{s}} \in \mathbb{R}^{n_s+n_p}\}$, such that $\mathcal{S} \subset \tilde{\mathcal{S}}$ and infer using the set $\tilde{\mathcal{I}} = \{\tilde{\mathbf{s}}_t^i : ||_i^{n/n_s} \mathbf{s}_t^i = \mathbf{x}_t,\ \cap_i^{n/n_s} \mathbf{s}_t^i = \emptyset\}$, where the mapping between $\tilde{\mathbf{s}}_t^i$ and $\mathbf{s}_t^i$ is given by the description above.

Another issue which arises from training on sectors is that, although the underlying physics is the same in each sector, the relative locations of the applied controls vary depending on where a sector is extracted from. To tackle this, we add positional encoding by appending the position of the extracted sector and the relative control locations (with the locations of controls outside the sector zeroed out) to the control vector $\mathbf{u}_t$. Further to this, permeability information for each cell is passed in as separate channels to provide flow information, aiding the greatly reduced information available to the model.

For the case of predicting well outputs, we extend the above technique to E2CO and propose Localized Learning for Embed to Control and Observe (LL-E2CO), seen in figure 3. Here we make the same modification to the transition layer as the original E2CO model but now, since we are training on sectors, we need to take into account the fact that each sector only has information about well controls within its boundaries and does not have access to information outside its boundaries. As such, when passing controls $\mathbf{u}_t$ into the transition layer to calculate $\mathbf{y}_{t+1}$, we zero out the control values of all wells not pertaining to the sector in question. This also acts as an additional proxy for positional encoding of the sector as the model should, in theory, learn the position of the sector based on which well controls are zeroed out.

4. Results

Testing was conducted for the 2D and 3D cases detailed earlier and the respective results are compared to 'ground truth' simulation data. In the 2D 60×60 case, empirically, we found that sector sizes of $(20+4) \times (20+4)$ with an inference size of 20×20 produced the best results for the LL-E2C model. There was a slight improvement in training times between the E2C and LL-E2C models but we expect to see much a larger difference as we scale up to real world grid sizes. The saturation scalar values for the 2D case are shown in figure 4. Looking at the errors we see that they mainly peak around the wave fronts of the diffusion process. This can be attributed to the fact that this is where we see the most non-linearity in the system and hence the linearization process taking place in the transition layer does not approximate as well.

For the 3D $60 \times 220 \times 40$ case, which is much closer to the real world case, with sector sizes of $(20+4) \times (60+8) \times (20+4)$ and an inference size of $20 \times 60 \times 20$, we present a

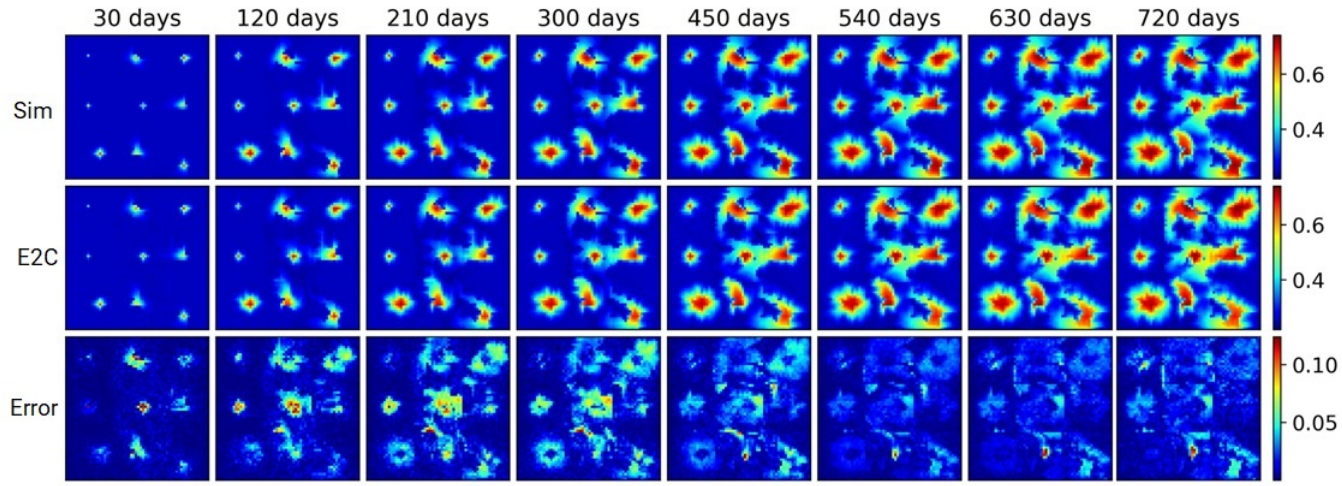


Figure 4. Saturation values predicted by LL-E2C for a single 2D test case. Each time step is 30 days.

Table 1. Training time results in hours

Method	Datasets			
	without Physics-informed Loss		with Physics-informed Loss	
	2D	3D	2D	3D
E2C	**0.4**	8.3	0.7	47.2
LL-E2C	0.5	**0.9**	**0.6**	**2.6**
E2CO	-	9.1	-	50
LL-E2CO	-	0.9	-	2.8

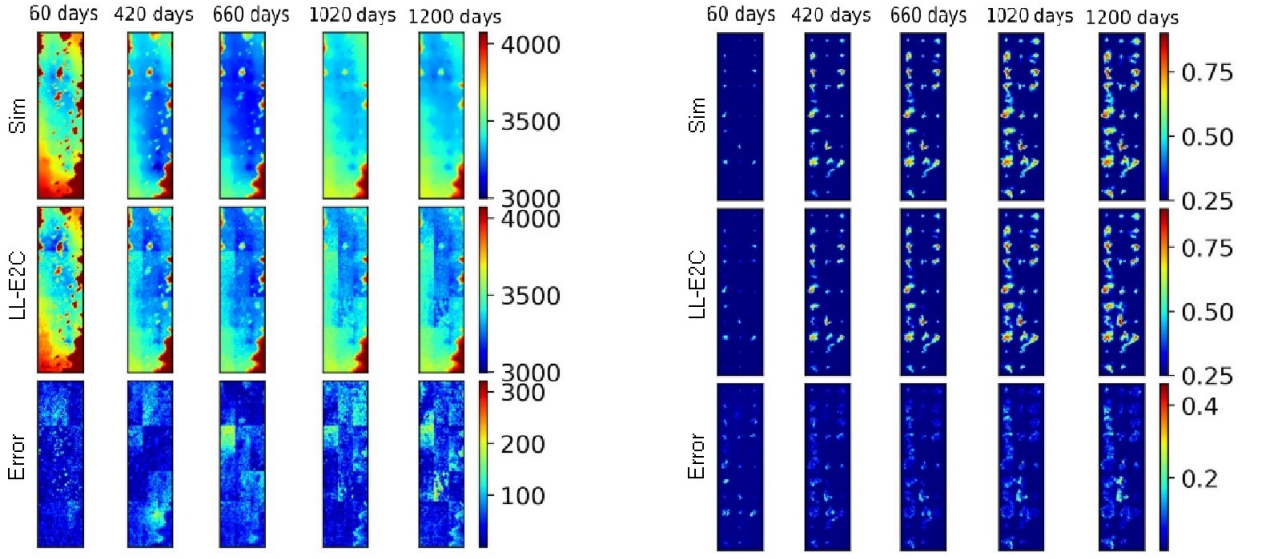


Figure 5. The left plot shows the pressure values from the simulator for a specific layer in the z direction, the predicted pressure values of the LL-E2C model and the absolute error between the two. The right plot shows the same for saturation values. Each time step is 60 days.

significant speed up time in training from 47.2 hours to 2.6 hours between the E2C and LL-E2C models and 50 hours to 2.8 hours between the E2CO and LL-E2CO models respectively as seen in table 1. In table 1, we also see the training times of the models with and without the use of physics-informed losses. Noting that we see a significant increase in

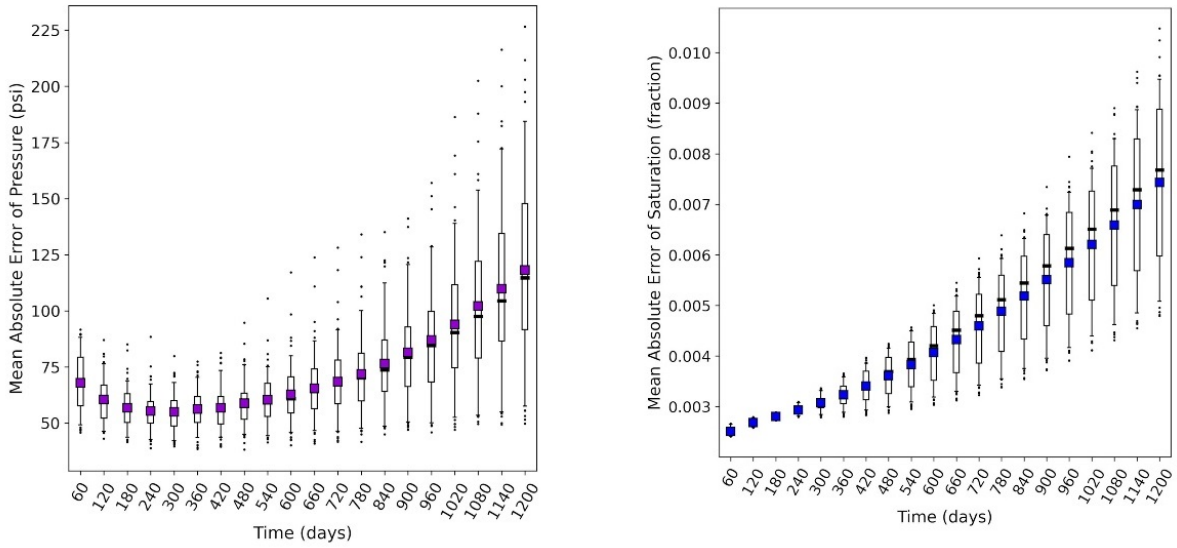


Figure 6. Mean absolute errors of pressure and saturation predictions for the LL-E2C model. Each dot corresponds to an individual test case with the bars signifying 1σ confidence intervals. These values correspond to the 3D case with each time step of 60 days.

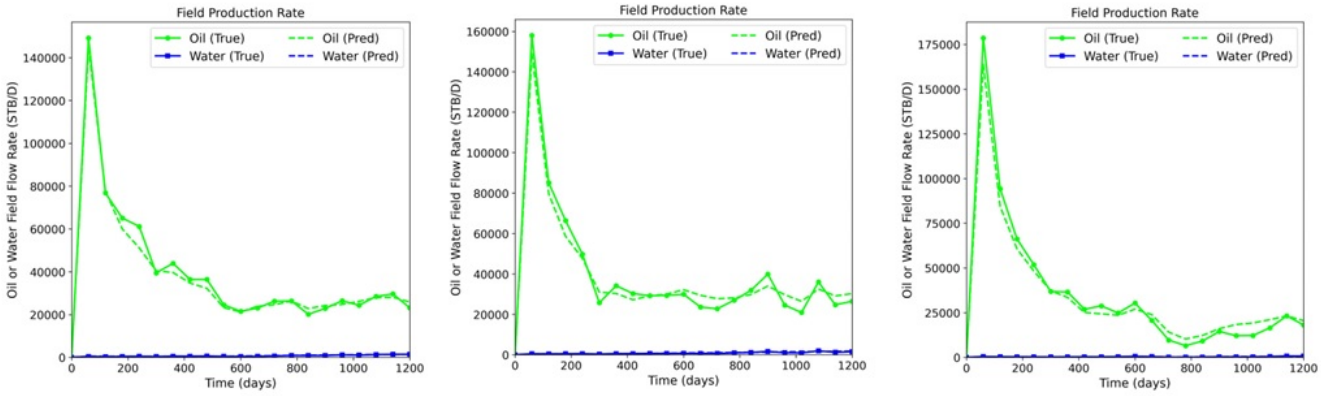


Figure 7. Field production rates of oil and water for 3 different test cases for the LL-E2CO model. The values correspond to the 3D case with each time step corresponding to 60 days.

accuracy with the inclusion of physics-informed losses, we can conclude that the LL-E2C(O) models are much more powerful for practical use as their training times rival even the E2C(O) models without their physics-informed losses incorporated.

Figure 5 shows the pressure and saturation predictions for 20 time steps using the LL-E2C model for a slice of the grid in the z direction. By analysing the test errors, we notice that saturation predictions appear to be more accurate than pressure predictions. This is likely due to the fact that pressure is a more global state variable and hence Localized Learning naturally finds it more difficult to model. A more thorough analysis of these errors were carried out and is presented in figure 6. The model was trained and run for all the different test cases and the mean absolute errors in pressure and saturation values are plotted. We observe that the absolute errors in predictions are well within the acceptable range for the purposes of reservoir simulation surrogate modelling, with mean average absolute errors over all test cases ranging between 50 psi to 125 psi for pressure and 0.002 to 0.007 fraction for saturation. We also observe that, although mean average errors increase as we predict further into the future, which is expected due to compounding errors, the model still produces much better results than can be reasonably expected in latter predictions. We attribute this to the fact that the system exhibits maximal non-linearity early on in the time evolution and as the wave fronts of pressure and saturation diffuse out, the transition layer becomes a better approximation of the locally linear dynamics. This leads to a better performance of the LL-E2C

model.

When testing the LL-E2CO model, we are mainly concerned with well outputs. As such, in this specific study, we look at the well production rates of oil and water in a reservoir under different well controls. The same analysis can be applied for the CO_2 storage task in CCS processes. The model predicts the outputs of each individual well in the system for both the 2D and 3D cases and in figure 7 we present the field production rates of the 3D case. Field rates are the aggregation of all the well production rates and are a good indication of the accuracy of the model. From the figure we see that the predicted production rates of oil are very close to the actual values produced by the simulator for the 3 cases. At all time steps, and even at later time steps when prediction are expected to diverge more, the values fall within 5% error. This is likely due to the same reasoning mentioned previously for LL-E2C.

A major advantage of machine learning models over traditional numerical solvers is their ability to parallelize inference by making batchwise computations. We are able to pass multiple test cases simultaneously into our model (which is what we would like to do when trying to find optimal control solutions over a large search space) and extract individual predictions where necessary. Thus, comparing inference times of the proposed LL-E2C(O) with the computation time of HFS, we observe a speed up of the order 10^4 which brings to light the true strength of machine learning models in this task.

5. Conclusion

In this work, we motivate the application of machine learning to subsurface fluid modelling and propose a novel method called Localized Learning to predict future states of an advection-diffusion-reaction system, with only a fraction of the available data. Our method reduces training times by over an order of magnitude for typical 3D cases, while also providing comparable accuracy results for all time steps. In the future, we hope to expand this work to unstructured grids and meshes.

6. Acknowledgements

The authors would like to thank SLB management for permission to publish this work. This work was done while Surya T. Sathujoda was a research intern at SLB.

References

Retraining Criteria for TPWL/POD Surrogate Based Waterflodding Optimization, volume Day 3 Wed, February 25, 2015 of *SPE Reservoir Simulation Conference*, 02 2015. doi: SPE-173252-MS. D031S011R007.

Atadeger, A., Sheth, S., Vera, G., Banerjee, R., and Onur, M. Deep learning-based proxy models to simulate subsurface flow of three-dimensional reservoir systems. 2022(1):1–32, 2022. ISSN 2214-4609. doi: https://doi.org/10.3997/2214-4609.202244049.

Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. Model reduction and neural networks for parametric pdes, 2021.

Cardoso, M. A. and Durlofsky, L. J. Linearized reduced-order models for subsurface flow simulation. *J. Comput. Phys.*, 229(3):681–700, 2 2010. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.10.004.

Coutinho, E. J. R., Dall'Aqua, M., and Gildin, E. Physics-aware deep-learning-based proxy reservoir simulation model equipped with state and well output prediction. *Frontiers in Applied Mathematics and Statistics*, 7, 2021. ISSN 2297-4687. doi: 10.3389/fams.2021.651178.

He, J. and Durlofsky, L. Reduced-order modeling for compositional simulation using trajectory piecewise linearization. *SPE Journal*, 19, 02 2013. doi: 10.2118/163634-MS.

He, J. and Durlofsky, L. J. Constraint reduction procedures for reduced-order subsurface flow models based on pod–tpwl. *International Journal for Numerical Methods in Engineering*, 103(1):1–30, 2015. doi: https://doi.org/10.1002/nme.4874.

He, J., Sætrom, J., and Durlofsky, L. J. Enhanced linearized reduced-order models for subsurface flow simulation. *J. Comput. Phys.*, 230(23):8313–8341, 9 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2011.06.007.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Jin, Z. L. and Durlofsky, L. J. Reduced-order modeling of co2 storage operations. *International Journal of Greenhouse Gas Control*, 68:49–67, 2018. ISSN 1750-5836. doi: https://doi.org/10.1016/j.ijggc.2017.08.017.

Jin, Z. L., Liu, Y., and Durlofsky, L. J. Deep-learning-based surrogate model for reservoir simulation with time-varying well controls. *Journal of Petroleum Science and Engineering*, 192:107273, 2020. ISSN 0920-4105. doi: https://doi.org/10.1016/j.petrol.2020.107273.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces, 2023.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations, 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, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations, 2021.

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, 2022.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. doi: 10.1038/s42256-021-00302-5.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015.

Rousset, M. A. H., Huang, C. K., Klie, H., and Durlofsky, L. J. Reduced-order modeling for thermal recovery processes. *Computational Geosciences*, 18(3): 401–415, 8 2014. ISSN 1573-1499. doi: 10.1007/s10596-013-9369-8.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2746–2754, Cambridge, MA, USA, 2015. MIT Press.

Zhu, Y. and Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.04.018.