

Nonlinearly Coupled Permafrost Dynamics From Graph Neural Networks

Dunyu Liu

dliu@ig.utexas.edu

The University of Texas at Austin

Kehua You

The University of Texas at Austin

Pengli Ren

The University of Texas at Austin

Hewei Tang

The University of Texas at Austin

Research Article

Keywords: subsea permafrost, graph neural networks, surrogate model, machine learning emulator, thermo-hydro-chemical coupling, multiphysics

Posted Date: September 17th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10910110/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Nonlinearly Coupled Permafrost Dynamics From Graph Neural Networks

Dunyu Liu^{1*}, Kehua You¹, Pengli Ren^{2,3}, Hewei Tang^{2,3}

^{1*}Institute for Geophysics, Jackson School of Geosciences, The University of Texas at Austin, Austin, TX, USA.

²Hildebrand Department of Petroleum and Geosystems Engineering, The University of Texas at Austin, Austin, TX, USA.

³The Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA.

*Corresponding author(s). E-mail(s): dliu@ig.utexas.edu;

Abstract

Subsea permafrost beneath Arctic continental shelves evolves under coupled water and heat flow, salt transport, and ice-water phase change forced by Pleistocene sea-level and surface-temperature variations. Physics-based simulation resolves these dynamics but is too expensive for ensemble studies. We present PermaCycleGNS, a Graph Network-based Simulator (GNS) that autoregressively predicts the joint evolution of sub-seafloor pore pressure, temperature, salinity, and ice saturation over the past 400,000 years. Trained on 661 trajectories from a five physical parameter space, it reaches normalized mean squared errors of 10^{-5} – 10^{-4} on pressure, temperature, and salinity and 6.8×10^{-4} on ice saturation on held-out tests; trained on centroid-proximal cases and tested at parameter-space corners, it reaches 2.3 – 4.3×10^{-5} on pressure, temperature, and salinity and 9.0×10^{-4} on ice saturation. A multistep backpropagation-through-time curriculum lowers ice-saturation normalized MSE from 3.4×10^{-3} (single-step) to 6.8×10^{-4} . Rollout is 8,800–200,000 $\times$ faster than the physics-based simulator, enabling sensitivity analyses, uncertainty quantification, and inversion for permafrost dynamics.

Keywords: subsea permafrost, graph neural networks, surrogate model, machine learning emulator, thermo-hydro-chemical coupling, multiphysics

1 Introduction

Subsea permafrost beneath Arctic continental shelves is a persistent legacy of Pleistocene glacial-interglacial cycles, when lowered sea level exposed the seafloor to Arctic air temperatures and froze hundreds of meters of sediment. Thawing of subsea permafrost could destabilize the shelf sediment, threaten coastal and seafloor infrastructure, and induce methane emission into the seawater [1–3]. Modeling the dynamic evolution of the subsea permafrost requires solving coupled water flow, heat and salt transport, and ice-water phase transition on time scales of 10^5 years. With sub-kiloyear resolution, this means 10^3 - 10^4 time steps per simulation, with each step requiring an implicit, computationally expensive Newton-Raphson solver. As a result, a single one-dimensional (1-D) simulation using the state-of-the-art physics-based numerical simulator (PBNS) PermaCycle may take hours of CPU time [4, 5]. This computational challenge makes parameter sweeps for sensitivity studies, uncertainty quantification, and inverse problems prohibitive.

Surrogate models offer a way around this cost. Recently, machine learning (ML) surrogates based on deep neural networks have started to be used to emulate a variety of dynamic physical systems. Among these, the Graph Network-based Simulator (GNS) has emerged as a promising tool to capture the spatiotemporal evolution of complex, high-dimensional physical systems by emulating all intermediate states autoregressively (e.g., [6–9]). GNS shows strong generalization across initial conditions, parameter-space extrapolation, and model domain sizes, and appears to be general to the physics underlying the dynamic system. They have been applied to particle granular flow [10, 11], mesh-based fluid and solid mechanics [7], weather [8], and most recently earthquake dynamic rupture [9]. Compared to direct end-to-end mappings from inputs to certain outputs, GNS exposes intermediate states and inherits inductive biases, i.e., locality, from the message-passing mechanism. This allows GNS to enable generalization to system sizes, parameter regions, and distribution complexities outside the training set, as recently demonstrated by Liu and Becker [9] for earthquake rupture dynamics.

Here, we develop a GNS for subsea permafrost dynamics, adapting the architecture of Liu and Becker [9] from a 2-D fault interface for slip rate evolutions, to a 1-D vertical column with joint pore pressure, temperature, pore water salinity, and ice saturation evolutions. The surrogate takes as input the initial state, the five global controlling parameters of each scenario, and predefined time-dependent top-boundary state; the latter is enforced at the surface node during rollout so that only the interior is predicted by the network. Our goal is to train the surrogate well to generalize across unseen parameter combinations, comparing a random training/testing split (interpolation) to a distance-from-centroid split (corner extrapolation to extremes). We use per-variable normalized mean squared error (MSE) to quantify the performance of GNS predictions and report computational speedup.

The paper is organized as follows. Section 2 describes the high-fidelity PBNS (PermaCycle), the dataset, and the GNS architecture. Section 3 presents the per-variable and ensemble accuracy of the GNS, the interpolation vs extrapolation comparison, computational performance, and sensitivity analyses of hyperparameters. Section 4

discusses what the surrogate appears to learn and where its limitations, and Section 5 for conclusions.

2 Methods

2.1 The High Fidelity PBNS: PermaCycle

PermaCycle is a MATLAB-based numerical simulator for predicting the dynamic evolution of subsea permafrost along the Arctic continental shelves over the past glacial-interglacial cycles [5]. This model simulates the coupled water flow, heat and salt transport, and ice-water phase transition from seafloor/ground surface to 1000 m sediment depth. It uses a 1-D finite-volume approach to solve the mass conservation equations for water and salt, and the energy conservation equation. Primary variables include pore-water pressure p_w , temperature T , and salt mass fraction c_l ; saturations for water (s_w) and ice (s_i) are calculated from a predefined soil water freezing curve using the predicted temperature and salinity. Time integration uses a backward Euler scheme with adaptive timestep size dt and Newton-Raphson nonlinear iterations. A representative simulation discretizes the 1000 m sediment column into 50 equally sized cells and runs from 400 kyr before present ($t = 0$) to 1 kyr into the future ($t = 401$ kyr), with output saved at 1 kyr intervals, which gives 402 states per case (Figure 1A). Boundary conditions are specified at the top node as Dirichlet for pressure, temperature, and salinity, and at the bottom node with a constant geothermal heat flux q_t , a per-case parameter. The two driving time series, sea-level curve $\Delta sl(t)$ from a Pleistocene reconstruction [12] and Arctic ground surface temperature $T_{\text{surface}}(t)$, are global and identical across the ensemble.

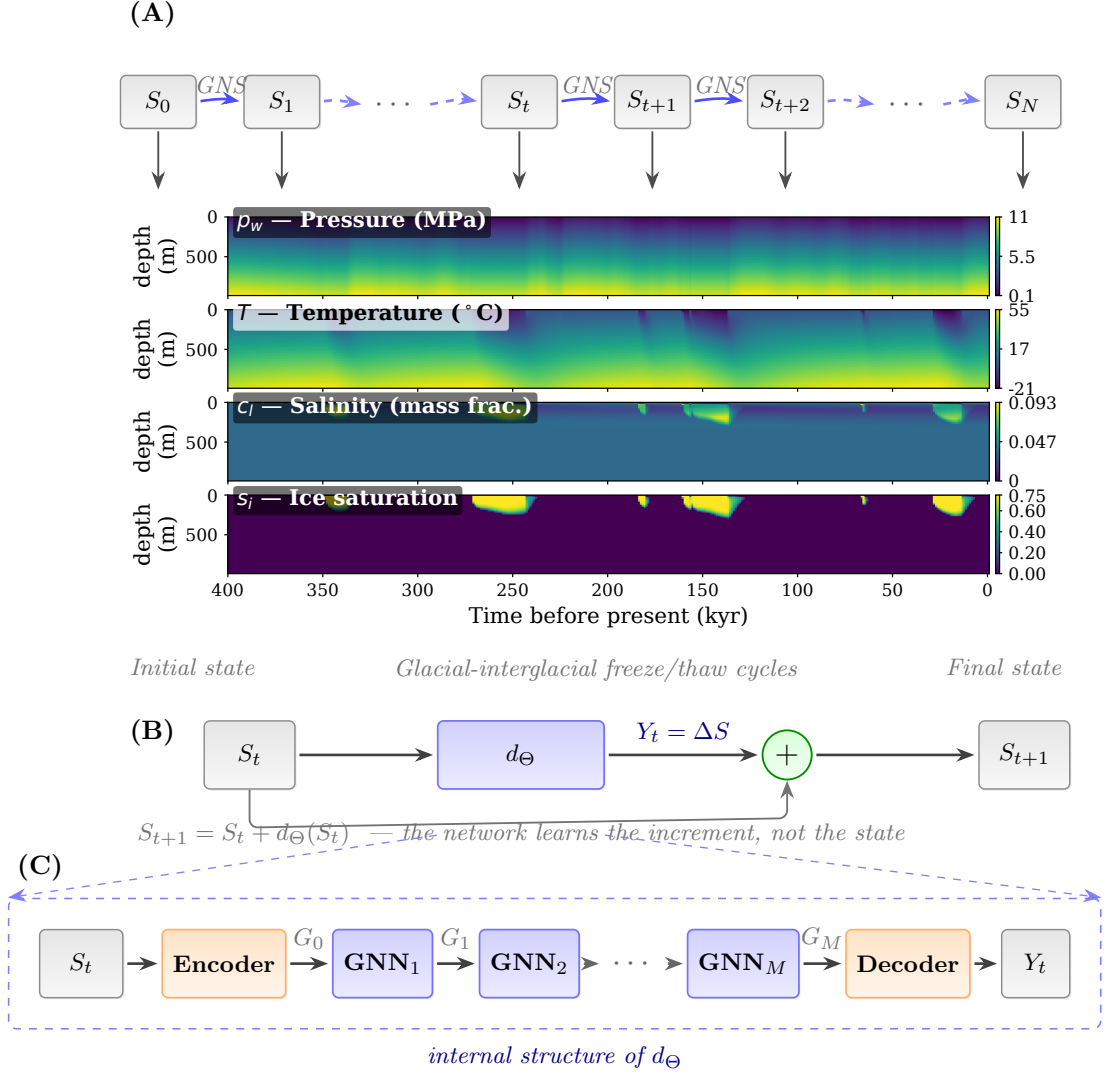


Fig. 1 GNS pipeline schematic. (A) Autoregressive evolution of the permafrost system state from S_0 to S_N : each solid arrow is one GNS step, ellipses elide the intermediate steps, and the four depth–time heatmaps show one ground-truth trajectory of pressure (p_w), temperature (T), salinity (c_l), and ice saturation (s_i) in physical units (the ice colorbar is labeled at its maximum, 0.75). (B) Per-step learned dynamics: d_Θ maps the state S_t to the dynamics $Y_t = \Delta S$, which is added to S_t to give $S_{t+1} = S_t + d_\Theta(S_t)$; the network learns the increment, not the state. (C) Internal structure of d_Θ (dashed inset): the encoder builds the latent graph G_0 from S_t ; M message-passing rounds with shared structure produce $G_1, \dots, G_M$; the decoder extracts Y_t from G_M . Color coding: blue = learned computation, peach = encoder/decoder, grey = state, green = deterministic update.

2.2 Graph Neural Networks and Message Passing

2.2.1 Graphs and Permafrost System State

We discretize the column into vertices V , one per depth node (51 in total), and edges E between consecutive vertices. The state at vertex i at time t is the four-channel vector $u_i^t = [p_w, T, c_l, s_i]_i^t \in \mathbb{R}^4$. Each vertex carries a node-property vector that includes the five global controlling parameters that vary across scenarios and the time-varying boundary conditions broadcast across all nodes; the column geometry enters through

relative edge attributes (Section 2.3). The global controlling parameters are the sediment thermal conductivity λ_s , the geothermal heat flux q_t , the e-folding depth of sediment porosity e_{fold} , the residual water saturation of the soil-water freezing curve $S_{\text{wr}}^{\text{freeze}}$, and the present-day water depth d_{simul} . The graph is fixed in topology across time and scenarios; only vertex attributes change.

2.2.2 Graph Neural Networks (GNNs) and Message Passing

In this graph, physical influence between neighboring depth nodes is realized computationally through message passing [6, 9, 13]: each vertex updates its state using signals aggregated from its immediate neighbors, mirroring how heat, fluid, and solute fluxes couple adjacent depths in the governing PDEs. Given vertex features v_i , edge features $e_{i,j}$ for the edge from vertex i to vertex j , and $N(i)$ the set of vertices with an edge into i , one message-passing round transforms graph $G = (V, E)$ into an updated graph $G' = (V', E')$ via three steps:

$$e'_{i,j} \leftarrow \phi_{\Theta_\phi}(v_i, v_j, e_{i,j}), \quad (\text{message construction}) \quad (1)$$

$$\bar{e}_i \leftarrow \sum_{j \in N(i)} e'_{i,j}, \quad (\text{message aggregation}) \quad (2)$$

$$v'_i \leftarrow \gamma_{\Theta_\gamma}(v_i, \bar{e}_i). \quad (\text{vertex update}) \quad (3)$$

Here Θ_ϕ and Θ_γ are learnable parameters of the message and update functions, both implemented as multi-layer perceptrons. An important hyperparameter is the number of message-passing steps M : each application of Equations 1–3 propagates information one hop farther across the graph.

2.3 The Surrogate, GNS Model

We build the permafrost surrogate on top of this message-passing machinery using the GNS architecture of Liu and Becker [9], adopted here without modification to its internal structure. That architecture — itself built on the open-source MeshNet GNS [10, 11] — couples four sub-networks (encoder, processor, decoder, updater) around the message-passing core just introduced. Figure 1 sketches the pipeline as applied here: panel A the autoregressive rollout across a full trajectory of permafrost states, panel B the per-step update of state, and panel C the internal encoder–processor–decoder structure of d_Θ . The changes relative to Liu and Becker [9] are the dimension and physical meaning of the state vector (four permafrost channels in place of fault slip rate), the top-boundary condition, and curriculum training schedule described next.

2.3.1 Input, Encoder, Processor, Decoder, and Updater

Each vertex i receives a feature vector $s_i^t \in S^t$ built from the current state u_i^t , the five global parameters (broadcast to every node since they describe the scenario as a whole), and the three top-boundary values (also broadcast, though only the surface node uses them). Depth enters only through the edge attributes, not s_i^t . The encoder, processor, decoder, and updater otherwise follow Liu and Becker [9], Choi and Kumar

[11] as a two-layer, 128-wide MLPs for the encoders and the $M = 10$ message-passing rounds, a two-layer, 128-wide MLP decoder, and an explicit-Euler updater, $u_i^{t+1} = u_i^t + dt y_i^t$; we refer to their Section 2.3 for the full derivation. At prediction (i.e., rollout) time, the predicted pressure, temperature, and salinity at the surface node are replaced with the known boundary values before the next step, while ice saturation at the surface and all four channels below it are left as the network predicted them.

2.4 Reference Data Sets

Our reference data set is an ensemble of 941 PermaCycle simulations generated by Latin hypercube sampling (LHS) [14] of the five physical parameters listed in Section 2.3. Each case contains a trajectory of 402 timesteps at $dt = 1$ kyr, 51 nodes (50 sediment plus a surface node), and four state channels per node.

To probe how well the surrogate generalizes beyond its training cases, we split the ensemble two ways, following the interpolation/extrapolation strategies as follows:

Data set Dr (random). A uniform random shuffle gives 661 train / 138 valid / 142 testing cases. Testing parameter combinations are within the LHS hypercube but unseen during training. Median test-to-nearest-train distance in the unit-normalized parameter space is 0.20.

Data set De (extrapolation). Cases are sorted by Euclidean distance from the parameter-space centroid. The closest 70% are used for training, the next 15% for validation, and the farthest 15% for testing (662/144/135 cases). Training cases have distance up to 0.70; testing cases lie at 0.77–1.02; a clean gap separates training from testing in parameter space, forcing the surrogate to predict at corners of the training distribution.

All four state channels are normalized to $[0, 1]$ using the global min/max computed over the union of training, validation, and testing trajectories; the constant parameters and the derived boundary conditions are similarly min/max normalized over all splits. We use this convention so the $[0, 1]$ bounds are exact and labels are not leaked into the network. Normalization statistics are persisted in a metadata file for de-normalization at rollout. Figure 2 shows the parameter-space layout of the two splits.

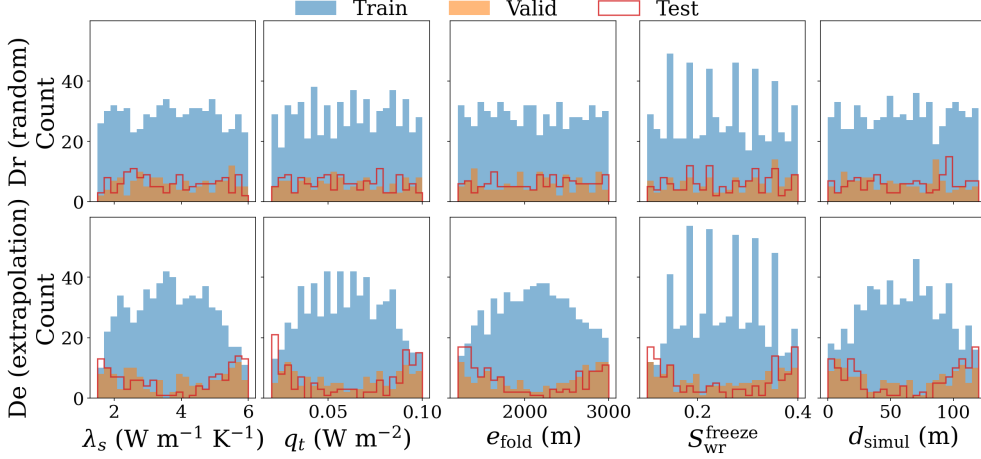


Fig. 2 Distributions of the five parameters ($\lambda_s, q_t, e_{\text{fold}}, S_{\text{wtr}}^{\text{freeze}}, d_{\text{simul}}$) for the training/validation/testing cases under the random split (Dr, top row; 661/138/142 cases) and the corner-extrapolation split (De, bottom row; 662/144/135 cases). In Dr, the testing set (red outline) overlaps the training distribution. In De, training cases are concentrated near the parameter-space centroid and the testing cases sit in the corners (bimodal red histograms in every panel), so the surrogate must extrapolate beyond the trained distribution.

2.5 Training and Prediction

We train the GNS to minimize the mean squared error (MSE) between predicted and ground-truth normalized state changes of the four state channels using the Adam optimizer [15] with the hyperparameters in Table 1. Each training sample is a pair of consecutive states drawn at random from the training trajectory pool. Following Sanchez-Gonzalez et al. [6], Choi and Kumar [11], and Liu and Becker [9], zero-mean Gaussian noise is added to the input state during training, so the network learns to correct the slightly perturbed inputs it will encounter during autoregressive rollout. Validation loss is averaged over 16 randomly drawn validation samples at each reporting interval to give a stable signal. At inference (*rollout*), the trained network is applied repeatedly: starting from the case’s initial state, each predicted state becomes the input for the next step, chaining together the full 401-step trajectory (Figure 1A). At each step, the surface-node channels for pressure, temperature, and salinity are replaced by the precomputed boundary condition ($T_{\text{top}}, p_w^{\text{top}}, c_l^{\text{top}}$) before the network processes the next input; all four channels at interior nodes come directly from the GNS.

The baseline model is trained with a multistep curriculum, following the incremental-unroll strategy of Lam et al. [8]. Instead of training the network to predict a single step ahead, the network is unrolled for K consecutive steps per gradient update, with full back-propagation through time (BPTT), so that it learns to correct its own drifted predictions rather than only ground-truth inputs; K increases from 1 to 6 in stages of 10^5 training steps, and each stage restarts the learning rate with a 5,000-step linear warmup followed by a half-cosine decay to 0.01η , where η is the base learning rate (Table 1). This multistep curriculum is the second addition relative to

Liu and Becker [9], which trains single-step only. Section 3.4 quantifies the accuracy gained by this curriculum relative to standard single-step training.

Table 1 GNS hyperparameters, identical for the random-split (Dr) and corner-extrapolation (De) trainings.

Parameter	Value
Learning rate η	10^{-4}
Batch size	2
Message-passing rounds M	10
Latent dimension	128
MLP hidden dimension	128
Number of MLP hidden layers	2
Gaussian noise std σ	0.01
Encoder input dimension	13
Multistep curriculum (baseline)	K=1→6, stage every 10^5 steps
Per-stage LR schedule	5,000-step warmup, cosine decay to 0.01η
Training steps	2×10^6
Reported checkpoint	10^6

3 Results

3.1 Performance on the Random Split (Dr)

Baseline configuration. Throughout the rest of the paper, the *baseline* GNS uses the architecture of Section 2.5 ($\eta = 10^{-4}$, $\sigma = 0.01$, $M = 10$ message-passing rounds, hidden = 128, latent = 128) trained for 10^6 steps with the multistep K=1→6 curriculum and per-stage warmup and cosine learning rate decay. Per-step optimization through K unrolled steps with full back-propagation through the unroll teaches the network to be robust to the drifted inputs it actually sees during autoregressive rollout. The multistep scheme improves prediction quality, in particular for the ice-saturation channel, where the single-step training counterpart is treated as an ablation in Section 3.3.

On the random split set (Dr), the trained baseline predicts well the joint state evolutions and reaches per-variable normalized MSE of 10^{-5} - 10^{-4} on the smoothly varying channels, i.e., pressure and temperature, and 6.8×10^{-4} on the ice-saturation channel across all 142 test trajectories at 10^6 training steps. The mean of all four state channels is 1.9×10^{-4} . We present evolutions of two test cases that show qualitatively different dynamics. Figure 3 shows case 0 with thin, repeated ice layers synchronized with the glacial-interglacial cycles. Figure 4 shows case 97 with deep, near-saturated ice column at maximum extent. Across both cases, pressure and temperature track the ground truth closely; salinity captures the surface-enrichment cone with mild smoothing; the ice-saturation field reproduces the timing and depth extent of the glacial-interglacial events. The difference rows in Figures 3–4 show the difference between PBNS ground truth and GNS predictions as a percentage of each variable’s dataset-wide range, capped at $\pm 10\%$; residuals stay well within this cap for pressure, temperature, and salinity, and only briefly saturate it at the sharp freeze-thaw fronts

in the ice-saturation channel. The time series comparison of four variables at the depth 50 m shows that the largest mismatches occur during the sharp changes of salinity and ice saturation (Appendix A). Nonetheless, the GNS predictions capture the starting times and durations of sharp phase changes well. Cases with no ice formation (67 of the 941 cases; the expected outcome for deep-water settings) are reported separately in Appendix B.

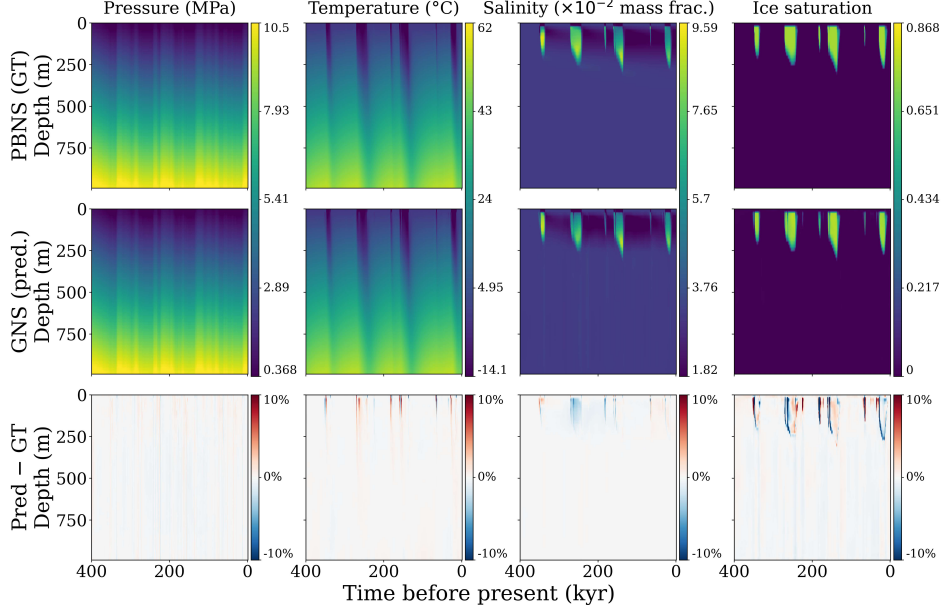


Fig. 3 Depth-time heatmaps for case 0 (Dr test set; $\lambda_s = 2.7 \text{ W m}^{-1} \text{ K}^{-1}$, $q_t = 0.077 \text{ W m}^{-2}$, $e_{\text{fold}} = 2877 \text{ m}$, $S_{\text{wr}}^{\text{freeze}} = 0.25$, $d_{\text{simul}} = 73 \text{ m}$). Top row: PBNS ground truth. Middle row: GNS prediction. Bottom row: prediction minus ground truth, expressed as a percentage of each variable’s global (dataset-wide) range and capped at $\pm 10\%$ — the same fixed scale for every variable and every case figure in this paper, including Figures 4 and B2, so panels are directly comparable across variables and cases.

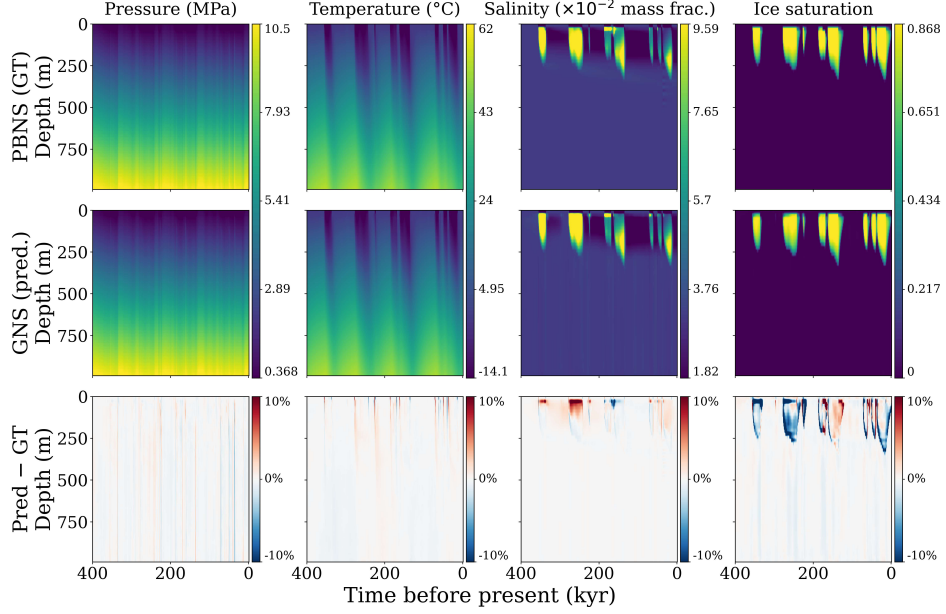


Fig. 4 Depth-time heatmaps for case 97 (Dr test set; $\lambda_s = 3.8 \text{ W m}^{-1} \text{ K}^{-1}$, $q_t = 0.066 \text{ W m}^{-2}$, $e_{\text{fold}} = 2691 \text{ m}$, $S_{\text{wt}}^{\text{freeze}} = 0.17$, $d_{\text{simul}} = 91 \text{ m}$). Layout, color ranges, and diff-row scale identical to Figure 3. The surrogate reproduces the persistence of the ice column and the salinity-front migration ahead of the freezing front.

Ice saturation remains the dominant residual at 6.8×10^{-4} , a factor of ~ 25 – 50 above the smoothly varying channels. We note that the relatively larger normalized MSE of sharp glacial-cycle freeze-thaw events persists under any training scheme. Multistep BPTT improves the absolute accuracy of each channel relative to single-step training (Table 4).

3.2 Performance on Corner Extrapolation (De)

On the dataset De (extrapolation split), we train the GNS from scratch with the same architecture and hyperparameter but on the 662 cases closest to the parameter-space centroid, and test on the 135 cases farthest from it. Figure 2 shows the resulting parameter-space gap. Table 2 compares the two setups, each with the $K=1 \rightarrow 6$ curriculum model trained on its own split and evaluated at the 10^6 -step checkpoint. The GNS remains accurate when extrapolating towards the corners of the parameter space where per-variable errors are within 1.3 – $1.7\times$ of the random-split results (pressure $1.7\times$, temperature $1.6\times$, salinity $1.7\times$, ice saturation $1.3\times$). Figures 5 and 6 show two extrapolation examples with qualitatively different ice-formation dynamics from the De test set: case 95 with thin, shallow ice layers formed during glacial-interglacial cycles, and case 86 with deep, persistent ice layer. The GNS performs well on such unseen, corner parameter regimes extrapolating from the training distribution. All heatmap figures share the same per-variable color range, so panels are directly comparable across cases and channels.

Table 2 Per-variable normalized MSE for the random-split (Dr, interpolation) and corner-extrapolation (De) test sets, each with the K=1→6 curriculum model trained on its own split and evaluated at the 10^6 -step checkpoint. Per-case averages over the 142 (Dr) and 135 (De) test trajectories.

Split	Step	avg of 4 vars	pressure	temperature	salinity	ice_saturation
Dr (interp.)	10^6	1.9×10^{-4}	1.4×10^{-5}	2.4×10^{-5}	2.6×10^{-5}	6.8×10^{-4}
De (extrap.)	10^6	2.5×10^{-4}	2.3×10^{-5}	3.8×10^{-5}	4.3×10^{-5}	9.0×10^{-4}

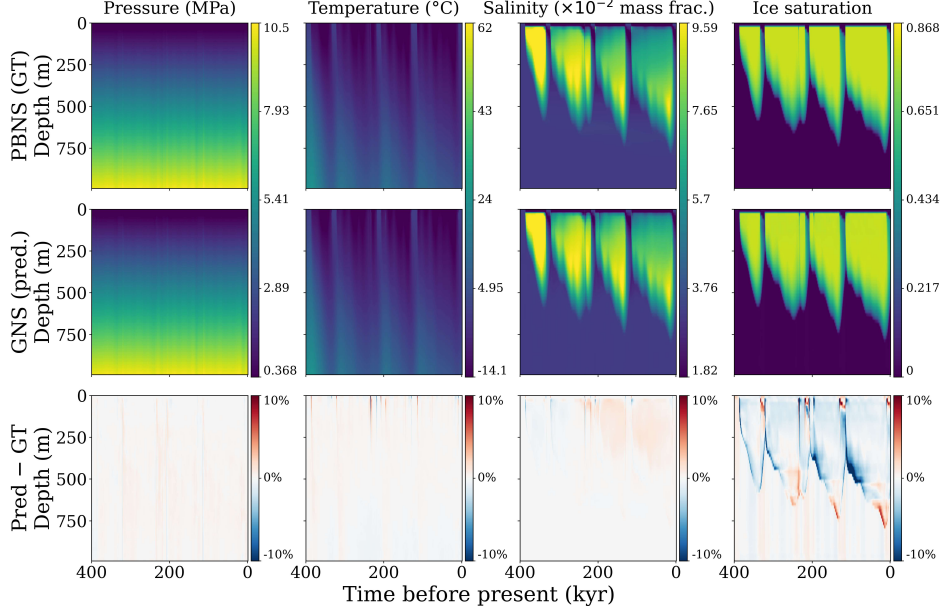


Fig. 5 GNS prediction on a corner-extrapolation test case (case 95: $\lambda_s = 5.9 \text{ W m}^{-1} \text{ K}^{-1}$, $q_t = 0.049 \text{ W m}^{-2}$, $e_{\text{fold}} = 2347 \text{ m}$, $S_{\text{wr}}^{\text{freeze}} = 0.37$, $d_{\text{simul}} = 6 \text{ m}$). Layout and scales identical to Figure 3 (top: PBNS ground truth, middle: GNS prediction, bottom: the difference between prediction minus ground truth relative to dataset range for each variable, capped at $\pm 10\%$.)

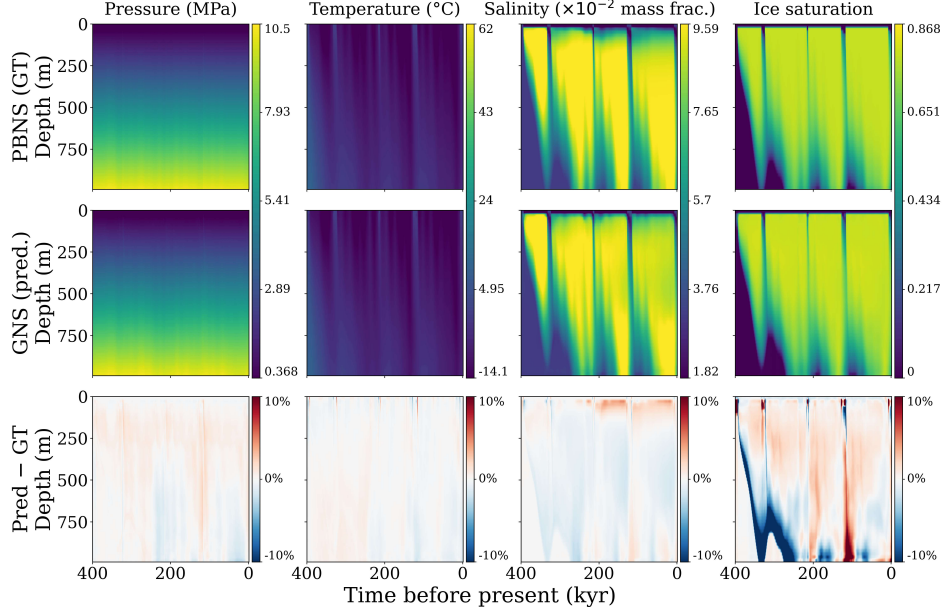


Fig. 6 As in Figure 5, but for case 86 ($\lambda_s = 3.2 \text{ W m}^{-1} \text{ K}^{-1}$, $q_t = 0.024 \text{ W m}^{-2}$, $e_{\text{fold}} = 2493 \text{ m}$, $S_{\text{wf}}^{\text{freeze}} = 0.11$, $d_{\text{simul}} = 13 \text{ m}$) with a different dynamics where persistent ice layer exists. (top: PBNS ground truth, middle: GNS prediction, bottom: the difference between prediction minus ground truth relative to dataset range for each variable, capped at $\pm 10\%$.)

3.3 Single-Step Training Ablation and Hyperparameter Sweep

Single-step training as the ablation. To isolate the contribution of multistep BPTT and the choice of hyperparameter such as learning rate, message-passing depth, and noise level, we present a hyperparameter sweep with *single-step* training, where the network is optimized to predict the next state change. Compared to the baseline configuration, we sweep five axes of hyperparameters one at a time while keeping the others fixed. The values of hyperparameters swept include learning rate $\eta \in \{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}$, noise $\sigma \in \{0.005, 0.01, 0.02\}$, message-passing rounds $M \in \{6, 10, 15\}$, hidden dimension (latent and MLP, swept jointly so they are equal) $\in \{16, 32, 64, 128, 256\}$, and batch size $\in \{1, 2, 4, 8\}$. All sweep configurations are reported at the 10^6 checkpoint. Each trained configuration is evaluated by a full test-set rollout and ranked by the average normalized MSE across the four state channels and all the test trajectories (Table 3).

Table 3 Random-split (Dr) hyperparameter sweep results at the 10^6 -step checkpoint, sorted ascending by average normalized MSE on the 142-trajectory test set. Each configuration trained to 2×10^6 steps.

Tag	Normalized MSE (10^6 steps)
$M = 6$	0.00073
$\eta = 2 \times 10^{-4}$	0.00075
batch = 8	0.00078
hidden = 256	0.00080
$M = 15$	0.00086
$\eta = 5 \times 10^{-5}$	0.00086
baseline ($\eta = 10^{-4}$, $\sigma = 10^{-2}$, $M = 10$, hd=128)	0.001
$\sigma = 0.005$	0.001
$\sigma = 0.02$	0.0011
hidden = 32	0.0013
hidden = 64	0.0014
batch = 4	0.0015
batch = 1	0.0021
hidden = 16	0.0029

The sweep shows that the baseline configuration ($\eta = 10^{-4}$, $\sigma = 10^{-2}$, $M = 10$, hd= 128, batch = 2) is not optimal but most configurations produce close performance for this permafrost system. The best configuration is $M = 6$ with normalized MSE at 7.3×10^{-4} , followed closely by $\eta = 2 \times 10^{-4}$ (7.5×10^{-4}), batch = 8 (7.8×10^{-4}), and hidden = 256 (8.0×10^{-4}), then the baseline (1.0×10^{-3}). The baseline remains within a factor of ~ 1.4 of the best single-step configuration, so the architectural defaults sit in a reasonable region of hyperparameter space. Instead, the table clearly show a batch size of 1 and small hidden layer dimension of 16 hurt model performance. The real headline gain in this paper comes from multistep BPTT training (Section 3.4).

3.4 Multistep Curriculum Training

The single-step training never sees the drifted inputs that arise during autoregressive rollout. A natural fix is to train on K unrolled steps per gradient update with full back-propagation through the unroll (BPTT), so the network is optimized for its actual rollout error accumulations [8]. We implement this in two configurations on top of the baseline GNS architecture: (i) a K=1→4 curriculum, with K incremented at 10^5 , 2×10^5 , and 3×10^5 steps and a final K=4 stage from 3×10^5 steps onward; (ii) a K=1→6 curriculum, extending to K=6 with stage transitions every 10^5 steps and a final K=6 stage from 5×10^5 steps onward. Each stage gets a 5,000-step linear LR warm-up from $\eta/10$ to η followed by a half-cosine decay to 0.01η over the rest of the stage.

Table 4 reports per-channel normalized MSE on the Dr test set at 10^6 training steps for the single-step baseline, the K=1→4 curriculum, and the K=1→6 curriculum. Multistep BPTT delivers a substantial drop on every channel — most importantly on the ice channel, where the K=1→6 curriculum reaches 6.8×10^{-4} , a $5.0\times$ improvement over the 10^6 -step single-step baseline (3.4×10^{-3}). Further increasing K shows marginal value where K=1→6 improves on ice saturation by about 13 % compared to K=1→4.

After each increase in K , the rollout error first rises for roughly 10^5 steps and then drops below its previous best as the learning rate warms up and decays. Per training step, K -step BPTT costs $K \times$ the forward+backward of single-step training. Rollout cost is unchanged, since multistep training affects only the optimization signal, not the per-step network or the rollout inference. The accuracy gain is therefore "free" at inference and entirely paid for in training-time compute.

Table 4 Multistep curriculum on the Dr test set at 10^6 training steps. All three rows use the same baseline architecture ($\eta = 10^{-4}$, $\sigma = 10^{-2}$, $M = 10$, $\text{hd}=128$); only the training curriculum differs.

Configuration	avg	pressure	temperature	salinity	ice saturation
Single-step baseline ($K=1$)	1.0×10^{-3}	4.5×10^{-4}	1.2×10^{-4}	1.2×10^{-4}	3.4×10^{-3}
$K=1 \rightarrow 4$ curriculum	2.2×10^{-4}	1.8×10^{-5}	2.8×10^{-5}	3.1×10^{-5}	7.8×10^{-4}
$K=1 \rightarrow 6$ curriculum (baseline)	1.9×10^{-4}	1.4×10^{-5}	2.4×10^{-5}	2.6×10^{-5}	6.8×10^{-4}

3.5 Computational Performance

Each PBNS case ran with a single-threaded MATLAB process, so the measured wall-clock equals the single-core CPU compute time. Across these cases the total time per case ranged from 3.2 to 7.0 hours (mean 5.0 h, median 5.9 h) on AMD EPYC-class hardware, with a per-step ($\Delta t_{\text{sim}} = 1$ kyr) cost ranging from 28 to 62 s of wall-clock time, with a median of 52 s.

GNS rollout speedup. GNS achieves a good speedup compared to the PBNS. However, we note that the speedup numbers should not be taken as face value without specifying the hardware resources used. Here, our GNS speedup numbers are reported as rollout on a single NVIDIA A100 GPU vs a single CPU of AMD EPYC hardware. A single-trajectory rollout of 402 steps of the joint state takes ~ 2.4 s on a single A100. At this rate the per-case speedup over the single-thread PBNS is about 4,800 to 10,500 $\times$ (median $\sim 8,800\times$). However, the 51-node 1-D graph never saturates the device and we exploit this idle headroom by batching multiple test trajectories into a single block-diagonal graph (so B trajectories share one forward pass per autoregressive step) and amortizing the per-step overhead across B cases. At $B = 142$ (the entire test set in one batched pass), the rollout completes in 14.2 s, i.e., 0.100 s per case, which is 24 $\times$ faster than the unbatched $B = 1$ baseline. If batched, per-case speedup over the single-thread PBNS rises to roughly 200,000 times. GPU utilization is 28% at $B = 1$ and settles near 22% for $B \geq 8$, and peak GPU memory at $B = 142$ is well under 1 GiB on a 40 GiB A100, so larger test sets would still fit comfortably in one pass.

Training the GNS for 10^6 single-step iterations takes roughly 3–4 hours on a single NVIDIA A100, which is a one time upfront cost. Multistep BPTT multiplies the per-step cost by K (Section 3.4). Table 5 summarizes these costs and computational performance.

Table 5 Computational cost for GNS rollout and training versus the PBNS reference.

Stage	Hardware	Cost
PBNS simulation, per case	1 CPU thread	3.2–7.0 h (median 5.9 h)
PBNS simulation, per step	1 CPU thread	28–62 s (median 52 s)
GNS rollout, $B = 1$	1 A100 GPU	~ 2.4 s/case (median)
GNS rollout, $B = 142$	1 A100 GPU	~ 0.11 s/case (median)
GNS training, 10^6 steps	1 A100 GPU	~ 3 –4 h
Dataset prep, 992 cases	1 CPU thread	~ 3 min
Speedup, unbatched	—	$\sim 8,800\times$ (median)
Speedup, batched ($B = 142$)	—	$\sim 198,000\times$ (median)

4 Discussion

Enforcing the simulator’s prescribed top-boundary state at the surface node during GNS rollout, rather than letting the network predict it, appears to be a useful inductive bias for this problem. Because the boundary state cannot drift over the rollout, error accumulation is bounded precisely in the near-surface channels where most sharp changes of the ice-saturation dynamics occur.

The interpolation-versus-extrapolation comparison quantifies how much of the reported accuracy depends on test cases lying within the trained distribution versus at its corners. With the $K=1\rightarrow 6$ curriculum trained to the same budget on each split, the average normalized MSE rises from 1.9×10^{-4} on the random split to 2.5×10^{-4} at the corners — a factor of only 1.4 (Table 2). Per-channel ratios are similarly modest: pressure $1.7\times$, temperature $1.6\times$, salinity $1.7\times$, and ice saturation $1.3\times$. Generalization is, of course, bounded by the training data distribution. Our work demonstrates that the GNS can extrapolate to parameter regimes further away from the training distribution without sacrificing much rollout performance, compared to generalization to parameter regime within training distribution.

Our GNS approach adopts the established GNS architecture (i.e., Liu and Becker [9] for earthquake rupture dynamics); what differs is the physical system, the addition of top-boundary injection, and the multistep curriculum. The workflow (autoregressive rollout, Adam optimization, Gaussian noise injection during training, message-passing depth $M = 10$) largely transfers, which indicates the GNS framework is agnostic to the physical system underlying the spatiotemporal evolutions it emulates, and points to a broader applicability of the same architecture and training recipe to accelerate other Earth-system surrogate problems governed by nonlinear coupled PDEs and forced by a small number of exogenous boundary signals.

Several limitations bound the scope of these results for sure. The surrogate operates on a fixed 1-D, 51-node column; two- and three-dimensional shelf geometries are naturally next step but rely on advancing existing PBNS in two- and three-dimensional. All 941 trajectories share the same Pleistocene sea-level and surface-temperature forcing, so generalization across forcing scenarios is our next step to explore. The configurations used here develop no free gas phase and other necessary physical processes should be incorporated. Finally, the corner-extrapolation test probes parameter extremes within

the sampled hypercube; behavior far outside these ranges is worth testing in future work.

5 Conclusion

We present a Graph Network-based Simulator for the 1-D thermo-hydro-chemical evolution occurring within Arctic subsea permafrost, adopting the GNS architecture of Liu and Becker [9] and demonstrating that it transfers to a physical system with no elastodynamic analog. The surrogate is trained on 941 successful trajectories over five physical parameters and predicts well the joint evolutions of pressure, temperature, salinity and ice-saturation autoregressively. The predictions reach per-variable normalized MSE of 10^{-5} – 10^{-4} on pressure, temperature, and salinity and 6.8×10^{-4} on the ice-saturation channel on a held-out random-split test set. Enforcing the simulator’s prescribed top-boundary state at the surface node during rollout aligns the network to solve the same problem as the Physics-based Numerical Simulator (PBNS). Training the per-step network with a multistep $K=1 \rightarrow 6$ BPTT curriculum, which exposes the network to its own drifted predictions during training and brings the ice-saturation MSE down by $5\times$ relative to standard one-step training. The surrogate generalizes well within the trained parameter distribution and remains accurate on a corner-extrapolation split where the testing cases lie at the corners of the parameter space, with average normalized MSE within $1.4\times$ of the within-distribution result. The computational speedup of GNS rollout on a single NVIDIA A100 compared to a single CPU PBNS can reach $\sim 8,800\times$ unbatched and $\sim 200,000\times$ batched. The speedup from GNS opens the door for parameter sensitivity studies, uncertainty quantification, and inverse problems for subsea permafrost dynamics. Our work demonstrates that the GNS architecture is agnostic about the physical systems underlying the spatiotemporal evolutions, and its potential to accelerate other Earth modeling systems currently limited by the cost of PBNS.

Acknowledgements. This work was supported by the U.S. National Science Foundation under award 2530985. Model training and evaluation were performed on an NVIDIA A100 GPU at Institute for Geophysics at the University of Texas at Austin.

Author contribution. Dunnyu Liu: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, review & editing. Kehua You: Conceptualization, Data curation, Software, Funding acquisition, Supervision, Writing – review & editing. Pengli Ren: Investigation, Writing – review & editing. Hewei Tang: Funding acquisition, Supervision, Writing – review & editing.

Funding. This research is funded by the U.S. National Science Foundation under award 2530985.

Data availability. The training/validation/test datasets for both splits, the trained model checkpoints reported in this paper, and the evaluation summaries are archived on Zenodo (<https://doi.org/10.5281/zenodo.21909500>). The physics-based simulator PermaCycle [5] (MATLAB), used to generate all training, validation, and test data, is available on Zenodo (<https://doi.org/10.5281/zenodo.20043352>) and GitHub (<https://github.com/KehuaYou/PermaCycle>).

Code availability. PermaCycleGNS is open-source via <https://github.com/dunyuliu/PermaCycleGNS> and archived on Zenodo (<https://doi.org/10.5281/zenodo.21909500>).

Declarations

Conflict of interest. The authors have no conflicts to disclose.

Competing interests. The authors declare no competing interests.

Ethics approval. Not applicable.

Consent to participate. Not applicable.

Consent for publication. Not applicable.

Appendix A Time Series at a Representative Depth

The depth–time heatmaps in the main text show the full rollout at once but can make it hard to judge point-wise agreement. Figure A1 instead shows ground truth and predicted time series for all four state channels at a single depth (50 m) for case 0 (Dr test set), spanning the full 400 kyr rollout. The surrogate tracks pressure and temperature well; salinity and ice saturation track the timing and amplitude of each glacial-cycle freeze/thaw event, with the small deviations visible during transitions consistent with the diff-row residuals in Figures 3–B2.

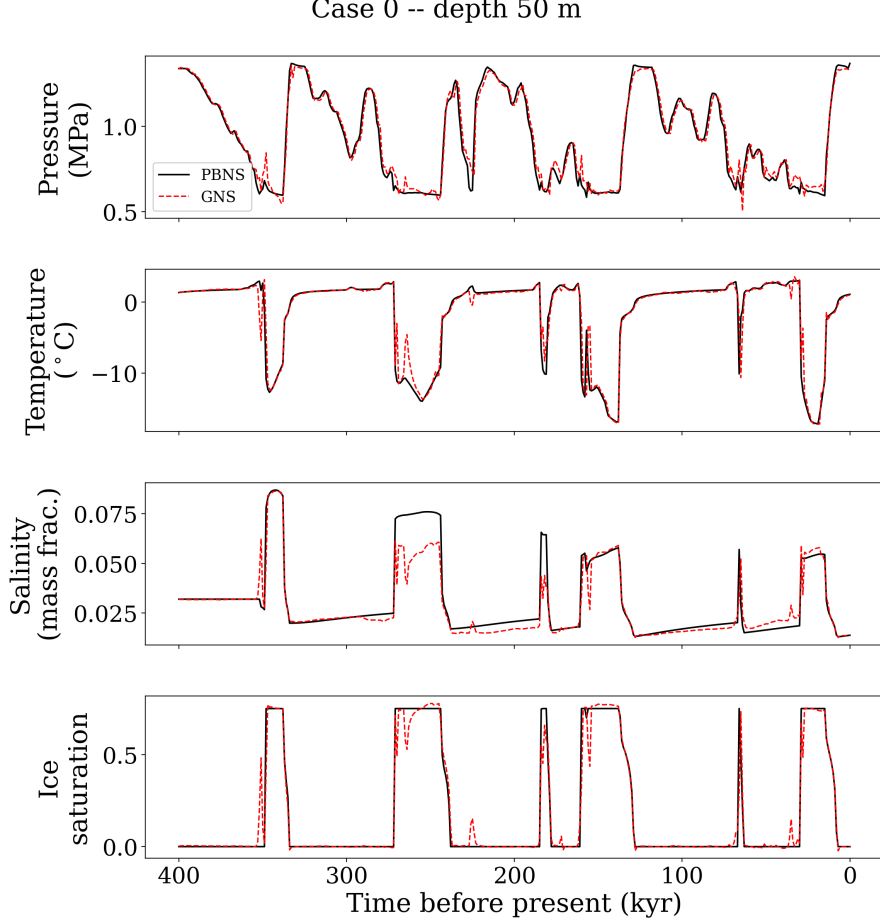


Fig. A1 Ground truth (black) versus GNS prediction (red dashed) at 50 m depth, case 0 (Dr test set), for all four state channels over the full 400 kyr autoregressive rollout.

Appendix B A Test Case Where No Ice Forms

Cases with no ice formation (67 of the 941 cases in the sweep) are the expected outcome for high geothermal heat flux q_t , deep-water configurations (d_{simul} near 120 m, so the seafloor temperature stays above the local freezing point most of the time), and warm Pleistocene minima; in these settings no ice forms over the 401 kyr simulation horizon. For these cases, the ice-saturation channel is identically zero throughout, and the surrogate’s residual on this channel is purely the noise of the network’s predicted-zero values. We include one such case here as a sanity check that the surrogate does not over-predict ice in regimes where none should occur.

Figure B2 shows case 60 (Dr test set), one of the no-ice cases. The ice-saturation panel is uniformly zero in the ground truth; the GNS prediction is correspondingly around zero, giving an ice-saturation normMSE of 6.2×10^{-6} , which is about three

orders of magnitude below the active-ice cases shown in the main text. Pressure, temperature, and salinity are likewise tracked closely. This gives evidence that the surrogate distinguishes the "no ice" regime from the active ice-cycling regime without explicit phase-state tagging.

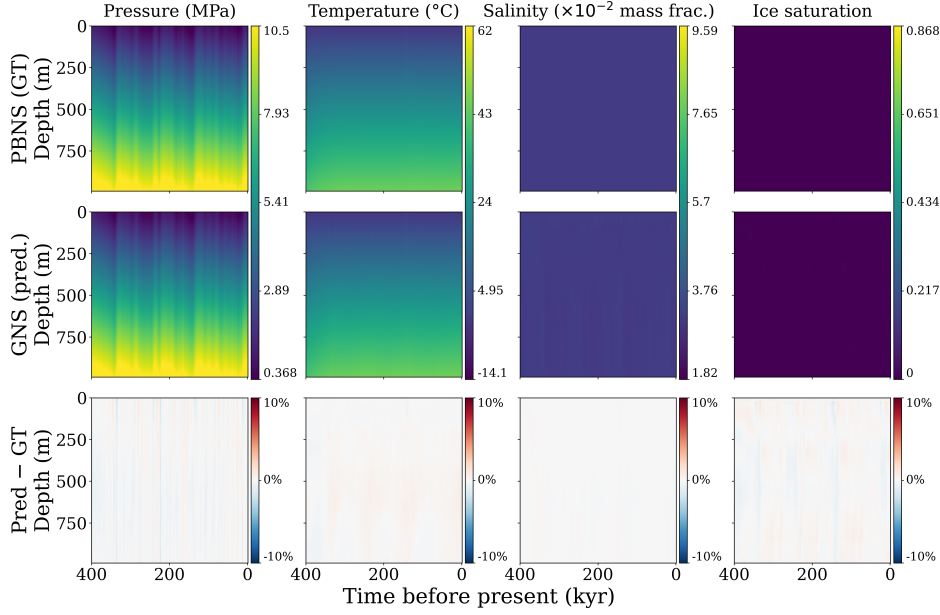


Fig. B2 Depth-time heatmaps for case 60 (Dr test set; $\lambda_s = 2.5 \text{ W m}^{-1} \text{ K}^{-1}$, $q_t = 0.053 \text{ W m}^{-2}$, $e_{\text{fold}} = 2233 \text{ m}$, $S_{\text{wr}}^{\text{freeze}} = 0.39$, $d_{\text{simul}} = 115 \text{ m}$; no ice forms in the simulation). Layout and scales identical to Figure 3 (top: PBNS ground truth, middle: GNS baseline, bottom: prediction minus ground truth as a percentage of each variable's global range, capped at $\pm 10\%$). The ice-saturation panel is uniformly zero in the top two rows; the diff row is near-uniformly zero.

References

- [1] Overduin PP, Schneider von Deimling T, Miesner F, Grigoriev MN, Ruppel C, Vasiliev A, et al. Submarine permafrost map in the Arctic modeled using 1-D transient heat flux (SuPerMAP). *Journal of Geophysical Research: Oceans*. 2019;124(6):3490–3507. <https://doi.org/10.1029/2018JC014675>.
- [2] Ruppel CD, Kessler JD. The interaction of climate change and methane hydrates. *Reviews of Geophysics*. 2017;55(1):126–168. <https://doi.org/10.1002/2016RG000534>.
- [3] Shakhova N, Semiletov I, Gustafsson Ö, Sergienko V, Lobkovsky L, Dudarev O, et al. Current rates and mechanisms of subsea permafrost degradation in the East Siberian Arctic Shelf. *Nature Communications*. 2017;8:15872. <https://doi.org/10.1038/ncomms15872>.

- [4] You K. Biodegradation of Ancient Organic Carbon Fuels Seabed Methane Emission at the Arctic Continental Shelves. *Global Biogeochemical Cycles*. 2024;38(2):e2023GB007999. <https://doi.org/10.1029/2023GB007999>.
- [5] You K.: PermaCycle v1.0.1: A Coupled Hydro–Thermal–Geochemical Model for Permafrost Dynamics. Zenodo. [Software]. Available from: <https://doi.org/10.5281/zenodo.20043352>.
- [6] Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia PW. Learning to Simulate Complex Physics With Graph Networks. In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR; 2020. p. 8459–8468.
- [7] Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW. Learning Mesh-Based Simulation With Graph Networks. *arXiv preprint arXiv:201003409*. 2020;.
- [8] Lam R, Sanchez-Gonzalez A, Willson M, Wirnsberger P, Fortunato M, Alet F, et al. Learning Skillful Medium-Range Global Weather Forecasting. *Science*. 2023;382(6677):1416–1421. <https://doi.org/10.1126/science.adi2336>.
- [9] Liu D, Becker TW. Earthquake Rupture Dynamics From Graph Neural Networks. *Journal of Geophysical Research: Solid Earth*. 2025;130(12):e2025JB031981. <https://doi.org/10.1029/2025JB031981>.
- [10] Kumar K, Vantassel J. GNS: A Generalizable Graph Neural Network-Based Simulator for Particulate and Fluid Modeling. *Journal of Open Source Software*. 2023;8(88):5025. <https://doi.org/10.21105/joss.05025>.
- [11] Choi Y, Kumar K. Graph Neural Network-Based Surrogate Model for Granular Flows. *Computers and Geotechnics*. 2024;166:106015. <https://doi.org/10.1016/j.compgeo.2023.106015>.
- [12] Creel RC, Miesner F, Wilkenskield S, Austermann J, Overduin PP. Glacial isostatic adjustment reduces past and future Arctic subsea permafrost. *Nature Communications*. 2024;15:3232. <https://doi.org/10.1038/s41467-024-45906-8>.
- [13] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural Message Passing for Quantum Chemistry. In: *Proceedings of the 34th International Conference on Machine Learning*; 2017. p. 1263–1272.
- [14] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*. 1979;21(2):239–245. <https://doi.org/10.1080/00401706.1979.10489755>.
- [15] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*. 2014;.