

Earthquake rupture dynamics from Graph Neural Networks

Dunyu Liu¹ and Thorsten W. Becker^{1,2,3}

¹ Institute for Geophysics, Jackson School of Geosciences, The University of Texas at Austin

² Department of Earth and Planetary Sciences, Jackson School of Geosciences, The University of Texas at Austin

³ Oden Institute for Computational Engineering & Sciences, The University of Texas at Austin

Highlights

1. We develop a Graph Network-based Simulator (GNS) for rupture dynamics.
2. The GNS generalizes to unseen hypocenters, pre-stress levels, and fault sizes with dozens to hundreds training scenarios.
3. The GNS achieves ~20-40 per-time-step speedup compared to physics-based *EQdyna* and may help with uncertainty quantification and Bayesian inference.

Abstract

Earthquakes arise from tectonic loading of complex fault systems consisting of heterogeneous material parameters, geometry, rheology, and prestress. All of those are subject to uncertainties, and their interactions and sensitivities for the dynamic rupture problem are incompletely understood. Here, we apply Graph Neural Networks (GNNs) to approximate the behavior learned from more computationally intensive, physics-based (“high-fidelity”) computations to build a GNN-based simulator (GNS) for earthquake rupture dynamics. Given only a minimum input –the hypocenter location– our GNS can reproduce rate-weakening friction governed dynamic rupture behavior, from nucleation to propagation and termination. Outside the training set, the GNS can generalize well to different hypocenter locations, fault sizes, and pre-stress state levels while achieving a factor ~20-40 per-time-step computational speedup. This may allow for more efficient estimates of the mapping from pre-earthquake state, as might be inferred from geodesy, to expected rupture dynamics, for example. By extracting a coarse-grained version of the underlying dynamics, the GNS provides new perspectives to explore the physics of rupture. Further development of GNS may enable new kinds of parameter space exploration and provide surrogates for Bayesian model inference, uncertainty quantification, and optimal experimental design.

Large earthquakes pose major seismic hazards, but their source processes are hard to constrain and predict due to many uncertainties. Scientists generally use “high-fidelity”, full physics-based models to study earthquake dynamics, but the high computational cost can limit the exploration of parameters and uncertainties. With the advance of GPU-accelerated computing and deep learning, Graph Network-based Simulators (GNSs) have emerged as powerful tools to model complex physical dynamics. Here, we apply GNS to learn and predict earthquake source dynamics governed by rate-weakening friction. Our GNS can generalize to unseen hypocenters, pre-stresses, and fault geometries. It also runs faster than the traditional physics-based approaches, providing a potential avenue for uncertainty quantification and inferring rupture physics from observations.

1. Introduction

Even though major faults in plate boundaries are mapped relatively well in seismically active regions, large earthquakes continue to surprise us with their complexity. Ruptures arise from tectonic loading of fault systems that are characterized by heterogeneous material parameters, irregular and multi-strand geometry, and variable stress state. Despite progress in incorporating observationally constrained stress conditions and laboratory-derived friction laws for complex rupture scenarios [e.g., Jia *et al.*, 2023; Ramos *et al.*, 2021; Ulrich *et al.*, 2019], many parameters influencing rupture dynamics remain poorly constrained. There are also remaining unknowns on the physical process side, such as to the role of fluids, frictional relationships, and rheology. Currently, there are few modeling frameworks capable of incorporating all relevant physics in a unified way to enable uncertainty quantification or Bayesian model inference for earthquake rupture.

Analysis typically must rely on a limited number of physics-based numerical simulations (PBNS) to explore aspects of the sensitivities of the nonlinear dynamics involved. For example, as for the basic rupture conditions, Andrews and Barall [2011] show how the combination of initial stress distributions and slip-weakening laws [e.g.,

Day, 1982; Ida, 1972] generate realistic rupture scenarios. Shi and Day [2013] show how fractal fault roughness and rate- and state- friction [Dieterich, 1979; Ruina, 1983] with normal stress dependence [Dieterich and Linker, 1992] reproduces realistic high frequency ground motion. Some of the additional effects explored with PBNS include the role of non-planar fault geometry and rate-weakening friction [e.g., Dunham et al., 2011; Luo and Duan, 2018] and the effects of fault valving and pore pressure evolution [e.g., Zhu et al., 2020]. Moreover, Duan and Oglesby [2006], Liu et al. [2022], and Shaw et al. [2022] highlight how multi-fault geometries and earthquake sequences shape stress evolution and rupture segmentation over time.

PBNS approaches have also been used for Bayesian inversions for pre-stress and friction parameters for single ruptures [e.g., Gallovič et al., 2019; Suhendi et al., 2025; Taufiqurrahman et al., 2022], for example. However, integrating a range of plausible physics into a 3-D model with multiple fault segments while allowing for exploration of both structure and physics uncertainties remains a significant computational and technical challenge. Yet, this is what is ultimately required for hazard estimates, such as when seeking to map from inferred fault loading distributions to likely rupture scenarios.

Very recently, machine learning surrogates based on deep neural networks have started to be used to emulate dynamic ruptures. RuptureNet2D [Gong et al., 2025], trained on 300,000 2-D dynamic rupture simulations, provides a mapping from initial stress, frictional parameters, and hypocenter locations to rupture arrival times and final slip. A reported speedup of ~1000 times compared to PBNS is partially achieved by targeting only the final output and bypassing the intermediate steps, and generalization is challenging when extrapolating to fault lengths and additional asperity patches beyond the training set. Tainpakdipat et al. [2025] present another end-to-end projection-type model using Fourier Neural Operators trained on 3000 scenarios with fractal initial stresses, to map initial stress and frictional parameters to subsampled timeseries of slip rate. Their models achieve a reported $\sim 10^5$ speed up, with the limitation that the predictions are confined to trained nucleation locations and fixed rupture duration.

We explore an alternative deep learning approach using Graph Neural Networks (GNNs) here. GNNs have emerged as promising tools to capture the temporal evolution of high-dimensional physical systems simulations by emulating all intermediate system states [e.g., Lam et al., 2023; Pfaff et al., 2020; Sanchez-Gonzalez et al.,

2020] and show strong generalization capabilities in terms of initial conditions, feature complexities, and model domain sizes. GNNs are particularly well suited for representing particle- and mesh-based systems, which many PBNS for rupture dynamics, such as finite-element and finite-difference methods, adopt [e.g., Harris et al., 2018; Liu and Duan, 2018; Premus et al., 2020; Ulrich et al., 2019; Wang and Day, 2020; Withers et al., 2018]. GNN-based simulators (GNSs) can capture a wide range of physical processes, including turbulent flow and solid and granular mechanics [Choi and Kumar, 2024; Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020] and other nonlinear systems such as weather [Lam et al., 2023; Price et al., 2024] and climate [Kochkov et al., 2024].

Besides the typical finding that GNS can provide computational speedup compared to PBNS, GNS' inductive nature allows generalization to system sizes not seen during training [Thangamuthu et al., 2022]. Moreover, incorporating physics-informed bias, such as inertia, in deep learning frameworks has also been shown to improve performance in the learning of dynamics [Choi and Kumar, 2024; Thangamuthu et al., 2022], and GNS show remarkable generalization and physics abstraction capabilities [Choi and Kumar, 2024; Pfaff et al., 2020; Thangamuthu et al., 2022].

Here, we develop a GNS for earthquake dynamics on a 2-D fault interface embedded in 3-D rock volume where the training data are generated from dozens to hundreds of 3-D PBNS rupture dynamics models and fault sliding is governed by friction with strong rate-weakening. We explore the GNS performance and generalization capabilities for three different types of setups –unseen hypocenters, pre-stress amplitudes and distributions, and fault sizes, and discuss GNS strengths, limitations, and potential future applications to earthquake physics and hazard.

2. Methods

2.1 The high fidelity, PBNS model

We use the CPU-parallelized finite-element software *EQdyna* [Duan and Oglesby, 2006; Liu and Duan, 2018; Liu et al., 2022] to generate earthquake rupture scenarios (sequences of fault sliding velocity states, or slip rates) at “high fidelity”, to train the GNS and as ground truth for testing and validation. *EQdyna* solves the 3-D elastodynamic equations

of motion with a friction law on fault interfaces for rupture propagation including wave radiation. *EQdyna* has been benchmarked extensively, *e.g.* within the *SCEC Spontaneous Rupture Code Verification Project* [Harris *et al.*, 2009, 2018]. Here, we choose the SCEC benchmark TPV104 as the reference, because the associated strong rate-weakening friction law produces significant slip rate changes during rupture. Moreover, this test case has been used for benchmarking by many rupture codes, ensuring comparability and easy reproducibility of our study. We then vary hypocenter locations and initial stress levels and stress anomaly distributions.

Figure 1A shows our reference setup, which models earthquake dynamics for a vertical, strike-slip, planar fault embedded in a 3-D isotropic, linear elastic half space, and breaking the surface. Slip evolution is governed by rate- and state- friction with strong-rate weakening (Table 1), following eqs. (1-7) in the SCEC TPV104 benchmark description (strike.scec.org/cvws/download/SCEC_validation_slip_law.pdf).

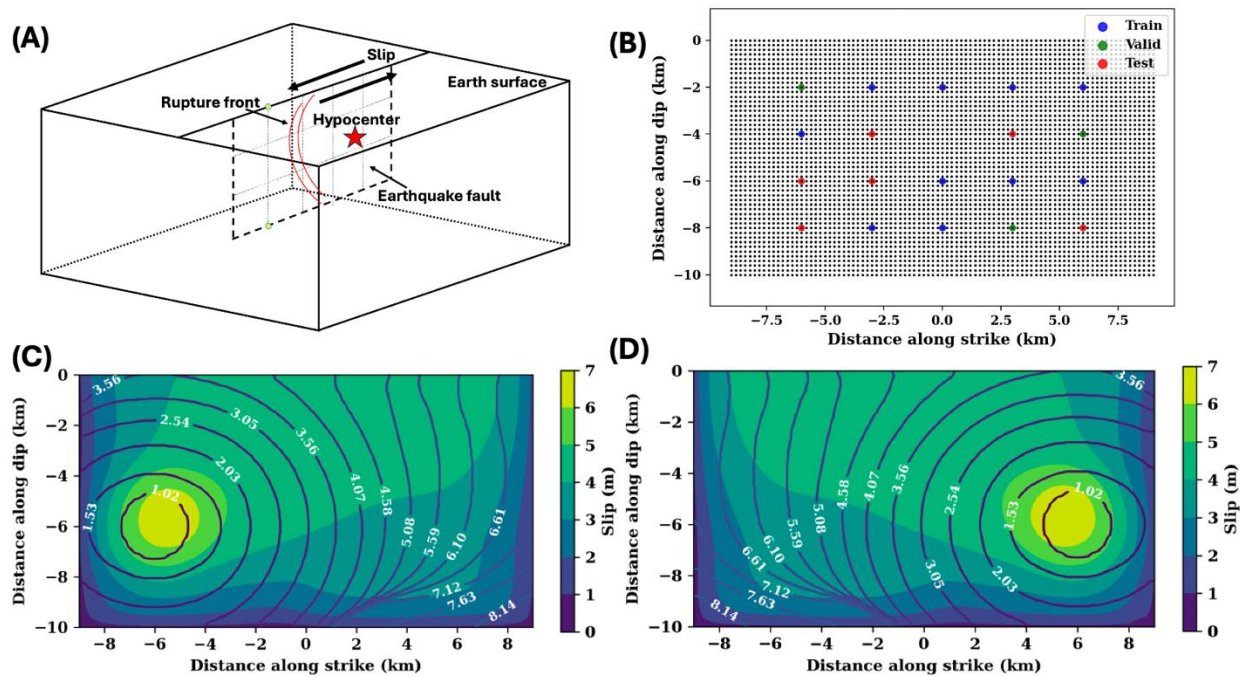


Figure 1. A) Model setup for earthquake rupture dynamics. A fault interface is embedded in a 3-D elastic volume, and the fault walls move relative to each other, with relative displacement defined as slip. Slip evolution is influenced by initial stress conditions, frictional mechanisms and other parameters such as hypocenter location. B) Hypocenter locations used to construct Dataset D1 for GNS with training (blue), validation (green), and test (red) sets. Gray dots represent fault nodes in the reference

model. C)-D). Final slip distribution and rupture time contours from two scenarios in the training set. White numbers indicate rupture arrival times in seconds.

We use a fault dimension of 18×10 km, 200 m on-fault grid size, and there are a total of 4,500 fault nodes in each scenario. The 3-D model domain boundaries are extended 20 km away from the nearest fault edges, respectively, and radiated seismic waves are damped by absorbing boundary conditions [Liu and Duan, 2018]. We start with homogeneous initial stresses and frictional parameters; at the hypocenter, nucleation is driven by imposing an additional shear stress over 1 s and within a 3 km radius, as per eqs. (14-16) in the benchmark. Because the fault is uniformly velocity weakening, rupture front reflections from fault boundaries are expected. This setup is intended to test the GNS' capability to learn and predict the dynamics beyond simple, crack-like extension of the rupture, such as the effect of boundaries.

Each computational scenario lasts 15 s, which is long enough for rupture to propagate over the whole fault and die out. With a time increment of 0.0167 s, each scenario contains 900 time steps. At the beginning of each scenario, the earthquake is under forced nucleation with minimal slip rate information on the fault. Therefore, we use rupture states after initiation at time $t = 1.2$ s to train the GNS models; this gives 826 time steps per scenario.

Table 1. Key parameters for the reference, physics-based numerical model

Fault length along strike (km)	18
Fault width along dip (km)	10
Grid size (m)	200
Initial shear stress (MPa)	40
Initial normal stress (MPa)	120
Density (km/m ³)	2670
S-wave velocity (km/s)	3.464
P-wave velocity (km/s)	6
<i>Rate- and state- friction parameters</i>	
Reference friction coefficient	0.6
Reference slip rate (m/s)	10^{-6}
a	0.01
b	0.014

Characteristic length, L , (m)	0.4
Friction threshold for strong rate weakening, f_w	0.2
Slip rate threshold for strong rate weakening, V_w , (m/s)	0.1
Initial state variable value, Ψ	0.563591842632738
Additional peak shear stress for nucleation (MPa)	45
Nucleation radius (km)	3
Artificial nucleation duration (s)	1

2.2 Graph neural networks and message passing

2.2.1 Graphs and earthquake system rupture state

Graph formalisms can be applied to emulate physical system interactions [e.g., Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020]. In an elastic solid, a frictional interface which governs earthquake rupture dynamics can be modeled as a graph $G = (\mathbf{V}, \mathbf{E})$ where a set of vertex variables $\mathbf{V}$ ($\mathbf{v}_i \in \mathbf{V}$) captures the kinematics of discretized vertices on the interface, and a set of edges $\mathbf{E}$ ($\mathbf{e}_{i,j} \in \mathbf{E}$) represents physical interactions between neighboring vertices through their relative locations ($\mathbf{v}_i$ and $\mathbf{v}_j$).

In our finite-element approach [e.g., Liu and Duan, 2018; Liu et al., 2022], we discretize the fault interface into vertices $\mathbf{V}_{FE}$ which are connected by edges $\mathbf{E}_{FE}$. We define a rupture state at time t as $S_t(\dot{\mathbf{u}}, \mathbf{x})$, for all the vertices' slip rates $\dot{\mathbf{u}}_i$ and their locations $\mathbf{x}_i$ on the fault interface ($i \in \mathbf{V}_{FE}$). Slip rate is the relative velocity between two fault walls bordering the interface. We then use a graph $G = (\mathbf{V}_{FE}, \mathbf{E}_{FE})$ to represent the rupture state and its evolution.

2.2.2 Graph Neural Networks (GNNs) and Message Passing

A Graph Neural Network (GNN) is an essential element in GNS architecture. The GNN takes a graph $G = (\mathbf{V}_{FE}, \mathbf{E}_{FE})$ as the input and outputs a new graph $G' = (\mathbf{V}'_{FE}, \mathbf{E}'_{FE})$ with updated $\dot{\mathbf{u}}'_i$ and $\mathbf{e}'_{i,j}$ through message passing [Gilmer et al., 2017]. Message passing is the process by means information is propagated through the graph structure, e.g., capturing the transport of energy and momentum. In the dynamic rupture context, the GNN takes graph $G = (\mathbf{V}_{FE}, \mathbf{E}_{FE})$ as the input that describes the current system state S_t and updates to a new $G' = (\mathbf{V}'_{FE}, \mathbf{E}'_{FE})$, which after decoder and updaters, returns the next state S_{t+1} .

Message passing contains three steps:

1. Message construction $\mathbf{e}'_{i,j} = \phi_{\Theta_\phi}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{i,j})$ (1)

2. Message aggregation $\bar{\mathbf{v}}_i = \sum_{j \in N(i)} \mathbf{e}'_{i,j}$ (2)

3. Update to vertex $\mathbf{v}'_i = \gamma_{\Theta_\gamma}(\mathbf{v}_i, \bar{\mathbf{v}}_i)$ (3)

Here, Θ_ϕ and Θ_γ are a set of learnable parameters; ϕ_{Θ_ϕ} is the message construction function, a matrix operation, that takes in the attribute vectors of the connecting vertices ($\mathbf{v}_i$ and $\mathbf{v}_j$) and those of the edge connecting the two vertices ($\mathbf{e}_{i,j}$) and returns an updated edge feature vector $\mathbf{e}'_{i,j}$. Then, messages are aggregated toward a single vector $\bar{\mathbf{v}}_i$ where $N(i)$ is the set of sender vertices j related to vertex i . The last step is to update vertex features using the aggregated message $\bar{\mathbf{v}}_i$ and the current attribute vector $\mathbf{v}_i$, through the update function $\gamma_{\Theta_\gamma} = \Theta_\gamma(\bar{\mathbf{v}}_i + \mathbf{v}_i)$.

We refer readers to Figure 2 in *Choi and Kumar* [2024] for an illustration of the message passing concept in the GNN context. After message passing, the graph is updated to $G' = G(\mathbf{V}'_{FE}, \mathbf{E}'_{FE}) = G(\mathbf{v}'_i, \mathbf{e}'_{i,j})$. To propagate the information further into the network, multiple message passing steps can be used, where the involved “hyperparameter” choices affect model prediction performance and computational speed. In a supervised learning setup, learnable parameters are optimized through training.

2.3. The surrogate, GNS model

On top of the GNNs is the GNS architecture to learn and predict earthquake rupture state evolution. We adapt an open-source, mesh-based GNS, *MeshNet* [Choi and Kumar, 2024; Kumar and Choi, 2023; Kumar and Vantassel, 2023] and proceed to present details on GNS input, as well as the encoder, processor, decoder, and updater, as shown in Figure 2.

2.3.1 Input

The input to the GNS, $\mathbf{s}_i^t \in S^t$, is a vector containing current slip rate $\dot{\mathbf{u}}_i^t$ and on-fault location along strike and dip, $\mathbf{x}_i^t = [\mathbf{x}_{i,strike}^t, \mathbf{x}_{i,dip}^t]$, respectively, for all vertices. The current state is $\mathbf{s}_i^t = [\dot{\mathbf{u}}_i^t, \mathbf{x}_i^t]$. *Sanchez-Gonzalez et al.* [2020] show that for mesh-based GNS, only the current state should be used to predict the next one, without the need for memory of past states; this is different from the particle-based GNS examples for granular flow [Choi and Kumar, 2024]. Vertex attributes can be expanded to include other

features such as initial shear stress level, τ_i^t , or frictional parameters, pore pressure, or different frictional laws.

2.3.2 Encoder

The vertex and edge encoders, ε_{θ}^v and ε_{θ}^e converts $\mathbf{s}_i^t$ into the vertex and edge attribute vectors $\mathbf{v}_i^t$ and $\mathbf{e}_{i,j}^t$ and embed them into a latent graph $G_0 = (\mathbf{V}_0, \mathbf{E}_0)$ where $\mathbf{v}_i^t \in \mathbf{V}_0$ and $\mathbf{e}_{i,j}^t \in \mathbf{E}_0$.

$$\mathbf{v}_i^t = \varepsilon_{\theta}^v(\mathbf{u}_i^t), \quad \mathbf{e}_{i,j}^t = \varepsilon_{\theta}^e(\mathbf{x}_i^t) \quad (4)$$

A two-layered, 128-dimensional multi-layer perceptron (MLP) is used for the ε_{θ}^v and ε_{θ}^e , respectively. The vertex encoder ε_{θ}^v uses only slip rates $\mathbf{u}_i^t$ to make predictions without the current position information. The updated positions of vertices are predicted using the last positions and slip rates by an explicit Euler step.

2.3.3 Processor

Starting from the initial graph G_0 , the processor handles message passing (eqs. (1) to (3)) for M steps (*i.e.*, message passing steps; $G_0 \rightarrow G_1 \rightarrow \dots \rightarrow G_M$) and returns the updated graph G_M . We use two-layered 128-dimensional MLPs for both the message construction function, $\phi_{\theta_{\phi}}$, and vertex update function, $\gamma_{\theta_{\gamma}}$, and element-wise summation for the message aggregation in eqs. (1-3).

2.3.4 Decoder

The decoder δ_{θ}^v extracts the dynamics $\mathbf{y}_i^t \in Y_t$ from the vertices $\mathbf{v}_i^t$ using the final graph G_M following $\mathbf{y}_i^t = \delta_{\theta}^v(\mathbf{v}_i^t)$. A two-layer 128-dimensional MLP is used for δ_{θ}^v .

2.3.5 Updater

The dynamics $\mathbf{y}_i^t$, analogous to slip acceleration, are used to predict the next time-step slip rate, which is updated using Euler integration to get the next time-step location $\mathbf{x}_i^{t+1}$. As noted by *Choi and Kumar* [2024], this updater imposes the inductive biases of an inertial frame to force the GNS to learn the interaction dynamics to improve learning efficiency. A traditional neural network must learn both the update function and the interaction dynamics. Ideally, this implies that the GNS learns the actual dynamics, *i.e.* a coarse-grained version of local physical interactions, regardless of absolute vertex

positions. This approach permits the GNS to generalize beyond the training fault plane size, hypocenter locations, stress levels and distributions, as will be shown below.

The task then is to predict the next state, $S_{t+1}(\dot{\mathbf{u}}, \mathbf{x})$, from the current, $S_t(\dot{\mathbf{u}}, \mathbf{x})$, using the GNS (Figure 2A). Following *Sanchez-Gonzalez et al. [2020]* and *Choi and Kumar [2024]*, Figure 2B shows the structure of the GNS which uses a learned dynamic model, d_θ , to predict dynamics, Y_t , analogous to slip acceleration, from an input state, S_t , and the updater to estimate the next state, S_{t+1} . Figure 2C shows the structure of d_θ , which contains an *encoder* to construct a graph G_0 from the input state S_0 , a *processor* using message passing [Gilmer et al., 2017] through M GNNs to update graphs, and a *decoder* to extract dynamics information, Y_t , from the last graph, G_M .

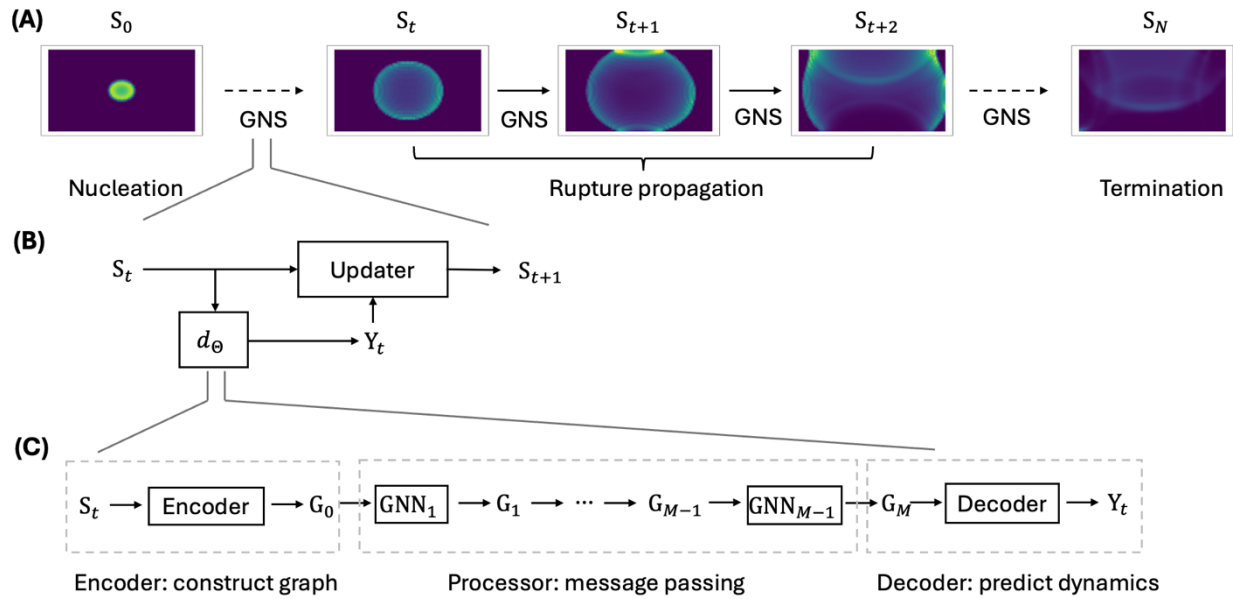


Figure 2. Concept of using Graph Network-based Simulator (GNS) to predict earthquake rupture dynamics with nucleation, rupture propagation, and termination, and the GNS architecture (modified from *Choi and Kumar [2024]* and *Sanchez-Gonzalez et al. [2020]*). **A)** With an initial state S_0 as input, the GNS predicts the next state and autoregressively generate state predictions for entire ruptures. **B)** The GNS predicts dynamics, Y_t , using the learned dynamics model, d_θ , and then uses an updater to predict next state, S_{t+1} . **C)** The structure of the learned dynamics model d_θ , which includes an encoder to construct graph G_0 from input state S_0 , a processor using

message passing through M Graph Neural Networks (GNNs) to update graphs, and a decoder to extract dynamics information Y_t from the last graph G_M .

2.4 Reference datasets

We prepare two datasets to train the GNS focusing on different objectives and a third dataset to test GNS performance on more complex cases. Dataset D1 has homogenous initial shear stress of 40 MPa on the fault but variable hypocenter locations. Our goal is to test whether the GNS can predict the right rupture dynamics even though hypocenter locations are different from those in the training data. Figure 1B shows the distribution of hypocenters for rupture scenarios in the training, validation, and testing sets, each of which contains 10, 3, and 6 scenarios, respectively. Figures 1C and D show final slip distributions and rupture front arrival time contours from two scenarios in the training set as an example.

Dataset D2, in addition to various hypocenters, contains heterogeneous initial stress cases, here initially chosen for simplicity to be in the form of 4×4 km asperities at 35, 45, 50, and 55 MPa stress levels, representing 10% of all the fault vertices. Figures 3A-E show the five hypocenter locations and associated locations of asperities with 35/55 MPa initial stress in the training set for M2, which totals at 30 scenarios. Figure 3F shows hypocenters and stress asperities of selected scenarios in the test set, whose asperity stresses (45/50 MPa) differ from those in the training set. We want to see if the GNS trained on D2 can generalize to unseen stress amplitudes and asperity locations, in combination with unseen hypocenter locations, and predict the associated modulations of rupture speed. To explore the effect of the size of training set, we also train model M3 on additional scenarios from dataset D2; 37 scenarios are selected for each asperity stress level, which has five hypocenter locations. With four asperity stress levels, the total number of scenarios in the training set is 148 scenarios.

To test the limits of the GNS' generalization capability, we build another Dataset D3 with 15 scenarios using fractal-type, irregularly distributed, "rough" initial stress. There are five scenarios with different hypocenter locations for each roughness R of 0.1, 0.5, and 0.9, using the diamond square algorithm (Fournier et al., 1982), for initial stresses as shown in Figures 11A, 12A, and 13A. We test GNS M2 and M3 on D3 to check if the GNS can capture more complex stress distributions based on learning relatively simple,

and different spatial character, features in the training set, and examine the impact of training set size.

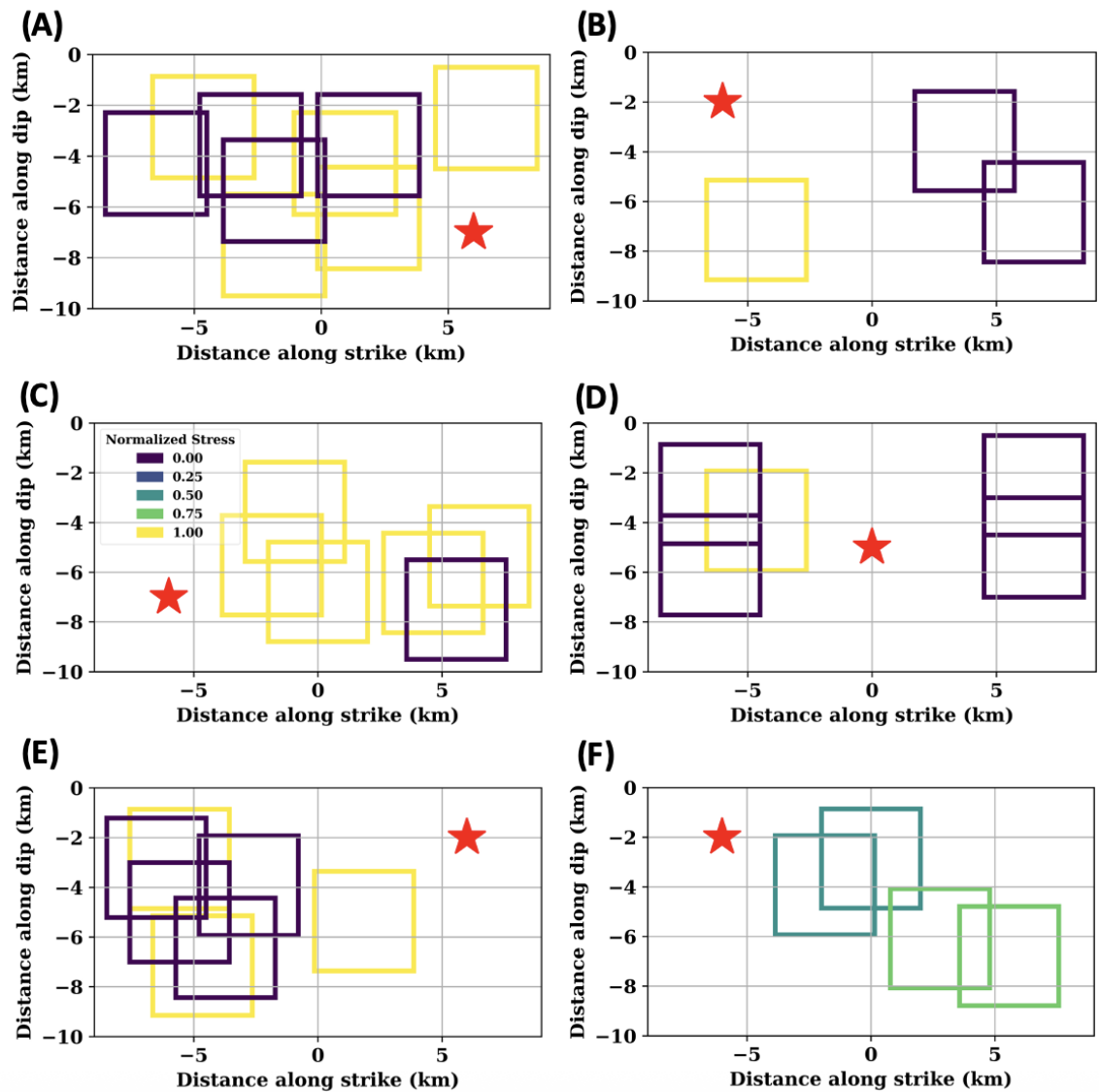


Figure 3. Hypocenter and stress anomaly (*i.e.*, asperity) locations (boxes) and normalized asperity stress levels at 0, 0.25 (outside of asperity), 0.5, 0.75, and 1, which linearly corresponds to 35, 40 (outside of asperity), 45, 50, and 55 MPa. A)-E) 30 scenarios used in the training set, which contains only normalized asperity stress levels at zero and unity. F) Initial stress setups in selected scenarios from the test set with normalized stress levels at 0.5 and 0.75, which are unseen in the training set.

2.5 Training and prediction

We train the GNS' learnable parameter sets to minimize the mean square error (MSE) between predicted normalized slip acceleration proxy $\mathbf{y}_i^t$ and normalized ground truth slip acceleration $\dot{\mathbf{u}}_t$, computed by derivatives of slip rate $\dot{\mathbf{u}}_t$, for all the fault vertices based on a standard gradient-based optimizer, *Adam* [Kingma and Ba, 2014] run over training time steps and epochs. GNS performance is affected by the choices of hyperparameters, such as learning rate, number of message passing steps in GNNs, and noise levels. We start from hyperparameters optimized for granular dynamics [Choi and Kumar, 2024] including the number of message passing steps of 10, learning rate of 10^{-4} , batch size of 2, and Gaussian distributed noise with standard deviation of 0.02 m/s (Table 2). These hyperparameters achieve a good balance between predicting performance and training cost for our problem as well [cf. Pfaff et al., 2020]. We also explore the effect of hyperparameter choices (Table 2) on training and prediction performance (Section 3.7).

During the prediction phase of the GNS, *i.e.*, *rollout*, we take an initial state which contains relatively small slip rate distributions (with peak ~ 1 m/s) inside the nucleation zone at $t = 1.2$ s simulation time, and the GNS then autoregressively generates the entire time sequence of earthquake rupture.

Table 2. GNS Hyperparameters

Model	M1/M2	M3
Learning rate	10^{-4}	3×10^{-5}
Batch number	2	8
Number of message passing steps	5/10/15	5/10/15
Gaussian noise level	0.02/0.005	0.02/0.005
Number of hidden layers in multi-layer perceptron	2	2
Dimension of hidden layers in multi-layer perceptron	128	128

Figures 4A, 4B, and 4C show the training and validation loss curves for GNS M1, M2, and M3 trained on Datasets D1 (10 scenarios), D2 (30 scenarios), and D2 (148 scenarios), respectively. For M1, the training and valid loss show rapid descent until 1 million steps and a slower decrease afterward. There is no apparent overfitting (where we would

expect validation loss increase and training loss to keep decreasing) over the whole training process. However, we do not expect significantly improved performance after 3 million steps. We thus consider the GNS for M1 trained at 3 million steps.

For M2, the training loss follows a similar trend to that in M1, with improvements diminishing after 1 million steps. Again, there is no sign of overfitting, and we choose the GNS for M2 trained at 3 million steps. The strong oscillations and training and validation losses reflect the fast learning rate of 10^{-4} used. For M3 with a lower learning of 3×10^{-5} and larger batch size of 8, the oscillations of losses over training steps are smaller. We choose M3 trained at 2.7 million steps as the training improvement diminishing with more steps. Interestingly, the strong oscillations appear not affect GNS performance. We also train M2 at a same slower learning of 3×10^{-5} , but it appears not affect the performance and oscillations. Adaptive learning rates or alternative optimizer choices may help suppress the oscillations and improve training stability or convergence speed.

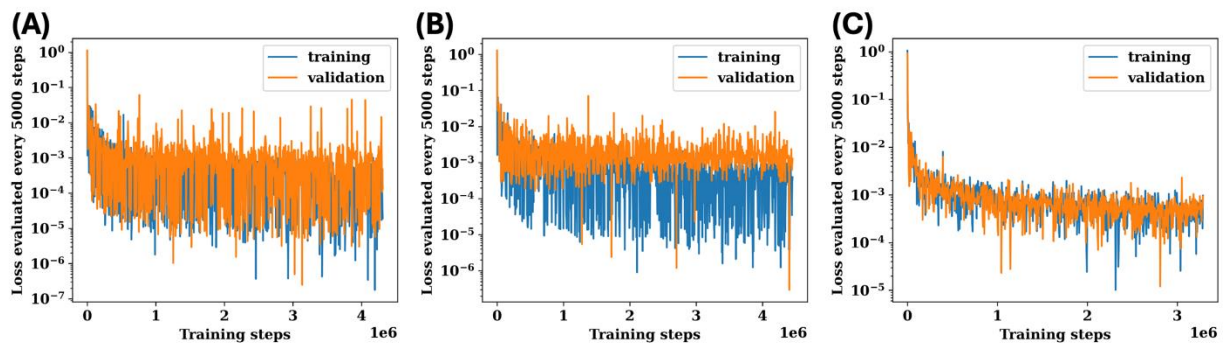


Figure 4. Training and validation loss for dataset M1 (A), M2 (B), and M3 (C) tabulated every 5000 steps.

3. Results

3.1 Predicting rupture dynamics from unseen hypocenters

We applied the M1 trained GNS to predict rupture propagation from unseen hypocenter locations. For this, we provide an initial state with slip rate distribution near the hypocenter and the GNS then generates the entire time sequence of rupture states autoregressively. Figure 5 shows rupture time contour comparisons between GNS

predictions and PBNS ground truth from four scenarios. We report three metrics, root mean square error of slip rate time series over all the fault vertices (SR RMSE), root mean square error of rupture arrival times over all the fault vertices (RT RMSE), and the difference between predicted and ground truth event moment magnitude, M_w , respectively.

The GNS predicts rupture front arrivals well overall, showing RT RMSE of 0.04-0.06 s for Figure 5A-C, and SR RMSE of 0.33-0.45 m/s, where peak slip rates are in the range of 20-30 m/s. Figure 5D shows a scenario where rupture terminates early close to the bottom of the fault, where rupture time contours merge. The GNS prediction deteriorates where ruptures terminate early, showing RT RMSE of 0.47 s and SR RMSE of 0.81 m/s. Early rupture termination is a rare feature in the training set; this indicates that the GNS' performance gets worse for sparsely represented features. In terms of event magnitudes, all four models perform well, showing magnitude errors between 0.01 to 0.07. This means the GNS learns to preserve the total earthquake moment, although some rupture details for early termination scenarios are not well captured.

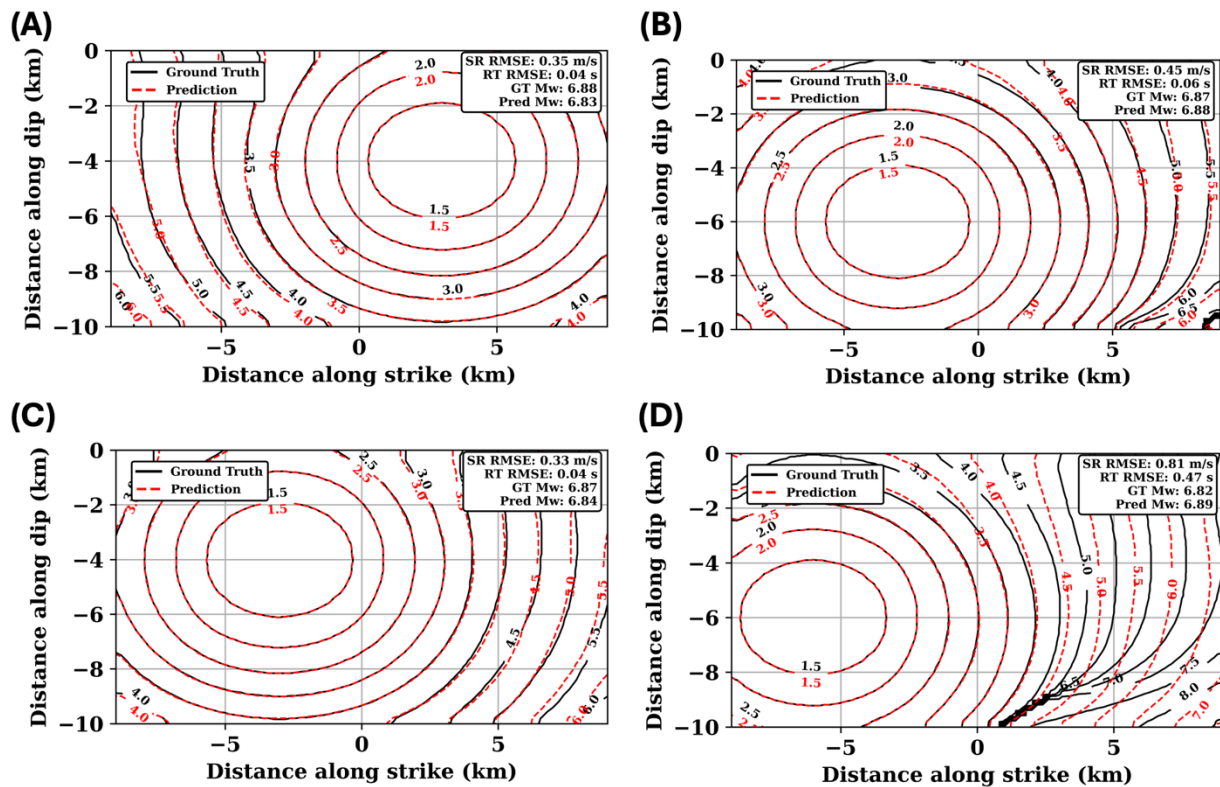


Figure 5. Rupture time contour comparisons between GNS predicted dynamic ruptures (red) and PBNS ground truth (black) for four cases with hypocenter locations unseen in

the training set. Numbers are seconds since initiation when the rupture front arrives at that location and slip rate rises above 0.1 m/s. SR RMSE: Slip rate root mean square error for the slip rate time series on all fault vertices between GNS prediction and ground truth. RT RMSE: Rupture time root mean square error for rupture arrival times on all fault vertices between GNS prediction and ground truth. GT Mw: Ground truth event moment magnitude. Pred Mw: GNS predicted magnitude.

Figures 6A-D show slip rate snapshots at times $t = 2.2, 3.2, 4.2$ and 5.2 s after rupture initiation for scenario Figure 5A. The GNS can predict the location and amplitude of the localized high slip rate rupture fronts well. Early into the rupture, the nucleation phase is captured accurately (Figure 6A), and Figure 6B shows that the reflection from the free surface and right boundary are also modeled accurately. Later into the rupture, the weaker reflection from the bottom of the fault and the surface reflection traveling to the middle depth of the fault are likewise captured (Figure 6C), along with the termination phase of rupture where slip rate drops to zero (Figure 6D). Moreover, the GNS can predict the termination of ruptures with zero slip rate across the fault (not shown).

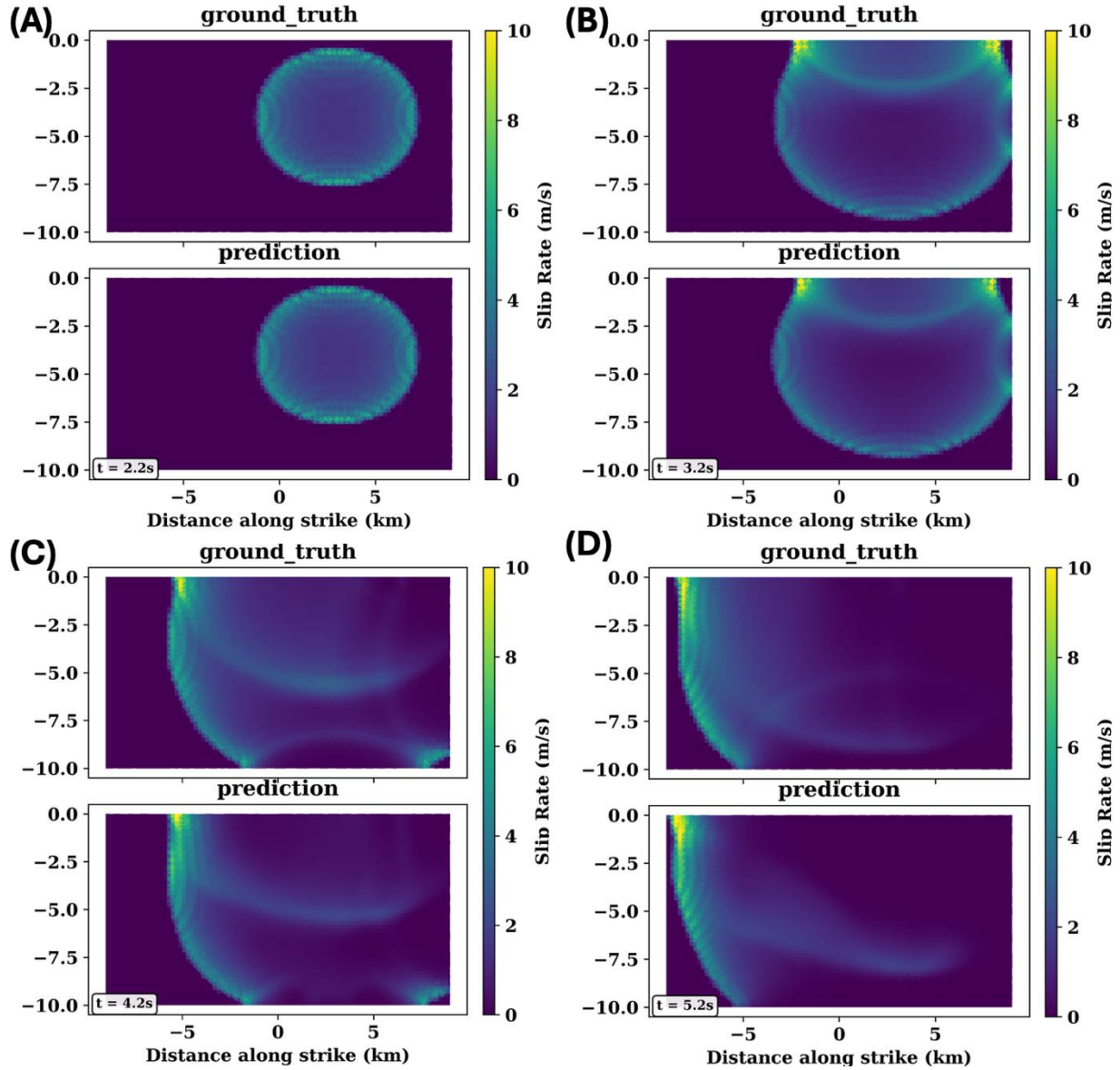


Figure 6. Slip rate snapshots of GNS prediction and PBNS ground truth at times $t = 2.2$, 3.2, 4.2, and 5.2 s after rupture initiation.

3.2 Predicting rupture dynamics from unseen stress heterogeneity

In a first step towards increased complexity, we explore how flexibly the structure of learned interactions can accommodate spatial variations in initial stress conditions. For this, we first apply the trained GNS M2 to scenarios with a single asperity of both unseen location and stress levels of 45/50 MPa (0.5/0.75 normalized stress). Figure 7 shows rupture time contour comparisons, together with SR RMSE, RT RMSE, and magnitude errors. Remarkably, to us, the GNS captures both the arrivals of rupture

fronts at asperities and amplitudes of accelerations from these unseen stress levels and locations, showing SR RMSEs in the range of 0.42-0.62 m/s and RT RMSEs of 0.05-0.09 s, and magnitude errors no larger than 0.05. This indicates that heterogeneous material properties can be incorporated into the learned network. Should hold more broadly, a wide range of classes of heterogeneity, from fault roughness to friction parameters, could be considered within the GNS framework.

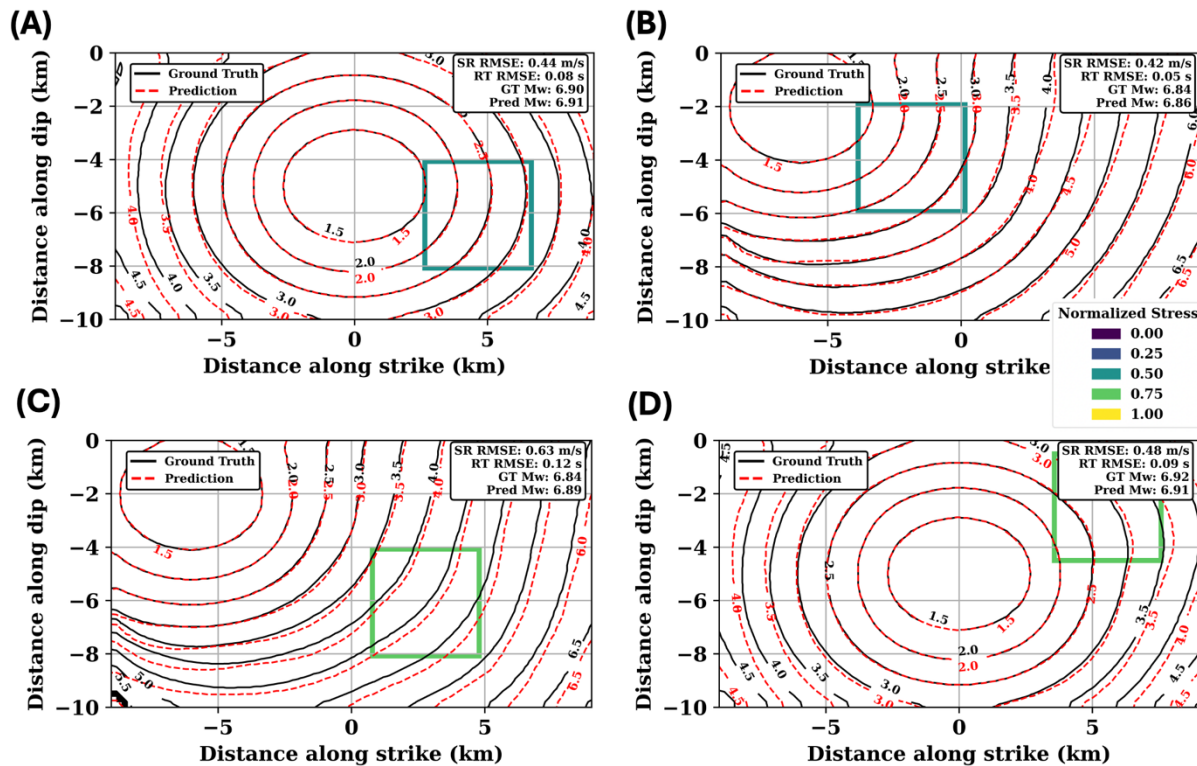


Figure 7. Rupture time contour comparisons between GNS predicted dynamic ruptures (red) and PBNS ground truth (black) for four scenarios with unseen stress level and location of asperities (boxes) in the training set. Labels and error metrics else as in Figure 5, see there for description.

We then apply the GNS to scenarios with checkerboard-type multiple stress asperities, a more complex heterogeneity pattern, at 35/55 MPa stress with an unseen hypocenter. Figure 8 shows rupture time contour comparisons. The GNS captures rupture acceleration and deceleration caused by asperities reasonably well in both scenarios, showing SR RMSEs of 0.57 to 0.72 m/s and RT RMSEs of 0.11s and magnitude error no larger than 0.05.

Figures 9 and 10 show slip rate time series at asperity centers for models as in Figure 8A and B, respectively. The first major peaks in terms of amplitude and arrival time are well predicted by GNS. In Figure 9, stations along-strike and dip coordinates $(-1, -7)$ and $(3, -7)$ appear to miss the small, secondary peaks later in the rupture. However, a larger secondary peak is captured at station $(-1, -3)$. In Figure 10, all secondary peaks are well captured. This indicates that the GNS, as trained, may occasionally miss secondary peaks at specific sites, and derived products such as ground shaking from secondary phases would be underestimated. However, the dominant control on ground shaking, earthquake moments, are well predicted for both scenarios, as shown in Figure 8.

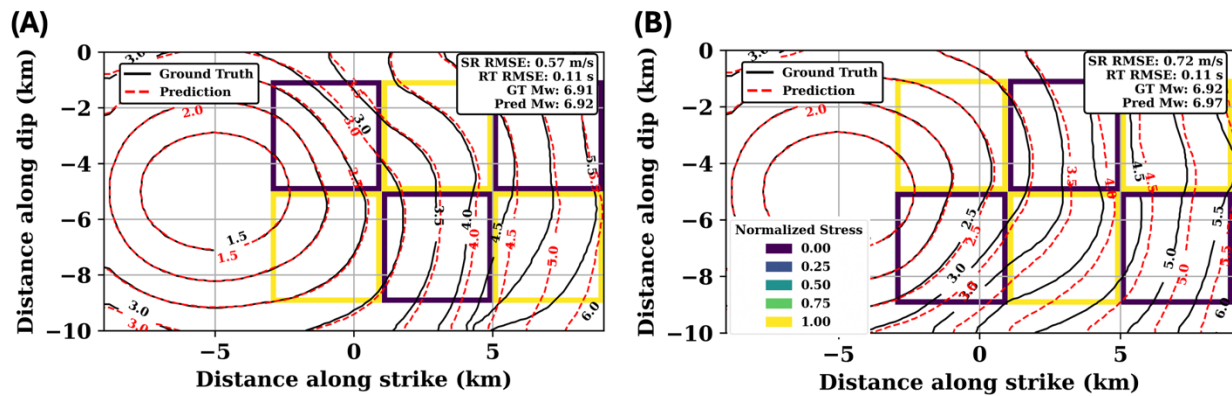


Figure 8. Rupture time contour comparisons between GNS predicted dynamic ruptures (red) and PBNS ground truth (black) for two scenarios with checkerboard multiple asperities not found in the training set. Labels and error metrics else as in Figure 5, see there for description.

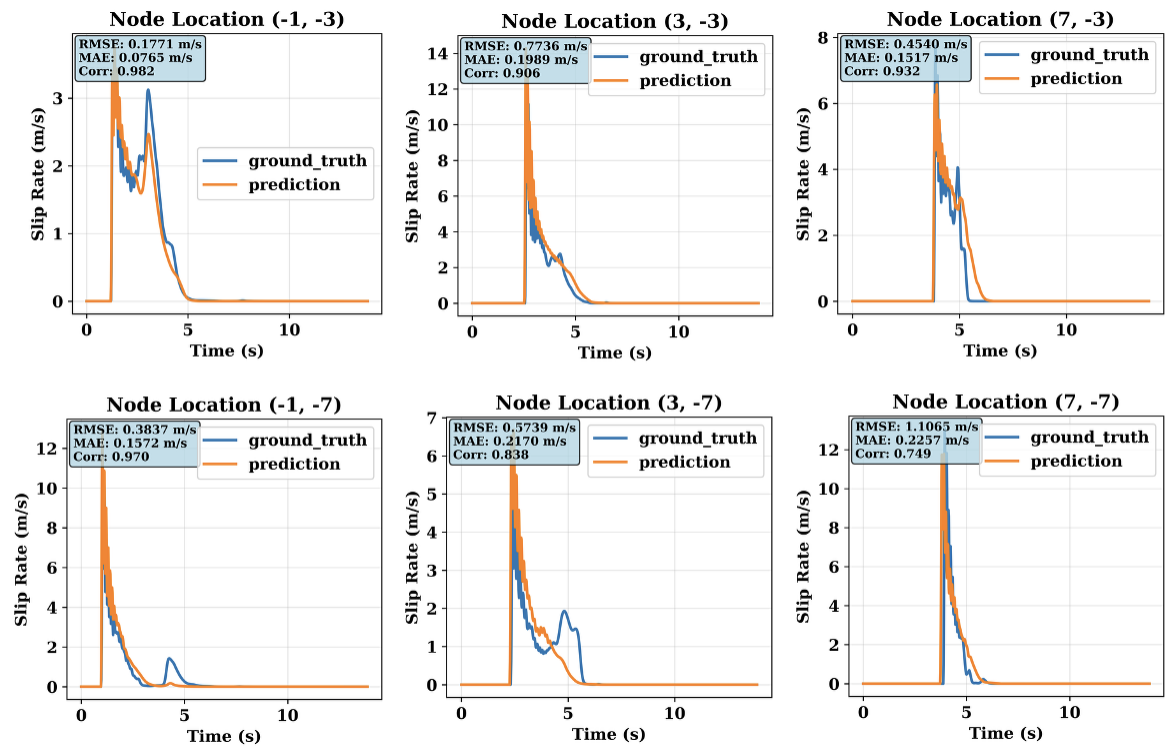


Figure 9. Comparisons of slip rate evolutions at the center of the asperities of the model shown in Figure 8A. Prediction is from the GNS, ground truth from the PBNS. RMSE: root mean square error for each time series; MAE: mean average error for each time series; Corr: correlation between the predicted and ground truth time series.

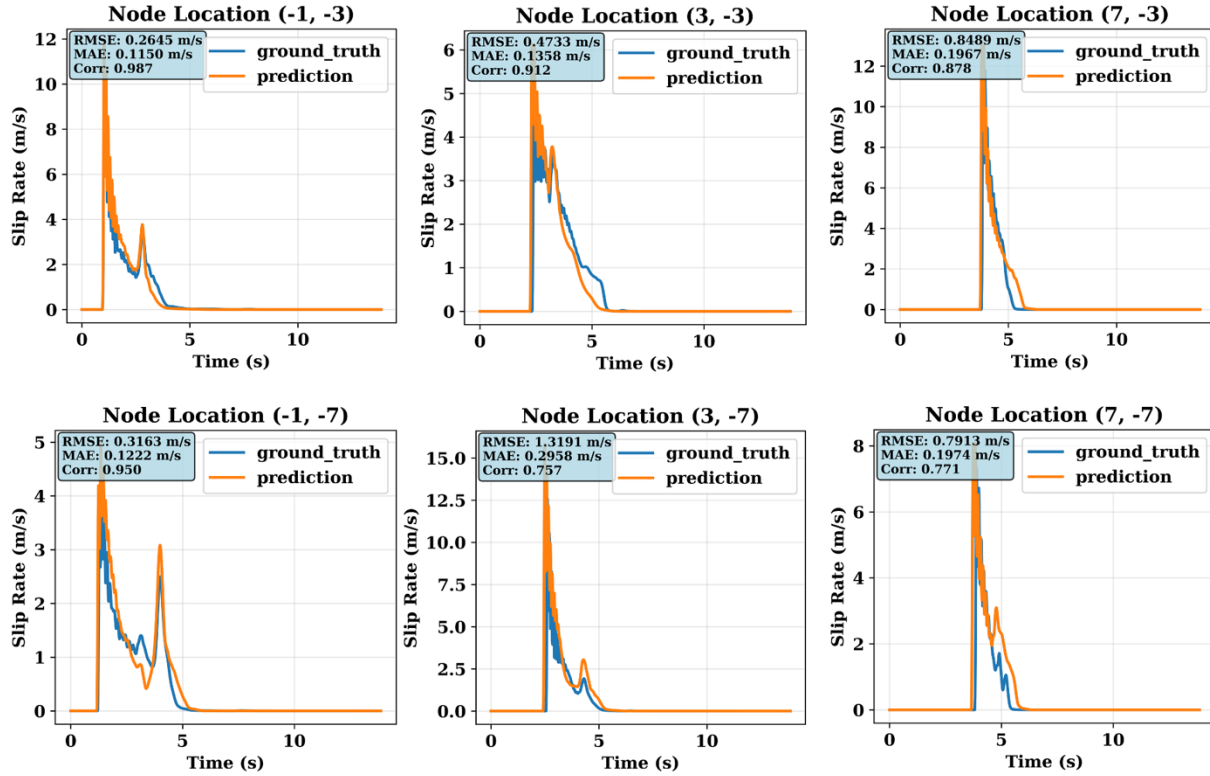


Figure 10. Comparisons of slip rate evolutions at asperity centers in the model in Figure 8B, else as in Figure 9.

3.3 Rupture dynamics predictions from unseen fractal initial stress.

To explore the limit of the GNS generalization capability further, we apply GNS M2, which is only trained on two end-member stress amplitudes with a single rectangular asperity, to fractal-type distributed stresses, with amplitudes varying from 35 to 55 MPa. Figure 11 shows rupture time contour comparisons between M2 predictions and PBNS ground truth for ruptures with unseen hypocenter locations, and initial stress levels and patterns with roughness in the diamond square algorithm of $R = 0.1$. For Figure 11B and 11C, the GNS predicts RT RMSE of 0.11 to 0.16s and SR RMSE of 0.57-0.58 m/s, and magnitude errors of 0.07 and 0.01. For Figure 11D, where early rupture termination occurs, GNS prediction deteriorates to SR RMSE of 0.92 m/s and RT RMSE of 0.95 s, but moment magnitude error is still within 0.06.

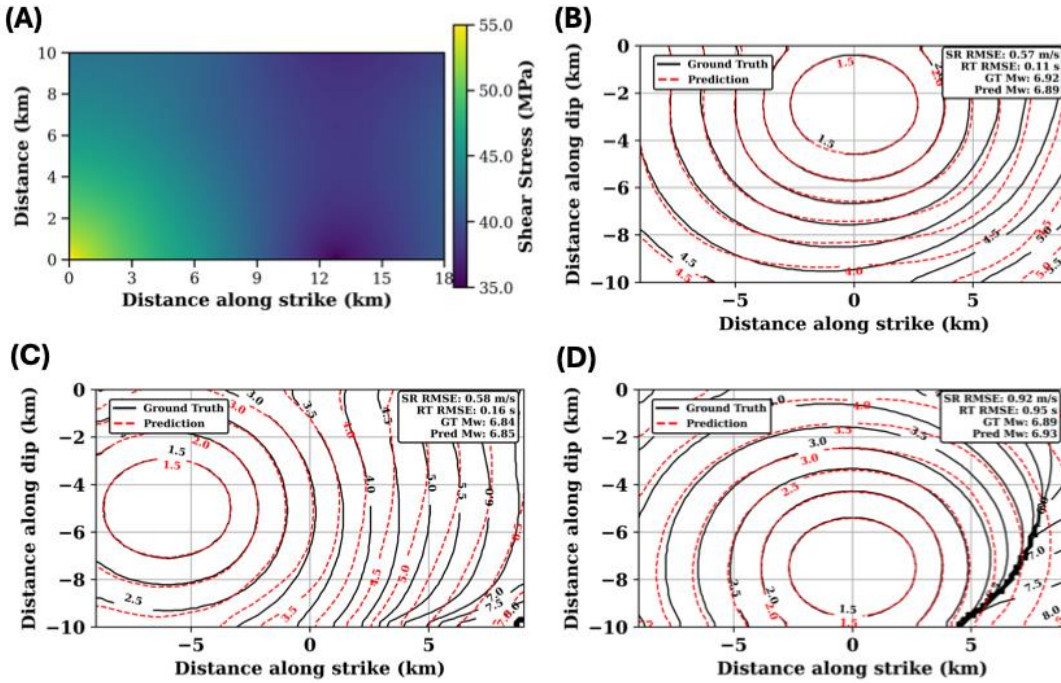


Figure 11. A) Smooth fractal-type initial stress with roughness in the diamond square algorithm of $R = 0.1$. B)-D) Rupture time contour comparisons between GNS M2 predicted dynamic ruptures (red) and PBNS ground truth (black) for three scenarios with unseen stress levels and complex patterns and hypocenter locations in the training set. Labels and error metrics else as in Figure 5, see there for description.

Figure 12 shows the initial stress distribution and rupture time contour comparisons for a rougher initial stress with $R = 0.5$. Error metrics SR RMSEs are 0.68 and 0.66 m/s and RT RMSEs are 0.07 and 0.11s for B and C, respectively. Figure 12D shows a larger RT RMSE of 0.38 s, and SR RMSE of 0.82 m/s. Again, the magnitudes are well predicted with errors no larger than 0.07.

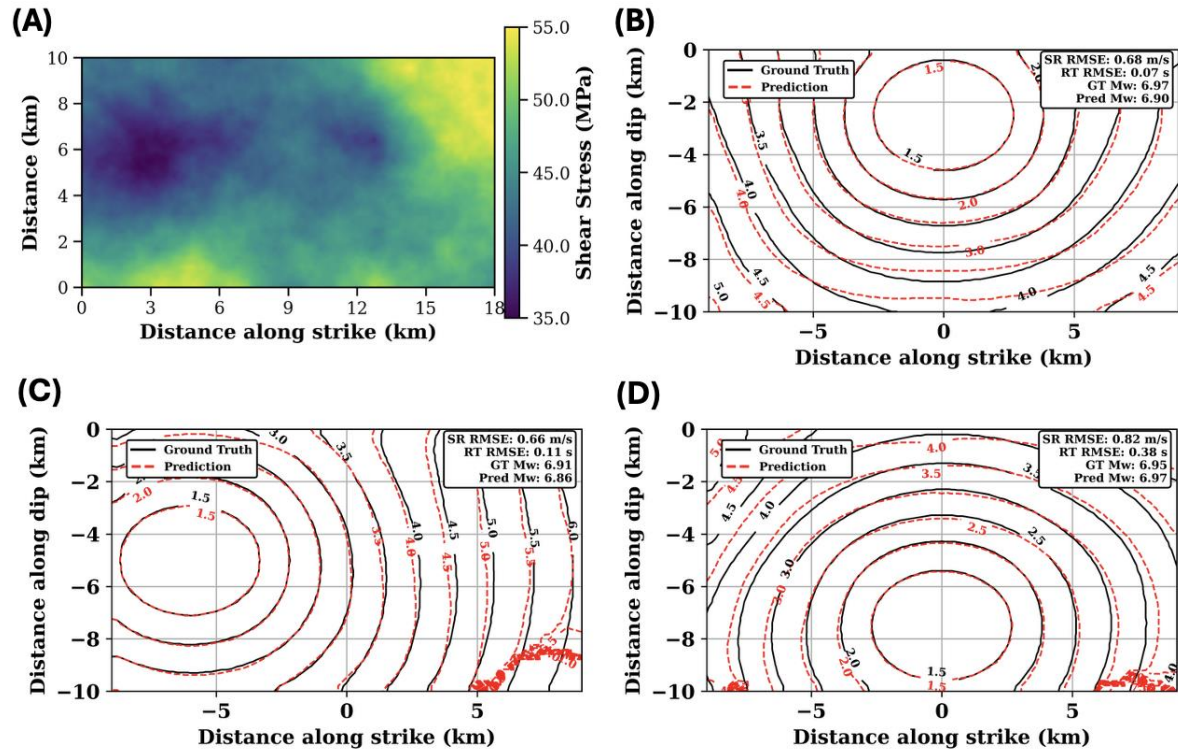


Figure 12. A) Fractal initial stress with intermediate roughness of $R = 0.5$. **B)-D)** Rupture time contour comparisons between GNS M2 predicted dynamic ruptures (red) and PBNS ground truth (black) for three scenarios with unseen stress levels and complex patterns and hypocenter locations in the training set. Labels and error metrics else as in Figure 5, see there for description.

For initial stresses with an even higher roughness of $R = 0.9$ (Figure 13A), M2 still predicts reasonable rupture dynamics and magnitudes (Table 4, D3 cases T10-T14), but the SR RMSE, RT RMSE, and magnitude errors get consistently larger. The results indicate that the features in the relatively small training set for M2 are not comprehensive enough in their spatial character to train the GNS for more complex dynamics. Using M3, a GNS with an enlarged training set with 148 scenarios, but still constant amplitude, rectangular asperities, the predictions improve significantly, with RT RMSE no larger than 0.11 s and SR RMSEs of 0.46 to 0.76 m/s (13B-D and Table 4). This indicates that there are no fundamental limitations with the GNS approach but that the spectral character of fault zone heterogeneity affects the training process and strategy.

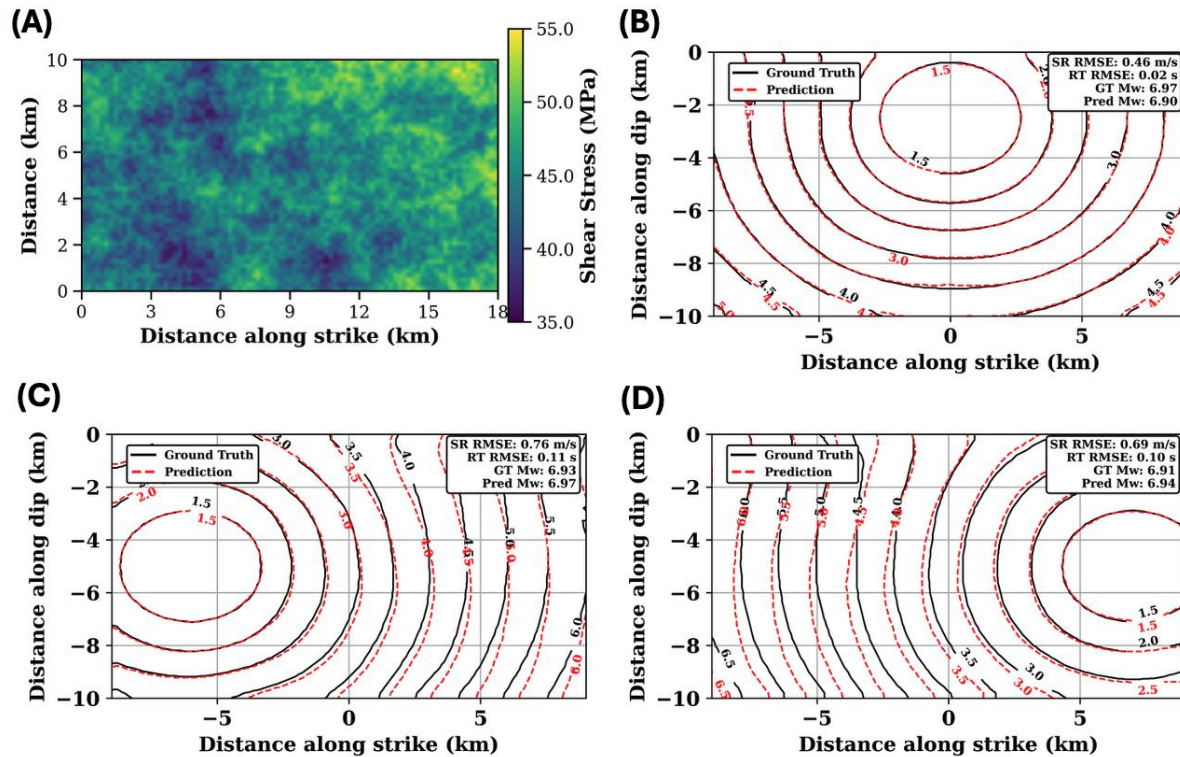


Figure 13. A) Fractal type initial stress with high roughness $R = 0.9$. B)-D) Rupture time contour comparisons between GNS M3 predicted dynamic ruptures (red) and PBNS ground truth (black) for three scenarios with unseen stress levels and complex patterns and hypocenter locations in the training set. Labels and error metrics else as in Figure 5, see there for description.

3.4 Generalization to larger fault geometry size.

To explore geometrical extrapolation capabilities, we apply GNS M1 to a fault of 40 km length along strike, more than twice the original 18 km training length. At the same spatial resolution, the larger fault model contains an additional 122% fault vertices. While prediction errors for rupture arrival times start to accumulate for long duration state prediction, the rupture time contours generally compare well with a RT RMSE of 0.12 s (Figure 14). The SR RMSE is 0.61 m/s, at a similar level to the scenarios with complex heterogeneous stress levels. The magnitude error is 0.09. This substantiates that the inductive nature of a GNS allows generalization to out of training model geometries [Pfaff *et al.*, 2020]. We also apply the GNS to a smaller fault size of 10 km by

5 km along strike and dip, respectively. The SR RMSE of 0.21 m/s, RT RMSE of 0.01 s, and magnitude error of 0.03 for D1.T_small (Table 3) show that the GNS can generalize to smaller fault geometry, confirming its geometrical interpolation capability.

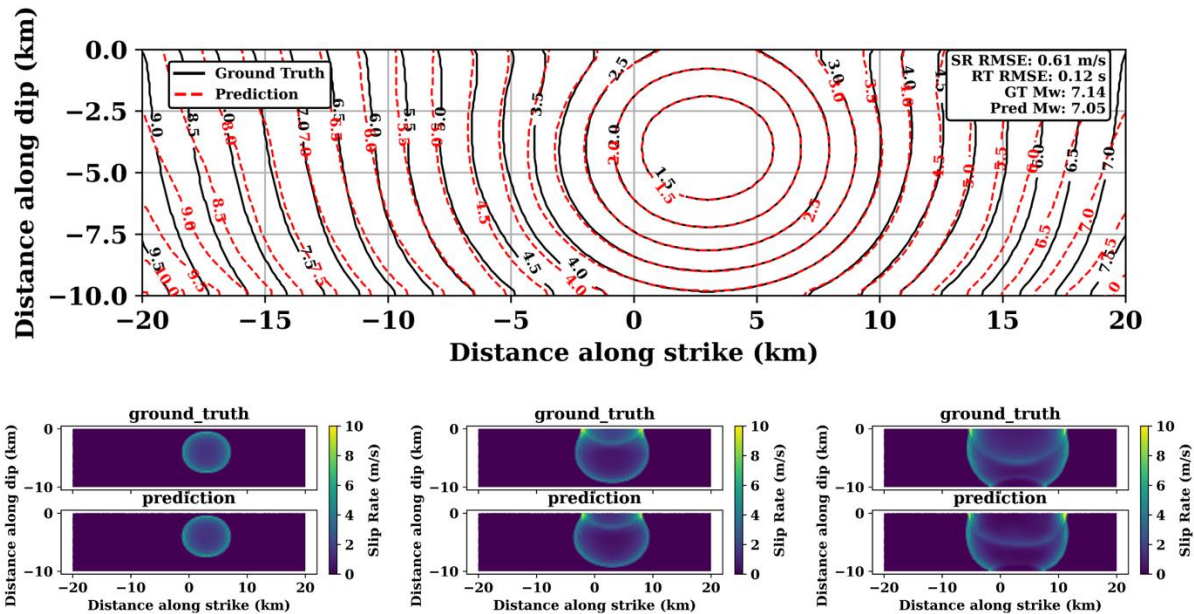


Figure 14. Rupture time contour comparisons between GNS predicted dynamic ruptures (red) and PBNS ground truth (black) for a larger fault exceeding the dimensions found in the training set. Contours and labeling as in Figure 5. Slip rate snapshot comparisons are shown for three time steps.

3.5 Backward compatibility

Model M2 is more general, or sophisticated, than M1 given the stress heterogeneity involved in training, and one might ask if M2 is overfit, or backward compatible with scenarios with less complexity, *e.g.*, homogenous initial stresses but different hypocenter locations. We apply M2 to the D1 test dataset and the larger fault test, and Table 3 shows misfit metrics for five testing scenarios. The errors are similar to the M1 case, which indicates that M2 is indeed backward compatible with models with less complexity.

Table 3. Root mean square error of slip rate time series over all fault vertices (SR RMSE) and root mean square error of rupture time arrivals (RT RMSE) and moment magnitude error between GNS predictions using M1 and M2 on dataset D1 and corresponding PBNS ground truths.

Model	M1			M2		
Metrics	SR RMSE (m/s)	RT RMSE (s)	M_w error	SR RMSE (m/s)	RT RMSE (s)	M_w error
D1.T0 (Fig 5A)	0.35	0.04	0.05	0.39	0.05	0.0
D1.T1 (Fig 5B)	0.45	0.06	0.01	0.49	0.17	0.01
D1.T2 (Fig 5C)	0.81	0.47	0.07	0.93	0.79	0.09
D1.T3 (Fig 5D)	0.33	0.04	0.03	0.41	0.05	0.01
D1.T_large (Fig 14)	0.61	0.12	0.09	0.51	0.07	0.05
D1.T_small	0.21	0.01	0.03	0.33	0.02	0.06

3.6 The impact of enlarged training set

We intend to keep the training set size as small as possible, and as shown in Section 3.3, M2 with 30 scenarios approaches its limit in predicting dynamic ruptures with high fractal-type initial stresses. Table 4 shows misfit metrics for M2 and M3 predictions with initial stresses of different fractal roughness and with hypocenter locations. For the scenarios with roughness $R = 0.1$, M2 is competitive to M3. However, with increased complexity of $R = 0.5$ and $R = 0.9$, M3 consistently performs better than M2. The results substantiate that the training set size needs to match the complexity of the prediction tasks.

Table 4. Root mean square error of slip rate time series over all fault vertices (SR RMSE) and root mean square error of rupture time arrivals (RT RMSE) and moment magnitude error between GNS predictions using M2 and M3 for dataset D3 with different fractal initial stresses and corresponding PBNS ground truths.

Model	M2			M3			Better model
Metrics	SR	RT	M_w error	SR	RT	M_w error	

	RMSE (m/s)	RMSE (s)		RMSE (m/s)	RMSE (s)		
Low stress roughness $R = 0.1$							
D3.T0	0.57	0.11	0.03	0.35	0.04	0.01	M3
D3.T1 (Fig 11B)	0.58	0.16	0.01	0.71	0.27	0.08	M2
D3.T3 (Fig 11C)	0.92	0.95	0.04	0.82	0.88	0.05	M3
D3.T4 (Fig 11D)	0.53	0.08	0.01	0.73	0.15	0.05	M2
Intermediate stress roughness $R = 0.5$							
D3.T5 (Fig 12B)	0.68	0.07	0.07	0.46	0.03	0.07	M3
D3.T6 (Fig 12C)	0.66	0.11	0.05	0.63	0.09	0.02	M3
D3.T7	1.08	0.37	0.01	0.88	0.13	0.07	M3
D3.T8 (Fig 12D)	0.82	0.38	0.02	0.88	0.12	0.06	M3
D3.T9	0.9	0.21	0.07	0.93	0.33	0.03	M3
High stress roughness $R = 0.9$							
D3.T10 (Fig 13B)	0.65	0.06	0.1	0.46	0.02	0.07	M3
D3.T11 (Fig 13C)	0.93	0.17	0.02	0.76	0.11	0.04	M3
D3.T12 (Fig 13D)	3.2	0.28	0.19	0.69	0.1	0.03	M3
D3.T13	0.99	0.13	0.09	0.82	0.11	0.02	M3
D3.T14	0.94	0.16	0.07	0.66	0.24	0	M3

539

540 *3.7 Sensitivity analyses of hyperparameters*

3.7.1 Message passing steps

We apply M2 to the D1 test case T_large using different message passing steps of 5, 10, and 15. Table 5 shows misfit metrics. The model with 5 message passing steps gives large SR RMSE and RT RMSE, which indicates that it is insufficient to capture the rupture dynamics. The predictions using models with 10 and 15 message passing steps match the ground truth well. Interestingly, performance of the model using 15 message passing steps is worse compared to that with 10 steps. This agrees with findings by *Choi and Kumar* [2024] and *Sanchez-Gonzalez et al.* [2020] for other physical systems. The results are consistent with the expectation that the physics for dynamic ruptures are dominated by local and short-range interactions, spanning around 10 grid cells for our experiments.

Table 5. Root mean square error of slip rate time series over all fault vertices (SR RMSE) and root mean square error of rupture time arrivals (RT RMSE) and magnitude error between GNS predictions using M2, with different message passing steps, for the large homogeneous test rupture and corresponding PBNS ground truths.

Model message passing steps /Metrics	SR RMSE (m/s)	RT RMSE (s)	M_w error
5	3.66	1.17	0.25
10	0.51	0.07	0.05
15	0.70	0.17	0.07

3.7.2 Gaussian noise level added in training

We explore two Gaussian noise standard deviation levels, 0.02 (default) and 0.005 m/s (0.2% and 0.05%, respectively, in terms of ~10 m/s slip rate at rupture fronts in this study). *Lam et al.* [2023] show that a proper level of noise added in the training trajectories are necessary to stabilize the GNS predictions. For M2 with 0.005 m/s noise, the predictions indeed fail for half of scenarios generating SR RMSE at ~100 m/s; this confirms that a ~0.2% level of noise, or so, is necessary for stable GNS performance.

3.7.3 Learning rate and batch size

For M1 and M2, which have 10 and 30 scenarios in the training sets, respectively, we use a fast learning rate of 10^{-4} and a small batch size of 2. Given the modest complexity involved, such learning rates and small batch size do not lead to overfitting. Although the training and validation losses oscillate widely, the model performance appears robust for most of the tasks. For M3, which has 148 scenarios involved, a batch size of 2 does not allow for enough complexity to be incorporated in each epoch, and the learning rate of 10^{-4} leads to overfitting and poor prediction. We thus use a learning rate of 3×10^{-5} and a batch size of 8, which significantly improves model performance (Table 4).

3.8 Computational performance

Table 6 shows computational cost per timestep of the GNS on a single Nvidia A100 GPU and the PBNS implemented by means of the *EQdyna* code, using 8 CPU cores on a workstation with AMD EPYC 7543, both architectures were released in 2020. The GNS run times are ~29-41 times faster than the ground truth simulations by *EQdyna*. This PBNS approach should be very efficient given the explicit time integration, under-integrated hexagons with hourglass control, and no solving of system of equations [e.g., Liu and Duan, 2018]. The GNS still achieves an order of magnitude, ~19-fold speedup compared to the PBNS for 200 m resolution models, and a ~41-fold for 100 m resolution, i.e. finer resolved models should benefit from the GNS even more.

However, a fair comparison between ML and high-fidelity type approaches is difficult. This is true in general, and here in particular. For example, we compare a PBNS *EQdyna* run using eight CPUs with the GNS which uses a single GPU. The GNS predicts dynamics on a 2-D surface, the fault interface, only, while *EQdyna* needs to solve the full physics equations for the 3-D volume surrounding the fault. A GPU-accelerated PBNS might achieve 10 fold speedup compared to single CPU [e.g., Premus et al., 2020], for example, a suggestion we cannot directly test as *EQdyna* currently only runs on CPUs. Either way, accepting what we suggest is roughly an order of magnitude speedup of the GNS compared to the PBNS for what might be a typical application is certainly not as striking an advantage as the performance reported for the mapping-type ML approaches. However, we note that we try to be conservative when stating speedup because we consider all time steps of the system evolution as provided by our GNS approach.

Tainpakdipat et al. (2025) use a single CPU PBNS computation as a benchmark and predict 76 time steps, subsampled 20-fold from the time steps of their PBNS. Gong et al. (2025) use 8 CPU to run their PBNS benchmarks but the ML surrogate only predicts the final time step for the rupture sequence, skipping 1500-4000 timesteps from the PBNS. Adjusting for the time steps predicted, and the CPU cores used to run the PBNS, our GNS speedup would nominally be ~ 3200 or $\sim 30,000$ -80,000 if computed in a similar way to Tainpakdipat et al. (2025) or Gong et al. (2025), with the caveats from above.

Moreover, the GNS appears quite general, with strong generalization capability, and requires relatively smaller training sets, compared to other ML approaches, where training costs can be significant. There are still ample opportunities to further optimize the GNS approach for enhanced speedup. For example, one could consider batch rollout of multiple scenarios and/or using mixed precision computing. Moreover, for GNS or other surrogate models, there is a trade-off between information yield and computational efficiency. Additional speedup could be achieved by sacrificing the information yield by, say, training the GNS on subsampled datasets. However, the advantage of the GNS is not just raw speedup for some end results, but the information yield of capturing the full spatiotemporal evolution of system states. Such outputs are valuable for uncertainty quantification and Bayesian inference, and the GNS approach is different from surrogate models that directly map input parameters to certain, restricted outputs, as noted.

Therefore, the GNS approach may still offer an efficient pathway to explore the high dimensionality of rupture dynamics parameter space and work well in real world applications. The GNS implementation is parallelized [Kumar and Vantassel, 2023] and can thus take advantage of the on-going investments in massively parallel GPU architectures. Table 6 shows the training cost of M2 and M3 per million training steps at 14.33 hours using a single Nvidia A100.

Table 5. Computational cost for GNS rollout and comparison to traditional PBNS.

Model grid size	GNS rollout per timestep (ms)	CPU run time per timestep (ms)	Speedup
200 m	~ 11	~ 322	~ 29.3

100 m	~35	~1449	~41.4
-------	-----	-------	-------

Table 6. Computational cost for training the GNS models.

Model	GNS training time per million steps using one Nvidia A100
M2/M3	14.33 hours

4 Discussion

While the overall generalization performance of our GNS is quite good, there are, of course, limitations when seeking to predict physical behaviors that are incompletely represented or absent from the training set. As expected, if the training set only contains scenarios with homogenous initial shear stress, as for D1, the GNS fails to predict rupture acceleration/deceleration caused by stress asperities, and we cannot expect to capture other effects, such as a different friction law, without dedicated training. For early termination of ruptures, scenarios which are sparsely represented in our training set, the GNS still predicts event magnitudes well, but performs poorly for rupture arrest as might be expected (e.g. Appendix A, testing scenario D3.T2). To allow the GNS to learn early rupture termination, or any physical features critical for future applications, the training dataset should be augmented with scenarios containing those desired features, or larger weights in loss functions could be applied when rare scenario samples are trained.

However, our experiments also show that the predictive power of a GNS can be incrementally improved by adding features such as heterogeneous initial stress to the training set. The GNS abstracted rules for propagating rupture appear to be automatically adapted in a suitable way by additional learning to capture the right sorts of regional interaction dynamics. We only train the GNS on modest complexity in selected parameters, such as at two asperity stress levels on a single rectangular patch, with only 30/148 scenarios. The GNS then generalizes to strongly spatially variable stress distributions, in combination with unseen hypocenter locations, and produces an estimate of the full spatiotemporal evolution of fault states.

Our GNS approach can be compared to the ML approaches of *Gong et al.* [2025] and *Tainpakdipat et al.* [2025] which are very efficient, but only provide end-to-end mappings, and rely on thousands or hundreds of thousands of training scenarios. *Gong et al.* [2025] show that their networks show larger errors for longer faults and more heterogeneous stress patches beyond the training set. This indicates that the GNS's generalization advantages may come with the cost of less speedup compared to the other approaches. More generally, different ML approaches show complementary strengths and weaknesses; it remains to be seen which use cases are suitable for what sort of surrogates, and how the GNS performance might be improved for specific applications.

Our results highlight how the GNS can extract a type of interaction "stencil" akin to a coarse-grained, local abstraction of the key physical relationships governing rupture (*i.e.*, conservation of momentum and mass plus constitutive law). This implies the potential to expand this analysis to a wider set of parameters and applications, such as how variations in pre-stress levels and spatial distribution affect the local interaction stencil. Since there is a link between the neural network and the physics through the interaction stencil, we might consider the GNS a candidate for a robust surrogate approach.

Additional features can be added to the GNS at selected levels, and extensions to a range of heterogeneity such as variations in frictional parameters, different types of frictional evolution laws, and fault roughness appear feasible using our framework. Further exploration of an expanded rupture dynamics parameter space is interesting, and the interactions captured by enhanced learning of a more general GNS might allow for model generalization to general classes of heterogeneities and constitutive laws.

Another pathway forward is to extend the current 2-D planar representation of fault geometry to fully networked fault systems by incorporating more fault geometric complexity of variations in fault dip and strike as vertex features. Representing 3-D fault network structures through a GNS 2-D graph projection may help reduce computational cost. In addition, opportunities also exist in understanding and then improving how the stencils are represented in the weights of the GNS networks, and what physical relationships result from different combinations of hyperparameters, such as message passing steps, entail to better emulate the real physical processes.

Currently, our GNS only predicts slip rate evolution on the fault interface, restricting its utility for seismic hazard assessment to source characteristics such as magnitude, final

slip distribution, and rupture duration. However, there is an opportunity to additionally incorporate wave propagation on the free surface for system states and use the GNS to predict rupture and ground shaking concurrently. Broadening the GNS to predict surface motions will further enhance its utility for seismic hazard assessment of ground shaking.

Efforts to build enhanced GNS surrogates, with parallelization, should reduce the computational costs. We speculate that applications may include surrogates to relate probabilistically represented fault zones to seismicity, and a more broadly trained GNS might represent one version of a much-needed comprehensive representation of our best guesses for the fundamental sets of multi-scale fault zone physics.

The GNS can then be a general, physics informed operator, generating likely rupture characteristics with quantified uncertainties based on incomplete, variable resolution, time-dependent, heterogeneous initial conditions and material parameters. While we are only concerned with the rupture dynamics problem here, a similar GNS approach should lend itself readily to other transient earthquake system problems such as fault network, post-seismic, and earthquake cycle stress heterogeneity evolution.

5 Conclusions

We present a Graph Network-based Simulator (GNS) for dynamic earthquake rupture. The GNS can predict rupture dynamics in comparable accuracy to high-fidelity physics-based simulations, in particular during the nucleation and rupture front propagation stage, with ~20-40 times per-time-step speedup compared to the physics-based modeling. The generalization capability of the GNS extends to scenarios with unseen hypocenter locations, stress asperity strength and geometry variations, and fault sizes. The GNS appears to learn a coarse-grained version of the physics governing local dynamics on the fault plane. This physics-informed structure of the GNS implies good robustness, as well as providing new perspectives on how to potentially explore different types of physical interactions of relevance for earthquake dynamics. With improved computational efficiency and expanded training, the GNS may be suited to serve as a comprehensive surrogate for future Bayesian model inference, uncertainty quantification, or optimal experimental design tasks. Those could cover rupture

dynamics applications within both a fundamental exploration of physics and an operational, time-dependent seismic hazard assessment context.

Acknowledgements

DL and TWB were partially funded by the National Science Foundation under grant EAR-2121666. DL was also supported by the Southern (now *Statewide*) California Earthquake Center (SCEC), and this is SCEC contribution no. 14197. SCEC is funded by National Science Foundation under Cooperative Agreement EAR-1600087 and the United States Geological Survey under Cooperative Agreement G17AC0047. We thank Dr. Mousavi and an anonymous reviewer for their comments on an earlier version which helped to improve the manuscript.

The authors do not have any conflicts of interest to declare.

Open Research

The code of the GNS for earthquake rupture dynamics, EQGNS, and the finite element software *EQdyna* are publicly available on GitHub and the Zenodo repository (Liu and Becker, 2025). Necessary scripts and configuration files used to generate *EQdyna* ground truth scenarios, training, validation, and testing datasets for GNS, and rollout results are archived under the Zenodo repository (Liu and Becker, 2025).

Appendix A. Predicting rare physical feature of early rupture termination.

Figure A1A shows the prediction of M1 on testing scenario D1.T3 where the rupture terminates early in the middle of the fault, where dense rupture curves merge, and the GNS, which is not trained to learn such rare physical dynamic pattern, fails to capture the termination. In this case, the prediction of magnitude errors at ~ 0.3 . Figure A1B shows the prediction of M3 on testing scenario D3.T2, where rupture terminates prematurely but ruptures coming from the top half of the fault keeps breaking the whole fault. The GNS predicts continuous ruptures where the ground truths show terminations. However, in this case, the magnitude is still well predicted.

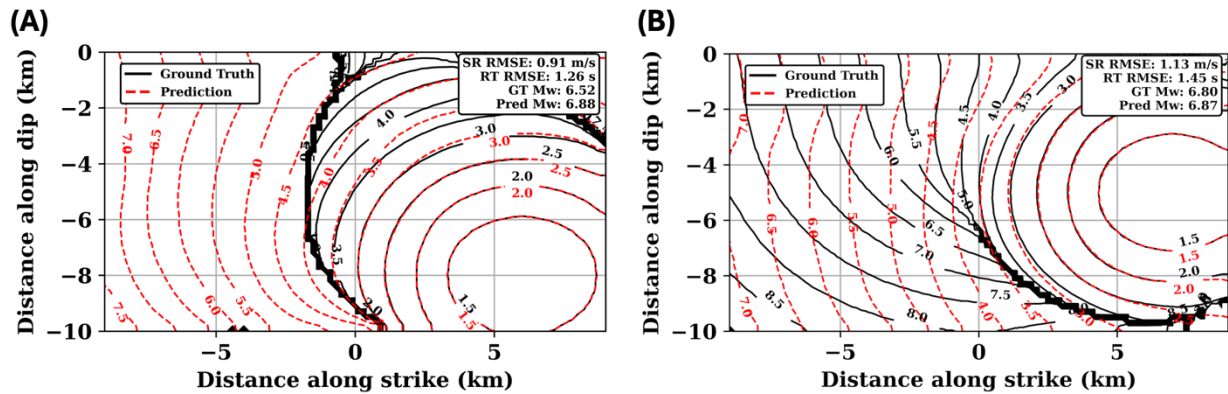


Figure A1. **A)** Rupture time contour comparisons between GNS M1 predicted dynamic ruptures (red) and PBNS ground truth (black) for testing scenario D1.T3 with unseen hypocenter locations in the training. **B)** Rupture time contour comparisons between GNS M3 predicted dynamic ruptures (red) and PBNS ground truth (black) for testing scenario D3.T2 with unseen stress levels and complex patterns and hypocenter locations in the training set. Labels and error metrics else as in Figure 5, see there for description.

References

- Andrews, D. J., and M. Barall (2011), Specifying Initial Stress for Dynamic Heterogeneous Earthquake Source Models, *Bulletin of the Seismological Society of America*, 101(5), 2408-2417, doi:10.1785/0120110012.
- Battaglia, P., R. Pascanu, M. Lai, and D. Jimenez Rezende (2016), Interaction networks for learning about objects, relations and physics, *Advances in neural information processing systems*, 29.
- Choi, Y., and K. Kumar (2024), Graph Neural Network-based surrogate model for granular flows, *Computers and Geotechnics*, 166, doi:10.1016/j.compgeo.2023.106015.
- Day, S. M. (1982), Three-dimensional simulation of spontaneous rupture: the effect of nonuniform prestress, *Bulletin of the Seismological Society of America*, 72(6A), 1881-1902.
- Dieterich, J. H. (1979), Modeling of Rock Friction .1. Experimental Results and Constitutive Equations, *Journal of Geophysical Research*, 84(Nb5), 2161-2168, doi:10.1029/JB084iB05p02161.
- Dieterich, J. H., and M. F. Linker (1992), Fault stability under conditions of variable normal stress, *Geophysical Research Letters*, 19(16), 1691-1694, doi:doi:10.1029/92GL01821.

772 Duan, B. (2012), Dynamic rupture of the 2011 Mw 9.0 Tohoku-Oki earthquake: Roles of
 773 a possible subducting seamount, *Journal of Geophysical Research*, 117(B5),
 774 doi:10.1029/2011JB009124.

775 Duan, B., and D. D. Oglesby (2006), Heterogeneous fault stresses from previous
 776 earthquakes and the effect on dynamics of parallel strike-slip faults, *Journal of*
 777 *Geophysical Research: Solid Earth*, 111(B5), doi:10.1029/2005jb004138.

778 Dunham, E. M., D. Belanger, L. Cong, and J. E. Kozdon (2011), Earthquake Ruptures
 779 with Strongly Rate-Weakening Friction and Off-Fault Plasticity, Part 2: Nonplanar
 780 Faults, *Bulletin of the Seismological Society of America*, 101(5), 2308-2322,
 781 doi:10.1785/0120100076.

782 Fournier, A., D. Fussell, and L. Carpenter (1982), Computer rendering of stochastic models,
 783 *Commun. ACM*, 25(6), 371-384, doi:10.1145/358523.358553.

784 Gallovič, F., L. Valentová, J. P. Ampuero, and A. A. Gabriel (2019), Bayesian Dynamic
 785 Finite-Fault Inversion: 1. Method and Synthetic Test, *Journal of Geophysical Research: Solid*
 786 *Earth*, 124(7), 6949-6969, doi:10.1029/2019jb017510.

787 Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl (2017), Neural
 788 Message Passing for Quantum Chemistry, in *Proceedings of the 34th International*
 789 *Conference on Machine Learning*, edited by P. Doina and T. Yee Whye, pp. 1263--1272,
 790 PMLR, Proceedings of Machine Learning Research.

791 Gong, Z., Z. Wang, C. Liang, A. Nienkötter, J. Wang, C. Ren, and X. Peng (2025),
 792 RuptureNet2D, a Deep Neural Network Based Surrogate for Dynamic Earthquake
 793 Rupture Simulation in Two Dimensions, *Journal of Geophysical Research: Solid Earth*,
 794 130(5), doi:10.1029/2024jb030069.

795 Harris, R. A., et al. (2018), A Suite of Exercises for Verifying Dynamic Earthquake
 796 Rupture Codes, *Seismological Research Letters*, 89(3), 1146-1162, doi:10.1785/0220170222.

797 Harris, R. A., et al. (2009), The SCEC/USGS Dynamic Earthquake Rupture Code
 798 Verification Exercise, *Seismological Research Letters*, 80(1), 119-126,
 799 doi:10.1785/gssrl.80.1.119.

800 Ida, Y. (1972), Cohesive force across the tip of a longitudinal-shear crack and Griffith's
 801 specific surface energy, *Journal of Geophysical Research*, 77(20), 3796-3805,
 802 doi:10.1029/JB077i020p03796.

803 Jia, Z., et al. (2023), The complex dynamics of the 2023 Kahramanmaraş, Turkey, Mw
 804 7.8-7.7 earthquake doublet, *Science*, 381(6661), 985-990, doi:doi:10.1126/science.adi0685.

805 Kingma, D. P., and J. Ba (2014), Adam: A method for stochastic optimization, *arXiv*
 806 *preprint arXiv:1412.6980*.

807 Kochkov, D., et al. (2024), Neural general circulation models for weather and climate,
 808 *Nature*, doi:10.1038/s41586-024-07744-y.

Kumar, K., and Y. Choi (2023), Cylinder flow with graph neural network-based simulator, edited, DesignSafe-CI, doi:10.17603/ds2-fzg7-1719.

Kumar, K., and J. Vantassel (2023), GNS: A generalizable Graph Neural Network-based simulator for particulate and fluid modeling, *Journal of Open Source Software*, 8(88), doi:10.21105/joss.05025.

Lam, R., et al. (2023), Learning skillful medium-range global weather forecasting, *Science*, 382(6677), 1416-1421, doi:10.1126/science.adi2336.

Liu, D., and B. Duan (2018), Scenario Earthquake and Ground-Motion Simulations in North China: Effects of Heterogeneous Fault Stress and 3D Basin Structure, *Bulletin of the Seismological Society of America*, 108(4), 2148-2169, doi:10.1785/0120170374.

Liu, D., B. Duan, K. Scharer, and D. Yule (2022), Observation-Constrained Multicycle Dynamic Models of the Southern San Andreas and the Northern San Jacinto Faults: Addressing Complexity in Paleoearthquake Extent and Recurrence With Realistic 2D Fault Geometry, *Journal of Geophysical Research: Solid Earth*, 127(2), e2021JB023420, doi:10.1029/2021JB023420.

Liu, D., and T. Becker (2025). Source code and dataset for research article "Earthquake rupture dynamics from Graph Neural Networks" [Software][Dataset]. In Earthquake rupture dynamics from Graph Neural Networks (1.0), Zenodo, doi: 10.5281/zenodo.17095311. (The reviewer link is https://zenodo.org/records/17095311?token=eyJhbGciOiJIUzUxMiJ9.eyJpZCI6ImY3YW RiZjFILWEzNWMTNDNmMC05N2M2LTZIYjc4NGM3NGZkYyIsImRhdGEiOnt9LCJyY W5kb20iOiI3YmJjZGZzOWFkOTYwMTI4ZThmNmMzYTUyNzZkMmY0MCJ9.3z2m9v 4m4xj8Ee-WordJ2_SnpxEtnkGh06IQZCvLfIFF2kEw4ubRQfx03w8f7ijU_Kz_TOByxMDOQpPUM mHZXQ. The GitHub repositories for *EQdyna* and *EQGNS* are <https://github.com/EQDYNA/EQdyna.git>, and <https://github.com/dunyuliu/EQGNS.git>, respectively.)

Luo, B., and B. Duan (2018), Dynamics of Nonplanar Thrust Faults Governed by Various Friction Laws, *Journal of Geophysical Research: Solid Earth*, 123(6), 5147-5168, doi:10.1029/2017jb015320.

Pfaff, T., M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia (2020), Learning mesh-based simulation with graph networks, *arXiv preprint arXiv:2010.03409*.

Premus, J., F. Gallovič, L. Hanyk, and A.-A. Gabriel (2020), FD3D_TSN: A Fast and Simple Code for Dynamic Rupture Simulations with GPU Acceleration, *Seismological Research Letters*, 91(5), 2881-2889, doi:10.1785/0220190374.

Price, I., et al. (2024), Probabilistic weather forecasting with machine learning, *Nature*, doi:10.1038/s41586-024-08252-9.

Ramos, M. D., Y. Huang, T. Ulrich, D. Li, A.-A. Gabriel, and A. M. Thomas (2021), Assessing Margin-Wide Rupture Behaviors Along the Cascadia Megathrust With 3-D Dynamic Rupture Simulations, *Journal of Geophysical Research: Solid Earth*, 126(7), e2021JB022005, doi:10.1029/2021JB022005.

Ruina, A. (1983), Slip Instability and State Variable Friction Laws, *Journal of Geophysical Research*, 88(Nb12), 359-370, doi:DOI 10.1029/JB088iB12p10359.

Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia (2020), Learning to simulate complex physics with graph networks, in *Proceedings of the 37th International Conference on Machine Learning*, edited, p. Article 784, JMLR.org.

Shaw, B. E., B. Fry, A. Nicol, A. Howell, and M. Gerstenberger (2022), An Earthquake Simulator for New Zealand, *Bulletin of the Seismological Society of America*, 112(2), 763-778, doi:10.1785/0120210087.

Shi, Z., and S. M. Day (2013), Rupture dynamics and ground motion from 3-D rough-fault simulations, *Journal of Geophysical Research: Solid Earth*, 118(3), 1122-1141, doi:10.1002/jgrb.50094.

Suhendi, C., B. Li, H. Vasyura-Bathke, J. Liu, S. Jónsson, and P. M. Mai (2025), Bayesian inversion and quantitative comparison for bilaterally quasi-symmetric rupture processes on a multisegment fault in the 2021 Mw7.4 Maduo earthquake, *Geophys J Int*, 240(1), 673-695, doi:10.1093/gji/ggae398.

Tainpakdipat, N., M. Abdelmeguid, C. Zhao, K. Azizzadenesheli, and A. Elbanna (2025), Fourier Neural Operators for Accelerating Earthquake Dynamic Rupture Simulations.

Taufiqurrahman, T., A. A. Gabriel, T. Ulrich, L. Valentová, and F. Gallovič (2022), Broadband Dynamic Rupture Modeling With Fractal Fault Roughness, Frictional Heterogeneity, Viscoelasticity and Topography: The 2016 Mw 6.2 Amatrice, Italy Earthquake, *Geophysical Research Letters*, 49(22), doi:10.1029/2022gl098872.

Thangamuthu, A., G. Kumar, S. Bishnoi, R. Bhattoo, N. Krishnan, and S. Ranu (2022), Unravelling the performance of physics-informed graph neural networks for dynamical systems, *Advances in Neural Information Processing Systems*, 35, 3691-3702.

Ulrich, T., A. A. Gabriel, J. P. Ampuero, and W. Xu (2019), Dynamic viability of the 2016 Mw 7.8 Kaikoura earthquake cascade on weak crustal faults, *Nat Commun*, 10(1), 1213, doi:10.1038/s41467-019-09125-w.

Wang, Y., and S. M. Day (2020), Effects of Off-Fault Inelasticity on Near-Fault Directivity Pulses, *Journal of Geophysical Research: Solid Earth*, 125(7), e2019JB019074, doi:<https://doi.org/10.1029/2019JB019074>.

Withers, K. B., K. B. Olsen, Z. Shi, and S. M. Day (2018), Validation of Deterministic Broadband Ground Motion and Variability from Dynamic Rupture Simulations of

882 Buried Thrust Earthquakes, *Bulletin of the Seismological Society of America*, 109(1), 212-228,
883 doi:10.1785/0120180005.
884 Zhu, W., K. L. Allison, E. M. Dunham, and Y. Yang (2020), Fault valving and pore
885 pressure evolution in simulations of earthquake sequences and aseismic slip, *Nature*
886 *Communications*, 11(1), 4833, doi:10.1038/s41467-020-18598-z.
887

Figure 1.

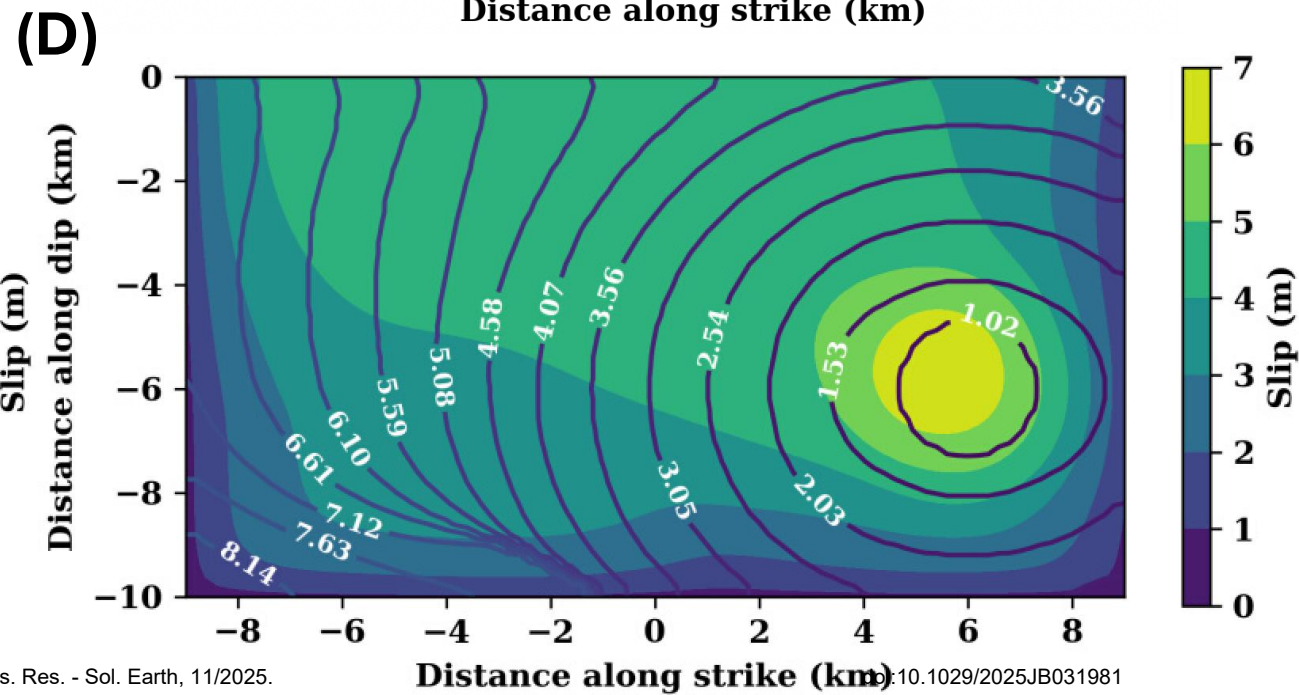
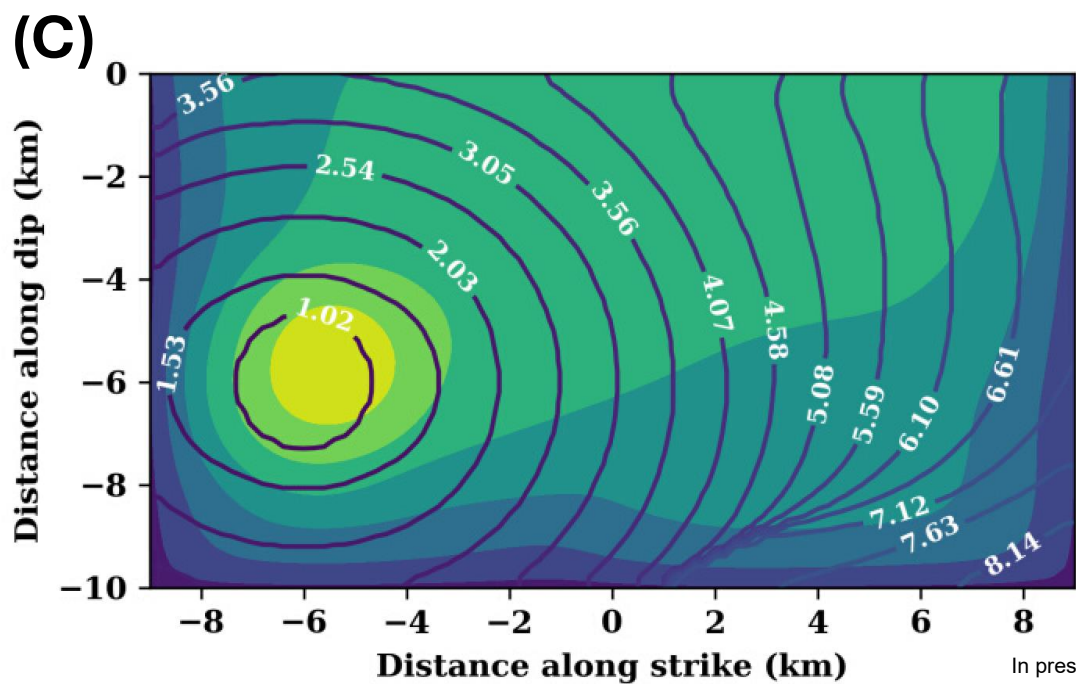
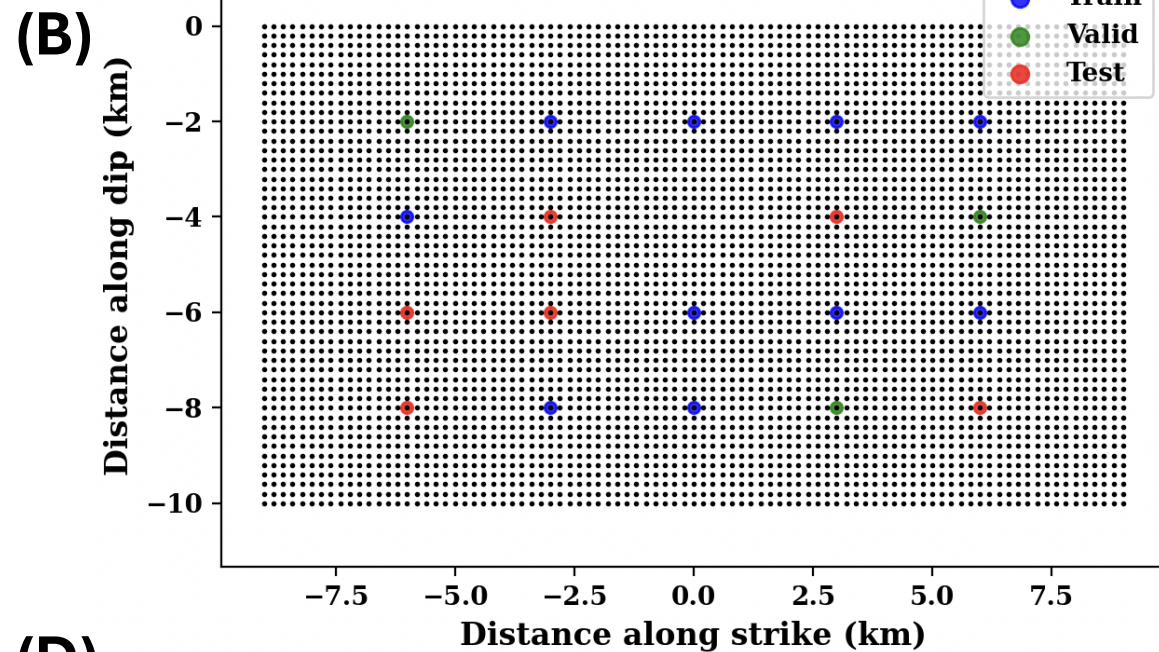
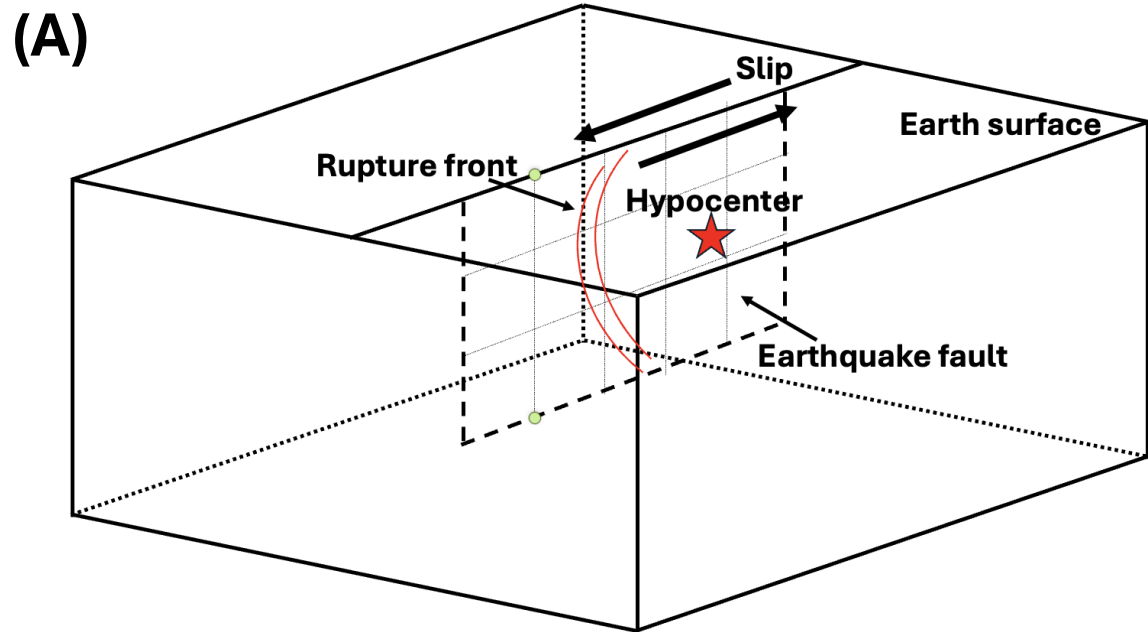


Figure 2.

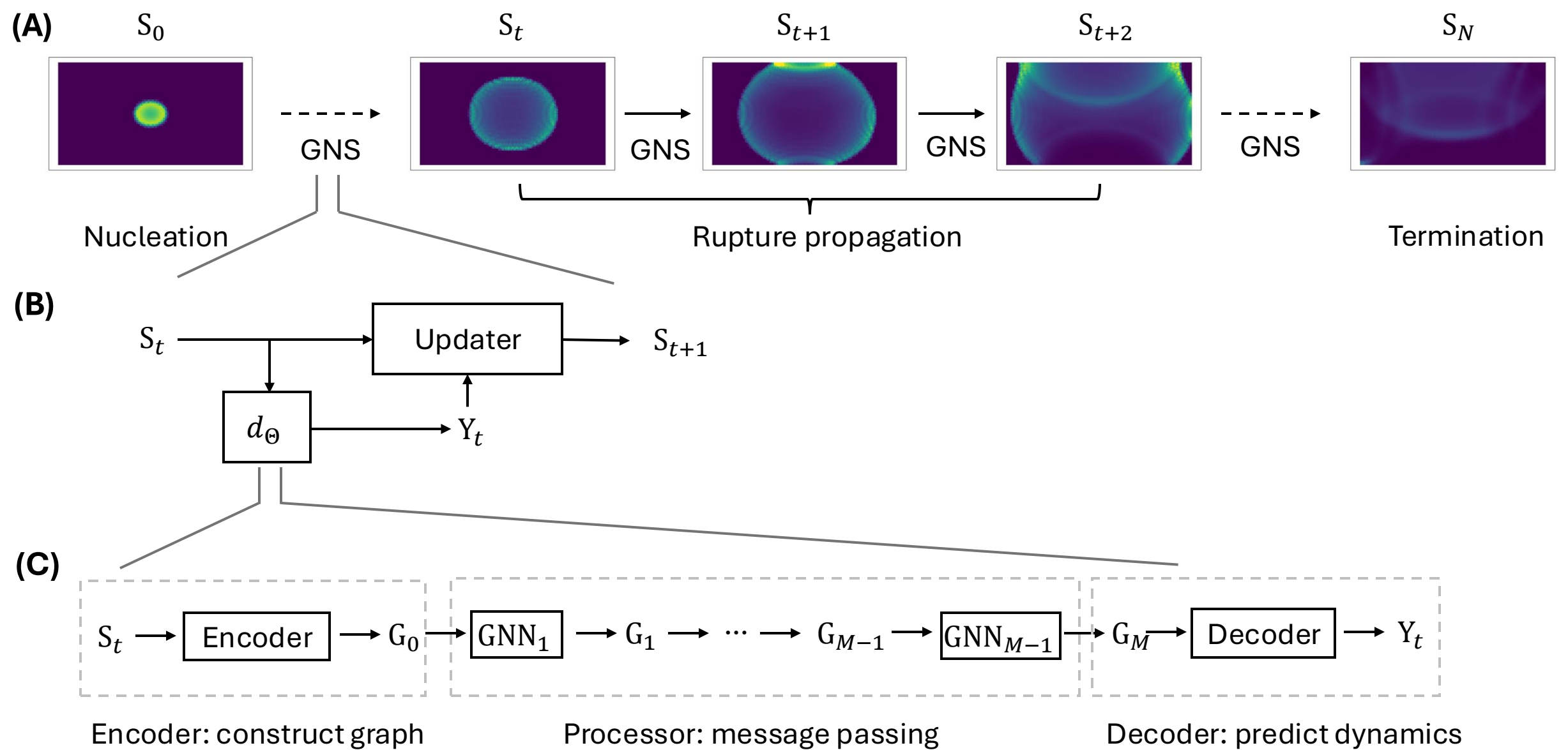


Figure 3.

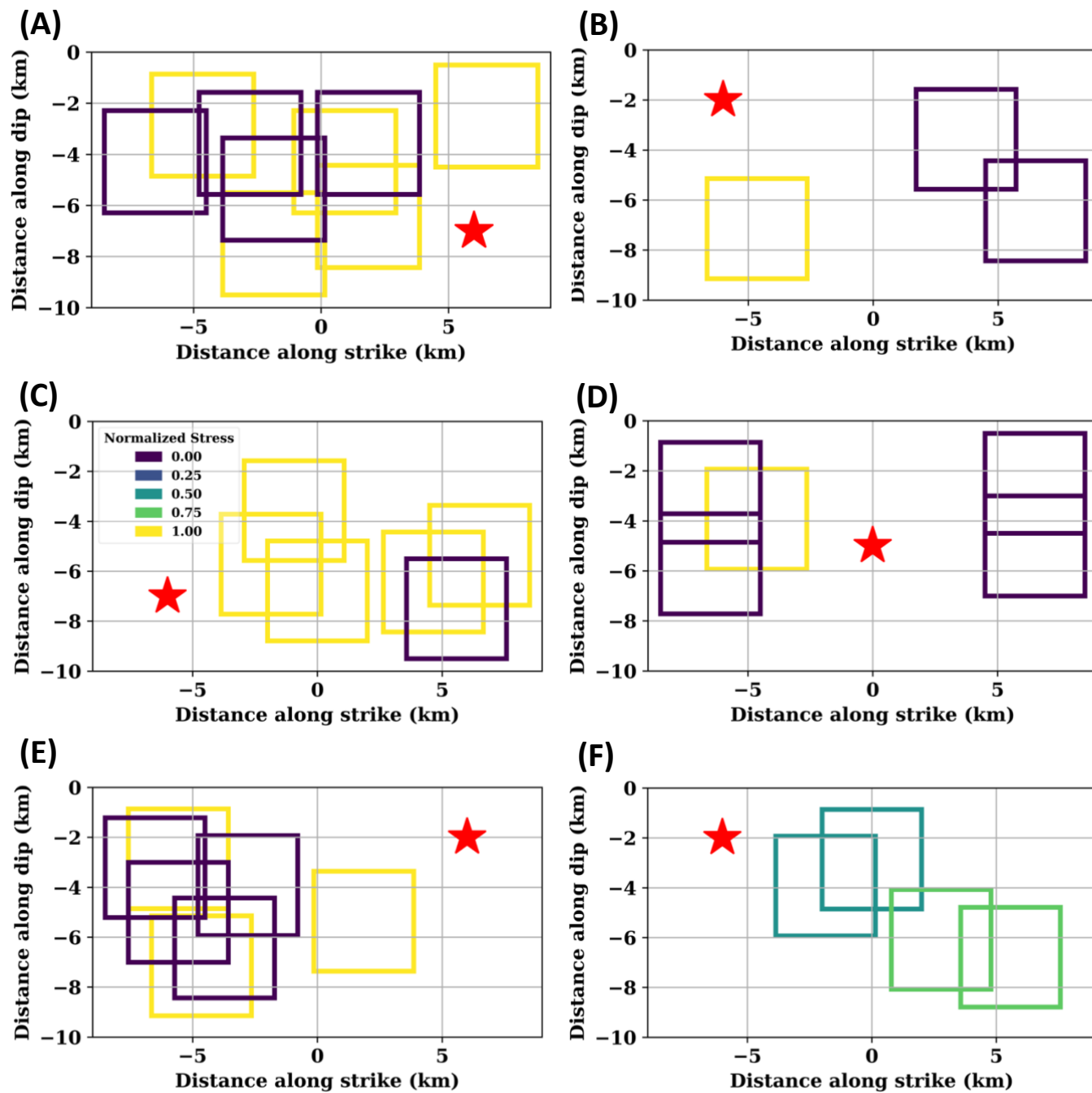


Figure 4.

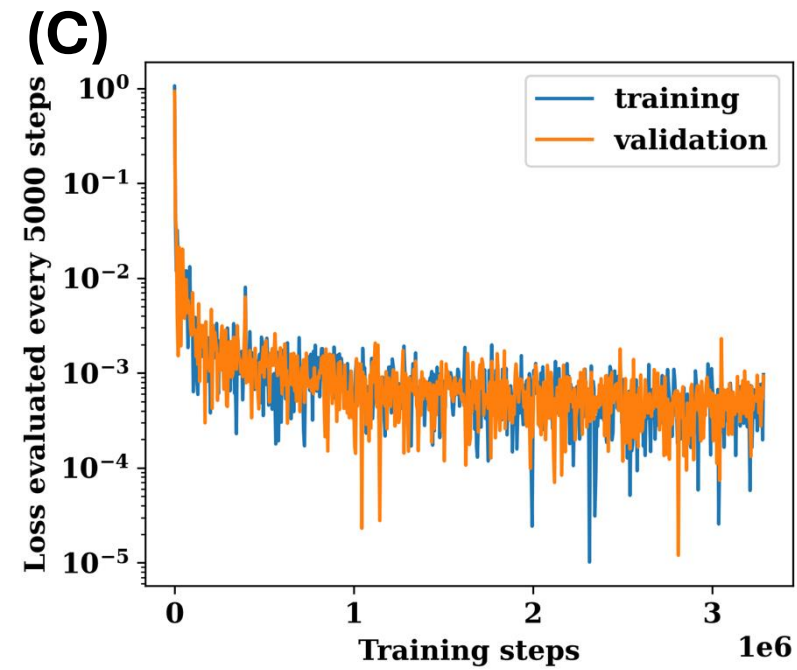
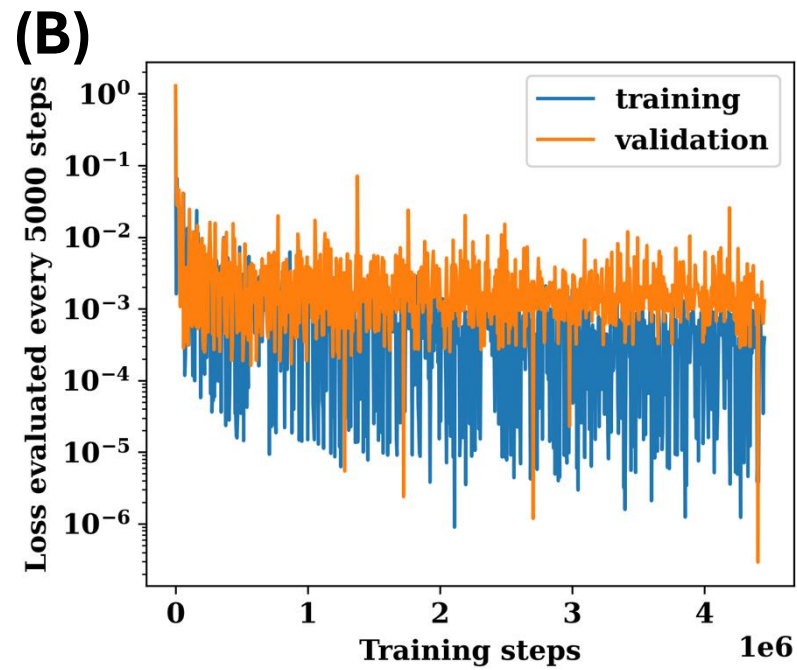
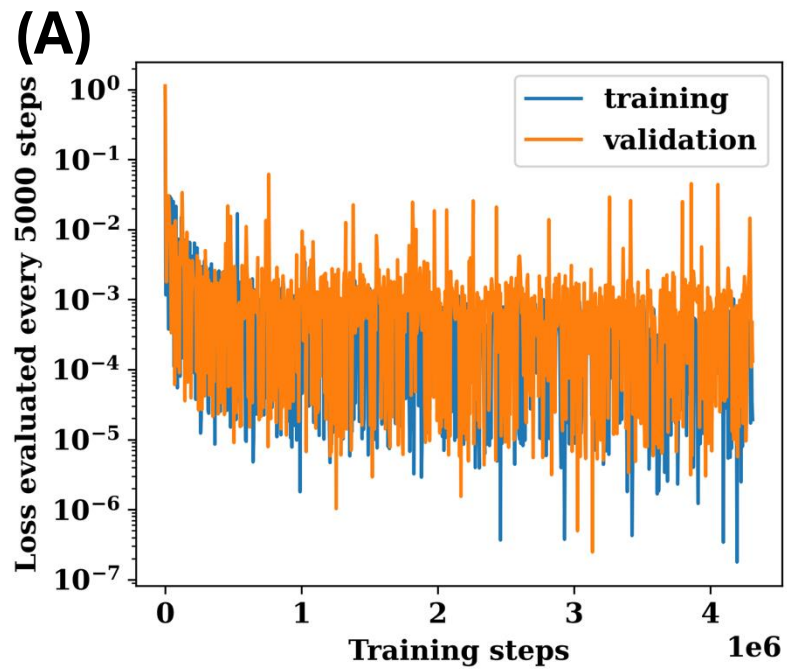


Figure 5.

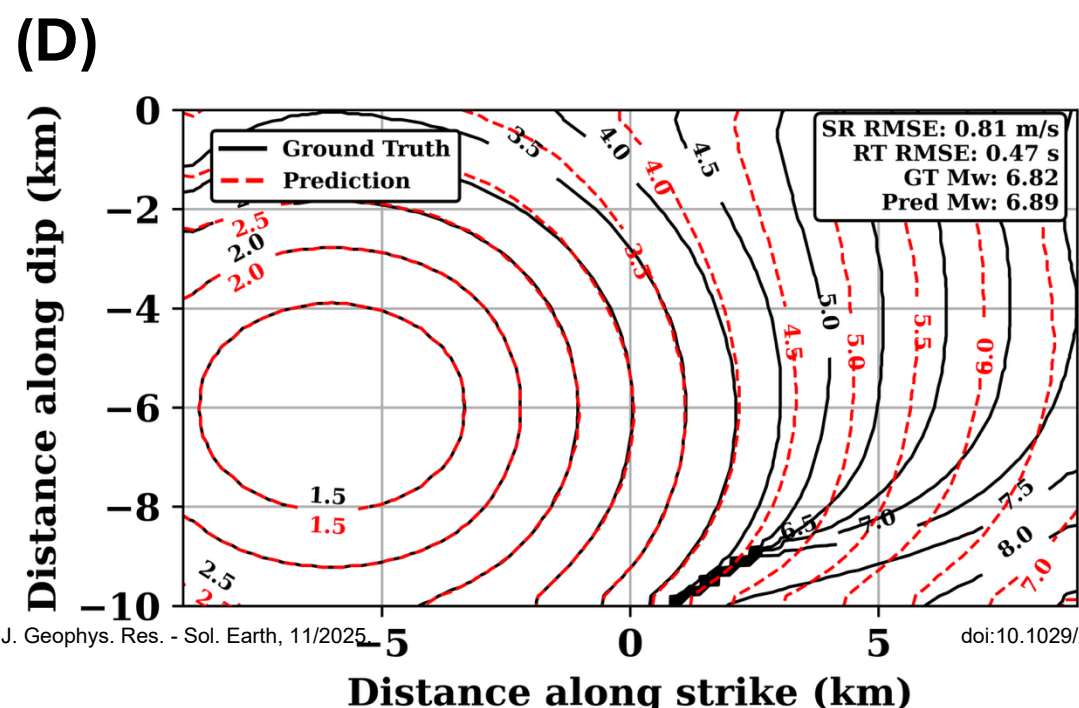
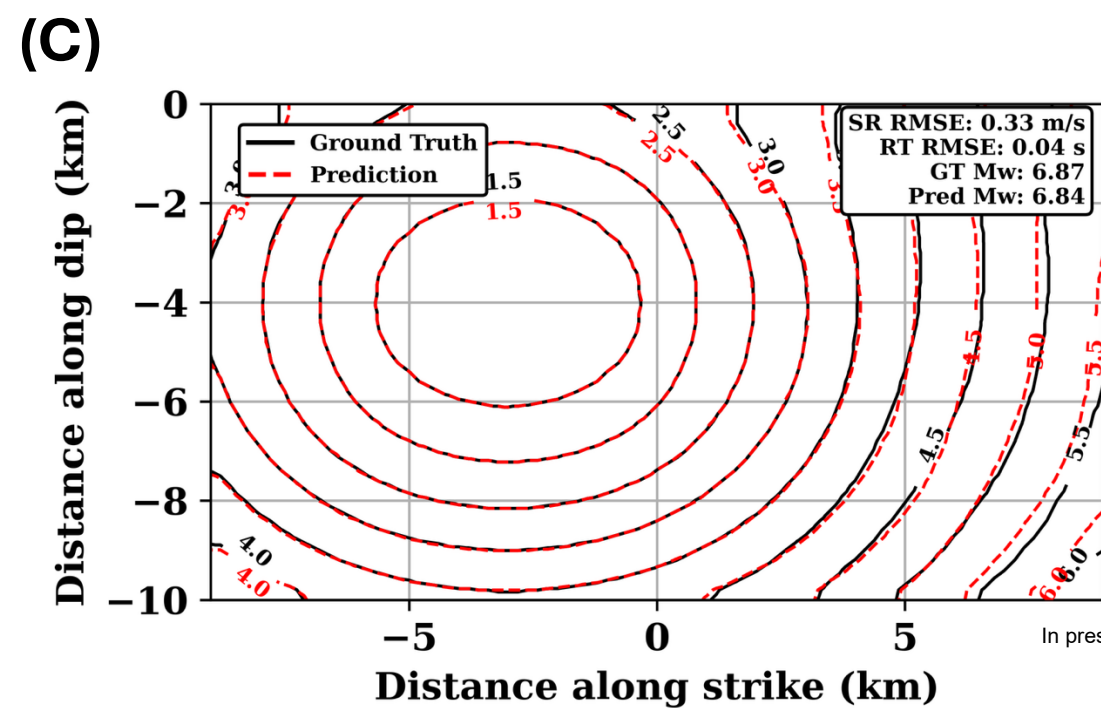
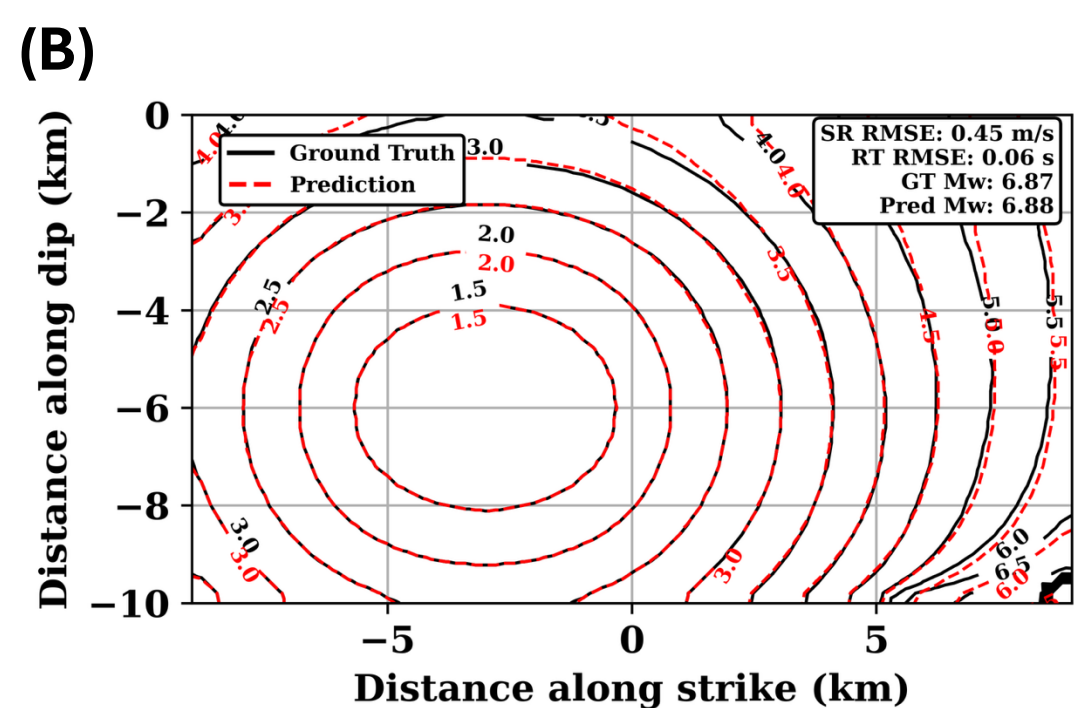
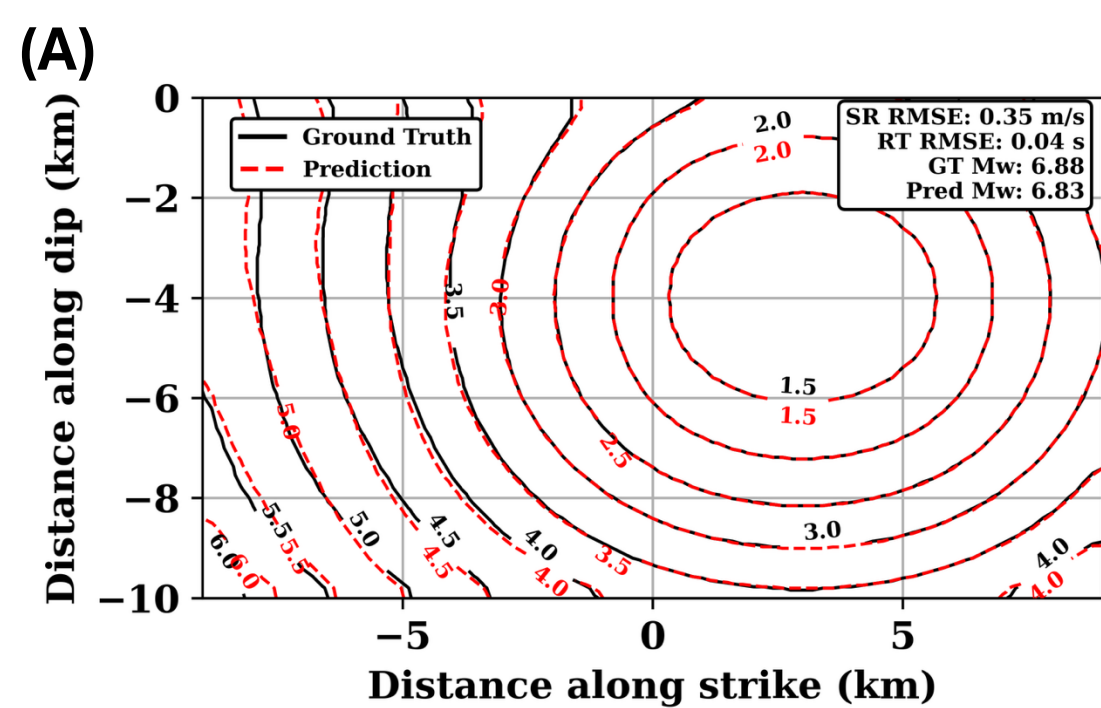


Figure 6.

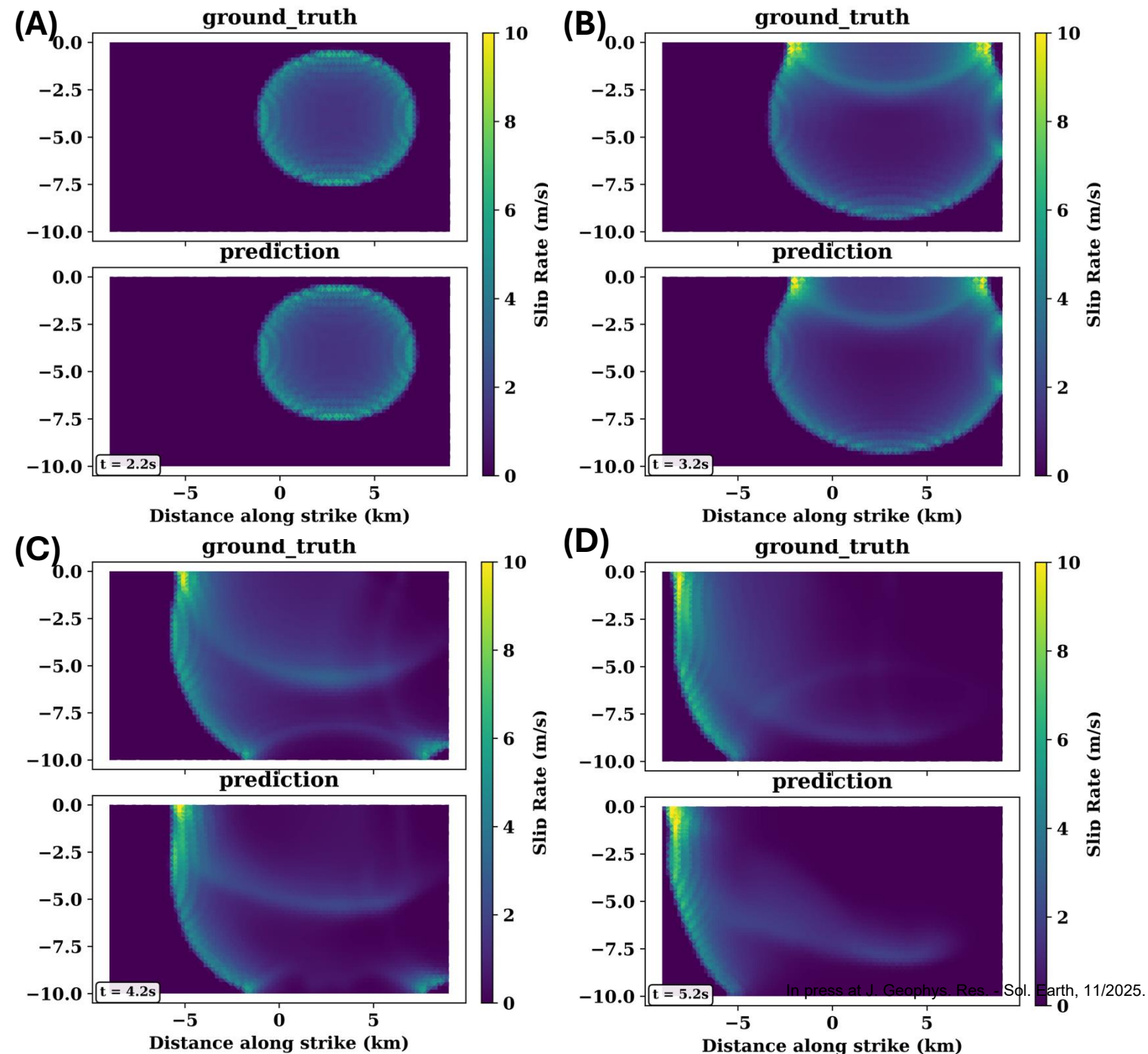


Figure 7.

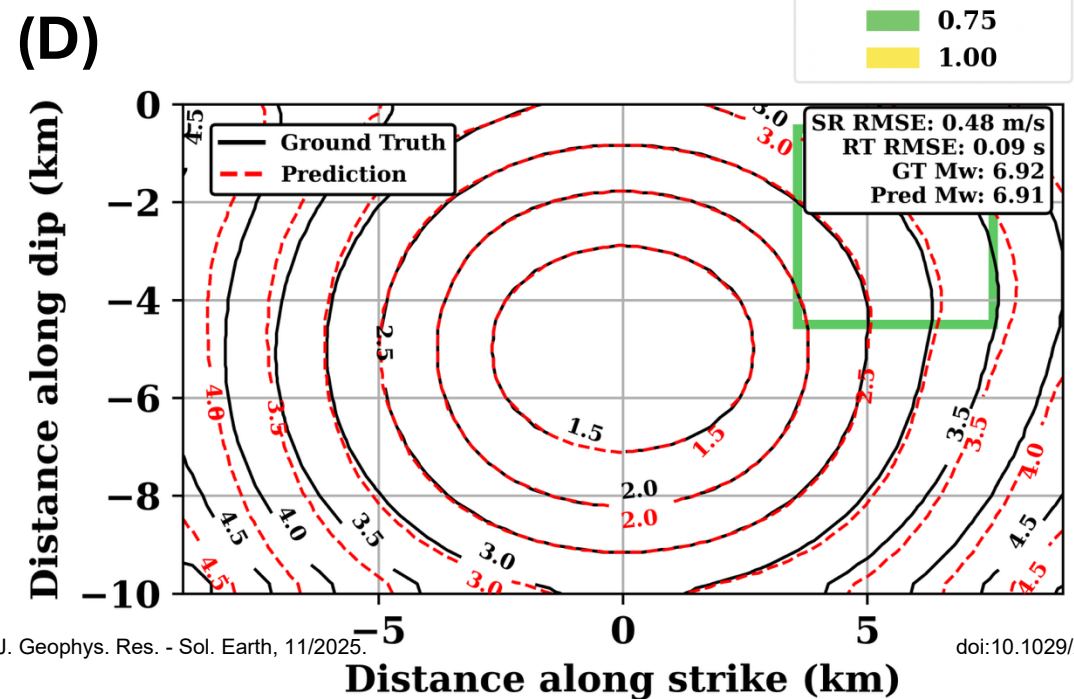
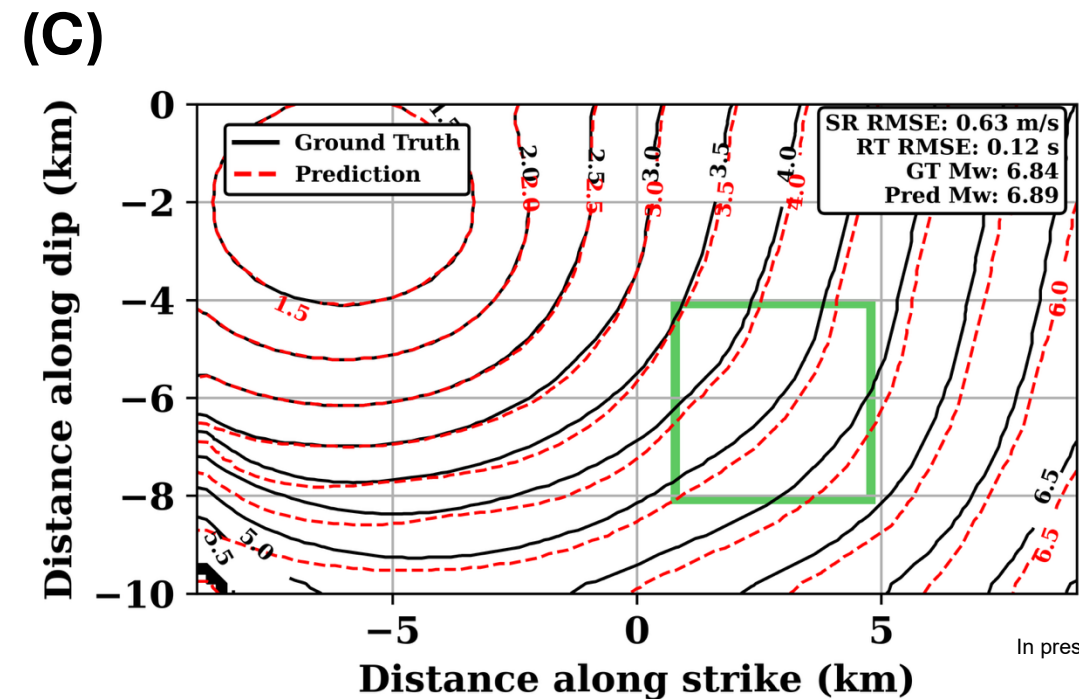
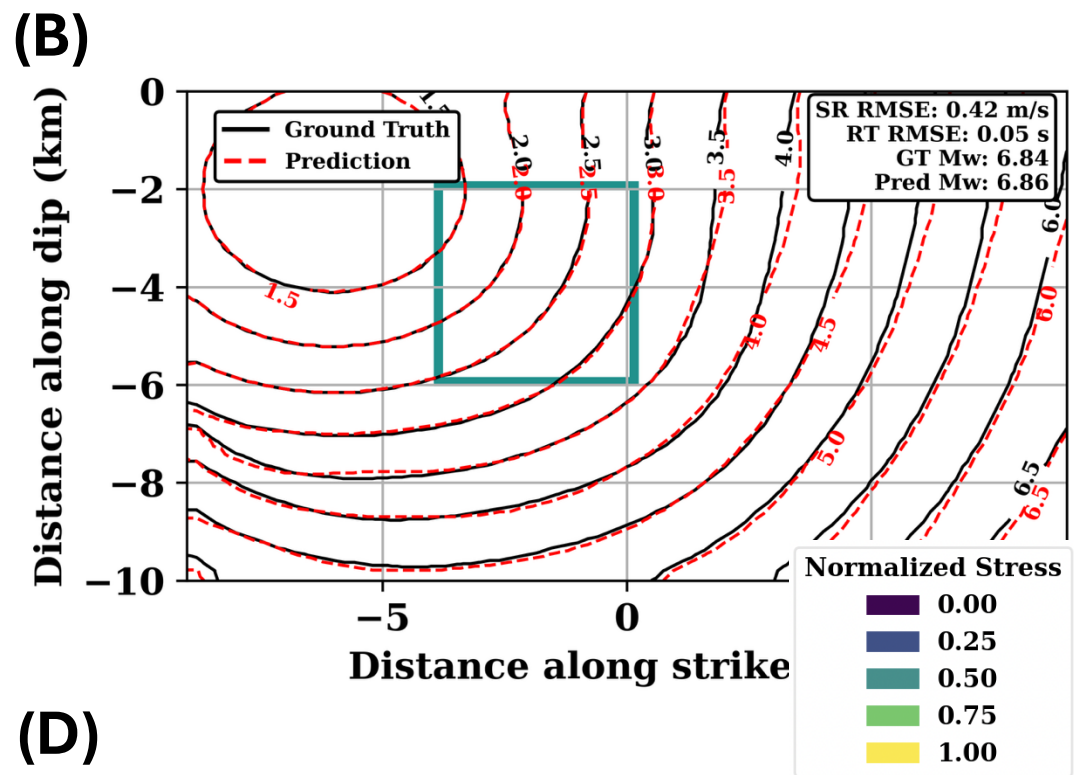
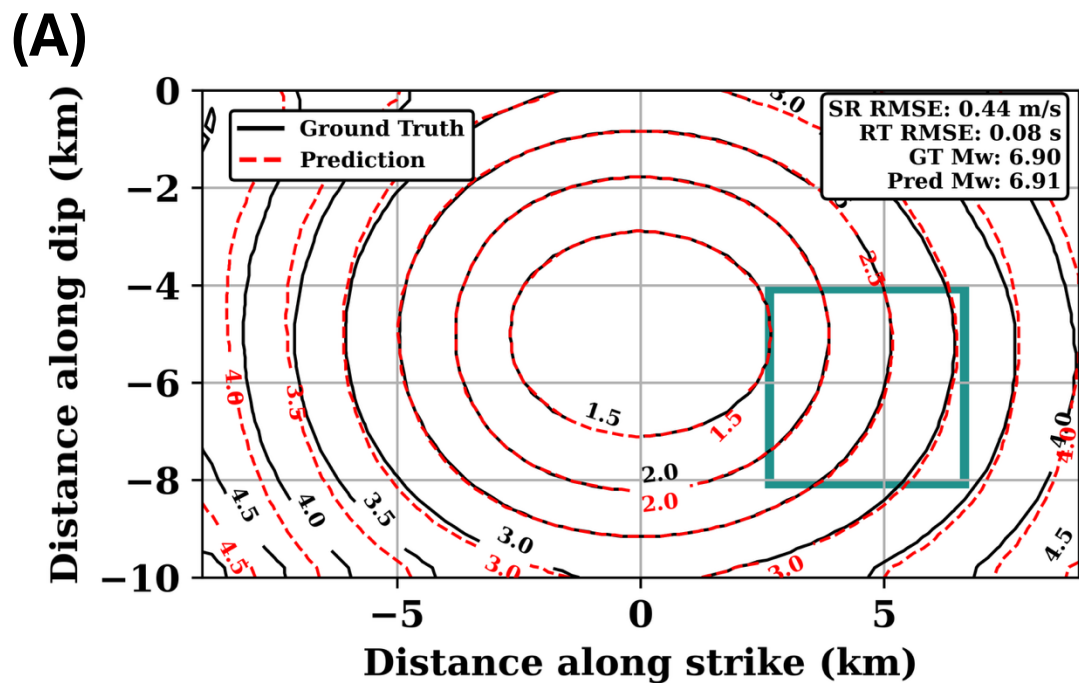
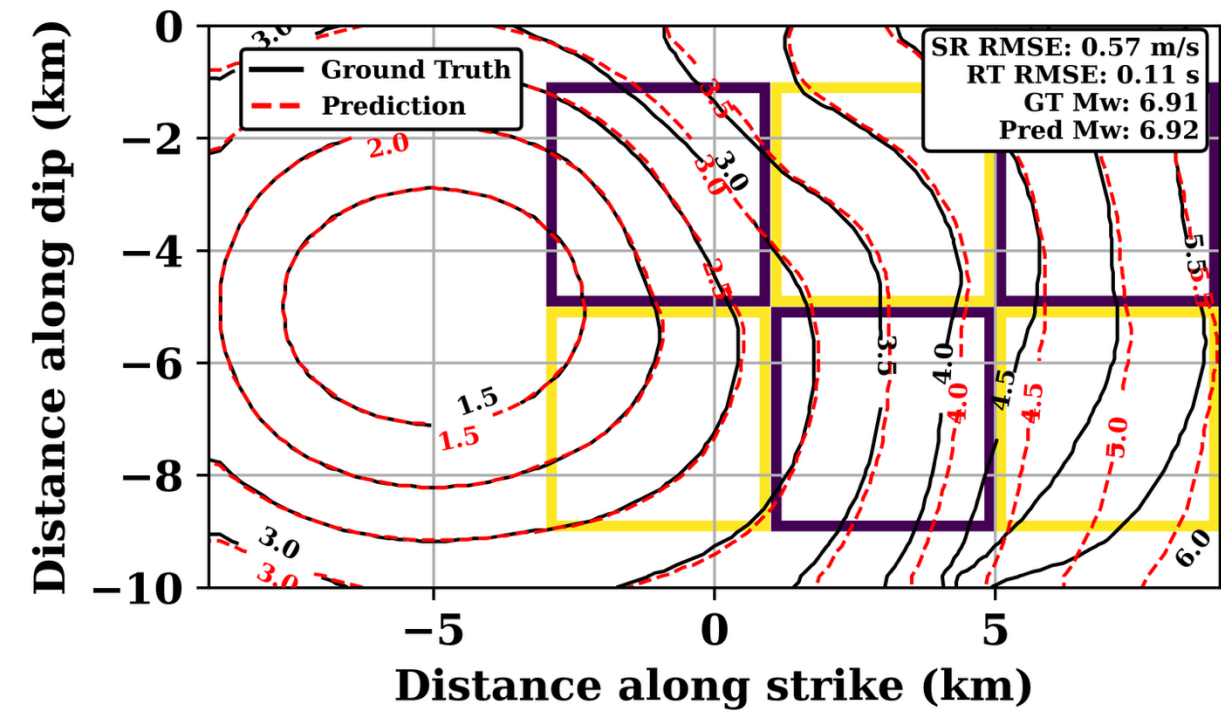


Figure 8.

(A)



(B)

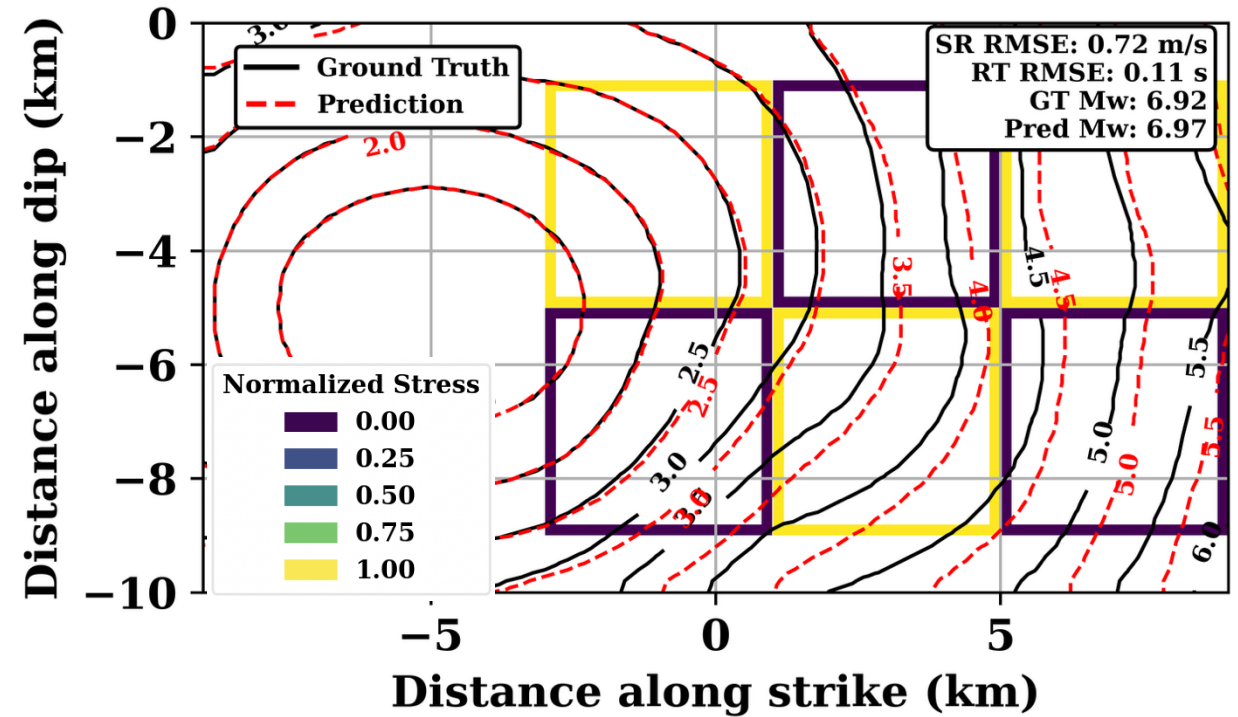
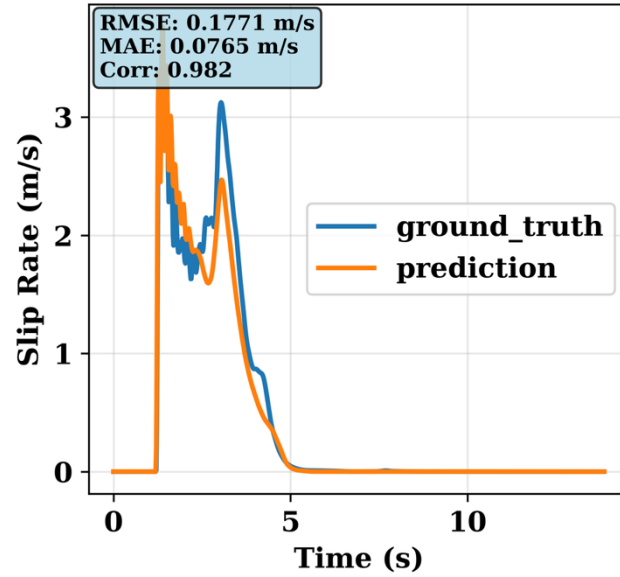
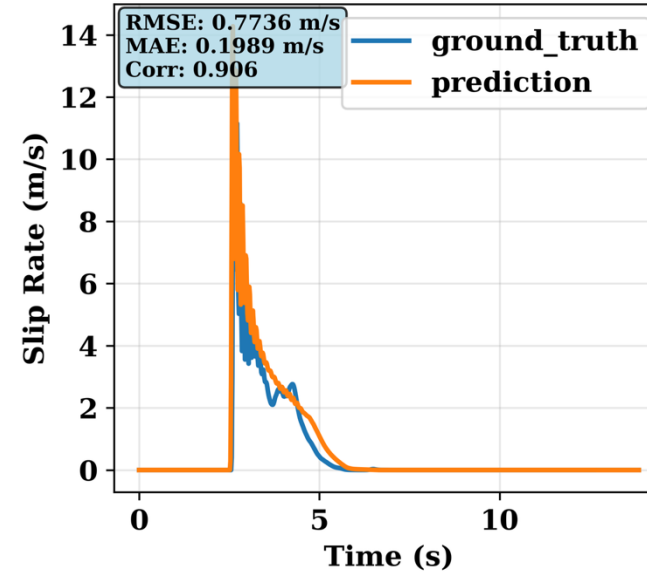


Figure 9.

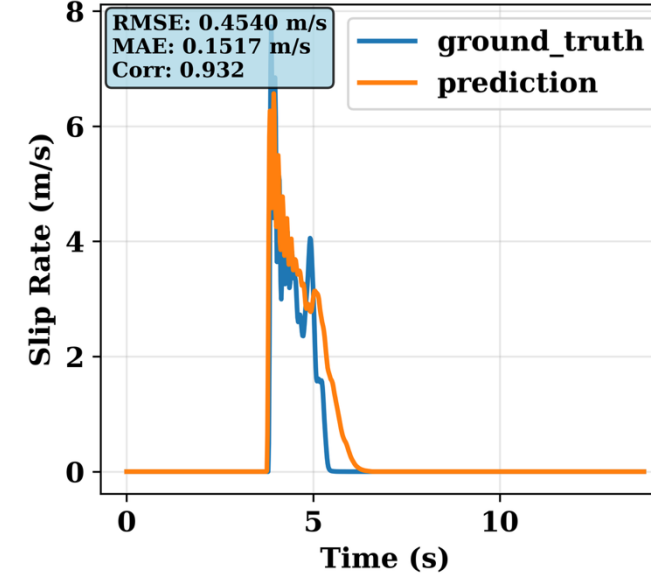
Node Location (-1, -3)



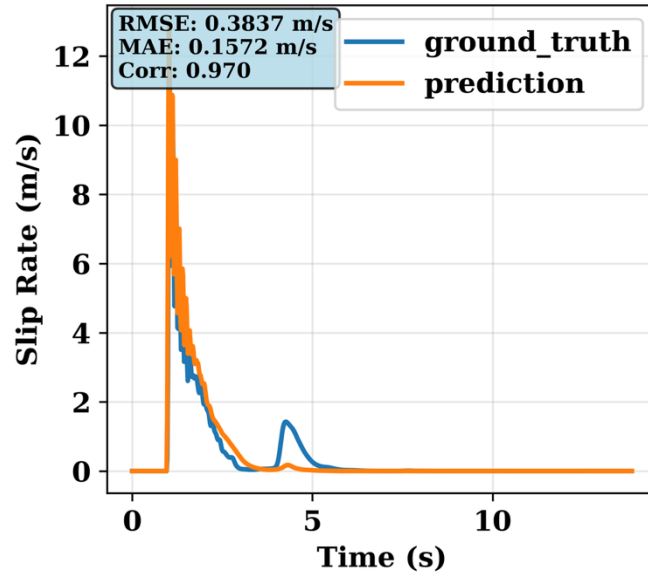
Node Location (3, -3)



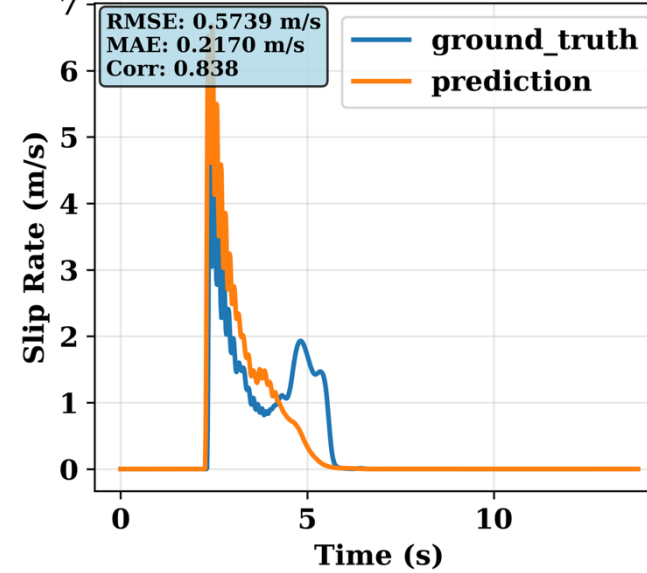
Node Location (7, -3)



Node Location (-1, -7)



Node Location (3, -7)



Node Location (7, -7)

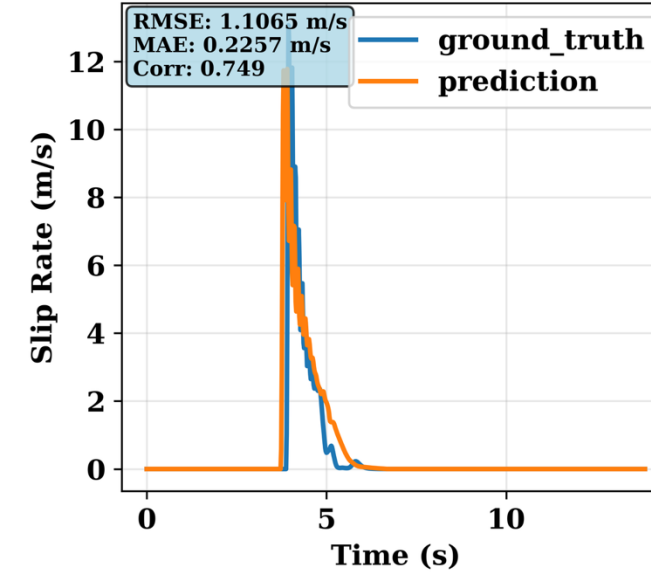


Figure 10.

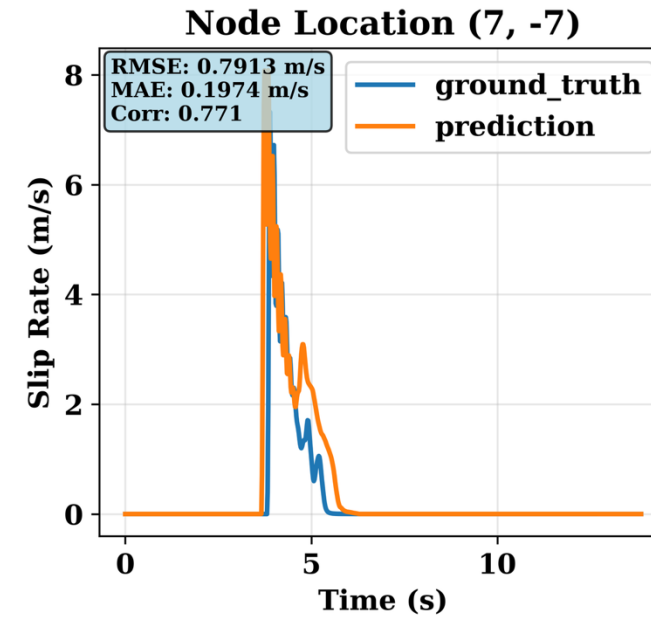
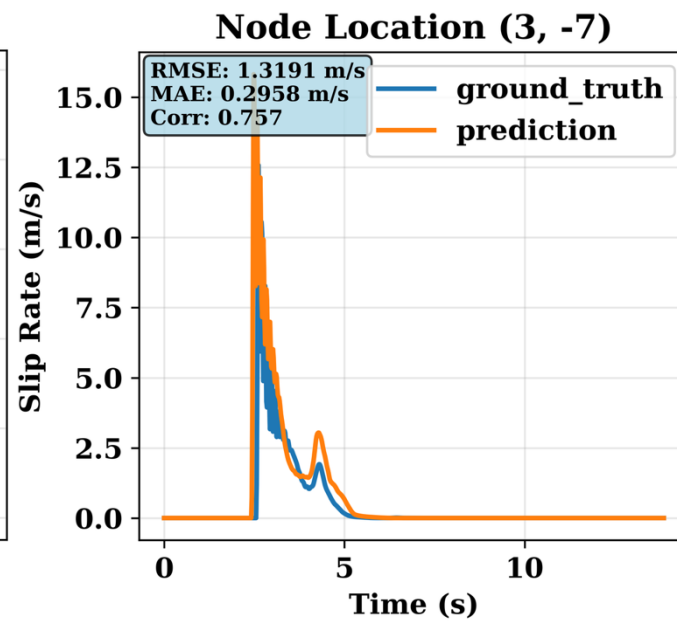
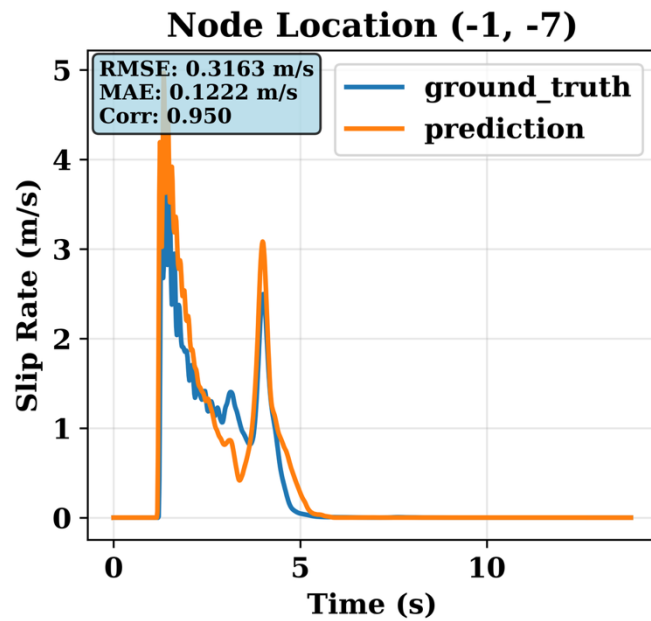
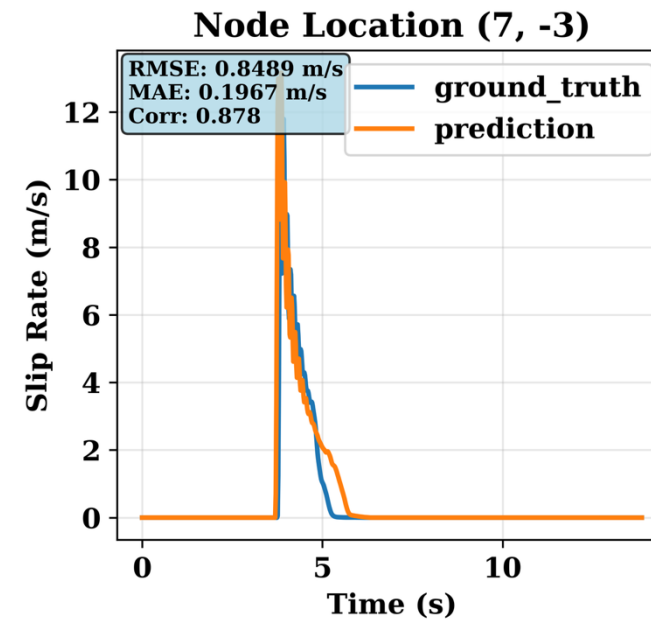
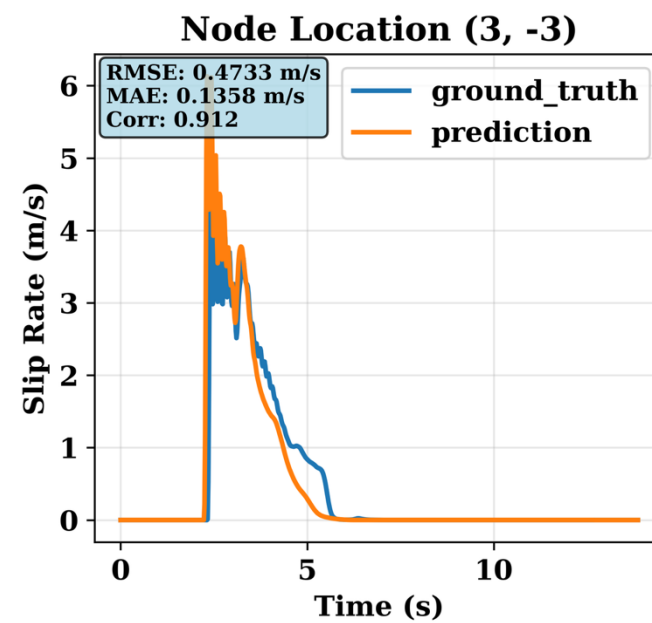
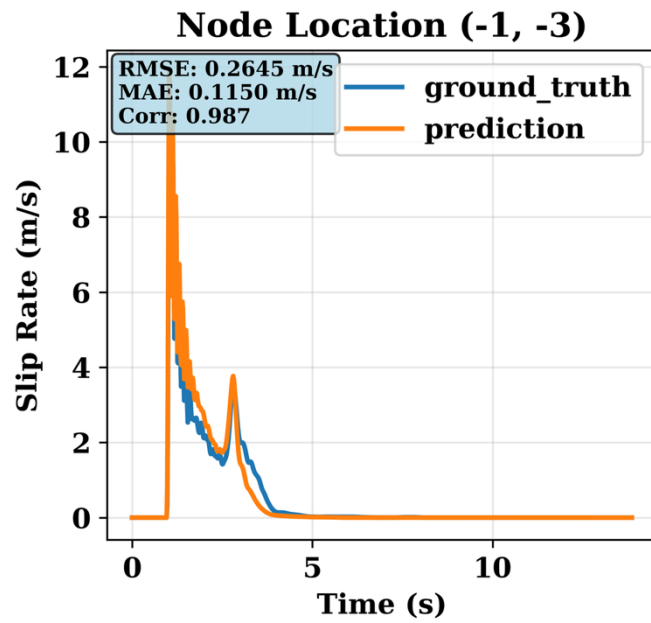


Figure 11.

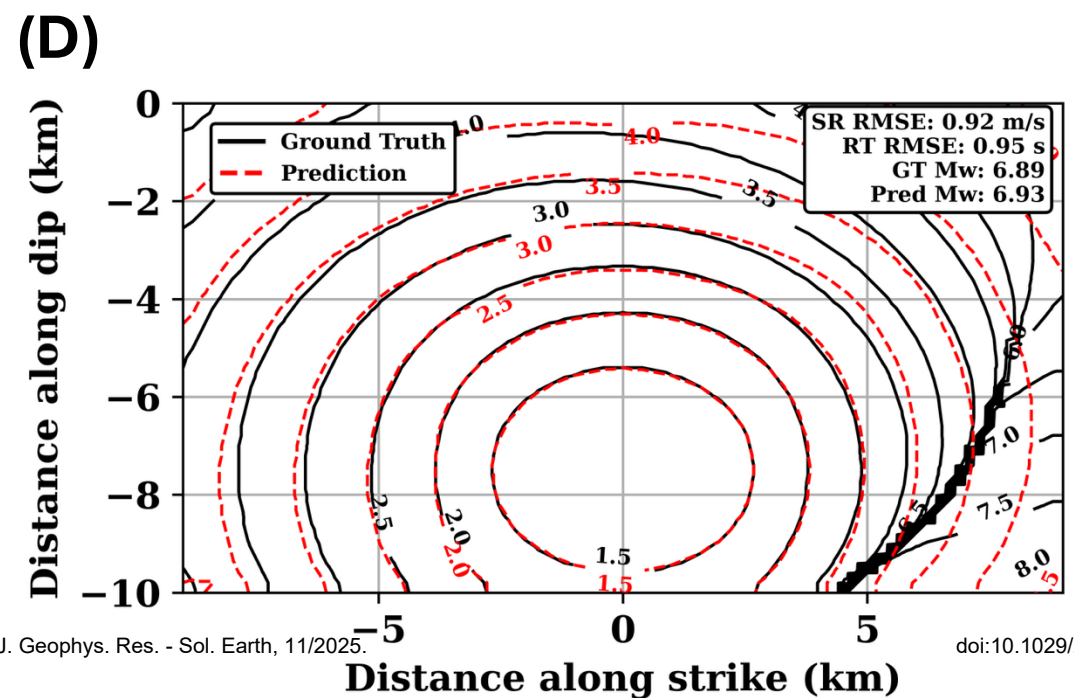
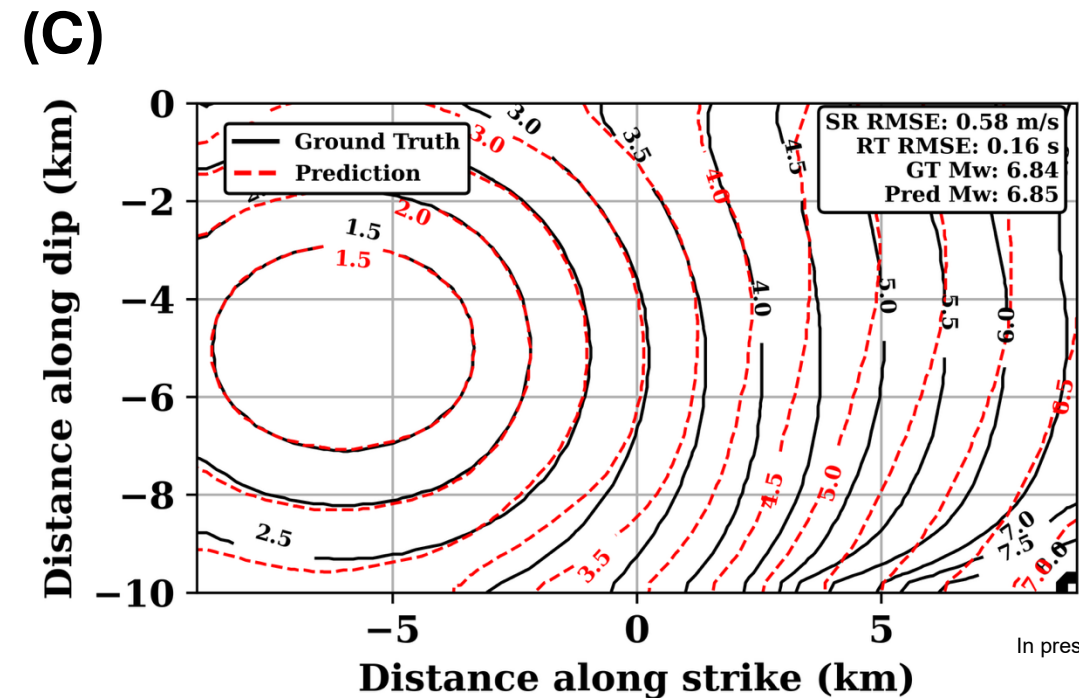
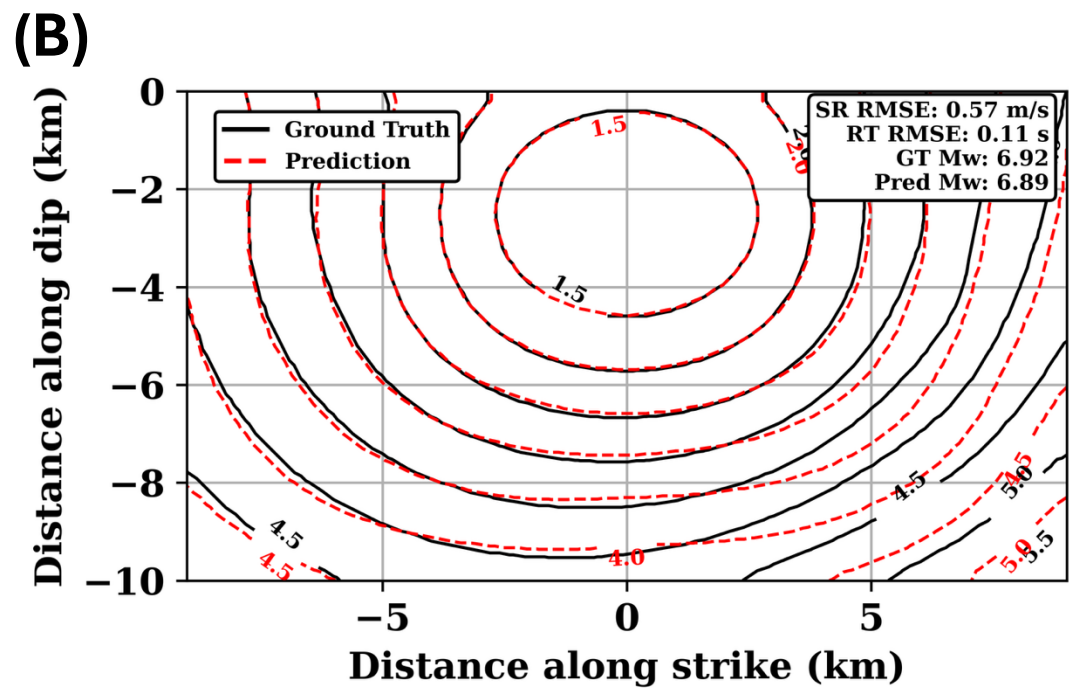
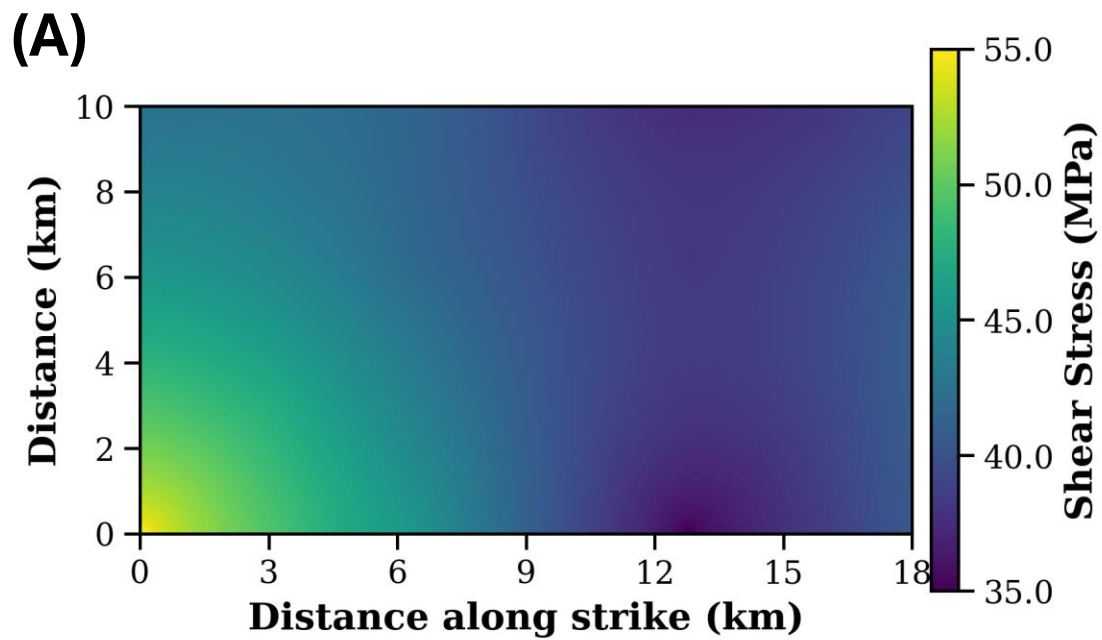


Figure 12.

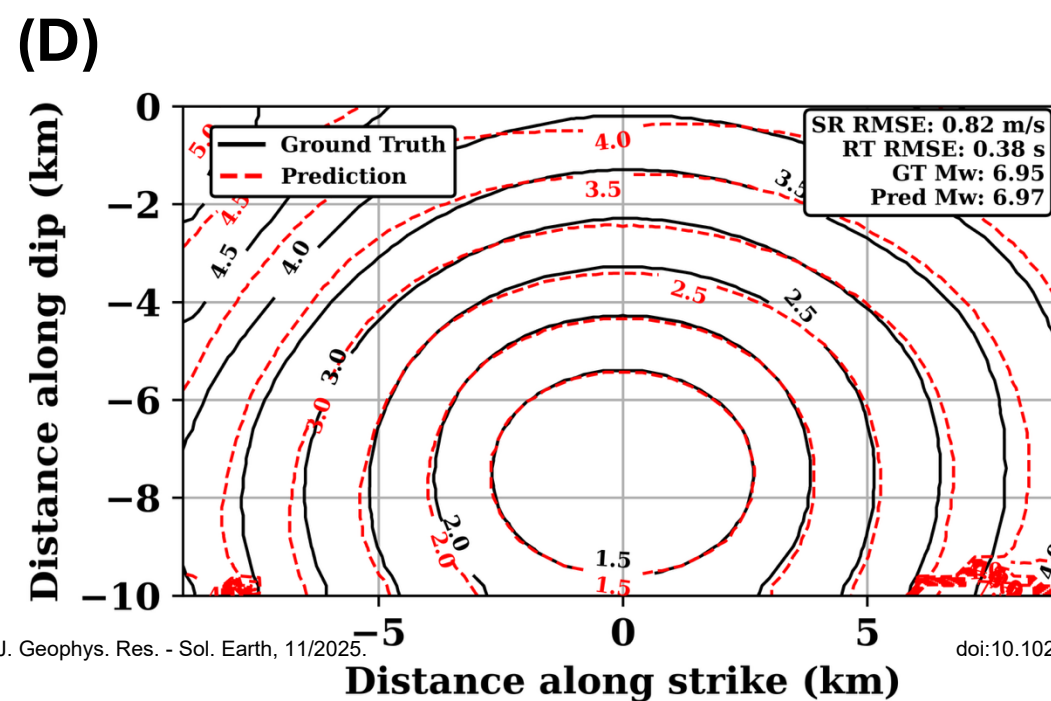
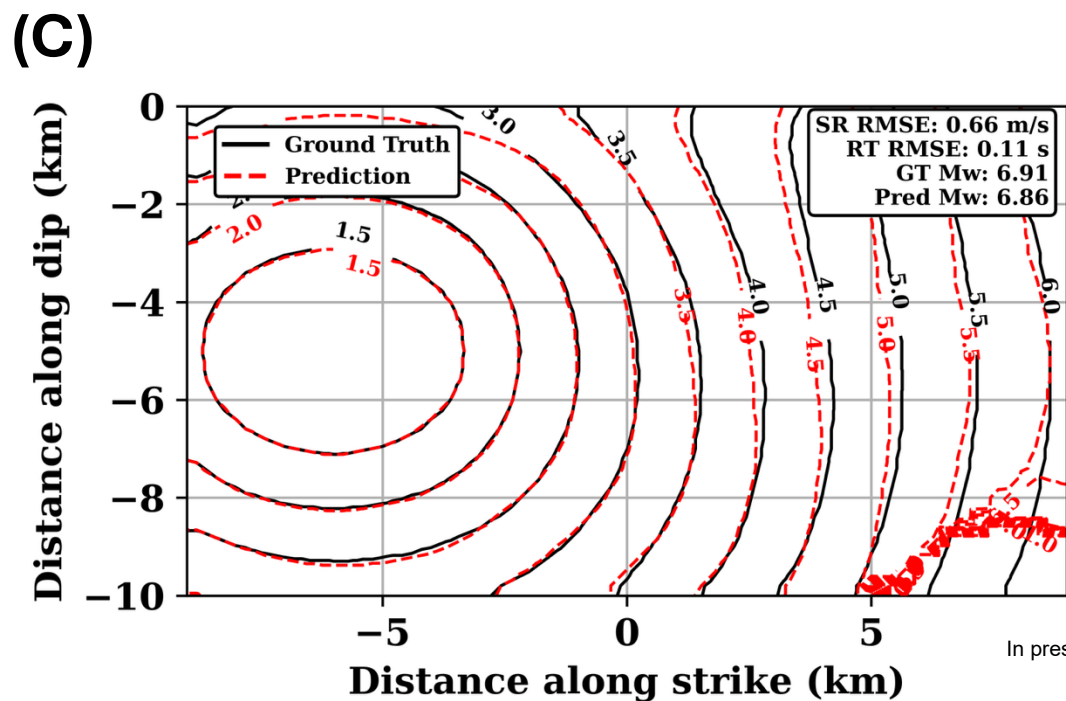
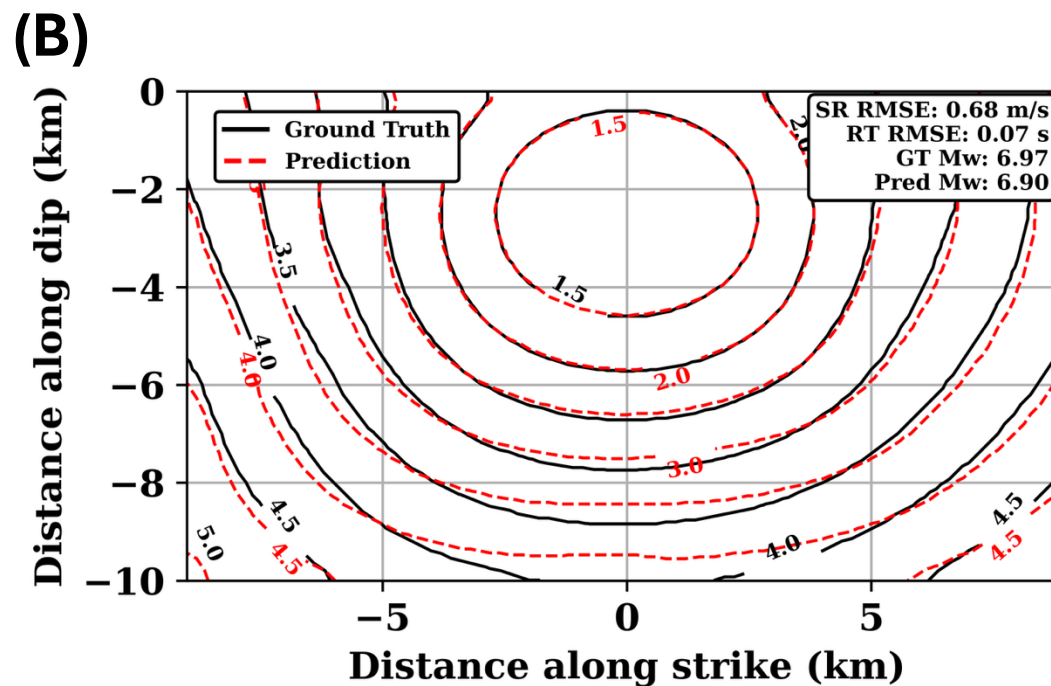
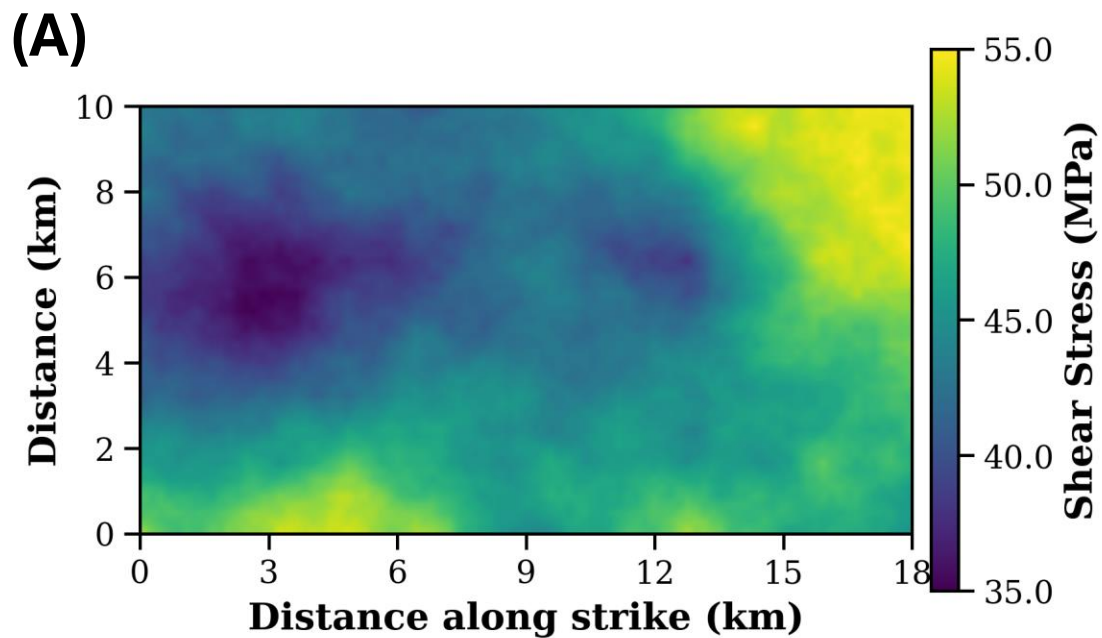


Figure 13.

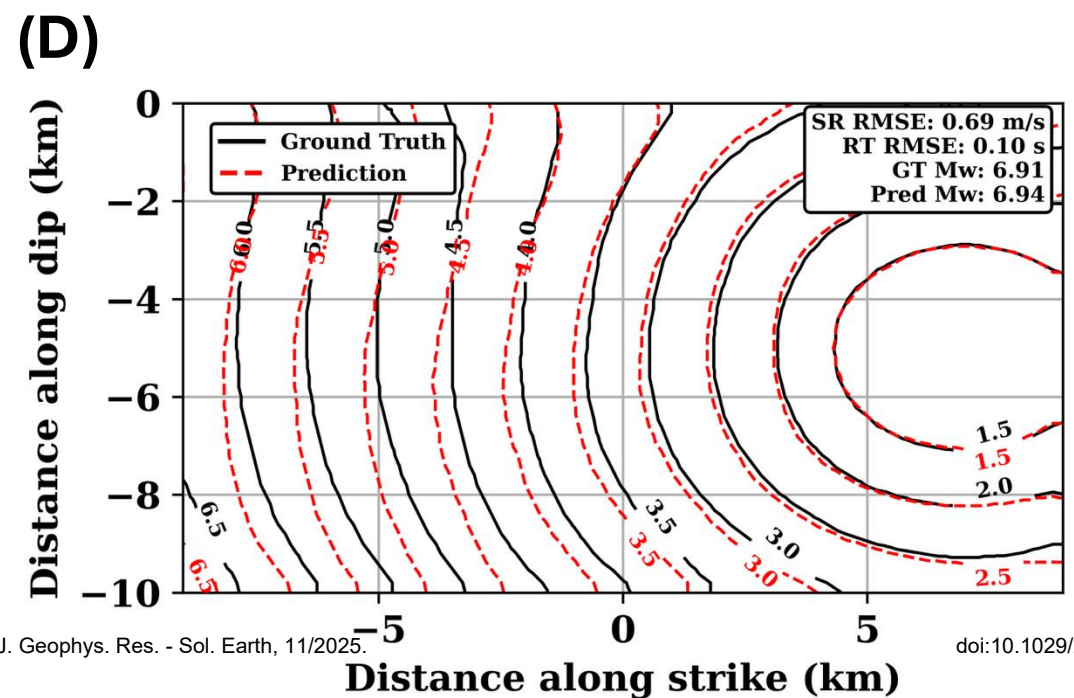
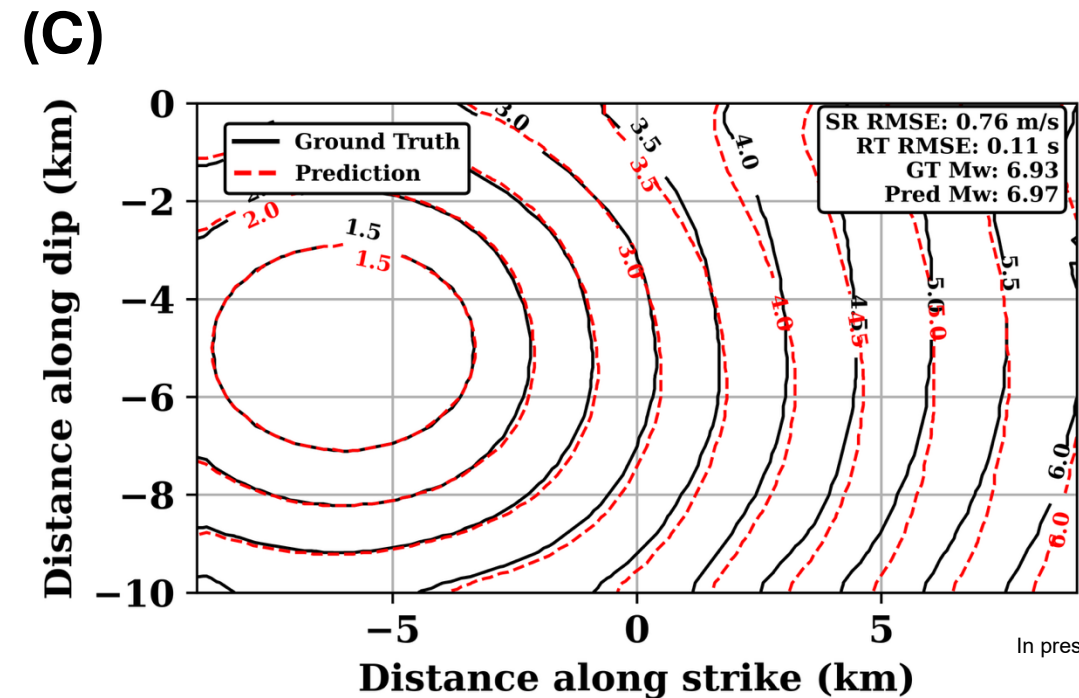
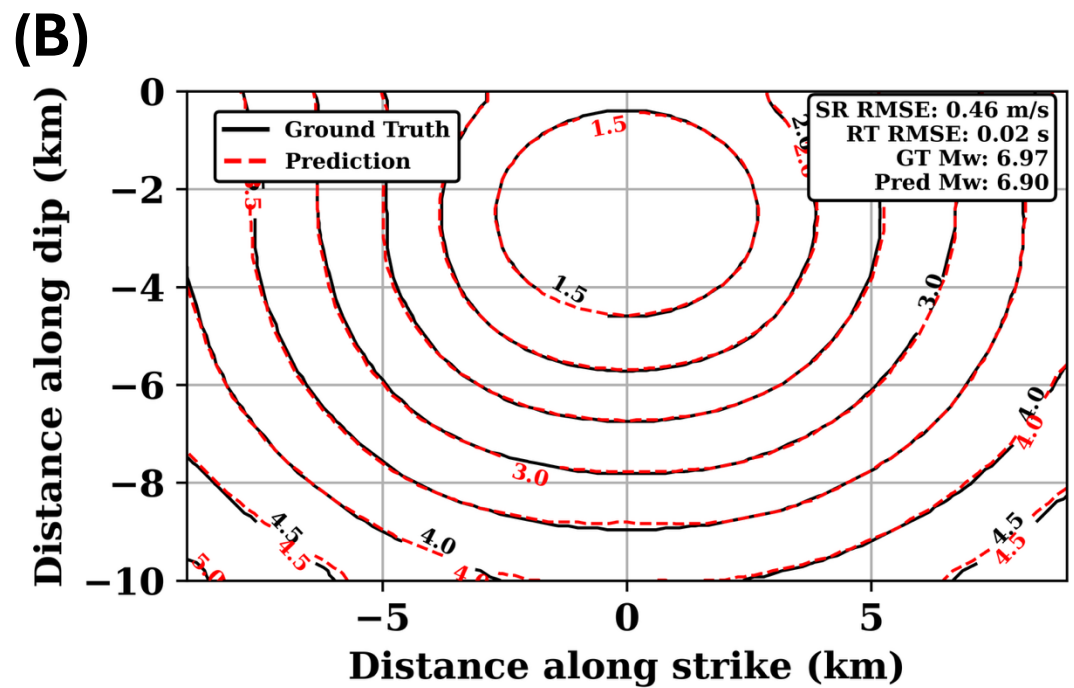
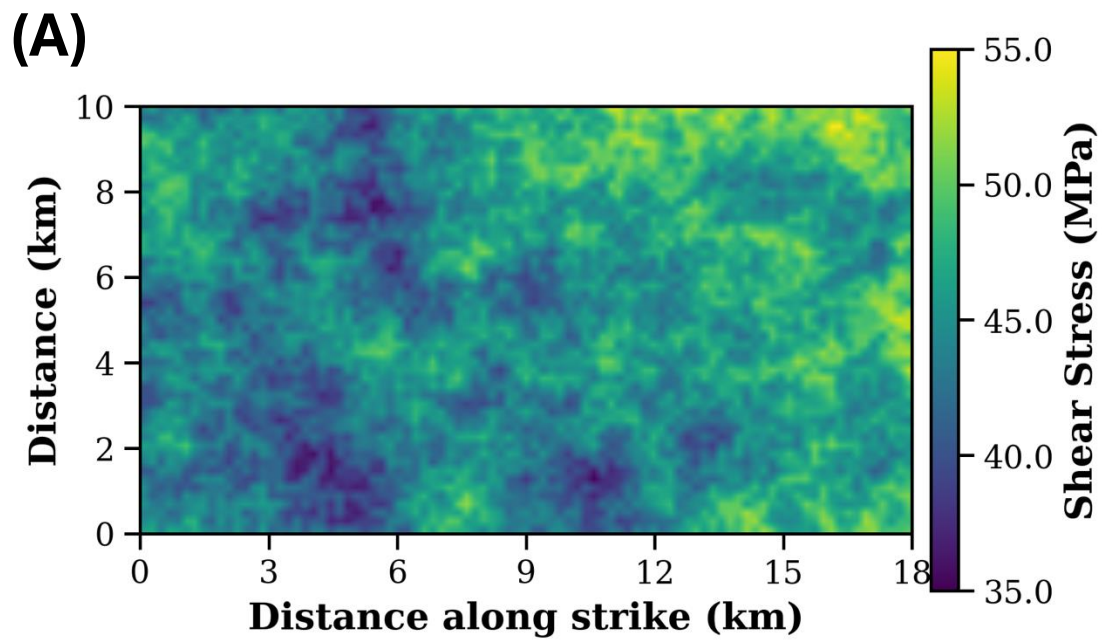


Figure 14.

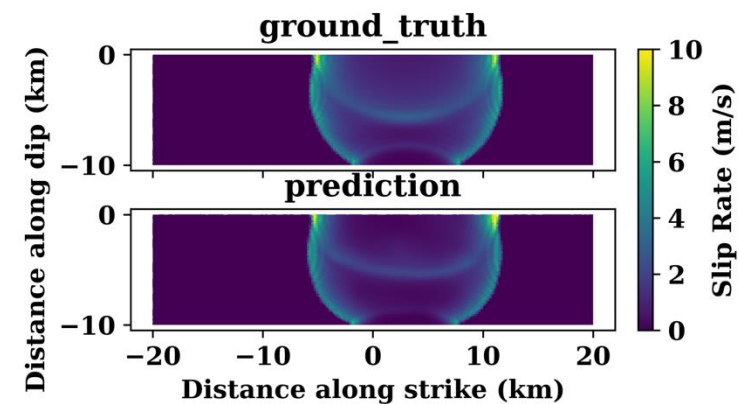
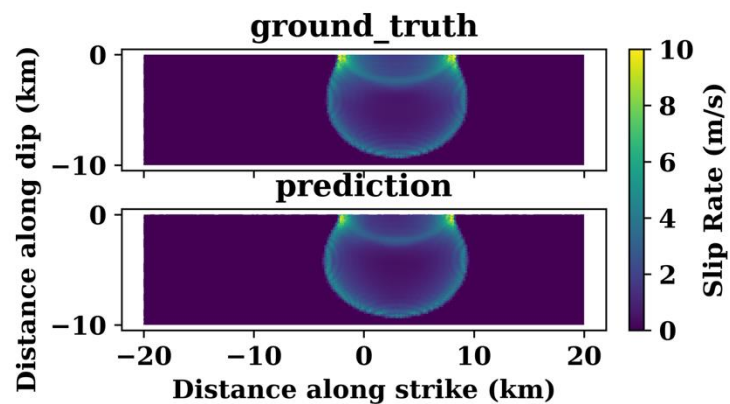
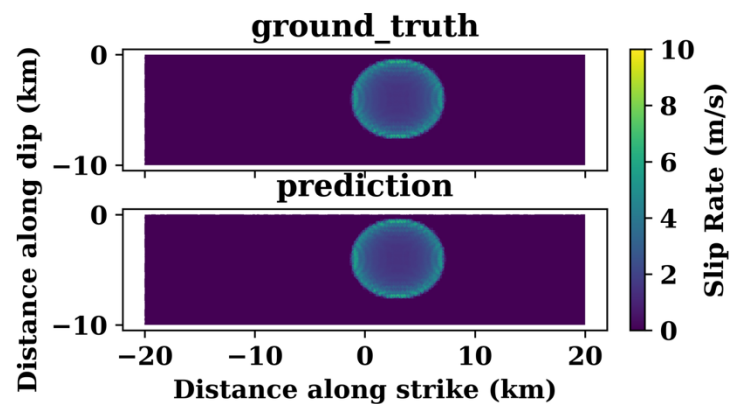
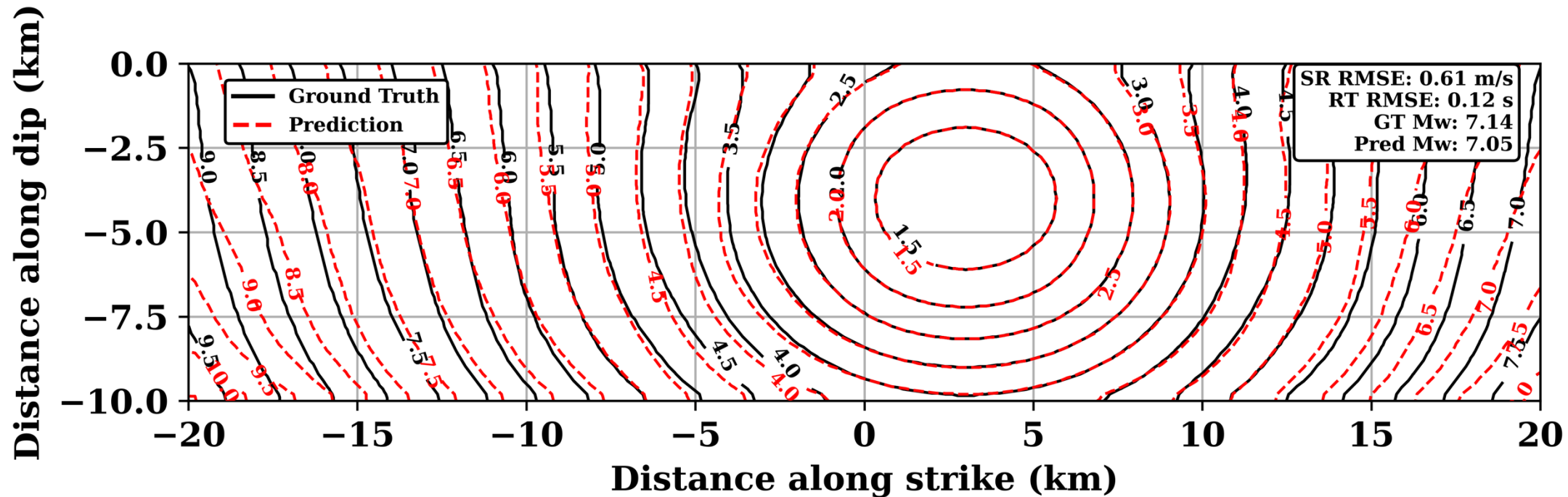


Figure A1.

