Learned Lagrangian Models of PDEs via Euler–Lagrange Residual Minimization

Lyra Zhornyak
University of Pennsylvania
zhornyak@seas.upenn.edu

Eric Forgoston
Montclair State University
eric.forgoston@montclair.edu

M. Ani Hsieh
University of Pennsylvania
mya@seas.upenn.edu

Abstract

We present the first method to directly use a learned continuous Lagrangian to forecast the dynamics of systems governed by partial differential equations, exploiting the inherent conservative structure to achieve stable long-range predictions. We develop an integrator that minimizes the squared Euler–Lagrange residual via a discretization-agnostic, structure-preserving construction on local space-time patches. Different from integrators for analytical models, integrators for learned models should decouple model error (phase error) from integration error (conservation error). By minimizing the residual locally on overlapping patches, we bypass the global implicit solve inherent to fixed discretizations, which slows time- and space-stepping and complicates learning. Our method scales linearly with domain size via Jacobi iteration, and places no structural requirements on the learned network, allowing it to be coupled with existing physics-guided machine learning (ML) methods. We validate our approach on a learned representation of a double pendulum, a one-dimensional wave equation, and a two-dimensional wave equation. Our method achieves long-horizon predictions with error comparable to classical symplectic methods while generalizing to spatially varying dynamics and arbitrary boundary conditions without retraining.

1 Introduction

Dynamical systems are mathematical models that describe how a system evolves over time, typically specified by a state space and a set of rules governing how the states change over time and space. Estimating dynamical systems models from data is challenging because the dynamics are often nonlinear and high-dimensional, making it difficult to identify both the model structure and parameters from noisy, sparse, or partially observed time series measurements of system states. Nevertheless, modern approaches like physics-informed/guided or structure-preserving machine learning have significantly improved our ability to model and capture complex dynamics from data [Thuerey et al., 2021]. In recent years, neural networks have successfully been used to learn Hamiltonians [Greydanus et al., 2019], Lagrangians [Cranmer et al., 2020, Lutter et al., 2019], and dynamics directly from data [Sanchez-Gonzalez et al., 2020, Chen et al., 2018]. Yet while these models capture the underlying physics with increasing accuracy, strikingly little attention is paid to how these models are *used* for tasks where forward integration plays a central role, *e.g.*, forward simulation, prediction and forecasting, estimation and filtering.

This oversight matters because forecasting a system's behavior places different demands on the learned model than finding a solution does. While these two uses are closely related — knowing how to produce a good local solution does help forecasting — their objectives are distinct: *solving* demands local accuracy and sensitivity to the equations, while *forecasting* demands stability and bounded long-term error. For systems that conserve energy (waves, vibrations, orbits, etc.), this distinction is particularly important as even small per-step errors compound over long simulations, eventually producing solutions that violate conservation laws and become physically meaningless.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing work has predominantly focused on producing better and better models of the dynamics, implicitly assuming that forecasting errors can be overcome by either more training or specialized architectures. This assumption fails most severely for chaotic systems and systems defined by partial differential equations (PDEs). Accumulated errors can violate the structure of the system, pushing a trajectory outside of the stable, learned regime, thereby causing a catastrophic failure in the prediction. Classical numerical analysis has long known that preserving structure can matter as much as per-step accuracy. For Hamiltonian and Lagrangian systems, symplectic integrators trade local fidelity for energy conservation and long-term stability [Reich, 1999, Marsden and West, 2001]. Despite this, most learned-physics methods treat the integrator as an implementation detail.

Motivated by this gap, we introduce Euler–Lagrange Minimization (ELM), the first integrator designed on structure-preserving principles for learned continuous Lagrangian models of PDEs. Prior work on learned variational integrators has been limited to finite-dimensional systems [Saemundsson et al., 2020]. ELM extends to multidimensional field theories by evaluating the Euler–Lagrange residual on Hermite-interpolated space-time patches and minimizing the residual locally via Newton iteration. The local construction maintains linear scaling with domain size by avoiding a coupled, global solution process, and the minimization-based framing naturally incorporates boundary conditions through constraints on the optimized variables. When the residual is driven to zero, the integrator is symplectic for Lagrangians with a constant mass matrix, inheriting the long-term energy conservation of classical variational integrators. Beyond sufficient smoothness of the learned Lagrangian, ELM imposes no structural constraints on the model used to learn it, facilitating use with domain-specific physics-guided approaches.

We validate and demonstrate the performance of ELM on three systems of increasing dynamical complexity: a chaotic double pendulum, where ELM achieves energy drift comparable to state-of-the-art symplectic methods; the one-dimensional (1D) wave equation, where a learned Lagrangian density outperforms Lagrangian Neural Networks [Cranmer et al., 2020] in both accuracy and compute time; and the two-dimensional (2D) wave equation, where (to the best of our knowledge) ELM provides the first forward simulation of a 2D PDE using a learned continuous Lagrangian density. By not preemptively discretizing space, ELM allows models trained on homogeneous domains to produce, without modification, physically correct reflection, transmission, and interference effects for spatially varying dynamics and new boundary conditions.

We summarize related work in Section 2 and describe our method and conditions for symplecticity in Section 3. Section 4 demonstrates the low long-horizon energy drift of ELM on all three systems and we conclude with a discussion on limitations and future work in Section 5. Code is publicly released.

2 Related Work

Operator Methods for Differential Equations The most conceptually straightforward method for learning a dynamical system might be directly modeling the underlying equations. Physics-Informed Neural Networks [Raissi et al., 2019], for example, add the residual of the governing equation applied to the network outputs as a loss term. However, this approach often requires strong prior knowledge on the function being learned to ensure training is successful [Krishnapriyan et al., 2021]. To bypass these limitations and to enable more general models, one can alternatively learn a map to an operator space where solutions are more easily found. Some examples include DeepONet [Lu et al., 2021], which uses a branch-trunk architecture to map the input and output to a common space; Fourier Neural Operators [Li et al., 2021], which learn operators using truncated Fourier modes to achieve resolution invariance; and other physics-informed operator learning variants such as PI-DeepONet [Wang et al., 2021] and PINO [Li et al., 2024]. Operator-based methods rely on a strict definition of the domain and are highly structured. To learn local physical laws and allow for arbitrary domains, graph neural network approaches such as Graph Network Simulators [Sanchez-Gonzalez et al., 2020] and MeshGraphNets [Pfaff et al., 2021] have successfully been shown to represent classical numerical schemes such as finite-differences [Brandstetter et al., 2022]. While these methods are effective on a wide variety of problems, none of them encode or preserve the conservation laws of the systems they model. As such, even small per-step errors in integration will accumulate and eventually violate fundamental conservation laws. Post-hoc remedies such as filtering can suppress some high-frequency error [Lippe et al., 2023], but do not address the structural cause of the drift. This limitation is not just architectural; these methods implicitly fuse the dynamics with their integrator, motivating efforts to instead learn the variational structure from which the dynamics arise.

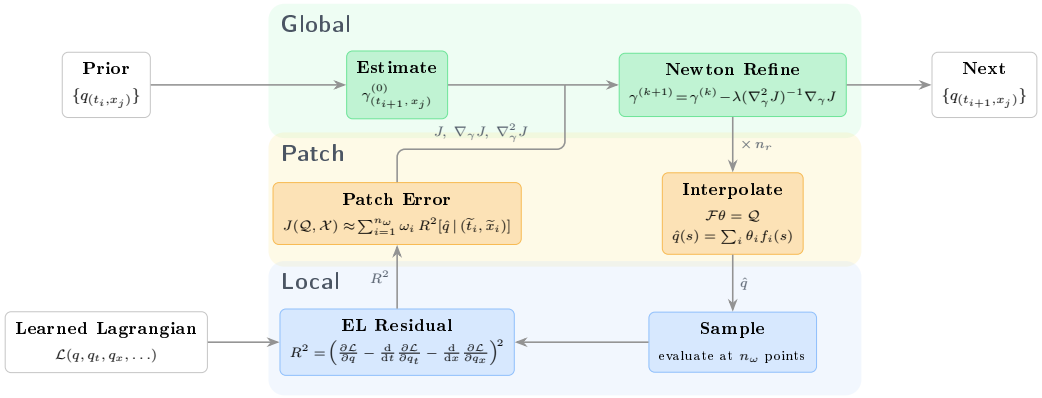


Figure 1: The construction of ELM at three scales. *Local*: the squared Euler–Lagrange residual R^2 is computed from the learned Lagrangian $\mathcal{L}$ and evaluated at n_ω quadrature points. *Patch*: Hermite interpolation fits a set of basis functions to known node states to form $\hat{q}$. The resulting residuals are summed to form the patch error J. *Global*: an initial estimate $\gamma^{(0)}$ is refined by damped Newton iteration using J; iterating n_r times around the loop advances the solution by one time step.

Structure-Preserving Learning Variational mechanics, by contrast, considers the whole trajectory rather than any individual point. Conserved quantities thus manifest as structural symmetries of the governing functions, *e.g.* a Lagrangian or a Hamiltonian. These functions can either be learned directly, as in Hamiltonian [Greydanus et al., 2019] and Lagrangian Neural Networks (LNN) [Cranmer et al., 2020], or used as the basis for designing the structure of the network, as in DeLaN [Lutter et al., 2019] and constrained formulations [Finzi et al., 2020] (see Yu and Wang [2024] for a broader survey). In most cases, the method to produce long-horizon forecasts, *i.e.* the integrator, is barely considered [Chen et al., 2020]. Existing works that explicitly preserve structure apply only to systems governed by ODEs [Saemundsson et al., 2020] or rely on fixed spatial discretizations. Offen and Ober-Blöbaum [2024] learn a discrete Lagrangian for a prescribed discretization which leads to an implicit update scheme that may couple all nodes into a single global solve. This is computationally expensive and can be numerically unstable, limiting this method to only 1D PDEs. By assuming that the grid is regular, FS-HNN [Li et al., 2026] extends structure-preserving approaches to two dimensions by combining a learned Hamiltonian functional (via DeepONet) and a learned convolutional kernel. FS-HNN is not symplectic, however, and achieves energy conservation by projecting the operator onto a skew-symmetric subspace. With ELM, we instead use learned continuous Lagrangian densities independent of the boundary conditions and discretization. The structure-preserving construction improves energy conservation by orders of magnitude over long time horizons.

Classical Variational Integrators Variational integrators, derived directly from the discrete Euler–Lagrange equations, are symplectic by construction [Marsden and West, 2001]. For PDEs that arise from field theories, this construction extends to multisymplectic integrators that preserve the underlying geometric structure [Marsden et al., 1998]. The long-time numerical stability of these multisymplectic integrators has been proven over exponential time scales [Reich, 1999, Hairer et al., 2006] where the computed energy of the system oscillates around its true value rather than accumulating error. The accuracy of a variational integrator depends on how well the discrete Lagrangian approximates the continuous one. Leok and Shingel [2012] and Hall and Leok [2015] demonstrate a Galerkin approach, approximating trajectories with a finite-dimensional function space, that can achieve arbitrarily high integration order. In contrast to these discrete constructions, our ELM method can be understood as a least-squares collocation of the strong-form Euler–Lagrange equation, specifically designed for learned Lagrangian field theories. Whereas classical variational integrators solve the discrete Euler–Lagrange equations of a discrete action constructed from a known Lagrangian, ELM minimizes the squared Euler–Lagrange residual on space-time patches using a learned Lagrangian, evaluated at a set of quadrature points. When this residual is zero, the integrator inherits the symplectic guarantees of classical theory (Section 3.4). In practice, we oversample the quadrature (using more points than required for symplecticity), trading exact symplecticity for stability of the patch solver and robustness to imperfections in the learned Lagrangian.

3 Methodology

Given a configuration manifold Q and coordinate space $X \subseteq \mathbb{R}^{n+1}$, suppose we have a learned map from the first jet bundle $\mathcal{L} : J^1(X, Q) \to \mathbb{R}$. We want to find the path $q : X \to Q$ such that the action

$$S[q] = \int_X \mathcal{L}(q(\xi), q_{\xi_0}(\xi), \ldots, q_{\xi_n}(\xi)) \, \mathrm{d}X \tag{1}$$

is stationary, where $\xi = (\xi_0, \ \ldots \ , \xi_n) \in X$ and $q_{\xi_i} = \partial q / \partial \xi_i$. In Lagrangian mechanics, $\mathcal{L}$ is the Lagrangian density and S is the action. As an example, for the 1D wave equation, $\mathcal{L}(q, q_t, q_x) = \frac{1}{2}\mu \, q_t^2 - \frac{1}{2}c^2 q_x^2$. For simplicity, this methodology is formulated assuming a PDE with one spatial and one temporal dimension ($(t, x) \in X = \mathbb{R}^2$). However, the formulation for ODEs and higher-dimensional PDEs proceeds in an identical manner.

Standard variational methods discretize the action $S[q]$ to form an integrator. This requires symbolic manipulation of $\mathcal{L}$ and so is not possible for learned Lagrangians. ELM instead represents a new subcategory of integrator: a collocation of the strong-form Euler–Lagrange equation. This allows a learned Lagrangian to be simulated directly, without converting to momentum space or inverting the Hessian of $\mathcal{L}$ with respect to the velocities. While ELM can equally simulate an analytical Lagrangian, its benefits are specific to learned, black-box Lagrangians. ELM is a minimization-based integrator designed for learned models, constructed at three scales (Figure 1). A local, pointwise residual measuring the deviation from the learned dynamics (Section 3.1) defines the error induced by a path approximated from a finite set of nodes (Section 3.2). Starting from an initial guess, a damped Newton solve swept via Jacobi iteration across overlapping patches advances the global state without a coupled solve (Section 3.3). Constructed on variational principles, this integrator is symplectic (to quadrature error) when the residual vanishes and structure-preserving in practice (Section 3.4).

3.1 Local Residual

Suppose we have some variation δq from the true path q such that $\delta q = 0$ on the boundary of the integration domain. Since $S[q]$ is a functional, a necessary condition for a minimum to exist is when its first variational derivative, δS, equals zero:

$$\delta S = \iint \left(\frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial q_t} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial \mathcal{L}}{\partial q_x} \right) \delta q \, \mathrm{d}t \, \mathrm{d}x = 0.$$

Since δq can be any function, by the Fundamental Lemma of the calculus of variations, $\delta S = 0$ only when the quantity in the brackets is zero everywhere on X. We denote this term as R:

$$R[q \mid (t, x)] \equiv \left(\frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial q_t} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial \mathcal{L}}{\partial q_x} \right) \Bigg|_{(t,x)}. \tag{2}$$

$R = 0$ is the Euler–Lagrange equation. Given an approximate path $\hat{q}$, the squared Euler–Lagrange residual $R^2[\hat{q} \mid (t, x)]$ is a measure of the deviation from the dynamics represented by $\mathcal{L}$.

3.2 Patch Error

Suppose we have a set of m coordinates and a set of corresponding field values and their derivatives:

$$\mathcal{X} = [\ (t_1, x_1) \quad \ldots \quad (t_m, x_m)\ ] \in X^m,$$
$$\mathcal{Q} = [\ q(\mathcal{X}) \quad q_t(\mathcal{X}) \quad q_x(\mathcal{X}) \quad q_{tx}(\mathcal{X})\ ]^T \in \mathbb{R}^{4m}.$$

Here $q(\mathcal{X}) = [\ q(t_1, x_1) \quad \ldots \quad q(t_m, x_m)\ ]$ denotes the m values taken by q at the coordinates $\mathcal{X}$. Together, $\mathcal{X}$ and $\mathcal{Q}$ define a patch. For a set of $4m$ functions $\mathcal{F} = \{f_1, \ldots, f_{4m}\}$, $f_i : X \to \mathbb{R}$, chosen so that Equation (3) is nonsingular, we construct a Hermite interpolation $\hat{q}$ by fitting $\mathcal{F}$ to this patch. The interpolation coefficients $\theta \in \mathbb{R}^{4m}$ are found by solving

$$\begin{bmatrix} f_1(\mathcal{X}) & \frac{\partial f_1(\mathcal{X})}{\partial t} & \frac{\partial f_1(\mathcal{X})}{\partial x} & \frac{\partial^2 f_1(\mathcal{X})}{\partial t \partial x} \\ \vdots & \vdots & \vdots & \vdots \\ f_{4m}(\mathcal{X}) & \frac{\partial f_{4m}(\mathcal{X})}{\partial t} & \frac{\partial f_{4m}(\mathcal{X})}{\partial x} & \frac{\partial^2 f_{4m}(\mathcal{X})}{\partial t \partial x} \end{bmatrix}^T \theta = \mathcal{Q}. \tag{3}$$

The resulting best fit function is denoted as $\hat{q} = \sum_{i=1}^{4m} \theta_i f_i$, a map from X to $\mathbb{R}$.

We define the residual accumulated by this interpolation as the patch error:

$$J(\mathcal{Q}, \mathcal{X}) \equiv \iint_{\mathrm{Conv}(\mathcal{X})} R^2[\hat{q} \mid (t, x)] \, \mathrm{d}t \, \mathrm{d}x \approx \sum_{i=1}^{n_\omega} \omega_i R^2 \left[\hat{q} \;\middle|\; (\widetilde{t}_i, \widetilde{x}_i)\right], \tag{4}$$

where $\mathrm{Conv}(\mathcal{X})$ is the convex hull of $\mathcal{X}$, from which we select n_ω points $(\widetilde{t}, \widetilde{x})$ and associated weights ω. Throughout this work, we take $\mathcal{F}$ to be a monomial basis in both coordinates (e.g. $\{t^a x^b \mid a \leq 3,\ b \leq 5\}$) for simplicity. Since $\mathcal{F}$ is the tensor-product monomial basis, Equation (3) is exactly two classical one-dimensional Hermite interpolations, guaranteeing the existence and uniqueness of its solution.

3.3 Global Integrator

To advance the solution by one time step, we seek the node $\gamma \in J^1(X, Q)$ at a new coordinate $\eta \in X$ that minimizes the patch error given the known states. This defines the integrator G:

$$G(\mathcal{Q}, \mathcal{X}, \eta) \equiv \arg\min_{\gamma} \; J([\mathcal{Q}; \gamma], \; [\mathcal{X}; \eta]). \tag{5}$$

While this minimization can be implemented in a variety of ways, we use damped Newton iteration for simplicity (Appendix A demonstrates that simple gradient descent is insufficient). Given an initial guess $\gamma^{(0)}$ extrapolated from previous time steps, the update is

$$\gamma^{(k+1)} = \gamma^{(k)} - \lambda \left(\nabla_\gamma^2 \left(J(\gamma^{(k)})\right)\right)^{-1} \nabla_\gamma \left(J(\gamma^{(k)})\right), \tag{6}$$

where $\lambda \in (0, 1]$ is a damping parameter. The choice of damping parameter is highly dependent on the patch overlap and the dynamical coupling of the system; for ODEs, $\lambda = 1$ is typically fine, whereas for 2D PDEs, $\lambda = 0.5$ is often needed. The number of required iterations, by contrast, is mostly dependent on the behavior of the system near the minimum and the selection of λ.

3.4 Structure and Symplecticity

Given a system whose dynamics are governed by an ODE and whose Lagrangian is given by $\mathcal{L}(q, q_t)$, we employ the Gauss–Legendre (GL) quadrature on n_ω points, which exactly integrates polynomials up to degree $2n_\omega - 1$. We define the discrete action on the patch $[t_0, t_1]$ as

$$S_d(q(t_0), q_t(t_0), q(t_1), q_t(t_1)) \equiv \sum_{i=1}^{n_\omega} \omega_i \, \mathcal{L}(\hat{q}(\widetilde{t}_i), \hat{q}_t(\widetilde{t}_i)).$$

Unlike classical discrete methods that advance $q \in Q$, ELM advances a state (q, q_t) on the tangent bundle TQ. From this state, the momentum is given by $p = \frac{\partial \mathcal{L}}{\partial q_t}$ and the differential of the discrete action can be expressed as

$$\mathrm{d}S_d = p(t_1) \, \mathrm{d}q(t_1) - p(t_0) \, \mathrm{d}q(t_0) + \sum_{i=1}^{n_\omega} \omega_i \, R[\hat{q} | \widetilde{t}_i] \, \mathrm{d}\hat{q}(\widetilde{t}_i) + \mathcal{O}(\Delta t^{2n_\omega + 1}). \tag{7}$$

We refer the interested reader to Appendix B for the full derivation.

Assume that the Newton iteration in Equation (6) converges such that $J = 0$. Since J is the sum of the squared residuals and $R^2 \geq 0$, this requires that $R = 0$ at each GL quadrature point and thus Equation (7) reduces to $\mathrm{d}S_d = p(t_1) \, \mathrm{d}q(t_1) - p(t_0) \, \mathrm{d}q(t_0)$. Under these conditions, the corresponding update can be represented by a type-1 generating function [Marsden and West, 2001] to within GL quadrature error. In Section 4.1, we verify that, with the appropriate choice of quadrature, J can be driven to within machine precision of 0. Specifically, $n_\omega = 2$ is chosen for ODEs, resulting in an integrator with order $2n_\omega = 4$ matching the order of the quadrature. For Lagrangians with a constant mass matrix, the quadrature error vanishes exactly to result in a symplectic method:

Proposition 1. *Let $\mathcal{L}(q, q_t)$ be C^2 with nonsingular Legendre transform, and consider one ELM step in which the Newton iteration converges to $J = 0$. If $\mathcal{L} = \frac{1}{2} q_t^\top M q_t - V(q)$ with M constant and symmetric, the resulting map preserves the canonical two-form and is thus symplectic. For a general $\mathcal{L}$, the resulting map is symplectic to within Gauss–Legendre quadrature error.*

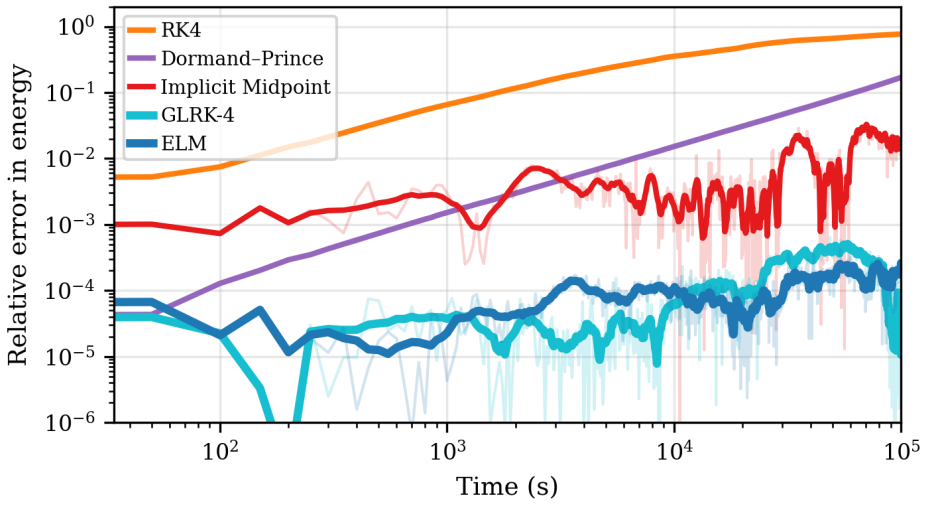


Figure 2: Relative error in energy $|E_t - E_0|/E_0$ over 100,000 s for a double pendulum with initial angles $q_1 = \frac{3\pi}{7}$ and $q_2 = \frac{3\pi}{4}$. The double pendulum has an analytical Lagrangian $\mathcal{L} = q_{1,t}^2 + \frac{1}{2}q_{2,t}^2 + q_{1,t}\, q_{2,t} \cos(q_1 - q_2) + 2\cos q_1 + \cos q_2$. Results are shown for ELM and for four classical integrators using an LNN-style estimate for the acceleration. All methods share the same learned Lagrangian. Faint lines are the raw values while bold lines show a moving average (11-point window) to remove the oscillations characteristic of symplectic integrators.

The proof is given in Appendix C; the general case follows from Appendix B.

In principle, the argument on symplecticity above could be extended to PDEs. In this work, however, we use a larger n_ω to keep the Jacobi iteration stable. This oversampling prevents J from reaching 0 except in edge cases where $q \in \operatorname{span} \mathcal{F}$. Since ELM is a minimization-based integrator with a symplectic construction, we refer to ELM as being structure-preserving. For PDEs, we conjecture the following:

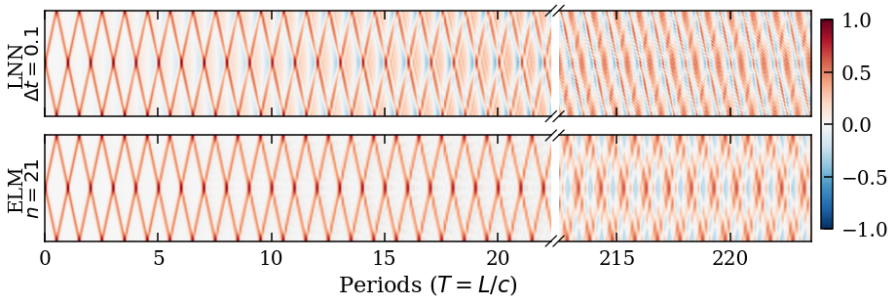
Conjecture 1. *(i) When ELM converges such that $J = 0$ for a PDE with a constant mass, the resulting step is symplectic. (ii) When ELM converges such that $J > 0$, the resulting symplectic defect is governed primarily by the residual R.*

Numerical evidence for both parts is given in Appendix C. We leave the bounding of the convergence of the Newton iteration and the backward error analysis for future work.

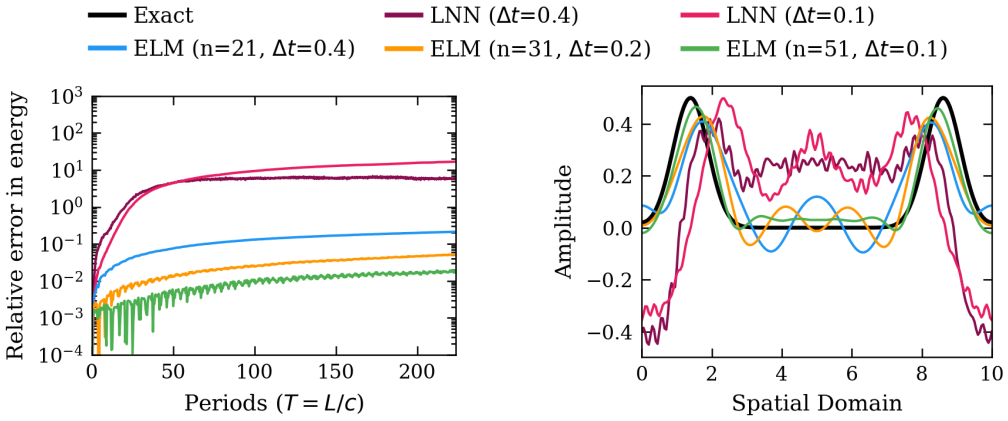
4 Experiments

4.1 Double Pendulum

We first verify that ELM's structure for ODEs holds in practice. Consider the chaotic double pendulum, governed by a nonlinear ODE. Its coupled dynamics and complicated Lagrangian make it a standard benchmark for long-horizon rollouts. Figure 2 shows the relative energy error over the full trajectory on a log-log scale. We compare ELM against four standard integrators using an LNN-style estimate for q_{tt} with the same underlying learned model of the system. The non-symplectic methods (the classical Runge–Kutta (RK4) and Dormand–Prince) result in continual and steady energy accumulation. In contrast, symplectic methods conserve a discrete Hamiltonian, so while they do oscillate, they remain bounded over long time-scales. We see in the figure that ELM matches the fourth-order Gauss–Legendre Runge–Kutta method (GLRK, Hairer et al., 2006) since ELM is also a fourth-order method (when optimized over two GL points).



(a) Spatio-temporal heatmaps of wave amplitude, showing the first ∼20 and the last ∼10 periods.



(b) Relative error in energy $|E_t - E_0|/E_0$.

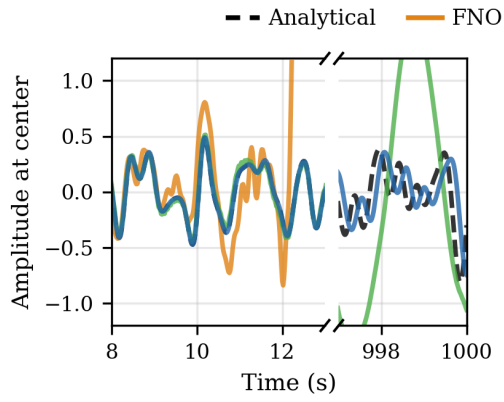
(c) Spatial profile $q(x)$ at $t = 1000$ s.

Figure 3: (a) Wave amplitude heatmaps, (b) relative error in energy, and (c) spatial profile for the 1D wave equation $q_{tt} = 0.05\, q_{xx}$, with periodic boundary conditions, and a Gaussian initial condition. LNN uses a velocity-Verlet integrator with two time-step sizes. ELM is shown at three spatio-temporal resolutions. Parameters for each method and training procedure are listed in Appendices A and D.

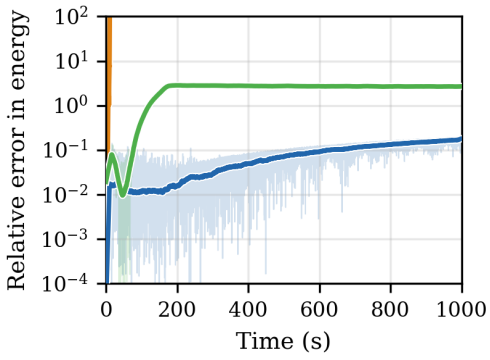
4.2 1D Wave Equation

Having shown the symplectic underpinnings of ELM for ODEs, we now demonstrate the strength of ELM as a discretization-agnostic, structure-preserving integrator that allows learned Lagrangian density models to be applied to PDEs for the first time. We compare the LNN wave model [Cranmer et al., 2020] against ELM on a 1D wave equation, both using learned Lagrangians of equivalent Multilayer Perceptron (MLP) architectures. LNN learns a discrete Lagrangian on a fixed discretization and integrates all nodes as a single ODE. Different from LNN, ELM learns the continuous Lagrangian density and integrates the learned Lagrangian using the methodology described in Section 3.

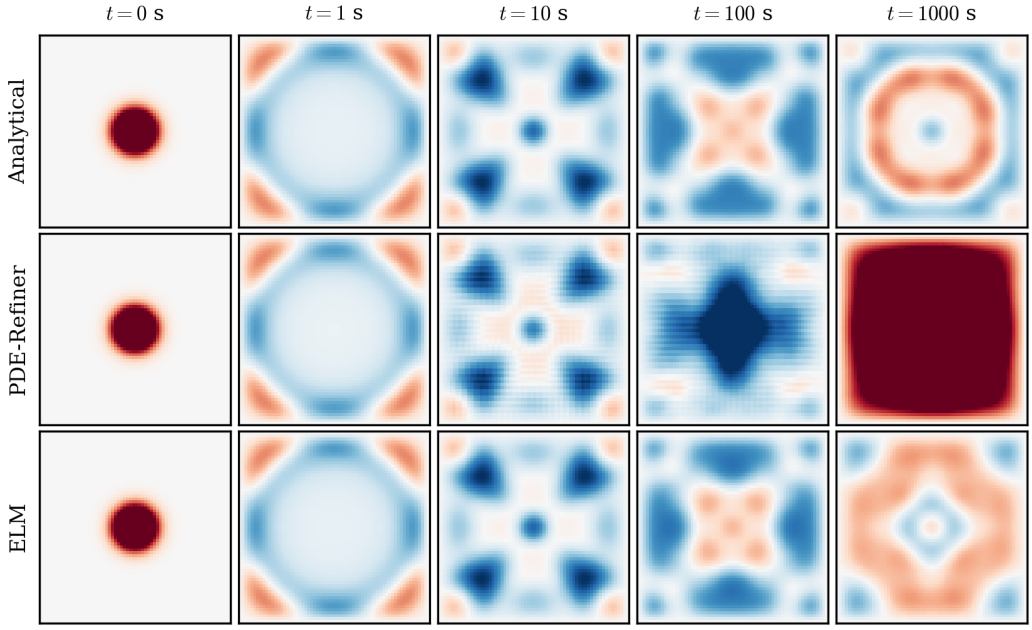
Over 10,000 s, the behavior of the two methods sharply diverges (Figure 3). Even at the coarsest settings, using only 21 spatial nodes, Figure 3a shows how ELM maintains discernible and symmetric wavefronts over longer periods while LNN devolves sooner. Figure 3b shows several orders of magnitude improvement in energy conservation, and the snapshots in Figure 3c demonstrate the increased spatial fidelity of ELM. Both improve as the number of nodes increases and time step size decreases. In contrast, the fixed error induced by the learned discretization in LNN leads to diminishing returns when using smaller step sizes. This highlights a major advantage of learning a continuous Lagrangian: we separate model errors from integration errors. Since *every* well-conditioned learned Lagrangian is a valid Lagrangian and represents some dynamics, the error in the learned model can only contribute to phase error while all energy drift is attributable to the integrator. Some additional results and discussion are provided in Appendix E, including heatmaps for the exact solution and additional integrators, plus the relative L^2 error over the full 10,000 s. Appendix F demonstrates that ELM's separation of model and integration error extends to certain classes of nonlinear PDEs.



(a) Center-node displacement $q(0, 0, t)$ showing $\sim$5 s before FNO diverges and the last $\sim$3 s simulated.

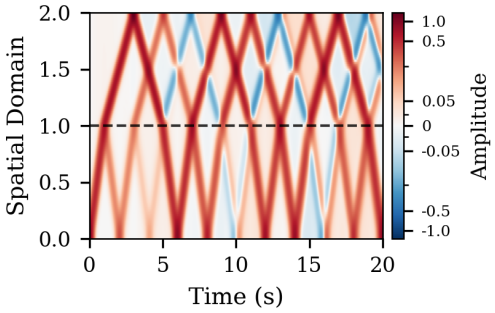


(b) Relative energy error $|E_t - E_0|/E_0$. Faded lines are raw values; bold lines show a moving average over a 10 s window.

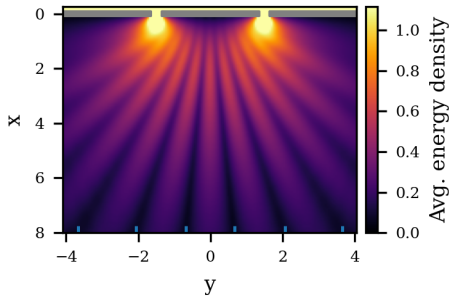


(c) Spatial snapshots $q(x, y)$ at $t = 0, 1, 10, 100, 1000$ s. Top: eigenmode ground truth. Middle: PDE-Refiner. Bottom: ELM. FNO is omitted as it diverges at $t \approx 22$ s.

Figure 4: (a) Center amplitude, (b) relative energy error, and (c) spatial snapshots in time for the 2D wave equation $q_{tt} = q_{xx} + q_{yy}$, using a 31 $\times$ 31 grid with Gaussian initial condition, and Dirichlet boundary conditions over 1,000 s ($\sim$350 periods of the fundamental mode). ELM uses a learned Lagrangian density; FNO and PDE-Refiner are autoregressive baselines on a 64 $\times$ 64 grid. The eigenmode ground truth is found using a 50 $\times$ 50 mode decomposition. Parameters for each method and training procedure are listed in Appendices A and D.



(a) Spatio-temporal heatmap of a wave traversing a sigmoid impedance interface at $x = 1$ (dashed line).



(b) Time-averaged energy density for a driven wave hitting two slits, demonstrating interference.

Figure 5: Demonstrations of (a) partial reflection and (b) diffraction and interference. Both panels use models trained only on homogeneous data with simple boundary conditions. Boundary effects are realized purely through the spatial variation of the Lagrangian or constraints on degrees of freedom. In Figure (b), the Fraunhofer minima for the fringes in this setup are shown as blue tick marks.

4.3 2D Wave Equation

To enable ELM to integrate systems of higher dimensions, only mechanical changes are needed: the basis functions must incorporate the new coordinate and one must integrate over it. Otherwise, everything proceeds identically. Unlike the periodic boundaries used for the 1D wave equation, the 2D wave equation is simulated in a box with Dirichlet boundary conditions, $q = 0$. On boundary nodes, q is fixed and only the spatial derivatives are optimized, using the nearest complete patch; this leaves $q = 0$ as a hard constraint.

Figure 4 compares ELM against a Fourier Neural Operator (FNO, Li et al., 2021) and PDE-Refiner [Lippe et al., 2023] over $1{,}000\,\mathrm{s}$. The center-node trace (Figure 4a) shows that ELM tracks the analytical ground truth with only minor phase drift and energy loss, while both FNO and PDE-Refiner have identifiable drift within the first $20\,\mathrm{s}$. The energy drift shown in Figure 4b tells a similar story. While PDE-Refiner fails more gracefully than FNO, it still devolves into a non-physical global oscillation seen in Figure 4c. ELM achieves better overall accuracy and energy conservation with $78{,}000$ times fewer model parameters than PDE-Refiner. The L^2 error is provided in Appendix G.

To the best of our knowledge, this is the first work to simulate a PDE with two spatial dimensions using a learned Lagrangian density over long time-horizons.

4.4 Emergent Boundary Effects

Both PDE results used identical boundary conditions in training as in inference. Since learned continuous Lagrangians are pointwise, models compose and transfer to new domains without retraining.

We first demonstrate this in 1D. Suppose we have two learned Lagrangians independently trained on separate homogeneous media with different wave speeds and we want to explore what happens when they are abutted. Here, we represent the transition as a sharp sigmoid blend between the two Lagrangians at $x = 1$. Figure 5a shows the result: the wave partially reflects and partially transmits at the interface, exactly as waves behave when they hit an impedance barrier. Compared to the analytical solution with a step boundary, the learned model accumulated only $\sim 1\%$ energy drift over $20\,\mathrm{s}$, with stable behavior at the interface. The domain edges use homogeneous Neumann boundary conditions.

The same flexibility extends to 2D. Figure 5b shows the mean energy over time when a driven wave hits a barrier with two slits, producing interference fringes whose y-positions are within 0.083 ($\sim 6\%$ of the fringe spacing) of those predicted by theory. This example has three types of boundary conditions: a sinusoidally driven input on one edge, an internal wall with Dirichlet boundary conditions, and absorbing boundaries on all others. Unlike the prior results, this configuration has driven and absorbing edges, so results demonstrate steady-state behavior. This uses the identical model from Section 4.3; barriers, slits, and driven boundaries are enforced purely through constraints on the degrees of freedom (see Appendix H). The operator baselines of Section 4.3 have no comparable method for transferring to this domain.

5 Discussion and Conclusion

We have presented Euler–Lagrange Minimization (ELM), a novel, structure-preserving integrator that uses learned Lagrangians for forward simulation of ODE and PDE systems. Conceptually, ELM separates *what* a system's physics are (the Lagrangian density) from *how* to simulate them (the variational integrator). ELM achieves this through a construction at three scales: the local squared Euler–Lagrange residual which measures deviation from the learned dynamics, a patch error which describes the error induced by a finite basis over known states, and a minimization of this error via damped Newton iteration. When the residual reaches zero, the integrator is symplectic for Lagrangians with a constant mass matrix, and symplectic up to quadrature error otherwise. When it does not, the integrator remains structure-preserving, maintaining orders of magnitude less energy drift over long rollouts compared to existing methods. For PDE systems in this work, we oversample the quadrature, trading exact symplecticity for the stability of the patch solver.

Our results show that ELM achieves energy conservation comparable to the symplectic GLRK integrator for the chaotic double pendulum. On 1D waves, ELM outperforms LNN at similar computational cost and, unlike LNN, scales accuracy with resolution. ELM also constitutes the first simulation of a 2D PDE using a learned Lagrangian density, outperforming operator baselines with orders of magnitude fewer parameters. More broadly, ELM avoids baking discretization into the learned model, so accuracy and compute can be traded at inference time.

A further consequence of pairing a learned continuous Lagrangian with local minimization is that models compose and transfer across domains without retraining. Spatial variation in the Lagrangian produces physically correct reflection and transmission at impedance interfaces; constraints on degrees of freedom realize Dirichlet, Neumann, driven, and absorbing boundaries through the same integrator. The identical model used for the homogeneous 2D wave produces double-slit interference when given a barrier and a driven edge. Operator-based methods, which fuse dynamics with discretization, cannot transfer this way. By not leaving the integrator as an implementation detail, a single trained model behaves as a physics primitive.

ELM's strength lies in the regimes of low data availability, domain transfer and flexibility, and long rollouts. Generating new data when the domain changes is a non-trivial task: operator methods such as FNO and PDE-Refiner may excel when the test domain matches the training domain, but have little to no adaptability when it changes, so new design requirements or geometries require the acquisition of an entirely new dataset. ELM can perform this domain transfer (Section 4.4). If a model is needed for applications such as a control loop, a neural surrogate may be trained from the resulting output. Additionally, in many experimental settings (structural vibration, acoustics, etc.) dense grids are essentially unobtainable but sparse, pointwise data are simple to collect and suitable for ELM's pointwise training (Appendix D.5).

Several limitations remain. ELM is currently more expensive per time step than standard integrators, particularly for the ODE case where symplectic methods are already available. The Jacobi sweep converges slowly for strongly-coupled systems, particularly in higher-dimensional PDEs where patches naturally overlap more. Finally, ELM currently accepts only Lagrangian densities with up to first derivatives, excluding systems such as beam mechanics, and its guarantees do not yet cover strong derivative nonlinearities (Appendix F). None of these limitations require rethinking the local minimization framework to address. Extending ELM to dissipative systems, higher-order field theories, and adaptive mesh refinement are natural directions for future work.

Taken together, the results of this paper argue that the integrator deserves the same care as the model to achieve flexible physics-learning systems. ELM offers a starting point for learned models that can be deployed, constrained, and combined at inference time rather than retrained per scenario.

Acknowledgments and Disclosure of Funding

We would like to thank Benjamin Shaffer and all members of the ScalAR lab for their thoughtful comments and discussions during the development of this work. Additionally, we thank our anonymous reviewers for their careful review of the initial draft, which helped sharpen our arguments. This work was supported by ONR Award N000142512171. The authors do not have any competing interests.

References

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, Y. Katariya, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

J. Brandstetter, D. Worrall, and M. Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations (ICLR)*, 2022.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer, Berlin Heidelberg, 2 edition, 2006.

J. Hall and M. Leok. Spectral variational integrators. *Numerische Mathematik*, 130(4):681–740, 2015.

A. U. Hamzaogullari and A. Ozakin. Learning relativistic geodesics and chaotic dynamics via stabilized Lagrangian neural networks. *arXiv preprint arXiv:2601.12519*, 2026.

P. Kidger and C. Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. In *NeurIPS 2021 Workshop on Differentiable Programming*, 2021.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

M. Leok and T. Shingel. General techniques for constructing variational integrators. *Frontiers of Mathematics in China*, 7(2):273–303, 2012.

Y. Li, Y. Yang, and C. Allen-Blanchette. Frequency-separable Hamiltonian neural network for multi-timescale dynamics. *arXiv preprint arXiv:2603.06354*, 2026.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

P. Lippe, B. Veeling, P. Perdikaris, R. Turner, and J. Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

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

M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations (ICLR)*, 2019.

J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10: 357–514, 2001.

J. E. Marsden, G. W. Patrick, and S. Shkoller. Multisymplectic geometry, variational integrators, and nonlinear PDEs. *Communications in Mathematical Physics*, 199(2):351–395, 1998.

G. Mur. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations. *IEEE Transactions on Electromagnetic Compatibility*, EMC-23 (4):377–382, 1981.

C. Offen and S. Ober-Blöbaum. Learning of discrete models of variational PDEs from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(1):013104, 2024.

T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

S. Reich. Backward error analysis for numerical integrators. *SIAM Journal on Numerical Analysis*, 36(5):1549–1570, 1999.

S. Saemundsson, A. Terenin, K. Hofmann, and M. Deisenroth. Variational integrator networks for physically structured embeddings. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3078–3087. PMLR, 2020.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, pages 8459–8468. PMLR, 2020.

J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*, volume 12 of *Texts in Applied Mathematics*. Springer, New York, 3 edition, 2002.

N. Thuerey, B. Holzschuh, P. Holl, G. Kohl, M. Lino, Q. Liu, P. Schnell, and F. Trost. *Physics-based Deep Learning*. WWW, 2021. URL `https://physicsbaseddeeplearning.org`.

S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021.

R. Yu and R. Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences*, 121(27):e2311808121, 2024.

A Computational Cost Summary

Table 1: Configuration, computational cost, and maximum relative energy error for each method in Section 4. Methods sharing a letter in the **Net** column use the same architecture and the same trained weights. More details on architecture and training can be found in Appendix D. Computational cost excludes JIT compilation. The double pendulum experiments run on CPU (float64) while wave experiments run on GPU (float32). Entries reported as mean $\pm$ standard deviation are over ten unselected seeds (0–9), each retraining the network with the same recipe on training data regenerated per seed; costs are for the original models shown in the figures.

Method	**Net**	**Params**	**Nodes**	Δt	**Steps**	**ms/step**	**Total**	$\lvert\Delta E/E_0\rvert$
Double pendulum (100,000 s; ELM n_r=3, $n_\omega = 2$)								
RK4	**A**		—	0.02	5,000,000	0.18	924 s	$71.8 \pm 7.4\%$
Dormand–Prince	**A**		—	adaptive	—	—	944 s	$16.91 \pm 0.23\%$
Implicit Midpoint	**A**	163,604	—	0.02	5,000,000	0.41	2,072 s	$6.1 \pm 2.1\%$
GLRK	**A**		—	0.02	5,000,000	0.76	3,815 s	$0.0267 \pm 0.0059\%$
ELM	**A**		—	0.02	5,000,000	2.4	11,977 s	$0.039 \pm 0.020\%$
1D wave (periodic boundaries; 10,000 s; ELM $n_r = 3$, $n_\omega = 3$ in time and 6 in space)								
LNN Δt=0.4	**B**	324,001	100	0.4	25,000	5.7	141 s	680%
LNN Δt=0.1	**B**		100	0.1	100,000	5.5	549 s	1,680%
ELM n=21	**C**		21	0.4	25,000	9.4	235 s	$17.6 \pm 3.9\%$
ELM n=31	**C**	322,801	31	0.2	50,000	9.6	481 s	$4.1 \pm 2.4\%$
ELM n=51	**C**		51	0.1	100,000	15	1,500 s	$4.1 \pm 4.0\%$
2D wave (Dirichlet boundaries, 1,000 s; ELM $n_r = 20$, $n_\omega = 3$ in time and 5×5 in space)								
ELM	**D**	1,881	31×31	0.02	50,000	968	48,400 s	19%
PDE-Refiner	**E**	∼146M	64×64	0.02	50,000	26	1,300 s	271%
FNO	**F**	138,301	64×64	0.02	50,000	2.9	145 s	diverges[†]

[†] NaN at $t = 21.9$ s; energy exceeds 10^{36} before divergence.

Newton iteration. The damped Newton iteration (specifically the repeated construction of the Hessian) is the main computational cost. While nothing in the method inherently requires this choice of optimizer, the resulting quadratic convergence is highly beneficial. Empirically, this is because the long-term behavior of the integrator relies on a good convergence of each step, and quadratic methods get closer to the true minimum than linear ones. Additionally, since every step slightly changes the minimum for the adjacent nodes, the larger Newton steps allow us to converge in drastically fewer iterations. The coupling also means that the Newton step must be damped slightly to prevent oscillations between iterations.

To provide numerical context: for the n=31 1D wave equation, if we use a simple gradient descent and tune the step size to the largest value that remains stable (smaller step sizes do not show better behavior), after $25{,}000$ steps the resulting residual ($\sim 10^{-3}$) is still roughly an order of magnitude higher than the residual after three Newton steps ($\sim 10^{-4}$).

Cost relative to neural surrogates. While FNO and PDE-Refiner are presented for comparison, these methods are neural surrogates while ELM is not. Instead of trying to provide a cost-efficient replacement for physical simulation, ELM learns a quantity that allows for the physical simulation of the system, and is best understood as a learned implicit solver. Actually simulating the dynamics gives the domain transfer of Section 4.4, but comes with additional computational cost. This cost should be balanced against the time and compute required to collect or generate large amounts of dense data and train a large model.

B Deriving the Generating Function

Our goal is to relate the differential of the discrete action $\mathrm{d}S_d$ to the Euler–Lagrange residual R.

Lemma 1. *The differential of the Lagrangian* $\mathrm{d}\mathcal{L}$ *can be expressed in terms of the residual* R *as*

$$\mathrm{d}\mathcal{L} = R\,\mathrm{d}q + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\right).$$

Proof. We proceed via product rule, noting that $\frac{\mathrm{d}}{\mathrm{d}t}\,\mathrm{d}q = \mathrm{d}q_t$:

$$\begin{aligned} R\,\mathrm{d}q + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\right) &= \left(\frac{\partial \mathcal{L}}{\partial q}\,\mathrm{d}q - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\right) + \left(\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q + \frac{\partial \mathcal{L}}{\partial q_t}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{d}q\right) \\ &= \frac{\partial \mathcal{L}}{\partial q}\,\mathrm{d}q + \frac{\partial \mathcal{L}}{\partial q_t}\mathrm{d}q_t \\ &= \mathrm{d}\mathcal{L} \end{aligned}$$

□

We now proceed in deriving the generating function form shown in Equation (7):

$$\begin{aligned} \mathrm{d}S_d &= \sum_{i=1}^{n_\omega} \omega_i\,\mathrm{d}\mathcal{L} \\ &= \sum \omega_i R\,\mathrm{d}q + \sum \omega_i \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\right) \\ &= \sum \omega_i R\,\mathrm{d}q + \int_{t_0}^{t_1} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\right)\,\mathrm{d}t + \mathcal{O}\left(\Delta t^{2n_\omega+1}\right) \\ &= \sum \omega_i R\,\mathrm{d}q + \frac{\partial \mathcal{L}}{\partial q_t}\,\mathrm{d}q\,\Big|_{t_0}^{t_1} + \mathcal{O}\left(\Delta t^{2n_\omega+1}\right) \\ &= \sum \omega_i R\,\mathrm{d}q + p(t_1)\,\mathrm{d}q(t_1) - p(t_0)\,\mathrm{d}q(t_0) + \mathcal{O}\left(\Delta t^{2n_\omega+1}\right). \end{aligned}$$

Here R and $\mathrm{d}q$ inside the sums are evaluated on $\hat{q}$ at the quadrature points $\widetilde{t_i}$, as in Equation (7). Transitioning between each line, this is the result of the following steps:

1. Applying Lemma 1.
2. Using the standard approximation error of GL quadrature.
3. Applying the fundamental theorem of calculus.
4. Using the Legendre transformation to obtain the momenta $p(t_0)$ and $p(t_1)$.

As discussed in Section 3.4, when $R \to 0$ and the step size Δt is sufficiently small, this means that the update induced by ELM can be represented by a type-1 generating function to within the $\mathcal{O}(\Delta t^{2n_\omega+1})$ term. Thus, it is symplectic to within quadrature error. This error term is the standard Gauss–Legendre approximation error [Stoer and Bulirsch, 2002]. For $n_\omega = 2$, the resulting defect is only $\frac{\Delta t^5}{4320}$ times a bound on the fourth time-derivative of the underlying integrand, for sufficiently smooth $\mathcal{L}$. For Lagrangians with a constant mass matrix, Appendix C shows that this approximation error vanishes from the symplectic defect.

C Exact Symplecticity for Constant-Mass Lagrangians

This appendix proves Proposition 1 and gives numerical evidence for Conjecture 1.

Proof of Proposition 1. For a time step Δt, denote the tangent map of the ELM step as A. For a Lagrangian $\mathcal{L} = \frac{1}{2} q_t^\top M q_t - V(q)$, we are trying to show that A preserves the canonical two-form $\Omega = M\,\mathrm{d}q \wedge \mathrm{d}q_t$.

Let $u(t), w(t)$ be the variations of $\hat{q}$ along two directions of the state (q, q_t) and let $u_0 = (u(0), u_t(0))$ and $u_1 = (u(\Delta t), u_t(\Delta t)) = A u_0$. Since $\hat{q}$ is a cubic depending linearly on its endpoint data, u and w are likewise cubic functions. As the solve has converged, we can differentiate the residual R with respect to this variation to obtain the variational conditions

$$M u_{tt}(\widetilde{t_i}) = -\nabla^2 V(\hat{q}(\widetilde{t_i}))\,u(\widetilde{t_i}),$$

and likewise for w. We now define the Wronskian as

$$W(t) = u^\top M w_t - w^\top M u_t.$$

Note that this is exactly the form of the two-form Ω. If we can show that $W(0) = W(\Delta t)$, then the map is symplectic. Taking the derivative,

$$W_t(t) = u^\top M w_{tt} - w^\top M u_{tt},$$

we find that since M is symmetric, higher-order terms cancel and W_t is a cubic function of t. Substituting the variational conditions above, $W_t(\widetilde{t}_i) = 0$ by the symmetry of $\nabla^2 V$. Combining everything together, we find that

$$W(\Delta t) - W(0) = \int_0^{\Delta t} W_t(t)\,\mathrm{d}t = \sum_{i=1}^{2} \omega_i W_t(\widetilde{t}_i) = 0$$

exactly, since two-point Gauss–Legendre quadrature of a cubic is exact. As u and w are arbitrary, we conclude that the step induced by ELM is symplectic.

Numerical evidence for Conjecture 1. When the step does not converge such that $J = 0$, the symplectic defect can be expressed as the exterior derivative of Equation (7):

$$\mathrm{d}p_1 \wedge \mathrm{d}q_1 - \mathrm{d}p_0 \wedge \mathrm{d}q_0 = -\sum_{i=1}^{n_\omega} \omega_i\, \mathrm{d}R[\hat{q} \mid \widetilde{t}_i] \wedge \mathrm{d}\hat{q}(\widetilde{t}_i) - \mathrm{d}\,\mathcal{O}(\Delta t^{2n_\omega+1}).$$

This equation only examines the symplecticity of one time step to the next, the update induced by ELM, not the mechanics of ELM. As a result, it applies without modification to PDEs when summing over the domain. It provides two independent ways of measuring the defect, both computed directly via automatic differentiation: (a) the true defect through the tangent map A, $A^\top \Omega A - \Omega$, and (b) the summation of $\mathrm{d}R \wedge \mathrm{d}\hat{q}$. Using the agreement of these two independent measurements as numerical evidence, we evaluate Conjecture 1 numerically. These measurements are computed in float64 rather than the float32 used elsewhere in the paper.

For part (i), we first validate that J reaches 0 in the edge case $q \in \operatorname{span}\mathcal{F}$ noted in Section 3.4. For any cubic polynomials f and g, $q(x,t) = f(x-ct) + g(x+ct)$ is an exact solution of the wave equation lying in $\operatorname{span}\mathcal{F}$. Pinning the boundary nodes to the exact solution (polynomials are not periodic), the converged step attains $J \approx 10^{-28}$ and reproduces the exact flow to $\sim 10^{-15}$. Accordingly, the defect (a) is $\sim 10^{-15}$ (accounting for the flux at the boundaries of the domain), matching (b) at $\sim 10^{-16}$.

For part (ii), for both the linear wave equation and the sine-Gordon equation (Appendix F) in the configurations used in this paper, we roll out from the initial condition for 100 steps and then measure both (a) and (b) for the next step. These two calculations, sharing no intermediate quantities, agree to $\sim 10^{-16}$. Rolling out the Born–Infeld equation in the same way, the two calculations differ by $\sim 0.05\%$. When $J > 0$, at least 99.9% of the symplectic defect for this equation is therefore attributable to the residual R, the quantity minimized by the integrator, even for this nonlinear system.

D Training Procedures

The model and all surrounding code were implemented in JAX [Bradbury et al., 2018] and with the Equinox library [Kidger and Garcia, 2021]. Training and inference were performed on a single machine with an NVIDIA® 5070 Ti.

The remainder of this appendix begins by describing the training and execution for ELM across all systems. Descriptions of architecture and training data follow for each system examined in Section 4. These subsections also describe how comparison methods were trained, where appropriate. Networks are named in line with the **Net** column of Table 1. The remaining subsections provide some further discussion on the data requirements for training ELM.

D.1 Training ELM

The three ELM networks (**A**, **C**, **D**) share a single training procedure. Each network minimizes the mean squared Euler–Lagrange residual (R^2) on a batch of pointwise samples, plus a small set of

gauge constraints that pin down the affine ambiguity of the Lagrangian. For ODEs, for any function $\mathcal{L}(q, q_t)$ which satisfies the Euler–Lagrange equations, the Lagrangian

$$\mathcal{L}^*(q, q_t) = A\mathcal{L}(q, q_t) + Bq_t + C$$

also satisfies the equations for any $A, B, C \in \mathbb{R}$. A similar statement holds for PDEs.

To counteract this ambiguity simply without affecting the solution space, a set of soft gauge constraints are added to the loss:

$$L = R^2 + \left(\frac{\partial^2 \mathcal{L}(0)}{\partial q_t^2} - 1\right)^2 + \left(\frac{\partial \mathcal{L}(0)}{\partial q_t}\right)^2 + \mathcal{L}(0)^2 \tag{8}$$

Combined, these gauge constraints seek to fix $(A, B, C) = (1, 0, 0)$. While R^2 is evaluated across the batch, these gauge constraints only fix $\mathcal{L}$ at the origin. While this is sufficient for the wave equation, more complicated Lagrangians such as the double pendulum require additional assistance.

The first step of any ELM rollout requires the Hermite degrees of freedom at $t = 0$:

1. (q, q_t) for the double pendulum.
2. (q, q_t, q_x, q_{xt}) for the 1D wave.
3. $(q, q_t, q_x, q_y, q_{xt}, q_{yt}, q_{xy}, q_{xyt})$ for the 2D wave.

These can either be computed analytically from the closed-form initial condition or estimated from data, the same way a finite-difference solver seeds its initial half-step. No multi-frame warm-up is required.

D.2 Double-pendulum network (A)

Using the loss described by Cranmer et al. [2020], a generic MLP can learn the double-pendulum Lagrangian directly from (q, q_t, q_{tt}) data. Let $M(q) = \frac{\partial^2 \mathcal{L}}{\partial q_t^2}$ be the mass matrix of the network. Then, the LNN acceleration can be computed as

$$q_{tt} = M(q)^{-1}\left(\frac{\partial \mathcal{L}}{\partial q} - \frac{\partial^2 \mathcal{L}}{\partial q \partial q_t} q_t\right),$$

whose difference against the acceleration in data is minimized during training. By taking the inverse of the mass matrix, this loss implicitly regularizes the network away from eigenvalues close to zero. Without this regularization, the gauge terms in Equation (8) are insufficient to prevent a collapse of the mass matrix in this system for a generic network.

While one solution to this problem might be to either use the original LNN loss or apply more recent work on stabilizing LNN training [Hamzaogullari and Ozakin, 2026], to keep the training procedure simple and consistent we instead adopt a DeLaN inspired architecture for the double-pendulum *only*. The new model is defined as

$$\mathcal{L}(q, q_t) = \tfrac{1}{2} q_t^T M(q) q_t - V(q),$$

where $M(q)$ is produced from a Cholesky factor and $V(q)$ is a separate scalar head. The network reads $(\sin q_0, \cos q_0, \sin q_1, \cos q_1, q_{t,0}, q_{t,1})$ through three hidden layers of width 200 with Softplus activations. It is trained for 200,000 steps on 20,000 samples of (q, q_t, q_{tt}) generated from the analytical double-pendulum Lagrangian, with angles drawn uniformly from $[-\pi, \pi]$ and angular velocities up to $q_t^{\max} = 8$.

The imposition of structure here is accepted for the following reasons:

- We know that a generic MLP can learn a double pendulum and that the resulting model can be used here without modification.
- All comparisons in Section 4.1 use the same network, so ELM does not gain any advantage.
- As ELM represents a structure-preserving integrator, and since any learned Lagrangian represents some set of dynamics, the model error is separate from the integration error and does not compound.

D.3 1D wave networks (B, C)

Networks **B** and **C** share hidden-layer width and differ only in the number of inputs. **B** tries to learn the spatial discretization, while **C** is a strictly pointwise function.

LNN baseline (B). The architecture from Cranmer et al. [2020] consists of an MLP with three hidden layers of width 400 and Softplus activations, taking the 6-tuple $(q(t, x_{i-1}), q(t, x_i), q(t, x_{i+1}), q_t(t, x_{i-1}), q_t(t, x_i), q_t(t, x_{i+1}))$ from a 3-node spatial stencil. This model was trained on 1D wave trajectories ($q_{tt} = 0.05\, q_{xx}$ with periodic boundary conditions) using the released code and training procedure, with the best model from 200 random seeds returned. LNN training is unstable enough that only $\sim$5–10% of seeds converge to a test loss below 1.5.

ELM (C). Training samples are drawn from random superpositions of plane waves on the relevant domain (5 modes, wavenumbers are uniform in $[-k_{\text{bound}}, k_{\text{bound}}]$, frequencies are set by the linear dispersion relation $\omega = c|k|$). All derivatives the loss touches are computed analytically from the closed-form superposition. Training was conducted with Adam with a cosine one-cycle schedule (learning rate $10^{-3} \to 0.2$).

The network is an MLP with three hidden layers of width 400 and Softplus activations, taking the *pointwise* 3-tuple (q, q_t, q_x) at a single space-time point. The model was trained for $50{,}000$ steps on $5{,}000$ plane-wave samples ($k_{\text{bound}} = 4$, $c^2 = 0.05$). To characterize seed variability, we retrained this model on ten unselected random seeds (0–9) under the identical recipe, regenerating the training data per seed. All ten seeds converge and all ten full-horizon rollouts are stable at every resolution, with the energy errors reported in Table 1. Even the worst unselected seed at n=31 (8.8%) conserves energy $78\times$ better than the best-of-200 LNN model at Δt=0.4.

D.4 2D wave networks (D, E, F)

ELM (D). Training samples are drawn identically to the 1D wave case, only this time using a superposition of 2D waves. The network is an MLP with two hidden layers of width 40 and Softplus activations, taking the *pointwise* 4-tuple (q, q_t, q_x, q_y) at a single space-time point. The model was trained for $100{,}000$ steps on $10{,}000$ plane-wave samples ($k_{\text{bound}} = 6$, $c^2 = 1$).

PDE-Refiner (E). A U-Net conditioned on a diffusion-step embedding, trained as a denoiser with 8 denoising steps and a 4-frame input history. The training data is 400 eigenmode-decomposed trajectories of the 2D wave on a 64×64 grid with $\Delta t = 0.02$ s. Half of this training data is a random superposition of modes, half is simulated initial Gaussian pulses. The model was trained for 400 epochs.

FNO (F). A standard Fourier Neural Operator, trained as a one-step predictor with a 10-frame input history. The training data is the same 400 eigenmode trajectories on 64×64 at $\Delta t = 0.02$ s, trained for 500 epochs.

Comparison of data requirements. The disparity between dataset sizes is worth quantifying. Network **D** sees 10^4 pointwise samples, each a 4-tuple of scalars, totalling 4×10^4 scalar values seen during training. Networks **E** and **F** each see 400 trajectories of length T on a 64×64 grid at $\Delta t_{\text{save}} = 0.02$ s, sliding the input/target window across each trajectory. With even a modest trajectory length of $T = 1$ this is on the order of 8×10^8 space-time scalar values per epoch, repeated for hundreds of epochs. ELM is trained on roughly five orders of magnitude less raw data, on a network up to five orders of magnitude smaller, and still produces the greater long time-horizon stability seen in Figure 4.

D.5 The cost of pointwise derivatives

ELM's training data and initial state require pointwise field derivatives, which may seem more demanding than the dense regular-grid snapshots that autoregressive baselines (**E**, **F**) consume. In practice, it is less so. ELM's loss is strictly pointwise: training samples need not lie on a regular grid, share a coordinate frame, or be uniformly spaced. In principle, six sensors placed near each other, sampled over sufficient time, produce all the data needed to train a 2D wave model for ELM. This is

the minimum needed to recover all necessary derivatives by finite differences. The autoregressive baselines, by contrast, are tied to a dense mesh for both training and rollout and require a multi-frame input history (4 frames for PDE-Refiner, 10 for FNO).

To test this, we train a model for the 1D wave equation using only position data recorded at three closely spaced points, with all derivatives estimated by finite differences and the recipe otherwise identical to network **C**. Using the n=31 configuration of Table 1, the final error in energy at $10{,}000\,\mathrm{s}$ is 2.94%, within the seed variability of network **C**.

E Extended 1D Wave Comparison

Figures 6 and 7 extend Section 4.2 with the relative L^2 error and full spatio-temporal heatmaps over the same $10{,}000\,\mathrm{s}$ problem ($q_{tt} = 0.05\, q_{xx}$, periodic boundary conditions, Gaussian initial condition).

In addition to the LNN and ELM-learned configurations of Figure 3, both figures include two compute-matched symplectic baselines on the same n=51, Δt=0.1 discretization: GLRK and ELM-ODE. Both methods share the same fourth-order finite differences to compute spatial derivatives and both still use the same underlying learned Lagrangian. To use GLRK, an LNN-style estimate for the second time derivative is used. ELM-ODE applies the ODE form of ELM (Section 3.4) to the same spatially discretized system.

ELM at all three resolutions mostly preserves wave structure for the full duration, with higher resolution producing sharper wavefronts. LNN at both time steps shows progressive amplitude decay and phase distortion, consistent with the energy drift reported in Figure 3. GLRK and ELM-ODE are virtually identical on the heatmap and L^2 plot, once again validating the symplecticity of ELM on ODEs. These methods also show that much of the distortion exhibited by ELM on the various configs is the inherent phase error in the learned Lagrangian. On short time scales, the joint space-time optimization of ELM achieves lower L^2 error than the space-discretized symplectic integrators. This suggests that minimizing on space-time patches avoids artifacts caused by fixed discretization.

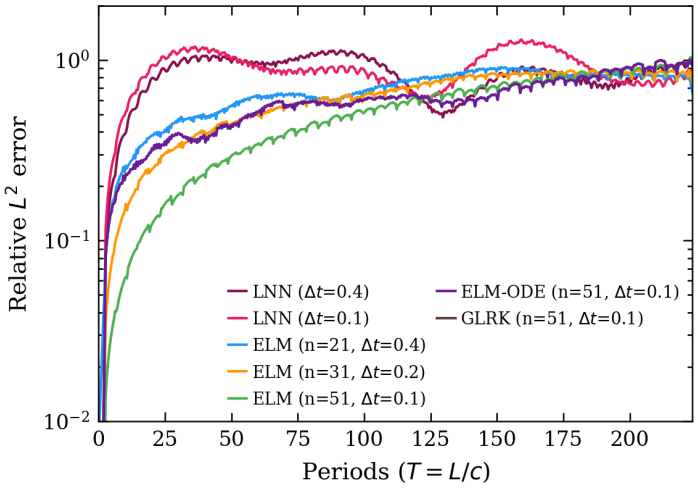


Figure 6: Relative L^2 error vs. exact solution over the full $10{,}000\,\mathrm{s}$ (224 periods) for the methods of Figure 3, plus the GLRK and ELM-ODE symplectic baselines.

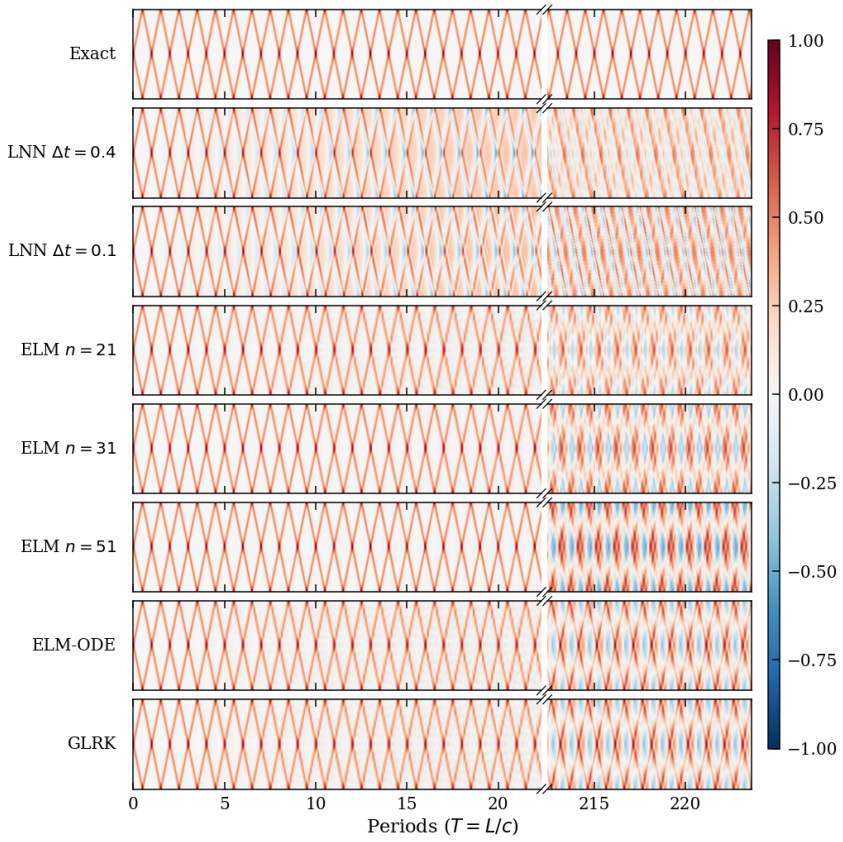


Figure 7: Spatio-temporal heatmaps $q(x,t)$ over $10{,}000\,\mathrm{s}$ (224 periods) extending Figure 3a. The figure shows the first $\sim$20 periods and the last $\sim$10. Parameters for each method are listed in Appendix A.

F Nonlinear Conservative PDEs

The applicability of ELM to nonlinear conservative PDEs depends on what type of nonlinearity we are including. If we restrict ourselves to Lagrangians with a nonlinear potential, ELM can be applied without modification to the method. Consider the sine-Gordon equation:

$$\mathcal{L} = \tfrac{1}{2}q_t^2 - \tfrac{1}{2}q_x^2 - (1 - \cos q).$$

We use the same training recipe as network **C** (Appendix D.3) on data generated from this equation, and run with the same integrator configuration as the 1D wave.

Breather. Given the "breather" solution and using $\omega = 0.6$ and $t = 0$ as initial conditions,

$$q(x,t) = 4\arctan\left[\frac{\eta}{\omega}\,\frac{\sin(\omega t)}{\cosh(\eta x)}\right], \qquad \eta = \sqrt{1-\omega^2},$$

over $1{,}000\,\mathrm{s}$ ($\sim$95 periods), using the same model, accuracy improves with resolution, just as with the linear wave model (Figure 8, Table 2).

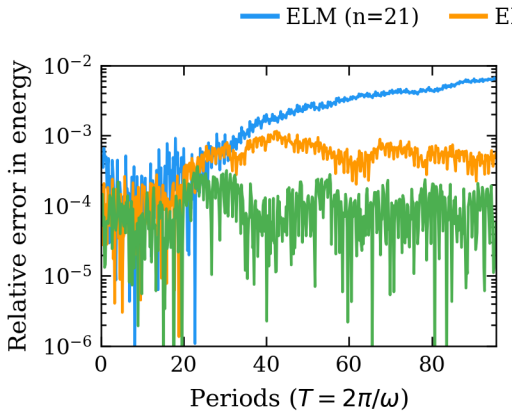
Kink–antikink collision. Similarly, we can simulate the kink–antikink collision with $v = 0.25$, initialized from the closed-form solution

$$q(x,t) = 4\arctan\left[\frac{\sinh(\gamma v t)}{v\cosh(\gamma x)}\right], \qquad \gamma = \frac{1}{\sqrt{1-v^2}},$$

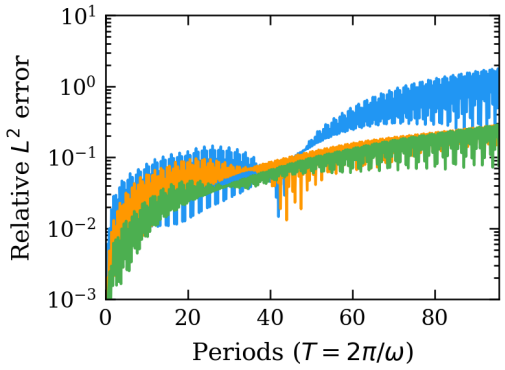
at $t = -12\,\mathrm{s}$, so that the kinks collide at $t = 0$. Using $n = 31$ and simulating for $24\,\mathrm{s}$, the kinks pass through each other elastically (Figure 8c); the error in energy across the collision is 0.18% with a maximum error of 0.61%, the final relative L^2 error is 5.0×10^{-2}, and the post-collision kink centers are recovered to within 0.21.

Table 2: Sine-Gordon results with the learned density. Energy is the maximum of $|E_t - E_0|/E_0$ over the rollout; L^2 is the relative error against the exact solution at the final time.

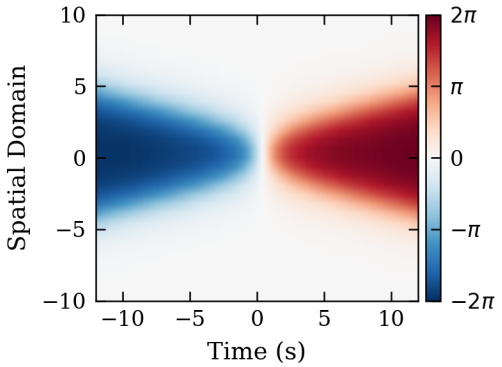
Case	**Nodes**	Δt	T	$\lvert\Delta E/E_0\rvert$	L^2
Breather	21	0.190	1,000 s	0.67%	0.97
Breather	31	0.129	1,000 s	0.12%	0.15
Breather	51	0.078	1,000 s	0.036%	0.077
Kink–antikink	31	0.129	24 s	0.61%	0.050



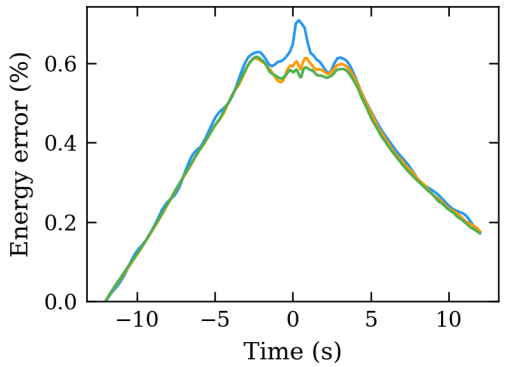
(a) Breather: relative error in energy $|E_t - E_0|/E_0$.



(b) Breather: relative L^2 error vs. the exact solution.



(c) Kink–antikink $q(x,t)$, $n = 31$, $v = 0.25$.



(d) Kink–antikink: relative error in energy.

Figure 8: Sine-Gordon with a learned Lagrangian density. (a, b) Breather over 1,000 s at three resolutions. (c, d) Kink–antikink collision, which occurs at $t = 0$; the kinks start and end at $x \approx \pm 4.3$.

Derivative nonlinearities. For more general classes of PDEs, the current formulation may accumulate drift more rapidly. The Born–Infeld equation, $\mathcal{L} = 1 - \sqrt{1 + q_x^2 - q_t^2}$, is quasilinear with strong nonlinearity in the derivatives, violating the assumptions made in Proposition 1. For a model trained on this equation, the error in energy reaches 2.2% by $t = 200$ s at $n = 51$ ($\Delta t = 0.01$, $n_r = 6$). No integrator, however, can be considered suitable for every problem. Where it applies, ELM retains the benefits demonstrated throughout the paper: no fixed discretization, easily implemented boundary conditions, and smaller data requirements.

G Extended 2D Wave Comparison

Figure 9 extends Section 4.3 with the relative L^2 error against the eigenmode ground truth over the full $1{,}000\,\mathrm{s}$ rollout. The qualitative result matches Figure 4b. ELM is the only method that stays below 10^0; PDE-Refiner saturates near 10^0, and FNO diverges at $t \approx 22\,\mathrm{s}$.

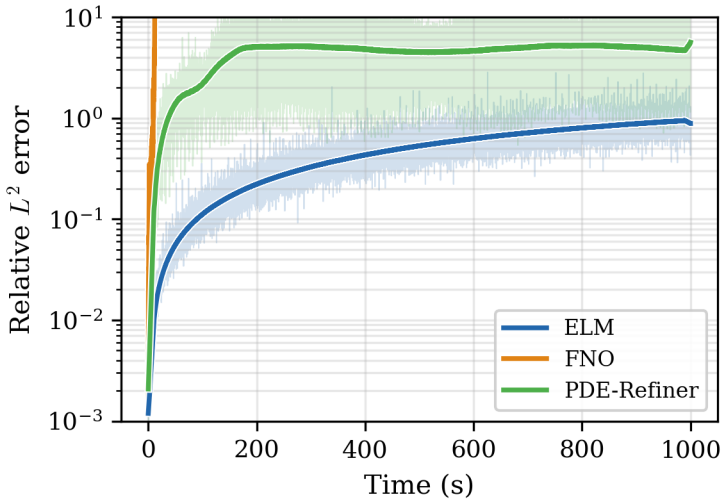


Figure 9: Relative L^2 error vs. eigenmode ground truth for the 2D wave over $1{,}000\,\mathrm{s}$ for the same system and methods as Figure 4.

H Boundary Condition Implementations

The experiments in Section 4 use five boundary conditions across the 1D and 2D wave equations. Table 3 lists the mathematical condition imposed at each boundary node and the section in which the boundary condition appears. In ELM, every boundary condition is realized by constraining a subset of the per-node variables before the Jacobi sweep; no boundary condition requires a change to the loss or to the integrator core.

We implement the absorbing boundary as a Mur boundary condition [Mur, 1981]. This boundary requires an estimate of the wave speed in the medium. For a homogeneous medium, the wave speed can be estimated from the learned model as

$$\hat{c}^2 = -\frac{\partial^2 \mathcal{L}}{\partial q_x^2} \Big/ \frac{\partial^2 \mathcal{L}}{\partial q_t^2}\bigg|_0 .$$

As the boundary condition is used only to prevent boundary reflections from distorting the qualitative interference pattern, this requirement is acceptable.

Table 3: Boundary conditions used in the experiments. For the periodic boundary, $q_\star$ indicates that all optimized variables are shared. For the Mur boundary, $s = \pm 1$ denotes the outward-normal sign ($+1$ for the right/top edges, -1 for left/bottom).

Name	**Condition**	**Used in**
Periodic	$q_\star(t, x_0) = q_\star(t, x_N)$	Section 4.2 (1D wave)
Dirichlet	$q = 0$	Section 4.3, slit barrier in Section 4.4
Neumann	$q_x = 0$	Section 4.4 (1D impedance edges)
Driven	$q = A\sin(\omega t)$	Section 4.4 (driven edge of double-slit)
Mur	$q_t + s\,c\,q_x = 0$	Section 4.4 (other three edges of double-slit)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Each claim in the abstract is expanded on in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: A summary of the symplecticity argument is provided in Section 3.4 with the full derivation of the generating function form in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full training and run information is provided in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All code necessary to reproduce the results, including data generation, is provided in the supplementary materials and is public at `https://github.com/lzhornyak/Euler-Lagrange-Minimization`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full training and run information is provided in Appendix D. Parameter information is provided Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 1 reports the energy error for the double pendulum and 1D wave as the mean $\pm$ standard deviation over ten unselected training seeds; the remaining results use a single trained model.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.2 describes the hardware needed and Appendix A describes the time cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All points in the code of Ethics are adhered to.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work describes a method to produce very long time-horizon predictions for certain PDE systems in a way that was not possible before. While this could have impacts on how physical systems are modelled, the societal impacts are so diffuse and downstream that it is not included in the paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: All models are of known systems and all data is generated.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All comparison models used in Section 4 are cited appropriately.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code used to generate the results are provided and documented in the supplementary materials.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects are used.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are used.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods of this paper were developed and implemented without the use of LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.