Neuromorphic
Computing and
Engineering

PURPOSE-LED
PUBLISHING™

PAPER • **OPEN ACCESS**

Generalizing equilibrium propagation to Lagrangian systems with arbitrary boundary conditions and equivalence with Hamiltonian echo learning

To cite this article: Guillaume Pourcel *et al* 2026 *Neuromorph. Comput. Eng.* **6** 044010

View the article online for updates and enhancements.

You may also like

- Storing sensor events in the interconnection strength of conducting polymer dendrites
Antoine Baron, Bilel Hafsi, Enrique H Balaguera et al.
- A low-power multichannel seizure detection system based on MoS_2 neuromorphic reservoir computing
Lorenzo Pittoni, Matteo Farronato, Arianna Capodiferro et al.
- Continuous attribution on discrete spikes: revealing the explainability paradox in spiking neural networks
M Oltan Sevinc, Liao Wu and Francisco Cruz

This content was downloaded from IP address 192.16.191.136 on 10/11/2026 at 09:10

NEUROMORPHIC
Computing and Engineering

Check for updates

OPEN ACCESS

RECEIVED
2 April 2026

REVISED
7 September 2026

ACCEPTED FOR PUBLICATION
25 September 2026

PUBLISHED
2 November 2026

Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Generalizing equilibrium propagation to Lagrangian systems with arbitrary boundary conditions and equivalence with Hamiltonian echo learning

Guillaume Pourcel[1,2,*], **Debabrota Basu**[2,7], **Maxence Ernoult**[5,6,7] and **Aditya Gilra**[3,4,7]

[1] University of Groningen, Bernoulli Institute & CogniGron, Groningen, The Netherlands
[2] Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 - CRIStAL, Lille, France
[3] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[4] Wirtschaftsuniversität, Vienna, Austria
[5] Independant researcher
[6] Current address: Google DeepMind, Paris, France.
[7] Equal senior authorship, listed in alphabetical order.
[*] Author to whom any correspondence should be addressed.

E-mail: g.a.pourcel@rug.nl, debabrota.basu@inria.fr, ernoult.m@gmail.com and aditya.gilra@wu.ac.at

Keywords: backpropagation algorithm, contrastive hebbian learning, variational principles, forward-only learning, energy-based models, neuromorphic computing, on-chip learning

Abstract

Equilibrium Propagation (EP) is a learning algorithm that applies to Energy-Based Models (EBMs) on static inputs. It estimates loss gradients by contrasting two steady states of the same EBM, rather than resorting to explicit adjoint dynamics. EP originally appealed as a plausible learning theory for biological substrates and has more recently attracted interest for its amenability to analog hardware. Extending EP to time-varying inputs and outputs is a challenging problem, as the variational description must apply to the entire system trajectory rather than just its steady state. While the use of the action of a Lagrangian system as an energy function appears as a natural choice—which we herein refer to as *Lagrangian Equilibrium Propagation* (LEP)—careful consideration of boundary conditions was largely overlooked in prior studies although it becomes essential. It is also unclear how applying LEP to Lagrangian systems theoretically relates to applying *Hamiltonian Echo Learning* (HEL) algorithms—i.e. Hamiltonian Echo Backpropagation and Recurrent Hamiltonian Echo Learning —to Hamiltonian systems. In this work, we thoroughly revisit LEP and demonstrate that different learning algorithms can be obtained depending on the boundary conditions of the system, many of which are impractical to simulate—e.g. with a prohibitive memory or computational cost, or requiring explicit Jacobian computation. We also show that HEL algorithms, which are much easier to simulate, can be explicitly cast as a special case of LEP where the initial conditions can be picked arbitrarily. Building upon this connection enables the extension of LEP to a broader class of systems with dissipation terms. By filtering out intractable instantiations of LEP and building an explicit mapping between HEL and LEP algorithms, this work facilitates the simulation of self-learning Lagrangian systems as well as extensions of LEP to broader classes of physical systems.

Contents

1. Introduction 3
2. Preliminaries and problem formulation 5
   2.1. The learning problem: supervised learning with time-varying input 5
   2.2. A primer on Lagrangian and Hamiltonian models 5
   2.3. Connecting Lagrangian and Hamiltonian formulations via the Legendre transform 7
3. EP: from static to time-varying input 8

© 2026 The Author(s). Published by IOP Publishing Ltd

3.1. EP: variational principle in *vector* space 8
3.2. *LEP*: variational principle in *functional* space 9
3.3. Instantiations of LEP 12
3.3.1. CBPVP on position 12
3.3.2. CIVP 14
3.3.3. Towards a practical implementation of LEP 15
4. RHEL 16
4.1. Hamiltonian system formulation 16
4.2. Two-phase learning procedure 16
4.3. Gradient computation 17
4.4. Contrast with variational approaches 17
5. RHEL is a particular case of the LEP 18
5.1. Instantiation of the LEP as a PFVP 18
5.1.1. Definition of the PFVP 18
5.1.2. Practical computation of the PFVP 18
5.2. Boundary residual cancellation in PFVP 19
5.3. Hamiltonian-Lagrangian equivalence via legendre transform 20
5.4. Empirical validation 21
5.4.1. Example of equivalence: fixed Hamiltonian initial conditions 21
5.4.2. Hopfield-inspired system with learnable time constants 22
5.4.3. Experimental setup 23
6. From LEP to dissipative LEP 24
6.1. Energy conservation in standard Lagrangian systems 24
6.2. Dissipative LEP 24
6.3. Empirical validation 25
7. Discussions and future works 28
Acknowledgments 29
Data availability statement 29
Appendix A. Derivative and shape conventions 30
Appendix B. Glossary 30
Appendix C. Preparatory results 31
Appendix D. Proof of proposition 1: Legendre transform 34
Appendix E. Proof of lemma 1. Euler–Lagrange solutions and the action functional 35
Appendix F. Proof theorem 1. Lagrangian EP gradient estimator 36
Appendix G. Proof of LEP Corollaries 37
G.1. Proof of corollary 2: gradient estimator for CIVP 37
G.2. Proof of corollary 1: Gradient estimator for CBPVP 38
Appendix H. Proof of theorem 2: Gradient estimator from RHEL with parametrized initial state 38
H.1. Notation correspondence 38
H.2. Gradient decomposition for parametrized initial states 39
H.3. Note on bidirectional vs unidirectional nudging 40
Appendix I. Proof of proposition 2: The bouncing-back trick 40
Appendix J. Proof theorem 3: PFVP cancels the boundary residuals 41
Appendix K. Proof theorem 4: Equivalence between Lagrangian EP and Recurrent Hamiltonian Echo Learning 42
K.1. Relating the solutions of Euler–Lagrange and Hamilton equations 42
K.2. Constructing the invertible mapping between PFVP and HES 45
K.2.1. Free-phase and forward phase 45
K.2.2. Nudged-phase and echo-phase 45
K.3. Gradient equivalence 47
Appendix L. Dissipative LEP 49
L.1. Proof of theorem 5: dissipative LEP 49
L.2. Proof of proposition 5: energy dissipation 50
Appendix M. Dissipative Harmonic Oscillators: Complete Derivation 51
M.1.Derivation of free and nudged dynamics 51
M.2.Time-reversibility and Parametric Final Value Problem (PFVP) implementation 51
M.3.Gradient estimator with fixed initial conditions 52
M.4.Energy evolution for Harmonic oscillators 53
M.5.Proof of proposition 6: time-reversal for dissipative systems 54

Appendix N. Electrical-Circuit Implementation of Dissipative LEP 55
N.1. A single oscillator is a parallel RLC node 56
N.2. A network: coupling springs become coupling inductors 57
N.3. Making a node dissipative—and flipping the sign of dissipation 57
N.4. The momentum flip: an H-bridge on each mass capacitor 58
N.5. The learning rule: local and measurable, as in analog EqProp 58
Appendix O. Computational Complexity Analysis of LEP Instantiations 59
O.1. Motivation 59
O.2. Setup and notation 59
O.3. CIVP (constant initial value problem) 60
O.4. CBPVP (constant boundary position value problem) 61
O.5. PFVP/RHEL (parametric final value problem) 62
O.6. Summary 63
Appendix P. Experimental Details 63
P.1. Hopfield-inspired system (figure 5) 63
P.2. Dissipative harmonic oscillators (figure 6) 63
Appendix Q. Learning Performance on the Teacher–Student Task 64
Appendix R. Generalization to Anisotropic Damping 65
R.1. Anisotropic exponential integrating-factor Lagrangian 65
R.2. Physical interpretation: time-varying coupling 65
R.3. Comparison with alternative approaches 66
Appendix S. Unconstrained action minimization 66
References 66

1. Introduction

The search for an alternative to backpropagation. Historically, feedforward networks alongside backpropagation have accidentally dominated the deep learning landscape over the last decade as the result of a 'hardware lottery' [Hoo20]: algorithms fitting the best available hardware win. Thanks to fine-grained CMOS-based compute primitives along with the development of hardware-agnostic compilation flows, digital hardware (e.g. CPUs, GPUs, TPUs [JYP+17]) provides the flexibility to execute any feedforward computational graph, including the exact implementation of backpropagation with the least amount of engineering. However this comes at the cost of digital overhead, complex memory hierarchies, and resulting data movement. In the short run, this motivates the search for 'IO–aware' algorithms [DFE+22] to mitigate High-Bandwidth Memory accesses, quantization algorithms to further reduce the memory cost and computational cost of *verbatim* backpropagation for on-device applications [LZC+22], and many other approaches going beyond the scope of this paper. Yet, none of these approaches truly leverage the underlying low-power transistor physics. Instead, transistor circuits remain abstracted away into implementing huge boolean functions in a stateless, unidirectional and deterministic fashion, which entails a significant energy consumption [ABB+25]. In the longer run, a radically different approach is the search for higher-level *analog* compute primitives, in particular, primitives for alternative learning and inference algorithms grounded in the analog physics of the underlying hardware [JNv23, LWWM24].

An important direction of research to achieve this goal is the development of learning algorithms that unify inference and learning within a *single* physical circuit [S+86, Spa92, FFS07, SB17, GG17, RKLH22, Hin22, LPM23, DBS+24]. This challenge, which we herein motivate for alternative hardware design, historically originated from neurosciences: biological neural networks face similar constraints, as 'non-local' algorithms such as backpropagation are widely considered biologically implausible for training neural networks [RHW86, LSM+20]. For instance, the strict implementation of backpropagation in biological systems would require a dedicated side network sharing parameters from the inference circuit to propagate error derivatives backward through the system, a problem coined weight transport [LCTA16, AWH+19]. The search for backpropagation alternatives therefore holds promise for both providing insights into how the brain might learn [RLB+19, PCG+23] and designing energy-efficient analog hardware [MRS+24].

Equilibrium Propagation (EP) and its limitations. *EP* [SB17] is a learning algorithm using a single circuit for inference and gradient computation and yielding an unbiased, variance-free gradient estimation—which is in stark contrast with alternative approaches relying on the use of noise injection [S+86, Spa92, FFS07, RKLH22]. A fundamental requirement of EP is that the models that are used

should be energy-based. Energy-based Models (EBMs) are models whose prediction is implicitly given as the minimum of some energy function. Therefore, EP falls under the umbrella of implicit learning algorithms such as implicit differentiation (ID) [BB08] which train implicit models [BKK19] to have steady states mapping static input–output pairs. EP is endowed with strong theoretical guarantees [SB19, EGQ+19] as it can be shown to be equivalent to a variant of ID called Recurrent Backpropagation [Alm89, Pin89]. While EP has been predominantly explored on Deep Hopfield Networks [Ros60, Hop82, SB17, EGQ+19, LES+21, LZ22, SEKK23, NE24], the application of EP to resistive networks [KPM+20, Sce24] has ushered in an exciting direction of research for learning algorithms amenable to analog hardware, with projected energy savings of four orders of magnitude [YKWK23]. Beyond the single-circuit property, EP naturally yields *local* learning rules whenever the energy is sum-separable [SEKK23], and can be made agnostic to the underlying physics [SMBO22]. Hopfield-inspired models further give rise to local Hebbian learning rules [SB17]. These properties resonate with neuroscience, where the same neural circuitry appears to be involved in both inference and learning [SMS+24, AdHLG24].

Yet, a major conundrum is to extend EP to *time-varying inputs and outputs*. One straightforward approach would be to consider well-crafted EBMs which adiabatically evolve with incoming inputs—i.e. at each time step, the system settles to equilibrium under the influence of the current input and of the steady state reached under the previous input. Such EBMs would formally fall under the umbrella of Feedforward-tied EBMs [NE24], which read as feedforward composition of EBM blocks and are reminiscent of fast associative memory models [BHM+16]. However, this approach is tied to a very specific class of models, would be costly to simulate (i.e. computing a steady state for each incoming input) and would be memory intensive (i.e. it would require storing the whole sequence of steady states and traversing them backwards for EP-based gradient estimation). A more general approach to extend EP to the temporal realm is to instead consider dissipation-free systems and pick their *action* as an energy function [Sce21, Ken21], which we herein refer to as *Lagrangian-based Equilibrium Propagation* (LEP). In the LEP setup, the variational object is no longer a steady state alone but *the whole physical trajectory*, characterized as a stationary point of the action under the imposed boundary conditions. Crucially, both [Sce21, Ken21] implicitly assumed boundary-value-problem conditions—i.e. vanishing variations at both endpoints—yet neither study provided a practical algorithm nor implementation, raising the question of how feasible this assumption actually is. More broadly, existing LEP proposals remain theoretical and did not lead to any practical algorithmic prescriptions, which we diagnose as due to the need to *carefully handle boundary conditions* arising in the underlying variational problem. This limitation raises our first key question:

> *Can EP be generalised to design efficient and practically-implementable learning algorithms for time-varying inputs and outputs?*

Hamiltonian-based approaches. In parallel to EP research, learning algorithms grounded in *reversible* Hamiltonian dynamics have emerged as another promising direction of research. One such algorithm, Hamiltonian Echo Backpropagation (HEB, [LPM23]), was developed with theoretical physics tools to train the initial conditions of physical systems governed by field equations for static input-output mappings. More recently, Recurrent Hamiltonian Echo Learning (RHEL) was introduced as a generalization of HEB to time-varying inputs and outputs [PE25]. Like EP, these Hamiltonian-based approaches, which we herein label as *Hamiltonian Echo Learning (HEL)* algorithms, enable a single physical system to perform both inference and learning whilst maintaining the theoretical equivalence to backpropagation through time (BPTT). Interestingly, HEL methods were also independently found to yield local Hebbian learning rules [DP25], and to lend themselves to be agnostic to the underlying physics [PE25]. Since HEL algorithms originate from a different formalism than that of LEP, this motivates our second key question:

> *How do HEL algorithms relate to LEP?*

In this paper, we address both questions through a theoretical analysis that reveals the connection between these approaches. Our contributions are organized as follows:

- We revisit *LEP*, which extends the variational formulation of EP to temporal trajectories (section 3.2). Our formulation generalizes previous studies [Sce21, Ken21] by carefully analyzing the effect of different boundary conditions, explicitly treating both the boundary-value assumption of prior work (CBPVP, section 3.3.1) and the initial-value alternative (CIVP, section 3.3.2).

- We show that the boundary-value formulation (CBPVP) assumed by prior work eliminates boundary residuals from the learning rule but requires an expensive non-causal iterative solver whose cost dominates the overall complexity (section 3.3). We then show that the natural causal alternative (CIVP), which restores forward Euler–Lagrange integration, introduces intractable boundary residual terms. Neither formulation leads to a practical algorithm on its own.
- We demonstrate that *RHEL can be derived as a special case of LEP* by constructing an associated reversible Lagrangian system with carefully chosen boundary conditions (PFVP) that eliminate the problematic residual terms while preserving causal forward integration—yielding a *first practical implementation of LEP*. Further, we establish the mathematical equivalence of RHEL and LEP through the Legendre transformation (section 5). We empirically validate this equivalence with a numerical analysis comparing the gradient estimates obtained by LEP and RHEL (section 5.4).
- Finally, we directly leverage the connection between RHEL and LEP to come up with a variant of LEP that applies to dissipative Lagrangians which we call *Dissipative LEP* (section 6.2). In the formulation studied here, switching from the free phase to the echo phase requires reversing the damping coefficient from $+\gamma$ to $-\gamma$ while preserving its magnitude, thereby turning energy loss into matched energy injection. Under this condition, we empirically show that gradients can be correctly estimated.

As a reading guide, table 2 (end of section 3.2) summarizes upfront the computational trade-offs between the three boundary-condition formulations studied in this work (CIVP, CBPVP, PFVP) and announces the destination of the theoretical development: only PFVP yields a causal, forward-only, streaming learning algorithm, at the cost of requiring time-reversible dynamics.

2. Preliminaries and problem formulation

2.1. The learning problem: supervised learning with time-varying input

We consider the supervised learning problem, where the goal is to predict a target trajectory $\boldsymbol{y}(t) \in \mathbb{R}^{d_y}$ given an input trajectory $\boldsymbol{x}(t) \in \mathbb{R}^{d_x}$ over a continuous time interval $t \in [0, T]$. The model is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ and produces predictions through a continuous state trajectory $\boldsymbol{s}_t(\boldsymbol{\theta}) \in \mathbb{R}^{d_s}$ that evolves over time according to the system dynamics. In the context of continuous time systems, the state-trajectory is typically defined as the solution of an Ordinary Differential equation (ODE).

The learning objective is to minimize a cost functional $C[\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}), \boldsymbol{y}]$ that measures the discrepancy between the produced trajectory and the target. Formally,

$$C[\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}), \boldsymbol{y}] := \int_0^T c(\boldsymbol{s}_t(\boldsymbol{\theta}, \boldsymbol{x}_t), \boldsymbol{y}_t)\, \mathrm{d}t,$$

where $c(\cdot, \cdot) : \mathbb{R}^{d_s} \times \mathbb{R}^{d_y} \to \mathbb{R}$ is an instantaneous cost function that evaluates the prediction error at time t and $\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}) := \{\boldsymbol{s}_t(\boldsymbol{\theta}, \boldsymbol{x}_t) : t \in [0, T]\}$ represents the entire trajectory. Commonly, c takes the form of an ℓ_2 loss function, $c(\boldsymbol{s}_t, \boldsymbol{y}_t) := \frac{1}{2}\|\boldsymbol{s}_t^{\text{out}} - \boldsymbol{y}_t\|_2^2$, where $\boldsymbol{s}_t^{\text{out}} \in \mathbb{R}^{d_y}$ denotes an appropriately selected subset of state variables. More generally, c can be any differentiable function that quantifies the instantaneous prediction error.

The parameters $\boldsymbol{\theta}$ are optimized to minimize the cost functional $C[\boldsymbol{s}(\boldsymbol{\theta}, \boldsymbol{x}), \boldsymbol{y}]$. One popular approach to solve this minimization problem is to use gradient descent-type optimization algorithms. Modern machine learning owes much of its success to the generality and scalability of gradient-based optimization. This requires computing the gradient of the learning objective with respect to the parameters $\boldsymbol{\theta}$. While several methods have been proposed to compute this gradient, most rely on explicit backward passes through computational graphs [RHW86, LBH15], making them unsuitable for analog hardware implementations or plausible explanations for biological learning.

This limitation has motivated the development of alternative learning paradigms. Among the existing approaches, the *EP* [SB17] framework stands out as a particularly promising one for designing a single system that can perform inference and learning.

2.2. A primer on Lagrangian and Hamiltonian models

In this paper, the learning algorithms considered are constraining the kind of trajectories that can be used. In particular, we will only consider state trajectories $\boldsymbol{s}_t(\boldsymbol{\theta})$ that arise from Lagrangian and Hamiltonian dynamics. Both Hamiltonian and Lagrangian dynamics provide frameworks for formulating specific dynamical systems using a scalar-valued function: the Lagrangian or the Hamiltonian, defined as follows:

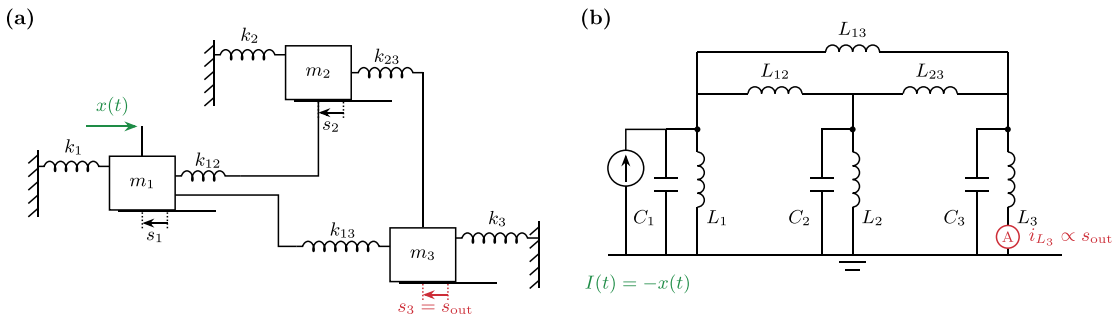


Figure 1. Driven conservative coupled-oscillator network and its electrical twin. (a) Three horizontally sliding masses with grounding springs k_i, pairwise coupling springs k_{ij}, external force $x(t)$ on m_1, and output displacement $s_{\text{out}} = s_3$. (b) The equivalent lossless circuit: position s_i corresponds to node flux $\phi_i = \int V_i \, \mathrm{d}t$, velocity $\dot{s}_i$ to node voltage V_i, mass m_i to capacitance C_i, grounding spring k_i to inductor $L_i = 1/k_i$, and coupling spring k_{ij} to inductor $L_{ij} = 1/k_{ij}$ between nodes i and j. With the upward source direction shown, $I(t) = -x(t)$ matches the manuscript's drive convention, and $\mathrm{i}_{L_3} \propto s_3$ provides the output. Both panels realize the dynamics $M\ddot{\boldsymbol{s}} + K\boldsymbol{s} = -x(t)e_1$, equivalently described by the Hamiltonian or Lagrangian given above.

- The Lagrangian $\mathcal{L}(\boldsymbol{s}, \dot{\boldsymbol{s}}, \boldsymbol{x}, \boldsymbol{\theta}) : \mathbb{R}^{d_s} \times \mathbb{R}^{d_s} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}$ is a function of the state $\boldsymbol{s}$, its time derivative $\dot{\boldsymbol{s}}$ (velocity), the input $\boldsymbol{x}$, and parameters $\boldsymbol{\theta}$. The dynamics are then defined by the Euler–Lagrange equations:

$$d_t \partial_{\dot{\boldsymbol{s}}} \mathcal{L} - \partial_{\boldsymbol{s}} \mathcal{L} = 0.$$

- The Hamiltonian $H(\boldsymbol{s}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\theta}) : \mathbb{R}^{d_s} \times \mathbb{R}^{d_s} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}$ is a function of the position $\boldsymbol{s}$, momentum $\boldsymbol{p}$, the input $\boldsymbol{x}$, and parameters $\boldsymbol{\theta}$. The dynamics are defined by Hamilton's equations:

$$\begin{pmatrix} d_t \boldsymbol{s} \\ d_t \boldsymbol{p} \end{pmatrix} = \boldsymbol{J} \begin{pmatrix} \partial_{\boldsymbol{s}} H \\ \partial_{\boldsymbol{p}} H \end{pmatrix},$$

  where $\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$ is the canonical symplectic matrix.

Toy example: Driven coupled harmonic oscillators (3 masses). A simple physical system that can be expressed in both Lagrangian and Hamiltonian form is a set of three coupled harmonic oscillators, depicted in figure 1. Let $\boldsymbol{s} = (s_1, s_2, s_3)^\top$ be the displacements and $\boldsymbol{p} = (p_1, p_2, p_3)^\top$ the momenta, with mass vector $\boldsymbol{m} = (m_1, m_2, m_3)^\top$ where $m_i > 0$, per-mass spring constants $k_i \geqslant 0$, and pairwise spring couplings $k_{ij} = k_{ji} \geqslant 0$. An external input $x(t)$ acts on the first mass (the output is $y(t) = s_3(t)$). The learnable parameters are $\boldsymbol{\theta} = (m_1, m_2, m_3, k_1, k_2, k_3, k_{12}, k_{13}, k_{23})^\top$.

The system is described by the Hamiltonian

$$H(\boldsymbol{s}, \boldsymbol{p}, x, \boldsymbol{\theta}) = \frac{1}{2} \left(\boldsymbol{m}^{-1} \odot \boldsymbol{p} \right)^\top \boldsymbol{p} + \frac{1}{2} \sum_{i=1}^{3} k_i s_i^2 + \frac{1}{2} \sum_{i<j} k_{ij} \left(s_j - s_i \right)^2 + s_1 x,$$

and equivalently by the Lagrangian

$$\mathcal{L}(\boldsymbol{s}, \dot{\boldsymbol{s}}, x, \boldsymbol{\theta}) = \frac{1}{2} \left(\boldsymbol{m} \odot \dot{\boldsymbol{s}} \right)^\top \dot{\boldsymbol{s}} - \frac{1}{2} \sum_{i=1}^{3} k_i s_i^2 - \frac{1}{2} \sum_{i<j} k_{ij} \left(s_j - s_i \right)^2 - s_1 x.$$

Both formulations lead to the same second-order dynamics:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}} + K\boldsymbol{s} = -x e_1,$$

where $\odot$ denotes element-wise multiplication (Hadamard product), the stiffness matrix K has $K_{ii} = k_i + \sum_{j \neq i} k_{ij}$ and $K_{ij} = -k_{ij}$ for $i \neq j$, and $e_1 = (1, 0, 0)^\top$ is the first canonical basis vector selecting the first mass (the driven coordinate).

Figure 1(b) is a direct lossless electrical realization of the same dynamics, not a separate model. At node i, define the node flux $\phi_i(t) = \int_0^t V_i(\tau) \, \mathrm{d}\tau$, so that the node voltage $V_i = \dot{\phi}_i$ corresponds to the mass velocity $\dot{s}_i$. Kirchhoff's current law gives

$$C_i \ddot{\phi}_i + \frac{\phi_i}{L_i} + \sum_{j \neq i} \frac{\phi_i - \phi_j}{L_{ij}} = \delta_{i1} I(t).$$

Table 1. Machine learning models with Hamiltonian and Lagrangian formulations.

Model	Hamiltonian (H)	Lagrangian (L)	Dynamics	Constraint
UniCORNN [RM21]	$\frac{1}{2}\|\boldsymbol{p}\|^2 + \frac{\alpha}{2}\|\boldsymbol{s}\|^2 + \mathbf{1}^\top \boldsymbol{W}^{-1}$ $\log(\cosh(\boldsymbol{W}\boldsymbol{s}+\boldsymbol{B}\boldsymbol{x}+\boldsymbol{b}))$	$\frac{1}{2}\|\dot{\boldsymbol{s}}\|^2 - \frac{\alpha}{2}\|\boldsymbol{s}\|^2 - \mathbf{1}^\top \boldsymbol{W}^{-1}$ $\log(\cosh(\boldsymbol{W}\boldsymbol{s}+\boldsymbol{B}\boldsymbol{x}+\boldsymbol{b}))$	$\ddot{\boldsymbol{s}} = \tanh(\boldsymbol{W}\boldsymbol{s}+\boldsymbol{B}\boldsymbol{x}+\boldsymbol{b}) - \alpha\boldsymbol{s}$	$\boldsymbol{W}$ diagonal
LinOSS [RR25]	$\frac{1}{2}\|\boldsymbol{p}\|^2 + \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{W}\boldsymbol{s} - \boldsymbol{s}^\top \boldsymbol{B}\boldsymbol{x}$	$\frac{1}{2}\|\dot{\boldsymbol{s}}\|^2 - \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^\top \boldsymbol{B}\boldsymbol{x}$	$\ddot{\boldsymbol{s}} = -\boldsymbol{W}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{x}$	$\boldsymbol{W}$ symmetric
Hopfield [DP25]	$\frac{1}{2}\boldsymbol{p}^\top \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{p} + \boldsymbol{b}^\top \rho(\boldsymbol{s})$ $+\frac{1}{2}\rho(\boldsymbol{s})^\top \boldsymbol{W}\rho(\boldsymbol{s}) + \rho(\boldsymbol{s})^\top \boldsymbol{B}\rho(\boldsymbol{x})$	$\frac{1}{2}\dot{\boldsymbol{s}}^\top \mathrm{diag}(\boldsymbol{\tau})\dot{\boldsymbol{s}} - \boldsymbol{b}^\top \rho(\boldsymbol{s})$ $-\frac{1}{2}\rho(\boldsymbol{s})^\top \boldsymbol{W}\rho(\boldsymbol{s}) - \rho(\boldsymbol{s})^\top \boldsymbol{B}\rho(\boldsymbol{x})$	$\mathrm{diag}(\boldsymbol{\tau})\ddot{\boldsymbol{s}} = -\rho'(\boldsymbol{s}) \odot$ $(\boldsymbol{W}\rho(\boldsymbol{s}) + \boldsymbol{b} + \boldsymbol{B}\rho(\boldsymbol{x}))$	$\boldsymbol{W}$ symmetric

Therefore, the identifications $C_i = m_i$, $L_i = 1/k_i$, $L_{ij} = 1/k_{ij}$, $\phi_i = s_i$, and $I(t) = -x(t)$ reproduce the mechanical equation above term by term. The output position can likewise be read from the grounding-inductor current $\mathrm{i}_{L_3} = \phi_3/L_3 \propto s_3$. Appendix N develops this circuit correspondence in detail.

Machine learning examples. Lagrangian and Hamiltonian formulations are widely used in physics, and correspond to a broad class of physical systems. Recently, they have been applied to machine learning and neuroscience. In machine learning, they have been used to design RNNs with desirable vanishing or exploding gradient properties (UniCORNN, [RM21]), and to design efficient modern state space model architectures (LinOSS, [RR25])—see table 1 for their Lagrangian and Hamiltonian formulations and dynamics.

More generally, this research aligns with the renewed interest in RNNs as computationally efficient alternatives to Transformers, where state-based dynamical systems eliminate the quadratic cost of attention while maintaining comparable performance on long-range sequence tasks [OSG+23]. In neuroscience, it was proposed that RHEL could be implemented in a biologically plausible way via a Hamiltonian inspired by Hopfield energy functions [DP25].

2.3. Connecting Lagrangian and Hamiltonian formulations via the Legendre transform

Hamiltonian and Lagrangian formalisms offer complementary perspectives on the same underlying dynamics. Each formalism possesses distinct mathematical structure that favors different proof techniques: the Hamiltonian framework, with its symplectic geometry and phase-space representation, naturally accommodates tools such as Green's functions [LPM23] and adjoint methods [PE25]. These techniques proved instrumental in deriving HEL. Conversely, the Lagrangian framework foregrounds the variational structure of trajectories, which makes it particularly amenable to EP.

The Legendre transform provides a bridge between these two representations and allows the results established in one formalism to be translated into the other.

Proposition 1 (Legendre transform). *Let $(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) \in \mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$ and $(\boldsymbol{s}_t, \boldsymbol{p}_t) \in \mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$ denote tuples of position–velocity and position–momentum variables, respectively. The Legendre transform establishes a pointwise-in-time, locally invertible mapping between the Lagrangian and Hamiltonian representations, with $L, H \in C^2$:*
(a) Forward transform ($L \to H$).

$$\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t), \qquad H(\boldsymbol{s}_t, \boldsymbol{p}_t) = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t),$$

which is well-defined whenever $\det(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L) \neq 0$.
(b) Backward transform ($H \to L$).

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t), \qquad L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - H(\boldsymbol{s}_t, \boldsymbol{p}_t),$$

which is well-defined whenever $\det(\partial^2_{\boldsymbol{p},\boldsymbol{p}} H) \neq 0$.
Since the Hessians satisfy $\partial^2_{\boldsymbol{p},\boldsymbol{p}} H = (\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L)^{-1}$, the well-definiteness conditions are equivalent.

Since $L \in C^2$ and $\det(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L) \neq 0$, the Euler–Lagrange equation can be rewritten as a first-order ODE whose right-hand side is locally Lipschitz, which guarantees uniqueness of solutions to initial value problems. We invoke this uniqueness property without further comment in the sequel. The Hessian condition is satisfied for all models of table 1: $\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L$ is the identity for UniCORNN and LinOSS, and $\mathrm{diag}(\boldsymbol{\tau})$ for the Hopfield-inspired model, which is invertible provided that $\tau_i > 0$ for every i. Since the cost c is velocity-independent, this condition is unchanged by nudging, with $\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L_\beta = \partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L_0$ for every β.

3. EP: from static to time-varying input

In this paper, we refer to the EP framework as a general recipe to design learning algorithms, where the model to be trained admits a variational description [Sce21]. The core mechanics underpinning EP are fundamentally *contrastive*: EP proceeds by solving two related variational problems:

- the *free* problem, which defines model inference, i.e. the 'forward pass' of the model to be trained,
- the *nudged* problem, which is a perturbation of the free problem with infinitesimally lower prediction error controlled by some nudging parameter.

Therefore, EP mechanics perform two relaxations to equilibrium, e.g. two 'forward passes', to estimate gradients without requiring explicit backward passes through the computational graph.

3.1. EP: variational principle in *vector* space

In the original formulation of EP, the nudged problem is defined via an augmented energy function that linearly combines an energy function with the learning cost function:

$$E_\beta\left(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0,\boldsymbol{y}_0\right) := E\left(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0\right) + \beta C\left(\boldsymbol{s},\boldsymbol{y}_0\right).$$

Here, $E(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0)$ is the energy function, i.e. a scalar-valued function that takes as input a state vector $\boldsymbol{s}\in\mathbb{R}^{d_s}$, a learnable parameter vector $\boldsymbol{\theta}$, and a **static** input $\boldsymbol{x}_0\in\mathbb{R}^{d_x}$. The cost function $C(\boldsymbol{s},\boldsymbol{y}_0)$ in this setup takes as input a **static** output target $\boldsymbol{y}_0\in\mathbb{R}^{d_y}$ and the static state vector. The nudging parameter $\beta\in\mathbb{R}$ controls the influence of the cost on the augmented energy. This augmented energy defines a vector-valued implicit function $(\boldsymbol{\theta},\beta)\mapsto\boldsymbol{s}^\beta(\boldsymbol{\theta})$[6] through the nudged variational problem. Specifically, it is minimized at

$$\partial_{\boldsymbol{s}} E_\beta\left(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0,\boldsymbol{y}_0\right) = \boldsymbol{0}.$$

The model used for the machine learning task is the implicit function $\boldsymbol{\theta}\mapsto\boldsymbol{s}^0(\boldsymbol{\theta})$ defined by the free variational problem $\partial_{\boldsymbol{s}}E(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0)=\boldsymbol{0}$, and the learning objective is to minimize the cost $C(\boldsymbol{s}^0(\boldsymbol{\theta}),\boldsymbol{y}_0)$ by finding the gradient $\mathrm{d}_{\boldsymbol{\theta}}C(\boldsymbol{s}^0(\boldsymbol{\theta}),\boldsymbol{y}_0)$. The fundamental result of EP states that this gradient can be computed using [Sce21]

$$\mathrm{d}_{\boldsymbol{\theta}}C\left(\boldsymbol{s}^0\left(\boldsymbol{\theta}\right),\boldsymbol{y}_0\right) = \lim_{\beta\to 0}\frac{1}{\beta}\left[\partial_{\boldsymbol{\theta}}E_\beta\left(\boldsymbol{s}^\beta\left(\boldsymbol{\theta}\right),\boldsymbol{\theta},\boldsymbol{x}_0\right) - \partial_{\boldsymbol{\theta}}E_0\left(\boldsymbol{s}^0\left(\boldsymbol{\theta}\right),\boldsymbol{\theta},\boldsymbol{x}_0\right)\right]. \tag{1}$$

This suggests a two-phase procedure for gradient estimation via a finite difference method, illustrated in figure 2(a):

1. **Free phase:** Compute the output value of the implicit function $\boldsymbol{s}^0(\boldsymbol{\theta})$ (black cross $\times$) by finding a minimum of the energy function $E(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0)$ (black curve).
2. **Nudged phase:** Compute the output value of the implicit function $\boldsymbol{s}^\beta(\boldsymbol{\theta})$ (blue dot) for a small positive value of β by finding a slightly perturbed minimum of the augmented energy $E_\beta(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}_0,\boldsymbol{y}_0)$ (blue curve).

Note that multiple nudged phases with opposite nudging strength ($\pm\beta$) may be needed to reduce the bias of EP-based gradient estimation [LES+21]. In practice, these implicit function values can be found with a properly chosen root finding algorithm. As done in many past works [SB17, MZK+22], we pick gradient descent dynamics over the energy function as an example here. Simple fixed-point iteration [LES+21, LZ22, SEKK23] or coordinate descent [Sce24] may also be used depending on the models in use. In the free phase ($\beta=0$), the system evolves according to (figure 2(b), black curve):

$$d_t\boldsymbol{s}_t = -\partial_{\boldsymbol{s}}E\left(\boldsymbol{s}_t,\boldsymbol{\theta},\boldsymbol{x}_0\right), \tag{2}$$

[6] For notational simplicity, we omit the explicit dependence of the implicit function $(\boldsymbol{\theta},\beta)\mapsto\boldsymbol{s}^\beta(\boldsymbol{\theta})$ on $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$, as we consider the gradient computation for a fixed input-target pair.

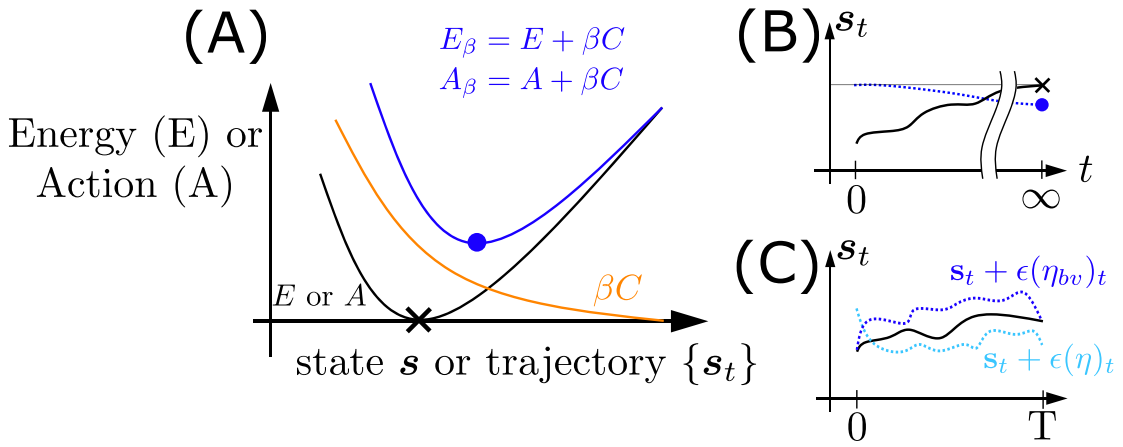


Figure 2. (a) EP trains variational systems. EP can train models that admit a variational description, whether as a state vector $\boldsymbol{s}$ or entire state trajectories (function of time) $\{\boldsymbol{s}_t\}$ that extremize a scalar function E (or functional A), represented by the black cross. To train these models to minimize a loss C (orange curve), one computes an extremum of the augmented energy E_β (or action A_β) represented by the blue curve. These two variational objects enable efficient gradient computation, leading to an improved energy (action) after a gradient update (dotted line). **Important caveat:** for the functional version, the trajectory is a stationary point (not necessarily a minimum) of the action; it is a true minimum only when considering *boundary-vanishing variations* (see Panel C). For general variations, additional boundary terms appear in the gradient computation—adapted from [Ern20]. (b) EP. The free phase (black curve) and nudged phase (dotted blue curve) consist of gradient descent on the energy and augmented energy, respectively. These phases run sequentially, the nudged phase starts from the end-state of the free phase, and the learning rule uses only the states at convergence. The trained model corresponds to the free equilibrium state (black cross). (c) LEP. The free phase (black curve) corresponds to a trajectory satisfying the Euler–Lagrange equations, while the nudged phase follows the Euler–Lagrange equation associated with the augmented action (dotted curve). For boundary-vanishing variation $\boldsymbol{s}_t + \epsilon(\boldsymbol{\eta}_{bv})_t$ (dotted blue curve), the gradient estimator is easy to compute because the boundary residual vanishes, but the boundary conditions are non-causal (both endpoints are constrained). On the contrary, for causal boundary conditions $\boldsymbol{s}_t + \epsilon\boldsymbol{\eta}_t$ (dotted cyan curve), trajectories can be efficiently computed via forward integration, but the gradient estimator has boundary residuals that can be hard to compute.

until convergence to $\boldsymbol{s}^0(\boldsymbol{\theta})$, i.e. $\lim_{t\to\infty} \boldsymbol{s}_t = \boldsymbol{s}^0(\boldsymbol{\theta})$. This temporal evolution is shown as the black curve in figure 2(b). In the nudged phase ($\beta > 0$), starting from the free equilibrium, the system follows (figure 2(b), blue dotted curve):

$$d_t \boldsymbol{s}_t = -\partial_{\boldsymbol{s}} E(\boldsymbol{s}_t, \boldsymbol{\theta}, \boldsymbol{x}_0) - \beta \partial_{\boldsymbol{s}} C(\boldsymbol{s}_t, \boldsymbol{y}_0)\,, \tag{3}$$

until convergence to $\boldsymbol{s}^\beta(\boldsymbol{\theta})$, i.e. $\lim_{t\to\infty} \boldsymbol{s}_t = \boldsymbol{s}^\beta(\boldsymbol{\theta})$. The corresponding dynamical trajectory is depicted as the blue dotted curve in figure 2(b). Importantly, the gradient descent dynamics in equations (2) and (3) are *neither physical*[7], nor explicitly trained to match a target trajectory. As mentioned earlier, they serve as a computational tool to reach the solution of the variational problem. Because of these dynamics, the solutions of these variational problems are often called 'equilibrium states' or 'fixed points'. The model corresponds to the free equilibrium, while the contrast between the nudged and free equilibria provides the necessary information to compute gradients through equation (1).

Limitations. The fact that we are only training the fixed point of the system highlights a major limitation of EP. It can only be used to train static input-output mappings (from $\boldsymbol{x}_0$ to $\boldsymbol{y}_0$). More precisely, the equilibrium state defined by equation (2) represents a *time-independent* configuration that encodes an implicit function $\boldsymbol{\theta} \mapsto \boldsymbol{s}^0(\boldsymbol{\theta})$ with static vector input $\boldsymbol{x}_0$ and *static* vector output $\boldsymbol{y}_0$. This fundamental constraint arises because energy function $E(\boldsymbol{s}, \boldsymbol{\theta}, \boldsymbol{x}_0)$ is applied only to instantaneous states rather than temporal trajectories.

A challenge lies in extending the variational principle underlying the framework of EP from vector spaces (where a single state $\boldsymbol{s}$ is described variationally) to *functional spaces*, where entire trajectories $\{\boldsymbol{s}_t : t \in [0, T]\}$ are described by a variational principle. Such an extension requires moving from energy functions defined on state vectors to an energy-like quantity defined on complete trajectories.

3.2. *LEP*: variational principle in *functional* space

Now, we generalise EP to describe entire trajectories through a variational problem, enabling us to train dynamical systems that map time-varying inputs to time-varying outputs. We refer to this extension as

[7] i.e. The physical system does not need to implement gradient-descent dynamics explicitly; it only has to find a minimum of the energy landscape.

LEP . To achieve this extension, we revisit the concept of augmented energy E_β to an *augmented action functional* A_β that integrates over a time-varying 'energy-like' quantity called the *Lagrangian* L_0 [Sce21]:

$$\underbrace{A_\beta\left[\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x},\boldsymbol{y}\right]}_{\text{augmented action}} := \int_0^T \underbrace{\left(L_0\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t\right)+\beta\; c\left(\boldsymbol{s}_t,\boldsymbol{y}_t\right)\right)}_{L_\beta\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t,\boldsymbol{y}_t\right)}\mathrm{d}t$$
$$= \underbrace{\int_0^T L_0\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t\right)\mathrm{d}t}_{:=\text{action } A[\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}]} + \beta\underbrace{\int_0^T c\left(\boldsymbol{s}_t,\boldsymbol{y}_t\right)\mathrm{d}t}_{:=\text{cost } C[\boldsymbol{s},\boldsymbol{y}]}. \tag{4}$$

Here $A[\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{x}]$ is a functional that serves as the temporal counterpart of the energy function E, operating on entire trajectories[8]. It integrates the Lagrangian $L_0(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t)$ over time, where the Lagrangian takes as input the state $\boldsymbol{s}_t$, its temporal derivative (velocity) $\dot{\boldsymbol{s}}_t$, and the time-varying input $\boldsymbol{x}_t$.

The augmented action functional A_β is the temporal analogue of E_β. It integrates the augmented Lagrangian L_β that extends the Lagrangian by including an additional nudging term $\beta c(\boldsymbol{s}_t,\boldsymbol{y}_t)$. The augmented action functional $A_\beta[\boldsymbol{s}]$ maps a trajectory $\boldsymbol{s} := \{\boldsymbol{s}_t : t\in[0,T]\}$ to a scalar value, generalizing the scalar-valued energy functions of EP to functional-valued quantities that capture temporal dynamics. For notational simplicity, we omit the dependence on inputs $\boldsymbol{x}$ and targets $\boldsymbol{y}$ (or their time-indexed versions $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$) whenever the context is clear.

Variational formulation and functional derivatives. The action functional enables us to define the variational problems that generalize EP to the temporal domain. Following standard variational calculus [Olv22], we define the *functional derivative* (or *variational derivative*) $\delta_{\boldsymbol{s}}A_\beta$ through the directional derivative with respect to trajectory variations $\boldsymbol{\eta} := \{\boldsymbol{\eta}_t : t\in[0,T]\}$:

$$\delta_{\boldsymbol{s}}A_\beta\cdot\boldsymbol{\eta} := d_\epsilon|_{\epsilon=0}A_\beta\left[\boldsymbol{s}+\epsilon\boldsymbol{\eta}\right],$$

where $\delta_{\boldsymbol{s}}A_\beta$ denotes the functional gradient with respect to the trajectory and $\cdot$ denotes the standard L^2 inner product on function space, i.e. $\boldsymbol{\eta}\cdot\boldsymbol{\eta}' := \int_0^T(\boldsymbol{\eta}_t)^\top(\boldsymbol{\eta}_t')\,\mathrm{d}t$. With this notation, the *nudged variational problem* is

$$\delta_{\boldsymbol{s}}A_\beta = 0 \quad\Leftrightarrow\quad \delta_{\boldsymbol{s}}A_\beta\cdot\boldsymbol{\eta} = 0 \text{ for all smooth variations } \boldsymbol{\eta} \quad \text{s.t.} \quad \boldsymbol{\eta}_0 = \boldsymbol{\eta}_T = 0.$$

In particular, for $\beta = 0$, the *free variational problem* is defined as $\delta_{\boldsymbol{s}}A_0 = 0$, corresponding to the system's natural dynamics without nudging. Unlike EP, where the variational problems are typically solved through gradient descent dynamics, these functional variational problems can be solved more directly using the Euler–Lagrange equations. The corresponding Euler–Lagrange expression is defined as

$$\mathrm{EL}\left(t,\boldsymbol{\theta},\beta\right) := \partial_{\boldsymbol{s}}L_\beta\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta}\right) - d_t\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta}\right)$$
$$= \partial_{\boldsymbol{s}}L_0\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta}\right) - d_t\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta}\right) + \beta\partial_{\boldsymbol{s}}c\left(\boldsymbol{s}_t\right).$$

The following classic result, namely the principle of stationary action [Olv22], generalized to arbitrary boundary conditions, establishes the fundamental connection between the variational formulation and the Euler–Lagrange equation.

Lemma 1 (Euler–Lagrange solutions and the action functional [Olv22]). *Let* $\mathbf{s}^\beta(\boldsymbol{\theta}) := \{\boldsymbol{s}_t^\beta(\boldsymbol{\theta}) : t\in[0,T]\}$ *be a trajectory solution of the Euler–Lagrange equation* $\mathrm{EL}(t,\boldsymbol{\theta},\beta) = 0$ *for all* $t\in[0,T]$*, and let* $\boldsymbol{\eta} := \{\boldsymbol{\eta}_t : t\in[0,T]\}$ *be any smooth variation. Then (see proof in appendix E):*

1. ***Boundary-vanishing variations:*** *For any variation* $\boldsymbol{\eta}_{bv}$ *that vanishes at the boundaries, i.e.* $(\boldsymbol{\eta}_{bv})_0 = (\boldsymbol{\eta}_{bv})_T = \mathbf{0}$*,* $\mathbf{s}^\beta(\boldsymbol{\theta})$ *is a critical point of the action functional* $A_\beta[\mathbf{s}]$*:*

$$\delta_{\boldsymbol{s}}A_\beta\cdot\boldsymbol{\eta}_{bv} = d_\epsilon|_{\epsilon=0}A_\beta\left[\mathbf{s}^\beta+\epsilon\boldsymbol{\eta}_{bv}\right] = 0.$$

2. ***General formula for arbitrary variations:*** *For an arbitrary variation* $\boldsymbol{\eta}$ *(not necessarily vanishing at the boundaries), the directional derivative of the action is given by:*

$$\delta_{\boldsymbol{s}}A_\beta\cdot\boldsymbol{\eta} = d_\epsilon|_{\epsilon=0}A_\beta\left[\mathbf{s}^\beta+\epsilon\boldsymbol{\eta}\right] = \left[\boldsymbol{\eta}_t^\top\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_t^\beta,\dot{\boldsymbol{s}}_t^\beta,\boldsymbol{\theta}\right)\right]_0^T. \tag{5}$$

[8] Note that we do not have to write $\dot{\boldsymbol{s}}$ as input of the action A, because it can be derived from $\boldsymbol{s}$ via the time derivative transformation.

When $\boldsymbol{\eta}_0 \neq \mathbf{0}$ or $\boldsymbol{\eta}_T \neq \mathbf{0}$, $\mathbf{s}^\beta(\boldsymbol{\theta})$ is not generally a critical point. The boundary terms must be handled separately depending on the specific boundary conditions imposed on the problem.

Note (Parametric perturbations). A similar result holds when the linear perturbation $\epsilon\boldsymbol{\eta}$ is replaced by a general smooth parametric perturbation $\boldsymbol{\eta}(\epsilon)$ with $\boldsymbol{\eta}(0) = \mathbf{0}$: the variation $\boldsymbol{\eta}$ in equation (5) is simply replaced by $\partial_\epsilon|_{\epsilon=0}\boldsymbol{\eta}(\epsilon)$ (see proof in appendix E). This generalization is the result that will be used to prove our central result, theorem 1, where we evaluate β and $\boldsymbol{\theta}$ perturbations of the trajectory $\boldsymbol{s}^\beta(\boldsymbol{\theta})$.

This principle establishes that Euler–Lagrange solutions correspond to critical points of the action functional for boundary-vanishing variations (Case 1). This variational property enables extending EP to temporal domains: instead of computing gradients through explicit differentiation, we can approximate them by contrasting two EL trajectories—the free trajectory $\boldsymbol{s}^0(\boldsymbol{\theta})$ and the β-nudged trajectory $\boldsymbol{s}^\beta(\boldsymbol{\theta})$—analogous to the two phases in EP (section 3).

However, for arbitrary variations (Case 2), the nudged trajectory must satisfy the same boundary conditions as the free trajectory at both $t = 0$ and $t = T$. This defines a two-point boundary value problem (BVP) that cannot be solved by forward integration from initial conditions. We call boundary conditions that only constrain the initial state *causal*, since they allow forward-in-time computation; conversely, boundary conditions that constrain both endpoints are *non-causal*. Previous work [Sce21, Ken21] implicitly assumed non-causal boundary conditions, leaving this difficulty of satisfying them unaddressed. Relaxing the boundary conditions to causal ones permits efficient trajectory computation, but may introduce additional terms in the gradient formula—see theorem 1.

To understand this tradeoff between causal trajectory computation and tractable gradient formulas, we derive LEP for *arbitrary boundary conditions*. theorem 1 provides our primary result: it explicitly characterizes both the learning rule and the boundary terms that arise for any choice of boundary conditions.

Theorem 1 (LEP for arbitrary boundary conditions). *Let $t \mapsto \boldsymbol{s}_t^\beta(\boldsymbol{\theta})$ denote the solution to the Euler–Lagrange equation $\mathrm{EL}(t, \boldsymbol{\theta}, \beta) = 0$ with arbitrary boundary conditions. The gradient of the objective functional with respect to $\boldsymbol{\theta}$ is given by (with all β-derivatives evaluated at $\beta = 0$):*

$$d_{\boldsymbol{\theta}} C\left[\boldsymbol{s}^0(\boldsymbol{\theta})\right] = d_\beta \left(\int_0^T \partial_{\boldsymbol{\theta}} L_\beta \left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta} \right) \mathrm{dt} \right) \tag{6}$$

$$+ \underbrace{\left[\left(\partial_{\boldsymbol{\theta}} \boldsymbol{s}_t^0\right)^\top d_\beta \partial_{\dot{\boldsymbol{s}}} L_\beta \left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta} \right) - \left(d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0 \left(\boldsymbol{s}_t^0, \dot{\boldsymbol{s}}_t^0, \boldsymbol{\theta} \right) \right)^\top \partial_\beta \boldsymbol{s}_t^\beta \right]_0^T}_{\text{boundary residuals}}. \tag{7}$$

Note that we have omitted the explicit $\boldsymbol{\theta}$ dependence in the state trajectories $\boldsymbol{s}_t^\beta(\boldsymbol{\theta})$ and their derivatives $\dot{\boldsymbol{s}}_t^\beta(\boldsymbol{\theta})$ for notational simplicity. We adopt this convention throughout the remainder of this work.

Remark 1 (stationarity suffices). The proof of theorem 1 uses only that the trajectory $\boldsymbol{s}^\beta(\boldsymbol{\theta})$ satisfies the Euler–Lagrange equations under the imposed boundary conditions, i.e. that it is a *stationary* point of the augmented action. Whether this stationary trajectory is a minimum, a maximum, or—as is generic for mechanical actions—a saddle point never enters the argument. This property is implicit in all the results that follow from theorem 1. We state it explicitly here because action principles are often read as minimization principles, echoing the analogous observation made for static EP by Zucchet and Sacramento [ZS22].

Gradient formula interpretation. The first term on the right-hand side of (6) directly generalizes the EP learning rule (equation (1)): instead of computing differences between energy function parameters derivatives at two fixed points, we integrate differences between Lagrangian parameter derivatives over entire trajectories. This integration reflects the fact that we are now training the complete temporal evolution rather than an equilibrium state.

The second term, which we call *boundary residuals*, represents a fundamental difficulty that arises from extending EP to temporal domains. These terms emerge from the integration by parts required in the derivation of theorem 1 (see proof in appendix F) and depend on the boundary conditions imposed on the trajectories $\boldsymbol{s}^\beta(\boldsymbol{\theta})$. The fact that we have not yet specified these boundary conditions is why we refer to our theorem as a 'generalization to arbitrary boundary conditions'. As we explore in the following sections, different choices of boundary conditions yield different learning algorithms.

Implementation procedure. Focusing on the first term suggests a two-phase procedure analogous to EP, as illustrated in figure 2:

1. **Free phase**: Compute the trajectory $\boldsymbol{s}^0(\boldsymbol{\theta})$ (black cross in figure 2(a)) that is a stationary point of the action functional A_0 (black curve in figure 2(a)) by solving the associated Euler–Lagrange equation $\mathrm{EL}(\boldsymbol{\theta}, 0) = 0$ over the time interval $[0, T]$. The temporal evolution is highlighted by the black curve in figure 2(c).
2. **Nudged phase**: Compute the trajectory $\boldsymbol{s}^\beta(\boldsymbol{\theta})$ (blue dot in figure 2(a)) for a small positive value of β by solving the perturbed Euler–Lagrange equation $\mathrm{EL}(\boldsymbol{\theta}, \beta) = 0$, corresponding to the minimum of the augmented action A_β (blue curve in figure 2(a)). The corresponding dynamics are shown as the dotted blue curve in figure 2(c).
3. **Learning rule**: Estimate the gradient using the finite difference approximation of the first term in (6), combined with appropriate handling of the boundary residuals (see below in section 3.3).

Computational challenges. Unlike standard EP, LEP faces two computational challenges controlled by the choice of boundary conditions:

1. **Boundary residuals in the learning rule.** The boundary residuals in equation (7) involve $\boldsymbol{\theta}$-derivatives like $\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_T$ that would require differentiating through the ODE solver—defeating the purpose of this work.
2. **Non-causal boundary conditions.** Even when boundary residuals vanish, as previous work assumed [Sce21, Ken21], computing the nudged trajectory $\boldsymbol{s}^\beta(\boldsymbol{\theta})$ presents its own difficulties. For boundary residuals to vanish, the nudged trajectory must satisfy the same boundary conditions as the free trajectory (boundary-vanishing variations). This means one must find a trajectory that both satisfies the Euler–Lagrange equations *and* matches prescribed values at *both* endpoints—a non-causal BVP (see section 3.3.1).

These challenges motivate the search for boundary conditions that are **both causal and free of boundary residuals**, as we explore in section 3.3.

Roadmap. Table 2 summarizes, ahead of the derivations, the three boundary-condition choices studied in the remainder of the paper, together with their computational costs. CIVP (section 3.3.2) is causal but its gradient estimator retains costly boundary residuals; CBPVP (section 3.3.1), the assumption of prior work [Sce21, Ken21], is free of residuals but non-causal; PFVP (section 5), which we show underlies the RHEL algorithm (section 4), combines both advantages for time-reversible systems. These costs are derived in the following sections and in appendix O.

3.3. Instantiations of LEP

In this section, we demonstrate how to instantiate LEP by constructing the function $t \mapsto \boldsymbol{s}^\beta_t(\boldsymbol{\theta})$ through different boundary specifications. We first consider the Constant Boundary Position Value Problem (CBPVP), which corresponds to the boundary-value-problem assumption made by [Sce21, Ken21]. We then consider the Constant Initial Value Problem (CIVP) as a natural causal alternative. As we show, each resolves one of the two computational challenges identified above, but not both. Importantly, boundary conditions must be specified for an entire *family* of trajectories—those corresponding to different values of $\boldsymbol{\theta}$ and β. Figure 3 illustrates how different types of boundary conditions constrain these families: some fix both endpoints, others fix the initial state across all trajectories, and so on.

3.3.1. CBPVP on position

The boundary-value-problem assumption made by [Sce21, Ken21] corresponds to the *CBPVP*, where trajectories are constrained by conditions at both temporal boundaries:

$$\forall t \in [0, T], \quad t \mapsto \boldsymbol{s}^\beta_{\leftrightarrow, t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_T)) \text{ satisfies: } \begin{cases} \mathrm{EL}\,(t, \boldsymbol{\theta}, \beta) = 0 \\ \boldsymbol{s}^\beta_{\leftrightarrow, 0}(\boldsymbol{\theta}) = \boldsymbol{\alpha}_0 \\ \boldsymbol{s}^\beta_{\leftrightarrow, T}(\boldsymbol{\theta}) = \boldsymbol{\alpha}_T \end{cases}$$

where $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_T$ now represent the fixed positions at the initial and final times, respectively. This formulation is depicted in figure 3(b), where all trajectories connect the same boundary points but follow different internal dynamics. Applying theorem 1 to this boundary condition choice yields a direct instantiation of the general gradient formula with significant simplification due to the elimination of boundary residual terms.

Table 2. The three instantiations of LEP: recipe and computational cost (roadmap for sections 3.3–5). *Top block:* all formulations follow the same two-phase procedure and differ in which boundary conditions pin the trajectories (column subtitles) and, consequently, in how each phase is computed: straight arrows denote forward integration of the Euler–Lagrange equations (black: free phase; dashed blue: nudged phase); the ball-in-valley symbol denotes iterative extremization of the action functional over the whole discretized trajectory. In PFVP the nudged phase is the *echo*: it restarts from the final position of the free phase after reversing its velocity (equivalently, its momentum in RHEL), then integrates the same system forward again with the input presented backward; it is the only formulation whose phases run sequentially. The third row lists the terms of theorem 1 that remain to be computed: CIVP retains both final-boundary residuals, CBPVP cancels all boundary residuals, and PFVP reduces them to one starting-boundary residual for fixed initial conditions (theorem 3). *Bottom block:* **orange** marks the dominant cost that makes a method impractical, green efficient scaling; detailed derivations in appendix O. N: number of discrete time steps; d_s: state dimension; d_θ: number of learnable parameters; K: number of iterations of the action-extremization solver, growing with N and d_s. *Forward-only*: gradient computation reuses only forward passes of the physical dynamics; *streaming*: temporal data are processed sequentially in time.

	CIVP *fixed initial state*	CBPVP *fixed endpoint positions*	PFVP / RHEL *echo from free endpoint*
How the two phases are computed			
1. Free phase	→ integrate free EL; input presented forward starting position & velocity fixed	↺ extremize the free action over the whole trajectory start & end positions fixed; velocities free	→ integrate free EL; input presented forward starting position & velocity fixed; retain final position & velocity (momentum in RHEL)
2. Nudged phase	- → integrate nudged EL; input presented forward same starting position & velocity as free phase *(parallel)*	↺ extremize the nudged action over the whole trajectory same start & end positions as free phase; velocities free *(parallel)*	- → integrate nudged EL; input presented backward (echo) start from the final free-phase position, with velocity reversed (momentum in RHEL) *(sequential)*
3. Gradient terms from theorem 1	Integral term + two final-boundary residuals	Integral term only	Integral term + one starting-boundary residual
Complexity & characteristics			
Time: dynamics	$\mathcal{O}(Nd_s^2)$	**$\mathcal{O}(KNd_s^2)$**	$\mathcal{O}(Nd_s^2)$
Time: gradient	$\mathcal{O}(Nd_s^3)$	$\mathcal{O}(Nd_\theta)$	$\mathcal{O}(Nd_\theta)$
Memory: dynamics	$\mathcal{O}(d_s)$	**$\mathcal{O}(Nd_s)$**	$\mathcal{O}(d_s)$
Memory: gradient	**$\mathcal{O}(Nd_s)$**	$\mathcal{O}(d_\theta)$	$\mathcal{O}(d_\theta)$
Forward-only	x	✓	✓
Streaming	✓	x	✓
Bottleneck	BPTT memory	BVP iterations	None

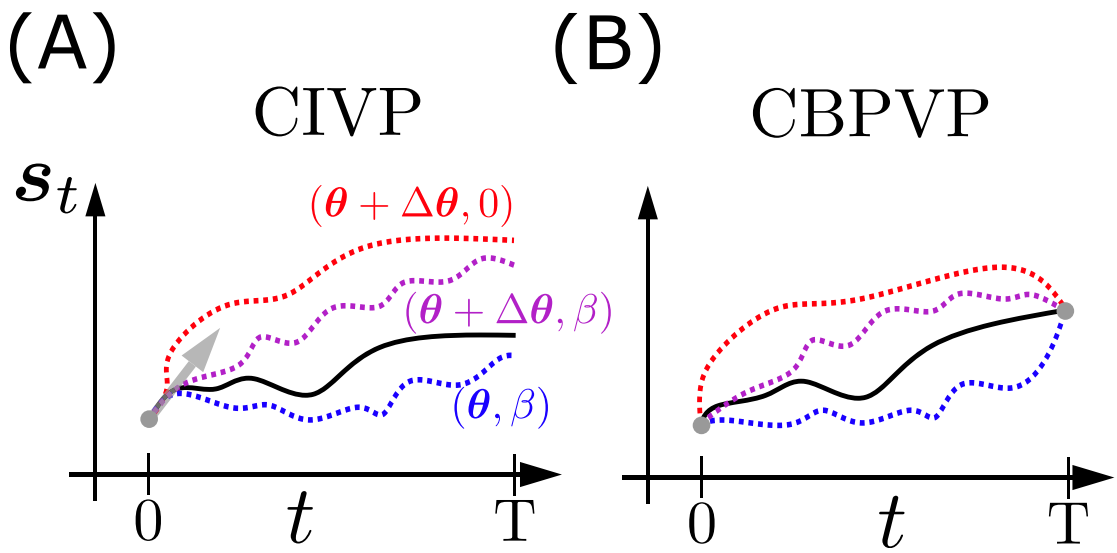


Figure 3. Different boundary condition formulations for LEP. The two panels use a consistent color scheme: black curves represent the free trajectory $s^0(\theta)$ used for inference, and blue dotted curves show the β-nudged trajectories $s^\beta(\theta)$ used for learning. Boundary conditions are depicted in grey, with dots for positions and arrows for velocities. To illustrate how boundary conditions constrain the entire family of trajectories, we also display θ-perturbed trajectories $s^0(\theta + \Delta\theta)$ (red dotted curve) and combined perturbations $s^\beta(\theta + \Delta\theta)$ (purple dotted curve). (a) Constant Initial Value Problem (CIVP). All trajectories share the same initial conditions $(s_0, \dot{s}_0) = (\alpha_0, \gamma_0)$, depicted as a grey dot for the initial position α_0 and a grey arrow for the initial velocity γ_0, but evolve differently due to parameters or nudging perturbations. (b) Constant Boundary Position Value Problem (CBPVP). All trajectories satisfy boundary conditions requiring fixed positions at $t = 0$ and $t = T$, $(s_0, s_T) = (\alpha_0, \alpha_T)$, depicted as grey dots, but their dynamics differ due to parameters or nudging perturbation.

Corollary 1 (gradient estimator for CBPVP). *The gradient of the objective functional for* $\boldsymbol{s}^{\beta}_{\leftrightarrow}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_T))$ *is given by:*

$$d_{\boldsymbol{\theta}} C\left[\boldsymbol{s}^{0}_{\leftrightarrow}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_T))\right] = \lim_{\beta\to 0}\frac{1}{\beta}\Delta^{\mathrm{CBPVP}}(\beta), \tag{8}$$

where the finite difference gradient estimator simplifies to:

$$\Delta^{\mathrm{CBPVP}}(\beta) := \int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^{\beta}_{\leftrightarrow,t},\dot{\boldsymbol{s}}^{\beta}_{\leftrightarrow,t},\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^{0}_{\leftrightarrow,t},\dot{\boldsymbol{s}}^{0}_{\leftrightarrow,t},\boldsymbol{\theta}\right)\right] \mathrm{d}t.$$

No boundary residuals but non-causal boundary conditions. The CBPVP formulation resolves the boundary residual challenge: both endpoints are fixed independently of $\boldsymbol{\theta}$ and β, causing all residual terms to vanish. This yields a simple gradient estimator that only requires integrating differences between Lagrangian derivatives over the two trajectories (equation (8)). However, given only the two endpoint conditions $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_T$, the Euler–Lagrange equation cannot be solved by forward integration from an initial condition. Instead, one must solve a two-point BVP—finding a trajectory that satisfies both the Euler–Lagrange equations *and* the prescribed endpoint constraints.

As an alternative to Euler–Lagrange forward integration, one can exploit the variational characterization to solve this two-point BVP: by lemma 1, $\boldsymbol{s}^{\beta}_{\leftrightarrow}$ is equivalently a critical point of the action subject to boundary constraints:

$$\delta_{\boldsymbol{s}} A_\beta\left[\boldsymbol{s}^{\beta}_{\leftrightarrow}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_T))\right] = 0 \quad \text{subject to} \quad \boldsymbol{s}^{\beta}_{\leftrightarrow,0} = \boldsymbol{\alpha}_0,\ \boldsymbol{s}^{\beta}_{\leftrightarrow,T} = \boldsymbol{\alpha}_T,$$

where, consistently with the note following theorem 1, stationarity rather than minimality is the defining property. A critical point can be found via gradient descent (or other root finding algorithm) on the action functional, which takes the form of a partial differential equation [Olv22]:

$$d_\tau \boldsymbol{s}_{\leftrightarrow} = -\delta_{\boldsymbol{s}} A_\beta = -\mathrm{EL}(t,\boldsymbol{\theta},\beta) \qquad \text{subject to} \quad \boldsymbol{s}_{\leftrightarrow,0} = \boldsymbol{\alpha}_0,\ \boldsymbol{s}_{\leftrightarrow,T} = \boldsymbol{\alpha}_T,$$

where τ is an artificial optimization time and $\delta_{\boldsymbol{s}} A_\beta$ is the functional gradient. In practice, the physical time $t\in[0,T]$ is discretized into N bins, turning the trajectory into a vector of size $N\times d_s$. The system then evolves iteratively in τ—analogous to the root-finding algorithms used in standard EP, but applied to this much larger state space—until the trajectory converges to a critical point where $\mathrm{EL}(t,\boldsymbol{\theta},\beta)=0$. As we quantify in table 2, this iterative solver dominates the overall cost at $\mathcal{O}(KNd_s^2)$, where K grows with N and d_s.

CBPVP eliminates boundary residuals but at the cost of non-causal trajectory computation, making it less appealing than LEP instantiations that would require simple forward passes through an ODE.

Remark 2 (unconstrained action minimization). If one is willing to accept iterative optimization—rather than forward integration via Euler–Lagrange equations—then boundary conditions need not be imposed at all. Minimizing the action functional *without* boundary constraints yields a variational formulation analogous to standard EP, where boundary residuals vanish entirely in theorem 1. However, this approach inherits the same non-causal drawbacks as CBPVP and is in fact more expensive, since the full trajectory *including* its endpoints becomes part of the optimization variables. We elaborate on this observation in appendix S.

3.3.2. CIVP

A natural attempt to restore causality is the *CIVP* , where trajectories are constructed through straightforward forward integration:

$$\forall t\in[0,T] \quad t\mapsto \boldsymbol{s}^{\beta}_{\rightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)) \text{ satisfies: } \begin{cases} \mathrm{EL}(t,\boldsymbol{\theta},\beta)=0 \\ \boldsymbol{s}^{\beta}_{\rightarrow,0}(\boldsymbol{\theta})=\boldsymbol{\alpha}_0 \\ \dot{\boldsymbol{s}}^{\beta}_{\rightarrow,0}(\boldsymbol{\theta})=\boldsymbol{\gamma}_0 \end{cases}$$

where $\boldsymbol{\alpha}_0\in\mathbb{R}^d$ and $\boldsymbol{\gamma}_0\in\mathbb{R}^d$ are the initial position and velocity conditions at $t=0$, respectively. This formulation defines a family of trajectories that all originate from the same initial state but evolve according to different dynamics due to parameter or nudging perturbations, as illustrated in figure 3(a). Unlike CBPVP, the Euler–Lagrange equation can be directly integrated forward from the initial conditions—the trajectory computation is therefore causal and efficient at $\mathcal{O}(Nd_s^2)$. Applying theorem 1 to this boundary condition choice yields a direct instantiation of the general gradient formula with some simplification due to the fixed initial conditions.

Corollary 2 (Gradient estimator for CIVP). *The gradient of the objective functional for* $\boldsymbol{s}^0_{\rightarrow}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0))$ *is given by:*

$$d_{\boldsymbol{\theta}} C\left[\boldsymbol{s}^0_{\rightarrow}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0))\right] = \lim_{\beta \to 0} \Delta^{\mathrm{CIVP}}(\beta),$$

where

$$\begin{aligned}\Delta^{\mathrm{CIVP}}(\beta) := \frac{1}{\beta}\Bigg[& \int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\rightarrow,t}, \dot{\boldsymbol{s}}^\beta_{\rightarrow,t}, \boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\rightarrow,t}, \dot{\boldsymbol{s}}^0_{\rightarrow,t}, \boldsymbol{\theta}\right)\right] \mathrm{dt} \\ & + \underbrace{\left(\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_{\rightarrow,T}\right)^\top}_{\text{costlyresidual}} \left(\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}^\beta_{\rightarrow,T}, \dot{\boldsymbol{s}}^\beta_{\rightarrow,T}, \boldsymbol{\theta}\right) - \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^0_{\rightarrow,T}, \dot{\boldsymbol{s}}^0_{\rightarrow,T}, \boldsymbol{\theta}\right)\right) \\ & - \underbrace{\left(d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^0_{\rightarrow,T}, \dot{\boldsymbol{s}}^0_{\rightarrow,T}, \boldsymbol{\theta}\right)\right)^\top}_{\text{costlyresidual}} \left(\boldsymbol{s}^\beta_{\rightarrow,T} - \boldsymbol{s}^0_{\rightarrow,T}\right)\Bigg]. \end{aligned} \tag{9}$$

Causal boundary conditions, but intractable boundary residuals. While CIVP restores causal forward integration, it suffers from significant computational limitations due to the boundary residual terms in Equation (9). In particular, the remaining residuals at time T involve derivatives of the trajectory with respect to parameters ($\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_{\rightarrow,T}$) and mixed derivatives of the Lagrangian ($d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0$), which cannot be efficiently computed using finite differences due to the high dimensionality of the parameter space (see section O.3 for a detailed complexity analysis showing these terms require $\mathcal{O}(Nd_s^3)$ time and $\mathcal{O}(Nd_s)$ memory). The only simplification occurs at $t = 0$, where the boundary residuals vanish due to the fixed initial conditions, but this is insufficient to yield a practical learning algorithm.

3.3.3. Towards a practical implementation of LEP

Designing efficient algorithms. The bottom block of table 2 quantifies the trade-off between CBPVP and CIVP in terms of computational complexity, where N denotes the number of discrete time steps, d_s the state dimension, d_θ the number of learnable parameters, and K the number of iterations required for the BVP solver convergence. For CBPVP, gradient computation is efficient at $\mathcal{O}(Nd_\theta)$ with only $\mathcal{O}(d_\theta)$ memory, but the iterative BVP solver dominates at $\mathcal{O}(KNd_s^2)$ time, where K can be expected to be a growing quantity of N and d_s. For CIVP, trajectory computation is efficient at $\mathcal{O}(Nd_s^2)$, but evaluating the boundary residuals requires a complexity of $\mathcal{O}(Nd_s^3)$ and storing intermediate states, incurring $\mathcal{O}(Nd_s)$ memory—when done using backpropagation through time (see appendix O for details).

This motivates the search for boundary conditions that are both causal and free of boundary residuals. In the following sections, we demonstrate that the Parametric Final Value Problem (PFVP) formulation, which underlies the RHEL algorithm, achieves both properties for time-reversible systems—attaining efficient $\mathcal{O}(Nd_s^2)$ dynamics and $\mathcal{O}(Nd_\theta)$ gradient computation without the bottlenecks of either CIVP or CBPVP (see table 2, introduced at the end of section 3.2).

Designing easy-to-implement algorithms. Beyond computational efficiency in time and memory, a central appeal of LEP (and EP) is that, under certain conditions, it can be *forward-only*.

An algorithm is *forward-only* if it only requires running the same physical system forward in time—no separate backward pass through a computational graph is needed. In practice, gradient computation reuses the same dynamical system as inference, requiring only two forward passes: a free phase and a nudged phase.

As summarized in table 2, CIVP is *not* forward-only: it requires an explicit backward pass through the stored computational graph to evaluate the boundary residual terms of the gradient estimator. CBPVP is forward-only, since both phases run the same iterative boundary-value-problem solver and no separate backward circuit is needed, but at the cost of an expensive iterative procedure. As we show in section 5, PFVP/RHEL satisfies the forward-only property while avoiding this overhead: both the free and echo phases consist of pure forward integration, with no iterative solver required (see appendix O for a detailed comparison).

In LEP, a further refinement of the forward-only property matters: *streaming*. An algorithm is streaming if it can process temporal data sequentially from $t = 0$ to $t = T$ without requiring access to the entire time horizon at once. As shown in table 2, causal boundary conditions (CIVP and PFVP/RHEL) naturally enable streaming, while CBPVP's non-causal boundary conditions, despite being forward-only, require all N time steps to be processed simultaneously, precluding streaming operation.

4. RHEL

RHEL presents a fundamentally different approach to temporal credit assignment compared to the variational formulations discussed in the previous section. Unlike EP methods that rely on variational principles and careful specification of boundary conditions, RHEL operates directly on the dynamics of Hamiltonian physical systems without requiring an underlying action functional or BVP formulation.

4.1. Hamiltonian system formulation

In RHEL, the system to be trained is described by a Hamiltonian function $H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)$, where $\mathbf{\Phi}_t(\boldsymbol{\theta}) \in \mathbb{R}^{2d}$ represents the complete state of the system at time t. This state vector is composed of both position and momentum coordinates:

$$\mathbf{\Phi}_t := \begin{pmatrix} \boldsymbol{s}_t \\ \boldsymbol{p}_t \end{pmatrix} \in \mathbb{R}^{2d},$$

where $\boldsymbol{s}_t \in \mathbb{R}^d$ represents the position coordinates and $\boldsymbol{p}_t \in \mathbb{R}^d$ represents the momentum coordinates.

The evolution of the system follows Hamilton's equations of motion:

$$d_t \mathbf{\Phi}_t = \boldsymbol{J} \cdot \partial_{\mathbf{\Phi}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t),$$

where $\boldsymbol{J}$ is the canonical symplectic matrix:

$$\boldsymbol{J} := \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2d \times 2d}.$$

A crucial requirement for RHEL is that the Hamiltonian must be time-reversible, meaning it satisfies:

$$H(\mathbf{\Sigma}_z \mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t) = H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t),$$

where $\mathbf{\Sigma}_z$ is the momentum-flipping operator:

$$\mathbf{\Sigma}_z := \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{I} \end{bmatrix}.$$

This time-reversibility property ensures that the system can exactly retrace its trajectory when the momentum is reversed, which is fundamental to the echo mechanism.

4.2. Two-phase learning procedure

RHEL implements a two-phase learning procedure that leverages the time-reversible nature of Hamiltonian systems. Notably, this procedure does not require solving variational problems or specifying complex boundary conditions.

Forward phase: The first phase computes the natural evolution of the system from initial conditions. For $t \in [0, T]$, the trajectory $t \mapsto \mathbf{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)$ satisfies:

$$\begin{cases} \partial_t \mathbf{\Phi}_t = \boldsymbol{J}\, \partial_{\mathbf{\Phi}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t) \\ \mathbf{\Phi}_0 = \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix} \end{cases},$$

This phase corresponds to the system's natural dynamics without any learning signal and produces the model's prediction.

Echo phase: The second phase begins by flipping the momentum of the final state and then evolving the system backward in time with a small nudging perturbation. For $t \in [0, T]$, the echo trajectory $t \mapsto \mathbf{\Phi}^e_t(\boldsymbol{\theta}, \mathbf{\Sigma}_z \mathbf{\Phi}_T(\boldsymbol{\theta}))$ satisfies:

$$\begin{cases} \partial_t \mathbf{\Phi}^e_t = \boldsymbol{J}\, \partial_{\mathbf{\Phi}} H(\mathbf{\Phi}^e_t, \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \beta \boldsymbol{J}\, \partial_{\mathbf{\Phi}} c\left(\mathbf{\Phi}^e_t, \boldsymbol{y}_{T-t}\right) \\ \mathbf{\Phi}^e_0 = \mathbf{\Sigma}_z \mathbf{\Phi}_T(\boldsymbol{\theta}) \end{cases}, \tag{10}$$

where $\beta > 0$ is a small nudging parameter.

The key insight is that without the perturbation term ($\beta = 0$), the system would exactly retrace its forward trajectory due to time-reversibility, returning to the initial state $\mathbf{\Phi}_0$. However, the nudging perturbation breaks this symmetry, and the resulting deviation encodes gradient information.

Contrary to the Lagrangian formulation, where we defined a function $t \mapsto \boldsymbol{s}_t(\boldsymbol{\theta}, \beta)$ through a unified BVP, RHEL operates with two distinct trajectories. We refer to this pair as a *Hamiltonian Echo System* (HES): $t \mapsto (\mathbf{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top), \mathbf{\Phi}^e_t(\boldsymbol{\theta}, \mathbf{\Sigma}_z \mathbf{\Phi}_T(\boldsymbol{\theta})))$. We also note that RHEL is also valid in the more general case where the cost function also depends on the momentum of the system (see equation (10)).

4.3. Gradient computation

The fundamental result of RHEL shows that gradients can be computed through finite differences between the perturbed and unperturbed Hamiltonian evaluations:

Theorem 2 (Gradient estimator from RHEL with parametrized initial state [PE25]). *The gradient of the objective functional is given by*[9]*:*

$$\mathrm{d}_{\boldsymbol{\theta}} C\left[\mathbf{\Phi}\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta}))^\top\right)\right] = \lim_{\beta \to 0} \Delta^{\mathrm{RHEL}}(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta})),$$

where the finite difference gradient estimator is:

$$\begin{aligned}\Delta^{\mathrm{RHEL}}(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta})) := \frac{1}{\beta}\Bigg[&- \int_0^T \left[\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}^e_t, \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)\right] \mathrm{d}t \\ &+ \left(\partial_{\boldsymbol{\theta}} \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix}\right)^\top \mathbf{\Sigma}_x \left(\begin{pmatrix} \boldsymbol{s}^e_T \\ \boldsymbol{p}^e_T \end{pmatrix} - \begin{pmatrix} \boldsymbol{\alpha}_0 \\ -\boldsymbol{\mu}_0 \end{pmatrix}\right)\Bigg], \end{aligned} \tag{11}$$

where $\mathbf{\Phi}^e_t$ is the echo trajectory at time t, and $\mathbf{\Phi}_t$ represents the forward trajectory evaluated at time t. We also used the helper matrix $\mathbf{\Sigma}_x$ defined as:

$$\mathbf{\Sigma}_x = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}.$$

When the initial conditions $\begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix}$ are independent of the parameters $\boldsymbol{\theta}$ (i.e. $\partial_{\boldsymbol{\theta}} \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix} = 0$), the boundary term vanishes and the estimator reduces to the integral term only.

Proof sketch. This result follows from theorem 3.1 in [PE25]. The detailed derivation, showing how to recover this result from [PE25], is provided in appendix H. □

4.4. Contrast with variational approaches

RHEL was originally derived without requiring a variational principle. Instead, it relies on establishing a direct mapping between the system dynamics and adjoint methods [PE25]. The central requirement in this approach is finding the correct mapping, which requires insight or good intuition about the structure of the problem. Attempts to generalize RHEL to the broader class of port-Hamiltonian systems [vdSJ14] using this mapping strategy have shown that the original mapping does not straightforwardly extend to such systems [PE25, appendix A.3.1].

The key insight, already exploited by RHEL, is that time-reversibility of Hamiltonian dynamics combined with a specific choice of boundary conditions can resolve the boundary residual problem identified in section 3.3.2. Specifically, the initial condition of the echo phase is defined as the momentum-flipped final state of the forward phase, allowing the system to approximately retrace its trajectory in reverse. We call this construction the *bouncing-backward kick* (formalized in proposition 2). Since Lagrangian systems also exhibit time-reversibility, the same construction carries over naturally to the LEP framework, where the kick acts on velocity rather than momentum. In the following section, we demonstrate that RHEL emerges as a special case of LEP.

Interestingly, LEP offers a more systematic derivation. Rather than relying on guesses about the correct mapping to adjoint methods, LEP starts from variational principles and lets the mathematical structure dictate the learning algorithm. This generality enables extensions that would be difficult to derive from the RHEL perspective alone. In particular, while the direct mapping approach struggled to handle dissipative systems such as port-Hamiltonians, the variational perspective naturally accommodates dissipation, as we demonstrate in section 6.

[9] We present the unidirectional formulation; the bidirectional version (centered differences) provides $O(\beta^2)$ accuracy (see appendix H.3).

5. RHEL is a particular case of the LEP

In this section, we demonstrate that RHEL can be recast as a particular instance of LEP when the system exhibits time-reversibility and the nudged trajectories are defined through a *PFVP* . This connection reveals the fundamental relationship between these seemingly different approaches to temporal credit assignment.

5.1. Instantiation of the LEP as a PFVP

5.1.1. Definition of the PFVP

We now introduce a novel boundary condition formulation that enables tractable trajectory generation while eliminating problematic boundary residuals. The key idea is to define *parametric* final boundary conditions $\boldsymbol{\alpha}_T(\boldsymbol{\theta})$ and $\boldsymbol{\gamma}_T(\boldsymbol{\theta})$ that depend on the parameters $\boldsymbol{\theta}$. This defines the *PFVP* :

$$\forall t \in [0,T] \quad t \mapsto \boldsymbol{s}^{\beta}_{\leftarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T(\boldsymbol{\theta}),\boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \text{ satisfies: } \begin{cases} \mathrm{EL}_r(t,\boldsymbol{\theta},\beta)=0 \\ \boldsymbol{s}^{\beta}_{\leftarrow,T}(\boldsymbol{\theta})=\boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ \dot{\boldsymbol{s}}^{\beta}_{\leftarrow,T}(\boldsymbol{\theta})=\boldsymbol{\gamma}_T(\boldsymbol{\theta}) \end{cases}, \tag{12}$$

where $\mathrm{EL}_r(t,\boldsymbol{\theta},\beta)$ denotes the time-indexed Euler–Lagrange equation with reversible Lagrangian L_r:

$$\mathrm{EL}_r(t,\boldsymbol{\theta},\beta) := \partial_{\boldsymbol{s}} L_\beta(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t,\boldsymbol{y}_t) - d_t \partial_{\dot{\boldsymbol{s}}} L_\beta(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t,\boldsymbol{y}_t).$$

A reversible Lagrangian satisfies the time-symmetry condition:

$$L_r(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t,\boldsymbol{y}_t) = L_r(\boldsymbol{s}_t,-\dot{\boldsymbol{s}}_t,\boldsymbol{\theta},\boldsymbol{x}_t,\boldsymbol{y}_t).$$

This ensures that solutions of the associated Euler–Lagrange equations are time-reversible: forward evolution followed by momentum reversal exactly retraces the original trajectory.

In our instantiation, the parametric boundary conditions $\boldsymbol{\alpha}_T(\boldsymbol{\theta})$ and $\boldsymbol{\gamma}_T(\boldsymbol{\theta})$ are defined with a CIVP with $\beta = 0$ (that will then be used for practically running the free phase, see section 5.1.2). Specifically, they correspond to the final position and velocity of this CIVP:

$$\begin{cases} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) := \boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)) \\ \boldsymbol{\gamma}_T(\boldsymbol{\theta}) := \dot{\boldsymbol{s}}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)) \end{cases}, \tag{13}$$

where $\boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$ and $\dot{\boldsymbol{s}}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$ are the final position and velocity from the CIVP solution without nudging (see section 3.3.2). This choice ensures that the free trajectory ($\beta = 0$) satisfies both the CIVP initial conditions and the PFVP final conditions simultaneously (see figure 4(a)).

5.1.2. Practical computation of the PFVP

Final value problems are generally difficult to solve, as one must find initial conditions that produce prescribed final states—typically requiring iterative root-finding or constrained optimization (see section 3.3.1). However, the PFVP formulation admits efficient computation by converting both phases into Initial Value Problems (IVPs).

Free phase. By construction, the free trajectory is obtained directly from the CIVP. The FVP $\boldsymbol{s}^0_{\leftarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T(\boldsymbol{\theta}),\boldsymbol{\gamma}_T(\boldsymbol{\theta})))$ is equivalent to the CIVP $\boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$ with constant initial conditions $\boldsymbol{\alpha}_0$ and $\boldsymbol{\gamma}_0$ (See proposition 4 for details). This trajectory can be computed via standard forward integration from $t = 0$ to $t = T$.

Nudged phase: the bouncing-backward kick. For the nudged trajectory ($\beta \neq 0$), we exploit the time-reversibility of the system to convert the PFVP into an IVP. The key insight is that applying a *velocity kick*—reversing the velocity at the final boundary—allows us to integrate the *same*[10] dynamical system forward in time rather than solving a final value problem. We call this the *bouncing-backward kick*: the system 'bounces' off the final state of the free phase and retraces its path backward in physical time, using only forward integration. In the Lagrangian formulation, the kick acts on the *velocity* ($\boldsymbol{\gamma}_T \to -\boldsymbol{\gamma}_T$); in the equivalent Hamiltonian formulation (RHEL), it acts on the *momentum* ($\boldsymbol{p} \to -\boldsymbol{p}$, the Σ_z flip).

[10] Both phases integrate the *same* Euler–Lagrange equations, unlike the adjoint state method [CRBD18], which often uses time-reversibility but integrates a different ODE to recompute activations during the backward pass, on top of integrating the adjoint equations themselves.

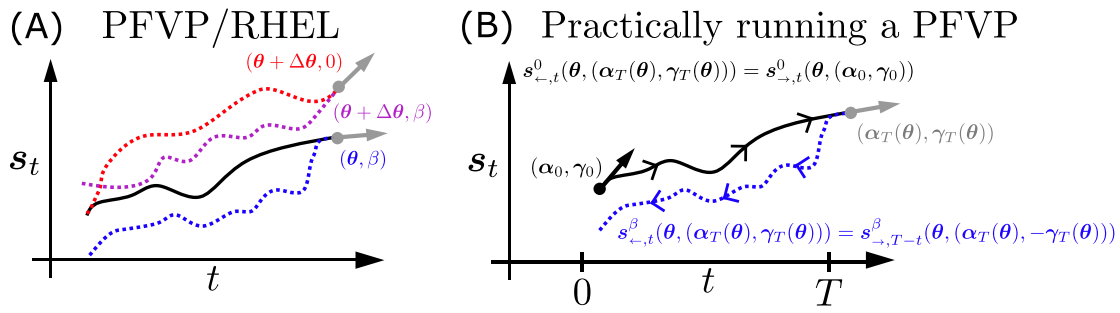


Figure 4. Parametric Final Value Problem (PFVP) for LEP. The two panels use a consistent color scheme: black curves represent the free trajectory $s^0(\theta)$ used for inference, and blue dotted curves show the β-nudged trajectories $s^\beta(\theta)$ used for learning. The boundary conditions (parametric final value problem) are depicted in grey, with a dot for the position $\alpha_T(\theta)=s^0_{\rightarrow,T}(\theta)$ and arrow $\gamma_T(\theta)=\dot{s}^0_{\rightarrow,T}(\theta)$. To illustrate how boundary conditions constrain the entire family of trajectories, we also display θ-perturbed trajectories $s^0(\theta+\Delta\theta)$ (red dotted curve) and combined perturbations $s^\beta(\theta+\Delta\theta)$ (purple dotted curve). (a) We observe the effect of the *parametric* final value condition: only trajectories that share the same θ (blue and black vs red and purple, respectively) satisfy the same position (grey dot) and velocity final value conditions (grey arrow). (b) The arrows on the curves (blue and black) indicate the direction of integration of their respective IVPs. Although Final Value Problems (FVPs) are generally difficult to solve, both the free phase (black curve) and the nudged phase (blue curve) can be efficiently computed by reformulating them as Initial Value Problems (IVPs). For the free phase, the FVP $s^0_{\leftarrow,t}(\theta,(\alpha_T,\gamma_T))$ is equivalent to the CIVP $s^0_{\rightarrow,t}(\theta,(\alpha_0,\gamma_0))$ with initial conditions α_0 and γ_0 shown in black (dot and arrow, respectively). For the nudged phase, we exploit time-reversibility (proposition 2): the PFVP $s^\beta_{\leftarrow,t}(\theta,(\alpha_T,\gamma_T))$ becomes the Parametric Initial Value Problem (PIVP) $s^\beta_{t'}(\theta,(\alpha_T,-\gamma_T))$ starting from the momentum-reversed final conditions $(\alpha_T,-\gamma_T)$. This PIVP is then integrated *forward* in integration time $t'=T-t$ from 0 to T (corresponding to t going backward from T to 0), as illustrated by the blue arrows. This PFVP formulation, expressed through Lagrangian mechanics, corresponds exactly to the Hamiltonian formulation of RHEL after applying the forward Legendre transform (see theorem 4).

Proposition 2 (Bouncing-backward kick: PFVP-to-IVP reduction). *The solution of the time-reversible PFVP (12) with boundary conditions $\alpha_T(\theta)$ and $\gamma_T(\theta)$ satisfies:*

$$\forall t\in[0,T]\qquad s^\beta_{\leftarrow,t}(\theta,(\alpha_T(\theta),\gamma_T(\theta)))=s^\beta_{\rightarrow,t'}(\theta,(\alpha_T(\theta),-\gamma_T(\theta)))\quad\text{with } t'=T-t,$$

where $t'\mapsto s^\beta_{\rightarrow,t'}(\theta,(\alpha_T(\theta),-\gamma_T(\theta)))$ is the solution of the IVP with velocity-reversed initial conditions, integrated forward in time t' from 0 to T (where $t'=T-t$ relates the integration time t' to the time t):

$$\forall t'\in[0,T]\quad t'\mapsto s^\beta_{\rightarrow,t'}(\theta,(\alpha_T(\theta),-\gamma_T(\theta)))\text{ satisfies: }\begin{cases}\mathrm{EL}_r(t',\theta,\beta)=0\\ s^\beta_{\rightarrow,0}(\theta)=\alpha_T(\theta)\\ \dot{s}^\beta_{\rightarrow,0}(\theta)=-\gamma_T(\theta)\end{cases}$$

The proposition states that the PFVP solution $s^\beta_{\leftarrow,t}$ at physical time t equals the IVP solution s^β_{T-t} at integration time $T-t$. Crucially, the Euler–Lagrange equation $\mathrm{EL}_r(T-t,\theta,\beta)$ is evaluated with the input x_{T-t} and target y_{T-t} corresponding to physical time $T-t$, meaning the input and target sequences are played backward during integration.

In practice, this gives a simple algorithm for the nudged phase: (1) start from the final state of the free phase with reversed velocity $(\alpha_T(\theta),-\gamma_T(\theta))$, and (2) introduce a new integration time variable $t'=T-t$ and integrate the IVP *forward in time* t' from $t'=0$ to $t'=T$ (corresponding to physical time t going backward from T to 0) while feeding the inputs and targets in reverse temporal order. The resulting IVP trajectory $s^\beta_{t'}$, yields the desired PFVP solution $s^\beta_{\leftarrow,t}$ (see figure 4(b)).

5.2. Boundary residual cancellation in PFVP

Applying theorem 1 to this parametric boundary condition choice yields a remarkable instantiation of the general gradient formula where both the boundary conditions and the time-reversibility cause the boundary residuals to partially cancel.

Theorem 3 (PFVP boundary residual cancellation). *Recall that the parametric boundary conditions $\alpha_T(\theta):=s^0_{\rightarrow,T}(\theta,(\alpha_0,\gamma_0))$ and $\gamma_T(\theta):=\dot{s}^0_{\rightarrow,T}(\theta,(\alpha_0,\gamma_0))$ are defined as the final position and velocity of the free-phase CIVP (equation (13)). The boundary residuals in theorem 1 vanish at $t=T$ and reduce to easy-to-compute terms at $t=0$ for the PFVP formulation $s^\beta_{\leftarrow}(\theta,(\alpha_T,\gamma_T))$. The gradient of the objective functional is given by:*

$$d_\theta C\left[s^0_{\leftarrow}(\theta,(\alpha_T,\gamma_T))\right]=\lim_{\beta\to0}\Delta^{\mathrm{PFVP}}(\beta,\alpha_0(\theta),\gamma_0(\theta)),$$

where the PFVP gradient estimator simplifies to:

$$\begin{aligned}\Delta^{\mathrm{PFVP}}\left(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta})\right) := \frac{1}{\beta}\Bigg[& \int_0^T \left(\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t}, \dot{\boldsymbol{s}}^\beta_{\leftarrow,t}, \boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\leftarrow,t}, \dot{\boldsymbol{s}}^0_{\leftarrow,t}, \boldsymbol{\theta}\right)\right) \mathrm{dt} \\ & + \left(d_{\boldsymbol{\theta}\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\right)^\top \left(\boldsymbol{s}^\beta_{\leftarrow,0} - \boldsymbol{\alpha}_0(\boldsymbol{\theta})\right) \\ & - \left(\partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}_0\right)^\top \left(\partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^\beta_{\leftarrow,0}, \dot{\boldsymbol{s}}^\beta_{\leftarrow,0}, \boldsymbol{\theta}\right) - \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\right)\Bigg].\end{aligned}$$

Note: When the initial conditions $\boldsymbol{\alpha}_0$ and $\boldsymbol{\gamma}_0$ are independent of $\boldsymbol{\theta}$ (i.e. $\partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}_0 = 0$), the boundary residual simplifies to a single term: $\left(\partial_{\boldsymbol{\theta}\dot{\boldsymbol{s}}} L_0(\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0, \boldsymbol{\theta})\right)^\top \left(\boldsymbol{s}^\beta_{\leftarrow,0} - \boldsymbol{\alpha}_0\right)$.

Computational advantages. The PFVP formulation resolves both computational challenges identified earlier. Unlike CIVP, it avoids intractable boundary residuals that would require backpropagation-like computations. Unlike CBPVP, it uses causal boundary conditions—trajectories are computed via simple forward integration rather than iterative solvers, enabling efficient streaming computation.

Table 2 confirms these advantages quantitatively. PFVP achieves efficient trajectory generation at $\mathcal{O}(Nd_s^2)$ time with only $\mathcal{O}(d_s)$ memory, matching CIVP's forward integration cost. Simultaneously, its gradient computation scales as $\mathcal{O}(Nd_\theta)$ with $\mathcal{O}(d_\theta)$ memory, matching CBPVP's efficient gradient estimation.

Comparison with previous work. Recently, [Mas25] proposed a LEP formulation; however, their work considers only *fixed* boundary conditions (such as our CBPVP). The central novelty of our PFVP is making the final boundary *parametric*: the terminal constraints $\boldsymbol{s}^\beta_{\leftarrow,T}(\boldsymbol{\theta}) = \boldsymbol{\alpha}_T$ and $\dot{\boldsymbol{s}}^\beta_{\leftarrow,T}(\boldsymbol{\theta}) = \boldsymbol{\gamma}_T$ depend on $\boldsymbol{\theta}$ through the free-phase CIVP (equation (13)).

Fixing $\boldsymbol{\alpha}_T$ and $\boldsymbol{\gamma}_T$ independently of $\boldsymbol{\theta}$ would make the system less expressive and make the initial state depend on $\boldsymbol{\theta}$, forcing the initial conditions to change at every training step. To run an inference with the input in the forward direction, one would need to recompute it after each training step.

5.3. Hamiltonian-Lagrangian equivalence via legendre transform

We now establish the precise mathematical relationship between the PFVP formulation of LEP and RHEL. We first introduce the Legendre transform and its condition of well-definiteness to define a bijection between two pairs of variables. We use this to show the equivalence between LEP and RHEL.

This transform is important for our work because it allows us to map solutions of the Euler–Lagrange equations bijectively to solutions of the Hamiltonian equations.

Theorem 4 (LEP-RHEL Equivalence via Legendre Transform). *The time-local Legendre transform (proposition 1), applied pointwise along trajectories, creates an equivalence between LEP and RHEL at the level of trajectories (1) and gradient estimators (2).*

(1) Trajectory Equivalence. *The PFVP formulation of LEP and the HES formulation of RHEL establish a bijection between solutions of Euler–Lagrange and Hamiltonian equations:*

$$t \mapsto \boldsymbol{s}^\beta_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T, \boldsymbol{\gamma}_T)) \quad \longleftrightarrow \quad t \mapsto \left(\boldsymbol{\Phi}_t\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right), \boldsymbol{\Phi}^e_t(\boldsymbol{\theta}, \Sigma_z \boldsymbol{\Phi}_T(\boldsymbol{\theta}))\right),$$

where the Legendre transformation induces the invertible relation between $(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}))$ and $\begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta}) \\ \boldsymbol{\mu}_0(\boldsymbol{\theta})\end{pmatrix}$:

$$\begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta}) \\ \boldsymbol{\mu}_0(\boldsymbol{\theta})\end{pmatrix} = \begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta}) \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\end{pmatrix} \quad \text{and} \quad \begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta}) \\ \boldsymbol{\gamma}_0(\boldsymbol{\theta})\end{pmatrix} = \begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta}) \\ \partial_{\boldsymbol{p}} H_0\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\end{pmatrix}, \tag{14}$$

where $\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0$ are the Lagrangian initial conditions (position and velocity at $t = 0$), and $\begin{pmatrix}\boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0\end{pmatrix}$ are the Hamiltonian initial conditions (position and momentum at $t = 0$), related via the bijective mapping of equation (14).

(2) Gradient Equivalence. *Under the respective Legendre transforms, the gradient estimators are identical:*

$$\Delta^{\mathrm{PFVP}}\left(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta})\right) = \Delta^{\mathrm{RHEL}}\left(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta})\right).$$

LEP (Lagrangian)

$$\Delta^{\mathrm{PFVP}}(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)=\frac{1}{\beta}\left[\int_0^T\left(\partial_{\boldsymbol{\theta}}L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right)dt\right.$$
$$\left.+\begin{pmatrix}\boldsymbol{\alpha}_0\\ \partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}^{\top}\boldsymbol{\Sigma}_x\left(\begin{pmatrix}\boldsymbol{s}^\beta_{\leftarrow,0}\\ -\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,0},\dot{\boldsymbol{s}}^\beta_{\leftarrow,0},\boldsymbol{\theta}\right)\end{pmatrix}-\begin{pmatrix}\boldsymbol{\alpha}_0\\ -\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}\right)\right].$$

RHEL (Hamiltonian)

$$\Delta^{\mathrm{RHEL}}(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\mu}_0)=\frac{1}{\beta}\left[-\int_0^T\left(\partial_{\boldsymbol{\theta}}H_\beta\left(\boldsymbol{\Phi}^e_t,\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t,\boldsymbol{\theta}\right)\right)\mathrm{d}t+\left(\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \boldsymbol{\mu}_0\end{pmatrix}\right)^{\top}\boldsymbol{\Sigma}_x\left(\begin{pmatrix}\boldsymbol{s}^e_T\\ \boldsymbol{p}^e_T\end{pmatrix}-\begin{pmatrix}\boldsymbol{\alpha}_0\\ -\boldsymbol{\mu}_0\end{pmatrix}\right)\right].$$

where the $\boldsymbol{\theta}$ dependencies on $\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0$ and $\boldsymbol{\alpha}_0,\boldsymbol{\mu}_0$—which are constrained by equation (14)—were dropped for readability. The color coding highlights terms that are equal between LEP and RHEL: blue for the integral terms, red for the parameter derivatives before $\boldsymbol{\Sigma}_x$, and green for the state differences after $\boldsymbol{\Sigma}_x$.

Sketch of the proof. The proof proceeds in three steps.

(1) **Legendre correspondence.** We first show that the Legendre transform establishes a bijection between solutions of the Euler–Lagrange and Hamilton equations. Since the transform itself depends on the parameters $\boldsymbol{\theta}$, it not only maps entire trajectories between the two formalisms but also reparametrizes their initial conditions in a $\boldsymbol{\theta}$-dependent manner.

(2) **PFVP–HES construction.** For both $\beta=0$ and $\beta\neq 0$, we construct the HES from the PFVP through a sequence of maps (including the Legendre transform), each of which is bijective.

(3) **Gradient equivalence.** Finally, applying the Legendre transform to the PFVP gradient estimator yields the RHEL gradient expression. Term by term, the Lagrangian estimator in LEP matches the Hamiltonian estimator in RHEL, establishing full gradient equivalence.

□

Theoretical significance. The combination of theorems 3 and 4 establishes a fundamental result: RHEL can be derived from first principles using variational methods of EP. Theorem 3 demonstrates that the PFVP formulation is a solution instance of LEP, the first one we found that does not have problematic boundary residuals, and thus can be used to train Lagrangian systems. Furthermore, we can also recover the RHEL learning rule for Hamiltonian systems: theorem 4 shows that this computationally viable LEP formulation is mathematically equivalent to RHEL through the Legendre transformation. This equivalence provides a new theoretical foundation for RHEL, revealing that its distinctive properties—forward-only computation, scalability independent of model size, and local learning—emerge naturally from the variational structure of physical systems rather than being only the consequence of specific Hamiltonian dynamics.

5.4. Empirical validation

We now provide numerical validation of theorem 4 by training a Hopfield-inspired dynamical system using both RHEL (Hamiltonian formulation) and LEP (Lagrangian formulation), demonstrating that the two approaches yield identical gradients.

5.4.1. Example of equivalence: fixed Hamiltonian initial conditions

Learning rule analysis. Consider the case where the Hamiltonian initial conditions $\boldsymbol{\alpha}_0$ and $\boldsymbol{\mu}_0$ are fixed independently of $\boldsymbol{\theta}$, i.e. $\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0=0$ and $\partial_{\boldsymbol{\theta}}\boldsymbol{\mu}_0=0$. In this setting, the red boundary term in theorem 4 vanishes, and both gradient estimators reduce to the blue integral term only:

$$\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \boldsymbol{\mu}_0\end{pmatrix}=\mathbf{0}\quad\Rightarrow\quad\Delta^{\mathrm{RHEL}}(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\mu}_0)=-\frac{1}{\beta}\int_0^T\left[\partial_{\boldsymbol{\theta}}H_\beta\left(\boldsymbol{\Phi}^e_t,\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t,\boldsymbol{\theta}\right)\right]\mathrm{d}t$$
$$=\Delta^{\mathrm{PFVP}}\left(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0\left(\boldsymbol{\theta}\right)\right),$$

where $\boldsymbol{\gamma}_0(\boldsymbol{\theta})=\partial_{\boldsymbol{p}}H_0(\boldsymbol{\alpha}_0,\boldsymbol{\mu}_0,\boldsymbol{\theta})$ is the corresponding Lagrangian initial velocity. The LEP gradient estimator takes the equivalent form:

$$\Delta^{\mathrm{PFVP}}\left(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0\left(\boldsymbol{\theta}\right)\right)=\frac{1}{\beta}\int_0^T\left[\partial_{\boldsymbol{\theta}}L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right]\mathrm{d}t.$$

Table 3. Parameter gradients for the Hopfield-inspired system. The symbol $\odot$ denotes element-wise multiplication. The relation $\partial_{\boldsymbol{\theta}} H_0 = -\partial_{\boldsymbol{\theta}} L_0$ (lemma 4) is verified for each parameter. For $\boldsymbol{W}$, the gradient simplifies due to its symmetry. For the time constant $\boldsymbol{\tau}$, using $\dot{\boldsymbol{s}} = \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{p}$ confirms that $\frac{1}{2}\dot{\boldsymbol{s}} \odot \dot{\boldsymbol{s}} = \frac{1}{2}\boldsymbol{p} \odot \boldsymbol{p} \odot \boldsymbol{\tau}^{-2}$.

Parameter	LEP: $\partial_{\boldsymbol{\theta}} L_0$	RHEL: $\partial_{\boldsymbol{\theta}} H_0$
W	$-\frac{1}{2}\rho(\boldsymbol{s})\rho(\boldsymbol{s})^\top$	$\frac{1}{2}\rho(\boldsymbol{s})\rho(\boldsymbol{s})^\top$
B	$-\rho(\boldsymbol{s})$	$\rho(\boldsymbol{s})$
τ	$\frac{1}{2}\dot{\boldsymbol{s}} \odot \dot{\boldsymbol{s}}$	$-\frac{1}{2}\boldsymbol{p} \odot \boldsymbol{p} \odot \boldsymbol{\tau}^{-2}$

Both learning rules compare parameter derivatives along the free and nudged trajectories, differing only in whether Hamiltonian or Lagrangian variables are used.

Initial condition analysis. Crucially, fixing the *Hamiltonian* initial conditions induces *parametric* Lagrangian initial conditions. Through the Legendre transform (equation (14)), the initial velocity in the Lagrangian formulation is:

$$\boldsymbol{\gamma}_0(\boldsymbol{\theta}) = \partial_{\boldsymbol{p}} H_0(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0, \boldsymbol{\theta}).$$

When H_0 depends on $\boldsymbol{\theta}$ (e.g. through a mass matrix or time constant parameters), the initial velocity $\boldsymbol{\gamma}_0$ becomes $\boldsymbol{\theta}$-dependent even though the Hamiltonian initial conditions are fixed. This subtlety is illustrated in figure 5(b), where the Lagrangian phase portraits show varying initial velocities across training epochs as parameters evolve.

Remark 3 (Simplification for zero initial momentum). In practice, if one wishes to avoid implementing the boundary term in the learning rule, one can set $\boldsymbol{\mu}_0 = \mathbf{0}$. This yields $\boldsymbol{\gamma}_0 = \partial_{\boldsymbol{p}} H_0(\boldsymbol{\alpha}_0, \mathbf{0}, \boldsymbol{\theta}) = \mathbf{0}$ for standard kinetic energies, making both initial conditions non-parametric.

5.4.2. Hopfield-inspired system with learnable time constants

We validate our theoretical results on a Hopfield-inspired dynamical system, based on the Hopfield model in table 1. For simplicity, we set $\alpha = 0$ and $b = \mathbf{0}$ (no regularization or bias in the potential). The Lagrangian takes the form (see table 1):

$$L_0(\boldsymbol{s}, \dot{\boldsymbol{s}}, \boldsymbol{\theta}, \boldsymbol{x}) = \frac{1}{2}\dot{\boldsymbol{s}}^\top \mathrm{diag}(\boldsymbol{\tau})\,\dot{\boldsymbol{s}} - \frac{1}{2}\rho(\boldsymbol{s})^\top \boldsymbol{W}\rho(\boldsymbol{s}) - \boldsymbol{B}^\top \rho(\boldsymbol{s}) - \rho(\boldsymbol{x})^\top \rho(\boldsymbol{s}), \tag{15}$$

where $\boldsymbol{s} \in \mathbb{R}^d$ is the state, $\rho(\cdot)$ is an element-wise activation function (e.g. tanh), $\boldsymbol{\tau} \in \mathbb{R}^d_{>0}$ is a vector of learnable time constants, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is the symmetric recurrent weight matrix, $\boldsymbol{B} \in \mathbb{R}^d$ is a bias vector, and $\boldsymbol{x}_t \in \mathbb{R}^d$ is the time-varying input. The learnable parameters are $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{B}, \boldsymbol{\tau})$.

Parameter gradients. Table 3 summarizes the parameter gradients in both formalisms. Notably, the gradient with respect to $\boldsymbol{W}$ takes the form $\rho(\boldsymbol{s})\rho(\boldsymbol{s})^\top$, which corresponds to a *Hebbian learning rule*—one of the most famous and oldest learning rules in neuroscience [DP25]. Additionally, the gradient with respect to $\boldsymbol{\tau}$ takes different forms in each formulation: in the Lagrangian it depends on velocities $\dot{\boldsymbol{s}}$, while in the Hamiltonian it depends on momenta $\boldsymbol{p}$. These are related through the Legendre transform and yield identical learning signals.

Initial condition mapping. For fixed Hamiltonian initial conditions $(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)$, the corresponding Lagrangian initial conditions are:

$$\begin{aligned} &\text{Position:} \quad \boldsymbol{\alpha}_0 \quad \text{(unchanged)} \\ &\text{Velocity:} \quad \boldsymbol{\gamma}_0 = \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{\mu}_0 \quad (\boldsymbol{\theta}-\text{dependent through}\,\boldsymbol{\tau}). \end{aligned}$$

This $\boldsymbol{\theta}$-dependence of the Lagrangian initial velocity through the learnable time constants $\boldsymbol{\tau}$ is what makes the initial conditions parametric in the LEP formulation, as illustrated in figure 5(b) where the initial velocity changes across training epochs.

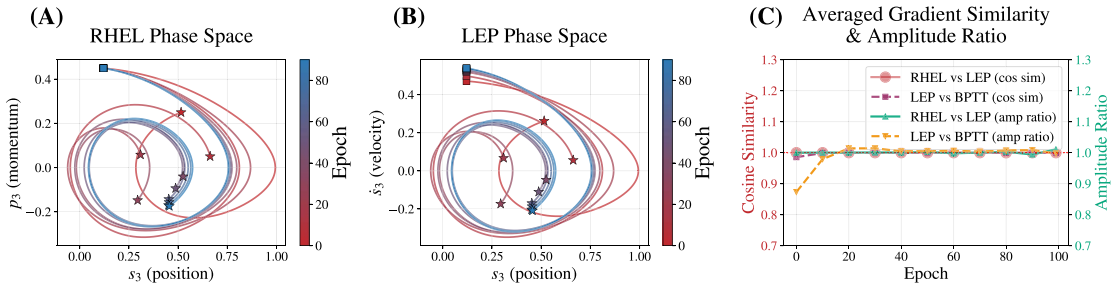


Figure 5. Numerical validation of theorem 4 with two separate training runs. A 6-dimensional Hopfield-inspired system (equation (15)) is trained in two separate runs from the same initial parameters: one with Hamiltonian parameterization + RHEL, one with Lagrangian parameterization + LEP. The phase portrait of 'hidden layer' neuron 3 (position $s^{(3)}$ vs. momentum/velocity) is shown across training epochs (colored trajectories from red to blue); squares mark initial conditions, stars mark final conditions. (a) RHEL training run with Hamiltonian parameterization $(\boldsymbol{s}, \boldsymbol{p})$. (b) LEP training run with Lagrangian parameterization $(\boldsymbol{s}, \dot{\boldsymbol{s}})$. (c) Cosine similarity and amplitude ratio between gradient estimates: LEP vs BPTT (purple, orange) and RHEL vs LEP (red, green).

5.4.3. Experimental setup

Task. We consider a teacher-student learning setup with a 6-dimensional system ($d = 6$). The input signal $\boldsymbol{x}_t$ is injected into neuron 0 and consists of a superposition of 10 random sine waves:

$$x_t = \frac{1}{n_{\text{waves}}} \sum_{k=1}^{n_{\text{waves}}} a_k \sin\left(2\pi f_k t + \phi_k\right),$$

where frequencies f_k are uniformly sampled from $[10^{-2}, 1]$ Hz, phases ϕ_k from $[0, 2\pi]$, and amplitudes a_k from $[0.5, 1.5]$. The target output $\boldsymbol{y}_t$ is generated by a teacher network with the same architecture but different random initialization. The input is thus a continuously arriving, time-varying signal: within each phase, the student processes it by forward integration as it arrives, without ever storing the state trajectory, while the same teacher-generated input and target trajectories are reused across training epochs. The cost function is the squared error on neuron 5: $c(\boldsymbol{s}_t, \boldsymbol{y}_t) = \frac{1}{2}(s_t^{(5)} - y_t^{(5)})^2$. We use Euler integration with time step $\mathrm{dt} = 0.001$, total duration $T = 10$, and nudging strength $\beta = 0.01$.

Parameter initialization. The weight matrix $\boldsymbol{W}$ is initialized via QR decomposition: a random orthogonal matrix $\boldsymbol{U}$ is obtained from the QR factorization of a Gaussian matrix, and eigenvalues are sampled uniformly from $[0.1, 1.0]$, yielding $\boldsymbol{W} = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\lambda})\,\boldsymbol{U}^\top$ for controlled spectral properties. Time constants $\boldsymbol{\tau}$ are sampled uniformly from $[0.5, 1.0]$. We use the Adam optimizer with learning rate 0.005 and random seed 50. Full hyperparameter details are given in appendix P.

We perform two separate training runs of 100 epochs each, both starting from the same initial parameter values $\boldsymbol{\theta}_0$: (1) a RHEL training run using Hamiltonian parameterization with state variables $(\boldsymbol{s}, \boldsymbol{p})$ and learning rules from table 3 (right column), and (2) a LEP training run using Lagrangian parameterization with state variables $(\boldsymbol{s}, \dot{\boldsymbol{s}})$ and learning rules from table 3 (left column). The Hamiltonian initial conditions $(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)$ are fixed and identical for both runs; in the LEP run, these map to Lagrangian initial conditions $(\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0)$ where $\boldsymbol{\gamma}_0 = \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{\mu}_0$ evolves as $\boldsymbol{\tau}$ changes during training. During LEP training, at every gradient update we also compute the gradient provided by automatic differentiation (BPTT) for comparison.

The experiment confirms the predictions of theorem 4. The two separate training runs—one with Hamiltonian parameterization and RHEL learning rule, one with Lagrangian parameterization and LEP learning rule—both start from the same initial parameter values $\boldsymbol{\theta}_0$ and evolve the parameters independently. In the RHEL run (figure 5(a)), the Hamiltonian initial conditions $(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)$ remain fixed across training epochs while the input signal (a superposition of sine waves) drives complex oscillatory dynamics. In the LEP run (figure 5(b)), the same fixed Hamiltonian initial conditions $(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)$ map to Lagrangian initial conditions where the initial velocity $\boldsymbol{\gamma}_0 = \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{\mu}_0$ shifts across epochs as the time constant parameters $\boldsymbol{\tau}$ evolve during training, illustrating the $\boldsymbol{\theta}$-dependence of boundary conditions under the Legendre transform. Despite these two independent training runs using different parameterizations and learning rules, the LEP and RHEL gradient estimates agree nearly perfectly throughout training (cosine similarity ≈ 1, amplitude ratio ≈ 1), and both closely match the ground-truth BPTT gradients obtained via automatic differentiation. appendix Q reports learning curves from an additional

replication of the same teacher–student task and the reconstruction of the teacher's output trajectory; the Discussion paragraph *Relation to BPTT* discusses why gradient agreement is the relevant validation target.

6. From LEP to dissipative LEP

The non-dissipative nature of standard Hamiltonian/Lagrangian systems has been recognized as a limitation in both the LEP and HEL literatures, on two fronts. From a hardware perspective, energy conservation restricts the class of physical systems where LEP can be implemented; to address this, Kendall *et al* [Ken21] proposed using fractional calculus to extend Lagrangian mechanics to dissipative dynamics. From a machine learning perspective, the absence of dissipation means that, like Unitary RNNs before them [JGP+17], Lagrangian/Hamiltonian systems cannot forget [PE25, LPM23, BRR25].

In this section, we take a first step toward addressing this limitation by extending LEP to dissipative systems. We show that dissipation can be introduced through an exponential integrating factor in the Lagrangian, and made practical via the PFVP formulation: during the free phase, the system genuinely dissipates energy, while during the nudge phase, energy is pumped back in.

6.1. Energy conservation in standard Lagrangian systems

To understand the non-dissipative nature of standard Lagrangian systems, we first consider an *isolated system* without external input. Let $L_0^{\text{iso}}(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta})$ denote the Lagrangian of the isolated system, obtained by setting $\boldsymbol{x}_t = 0$ in the full Lagrangian $L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)$. For any Lagrangian system, there exists a conserved quantity [Olv22]:

$$E = \dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - L_0^{\text{iso}}. \tag{16}$$

This quantity E is the *physical energy* of the system: kinetic energy plus internal potential energy, corresponding to the standard notion of mechanical energy in classical physics.

For the isolated system satisfying the Euler–Lagrange equations $\partial_{\boldsymbol{s}} L_0^{\text{iso}} - d_t \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} = 0$, this energy is conserved: $d_t E = 0$. This can be verified by direct computation:

$$\begin{aligned} d_t E &= d_t \left(\dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} \right) - d_t L_0^{\text{iso}} \\ &= \dot{\boldsymbol{s}}_t^\top \left(d_t \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - \partial_{\boldsymbol{s}} L_0^{\text{iso}} \right) = 0. \end{aligned}$$

Note that when the external input $\boldsymbol{x}_t$ is applied, it introduces in the Lagrangian $L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)$ a time dependence that breaks this energy conservation. The system can exchange energy with its environment through the input, but does not dissipate energy by itself.

6.2. Dissipative LEP

To address the limitation identified above, we extend LEP to dissipative systems by introducing an explicitly time-dependent Lagrangian through an exponential integrating factor. This approach generalizes a known method for simulating dissipation [Rie96] to the multivariate case.

Construction of the dissipative Lagrangian. We scale the standard physical Lagrangian L_0 by an exponential factor, yielding the *dissipative Lagrangian*:

$$L_\beta^{\text{diss}}(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, \boldsymbol{x}_t, \boldsymbol{y}_t) := \exp(\zeta t) \cdot L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, \boldsymbol{x}_t) + \beta\, c(\boldsymbol{s}_t, \boldsymbol{y}_t), \tag{17}$$

where $\zeta > 0$ is a scalar damping coefficient. The exponential factor $\exp(\zeta t)$ acts as an integrating factor that introduces dissipation into the dynamics while maintaining the variational structure needed for gradient estimation.

Dissipative gradient estimator. We now present the dissipative counterpart of theorem 3. The structure remains similar, but with additional terms arising from the exponential time-weighting.

Theorem 5 (Dissipative LEP with PFVP). *Let $t \mapsto \boldsymbol{s}_{\leftarrow,t}^{\beta}(\boldsymbol{\theta})$ denote the solution to the dissipative Euler–Lagrange equation:*

$$\text{EL}_{\text{diss}}(t, \boldsymbol{\theta}, \beta) := \partial_{\boldsymbol{s}} L_0 - d_t \partial_{\dot{\boldsymbol{s}}} L_0 - \zeta\, \partial_{\dot{\boldsymbol{s}}} L_0 + \beta \exp(-\zeta t)\, \partial_{\boldsymbol{s}} c = 0, \tag{18}$$

with PFVP boundary conditions. Then the gradient of the objective functional is given by:

$$\mathrm{d}_{\boldsymbol{\theta}}\mathcal{C}\left[\boldsymbol{s}^{0}_{\leftarrow}(\boldsymbol{\theta})\right] = \lim_{\beta\to 0}\Delta_{\mathrm{PFVP}}\left(\beta,\boldsymbol{\alpha}_0(\boldsymbol{\theta}),\boldsymbol{\gamma}_0(\boldsymbol{\theta})\right),$$

where the dissipative PFVP gradient estimator is:

$$\begin{aligned}\Delta_{\mathrm{PFVP}}(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0) := \frac{1}{\beta}\Bigg[&\int_0^T \exp(\zeta t)\cdot\left(\partial_{\boldsymbol{\theta}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,t},\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,t},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^{0}_{\leftarrow,t},\dot{\boldsymbol{s}}^{0}_{\leftarrow,t},\boldsymbol{\theta}\right)\right)\mathrm{d}t\\&+\left(\mathrm{d}_{\boldsymbol{\theta}}\partial_{\boldsymbol{s}}L_0(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta})\right)^{\top}\left(\boldsymbol{s}^{\beta}_{\leftarrow,0}-\boldsymbol{\alpha}_0\right)\\&-\left(\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0\right)^{\top}\left(\partial_{\boldsymbol{s}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,0},\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,0},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{s}}L_0(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta})\right)\Bigg],\end{aligned} \tag{19}$$

with $\boldsymbol{\alpha}_0=\boldsymbol{\alpha}_0(\boldsymbol{\theta})$ and $\boldsymbol{\gamma}_0=\boldsymbol{\gamma}_0(\boldsymbol{\theta})$ the initial conditions. The blue integral term weights the Lagrangian difference by the exponential factor, the red terms involve parameter derivatives of initial conditions, and the green terms measure state and momentum differences at the initial time.

Proof. See appendix L. □

Interpretation: dissipative terms. Compared to the conservative case, the dissipative formulation introduces a new exponentially-weighted term (shown in orange) in both the Euler–Lagrange equation and the gradient estimator:

- In the **Euler–Lagrange equation** (18): The term $-\zeta\,\partial_{\dot{\boldsymbol{s}}}L_0$ introduces friction-like damping, while the cost term acquires a down-weighting factor $\exp(-\zeta t)$ that reduces nudging strength at later times.
- In the **gradient estimator** (19): The integral term is weighted by $\exp(\zeta t)$, emphasizing later time steps. This reflects that dissipative dynamics progressively 'forget' early information, so gradients appropriately emphasize recent observations.
- For the **free phase** ($\beta=0$): The Euler–Lagrange equation reduces to $\partial_{\boldsymbol{s}}L_0-d_t\partial_{\dot{\boldsymbol{s}}}L_0-\zeta\,\partial_{\dot{\boldsymbol{s}}}L_0=0$, which is identical to applying the standard Euler–Lagrange equation to the exponentially-weighted Lagrangian $\exp(\zeta t)L_0$.

The boundary terms (red and green) remain identical to the conservative PFVP case.

Verification: energy dissipation. To confirm that the exponential integrating factor introduces a dissipative system with energy decay (rather than merely rescaling time), we analyze how energy evolves under the dissipative dynamics. We again consider the isolated system ($\boldsymbol{x}_t=0$) to cleanly isolate the effect of dissipation. We find that for a trajectory $t\mapsto\boldsymbol{s}_t$ satisfying the dissipative Euler–Lagrange equation (18) with $\beta=0$ and $\boldsymbol{x}_t=0$, the physical energy E (defined as in (16)) evolves as (see proposition 5 in appendix):

$$d_tE=-\zeta\,\dot{\boldsymbol{s}}_t^{\top}\,\partial_{\dot{\boldsymbol{s}}}L_0^{\mathrm{iso}}$$

In the special case with quadratic kinetic energy $E_{\mathrm{kin}}(\dot{\boldsymbol{s}}_t)=\frac{1}{2}\|\dot{\boldsymbol{s}}_t\|^2$, this reduces to:

$$d_tE=-\zeta\|\dot{\boldsymbol{s}}_t\|^2\leqslant 0$$

Since $\zeta>0$, energy is strictly dissipated whenever $\dot{\boldsymbol{s}}_t\neq 0$, confirming the physically expected behavior of a dissipative system.

6.3. Empirical validation

We now validate the dissipative LEP framework empirically. Our goals are twofold: first, to confirm that the exponential integrating-factor mechanism genuinely introduces dissipation, with energy transfers consistent with proposition 7; second, to verify that the dissipative LEP gradient estimator accurately

recovers parameter gradients, using autodiff/BPTT as a ground-truth baseline. We conduct these experiments on a system of $d = 6$ coupled damped harmonic oscillators (figure 6), extending the undamped system of section 2.2 with damping forces via the exponential integrating-factor introduced above.

System description. Consider a d-dimensional system of coupled harmonic oscillators with mass vector $\boldsymbol{m} \in \mathbb{R}^d_{>0}$, symmetric stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{d\times d}$, and scalar damping coefficient $\zeta > 0$. The damping vector is $\boldsymbol{\gamma} := \zeta \boldsymbol{m}$, making the damping force proportional to mass (for independent per-dimension damping coefficients γ_i decoupled from mass, see appendix R). An external input x_t drives the first oscillator, and the output is measured from the last oscillator $y_t = s_{d,t}$. The learnable parameters are $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{K}, \zeta\}$. We use fixed initial conditions $(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) = (\boldsymbol{\alpha}_0, \boldsymbol{0})$ (zero initial velocity), ensuring boundary terms vanish as explained in Remark 3. The Lagrangian of the undamped, input-driven system is:

$$L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t) = \frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t - \frac{1}{2}\boldsymbol{s}_t^\top \boldsymbol{K}\boldsymbol{s}_t - \boldsymbol{e}_1^\top \boldsymbol{s}_t x_t, \tag{20}$$

where $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^\top$ and $\odot$ denotes element-wise multiplication. The dissipative Lagrangian is $L^{\text{diss}}_\beta = \exp(\zeta t) \cdot L_0 + \beta\, c(\boldsymbol{s}_t, y_t)$ with cost $c(\boldsymbol{s}_t, y_t) = \frac{1}{2}(s_{d,t} - y_t)^2$.

Dynamics and gradient estimator: contrast with classical LEP. Table 4 summarizes the dissipative LEP equations. Both the free and nudged dynamics are integrated *forward in time* as IVPs. Compared to the classical (non-dissipative) LEP gradient estimator (theorem 3), the dissipative formulation introduces two key modifications:

1. **Sign-flipped damping in the nudge phase.** For the nudged phase, we apply the bouncing-backward kick (proposition 2): solving the PFVP backward in time from final conditions $(\boldsymbol{s}^\beta_T, \dot{\boldsymbol{s}}^\beta_T) = (\boldsymbol{s}^0_T, \dot{\boldsymbol{s}}^0_T)$ is equivalent to integrating forward with velocity-reversed initial conditions $(\boldsymbol{s}^0_T, -\dot{\boldsymbol{s}}^0_T)$ and, crucially, *sign-flipped damping* $(+\boldsymbol{\gamma} \to -\boldsymbol{\gamma})$. This sign flip reverses the energy flow: while the free phase dissipates energy, the nudged phase *pumps energy back* (see appendix M and proposition 6).
2. **Exponential weighting in nudging and learning rule.** The cost nudging term in the Euler–Lagrange equation (18) acquires a down-weighting factor $\exp(-\zeta t)$, and the gradient estimator (19) is weighted by $\exp(\zeta t)$. These exponential factors arise from the integrating-factor construction and are essential for correct gradient estimation.

With fixed initial conditions and PFVP final-condition matching, all boundary terms in theorem 5 vanish, leaving only the integral term that compares trajectories.

Classical LEP baseline. To isolate the importance of these modifications, we also evaluate a classical LEP baseline that correctly performs the sign-flipped damping $(+\boldsymbol{\gamma} \to -\boldsymbol{\gamma})$ during the nudge phase, but *omits both exponential factors*: it uses the standard nudging $\beta\, \partial_{\boldsymbol{s}} c$ instead of the down-weighted $\beta \exp(-\zeta t)\, \partial_{\boldsymbol{s}} c$ in the dynamics (18), and the standard unweighted integral $\int_0^T [\cdots]\, \mathrm{d}t$ instead of the exponentially-weighted $\int_0^T [\cdots] \exp(\zeta t)\, \mathrm{d}t$ in the gradient estimator (19). In other words, this baseline accounts for the dissipative dynamics (including the sign flip) but not for the effect of dissipation on the variational gradient formula.

Experimental observations. Figure 6 reports the outcomes of both validations. Regarding energy dynamics (Panels A–B), the free-phase energy decomposition shows kinetic and potential energy oscillating out of phase as energy transfers between modes, while the total energy increases over time due to the external input driving the system—kinetic energy starts at zero since $\dot{\boldsymbol{s}}_0 = \boldsymbol{0}$. The cumulative dissipation and input work are of comparable magnitude, confirming that dissipation is balanced by the energy injected by the external drive. The energy conservation relation $E(t) = E(0) + W_{\text{input}}(t) - D_{\text{diss}}(t)$ (proposition 7) is verified numerically, confirming that the integrating-factor mechanism produces physically consistent dissipative behavior. Regarding gradient accuracy (Panel C), the dissipative LEP gradient estimates for all parameters ($\boldsymbol{m}$, $\boldsymbol{K}$, ζ) closely match the autodiff/BPTT ground truth, with relative Euclidean distance below 0.10. In contrast, the classical LEP baseline (which performs the sign flip but omits the exponential factors) yields relative distances above 1 for $\boldsymbol{m}$ and $\boldsymbol{K}$, demonstrating that the sign-flipped damping alone is not sufficient—the exponential weighting in both the nudging and the learning rule is essential for correct gradient estimation in dissipative systems. Note that the classical LEP baseline produces an identically zero gradient for ζ: without the exponential weighting $e^{\zeta t}$ in the learning rule, the damping coefficient does not appear in the gradient estimator, so no LEP bar is shown for ζ in Panel C.

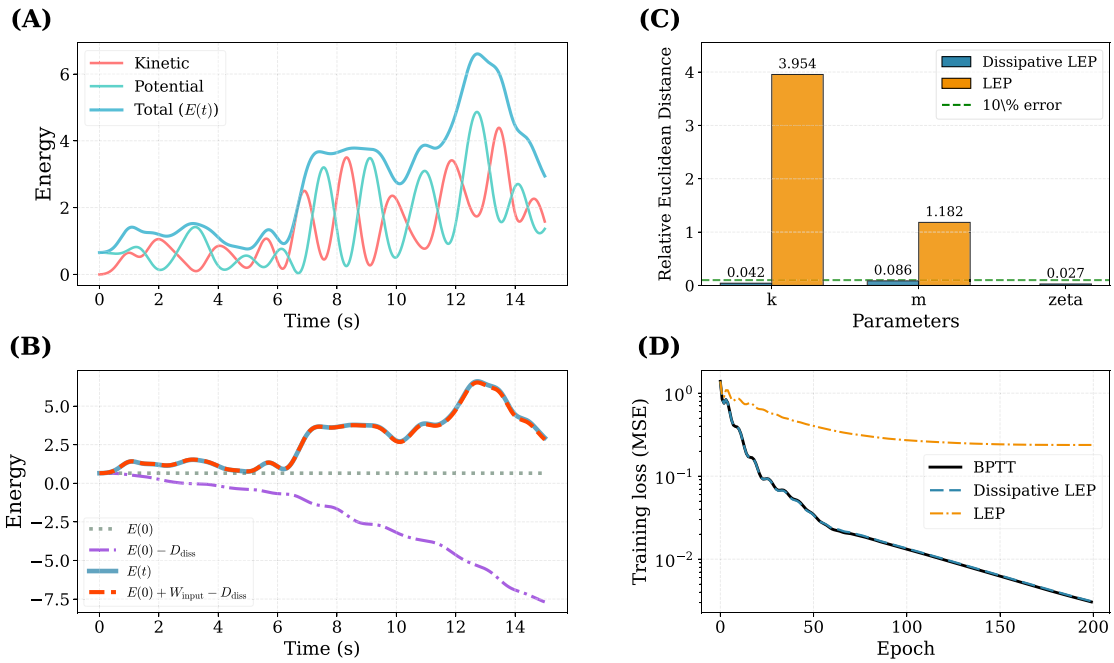


Figure 6. Empirical validation of dissipative LEP on $d = 6$ coupled damped harmonic oscillators. (a) Internal energy decomposition during the free phase: kinetic energy E_{kin}, internal potential U_{int}, and total energy E. (b) Cumulative energy balance: initial energy $E(0)$ (grey dashed), cumulative dissipation D_{diss} (purple), input work W_{input} (red), and total energy $E(t)$. (c) Relative Euclidean distance of gradient estimates to autodiff/BPTT for dissipative LEP (blue) and the classical LEP baseline (orange, which performs the sign-flipped damping but omits the exponential weighting in nudging and learning rule) across parameters $\boldsymbol{m}$, $\boldsymbol{K}$, and ζ. (d) Training loss (MSE) over 200 epochs of a teacher–student task on the *same* system, with all of $\boldsymbol{m}$, $\boldsymbol{K}$ and ζ trainable ($\zeta_0 = 0.1$ for the student, $\zeta = 0.2$ for the teacher). Dissipative LEP (blue, dashed) closely follows BPTT (black, solid) and recovers $\zeta \to 0.2$; the classical LEP baseline (orange), whose ζ-gradient is identically zero, plateaus roughly two orders of magnitude higher. Details in appendix P.

Table 4. Summary of dissipative LEP for coupled harmonic oscillators. Both the free and nudged phases are integrated forward in time as IVPs from their respective initial conditions; for the nudged phase, the backward PFVP is implemented through the bouncing-backward kick with reversed initial velocity and sign-flipped damping. The gradient estimator contains *only integral terms*—all boundary terms cancel due to fixed initial conditions and PFVP matching of final conditions. During inference (free phase), the energy $E(t) = E_{\text{kin}}(t) + U_{\text{int}}(t)$ (kinetic plus internal potential) evolves through two mechanisms: work done by the external input (force × velocity), and dissipation (proportional to $\boldsymbol{\gamma} \cdot \dot{\boldsymbol{s}}_t^2$, always removes energy).

Phase	Dynamics (IVP)	Time	Initial Conditions
Free ($\beta = 0$)	$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t^0 + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t^0 + \boldsymbol{K}\boldsymbol{s}_t^0 = -x_t \boldsymbol{e}_1$	$t \in [0, T]$	$(\boldsymbol{s}_0^0, \dot{\boldsymbol{s}}_0^0) = (\boldsymbol{\alpha}_0, \boldsymbol{0})$
Nudged ($\beta > 0$)	$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_{t'}^\beta - \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_{t'}^\beta + \boldsymbol{K}\boldsymbol{s}_{t'}^\beta = -x_{T-t'} \boldsymbol{e}_1 - \beta e^{-\zeta(T-t')} \boldsymbol{e}_d (s_{d,t'}^\beta - y_{T-t'})$	$t' \in [0, T]$	$(\boldsymbol{s}_0^\beta, \dot{\boldsymbol{s}}_0^\beta) = (\boldsymbol{s}_T^0, -\dot{\boldsymbol{s}}_T^0)$

Gradient estimator: $\mathrm{d}_{\boldsymbol{\theta}} \mathcal{C}[\boldsymbol{s}^0(\boldsymbol{\theta})] = \lim_{\beta \to 0} \frac{1}{\beta} \int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}, x_t) - \partial_{\boldsymbol{\theta}} L_0(\boldsymbol{s}_t^0, \dot{\boldsymbol{s}}_t^0, \boldsymbol{\theta}, x_t) \right] \exp(\zeta t)\, \mathrm{d}t$

with $\partial_{m_i} L_0 = \frac{1}{2} \dot{s}_{i,t}^2$, $\quad \partial_{\boldsymbol{K}} L_0 = -\frac{1}{2} \boldsymbol{s}_t \boldsymbol{s}_t^\top$, $\quad e^{-\zeta t} \partial_\zeta [e^{\zeta t} L_0] = t \cdot L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t)$

Energy definition: $E(t) = \underbrace{\frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t}_{E_{\text{kin}}(t)} + \underbrace{\frac{1}{2} \boldsymbol{s}_t^\top \boldsymbol{K} \boldsymbol{s}_t}_{U_{\text{int}}(t)}$

Energy transfer during inference (free phase) (proposition 7): $E(t) = E(0) + W_{\text{input}}(t) - D_{\text{diss}}(t)$

- **Input work:** $W_{\text{input}}(t) = -\int_0^t \dot{s}_{1,\tau}\, x_\tau\, \mathrm{d}\tau$ (power = force × velocity; can inject or extract energy)
- **Dissipation:** $D_{\text{diss}}(t) = \int_0^t \boldsymbol{\gamma} \cdot \dot{\boldsymbol{s}}_\tau^2\, \mathrm{d}\tau \geqslant 0$ (always removes energy)

Beyond this single-point gradient check, Panel D reports an end-to-end *training* run on the same system. In this teacher–student task, the student is initialized with parameters that differ from the teacher's—in particular, its damping coefficient is half the teacher's ($\zeta_0 = 0.1$ vs $\zeta = 0.2$)—and all of $\boldsymbol{m}$, $\boldsymbol{K}$ and ζ are trained. Over the 200 training epochs, dissipative LEP closely follows BPTT, reducing the loss by more than two orders of magnitude and recovering the teacher's damping coefficient $\zeta \to 0.2$; the small residual gap is consistent with the $O(\beta)$ bias of the finite-nudging estimator ($\beta = 0.01$). The classical LEP baseline initially reduces the loss through its potential-parameter updates, but its gradient with respect to ζ is identically zero: it cannot learn the damping coefficient and plateaus roughly two orders of magnitude above dissipative LEP. Full experimental details are given in appendix P.

Physical realizability of the sign-switched dissipation. The dissipation-specific control required by dissipative LEP is narrower than arbitrary control: within each phase the damping coefficient is fixed, and at $t = T$ it is reversed once from $+\gamma$ to $-\gamma$ with the same magnitude. Realizing this operation is substrate-dependent. Appendix N presents a concrete feasibility construction for the coupled-oscillator system (figure 7), in which the sign reversal is implemented by switching between matched positive and negative resistances; it also discusses the required velocity reversal, calibration, and parasitic losses. This construction is not a substrate-independent implementation, and the required operations must be assessed separately for other platforms.

Comparison with other approaches. Kendall *et al* [Ken21] proposed using fractional calculus to extend Lagrangian systems to dissipative dynamics, building on earlier work [Rie96]. While promising, this approach is limited to non-standard fractional dissipative elements, and they implicitly assumed fixed boundary conditions (equivalent to CBPVP), which would need to be reformulated as a PFVP for practical use. Another approach is to use periodic systems driven by periodic inputs [BH25], which also simplifies the boundary terms at the cost of restricting applicability to periodic systems (there is no need to match final conditions, but the input must be repeated multiple times). [Mas25] also proposed leveraging periodic systems (rather than the bouncing-backward kick used in our work), but introduced dissipation via the same exponential integrating factor as ours. Interestingly, all these approaches start from the mathematically guiding Lagrangian framework (see section 2.2). Finally, a method to simulate dissipativity within a non-dissipative system was proposed by [LPM23], where part of the system serves as an ancilla (an auxiliary subsystem that preserves information to maintain reversibility, a concept from reversible computing) for task-irrelevant information, yet can still be exploited for leveraging bouncing-backward kick.

7. Discussions and future works

Summary. This work sets out to address two questions.

(a) The first is whether *EP can be generalised to design efficient and practically-implementable learning algorithms for time-varying inputs and outputs*. We show that it can, through LEP that extends EP's variational principles from steady states to entire physical trajectories, provided that the boundary conditions are chosen carefully.

(b) The second question is *how HEL algorithms relate to this generalised EP* framework. We show that RHEL is a special case of LEP obtained by combining the PFVP boundary conditions with the Legendre transformation.

A central finding is that *the choice of boundary conditions has a decisive impact on whether the resulting learning algorithm is practical*. We show that the most natural choices lead to a trade-off between tractability of the gradient estimator and tractability of the trajectory computation. On one hand, the CIVP yields causal, easy-to-simulate trajectories but introduces boundary residual terms that are hard to compute and requires explicit backward passes. On the other hand, the CBPVP eliminates these residuals but imposes non-causal boundary conditions that require an iterative boundary value solver. The PFVP, combined with time-reversibility, resolves this trade-off: it eliminates boundary residuals entirely while preserving causal, forward-only, streaming computation with no iterative solver overhead.

By combining this PFVP formulation with the Legendre transformation, we establish that RHEL is a special case of LEP. This reveals that RHEL's distinctive properties, namely local learning rules, forward-only computation, and the 'bouncing-backward' echo phase, are not artifacts of Hamiltonian mechanics but arise naturally from the underlying variational structure.

Finally, we show that the variational framework of LEP provides guiding principles to extend these algorithms beyond conservative systems. By introducing an exponential integrating factor in the Lagrangian, dissipative dynamics can be accommodated within the PFVP framework provided the sign of the damping can be flipped during the echo phase. The variational derivation prescribes both the correct exponential weighting in the nudging and in the learning rule. Empirical validation on coupled damped harmonic oscillators confirms that this dissipative LEP gradient estimator accurately recovers BPTT gradients, and that omitting the prescribed weighting leads to incorrect gradients even when the sign-flipped damping is correctly applied.

The circuit of appendix N is a feasibility construction for the coupled-oscillator substrate, and not a substrate-independent implementation of dissipative LEP. For any other target substrate, realizability of the required operations must be assessed case by case.

Relation to BPTT. LEP and RHEL change how the loss gradient is evaluated, not the loss gradient itself: in the limit $\beta \to 0$, their estimators recover the gradient computed by BPTT. We therefore do not expect or claim an improvement in optimization quality over BPTT. Nor do we claim an improvement over RHEL, because the equivalence established above identifies RHEL as the Hamiltonian realization of PFVP/LEP. More broadly, our claims concern the gradient computation only, never the learning process built on top of it: the choice of optimizer, learning rate, or schedule is orthogonal to the estimator, exactly as in classical deep learning, and once the estimator recovers the BPTT gradient this standard toolbox applies verbatim. The training experiments of this paper (figure 6(d) and appendix Q) are accordingly end-to-end consistency checks of the estimators inside a complete optimization loop, not claims of a novel learning behavior: the present paper is primarily theoretical. Empirical evidence at larger scale exists in companion work: RHEL has been used to train multi-layer Hamiltonian state-space models (up to six stacked Hamiltonian recurrent blocks with hidden dimension 64 and state-space dimension 256) on six long-range UEA classification benchmarks and on a $\sim$50k-step regression task, matching BPTT performance across all models and tasks [PE25]; by the equivalence of theorem 4, this evidence carries over to the LEP/RHEL gradient-computation principle established here.

Limitations and future directions. First, the PFVP formulation still requires an echo phase, i.e. a second forward pass that can only begin after the free phase completes, making the algorithm inherently offline in the sense that gradients are not available during inference. Developing an online variant that eliminates this echo phase would bring LEP closer to Real-Time Recurrent Learning [WZ89], potentially offering more efficient alternatives to its notoriously high computational cost. Second, the elimination of boundary residuals relies on time-reversibility, which restricts applicability to conservative (or sign-controllable dissipative) systems. Extending the PFVP formulation beyond time-reversible systems, or identifying weaker sufficient conditions for boundary residual cancellation, would broaden its applicability. Third, while RHEL has been tested at larger scale on state-space models [PE25], neither LEP nor dissipative LEP have been validated on real physical systems. Finally, the present framework addresses deterministic dynamics derived from a Lagrangian: stochastic dynamics, as well as model-specific architectural mechanisms such as attention, lie beyond its current scope. These advances would further solidify the theoretical foundation for physics-based learning algorithms that unify inference and training within single physical systems, offering promising alternatives to conventional digital computing paradigms for future neuromorphic and analog computing architectures.

Acknowledgments

GP and DB acknowledge the CHIST-ERA grant for the CausalXRL project (CHIST-ERA-19-XAI-002) by L'Agence Nationale de la Recherche (ANR), France (grant reference ANR-21-CHR4-0007), and the ANR JCJC REPUBLIC project (ANR-22-CE23-0003-01) for supporting this work. GP acknowledges Herbert Jaeger for supporting the continuation of this research. ME started the project as an independent researcher before joining Google DeepMind and acknowledges Albert Cohen and James Laudon for their support and review of the manuscript.

Data availability statement

The data and code that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/guillaumepourcel/Lagrangian-Equilibrium-Propagation [ioppcitedata]. Data will be available from 30 December 2026.

Appendix A. Derivative and shape conventions

Throughout the paper, we adopt a fixed coordinate-based matrix convention. State variables are represented as column vectors $s \in \mathbb{R}^{d_s}$ and parameters as column vectors $\theta \in \mathbb{R}^{d_\theta}$.

Gradients with respect to state variables (including velocities) are column vectors:

$$\partial_s L \in \mathbb{R}^{d_s}, \qquad \partial_{\dot{s}} L \in \mathbb{R}^{d_s}.$$

Jacobians with respect to parameters are matrices acting on parameter variations:

$$\partial_\theta s \in \mathbb{R}^{d_s \times d_\theta}, \qquad \partial^2_{\theta,\dot{s}} L \in \mathbb{R}^{d_s \times d_\theta}.$$

Total derivatives vs partial derivatives. We distinguish between *partial derivatives* and *total derivatives*:

- ∂_θ denotes the *partial derivative* with respect to θ, holding all other explicit arguments fixed.
- d_θ denotes the *total derivative* with respect to θ, accounting for both explicit and implicit dependencies through the chain rule.

For example, $d_\theta \partial_{\dot{s}} L$ is a total derivative (Jacobian) with shape $\mathbb{R}^{d_s \times d_\theta}$ that accounts for how $\partial_{\dot{s}} L$ changes with θ through all dependencies, including implicit ones through the state variables.

Scalar or parameter-wise quantities are obtained via standard matrix multiplication. In particular, for any $v \in \mathbb{R}^{d_s}$,

$$\left(\partial^2_{\theta,\dot{s}} L\right)^\top v \in \mathbb{R}^{d_\theta},$$

where the transpose denotes the usual matrix transpose and ensures dimensional consistency. Equivalently, this corresponds componentwise to

$$\left[\left(\partial^2_{\theta,\dot{s}} L\right)^\top v\right]_j = \sum_{i=1}^{d_s} \partial^2_{\theta_j,\dot{s}_i} L\, v_i.$$

All transposes appearing in the paper are genuine matrix transposes introduced to make matrix products well-defined; no implicit row/column conventions are assumed. Under this convention, all gradient expressions and boundary residual terms are dimensionally consistent.

Functional (variational) derivative. We denote by $\delta_s A$ the *functional derivative* (or *variational derivative*) of a functional $A[s]$ with respect to the trajectory s. It is defined implicitly through the directional derivative: for any smooth variation η, $\delta_s A \cdot \eta := d_\epsilon|_{\epsilon=0} A[s + \epsilon\eta]$. Informally, $\delta_s A$ is the infinite-dimensional analogue of a gradient: it captures how A responds to infinitesimal deformations of the trajectory.

Appendix B. Glossary

Table 5 summarizes the different boundary condition formulations for Lagrangian Equilibrium Propagation (LEP) discussed in this paper. Each formulation defines how trajectories $\boldsymbol{s}_t^\beta(\boldsymbol{\theta})$ are specified through boundary conditions, leading to distinct computational properties and practical implications.

Table 5. Summary of LEP Boundary Condition Formulations. Comparison of the three main boundary condition formulations for Lagrangian Equilibrium Propagation.

Acronym	Definition	section
CIVP	**Constant Initial Value Problem:** All trajectories share fixed initial conditions independent of $\boldsymbol{\theta}$ and β. Defined by: $$\forall t \in [0,T] \quad t \mapsto \boldsymbol{s}^{\beta}_{\rightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)) \text{ satisfies: } \begin{cases} \mathrm{EL}(t,\boldsymbol{\theta},\beta)=0 \\ \boldsymbol{s}^{\beta}_{\rightarrow,0}(\boldsymbol{\theta})=\boldsymbol{\alpha}_0 \\ \dot{\boldsymbol{s}}^{\beta}_{\rightarrow,0}(\boldsymbol{\theta})=\boldsymbol{\gamma}_0 \end{cases}$$ Causal boundary conditions: forward integration from $t=0$. Suffers from intractable boundary residuals.	3.3.2
CBPVP	**Constant Boundary Position Value Problem:** All trajectories satisfy fixed position boundary conditions at both endpoints, independent of $\boldsymbol{\theta}$ and β. Defined by: $$\forall t \in [0,T], \quad t \mapsto \boldsymbol{s}^{\beta}_{\leftrightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_T)) \text{ satisfies: } \begin{cases} \mathrm{EL}(t,\boldsymbol{\theta},\beta)=0 \\ \boldsymbol{s}^{\beta}_{\leftrightarrow,0}(\boldsymbol{\theta})=\boldsymbol{\alpha}_0 \\ \boldsymbol{s}^{\beta}_{\leftrightarrow,T}(\boldsymbol{\theta})=\boldsymbol{\alpha}_T \end{cases}$$ Non-causal boundary conditions: requires solving a two-point boundary value problem (BVP). Eliminates boundary residuals but computationally expensive.	3.3.1
PFVP	**Parametric Final Value Problem:** Final boundary conditions depend on parameters $\boldsymbol{\theta}$. Defined by: $$\forall t \in [0,T] \quad t \mapsto \boldsymbol{s}^{\beta}_{\leftarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T(\boldsymbol{\theta}),\boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \text{ satisfies: } \begin{cases} \mathrm{EL}_r(t,\boldsymbol{\theta},\beta)=0 \\ \boldsymbol{s}^{\beta}_{\leftarrow,T}(\boldsymbol{\theta})=\boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ \dot{\boldsymbol{s}}^{\beta}_{\leftarrow,T}(\boldsymbol{\theta})=\boldsymbol{\gamma}_T(\boldsymbol{\theta}) \end{cases}$$ where parametric boundaries are derived from CIVP free phase: $\boldsymbol{\alpha}_T(\boldsymbol{\theta})=\boldsymbol{s}^{0}_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$ and $\boldsymbol{\gamma}_T(\boldsymbol{\theta})=\dot{\boldsymbol{s}}^{0}_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$.	

Appendix C. Preparatory results

Remark 4 (Regularity and uniqueness of solutions). Several proofs in this paper invoke uniqueness of solutions to initial value problems. The classical sufficient condition (the uniqueness part of the Picard–Lindelöf theorem) is that the Euler–Lagrange equation, once rewritten as a first-order system $\dot{\boldsymbol{z}}=\boldsymbol{f}(t,\boldsymbol{z})$, has a right-hand side $\boldsymbol{f}$ that is locally Lipschitz in $\boldsymbol{z}$. Two ingredients are needed:

1. **Mass-matrix invertibility.** The Hessian $\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}}L$ must be invertible so that $\ddot{\boldsymbol{s}}$ can be expressed as a function of $(\boldsymbol{s},\dot{\boldsymbol{s}},t)$. This is already required for the Legendre transform (proposition 1).
2. **Local Lipschitz continuity of the resulting right-hand side.** This property must be checked for the resulting first-order system; it is verified directly below for the models considered in this paper.

Verification for the models of table 1. For all three models the mass matrix is either the identity (**UniCORNN**, **LinOSS**) or $\mathrm{diag}(\boldsymbol{\tau})$ with $\tau_i > 0$ (**Hopfield**), hence invertible. Moreover, the right-hand sides are in fact *globally* Lipschitz: **LinOSS** is linear in $\boldsymbol{z}$; **UniCORNN** involves tanh, which is globally Lipschitz (derivative bounded by 1) composed with a linear map; **Hopfield** involves products of tanh and $\tanh'$, both globally bounded, composed with linear maps—the Lipschitz constant depends on $\|W\|$ but remains finite for any fixed W. Global Lipschitz continuity ensures that solutions exist and are unique on any interval $[0,T]$.

When the Hessian condition fails. For the Hopfield-inspired model, if $\tau_i = 0$, then the velocity Hessian is singular and $p_i = \tau_i \dot{s}_i = 0$ independently of $\dot{s}_i$. Consequently, $\dot{s}_i$ cannot be recovered from p_i, and the momentum–velocity relation of proposition 1 is not invertible for that coordinate. The Euler–Lagrange equation also loses its $\ddot{s}_i$ term. Therefore, the Lagrangian-to-Hamiltonian correspondence required by theorem 4 does not apply. This illustrates why the Hopfield-inspired model assumes $\tau_i > 0$ for every i.

Lemma 2 (Odd derivative property of reversible Lagrangian). *For a reversible Lagrangian L_β that satisfies $L_\beta(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta}) = L_\beta(\boldsymbol{s},-\dot{\boldsymbol{s}},\boldsymbol{\theta})$, the derivative with respect to $\dot{\boldsymbol{s}}$ satisfies:*

$$\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s},-\dot{\boldsymbol{s}},\boldsymbol{\theta}\right) = -\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta}\right).$$

Proof. Since the Lagrangian L_β is reversible, it satisfies

$$L_\beta\left(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta}\right) = L_\beta\left(\boldsymbol{s},-\dot{\boldsymbol{s}},\boldsymbol{\theta}\right),$$

i.e. it is even in $\dot{\boldsymbol{s}}$. Consequently, its derivative with respect to $\dot{\boldsymbol{s}}$ is odd, yielding the stated result. □

Proposition 3 (Least Action principle for parametrized perturbations). *Let $A[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}] = \int_0^T L(t,\boldsymbol{\theta},\boldsymbol{s}_t(\boldsymbol{\theta}),\dot{\boldsymbol{s}}_t(\boldsymbol{\theta}))\mathrm{d}t$ be a scalar functional of an arbitrary function $\boldsymbol{s}(\boldsymbol{\theta})$ that depends on some parameter vector $\boldsymbol{\theta}\in\mathbb{R}^p$. Further, A also has an explicit dependence on $\boldsymbol{\theta}$. Here, $\boldsymbol{\theta}$ is a non-time-varying parameter.*

If $\boldsymbol{s}(\boldsymbol{\theta})$ satisfies the Euler–Lagrange equations $\partial_{\boldsymbol{s}}L - d_t\partial_{\dot{\boldsymbol{s}}}L = 0$, then the implicit variation of A through $\boldsymbol{\theta}$ via $\boldsymbol{s}$ reduces to boundary terms:

$$\delta_{\boldsymbol{s}} A\left[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}\right]\delta_{\boldsymbol{\theta}}\boldsymbol{s}(\boldsymbol{\theta}) = \left[\left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_t(\boldsymbol{\theta})\right)^\top\cdot\partial_{\dot{\boldsymbol{s}}}L\left(t,\boldsymbol{\theta},\boldsymbol{s}_t(\boldsymbol{\theta}),\dot{\boldsymbol{s}}_t(\boldsymbol{\theta})\right)\right]_0^T.$$

Implicit variation (definition): *The implicit variation along each component θ_i is defined as the change in A due to θ_i acting only through $\boldsymbol{s}$, with the explicit $\boldsymbol{\theta}$-dependence of A held fixed:*

$$\delta_{\boldsymbol{s}} A\left[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}\right]\delta_{\theta_i}\boldsymbol{s}(\boldsymbol{\theta}) := d_\epsilon|_{\epsilon=0} A\left[\boldsymbol{s}(\boldsymbol{\theta}+\epsilon e_i),\boldsymbol{\theta}\right]. \tag{21}$$

Notation: *Here e_i denotes the ith canonical basis vector in $\mathbb{R}^p$ (the parameter space of $\boldsymbol{\theta}$). Each $\delta_{\boldsymbol{s}}A\delta_{\theta_i}\boldsymbol{s}$ is a scalar, and the full vector form $\delta_{\boldsymbol{s}}A[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}]\delta_{\boldsymbol{\theta}}\boldsymbol{s}(\boldsymbol{\theta})\in\mathbb{R}^p$ is the vector obtained by concatenation:* $\delta_{\boldsymbol{s}}A\delta_{\boldsymbol{\theta}}\boldsymbol{s} = \left(\delta_{\boldsymbol{s}}A\delta_{\theta_1}\boldsymbol{s},\ldots,\delta_{\boldsymbol{s}}A\delta_{\theta_p}\boldsymbol{s}\right)^\top$.

Proof. We prove the result component-wise using lemma 1. Fix a component θ_i and consider the perturbation $\boldsymbol{\eta}(\epsilon) := \boldsymbol{s}(\boldsymbol{\theta}+\epsilon e_i) - \boldsymbol{s}(\boldsymbol{\theta})$, which satisfies $\boldsymbol{\eta}(0) = \mathbf{0}$ and $\partial_\epsilon|_{\epsilon=0}\boldsymbol{\eta}_t(\epsilon) = \partial_{\theta_i}\boldsymbol{s}_t(\boldsymbol{\theta})$. From definition (21):

$$\delta_{\boldsymbol{s}} A\left[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}\right]\delta_{\theta_i}\boldsymbol{s}(\boldsymbol{\theta}) = d_\epsilon|_{\epsilon=0} A\left[\boldsymbol{s}(\boldsymbol{\theta}+\epsilon e_i),\boldsymbol{\theta}\right] = d_\epsilon|_{\epsilon=0} A\left[\boldsymbol{s}(\boldsymbol{\theta})+\boldsymbol{\eta}(\epsilon),\boldsymbol{\theta}\right].$$

Applying lemma 1 to parametric perturbations (see note after the Lemma and proof in appendix E), since $\boldsymbol{s}(\boldsymbol{\theta})$ satisfies the Euler–Lagrange equations:

$$\delta_{\boldsymbol{s}} A\left[\boldsymbol{s}(\boldsymbol{\theta}),\boldsymbol{\theta}\right]\delta_{\theta_i}\boldsymbol{s}(\boldsymbol{\theta}) = \left[\left(\partial_{\theta_i}\boldsymbol{s}_t(\boldsymbol{\theta})\right)^\top\cdot\partial_{\dot{\boldsymbol{s}}}L\left(t,\boldsymbol{\theta},\boldsymbol{s}_t(\boldsymbol{\theta}),\dot{\boldsymbol{s}}_t(\boldsymbol{\theta})\right)\right]_0^T.$$

Concatenating over all components $i = 1,\ldots,p$ yields the full vector result. The same analysis applies with respect to any other parameter (e.g. β). □

Proposition 4 (Equivalence between CIVP and PFVP). *The PFVP free solution that terminates at the final state of the corresponding CIVP free solution coincides with that CIVP free solution. Namely, for any $(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)$ and all $t\in[0,T]$,*

$$\boldsymbol{s}^0_{\leftarrow,t}\left(\boldsymbol{\theta},\left(\boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)),\dot{\boldsymbol{s}}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))\right)\right) = \boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)).$$

Proof. Define the terminal state of the CIVP trajectory by

$$(\boldsymbol{\alpha}_T,\boldsymbol{\gamma}_T) := \left(\boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0)),\dot{\boldsymbol{s}}^0_{\rightarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))\right).$$

By definition of the PFVP (Definition 13), the trajectory $t\mapsto\boldsymbol{s}^0_{\leftarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T,\boldsymbol{\gamma}_T))$ is a solution of the same Euler–Lagrange dynamics as $t\mapsto\boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta},(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0))$ and satisfies the terminal condition

$$\left(\boldsymbol{s}^0_{\leftarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T,\boldsymbol{\gamma}_T)),\dot{\boldsymbol{s}}^0_{\leftarrow,T}(\boldsymbol{\theta},(\boldsymbol{\alpha}_T,\boldsymbol{\gamma}_T))\right) = (\boldsymbol{\alpha}_T,\boldsymbol{\gamma}_T).$$

Therefore, the two trajectories share the same state at time T while solving the same ODE. By uniqueness of solutions (Remark 4), they must coincide on $[0,T]$, which proves the claim. □

Lemma 3 (IVP-FVP equivalence for reversible Hamiltonian systems). *For a reversible Hamiltonian system, the IVP solution starting from momentum-flipped initial conditions is equivalent to the time-reversed FVP solution:*

$$\forall t \in [0,T] \quad \boldsymbol{\Phi}_{IVP,t}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_z \boldsymbol{\lambda}_0) = \boldsymbol{\Sigma}_z \boldsymbol{\Phi}_{FVP,T-t}(\boldsymbol{\theta}, \boldsymbol{\lambda}_0),$$

where $\boldsymbol{\Sigma}_z = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}$ *is the momentum-flipping operator.*

Proof by reference and relation to Proposition 2. (1) Analogy with Proposition 2. Proposition 2 proves reversibility of the time-reversible PFVP solution in the Lagrangian formulation: the FVP can be reduced to an IVP by applying the time-reversal symmetry (reverse time and flip the time-odd variable, namely the velocity). The present lemma (lemma 3) is the Hamiltonian analogue of that statement: in Hamiltonian coordinates, time reversal acts by leaving the position unchanged and flipping the conjugate momentum. Hence the same 'FVP ↔ IVP' conversion holds, with velocity flip replaced by momentum flip.

(2) Proof is contained in the RHEL paper. This reversibility property is established in the RHEL paper [PE25], appendix A.1. Specifically, lemma A.3 in [PE25] proves time-reversal invariance of the Hamiltonian dynamics under the momentum-flip involution (with the appropriate time-reversal of the forcing/input), and corollary A.3 in [PE25] deduces the corresponding reversibility/echo (trajectory retracing) property. The present lemma is exactly the specialization of these results to the IVP–FVP equivalence stated here. □

Lemma 4 (Parameter-gradient relation under Legendre transform). *Let* $t \mapsto \boldsymbol{\Phi}_t = (\boldsymbol{s}_t, \boldsymbol{p}_t)$ *be a solution of Hamilton's equations with Hamiltonian* $H(\boldsymbol{\Phi}_t, \boldsymbol{\theta})$. *Let* $t \mapsto (\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)$ *be the associated Lagrangian trajectory defined through the backward Legendre transform (proposition 1). Then:*

$$\partial_{\boldsymbol{\theta}} H(\boldsymbol{\Phi}_t, \boldsymbol{\theta}) = -\partial_{\boldsymbol{\theta}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}).$$

Proof. The momentum is defined by the (forward) Legendre transform (proposition 1(a)):

$$\boldsymbol{p}_t := \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}). \tag{22}$$

The Hamiltonian is constructed as:

$$H(\boldsymbol{\Phi}_t, \boldsymbol{\theta}) := \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}),$$

where $\dot{\boldsymbol{s}}_t$ is determined implicitly by $(\boldsymbol{s}_t, \boldsymbol{p}_t, \boldsymbol{\theta})$ through equation (22). In particular, when the Legendre transform is well-defined (i.e. $\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L$ is invertible), we may invert $\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta})$ to obtain a (local) implicit function $\dot{\boldsymbol{s}}_t = \dot{\boldsymbol{s}}(\boldsymbol{s}_t, \boldsymbol{p}_t, \boldsymbol{\theta})$. Thus $\dot{\boldsymbol{s}}_t$ is not an independent variable here: once $\boldsymbol{\Phi}_t = (\boldsymbol{s}_t, \boldsymbol{p}_t)$ is held fixed, the right-hand side is understood as a function of $(\boldsymbol{\Phi}_t, \boldsymbol{\theta})$ only.

Taking the derivative with respect to $\boldsymbol{\theta}$ (holding $\boldsymbol{\Phi}_t = (\boldsymbol{s}_t, \boldsymbol{p}_t)$ fixed):

$$\begin{aligned} \partial_{\boldsymbol{\theta}} H(\boldsymbol{\Phi}_t, \boldsymbol{\theta}) &= \partial_{\boldsymbol{\theta}} \left[\boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}) \right] \\ &= \boldsymbol{p}_t^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t - \partial_{\boldsymbol{\theta}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}) - (\partial_{\dot{\boldsymbol{s}}} L)^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t. \end{aligned}$$

Here, $\partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t$ represents the derivative of the implicit function $\dot{\boldsymbol{s}}_t(\boldsymbol{s}_t, \boldsymbol{p}_t, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. This derivative captures how the velocity $\dot{\boldsymbol{s}}_t$ changes with $\boldsymbol{\theta}$ while keeping the Hamiltonian state $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ fixed.

The key observation is that the first and third terms cancel:

$$\begin{aligned} \boldsymbol{p}_t^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t - (\partial_{\dot{\boldsymbol{s}}} L)^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t &= \boldsymbol{p}_t^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t - \boldsymbol{p}_t^\top \partial_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_t \quad \text{(using equation (22))} \\ &= 0. \end{aligned}$$

Therefore:

$$\partial_{\boldsymbol{\theta}} H(\boldsymbol{\Phi}_t, \boldsymbol{\theta}) = -\partial_{\boldsymbol{\theta}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}).$$

□

Lemma 5 (Independence of augmented Lagrangian and Hamiltonian derivatives). *Let* L_β *be the augmented Lagrangian defined in Equations (4) where the cost term c does not depend on* $\dot{\boldsymbol{s}}$ *or* $\boldsymbol{\theta}$. *Let* H_β *be the corresponding Hamiltonian obtained via Legendre transform. Then, for all* $(\boldsymbol{s}, \dot{\boldsymbol{s}}, \boldsymbol{\theta})$ *(or* $(\boldsymbol{\Phi}, \boldsymbol{\theta})$ *for Hamiltonian) and all* β*:*

1. $\partial_{\dot{\boldsymbol{s}}} L_\beta(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta}) = \partial_{\dot{\boldsymbol{s}}} L_0(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta})$ *(Lagrangian velocity derivative)*
2. $\partial_{\boldsymbol{\theta}} L_\beta(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} L_0(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta})$ *(Lagrangian parameter derivative)*
3. $\partial_{\boldsymbol{\theta}} H_\beta(\boldsymbol{\Phi},\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} H_0(\boldsymbol{\Phi},\boldsymbol{\theta})$ *(Hamiltonian parameter derivative)*

Note: The result also holds for the Hamiltonian momentum derivative: $\partial_{\boldsymbol{p}} H_\beta(\boldsymbol{s},\boldsymbol{p},\boldsymbol{\theta}) = \partial_{\boldsymbol{p}} H_0(\boldsymbol{s},\boldsymbol{p},\boldsymbol{\theta})$, though this property is not needed in this paper.

Proof. Since $L_\beta = L_0 + \beta c$ where c depends only on $\boldsymbol{s}$ (not on $\dot{\boldsymbol{s}}$ or $\boldsymbol{\theta}$), the first two properties follow immediately. For the third property, since H_β is obtained from L_β via Legendre transform and the transform preserves parameter derivatives (as shown in lemma 4), we have $\partial_{\boldsymbol{\theta}} H_\beta = -\partial_{\boldsymbol{\theta}} L_\beta = -\partial_{\boldsymbol{\theta}} L_0 = \partial_{\boldsymbol{\theta}} H_0$. □

Appendix D. Proof of proposition 1: Legendre transform

Proof. [Proof of proposition 1] We first justify the forward transform, then the backward one, and finally the equivalence of the Hessian conditions.

(a) Forward transform. Fix t and regard L as a function of $(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)$. Define

$$\boldsymbol{p}_t \;=\; \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)\,.$$

For fixed $\boldsymbol{s}_t$, the Jacobian of the map

$$\dot{\boldsymbol{s}}_t \mapsto \boldsymbol{p}_t$$

is precisely the Hessian $\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)$.

If

$$\det\left(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)\right) \neq 0,$$

then, by the inverse function theorem, this map is locally invertible: there exists a unique smooth function

$$\dot{\boldsymbol{s}}_t = \dot{\boldsymbol{s}}_t(\boldsymbol{s}_t,\boldsymbol{p}_t)$$

in a neighbourhood of $(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)$. We then define the Hamiltonian

$$H(\boldsymbol{s}_t,\boldsymbol{p}_t) \;=\; \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t(\boldsymbol{s}_t,\boldsymbol{p}_t) - L\left(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t(\boldsymbol{s}_t,\boldsymbol{p}_t)\right),$$

which yields the forward transform

$$(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t) \mapsto (\boldsymbol{s}_t,\boldsymbol{p}_t), \qquad \boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L, \quad H = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - L,$$

and is locally one-to-one with smooth inverse $(\boldsymbol{s}_t,\boldsymbol{p}_t) \mapsto (\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t)$.

(b) Backward transform. Conversely, fix t and regard H as a function of $(\boldsymbol{s}_t,\boldsymbol{p}_t)$, and define

$$\dot{\boldsymbol{s}}_t \;=\; \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t,\boldsymbol{p}_t)\,.$$

For fixed $\boldsymbol{s}_t$, the Jacobian of the map

$$\boldsymbol{p}_t \mapsto \dot{\boldsymbol{s}}_t$$

is the Hessian $\partial^2_{\boldsymbol{p},\boldsymbol{p}} H(\boldsymbol{s}_t,\boldsymbol{p}_t)$.

If

$$\det\left(\partial^2_{\boldsymbol{p},\boldsymbol{p}} H(\boldsymbol{s}_t,\boldsymbol{p}_t)\right) \neq 0,$$

then, again by the inverse function theorem, this map is locally invertible, so there exists a unique smooth function

$$\boldsymbol{p}_t = \boldsymbol{p}_t(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t),$$

and we can define the Lagrangian via

$$L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) \;=\; \boldsymbol{p}_t(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)^\top \dot{\boldsymbol{s}}_t - H(\boldsymbol{s}_t, \boldsymbol{p}_t(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t))\,,$$

which yields the backward transform

$$(\boldsymbol{s}_t, \boldsymbol{p}_t) \mapsto (\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)\,, \qquad \dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H, \quad L = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - H,$$

again locally one-to-one with smooth inverse.

Equivalence of Hessian conditions. Assume L and H are related by the Legendre transform as above. For fixed $\boldsymbol{s}_t$, we have

$$\boldsymbol{p}_t \;=\; \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)\,, \qquad \dot{\boldsymbol{s}}_t \;=\; \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t)\,.$$

Differentiate the first relation w.r.t. $\dot{\boldsymbol{s}}_t$:

$$\partial_{\dot{\boldsymbol{s}}_t} \boldsymbol{p}_t \;=\; \partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)\,.$$

Differentiate the second relation w.r.t. $\boldsymbol{p}_t$:

$$\partial_{\boldsymbol{p}_t} \dot{\boldsymbol{s}}_t \;=\; \partial^2_{\boldsymbol{p},\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t)\,.$$

Since the maps $\dot{\boldsymbol{s}}_t \mapsto \boldsymbol{p}_t$ and $\boldsymbol{p}_t \mapsto \dot{\boldsymbol{s}}_t$ are inverse to each other (for fixed $\boldsymbol{s}_t$), their Jacobians are matrix inverses:

$$\partial_{\boldsymbol{p}_t} \dot{\boldsymbol{s}}_t = (\partial_{\dot{\boldsymbol{s}}_t} \boldsymbol{p}_t)^{-1}\,.$$

Hence

$$\partial^2_{\boldsymbol{p},\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t) = \left(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)\right)^{-1}.$$

In particular,

$$\det\left(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L\right) \neq 0 \quad \Longleftrightarrow \quad \det\left(\partial^2_{\boldsymbol{p},\boldsymbol{p}} H\right) \neq 0,$$

so the forward and backward non-singularity conditions are equivalent, and the Legendre transform is invertible in both directions. □

Appendix E. Proof of lemma 1. Euler–Lagrange solutions and the action functional

Proof of lemma 1. We prove the result for a general smooth perturbation $\epsilon \mapsto \boldsymbol{\eta}(\epsilon)$ with $\boldsymbol{\eta}(0) = \mathbf{0}$ (as noted after lemma 1); the linear case $\boldsymbol{\eta}(\epsilon) = \epsilon\boldsymbol{\eta}$ follows by specialization.

Expanding the action functional along the perturbation:

$$A_\beta\left[\mathbf{s}^\beta + \boldsymbol{\eta}(\epsilon)\right] = \int_0^T L_\beta\left(\boldsymbol{s}_t^\beta + \boldsymbol{\eta}_t(\epsilon)\,,\, \dot{\boldsymbol{s}}_t^\beta + \dot{\boldsymbol{\eta}}_t(\epsilon)\,,\, \boldsymbol{\theta}\right) \mathrm{dt},$$

where $\dot{\boldsymbol{\eta}}_t(\epsilon) := d_t \boldsymbol{\eta}_t(\epsilon)$ denotes the time derivative at fixed ϵ. Differentiating with respect to ϵ and evaluating at $\epsilon = 0$, the chain rule gives:

$$d_\epsilon|_{\epsilon=0} A_\beta\left[\mathbf{s}^\beta + \boldsymbol{\eta}(\epsilon)\right] = \int_0^T \left[\left(\partial_\epsilon|_{\epsilon=0} \boldsymbol{\eta}_t(\epsilon)\right)^\top \partial_{\boldsymbol{s}} L_\beta\left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}\right) + \left(d_t\, \partial_\epsilon|_{\epsilon=0} \boldsymbol{\eta}_t(\epsilon)\right)^\top \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}\right)\right] \mathrm{dt}.$$

Here we used $\boldsymbol{\eta}(0) = \mathbf{0}$ (i.e. $\boldsymbol{\eta}_t(0) = \mathbf{0}$ and $\dot{\boldsymbol{\eta}}_t(0) = \mathbf{0}$ for all t) to ensure that the partial derivatives of L_β are evaluated at the original trajectory $(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta)$. We also used the commutativity of ∂_ϵ and d_t (valid by smoothness) to write $\partial_\epsilon|_{\epsilon=0} \dot{\boldsymbol{\eta}}_t(\epsilon) = d_t\, \partial_\epsilon|_{\epsilon=0} \boldsymbol{\eta}_t(\epsilon)$. Applying integration by parts to the second term:

$$= \int_0^T \left(\partial_\epsilon|_{\epsilon=0} \boldsymbol{\eta}_t(\epsilon)\right)^\top \left[\partial_{\boldsymbol{s}} L_\beta - d_t \partial_{\dot{\boldsymbol{s}}} L_\beta\right] \mathrm{dt} + \left[\left(\partial_\epsilon|_{\epsilon=0} \boldsymbol{\eta}_t(\epsilon)\right)^\top \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}\right)\right]_0^T.$$

Since $\mathbf{s}^\beta$ satisfies the Euler–Lagrange equation $\mathrm{EL}(t,\boldsymbol{\theta},\beta) = \partial_{\mathbf{s}} L_\beta - d_t \partial_{\dot{\mathbf{s}}} L_\beta = 0$, the integral vanishes, yielding:

$$d_\epsilon|_{\epsilon=0} A_\beta\left[\mathbf{s}^\beta + \boldsymbol{\eta}(\epsilon)\right] = \left[\left(\partial_\epsilon|_{\epsilon=0}\boldsymbol{\eta}_t(\epsilon)\right)^\top \partial_{\dot{\mathbf{s}}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right)\right]_0^{\mathrm{T}}. \tag{23}$$

This establishes the general case for parametric perturbations (see note after lemma 1). For the linear perturbation $\boldsymbol{\eta}(\epsilon) = \epsilon\boldsymbol{\eta}$, we have $\partial_\epsilon|_{\epsilon=0}\boldsymbol{\eta}(\epsilon) = \boldsymbol{\eta}$, and equation (23) becomes:

$$\delta_{\mathbf{s}} A_\beta \cdot \boldsymbol{\eta} = d_\epsilon|_{\epsilon=0} A_\beta\left[\mathbf{s}^\beta + \epsilon\boldsymbol{\eta}\right] = \left[\boldsymbol{\eta}_t^\top \partial_{\dot{\mathbf{s}}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right)\right]_0^{\mathrm{T}}.$$

This establishes Case 2 (the general formula for arbitrary $\boldsymbol{\eta}$). Case 1 follows immediately: when $\boldsymbol{\eta}_0 = \boldsymbol{\eta}_T = \mathbf{0}$, the boundary terms vanish and $\delta_{\mathbf{s}} A_\beta \cdot \boldsymbol{\eta} = 0$. □

Appendix F. Proof theorem 1. Lagrangian EP gradient estimator

Proof of theorem 1. We consider the cross-derivatives of the action functional $A_\beta[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}]$. Since A_β depends on $\boldsymbol{\theta}$ both *explicitly* (through $L_\beta(\cdot,\cdot,\boldsymbol{\theta})$) and *implicitly* (through the trajectory $\mathbf{s}^\beta(\boldsymbol{\theta})$), the total derivative decomposes as $d_{\boldsymbol{\theta}} A_\beta = \partial_{\boldsymbol{\theta}} A_\beta + \delta_{\mathbf{s}} A_\beta\, \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta$, where $\partial_{\boldsymbol{\theta}}$ acts on the explicit $\boldsymbol{\theta}$-dependence at fixed trajectory and $\delta_{\mathbf{s}} A_\beta\, \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta$ captures the implicit variation through $\mathbf{s}^\beta(\boldsymbol{\theta})$ (see proposition 3 and the notation conventions in appendix A).

First, differentiating with respect to $\boldsymbol{\theta}$ then β (at $\beta = 0$):

$$\begin{aligned} d_{\beta\boldsymbol{\theta}} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right] &= d_\beta|_{\beta=0}\left(\partial_{\boldsymbol{\theta}} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right] + \delta_{\mathbf{s}} A_\beta \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta\right) \\ &= d_\beta|_{\beta=0} \int_0^T \partial_{\boldsymbol{\theta}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right) \mathrm{d}t + d_\beta|_{\beta=0}\left(\delta_{\mathbf{s}} A_\beta \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta\right). \end{aligned} \tag{24}$$

Second, differentiating with respect to β (at $\beta = 0$) then $\boldsymbol{\theta}$:

$$\begin{aligned} d_{\boldsymbol{\theta}\beta} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right] &= d_{\boldsymbol{\theta}}\left(\partial_\beta|_{\beta=0} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right] + \delta_{\mathbf{s}} A_0 \delta_\beta|_{\beta=0} \mathbf{s}^\beta\right) \\ &= d_{\boldsymbol{\theta}}\left(C\left[\mathbf{s}^0(\boldsymbol{\theta})\right] + \delta_{\mathbf{s}} A_0 \delta_\beta|_{\beta=0} \mathbf{s}^\beta\right), \end{aligned} \tag{25}$$

where we used the fact that $\partial_\beta|_{\beta=0} A_\beta[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}] = \int_0^T c(\mathbf{s}_t^0(\boldsymbol{\theta}))\mathrm{d}t = C[\mathbf{s}^0(\boldsymbol{\theta})]$.

By Schwarz's theorem (symmetry of mixed partial derivatives), we have (since β and $\boldsymbol{\theta}$ are independent, we can use d instead of ∂):

$$d_{\beta\boldsymbol{\theta}} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right] = d_{\boldsymbol{\theta}\beta} A_\beta\left[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}\right].$$

This requires the composite map $(\beta, \boldsymbol{\theta}) \mapsto A_\beta[\mathbf{s}^\beta(\boldsymbol{\theta}), \boldsymbol{\theta}]$ to be C^2. This holds whenever the Lagrangian is C^2 in all its arguments and the trajectory map $(\beta, \boldsymbol{\theta}) \mapsto \mathbf{s}^\beta(\boldsymbol{\theta})$ is C^2, which follows from the standard smooth dependence of ODE solutions on parameters.

Equating the right-hand sides of equations (24) and (25), we obtain:

$$d_{\boldsymbol{\theta}} C\left[\mathbf{s}^0\right] = d_\beta \int_0^T \partial_{\boldsymbol{\theta}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right) \mathrm{d}t + \left(d_\beta\left(\delta_{\mathbf{s}} A_\beta \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta\right) - d_{\boldsymbol{\theta}}\left(\delta_{\mathbf{s}} A_0 \delta_\beta \mathbf{s}^\beta\right)\right). \tag{26}$$

From proposition 3 (derived from lemma 1), the variation through implicit dependence gives:

$$d_\beta\left(\delta_{\mathbf{s}} A_\beta \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta\right) = d_\beta\left[\left(\partial_{\boldsymbol{\theta}} \mathbf{s}_t^\beta\right)^\top \cdot \partial_{\dot{\mathbf{s}}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right)\right]_0^{\mathrm{T}}. \tag{27}$$

Applying the product rule of differentiation:

$$d_\beta\left(\delta_{\mathbf{s}} A_\beta \delta_{\boldsymbol{\theta}} \mathbf{s}^\beta\right) = \left[\left(\partial_{\beta\boldsymbol{\theta}} \mathbf{s}_t^\beta\right)^\top \cdot \partial_{\dot{\mathbf{s}}} L_0\left(\mathbf{s}_t^0, \dot{\mathbf{s}}_t^0, \boldsymbol{\theta}\right) + \left(\partial_{\boldsymbol{\theta}} \mathbf{s}_t^0\right)^\top \cdot d_\beta \partial_{\dot{\mathbf{s}}} L_\beta\left(\mathbf{s}_t^\beta, \dot{\mathbf{s}}_t^\beta, \boldsymbol{\theta}\right)\right]_0^{\mathrm{T}}. \tag{28}$$

By the same reasoning applied to (27) with the roles of β and $\boldsymbol{\theta}$ exchanged:

$$d_{\boldsymbol{\theta}}\left(\delta_{\mathbf{s}} A_0 \delta_\beta \mathbf{s}^\beta\right) = \left[\left(\partial_{\boldsymbol{\theta}\beta} \mathbf{s}_t^\beta\right)^\top \cdot \partial_{\dot{\mathbf{s}}} L_0\left(\mathbf{s}_t^0, \dot{\mathbf{s}}_t^0, \boldsymbol{\theta}\right) + \left(d_{\boldsymbol{\theta}} \partial_{\dot{\mathbf{s}}} L_0\left(\mathbf{s}_t^0, \dot{\mathbf{s}}_t^0, \boldsymbol{\theta}\right)\right)^\top \cdot \left(\partial_\beta \mathbf{s}_t^\beta\right)\right]_0^{\mathrm{T}}. \tag{29}$$

Using the symmetry of cross-derivatives, $\partial_{\boldsymbol{\theta}\beta}\boldsymbol{s}_t^{\beta} = \partial_{\beta\boldsymbol{\theta}}\boldsymbol{s}_t^{\beta}$, the first terms in equations (28) and (29) cancel:

$$d_\beta\left(\delta_{\boldsymbol{s}}A_\beta\delta_{\boldsymbol{\theta}}\boldsymbol{s}^\beta\right) - d_{\boldsymbol{\theta}}\left(\delta_{\boldsymbol{s}}A_0\delta_\beta\boldsymbol{s}^\beta\right) = \left[\left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_t^0\right)^\top \cdot d_\beta\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_t^0, \dot{\boldsymbol{s}}_t^0, \boldsymbol{\theta}\right)\right)^\top \cdot \partial_\beta\boldsymbol{s}_t^\beta\right]_0^{\mathrm{T}}. \tag{30}$$

Substituting equation (30) into Equation (26) yields the final result. □

Appendix G. Proof of LEP Corollaries

G.1. Proof of corollary 2: gradient estimator for CIVP

Proof of corollary 2. We apply theorem 1 to the CIVP formulation and analyze the boundary residual terms. From theorem 1, the boundary residual term is:

$$\left[\left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,t}^0\right)^\top d_\beta\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_{\rightarrow,t}^\beta, \dot{\boldsymbol{s}}_{\rightarrow,t}^\beta, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,t}^0, \dot{\boldsymbol{s}}_{\rightarrow,t}^0, \boldsymbol{\theta}\right)\right)^\top \partial_\beta\boldsymbol{s}_{\rightarrow,t}^\beta\right]_0^{\mathrm{T}}.$$

We examine the boundary conditions at both temporal endpoints.

Analysis at $t = 0$: The boundary residual vanishes due to the constant initial value constraints. By the CIVP construction, all trajectories satisfy the boundary conditions $\boldsymbol{s}_{\rightarrow,0}^\beta(\boldsymbol{\theta}) = \boldsymbol{\alpha}_0$ and $\dot{\boldsymbol{s}}_{\rightarrow,0}^\beta(\boldsymbol{\theta}) = \boldsymbol{\gamma}_0$, which are independent of both $\boldsymbol{\theta}$ and β.

The left term vanishes because:

$$\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,0}^0 = \partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0 = \mathbf{0}.$$

The right term vanishes because:

$$\partial_\beta\boldsymbol{s}_{\rightarrow,0}^\beta = \partial_\beta\boldsymbol{\alpha}_0 = \mathbf{0}.$$

Therefore, both boundary residual terms are zero at $t = 0$.

Analysis at $t = T$: The boundary residual does not cancel due to the absence of constraints at the final time. Unlike at the initial conditions, no boundary value constraints are imposed at $t = T$, so both $\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,T}^0$ and $\partial_\beta\boldsymbol{s}_{\rightarrow,T}^\beta$ are generally non-zero. Notably, since β is scalar, the β derivatives can easily be estimated via finite differences. To emphasize this, we can rewrite the left term as:

$$\left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,T}^0\right)^\top d_\beta\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_{\rightarrow,T}^\beta, \dot{\boldsymbol{s}}_{\rightarrow,T}^\beta, \boldsymbol{\theta}\right) = \lim_{\beta\to 0}\frac{1}{\beta}\left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,T}^0\right)^\top\left[\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_{\rightarrow,T}^\beta, \dot{\boldsymbol{s}}_{\rightarrow,T}^\beta, \boldsymbol{\theta}\right) - \partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,T}^0, \dot{\boldsymbol{s}}_{\rightarrow,T}^0, \boldsymbol{\theta}\right)\right].$$

Similarly, the right term becomes:

$$\left(d_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,T}^0, \dot{\boldsymbol{s}}_{\rightarrow,T}^0, \boldsymbol{\theta}\right)\right)^\top \partial_\beta\boldsymbol{s}_{\rightarrow,T}^\beta = \lim_{\beta\to 0}\frac{1}{\beta}\left(d_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,T}^0, \dot{\boldsymbol{s}}_{\rightarrow,T}^0, \boldsymbol{\theta}\right)\right)^\top\left(\boldsymbol{s}_{\rightarrow,T}^\beta - \boldsymbol{s}_{\rightarrow,T}^0\right).$$

Final result: Combining the integral term (in finite difference form) from theorem 1 with the boundary analysis and applying the finite difference approximation, we obtain:

$$\begin{aligned} d_{\boldsymbol{\theta}}C\left[\boldsymbol{s}_{\rightarrow}^0(\boldsymbol{\theta})\right] = \lim_{\beta\to 0}\frac{1}{\beta}\Bigg[&\int_0^T\left[\partial_{\boldsymbol{\theta}}L_\beta\left(\boldsymbol{s}_{\rightarrow,t}^\beta, \dot{\boldsymbol{s}}_{\rightarrow,t}^\beta, \boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}_{\rightarrow,t}^0, \dot{\boldsymbol{s}}_{\rightarrow,t}^0, \boldsymbol{\theta}\right)\right]\mathrm{dt} \\ &+ \left(\partial_{\boldsymbol{\theta}}\boldsymbol{s}_{\rightarrow,T}^0\right)^\top\left(\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}_{\rightarrow,T}^\beta, \dot{\boldsymbol{s}}_{\rightarrow,T}^\beta, \boldsymbol{\theta}\right) - \partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,T}^0, \dot{\boldsymbol{s}}_{\rightarrow,T}^0, \boldsymbol{\theta}\right)\right) \\ &- \left(d_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{s}_{\rightarrow,T}^0, \dot{\boldsymbol{s}}_{\rightarrow,T}^0, \boldsymbol{\theta}\right)\right)^\top\left(\boldsymbol{s}_{\rightarrow,T}^\beta - \boldsymbol{s}_{\rightarrow,T}^0\right)\Bigg]. \end{aligned}$$

The boundary residuals at $t = T$ remain due to the absence of final time constraints. □

G.2. Proof of corollary 1: Gradient estimator for CBPVP

Proof of corollary 1. We apply theorem 1 to the CBPVP formulation and analyze the boundary residual terms. From theorem 1, the boundary residual term is:

$$\left[\left(\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_{\leftrightarrow,t}\right)^\top d_\beta \partial_{\dot{\boldsymbol{s}}} L_\beta \left(\boldsymbol{s}^\beta_{\leftrightarrow,t}, \dot{\boldsymbol{s}}^\beta_{\leftrightarrow,t}, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0 \left(\boldsymbol{s}^0_{\leftrightarrow,t}, \dot{\boldsymbol{s}}^0_{\leftrightarrow,t}, \boldsymbol{\theta}\right)\right)^\top \partial_\beta \boldsymbol{s}^\beta_{\leftrightarrow,t}\right]_0^{\mathrm{T}}.$$

We examine the boundary conditions at both temporal endpoints.

Analysis at $t = 0$: The boundary residual vanishes due to the constant initial position constraint. By the CBPVP construction, all trajectories satisfy the boundary condition $\boldsymbol{s}^\beta_{\leftrightarrow,0}(\boldsymbol{\theta}) = \boldsymbol{\alpha}_0$, which is independent of both $\boldsymbol{\theta}$ and β.

The left term vanishes because:

$$\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_{\leftrightarrow,0} = \partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}_0 = \mathbf{0}.$$

The right term vanishes because:

$$\partial_\beta \boldsymbol{s}^\beta_{\leftrightarrow,0} = \partial_\beta \boldsymbol{\alpha}_0 = \mathbf{0}.$$

Therefore, both boundary residual terms are zero at $t = 0$.

Analysis at $t = T$: The boundary residual also vanishes due to the constant final position constraint. By the CBPVP construction, all trajectories satisfy the boundary condition $\boldsymbol{s}^\beta_{\leftrightarrow,T}(\boldsymbol{\theta}) = \boldsymbol{\alpha}_T$, which is independent of both $\boldsymbol{\theta}$ and β.

The left term vanishes because:

$$\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_{\leftrightarrow,T} = \partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}_T = \mathbf{0}.$$

The right term vanishes because:

$$\partial_\beta \boldsymbol{s}^\beta_{\leftrightarrow,T} = \partial_\beta \boldsymbol{\alpha}_T = \mathbf{0}.$$

Therefore, both boundary residual terms are zero at $t = T$.

Final result: Since the boundary residuals vanish at both endpoints, combining with the integral term from theorem 1 and applying the finite difference approximation, we obtain:

$$d_{\boldsymbol{\theta}} C\left[\boldsymbol{s}^0_{\leftrightarrow}(\boldsymbol{\theta})\right] = \lim_{\beta \to 0} \frac{1}{\beta} \int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta \left(\boldsymbol{s}^\beta_{\leftrightarrow,t}, \dot{\boldsymbol{s}}^\beta_{\leftrightarrow,t}, \boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0 \left(\boldsymbol{s}^0_{\leftrightarrow,t}, \dot{\boldsymbol{s}}^0_{\leftrightarrow,t}, \boldsymbol{\theta}\right)\right] \mathrm{dt}.$$

The CBPVP formulation eliminates all problematic boundary residual terms, yielding a clean gradient estimator that only requires integrating differences between Lagrangian derivatives over the two trajectories. □

Appendix H. Proof of theorem 2: Gradient estimator from RHEL with parametrized initial state

This section shows how theorem 2 can be recovered from the results in the RHEL paper [PE25]. We proceed in two steps: first clarifying the notation differences between the two papers, then deriving how the gradient with respect to a parametrized initial state combines with the gradient with respect to parameters.

H.1. Notation correspondence

The RHEL paper [PE25] uses a time convention where the forward pass runs from $t \in [-T, 0]$ and the echo phase runs from $t \in [0, T]$. In contrast, this paper uses $t \in [0, T]$ for the forward pass. The correspondences are:

Time indexing. For the forward trajectory:

- RHEL paper: forward trajectory $\boldsymbol{\Phi}(t)$ for $t \in [-T, 0]$

- This paper: forward trajectory $\mathbf{\Phi}_t$ for $t \in [0, T]$ with inputs $\boldsymbol{x}_t$
- Relationship: $\mathbf{\Phi}_t \leftrightarrow \mathbf{\Phi}(-t+T)$ in RHEL notation

For the echo trajectory:

- RHEL paper: echo trajectory $\mathbf{\Phi}^e(t,\epsilon)$ for $t \in [0, T]$ with inputs $\boldsymbol{u}(-t)$, where ϵ is the nudging strength
- This paper: echo trajectory $\mathbf{\Phi}^e_t$ for $t \in [0, T]$ with inputs $\boldsymbol{x}_{T-t}$. The dependence on the nudging strength β is left implicit in the subscript notation $\mathbf{\Phi}^e_t \equiv \mathbf{\Phi}^e_t(\beta)$, since β is fixed throughout the forward and echo phases and only appears explicitly in the limit $\beta \to 0$
- Relationship: inputs are time-reversed relative to forward pass

State variables. The phase space variables are:

- RHEL paper: $\mathbf{\Phi} = \begin{pmatrix} \phi \\ \boldsymbol{\pi} \end{pmatrix}$ where ϕ represents positions and $\boldsymbol{\pi}$ represents conjugate momenta
- This paper: $\mathbf{\Phi} = \begin{pmatrix} \boldsymbol{s} \\ \boldsymbol{p} \end{pmatrix}$ where $\boldsymbol{s}$ represents positions and $\boldsymbol{p}$ represents conjugate momenta, with $\boldsymbol{s}$ also denoted as $\boldsymbol{\alpha}$ and $\boldsymbol{p}$ as $\boldsymbol{\gamma}^p$ for initial conditions
- Correspondence for forward phase: $\mathbf{\Phi}_t = \begin{pmatrix} \boldsymbol{s}_t \\ \boldsymbol{p}_t \end{pmatrix}$ (this paper) $\leftrightarrow$ $\mathbf{\Phi}(t) = \begin{pmatrix} \phi_t \\ \boldsymbol{\pi}_t \end{pmatrix}$ (RHEL paper)
- Correspondence for echo phase: $\mathbf{\Phi}^e_t = \begin{pmatrix} \boldsymbol{s}^e_t \\ \boldsymbol{p}^e_t \end{pmatrix}$ (this paper) $\leftrightarrow$ $\mathbf{\Phi}^e(t) = \begin{pmatrix} \phi^e_t \\ \boldsymbol{\pi}^e_t \end{pmatrix}$ (RHEL paper)

Nudging parameter. The RHEL paper uses ϵ for bidirectional perturbations $\pm\epsilon$ while this paper uses β for unidirectional perturbation; both converge to the same gradient as $\epsilon, \beta \to 0$.

H.2. Gradient decomposition for parametrized initial states

We now show how to recover theorem 2 from the original RHEL result (theorem 3.1 in [PE25]). We proceed by first recalling the RHEL result for fixed initial conditions, then the gradient with respect to initial state, and finally combining them.

Step 1: Gradient with respect to parameters (fixed initial state).

When the initial state $\mathbf{\Phi}_0 = \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix}$ is held fixed (independent of $\boldsymbol{\theta}$), theorem 3.1 in [PE25] gives[11]:

$$\partial_{\boldsymbol{\theta}} C[\mathbf{\Phi}(\boldsymbol{\theta}, \mathbf{\Phi}_0)] = \lim_{\beta \to 0} \frac{1}{\beta} \left[-\int_0^T \left(\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}^e_t, \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t) \right) dt \right],$$

where $\mathbf{\Phi}_t$ is the forward trajectory and $\mathbf{\Phi}^e_t$ is the echo trajectory with nudging strength β.

Step 2: Gradient with respect to initial state (fixed parameters). The RHEL paper also provides the gradient of the cost with respect to the initial state (holding parameters $\boldsymbol{\theta}$ fixed). This sensitivity can be expressed through the echo trajectory difference at the final time:

$$\partial_{\mathbf{\Phi}_0} C = \lim_{\beta \to 0} \frac{1}{\beta} \mathbf{\Sigma}_x \left(\begin{pmatrix} \boldsymbol{s}^e_T \\ \boldsymbol{p}^e_T \end{pmatrix} - \begin{pmatrix} \boldsymbol{\alpha}_0 \\ -\boldsymbol{\mu}_0 \end{pmatrix} \right),$$

where $\begin{pmatrix} \boldsymbol{s}^e_T \\ \boldsymbol{p}^e_T \end{pmatrix} = \mathbf{\Phi}^e_T$ is the echo trajectory at final time T, and the sign flip in the momentum component arises from the momentum-flipping boundary condition of the echo phase.

[11] Note that the RHEL paper uses bidirectional nudging with perturbations $\pm\epsilon$ for improved numerical accuracy, while we present the unidirectional formulation here for simplicity. Both converge to the same gradient in the continuous-time limit; see the note at the end of this section for details.

Step 3: Combining both contributions via chain rule. When the initial state depends on parameters, $\mathbf{\Phi}_0(\boldsymbol{\theta}) = \begin{pmatrix}\boldsymbol{\alpha}_0(\boldsymbol{\theta})\\ \boldsymbol{\mu}_0(\boldsymbol{\theta})\end{pmatrix}$, the total gradient must account for both the direct dependence on $\boldsymbol{\theta}$ and the indirect dependence through $\mathbf{\Phi}_0(\boldsymbol{\theta})$. By the chain rule:

$$\mathrm{d}_{\boldsymbol{\theta}} C[\mathbf{\Phi}(\boldsymbol{\theta}, \mathbf{\Phi}_0(\boldsymbol{\theta}))] = \underbrace{\partial_{\boldsymbol{\theta}} C[\mathbf{\Phi}(\boldsymbol{\theta}, \mathbf{\Phi}_0)]}_{\text{direct, holding } \mathbf{\Phi}_0 \text{ fixed}} + \underbrace{(\partial_{\boldsymbol{\theta}} \mathbf{\Phi}_0)^\top \partial_{\mathbf{\Phi}_0} C}_{\text{indirect, through } \mathbf{\Phi}_0(\boldsymbol{\theta})}.$$

Substituting the expressions from Steps 1 and 2:

$$\mathrm{d}_{\boldsymbol{\theta}} C = \lim_{\beta \to 0} \frac{1}{\beta}\left[-\int_0^T (\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t^e, \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t))\, dt + \left(\partial_{\boldsymbol{\theta}} \begin{pmatrix}\boldsymbol{\alpha}_0\\ \boldsymbol{\mu}_0\end{pmatrix}\right)^\top \mathbf{\Sigma}_x \left(\begin{pmatrix}\boldsymbol{s}_T^e\\ \boldsymbol{p}_T^e\end{pmatrix} - \begin{pmatrix}\boldsymbol{\alpha}_0\\ -\boldsymbol{\mu}_0\end{pmatrix}\right)\right],$$

which is precisely the gradient estimator $\Delta^{\mathrm{RHEL}}(\beta, \boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\mu}_0(\boldsymbol{\theta}))$ in equation (11) of theorem 2.

H.3. Note on bidirectional vs unidirectional nudging

The RHEL paper uses bidirectional nudging with perturbations $\pm\beta$, computing:

$$\Delta^{\mathrm{RHEL}}(\beta) = \frac{1}{2\beta}\left[(\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t^e(+\beta), \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)) - (\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t^e(-\beta), \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t))\right],$$

which is a centered finite-difference approximation. In this paper, we present the unidirectional formulation:

$$\Delta^{\mathrm{RHEL}}(\beta) = \frac{1}{\beta}(\partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t^e(\beta), \boldsymbol{\theta}, \boldsymbol{x}_{T-t}) - \partial_{\boldsymbol{\theta}} H(\mathbf{\Phi}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)),$$

which is a forward finite-difference approximation. Both converge to the same gradient in the limit $\beta \to 0$. The bidirectional version has better numerical accuracy (second-order error $O(\beta^2)$ vs first-order $O(\beta)$), a well-known trick in equilibrium propagation [LES+21].

Appendix I. Proof of proposition 2: The bouncing-back trick

Proof of proposition 2. Define the time-reversed trajectory:

$$\boldsymbol{s}_{\mathrm{rev},t}^\beta := \boldsymbol{s}_{\leftarrow,T-t}^\beta(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T, \boldsymbol{\gamma}_T)). \tag{31}$$

By the chain rule ($\frac{\mathrm{d}(T-t)}{\mathrm{d}t} = -1$), its velocity and acceleration satisfy:

$$\dot{\boldsymbol{s}}_{\mathrm{rev},t}^\beta = -\dot{\boldsymbol{s}}_{\leftarrow,T-t}^\beta, \qquad \ddot{\boldsymbol{s}}_{\mathrm{rev},t}^\beta = \ddot{\boldsymbol{s}}_{\leftarrow,T-t}^\beta. \tag{32}$$

Step 1: $\boldsymbol{s}_{\mathrm{rev}}$ satisfies the Euler–Lagrange equation. We show that $\mathrm{EL}_r(t, \boldsymbol{\theta}, \beta) = 0$ along $\boldsymbol{s}_{\mathrm{rev}}$ by relating each term back to the Euler–Lagrange equation satisfied by $\boldsymbol{s}_{\leftarrow}^\beta$. Let $t' := T - t$.

Momentum term. By (32) and the odd-derivative property (lemma 2):

$$\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\mathrm{rev},t}^\beta, \dot{\boldsymbol{s}}_{\mathrm{rev},t}^\beta, \boldsymbol{\theta}\right) = \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, -\dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right) = -\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, \dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right).$$

Taking the total time derivative and using $d_t = -d_{t'}$:

$$\begin{aligned} d_t \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\mathrm{rev},t}^\beta, \dot{\boldsymbol{s}}_{\mathrm{rev},t}^\beta, \boldsymbol{\theta}\right) &= d_t\left[-\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, \dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right)\right] = (-d_{t'})\left[-\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, \dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right)\right] \\ &= d_{t'} \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, \dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right). \end{aligned}$$

Position term. Since L_β is even in $\dot{\boldsymbol{s}}$, so is $\partial_{\boldsymbol{s}} L_\beta$ (differentiating $L_\beta(\boldsymbol{s}, \dot{\boldsymbol{s}}, \boldsymbol{\theta}) = L_\beta(\boldsymbol{s}, -\dot{\boldsymbol{s}}, \boldsymbol{\theta})$ with respect to $\boldsymbol{s}$):

$$\partial_{\boldsymbol{s}} L_\beta\left(\boldsymbol{s}_{\mathrm{rev},t}^\beta, \dot{\boldsymbol{s}}_{\mathrm{rev},t}^\beta, \boldsymbol{\theta}\right) = \partial_{\boldsymbol{s}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, -\dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right) = \partial_{\boldsymbol{s}} L_\beta\left(\boldsymbol{s}_{\leftarrow,t'}^\beta, \dot{\boldsymbol{s}}_{\leftarrow,t'}^\beta, \boldsymbol{\theta}\right).$$

Combining:

$$\partial_{\dot{s}} L_\beta\left(s^\beta_{\mathrm{rev},t}, \dot{s}^\beta_{\mathrm{rev},t}, \boldsymbol{\theta}\right) - d_t \partial_{\dot{s}} L_\beta\left(s^\beta_{\mathrm{rev},t}, \dot{s}^\beta_{\mathrm{rev},t}, \boldsymbol{\theta}\right) = \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,t'}, \dot{s}^\beta_{\leftarrow,t'}, \boldsymbol{\theta}\right) - d_{t'} \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,t'}, \dot{s}^\beta_{\leftarrow,t'}, \boldsymbol{\theta}\right)$$
$$= 0. \quad (s^\beta_{\leftarrow} \text{satisfies Euler-Lagrange at } t' = T - t)$$

Note that $\mathrm{EL}_r(t', \boldsymbol{\theta}, \beta)$ is evaluated with input $\boldsymbol{x}_{t'}$ and target $\boldsymbol{y}_{t'}$ at time $t' = T - t$, so the nudged dynamics in the IVP automatically use the time-reversed input and target sequences.

Step 2: Initial conditions of s_{rev}. Using (31) and (32) with the PFVP boundary conditions $s^\beta_{\leftarrow,T} = \boldsymbol{\alpha}_T$ and $\dot{s}^\beta_{\leftarrow,T} = \boldsymbol{\gamma}_T$:

$$s^\beta_{\mathrm{rev},0} = s^\beta_{\leftarrow,T} = \boldsymbol{\alpha}_T, \qquad \dot{s}^\beta_{\mathrm{rev},0} = -\dot{s}^\beta_{\leftarrow,T} = -\boldsymbol{\gamma}_T.$$

Step 3: Uniqueness. Since s^β_{rev} and $s^\beta_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T, -\boldsymbol{\gamma}_T))$ both satisfy the same Euler–Lagrange equation with the same initial conditions $(\boldsymbol{\alpha}_T, -\boldsymbol{\gamma}_T)$ at $t = 0$, they are identical by uniqueness of the initial value problem (Remark 4):

$$s^\beta_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T, -\boldsymbol{\gamma}_T)) = s^\beta_{\mathrm{rev},t} = s^\beta_{\leftarrow,T-t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T, \boldsymbol{\gamma}_T)) \quad \text{(by (31))}.$$

A time translation $t' \leftarrow T - t$ gives the desired result. □

Appendix J. Proof theorem 3: PFVP cancels the boundary residuals

Proof of theorem 3 Let us analyze the boundary residual term from theorem 1 for the PFVP trajectories $t \mapsto s^\beta_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta})))$:

$$\left[\left(\partial_{\boldsymbol{\theta}} s^0_{\leftarrow,t}\right)^\top \cdot d_\beta \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,t}, \dot{s}^\beta_{\leftarrow,t}, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}} \partial_{\dot{s}} L_0\left(s^0_{\leftarrow,t}, \dot{s}^0_{\leftarrow,t}, \boldsymbol{\theta}\right)\right)^\top \cdot \partial_\beta s^\beta_{\leftarrow,t}\right]_0^T.$$

We examine the boundary conditions at both temporal endpoints.

Analysis at $t = T$: The boundary residual vanishes due to the parametric final value constraint.

The right term disappears because $\partial_\beta s^\beta_{\leftarrow,T} = 0$. By the PFVP construction, the nudged trajectory satisfies the boundary condition $s^\beta_{\leftarrow,T}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) = \boldsymbol{\alpha}_T(\boldsymbol{\theta})$, which is independent of β. The left term cancels because:

$$\begin{aligned} d_\beta \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,T}, \dot{s}^\beta_{\leftarrow,T}, \boldsymbol{\theta}\right) &= d_\beta \partial_{\dot{s}} L_0\left(s^\beta_{\leftarrow,T}, \dot{s}^\beta_{\leftarrow,T}, \boldsymbol{\theta}\right) + \partial^2_{\beta,\dot{s}} L_\beta\left(s^0_{\leftarrow,T}, \dot{s}^0_{\leftarrow,T}, \boldsymbol{\theta}\right) \\ &= \partial^2_{\beta,\dot{s}} L_\beta\left(s^0_{\leftarrow,T}, \dot{s}^0_{\leftarrow,T}, \boldsymbol{\theta}\right) \quad \text{(both } s^\beta_{\leftarrow,T} = \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \text{ and } \dot{s}^\beta_{\leftarrow,T} = \boldsymbol{\gamma}_T(\boldsymbol{\theta}) \text{ are } \beta\text{-independent)} \\ &= \partial_{\dot{s}} c\left(s^0_{\leftarrow,T}, \dot{s}^0_{\leftarrow,T}\right) \\ &= 0. \qquad \text{(cost function } c \text{ depends only on position, not velocity)} \end{aligned}$$

Analysis at $t = 0$: The boundary residual reduces to easy-to-compute terms at $t = 0$.

$$\begin{aligned} s^0_{\leftarrow,0} &= s^0_{\leftarrow,0}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \quad \text{(definition 12)} \\ &= s^0_0(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}))) \quad \text{(proposition 4 evaluated at } t = 0) \\ &= \boldsymbol{\alpha}_0(\boldsymbol{\theta}). \end{aligned}$$

Similarly we have $\dot{s}^0_{\leftarrow,0} = \boldsymbol{\gamma}_0(\boldsymbol{\theta})$.

Since $s^0_{\leftarrow,0} = \boldsymbol{\alpha}_0(\boldsymbol{\theta})$ and $\dot{s}^0_{\leftarrow,0} = \boldsymbol{\gamma}_0(\boldsymbol{\theta})$, the right term simplifies to:

$$d_{\boldsymbol{\theta}} \partial_{\dot{s}} L_\beta\left(s^0_{\leftarrow,0}, \dot{s}^0_{\leftarrow,0}, \boldsymbol{\theta}\right) = d_{\boldsymbol{\theta}} \partial_{\dot{s}} L_\beta\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right).$$

Final result: All terms cancel at $t = T$. At $t = 0$, the boundary residual evaluates to:

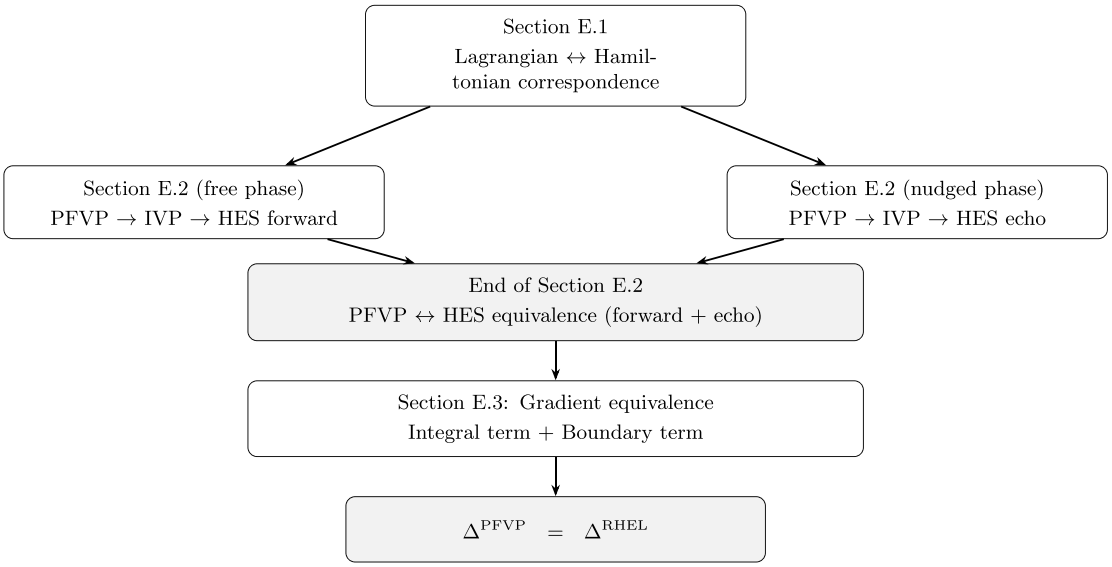
$$\begin{aligned} &\left[\left(\partial_{\boldsymbol{\theta}} s^0_{\leftarrow,t}\right)^\top d_\beta \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,t}, \dot{s}^\beta_{\leftarrow,t}, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}} \partial_{\dot{s}} L_0\left(s^0_{\leftarrow,t}, \dot{s}^0_{\leftarrow,t}, \boldsymbol{\theta}\right)\right)^\top \cdot \partial_\beta s^\beta_{\leftarrow,t}\right]_0^T \\ &= \left(\partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}_0(\boldsymbol{\theta})\right)^\top d_\beta \partial_{\dot{s}} L_\beta\left(s^\beta_{\leftarrow,0}, \dot{s}^\beta_{\leftarrow,0}, \boldsymbol{\theta}\right) - \left(d_{\boldsymbol{\theta}} \partial_{\dot{s}} L_0\left(\boldsymbol{\alpha}_0(\boldsymbol{\theta}), \boldsymbol{\gamma}_0(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\right)^\top \cdot \partial_\beta s^\beta_{\leftarrow,0}. \end{aligned}$$

This yields the desired result. □

Appendix K. Proof theorem 4: Equivalence between Lagrangian EP and Recurrent Hamiltonian Echo Learning

Proof roadmap. The proof proceeds in three stages.

1. **Lagrangian–Hamiltonian correspondence (section E.1).** We recall the classical result that the Legendre transform maps Euler–Lagrange trajectories bijectively to Hamilton trajectories (theorem 6).
2. **PFVP ↔ HES trajectory correspondence (section E.2).** Using theorem 6 together with the PFVP reversibility (proposition 2), we construct bijective maps between the PFVP free/nudged phases and the HES forward/echo phases.
3. **Gradient equivalence (section E.3).** We transform the PFVP gradient estimator term by term—first the integral term, then the boundary term—to obtain the RHEL estimator.



K.1. Relating the solutions of Euler–Lagrange and Hamilton equations

Here we first recall a classic theorem about the Legendre transform and how it is used in physics to relate solutions of the Euler–Lagrange and Hamilton's equation.

Theorem 6 (Equivalence of Lagrangian and Hamiltonian dynamics). *Assume the Legendre transform of proposition 1 is well defined along the trajectories considered, i.e. the Hessian condition* $\det(\partial^2_{\dot{s},\dot{s}} L(s_t, \dot{s}_t)) \neq 0$ *holds at each point along the trajectory.*

Then the Legendre transform maps solutions of the Euler–Lagrange equations bijectively to solutions of Hamilton's equations, together with their initial conditions.

1. ***Correspondence of initial conditions.*** *For every Lagrangian initial condition* $(s_0, \dot{s}_0)$ *there exists a unique Hamiltonian initial condition*

$$p_0 = \partial_{\dot{s}} L(s_0, \dot{s}_0),$$

and for every Hamiltonian initial condition (s_0, p_0) *there exists a unique Lagrangian initial condition*

$$\dot{s}_0 = \partial_p H(s_0, p_0).$$

Thus the Legendre map induces a bijection between initial conditions.

2. ***Correspondence of solutions.*** *Let* $p_t = \partial_{\dot{s}} L(s_t, \dot{s}_t)$. *Then:*
   - *If the trajectory* $t \mapsto s_t$ *satisfies the Euler–Lagrange equations*

$$\mathrm{d}_t(\partial_{\dot{s}} L(s_t, \dot{s}_t)) - \partial_s L(s_t, \dot{s}_t) = 0,$$

then the pair (s_t, p_t) *satisfies Hamilton's equations*

$$\dot{s}_t = \partial_p H(s_t, p_t), \qquad \dot{p}_t = -\partial_s H(s_t, p_t).$$

- *Conversely, if* $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ *satisfies Hamilton's equations, then* $\boldsymbol{s}_t$ *satisfies the Euler–Lagrange equations, with*

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t), \qquad \boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t).$$

Consequently, under a well-defined Legendre transform, there is a one-to-one correspondence between Lagrangian trajectories $\boldsymbol{s}_t$ *and Hamiltonian trajectories* $(\boldsymbol{s}_t, \boldsymbol{p}_t)$*, together with their initial conditions.*

Proof. By assumption, the Hessian condition $\det(\partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L) \neq 0$ holds along the trajectories considered, so the Legendre transform of proposition 1 gives a smooth locally invertible map

$$(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) \longleftrightarrow (\boldsymbol{s}_t, \boldsymbol{p}_t)$$

at each time t, with

$$\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t), \qquad H(\boldsymbol{s}_t, \boldsymbol{p}_t) = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t),$$

and inverse relations

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t), \qquad L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = \boldsymbol{p}_t^\top \dot{\boldsymbol{s}}_t - H(\boldsymbol{s}_t, \boldsymbol{p}_t).$$

We first show that this induces a bijection between initial conditions, then prove the equivalence of the equations of motion.

1. Correspondence of initial conditions. Since for each fixed $\boldsymbol{s}$ the map

$$\dot{\boldsymbol{s}} \mapsto \boldsymbol{p} = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}})$$

is locally invertible (by the non-degenerate Hessian condition of proposition 1), it follows in particular that at time $t = 0$ the map

$$(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) \longleftrightarrow (\boldsymbol{s}_0, \boldsymbol{p}_0), \qquad \boldsymbol{p}_0 = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0),$$

is one-to-one and onto, with inverse

$$\dot{\boldsymbol{s}}_0 = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_0, \boldsymbol{p}_0).$$

This proves the stated bijection between Lagrangian and Hamiltonian initial conditions.

2. Two basic identities for the Legendre transform. We now derive two standard identities that hold whenever H is the Legendre transform of L:

$$\partial_{\boldsymbol{p}} H(\boldsymbol{s}, \boldsymbol{p}) = \dot{\boldsymbol{s}}, \qquad \partial_{\boldsymbol{s}} H(\boldsymbol{s}, \boldsymbol{p}) = -\partial_{\boldsymbol{s}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}),$$

where $\dot{\boldsymbol{s}}$ is implicitly defined by $\boldsymbol{p} = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}})$.
Identity $\partial_{\boldsymbol{p}} H = \dot{\boldsymbol{s}}$. By definition of H,

$$H(\boldsymbol{s}, \boldsymbol{p}) = \boldsymbol{p}^\top \dot{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{p}) - L(\boldsymbol{s}, \dot{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{p})),$$

where we view $\dot{\boldsymbol{s}}$ as a function of $(\boldsymbol{s}, \boldsymbol{p})$ defined implicitly by

$$\boldsymbol{p} = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{p})).$$

Differentiating H with respect to $\boldsymbol{p}$ at fixed $\boldsymbol{s}$ gives

$$\partial_{\boldsymbol{p}} H = \dot{\boldsymbol{s}} + \left(\partial_{\boldsymbol{p}} \dot{\boldsymbol{s}}\right)^\top \boldsymbol{p} - \left(\partial_{\boldsymbol{p}} \dot{\boldsymbol{s}}\right)^\top \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}).$$

Since $\boldsymbol{p} = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}})$, the last two terms cancel, and we obtain

$$\partial_{\boldsymbol{p}} H(\boldsymbol{s}, \boldsymbol{p}) = \dot{\boldsymbol{s}}.$$

Identity $\partial_{\boldsymbol{s}} H = -\partial_{\boldsymbol{s}} L$. Again, from

$$H(\boldsymbol{s}, \boldsymbol{p}) = \boldsymbol{p}^\top \dot{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{p}) - L(\boldsymbol{s}, \dot{\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{p})),$$

differentiate with respect to $\boldsymbol{s}$ at fixed $\boldsymbol{p}$:

$$\partial_{\boldsymbol{s}} H = \boldsymbol{p}^{\top} \partial_{\boldsymbol{s}} \dot{\boldsymbol{s}} - \partial_{\boldsymbol{s}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}) - (\partial_{\boldsymbol{s}} \dot{\boldsymbol{s}})^{\top} \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}).$$

Using $\boldsymbol{p} = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}, \dot{\boldsymbol{s}})$, the first and third terms cancel, so

$$\partial_{\boldsymbol{s}} H(\boldsymbol{s}, \boldsymbol{p}) = -\partial_{\boldsymbol{s}} L(\boldsymbol{s}, \dot{\boldsymbol{s}}).$$

3. Euler–Lagrange ⇒ Hamilton. Assume the trajectory $t \mapsto \boldsymbol{s}_t$ satisfies the Euler–Lagrange equations

$$\mathrm{d}_t(\partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)) - \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = 0.$$

Define the momentum

$$\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t).$$

We must show that $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ satisfies Hamilton's equations

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t), \qquad \dot{\boldsymbol{p}}_t = -\partial_{\boldsymbol{s}} H(\boldsymbol{s}_t, \boldsymbol{p}_t).$$

The first Hamilton equation follows immediately from the identity $\partial_{\boldsymbol{p}} H = \dot{\boldsymbol{s}}$:

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t).$$

For the second equation, note that the Euler–Lagrange equation implies

$$\dot{\boldsymbol{p}}_t = \mathrm{d}_t(\partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)) = \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t).$$

Using the identity $\partial_{\boldsymbol{s}} H = -\partial_{\boldsymbol{s}} L$ evaluated along the trajectory, we obtain

$$\dot{\boldsymbol{p}}_t = \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = -\partial_{\boldsymbol{s}} H(\boldsymbol{s}_t, \boldsymbol{p}_t),$$

which is exactly the second Hamilton equation.

4. Hamilton ⇒ Euler–Lagrange. Conversely, assume $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ satisfies Hamilton's equations

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t), \qquad \dot{\boldsymbol{p}}_t = -\partial_{\boldsymbol{s}} H(\boldsymbol{s}_t, \boldsymbol{p}_t),$$

and that L and H are related by the Legendre transform as above.

Define the velocity via the inverse Legendre relation

$$\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t),$$

and define

$$\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t),$$

which is consistent by assumption (the Legendre map is a bijection).

We want to show that $\boldsymbol{s}_t$ satisfies the Euler–Lagrange equation

$$\mathrm{d}_t(\partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)) - \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = 0.$$

By definition of $\boldsymbol{p}_t$,

$$\mathrm{d}_t(\partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)) = \dot{\boldsymbol{p}}_t.$$

Using Hamilton's second equation and the identity $\partial_{\boldsymbol{s}} H = -\partial_{\boldsymbol{s}} L$, we obtain

$$\dot{\boldsymbol{p}}_t = -\partial_{\boldsymbol{s}} H(\boldsymbol{s}_t, \boldsymbol{p}_t) = \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t).$$

Therefore

$$\mathrm{d}_t(\partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)) - \partial_{\boldsymbol{s}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t) = 0,$$

which is precisely the Euler–Lagrange equation.

5. Bijection of trajectories. Steps 3 and 4 show that:

- Any trajectory $t \mapsto \boldsymbol{s}_t$ solving the Euler–Lagrange equation, together with $\boldsymbol{p}_t = \partial_{\dot{\boldsymbol{s}}} L(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t)$, yields a trajectory $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ solving Hamilton's equations.
- Any trajectory $(\boldsymbol{s}_t, \boldsymbol{p}_t)$ solving Hamilton's equations, together with $\dot{\boldsymbol{s}}_t = \partial_{\boldsymbol{p}} H(\boldsymbol{s}_t, \boldsymbol{p}_t)$, yields a trajectory $\boldsymbol{s}_t$ solving the Euler–Lagrange equation.

Combined with the bijection at the level of initial condition shown in step 1, this establishes the one-to-one correspondence between Lagrangian trajectories $\boldsymbol{s}_t$ and Hamiltonian trajectories $(\boldsymbol{s}_t, \boldsymbol{p}_t)$, together with their initial conditions. □

K.2. Constructing the invertible mapping between PFVP and HES

For readability, in this section we will omit the $\boldsymbol{\theta}$ dependence on the variable $\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0$ and $\boldsymbol{\alpha}_0^H$.

K.2.1. Free-phase and forward phase

From PFVP to HES. We now demonstrate how the forward phase of an HES can be constructed from the free phase of the PFVP. From proposition 4, we can express the free phase of the PFVP as a solution of a IVP:

$$\boldsymbol{s}^0_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) = \boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0)) \quad \text{for all } t \in [0, T].$$

where $\boldsymbol{\alpha}_T(\boldsymbol{\theta}) = \boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0))$ and $\boldsymbol{\gamma}_T(\boldsymbol{\theta}) = \dot{\boldsymbol{s}}^0_{\rightarrow,T}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0))$ (equation (13)). Applying the forward Legendre transformation of theorem 6 on this IVP we get the HES forward trajectory $\boldsymbol{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)$ that is a solution of the associated Hamilton equation of the IVP:

$$\forall t \in [0, T], \quad \boldsymbol{\Phi}_t\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right) := \begin{pmatrix} \boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0)) \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^0_{\rightarrow,t}, \dot{\boldsymbol{s}}^0_{\rightarrow,t}, \boldsymbol{\theta}\right) \end{pmatrix}, \quad \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix} := \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0, \boldsymbol{\theta}\right) \end{pmatrix}. \tag{33}$$

From HES to PFVP. To construct the forward phase we applied the two following transformations:

$$\underbrace{t \mapsto \boldsymbol{s}^0_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta})))}_{\text{PFVP free}} \xrightarrow[\text{proposition. 4}]{\text{PFVP}\to\text{IVP}} \underbrace{t \mapsto \boldsymbol{s}^0_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0))}_{\text{IVP free}} \xrightarrow[\text{theorem 6}]{\text{Legendre}} \underbrace{t \mapsto \boldsymbol{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)}_{\text{HES forward}}.$$

Since each of these two transformations is bijective, their composition is also a bijection. Hence the free phase of the PFVP can be constructed from the forward phase of the HES, and vice versa. Applying the inverse maps we get:

$$\forall t \in [0, T], \quad \begin{pmatrix} \boldsymbol{s}^0_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \\ \dot{\boldsymbol{s}}^0_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \end{pmatrix} := \begin{pmatrix} \boldsymbol{s}^0_t\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right) \\ \partial_{\boldsymbol{p}} H\left(\boldsymbol{\Phi}_t\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right), \boldsymbol{\theta}\right) \end{pmatrix},$$

where $\boldsymbol{s}^0_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)$ refers to the first vector component of $\boldsymbol{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)$, and $\partial_{\boldsymbol{p}} H$ means the derivative with respect to second vector component of $\boldsymbol{\Phi}_t(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top)$. Also, the initial condition of this PFVP are:

$$\begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\gamma}_0 \end{pmatrix} := \begin{pmatrix} \boldsymbol{\alpha}_0 \\ \partial_{\boldsymbol{p}} H(\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0, \boldsymbol{\theta}) \end{pmatrix}.$$

where $\begin{pmatrix} \boldsymbol{\alpha}_0 \\ \boldsymbol{\mu}_0 \end{pmatrix}$ with $\boldsymbol{\alpha}_0$ being the position and $\boldsymbol{\mu}_0$ being the momentum.

K.2.2. Nudged-phase and echo-phase

From PFVP to HES. We now show how the echo phase of the HES arises from the *nudged* PFVP. The nudged PFVP trajectory is defined by

$$t \mapsto \boldsymbol{s}^\beta_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))), \qquad t \in [0, T],$$

By proposition 2, this can be rewritten as a time translation of the IVP $t \mapsto \boldsymbol{s}^\beta_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta})))$:

$$\forall t \in [0, T], \qquad \boldsymbol{s}^\beta_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) = \boldsymbol{s}^\beta_{\rightarrow,T-t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta}))). \tag{34}$$

Applying the forward Legendre transform of theorem 6, to the nudged IVP yields the echo phase:

$$\forall t \in [0,T], \quad \boldsymbol{\Phi}_t^e\left(\boldsymbol{\theta}, \boldsymbol{\alpha}_0^{H,e}\right) := \begin{pmatrix} \boldsymbol{s}^{\beta}_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \\ \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}^{\beta}_{\rightarrow,t}, \dot{\boldsymbol{s}}^{\beta}_{\rightarrow,t}, \boldsymbol{\theta}\right) \end{pmatrix}, \quad \boldsymbol{\alpha}_0^{H,e} := \begin{pmatrix} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) \end{pmatrix}. \tag{35}$$

To get the full mapping to desired echo phase, we now show that $\boldsymbol{\alpha}_0^{H,e} = \boldsymbol{\Sigma}_z \boldsymbol{\Phi}_T$. We analyze the second component of $\boldsymbol{\alpha}_0^{H,e}$. By definition, it involves the term

$$\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right).$$

By lemma 2, we obtain

$$\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) = -\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right).$$

which gives:

$$\begin{aligned} \boldsymbol{\alpha}_0^{H,e} &= \begin{pmatrix} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ -\partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ -\partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) \end{pmatrix}. \quad \text{(lemma 5)} \end{aligned} \tag{36}$$

We now evaluate equation (33) at time $t = T$.

$$\boldsymbol{\Phi}_T\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right) = \begin{pmatrix} \boldsymbol{s}^0_{\rightarrow,T}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\gamma}_0)) \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^0_{\rightarrow,T}, \dot{\boldsymbol{s}}^0_{\rightarrow,T}, \boldsymbol{\theta}\right) \end{pmatrix}.$$

By the PFVP construction (equation (13)),

$$\boldsymbol{s}^0_{\rightarrow,T} = \boldsymbol{\alpha}_T(\boldsymbol{\theta}), \qquad \dot{\boldsymbol{s}}^0_{\rightarrow,T} = \boldsymbol{\gamma}_T(\boldsymbol{\theta}),$$

so that

$$\boldsymbol{\Phi}_T\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right) = \begin{pmatrix} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) \end{pmatrix}. \tag{37}$$

Taking this last equation (37) with equation (36), we have the final condition that makes the constructed echo-phase well-defined:

$$\boldsymbol{\alpha}_0^{H,e} = \boldsymbol{\Sigma}_z \boldsymbol{\Phi}_T\left(\boldsymbol{\theta}, (\boldsymbol{\alpha}_0, \boldsymbol{\mu}_0)^\top\right).$$

Rewriting our construction (equation (35)) in terms of PFVP variables, we have constructed $t \mapsto \boldsymbol{\Phi}_t^e(\boldsymbol{\theta}, \boldsymbol{\alpha}_0^{H,e})$ with:

$$\forall t \in [0,T], \quad \boldsymbol{\Phi}_t^e\left(\boldsymbol{\theta}, \boldsymbol{\alpha}_0^{H,e}\right) := \begin{pmatrix} \boldsymbol{s}^{\beta}_{\leftarrow,T-t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}))) \\ \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,T-t}, -\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,T-t}, \boldsymbol{\theta}\right) \end{pmatrix} \quad , \boldsymbol{\alpha}_0^{H,e} := \boldsymbol{\Sigma}_z \begin{pmatrix} \boldsymbol{\alpha}_T(\boldsymbol{\theta}) \\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta}), \boldsymbol{\theta}\right) \end{pmatrix}. \tag{38}$$

From HES to PFVP. To construct the echo phase, we applied the two following transformations:

$$\underbrace{t \mapsto \boldsymbol{s}^{\beta}_{\leftarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), \boldsymbol{\gamma}_T(\boldsymbol{\theta})))}_{\text{PFVP nudge}} \xrightarrow[\text{proposition 2}]{\text{PFVP}\to\text{IVP, time translation}} \underbrace{t \mapsto \boldsymbol{s}^{\beta}_{\rightarrow,t}(\boldsymbol{\theta}, (\boldsymbol{\alpha}_T(\boldsymbol{\theta}), -\boldsymbol{\gamma}_T(\boldsymbol{\theta})))}_{\text{IVP nudge}} \xrightarrow[\text{theorem 6}]{\text{Legendre}} \underbrace{t \mapsto \boldsymbol{\Phi}_t^e(\boldsymbol{\theta}, \boldsymbol{\Sigma}_z \boldsymbol{\Phi}_T)}_{\text{HES echo}}.$$

Since each of these two transformations is bijective, their composition is also a bijection. Hence the nudged phase of the PFVP can be constructed from the echo phase of the HES, and vice versa.

K.3. Gradient equivalence

We prove that the PFVP gradient estimator equals the RHEL gradient estimator by applying the forward Legendre transform. This direction of the proof leverages the trajectory correspondences already established in section K.2.

Starting Point: PFVP Gradient Estimator

The PFVP gradient estimator in Lagrangian variables is (from theorem 3):

$$\Delta^{\mathrm{PFVP}}(\beta,\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0) := \frac{1}{\beta}\Bigg[\underbrace{\int_0^T \left(\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right)\mathrm{d}t}_{\text{Integral term: }\boldsymbol{I}}$$
$$\underbrace{+\, d_{\boldsymbol{\theta}}\begin{pmatrix}\partial_{\dot{\boldsymbol{s}}} L_0(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta})\\ \boldsymbol{\alpha}_0\end{pmatrix}^\top \boldsymbol{\Sigma}_z \begin{pmatrix}\left(\boldsymbol{s}^\beta_{\leftarrow,0} - \boldsymbol{\alpha}_0\right)\\ \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,0},\dot{\boldsymbol{s}}^\beta_{\leftarrow,0},\boldsymbol{\theta}\right) - \partial_{\dot{\boldsymbol{s}}} L_0(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta})\end{pmatrix}}_{\text{Boundary term: }\boldsymbol{B}}\Bigg].$$

Our goal is to show that this gradient estimator is equivalent to the following one (theorem 2):

$$\Delta^{\mathrm{RHEL}}\left(\beta,\boldsymbol{\alpha}_0^H(\boldsymbol{\theta})\right) = -\frac{1}{\beta}\left(\int_0^T \left[\partial_{\boldsymbol{\theta}} H_\beta\left(\boldsymbol{\Phi}_t^e(\beta),\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} H_0\left(\boldsymbol{\Phi}_t,\boldsymbol{\theta}\right)\right]\mathrm{d}t - \left(\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0^H\right)^\top \boldsymbol{\Sigma}_x\left(\boldsymbol{\Phi}_T^e(\beta) - \boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0\right)\right).$$

Main Proof: Transforming PFVP to RHEL.

The proof relies on the trajectory correspondences established in section E.2. Rather than restating these correspondences, we will reference the relevant equations from E.2 as needed throughout the proof.

Step 1: Transform the Integral Term. We start with the integral term of PFVP:

$$\boldsymbol{I} = \int_0^T \left(\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right)\mathrm{d}t.$$

Step 1.1: Applying the Parameter-Gradient Relation.

To transform this integral, we use the parameter-gradient relation established in lemma 4: $\partial_{\boldsymbol{\theta}} H(\boldsymbol{\Phi}_t,\boldsymbol{\theta}) = -\partial_{\boldsymbol{\theta}} L(\boldsymbol{s}_t,\dot{\boldsymbol{s}}_t,\boldsymbol{\theta})$.

Recall from the beginning of Step 1:

$$\boldsymbol{I} = \int_0^T \left(\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right)\mathrm{d}t, \quad t\in[0,T].$$

By lemma 5, we have $\partial_{\boldsymbol{\theta}} L_\beta = \partial_{\boldsymbol{\theta}} L_0$. Thus:

$$\boldsymbol{I} = \int_0^T \left(\partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^\beta_{\leftarrow,t},\dot{\boldsymbol{s}}^\beta_{\leftarrow,t},\boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^0_{\leftarrow,t},\dot{\boldsymbol{s}}^0_{\leftarrow,t},\boldsymbol{\theta}\right)\right)\mathrm{d}t, \quad t\in[0,T].$$

To transform this to Hamiltonians, we recall the two equality from section E.2:

$$\boldsymbol{\Phi}_t^e = \begin{pmatrix}\boldsymbol{s}^\beta_{\leftarrow,T-t}\\ \partial_{\dot{\boldsymbol{s}}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,T-t},-\dot{\boldsymbol{s}}^\beta_{\leftarrow,T-t},\boldsymbol{\theta}\right)\end{pmatrix}, \quad t\in[0,T], \quad \text{(equation (38))}$$

$$\boldsymbol{\Phi}_t = \begin{pmatrix}\boldsymbol{s}^0_{\rightarrow,t}\\ \partial_{\dot{\boldsymbol{s}}} L_0\left(\boldsymbol{s}^0_{\rightarrow,t},\dot{\boldsymbol{s}}^0_{\rightarrow,t},\boldsymbol{\theta}\right)\end{pmatrix}, \quad t\in[0,T]. \quad \text{(equation (33))}$$

We apply lemma 4 to transform each term.

For the first term, we apply lemma 4 to the augmented system with Hamiltonian H_β and Lagrangian L_β at $\boldsymbol{\Phi}^e_{t'}$ with $t'\in[0,T]$. The Lagrangian trajectory corresponding to $\boldsymbol{\Phi}^e_{t'}$ is the IVP trajectory at time t', whose velocity is $-\dot{\boldsymbol{s}}^\beta_{\leftarrow,T-t'}$ (see equation (34)). Thus:

$$\partial_{\boldsymbol{\theta}} H_\beta\left(\boldsymbol{\Phi}^e_{t'},\boldsymbol{\theta}\right) = -\partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,T-t'},-\dot{\boldsymbol{s}}^\beta_{\leftarrow,T-t'},\boldsymbol{\theta}\right), \quad t'\in[0,T].$$

By lemma 5, we have $\partial_{\boldsymbol{\theta}} L_\beta = \partial_{\boldsymbol{\theta}} L_0$ and $\partial_{\boldsymbol{\theta}} H_\beta = \partial_{\boldsymbol{\theta}} H_0$, giving:

$$\partial_{\boldsymbol{\theta}} H_0\left(\boldsymbol{\Phi}^e_{t'},\boldsymbol{\theta}\right) = -\partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}^\beta_{\leftarrow,T-t'},-\dot{\boldsymbol{s}}^\beta_{\leftarrow,T-t'},\boldsymbol{\theta}\right), \quad t'\in[0,T].$$

Since L_0 is a reversible Lagrangian, i.e. $L_0(\boldsymbol{s},-\dot{\boldsymbol{s}},\boldsymbol{\theta})=L_0(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta})$ (see equation (12)), differentiating with respect to $\boldsymbol{\theta}$ gives $\partial_{\boldsymbol{\theta}}L_0(\boldsymbol{s},-\dot{\boldsymbol{s}},\boldsymbol{\theta})=\partial_{\boldsymbol{\theta}}L_0(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta})$. Change of variables $t'=T-t$ then gives:

$$\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^{\beta}_{\leftarrow,t},\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,t},\boldsymbol{\theta}\right)=-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{T-t},\boldsymbol{\theta}\right),\quad t\in[0,T].$$

For the second term, we apply lemma 4 to the non-augmented system with Hamiltonian H_0 and Lagrangian L_0 at $\boldsymbol{\Phi}_{t'}$ with $t'\in[0,T]$:

$$\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{t'},\boldsymbol{\theta}\right)=-\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^{0}_{\leftarrow,t'},\dot{\boldsymbol{s}}^{0}_{\leftarrow,t'},\boldsymbol{\theta}\right),\quad t'\in[0,T].$$

Therefore:

$$\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^{0}_{\leftarrow,t},\dot{\boldsymbol{s}}^{0}_{\leftarrow,t},\boldsymbol{\theta}\right)=-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{t},\boldsymbol{\theta}\right),\quad t\in[0,T].$$

Substituting both results into $\boldsymbol{I}$ for $t\in[0,T]$:

$$\begin{aligned}\boldsymbol{I}&=\int_0^T\left(-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{T-t},\boldsymbol{\theta}\right)-\left(-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{t},\boldsymbol{\theta}\right)\right)\right)\mathrm{d}t\\&=\int_0^T\left(-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{T-t},\boldsymbol{\theta}\right)+\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{t},\boldsymbol{\theta}\right)\right)\mathrm{d}t.\end{aligned}$$

Final change of variables: Let $t'=T-t$ so that $\mathrm{d}t'=-\mathrm{d}t$. When $t\in[0,T]$, we have $t'\in[T,0]$:

$$\begin{aligned}\boldsymbol{I}&=\int_T^0\left(\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{t'},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{T-t'},\boldsymbol{\theta}\right)\right)\left(-\mathrm{d}t'\right)\\&=-\int_0^T\left(\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{t},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{T-t},\boldsymbol{\theta}\right)\right)\mathrm{d}t\\&=-\int_0^T\left(\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}^{e}_{t},\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_{t},\boldsymbol{\theta}\right)\right)\mathrm{d}t.\end{aligned}$$

where the last equality uses the change of dummy integration variable $u=T-t$ in the second term only: $\int_0^T f(\boldsymbol{\Phi}_{T-t})\,\mathrm{d}t=\int_0^T f(\boldsymbol{\Phi}_u)\,\mathrm{d}u$.

This matches (up to sign) the integral term in RHEL.

Step 2: Transform the Boundary Term. The boundary term in PFVP (from theorem 3) is:

$$\boldsymbol{B}=d_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}^{\top}\boldsymbol{\Sigma}_x\begin{pmatrix}\left(\boldsymbol{s}^{\beta}_{\leftarrow,0}-\boldsymbol{\alpha}_0\right)\\-\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,0},\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,0},\boldsymbol{\theta}\right)+\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}.$$

Recall from section K.2 the mapping:

$$\begin{aligned}\boldsymbol{\Phi}^{e}_{T}&=\begin{pmatrix}\boldsymbol{s}^{\beta}_{\leftarrow,0}\\\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,0},-\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,0},\boldsymbol{\theta}\right)\end{pmatrix},\quad\text{(equation (38) at } t=T)\\\boldsymbol{\Phi}_0&=\begin{pmatrix}\boldsymbol{\alpha}_0\\\boldsymbol{\mu}_0\end{pmatrix}=\begin{pmatrix}\boldsymbol{\alpha}_0\\\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}.\quad\text{(equation (33) at } t=0)\end{aligned}\tag{39}$$

Therefore:

$$\begin{aligned}\boldsymbol{\Phi}^{e}_{T}-\boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0&=\begin{pmatrix}\boldsymbol{s}^{\beta}_{\leftarrow,0}-\boldsymbol{\alpha}_0\\\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,0},-\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,0},\boldsymbol{\theta}\right)+\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}\\&=\begin{pmatrix}\boldsymbol{s}^{\beta}_{\leftarrow,0}-\boldsymbol{\alpha}_0\\-\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^{\beta}_{\leftarrow,0},\dot{\boldsymbol{s}}^{\beta}_{\leftarrow,0},\boldsymbol{\theta}\right)+\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}.\quad\text{(lemma 2)}\end{aligned}\tag{40}$$

Also, from the initial condition $\begin{pmatrix}\boldsymbol{\alpha}_0\\\boldsymbol{\mu}_0\end{pmatrix}=\boldsymbol{\Phi}_0$ (equation (39)), we can deduce:

$$\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\\boldsymbol{\mu}_0\end{pmatrix}=\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\\boldsymbol{\mu}_0\end{pmatrix}=d_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}.\tag{41}$$

We now show that the RHEL boundary term equals $\boldsymbol{B}$. Starting from the RHEL boundary term:

$$\begin{aligned}\left(\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0^H\right)^\top \boldsymbol{\Sigma}_x\left(\boldsymbol{\Phi}_T^e-\boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0\right) &= \left(d_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}\right)^\top \boldsymbol{\Sigma}_x\left(\boldsymbol{\Phi}_T^e-\boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0\right) \quad \text{(substitute equation (41))}\\ &= \left(d_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}\right)^\top \boldsymbol{\Sigma}_x\begin{pmatrix}\boldsymbol{s}^\beta_{\leftarrow,0}-\boldsymbol{\alpha}_0\\ -\partial_{\dot{\boldsymbol{s}}}L_\beta\left(\boldsymbol{s}^\beta_{\leftarrow,0},\dot{\boldsymbol{s}}^\beta_{\leftarrow,0},\boldsymbol{\theta}\right)+\partial_{\dot{\boldsymbol{s}}}L_0\left(\boldsymbol{\alpha}_0,\boldsymbol{\gamma}_0,\boldsymbol{\theta}\right)\end{pmatrix}\\ &\quad \text{(substitute equation (40))}\\ &= \boldsymbol{B}. \quad \text{(matches the PFVP boundary term)}\end{aligned}$$

This shows the boundary terms match exactly.

Step 3: Combine and Conclude. Combining both terms from Step 1 and Step 2, we have:

$$\begin{aligned}\Delta^{\mathrm{PFVP}}(\beta) &= \frac{1}{\beta}(\boldsymbol{I}+\boldsymbol{B})\\ &= \frac{1}{\beta}\left(-\int_0^T\left(\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t^e,\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t,\boldsymbol{\theta}\right)\right)\mathrm{d}t+\left(\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \boldsymbol{\mu}_0\end{pmatrix}\right)^\top\boldsymbol{\Sigma}_x\left(\boldsymbol{\Phi}_T^e(\beta)-\boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0\right)\right)\\ &= -\frac{1}{\beta}\left(\int_0^T\left(\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t^e,\boldsymbol{\theta}\right)-\partial_{\boldsymbol{\theta}}H_0\left(\boldsymbol{\Phi}_t,\boldsymbol{\theta}\right)\right)\mathrm{d}t-\left(\partial_{\boldsymbol{\theta}}\begin{pmatrix}\boldsymbol{\alpha}_0\\ \boldsymbol{\mu}_0\end{pmatrix}\right)^\top\boldsymbol{\Sigma}_x\left(\boldsymbol{\Phi}_T^e(\beta)-\boldsymbol{\Sigma}_z\boldsymbol{\Phi}_0\right)\right)\\ &= \Delta^{\mathrm{RHEL}}\left(\beta,\boldsymbol{\alpha}_0(\boldsymbol{\theta}),\boldsymbol{\mu}_0(\boldsymbol{\theta})\right).\end{aligned}$$

Appendix L. Dissipative LEP

This appendix presents the general theory of dissipative LEP, including the proof of the main theorem and the energy dissipation property. The harmonic oscillator instantiation is presented in the following section as a concrete example.

L.1. Proof of theorem 5: dissipative LEP

Proof. We first derive the Euler–Lagrange equation (18), then apply theorem 3.

Step 1: Derivation of the dissipative Euler–Lagrange equation. The standard Euler–Lagrange equation for L_β^{diss} is:

$$\partial_{\boldsymbol{s}}L_\beta^{\mathrm{diss}}-d_t\partial_{\dot{\boldsymbol{s}}}L_\beta^{\mathrm{diss}}=0.$$

Since $c(\boldsymbol{s}_t,\boldsymbol{y}_t)$ does not depend on $\dot{\boldsymbol{s}}_t$, the velocity derivative is:

$$\partial_{\dot{\boldsymbol{s}}}L_\beta^{\mathrm{diss}}=\exp(\zeta t)\cdot\partial_{\dot{\boldsymbol{s}}}L_0.$$

Taking the time derivative using the product rule:

$$\begin{aligned}d_t\partial_{\dot{\boldsymbol{s}}}L_\beta^{\mathrm{diss}} &= \zeta\exp(\zeta t)\cdot\partial_{\dot{\boldsymbol{s}}}L_0+\exp(\zeta t)\cdot d_t\left(\partial_{\dot{\boldsymbol{s}}}L_0\right)\\ &= \exp(\zeta t)\cdot\left(\zeta\,\partial_{\dot{\boldsymbol{s}}}L_0+d_t\partial_{\dot{\boldsymbol{s}}}L_0\right).\end{aligned}$$

The position derivative is:

$$\partial_{\boldsymbol{s}}L_\beta^{\mathrm{diss}}=\exp(\zeta t)\cdot\partial_{\boldsymbol{s}}L_0+\beta\,\partial_{\boldsymbol{s}}c.$$

Substituting into the Euler–Lagrange equation $\partial_{\boldsymbol{s}}L_\beta^{\mathrm{diss}}-d_t\partial_{\dot{\boldsymbol{s}}}L_\beta^{\mathrm{diss}}=0$ and multiplying through by $\exp(-\zeta t)$ yields (18).

Step 2: Physical interpretation (free phase). For $\beta=0$, dividing by $\exp(\zeta t)\neq 0$:

$$\partial_{\boldsymbol{s}}L_0-d_t\partial_{\dot{\boldsymbol{s}}}L_0=\zeta\,\partial_{\dot{\boldsymbol{s}}}L_0.$$

This shows that the effect of the exponential time-scaling is to add a friction-like term proportional to $\partial_{\dot{\boldsymbol{s}}}L_0$ to the standard Euler–Lagrange equation. When the Lagrangian has quadratic kinetic energy ($\partial_{\dot{\boldsymbol{s}}}L_0=\dot{\boldsymbol{s}}$), this reduces to Newton's second law with viscous friction $\mathbf{F}_{\mathrm{friction}}=-\zeta\dot{\boldsymbol{s}}$.

Step 3: Application of theorem 3. Since $\partial_{\boldsymbol{\theta}} L_\beta^{\text{diss}} = \partial_{\boldsymbol{\theta}} L_0 \cdot \exp(\zeta t)$ (the cost c does not depend on $\boldsymbol{\theta}$), the integral term in the PFVP gradient estimator becomes:

$$\int_0^T \left(\partial_{\boldsymbol{\theta}} L_0 \left(\boldsymbol{s}_{\leftarrow,t}^{\beta}, \dot{\boldsymbol{s}}_{\leftarrow,t}^{\beta}, \boldsymbol{\theta}\right) - \partial_{\boldsymbol{\theta}} L_0 \left(\boldsymbol{s}_{\leftarrow,t}^{0}, \dot{\boldsymbol{s}}_{\leftarrow,t}^{0}, \boldsymbol{\theta}\right)\right) \exp(\zeta t)\, \mathrm{d}t.$$

For the boundary terms at $t = 0$, we have $\partial_{\dot{\boldsymbol{s}}} L_\beta^{\text{diss}} = \partial_{\dot{\boldsymbol{s}}} L_0 \cdot \exp(\zeta \cdot 0) = \partial_{\dot{\boldsymbol{s}}} L_0$, so they remain unchanged from theorem 3. The PFVP-to-IVP reduction (proposition 2) generalizes to the dissipative setting: since the undamped Lagrangian L_0 is time-reversible, the bouncing-backward kick applies with the replacement $\zeta \to -\zeta$ in the echo phase, corresponding to energy pumping during the nudged backward trajectory (see proposition 6).

Remark (Exponential weighting): The factor $\exp(\zeta t)$ weights later time steps exponentially more than earlier ones. □

L.2. Proof of proposition 5: energy dissipation

Proposition 5 (Energy Dissipation). *Consider the isolated dissipative system ($\boldsymbol{x}_t = 0$). For a trajectory $t \mapsto \boldsymbol{s}_t$ satisfying the dissipative Euler–Lagrange equation (18) with $\beta = 0$ and $\boldsymbol{x}_t = 0$, the physical energy E (defined as in (16)) evolves according to:*

$$d_t E = -\zeta\, \dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}. \tag{42}$$

Quadratic kinetic energy case: *When the Lagrangian admits a decomposition $L_0^{\text{iso}} = E_{\text{kin}}(\dot{\boldsymbol{s}}_t) - U_{\text{int}}(\boldsymbol{s}_t, \boldsymbol{\theta})$ with quadratic kinetic energy $E_{\text{kin}}(\dot{\boldsymbol{s}}_t) = \frac{1}{2}\|\dot{\boldsymbol{s}}_t\|^2$, we have $\partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} = \dot{\boldsymbol{s}}_t$, yielding:*

$$d_t E = -\zeta \|\dot{\boldsymbol{s}}_t\|^2 \leqslant 0. \tag{43}$$

Since $\zeta > 0$, energy is strictly dissipated whenever $\dot{\boldsymbol{s}}_t \neq 0$.

Proof. Starting from the energy definition (16):

$$E = \dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - L_0^{\text{iso}}.$$

Taking the time derivative:

$$\begin{aligned} d_t E &= \ddot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} + \dot{\boldsymbol{s}}_t^\top d_t \left(\partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}\right) - d_t L_0^{\text{iso}} \\ &= \ddot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} + \dot{\boldsymbol{s}}_t^\top d_t \left(\partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}\right) - \partial_{\boldsymbol{s}} L_0^{\text{iso}} \cdot \dot{\boldsymbol{s}}_t - \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} \cdot \ddot{\boldsymbol{s}}_t \\ &= \dot{\boldsymbol{s}}_t^\top \left(d_t \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - \partial_{\boldsymbol{s}} L_0^{\text{iso}}\right), \end{aligned}$$

where the first two terms (with $\ddot{\boldsymbol{s}}_t$) cancel, and the chain rule gives $d_t L_0^{\text{iso}} = \partial_{\boldsymbol{s}} L_0^{\text{iso}} \cdot \dot{\boldsymbol{s}}_t + \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} \cdot \ddot{\boldsymbol{s}}_t$.

For the isolated system ($\boldsymbol{x}_t = 0$) with $\beta = 0$, the dissipative Euler–Lagrange equation (18) reduces to:

$$\partial_{\boldsymbol{s}} L_0^{\text{iso}} - d_t \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - \zeta\, \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} = 0.$$

Rearranging:

$$d_t \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - \partial_{\boldsymbol{s}} L_0^{\text{iso}} = -\zeta\, \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}.$$

Substituting into the energy evolution expression:

$$d_t E = \dot{\boldsymbol{s}}_t^\top \left(-\zeta\, \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}\right) = -\zeta\, \dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}}.$$

This proves equation (42).

For the quadratic kinetic energy case where $L_0^{\text{iso}} = \frac{1}{2}\|\dot{\boldsymbol{s}}_t\|^2 - U_{\text{int}}(\boldsymbol{s}_t, \boldsymbol{\theta})$, we have:

$$\partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} = \dot{\boldsymbol{s}}_t.$$

Therefore:

$$d_t E = -\zeta\, \dot{\boldsymbol{s}}_t^\top \dot{\boldsymbol{s}}_t = -\zeta \|\dot{\boldsymbol{s}}_t\|^2 \leqslant 0.$$

Since $\zeta > 0$, this shows that energy is strictly dissipated (decreases) whenever $\dot{\boldsymbol{s}}_t \neq 0$, confirming the physically expected behavior of a dissipative system. □

Appendix M. Dissipative Harmonic Oscillators: Complete Derivation

This appendix provides the complete derivation of the dissipative harmonic oscillator system summarized in table 4 of section 6.3.

M.1. Derivation of free and nudged dynamics

Lagrangian and dissipative formulation. The physical Lagrangian is given by equation (20):

$$L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t) = \frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t - \frac{1}{2}\boldsymbol{s}_t^\top \boldsymbol{K}\boldsymbol{s}_t - \boldsymbol{e}_1^\top \boldsymbol{s}_t x_t,$$

where the kinetic energy uses the mass vector $\boldsymbol{m}$ with element-wise operations, and the potential energy uses the dense symmetric stiffness matrix $\boldsymbol{K}$ that couples all oscillators. The input coupling term $-\boldsymbol{e}_1^\top \boldsymbol{s}_t x_t = -s_{1,t} x_t$ describes the external force acting on the first oscillator.

Following the dissipative Lagrangian formulation (17), we use a scalar damping coefficient $\zeta > 0$. This gives the dissipative Lagrangian:

$$L_\beta^{\text{diss}}(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t, y_t) = \exp(\zeta t) \cdot L_0(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t) + \beta\, c(\boldsymbol{s}_t, y_t),$$

with cost function $c(\boldsymbol{s}_t, y_t) = \frac{1}{2}(s_{d,t} - y_t)^2$, where $s_{d,t}$ denotes the dth component of $\boldsymbol{s}_t$ (the last oscillator).

Free dynamics ($\beta = 0$). Applying the dissipative Euler–Lagrange equation (18), the free dynamics are:

$$\partial_{\boldsymbol{s}} L_0 - d_t \partial_{\dot{\boldsymbol{s}}} L_0 - \zeta\, \partial_{\dot{\boldsymbol{s}}} L_0 = \boldsymbol{0}.$$

Computing the gradients:

$$\partial_{\boldsymbol{s}} L_0 = -\boldsymbol{K}\boldsymbol{s}_t - x_t \boldsymbol{e}_1, \qquad \partial_{\dot{\boldsymbol{s}}} L_0 = \boldsymbol{m} \odot \dot{\boldsymbol{s}}_t.$$

Defining the element-wise damping vector $\boldsymbol{\gamma} := \zeta \boldsymbol{m} = (\zeta m_1, \ldots, \zeta m_d)^\top$, this yields the driven damped coupled harmonic oscillator equations:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t = -x_t \boldsymbol{e}_1.$$

This recovers the well-known damped harmonic oscillator equation with proportional damping (damping force proportional to mass with uniform coefficient ζ).

Nudged dynamics ($\beta > 0$). With the cost function term acting on the last oscillator, applying the dissipative Euler–Lagrange equation gives:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t^\beta + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t^\beta + \boldsymbol{K}\boldsymbol{s}_t^\beta = -x_t \boldsymbol{e}_1 - \beta \exp(-\zeta t)\, \boldsymbol{e}_d \left(s_{d,t}^\beta - y_t\right),$$

where $\boldsymbol{e}_d = (0, \ldots, 0, 1)^\top$ selects the last oscillator where the cost is applied. Note the exponential factor $\exp(-\zeta t)$ in the nudging term, which arises from the dissipative Lagrangian formulation and ensures that the nudging strength is properly weighted along the time-scaled trajectory.

M.2. Time-reversibility and Parametric Final Value Problem (PFVP) implementation

As in Lagrangian EP, both the free and nudged phases are formulated as *PFVPs*, where the final conditions at time T are parametrically determined by $\boldsymbol{\theta}$, while the initial conditions are fixed.

Free phase: The free dynamics are solved as a standard initial value problem, integrating forward in time from $t = 0$ to $t = T$ with fixed initial conditions $(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) = (\boldsymbol{\alpha}_0, \boldsymbol{0})$:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t^0 + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t^0 + \boldsymbol{K}\boldsymbol{s}_t^0 = -x_t \boldsymbol{e}_1, \quad t \in [0, T].$$

This yields the free trajectory and determines the final conditions $(\boldsymbol{s}_T^0, \dot{\boldsymbol{s}}_T^0)$.

Nudged phase: The nudged dynamics are formulated as a final value problem. To implement the PFVP condition that both free and nudged trajectories share the same final state, we solve the nudged dynamics *backward in time* from $t = T$ to $t = 0$, starting from the final conditions $(\boldsymbol{s}_T^\beta, \dot{\boldsymbol{s}}_T^\beta) = (\boldsymbol{s}_T^0, \dot{\boldsymbol{s}}_T^0)$.

The critical implementation detail is given by the following proposition:

Proposition 6 (Time-reversibility of dissipative PFVP). *Consider the dissipative dynamics with damping vector $\boldsymbol{\gamma} = \zeta \boldsymbol{m}$ (where $\zeta > 0$ is scalar), mass vector $\boldsymbol{m}$, and stiffness matrix $\boldsymbol{K}$:*

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t = \boldsymbol{f}(t)\,,$$

where $\boldsymbol{f}(t) \in \mathbb{R}^d$ is an external forcing term. The solution of the PFVP with final conditions $(\boldsymbol{s}_T, \dot{\boldsymbol{s}}_T)$ can be computed by integrating forward in time $t' \in [0, T]$ the Initial Value Problem with velocity-reversed initial conditions $(\boldsymbol{s}_T, -\dot{\boldsymbol{s}}_T)$ where the dissipative term changes sign:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_{t'} - \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_{t'} + \boldsymbol{K}\boldsymbol{s}_{t'} = \boldsymbol{f}(T - t')\,, \quad t' \in [0, T]\,,$$

with initial conditions $(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) = (\boldsymbol{s}_T, -\dot{\boldsymbol{s}}_T)$. The PFVP solution at physical time t is given by $\boldsymbol{s}_t = \boldsymbol{s}_{t'}$ where $t' = T - t$.

Proof. See appendix M.5. □

Applying proposition 6 to the nudged dynamics, we integrate *forward in time* $t' \in [0, T]$ starting from the velocity-reversed final conditions $(\boldsymbol{s}^\beta_T, -\dot{\boldsymbol{s}}^\beta_T) = (\boldsymbol{s}^0_T, -\dot{\boldsymbol{s}}^0_T)$:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}^\beta_{t'} - \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}^\beta_{t'} + \boldsymbol{K}\boldsymbol{s}^\beta_{t'} = -x_{T-t'}\boldsymbol{e}_1 - \beta \exp\left(-\zeta\,(T - t')\right) \boldsymbol{e}_d \left(s^\beta_{d,t'} - y_{T-t'}\right), \quad t' \in [0, T]\,.$$

Crucially, this is an Initial Value Problem that is integrated *forward* in integration time t' from 0 to T (corresponding to physical time t going backward from T to 0). The inputs $x_{T-t'}$ and targets $y_{T-t'}$ are fed in reverse temporal order: at integration time t', we use the input and target from physical time $T - t'$.

Physical interpretation: The sign flip has a natural physical interpretation. When we run a dissipative system forward in time, energy is dissipated and the system loses energy through friction (see equation (43), where the term $-\zeta\|\dot{\boldsymbol{s}}_t\|^2 < 0$ represents energy dissipation). When we run the nudge phase backward, the friction term must reverse its action—effectively *adding* energy back into the system (as $-\zeta$ becomes $+\zeta$, making the term positive).

M.3. Gradient estimator with fixed initial conditions

For fixed initial conditions $\boldsymbol{s}_0 = \boldsymbol{\alpha}_0$ (independent of $\boldsymbol{\theta}$) and zero initial velocity $\dot{\boldsymbol{s}}_0 = \boldsymbol{0}$, the gradient estimator from theorem 5 simplifies. The boundary terms in (19) cancel because:

- At $t = 0$: The initial conditions are fixed ($\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0 = \boldsymbol{0}$, $\partial_{\boldsymbol{\theta}}\boldsymbol{\gamma}_0 = \boldsymbol{0}$), so the boundary term involving $(\partial_{\boldsymbol{\theta}}\boldsymbol{\alpha}_0)^\top$ vanishes. The term $(\mathrm{d}_{\boldsymbol{\theta}}\partial_{\dot{\boldsymbol{s}}}L_0)^\top(\boldsymbol{s}^\beta_0 - \boldsymbol{\alpha}_0)$ also vanishes since both trajectories start from the same initial position ($\boldsymbol{s}^\beta_0 = \boldsymbol{s}^0_0 = \boldsymbol{\alpha}_0$).
- At $t = T$: With the PFVP formulation, the final conditions of both free and nudged trajectories are matched, so $(\boldsymbol{s}^\beta_T - \boldsymbol{s}^0_T) = \boldsymbol{0}$ and $(\dot{\boldsymbol{s}}^\beta_T - \dot{\boldsymbol{s}}^0_T) = \boldsymbol{0}$, eliminating any final boundary contributions.

Therefore, only the integral term remains:

$$\mathrm{d}_{\boldsymbol{\theta}}\mathcal{C}\left[\boldsymbol{s}^0(\boldsymbol{\theta})\right] = \lim_{\beta\to 0}\frac{1}{\beta}\int_0^T \left[\partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^\beta_t, \dot{\boldsymbol{s}}^\beta_t, \boldsymbol{\theta}, x_t\right) - \partial_{\boldsymbol{\theta}}L_0\left(\boldsymbol{s}^0_t, \dot{\boldsymbol{s}}^0_t, \boldsymbol{\theta}, x_t\right)\right] \exp(\zeta t)\,\mathrm{d}t.$$

The parameter gradients of L_0 are:

$$\begin{aligned}
\partial_{m_i}L_0 &= \frac{1}{2}\dot{s}^2_{i,t} \quad (\text{for each mass} i = 1, \ldots, \mathrm{d}) \\
\partial_{\mathbf{K}}L_0 &= -\frac{1}{2}\boldsymbol{s}_t\boldsymbol{s}_t^\top \quad (\text{yields a } \mathrm{d}\times\mathrm{d} \text{ matrix}) \\
\partial_{\zeta}L_0 &= t \cdot L_0\left(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, x_t\right).
\end{aligned}$$

The damping coefficient gradient involves the full Lagrangian weighted by time t, reflecting how damping affects the exponential time-weighting factor $\exp(\zeta t)$ in the dissipative formulation.

This gradient estimator provides an unbiased estimate of $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{C}$ by comparing the time-weighted Lagrangian along free and nudged trajectories, without requiring any boundary term corrections.

M.4. Energy evolution for Harmonic oscillators

We derive the explicit energy evolution for the dissipative harmonic oscillator system. Following section 6, we define the physical energy with respect to the isolated Lagrangian L_0^{iso} (obtained by setting $x_t = 0$).

Physical energy definition. For the harmonic oscillator, the isolated Lagrangian is:

$$L_0^{\text{iso}}(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t - \frac{1}{2}\boldsymbol{s}_t^\top \boldsymbol{K}\boldsymbol{s}_t.$$

The physical energy E (as defined in (16)) is:

$$E(t) = \dot{\boldsymbol{s}}_t^\top \partial_{\dot{\boldsymbol{s}}} L_0^{\text{iso}} - L_0^{\text{iso}} = \underbrace{\frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t}_{E_{\text{kin}}(t)} + \underbrace{\frac{1}{2}\boldsymbol{s}_t^\top \boldsymbol{K}\boldsymbol{s}_t}_{U_{\text{int}}(t)}.$$

This is the standard mechanical energy: kinetic energy plus internal potential energy.

Proposition 7 (Energy evolution for dissipative harmonic oscillators). *For the harmonic oscillator system with proportional damping $\boldsymbol{\gamma} = \zeta \boldsymbol{m}$, the physical energy $E(t) = E_{\text{kin}}(t) + U_{\text{int}}(t)$ evolves as:*

$$E(t) = E(0) + \underbrace{\left(-\int_0^t \dot{s}_{1,\tau}\, x_\tau \, \mathrm{d}\tau\right)}_{W_{\text{input}}(t)} - \underbrace{\int_0^t \boldsymbol{\gamma} \cdot \dot{\boldsymbol{s}}_\tau^2 \, \mathrm{d}\tau}_{D_{\text{diss}}(t)}, \tag{44}$$

where $\dot{\boldsymbol{s}}_\tau^2 = \dot{\boldsymbol{s}}_\tau \odot \dot{\boldsymbol{s}}_\tau$ denotes element-wise squaring.
Equivalently, using $E_{\text{kin}}(\tau) = \frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_\tau) \cdot \dot{\boldsymbol{s}}_\tau$ and $\boldsymbol{\gamma} = \zeta \boldsymbol{m}$:

$$E(t) = E(0) + W_{\text{input}}(t) - 2\zeta \int_0^t E_{\text{kin}}(\tau)\, \mathrm{d}\tau.$$

The energy contributions are:

- ***Input work*** $W_{\text{input}}(t) = -\int_0^t \dot{s}_{1,\tau}\, x_\tau \, \mathrm{d}\tau$*: Work done by the external force $F_{\text{ext}} = -x_t$ on the first oscillator. This follows the standard mechanics formula: power = force × velocity = $(-x_t) \cdot \dot{s}_{1,t}$. Can be positive (energy injection) or negative (energy extraction) depending on the correlation between velocity $\dot{s}_{1,\tau}$ and force $-x_\tau$.*
- ***Dissipation*** $D_{\text{diss}}(t) = \int_0^t \boldsymbol{\gamma} \cdot \dot{\boldsymbol{s}}_\tau^2 \, \mathrm{d}\tau = 2\zeta \int_0^t E_{\text{kin}}(\tau)\, \mathrm{d}\tau \geqslant 0$*: Energy dissipated by friction, proportional to the time-integrated kinetic energy. Always removes energy.*

Proof. We derive the energy evolution for the free phase ($\beta = 0$) from first principles.

Step 1: Energy definition. Following section 6, the physical energy is defined with respect to the isolated Lagrangian:

$$E = E_{\text{kin}} + U_{\text{int}} = \frac{1}{2}(\boldsymbol{m} \odot \dot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t + \frac{1}{2}\boldsymbol{s}_t^\top \boldsymbol{K}\boldsymbol{s}_t.$$

Step 2: Time derivative of E. Taking the total time derivative:

$$\begin{aligned} d_t E &= d_t E_{\text{kin}} + d_t U_{\text{int}} \\ &= (\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t) \cdot \dot{\boldsymbol{s}}_t + \boldsymbol{s}_t^\top \boldsymbol{K}\dot{\boldsymbol{s}}_t \\ &= \dot{\boldsymbol{s}}_t^\top (\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t). \end{aligned} \tag{45}$$

Step 3: Using the equations of motion. For the dissipative harmonic oscillator (free phase with $\beta = 0$), the equation of motion is:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t = -x_t \boldsymbol{e}_1.$$

Rearranging:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t = -x_t \boldsymbol{e}_1 - \boldsymbol{\gamma} \odot \dot{\boldsymbol{s}}_t. \tag{46}$$

Step 4: Final expression for d_tE. Substituting (46) into (45):

$$\begin{aligned} d_tE &= \dot{\boldsymbol{s}}_t^\top \left(-x_t\boldsymbol{e}_1 - \boldsymbol{\gamma}\odot\dot{\boldsymbol{s}}_t\right) \\ &= -x_t\dot{s}_{1,t} - \boldsymbol{\gamma}\cdot\dot{\boldsymbol{s}}_t^2, \end{aligned}$$

where $\dot{\boldsymbol{s}}_t^2 = \dot{\boldsymbol{s}}_t\odot\dot{\boldsymbol{s}}_t$ denotes element-wise squaring.

Equivalently, using $E_{\text{kin}}(t) = \frac{1}{2}(\boldsymbol{m}\odot\dot{\boldsymbol{s}}_t)\cdot\dot{\boldsymbol{s}}_t$ and $\boldsymbol{\gamma} = \zeta\boldsymbol{m}$:

$$d_tE = -x_t\dot{s}_{1,t} - 2\zeta\, E_{\text{kin}}(t)\,. \tag{47}$$

Physical interpretation. The energy $E = E_{\text{kin}} + U_{\text{int}}$ evolves with two power contributions:

- $P_{\text{input}} = -x_t\dot{s}_{1,t} = F_{\text{ext}}\cdot\dot{s}_{1,t}$: Power delivered by the external force $F_{\text{ext}} = -x_t$ acting on the first oscillator. This follows the standard mechanics formula: power = force × velocity. When the force and velocity are aligned (same sign), energy is injected; when opposed, energy is extracted.
- $P_{\text{diss}} = \boldsymbol{\gamma}\cdot\dot{\boldsymbol{s}}_t^2 = 2\zeta\, E_{\text{kin}}(t)\geqslant 0$: Power dissipated by friction (always positive, always removes energy from the system).

Integrating (47) from 0 to t yields the energy evolution (44):

$$E(t) - E(0) = -\int_0^t x_\tau\,\dot{s}_{1,\tau}\,\mathrm{d}\tau - \int_0^t \boldsymbol{\gamma}\cdot\dot{\boldsymbol{s}}_\tau^2\,\mathrm{d}\tau.$$

Special case: isolated system. When $x_t = 0$ (no external input), the energy evolution simplifies to:

$$d_tE = -\boldsymbol{\gamma}\cdot\dot{\boldsymbol{s}}_t^2 = -2\zeta\, E_{\text{kin}}(t)\leqslant 0.$$

This confirms that dissipation always removes energy from the system, as stated in proposition 5. □

M.5. Proof of proposition 6: time-reversal for dissipative systems

Proof of proposition 6. Consider the dissipative dynamics in the forward time direction. For simplicity, we present the proof for a single component (the multi-dimensional case follows by applying the same argument component-wise):

$$m\ddot{s}_t + \zeta m\dot{s}_t + ks_t = f(t)\,, \quad t\in[0,T]\,, \tag{48}$$

where m, ζ, and k are scalar parameters, and the damping is proportional to the mass.

To solve this equation backward in time from $t = T$ to $t = 0$ as a final value problem, we introduce the backward time parameter $t' = T - t$. As t runs from T to 0, t' runs from 0 to T.

Change of variables. Under the substitution $t = T - t'$, we have:

$$\begin{aligned} s_t = s_{T-t'} &\equiv \overleftarrow{s}_{t'} \\ d_t &= -d_{t'} \\ d_t^2 &= d_{t'}^2. \end{aligned}$$

First derivative transformation. The first time derivative transforms as:

$$\dot{s}_t = d_ts_t = d_t\overleftarrow{s}_{t'} = d_{t'}\overleftarrow{s}_{t'}\cdot d_tt' = -d_{t'}\overleftarrow{s}_{t'} = -\overleftarrow{\dot{s}}_{t'},$$

where $\overleftarrow{\dot{s}}_{t'} := d_{t'}\overleftarrow{s}_{t'}$.

Second derivative transformation. The second time derivative transforms as:

$$\begin{aligned} \ddot{s}_t = d_t^2s_t = d_t\left(d_ts_t\right) &= d_t\left(-\overleftarrow{\dot{s}}_{t'}\right) \\ &= -d_t\overleftarrow{\dot{s}}_{t'} = -d_{t'}\overleftarrow{\dot{s}}_{t'}\cdot d_tt' \\ &= -\overleftarrow{\ddot{s}}_{t'}\cdot(-1) = \overleftarrow{\ddot{s}}_{t'}. \end{aligned}$$

Equation transformation. Substituting these transformations into (48):

$$\overleftarrow{m\ddot{s}}_{t'} + \zeta m\left(-\overleftarrow{\dot{s}}_{t'}\right) + k\overleftarrow{s}_{t'} = f(T-t')$$
$$\overleftarrow{m\ddot{s}}_{t'} - \zeta m\overleftarrow{\dot{s}}_{t'} + k\overleftarrow{s}_{t'} = f(T-t'), \quad t' \in [0,T].$$

This establishes that the dissipative term $\zeta m\dot{s}_t$ changes sign to $-\zeta m\overleftarrow{\dot{s}}_{t'}$ when we transform to backward time, while the second-order term $m\ddot{s}_t$ remains unchanged (since it involves an even number of time derivatives).

Extension to vector case and IVP formulation. For the multi-dimensional case with mass vector $\boldsymbol{m}$, damping vector $\boldsymbol{\gamma} = \zeta\boldsymbol{m}$ (where $\zeta > 0$ is scalar), and stiffness matrix $\boldsymbol{K}$, the same time-reversal transformation applies component-wise. Following the same derivation as above with $\boldsymbol{s}_{\leftarrow,t'} := \boldsymbol{s}_{T-t'}$, we obtain the time-reversed equation. For the actual IVP formulation, we denote the solution trajectory simply as $\boldsymbol{s}_{t'}$ (dropping the tilde notation), which satisfies:

$$\boldsymbol{m}\odot\ddot{\boldsymbol{s}}_{t'} - \boldsymbol{\gamma}\odot\dot{\boldsymbol{s}}_{t'} + \boldsymbol{K}\boldsymbol{s}_{t'} = \boldsymbol{f}(T-t'), \quad t' \in [0,T].$$

Note that only the dissipative term (the damping force $\boldsymbol{\gamma}\odot\dot{\boldsymbol{s}}_t$) changes sign, while the stiffness term $\boldsymbol{K}\boldsymbol{s}_{t'}$ (a matrix-vector product) remains unchanged.

Physical interpretation. The sign change of the dissipative term under time reversal reflects the fact that dissipation is time-irreversible: in forward time, friction removes energy from the system ($-\gamma\dot{s}_t$ opposes the velocity), while in backward time, the effective friction must add energy back into the system to reconstruct trajectories consistent with the forward dynamics.

Velocity-reversed initial conditions. To solve the PFVP with final conditions $(\boldsymbol{s}_T, \dot{\boldsymbol{s}}_T)$, we use the IVP in the t' time coordinate with initial conditions:

$$(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) = (\boldsymbol{s}_T, -\dot{\boldsymbol{s}}_T).$$

Note the crucial sign flip on the initial velocity vector. This ensures that when we integrate forward in t' with the sign-flipped dissipative term, we reconstruct the trajectory that would have led to the desired final conditions in the original time coordinate t. □

Appendix N. Electrical-Circuit Implementation of Dissipative LEP

This appendix gives a feasibility construction for the damped coupled oscillators of section 6.3: it maps them onto an electrical circuit and describes idealized switching operations for reversing the dissipation and, separately, the velocity at $t = T$. The construction is elementary and self-contained, relying only on Kirchhoff's circuit laws and on the ideal closed-loop relations for operational amplifiers,

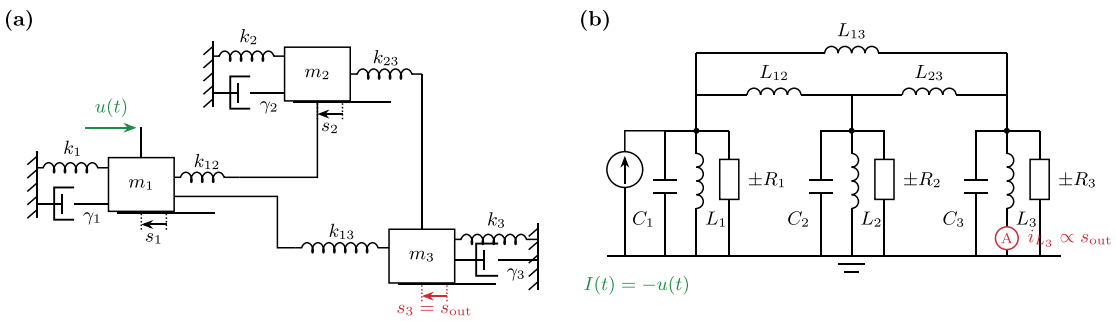


Figure 7. Electrical twin of the coupled damped oscillators. (a) The conservative mechanical network of figure 1 with one horizontal damper γ_i added in parallel with each grounding spring k_i. The three pairwise coupling springs k_{ij} retain the same all-to-all topology; $u(t)$ drives m_1 and $s_{\text{out}} = s_3$ is read from m_3. (b) The corresponding circuit adds one sign-switchable resistor branch $\pm R_i$ to each conservative C_i–L_i node cell, with $\gamma_i = 1/R_i$ and $\zeta = 1/(R_iC_i)$. The passive $+R_i$ branch dissipates energy during the free phase, whereas a negative resistance $-R_i$ synthesized by one op-amp returns that energy during the echo phase. At $t = T$, the idealized circuit uses a four-switch polarity-reversing bridge on every mass capacitor C_i to reverse $V_i \leftrightarrow \dot{s}_i$ while leaving the inductor currents $i_{L_i} \leftrightarrow s_i$ unchanged; appendix N discusses the ideal operation and physical switching transients.

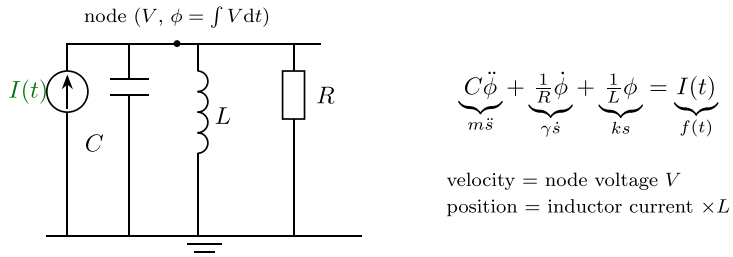


Figure 8. One damped oscillator = one parallel RLC node. A capacitor (mass), inductor (spring), and resistor (damper) from a common node to ground, driven by a current source (force). Kirchhoff's current law at the node reproduces $m\ddot{s} + \gamma\dot{s} + ks = f(t)$ exactly.

stated where needed. Both operations reduce to one synchronized switching event in this circuit model. Active negative-resistance elements and matched gain–loss RLC resonators have experimental precedent [SLZ+11, CWC+22], whereas the phase-switching arrangement developed here is a proposed implementation of the dissipative-LEP protocol.

N.1. A single oscillator is a parallel RLC node

The system to implement is the damped harmonic oscillator

$$m\ddot{s} + \gamma\dot{s} + ks = f(t)\,, \tag{49}$$

a mass m on a spring k with friction γ, driven by a force $f(t)$. The electrical version is a single circuit *node* tied to ground by three elements in parallel—a capacitor C, an inductor L, and a resistor R—and driven by a current source $I(t)$ (figure 8). Only two circuit facts are needed. First, each element relates its current to the voltage across the element, here the node voltage V since the other end is ground:

$$i_C = C\frac{\mathrm{d}V}{\mathrm{d}t} \quad \text{(capacitor)}\,, \qquad i_R = \frac{V}{R} \quad \text{(resistor)}\,, \qquad i_L = \frac{1}{L}\int V\mathrm{d}t \quad \text{(inductor)}\,,$$

where the inductor law is just $V = L\,\mathrm{d}i_L/\mathrm{d}t$ integrated once. Second, *Kirchhoff's current law* (KCL): the currents leaving the node through the three elements must sum to the current injected by the source. Writing $\phi \equiv \int V\mathrm{d}t$ for the running time-integral of the node voltage (the *node flux*; note $\dot{\phi} = V$), KCL reads

$$C\ddot{\phi} + \frac{1}{R}\dot{\phi} + \frac{1}{L}\phi = I(t)\,, \tag{50}$$

which is equation (49) term for term. The dictionary:

Mechanical	Electrical	Comment
position s	node flux $\phi = \int V\mathrm{d}t$	the integrated voltage
velocity $\dot{s}$	node voltage V	directly probed at the node
mass m	capacitance C	inertia
spring k	$1/L$ (inductor to ground)	restoring force
damping γ	$1/R$ (resistor to ground)	friction; $\zeta = \gamma/m = 1/(RC)$
force $f(t)$	injected current $I(t)$	the input
momentum $p = m\dot{s}$	capacitor charge $Q = CV$	

Two consequences of this dictionary matter for everything below. *(i)* The **velocity** of the oscillator is simply the node voltage—the easiest quantity to probe in any circuit—and the **position** is stored in the inductor current ($i_L = \phi/L \propto s$), also directly measurable. *(ii)* The state of the system lives in two continuous physical quantities that cannot jump under bounded drive: capacitor voltages (velocities) and inductor currents (positions). Every operation discussed below is an operation on these two families.

N.2. A network: coupling springs become coupling inductors

The coupled system of section 6.3 is built by replicating the node cell d times and adding one element per coupling: every coupling spring k_{ij} becomes a *coupling inductor* $L_{ij} = 1/k_{ij}$ wired directly **between nodes *i* and *j*** (figure 7(b)). The current through that inductor is $(\phi_i - \phi_j)/L_{ij}$—the electrical twin of the spring force $k_{ij}(s_i - s_j)$, which depends only on the *relative* displacement. KCL at node i then gives

$$C_i \ddot{\phi}_i + \frac{1}{R_i}\dot{\phi}_i + \frac{\phi_i}{L_i} + \sum_{j\neq i} \frac{\phi_i - \phi_j}{L_{ij}} = \delta_{i1}\, I(t)\,, \tag{51}$$

which is the coupled dynamics with

$$m_i = C_i, \qquad \gamma_i = \frac{1}{R_i}, \qquad K_{ii} = \frac{1}{L_i} + \sum_{j\neq i}\frac{1}{L_{ij}}, \qquad K_{ij} = -\frac{1}{L_{ij}}, \qquad \boldsymbol{f}(t) = I(t)\,\boldsymbol{e}_1. \tag{52}$$

Three practical remarks:

- **Symmetry for free.** A single physical inductor mediates the coupling in both directions, so $\boldsymbol{K}$ is automatically symmetric—the same reciprocity that makes a spring network symmetric.
- **Connectivity and sign class.** Any graph density, including all-to-all connectivity, can be realized within the passive spring-network sign class by adding one inductor per coupling (an $O(d^2)$ component count for a dense network). Passive inductors give attractive couplings, $k_{ij} = 1/L_{ij} > 0$ (equivalently $K_{ij} < 0$ off-diagonal), with $K_{ii} = 1/L_i + \sum_{j\neq i} 1/L_{ij}$; thus this construction does not realize an arbitrary symmetric stiffness matrix. A coupling of the opposite sign additionally requires a polarity-inverting (1:−1) transformer or an active element on that link.
- **Input and output.** The input force on oscillator 1 is a current injected at node 1; the output $s_{\text{out}} = s_3$ is read non-invasively as the current through the grounding inductor L_3 (proportional to ϕ_3), or as the integrated node-3 voltage.

N.3. Making a node dissipative—and flipping the sign of dissipation

Dissipation is one resistor. A node with only C and L is a lossless oscillator: energy sloshes forever between $\frac{1}{2}CV^2$ and $\frac{1}{2}Li_L^2$. This is the *conservative* case (standard LEP/RHEL). Adding the resistor makes it dissipative: the resistor draws a current V/R proportional to velocity—an electrical friction force—and burns energy at rate $P = V^2/R \geqslant 0$, giving damping rate $\zeta = 1/(RC)$. Each node chooses independently: omit the resistor (conservative node) or include it (dissipative node, per-node rate set by R_iC_i). One qualification: the scalar-ζ theory of section 6 requires all dissipative nodes to share a common rate, i.e. the products R_iC_i must be trimmed equal—and re-trimmed as the masses C_i train; genuinely independent per-node rates are covered by the anisotropic extension of appendix R.

Why the echo phase needs an *active* element. The echo phase of dissipative LEP requires flipping $+\boldsymbol{\gamma} \to -\boldsymbol{\gamma}$: a force that *aids* the velocity instead of opposing it, returning the energy dissipated during the free phase (proposition 6). Within this time-invariant RLC realization, a passive resistor cannot supply the required negative damping because its dissipated power V^2/R is non-negative. We therefore use an active negative-resistance element powered by an op-amp supply.

Negative resistance from one op-amp. The canonical circuit is the *negative impedance converter* (NIC), figure 9(a): an op-amp and three resistors [HH15, section 4.107, figure 4.104]. The same topology is shown by Schmidt *et al* [SBC19, section 2.2, figure 3]. In the ideal-op-amp approximation, the closed-loop relations $v_- = v_+$ and $i_- = i_+ = 0$ give:

$$\begin{aligned}
&\text{(i)} && v_B = v_A && \text{(equal input voltages)}\\
&\text{(ii)} && i_3 = v_B/R_3 = v_A/R_3 && \text{(Ohm's law through } R_3\text{)}\\
&\text{(iii)} && v_{\text{out}} = v_B + R_2 i_3 = v_A\left(1 + \tfrac{R_2}{R_3}\right) && \text{(same current through } R_2\text{)}\\
&\text{(iv)} && i_A = \frac{v_A - v_{\text{out}}}{R_1} = -\frac{R_2}{R_1 R_3}\, v_A && \text{(Ohm's law through } R_1\text{)}
\end{aligned}$$

Thus, in the ideal-op-amp approximation, the port presents the input resistance

$$R_{\text{in}} = \frac{v_A}{i_A} = -\frac{R_1 R_3}{R_2}.$$

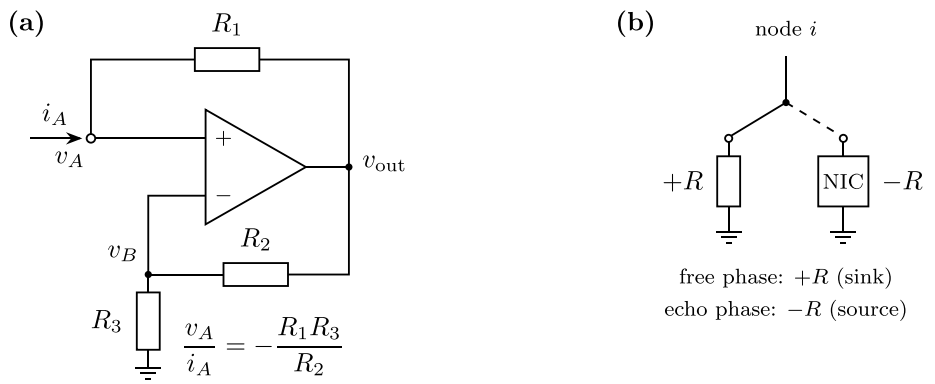


Figure 9. Sign-switchable dissipation. (a) In the ideal-op-amp approximation, the negative impedance converter (NIC) presents the input resistance $-R_1R_3/R_2$ at the port; the energy it injects during the echo phase comes from the op-amp's supply rails. (b) Each node's dissipative element is a selector between a passive resistor $+R$ (solid: free phase) and an NIC trimmed to $-R$ (dashed: echo phase). Flipping the sign of the dissipation is one control bit.

With $R_1 = R_2$, it is a resistor of value $\boldsymbol{-R_3}$. Current flows *out* of the terminal at higher voltage; the injected power is drawn from the op-amp's supply rails. In the mechanical picture, this is precisely the anti-friction force $+\gamma\dot{s}$.

The sign switch is one selector. Replace each node's resistor by a two-position switch selecting between a passive $+R$ (free phase: energy sink) and an NIC trimmed to $-R$ (echo phase: energy source), figure 9(b). Flipping the sign of the dissipation is then *one control bit*, toggled once at $t = T$. This selector is also less delicate than the momentum flip of section N.4, since the node's capacitor stays connected throughout the switch.

Practical caveat. For an exact reversal, the *net* damping during the echo phase must be the negative of the net free-phase damping. In the ideal circuit model this reduces to $R_{\text{NIC}} = -R_{\text{passive}}$. In physical hardware, the NIC must instead be trimmed to compensate the total free-phase loss, including intrinsic and parasitic losses.

N.4. The momentum flip: an H-bridge on each mass capacitor

The dissipation flip of the previous subsection is the operation specific to *dissipative* LEP; the echo phase also requires the velocity reversal already present in the conservative PFVP/RHEL protocol. The bouncing-backward kick (proposition 2) restarts the dynamics at $t = T$ from $(\boldsymbol{s}_T, -\dot{\boldsymbol{s}}_T)$: *keep every position, reverse every velocity*. In the circuit this is concrete:

- **Positions** live in the inductor currents (grounding and coupling): *touch nothing*. An undisturbed inductor keeps its current.
- **Velocities** live in the capacitor voltages: *invert each one*, $V_i \to -V_i$.

The polarity switch. The idealized circuit in figure 10 uses a four-switch polarity-reversing bridge—an *H-bridge*, a standard switch network in power electronics, routinely used to reverse the voltage applied to a load [EM20]—on each mass capacitor: at $t = T$ the bridge exchanges which capacitor terminal faces the node and which faces ground, changing V_i to $-V_i$ while leaving the stored energy $\frac{1}{2}C_i V_i^2$ unchanged in the ideal switching model. All bridges can share one control edge, so the whole velocity reversal is a single synchronized switching event. Physical switches introduce a brief transient and a small loss at the switching instant; sizing these imperfections is a hardware-design task beyond our scope, the circuit being an example realization of the protocol rather than an engineered design.

N.5. The learning rule: local and measurable, as in analog EqProp

The closest prior hardware proposal is the analog Equilibrium Propagation network of Kendall *et al* [KPM+20]: a *static* nonlinear resistive circuit whose central hardware result is that each synaptic conductance updates from a *local* signal—the squared voltage across that same resistor, contrasted between the free and nudged phases. Our network obeys the exact dual of this rule. Each coupling weight is an inductance L_{ij} with energy $\frac{1}{2}L_{ij}\,i_{ij}^2$, so the derivative of the Lagrangian with respect to L_{ij} is the squared current $\frac{1}{2}\,i_{ij}^2$—read off a small fixed series resistor, with no knowledge of L_{ij} itself required. Once the

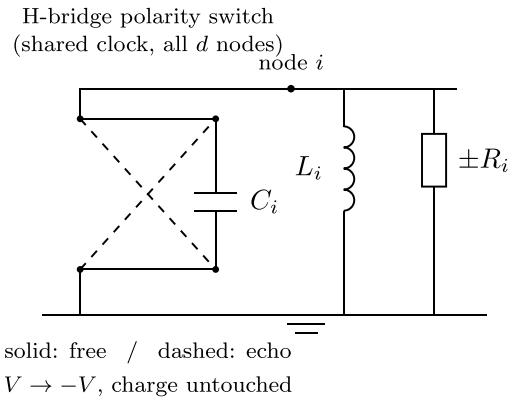


Figure 10. The momentum flip. Each mass capacitor sits behind an idealized four-switch polarity-reversing bridge (solid wiring: free phase; dashed: echo phase). At $t = T$, one shared control edge commands all d bridges: every capacitor voltage flips ($V \to -V$, velocities reversed), leaving the ideal capacitor energy $\frac{1}{2}CV^2$ unchanged, while every inductor current (every position) is untouched.

boundary terms cancel (as they do in the PFVP setting of theorem 3), the gradient-descent update with learning rate η contrasts this signal accumulated over the two physically-run phases,

$$\Delta L_{ij} = \frac{\eta}{2\beta}\left[\int_0^T i_{ij}^0(t)^2\,\mathrm{d}t - \int_0^T i_{ij}^\beta(T-t')^2\,\mathrm{d}t'\right], \tag{53}$$

where the first integral is accumulated during the free (inference) phase and the second during the echo phase, whose lab time t' runs from 0 to T while the network retraces the nudged trajectory backwards (i_{ij} is position-like, hence unchanged by the time reversal). The same recipe covers every other network parameter: the grounding inductor L_i (stiffness k_i) reads its own squared current, and the node capacitor C_i (mass m_i) its own squared node voltage. In the dissipative case (theorem 5), the same contrasts are accumulated with the exponential weight $e^{\zeta t}$—in the lab time of the echo, the decaying ramp $e^{\zeta(T-t')}$. This circuit is a feasibility construction for the coupled-oscillator substrate, not a substrate-independent implementation of dissipative LEP; for any other target substrate, realizability of the required operations must be assessed case by case.

Appendix O. Computational Complexity Analysis of LEP Instantiations

O.1. Motivation

Although the ultimate goal of LEP is to enable learning in continuous-time physical systems, understanding the computational complexity requires analyzing discrete-time implementations. This analysis serves two purposes. First, it provides concrete complexity characterizations for numerical simulations, which remain the primary means of validating these algorithms. Second, it reveals the fundamental scaling properties that carry over to continuous-time implementations, where the number of time steps N corresponds to the temporal resolution or duration of the physical process.

Throughout this appendix, we discretize the continuous-time dynamics using the simplest Euler integration scheme. While higher-order integrators may be preferred in practice for numerical stability, they do not change the asymptotic complexity with respect to the key parameters: sequence length N, state dimension d_s, and parameter count d_θ. The choice of Euler integration thus provides a lower bound on computational cost while maintaining clarity of exposition.

O.2. Setup and notation

We analyze the computational complexity of three instantiations of LEP: CIVP, CBPVP, and PFVP/RHEL. For concreteness, we consider the Hopfield Lagrangian from table 1:

$$L_0(\boldsymbol{s},\dot{\boldsymbol{s}},\boldsymbol{\theta},\boldsymbol{u}) = \frac{1}{2}\dot{\boldsymbol{s}}^\top \mathrm{diag}(\tau)\,\dot{\boldsymbol{s}} - \frac{\alpha}{2}\|\boldsymbol{s}\|^2 - b^\top\rho(\boldsymbol{s}) - \frac{1}{2}\rho(\boldsymbol{s})^\top W\rho(\boldsymbol{s}) - \rho(\boldsymbol{s})^\top B\rho(\boldsymbol{u}),$$

which yields the second-order dynamics:

$$\mathrm{diag}(\tau)\,\ddot{\boldsymbol{s}} = -\rho'(\boldsymbol{s}) \odot (\alpha\boldsymbol{s} + W\rho(\boldsymbol{s}) + b + B\rho(\boldsymbol{u})),$$

where $\tau \in \mathbb{R}^{d_s}_{>0}$ is a vector of learnable time constants, ρ denotes a pointwise nonlinearity (e.g. tanh), $W \in \mathbb{R}^{d_s \times d_s}$ is a symmetric weight matrix, $B \in \mathbb{R}^{d_s \times d_u}$ is the input coupling matrix, and $\odot$ denotes elementwise multiplication.

We adopt the following notation throughout. Let N denote the number of discrete time steps, corresponding to the sequence length. If the continuous-time dynamics span duration T and the integration step size is Δt, then $N = T/\Delta t$. Let d_s denote the state dimension, where both position $\boldsymbol{s}$ and velocity $\dot{\boldsymbol{s}}$ have dimension d_s. Let d_θ denote the number of learnable parameters; for the Hopfield model, $d_\theta = \mathcal{O}(d_s^2)$ due to the dense matrices W and B. For CBPVP, we additionally define K as the number of iterations required for the BVP solver to converge. If T_{gd} denotes the total optimization time needed for convergence and $\Delta\tau$ is the step size in the artificial relaxation time τ, then $K = T_{\text{gd}}/\Delta\tau$. Empirically, for systems related to Equilibrium Propagation, K typically scales with the number of neurons and the number of layers in hierarchical architectures [SB17]. In CBPVP, time is spatialized (section 3.3.1), so each discrete time step can be understood as a single layer. Under this analogy, d_s controls within-layer relaxation while N controls between-layer signal propagation, suggesting that K will generally grow with both N and d_s.

We denote by C_f the cost of one dynamics evaluation. For the Hopfield Lagrangian, each evaluation of the right-hand side $f(\boldsymbol{s}, \dot{\boldsymbol{s}}, \boldsymbol{\theta}, \boldsymbol{u}) = -\text{diag}(\tau)^{-1}\rho'(\boldsymbol{s}) \odot (\alpha \boldsymbol{s} + W\rho(\boldsymbol{s}) + b + B\rho(\boldsymbol{u}))$ requires computing the pointwise nonlinearity $\rho(\boldsymbol{s})$ in $\mathcal{O}(d_s)$ operations, the dense matrix-vector product $W\rho(\boldsymbol{s})$ in $\mathcal{O}(d_s^2)$ operations, and the input coupling $B\rho(\boldsymbol{u})$ in $\mathcal{O}(d_s \cdot d_u)$ operations. The diagonal scaling by $\text{diag}(\tau)^{-1}$ adds only $\mathcal{O}(d_s)$ operations. The total cost is therefore $C_f = \mathcal{O}(d_s^2)$, dominated by the dense matrix-vector multiplication. For architectures with diagonal or sparse weight matrices, this reduces to $C_f = \mathcal{O}(d_s)$.

O.3. CIVP (constant initial value problem)

The CIVP formulation is defined in section 3.3.2. In CIVP, all trajectories share fixed initial conditions $(\boldsymbol{s}_0, \dot{\boldsymbol{s}}_0) = (\boldsymbol{\alpha}, \boldsymbol{\gamma})$ that are independent of both the parameters $\boldsymbol{\theta}$ and the nudging strength β. The free and nudged trajectories are computed by forward integration from this common initial state.

Dynamics computation. Both the free phase ($\beta = 0$) and nudged phase ($\beta > 0$) constitute initial value problems that can be solved by forward integration. Using Euler discretization, the update rule takes the form:

$$\boldsymbol{s}_{t+1} = 2\boldsymbol{s}_t - \boldsymbol{s}_{t-1} + \Delta t^2 \cdot f(\boldsymbol{s}_t, \dot{\boldsymbol{s}}_t, \boldsymbol{\theta}, \boldsymbol{x}_t)\,.$$

Each time step requires one evaluation of the dynamics at cost $C_f = \mathcal{O}(d_s^2)$. With N time steps and two phases (free and nudged), the total dynamics computation requires $\mathcal{O}(N \cdot d_s^2)$ operations.

Regarding memory, the Euler integrator only requires access to the current and previous states to compute the next state. The dynamics computation therefore requires only $\mathcal{O}(d_s)$ memory.

Gradient computation. The CIVP gradient estimator, given by corollary 2, takes the form:

$$\Delta^{\text{CIVP}}(\beta) = \frac{1}{\beta}\left[\int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta - \partial_{\boldsymbol{\theta}} L_0\right] \mathrm{d}t + \left(\partial_{\dot{\boldsymbol{s}}} L_\beta - \partial_{\dot{\boldsymbol{s}}} L_0\right)^\top \partial_{\boldsymbol{\theta}} \boldsymbol{s}_T^0 - \left(\mathrm{d}_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0\right)^\top \left(\boldsymbol{s}_T^\beta - \boldsymbol{s}_T^0\right)\right]. \tag{54}$$

The problematic term is $\partial_{\boldsymbol{\theta}} \boldsymbol{s}_T^0 \in \mathbb{R}^{d_s \times d_\theta}$, which represents the sensitivity of the final state with respect to all parameters. This full Jacobian can be computed via backpropagation through time (BPTT), but since BPTT computes the gradient of a scalar output, one must run d_s separate backward passes—one for each component of $\boldsymbol{s}_T^0$—to obtain the complete matrix.

BPTT proceeds by first storing the entire forward trajectory $\{\boldsymbol{s}_t : t = 0, \ldots, N\}$, then executing backward passes to accumulate gradients. Each backward pass has the same computational structure as the forward pass, requiring $\mathcal{O}(N \cdot d_s^2)$ operations, so computing the full Jacobian costs $\mathcal{O}(d_s \cdot N \cdot d_s^2) = \mathcal{O}(N \cdot d_s^3)$ operations. Moreover, BPTT necessitates storing all intermediate states to enable the backward passes, resulting in $\mathcal{O}(N \cdot d_s)$ memory consumption.

The remaining terms in equation (54) are as follows. The integral term $\int_0^T [\partial_{\boldsymbol{\theta}} L_\beta - \partial_{\boldsymbol{\theta}} L_0]\,\mathrm{d}t$ requires $\mathcal{O}(N \cdot d_\theta)$ operations and can be accumulated during the two forward passes by maintaining two running sums:

$$\text{acc}_{\text{free}} \leftarrow \text{acc}_{\text{free}} + \partial_{\boldsymbol{\theta}} L_0\left(\boldsymbol{s}_t^0, \dot{\boldsymbol{s}}_t^0, \boldsymbol{\theta}\right) \cdot \Delta t$$

$$\text{acc}_{\text{nudged}} \leftarrow \text{acc}_{\text{nudged}} + \partial_{\boldsymbol{\theta}} L_\beta\left(\boldsymbol{s}_t^\beta, \dot{\boldsymbol{s}}_t^\beta, \boldsymbol{\theta}\right) \cdot \Delta t.$$

Each $\partial_{\boldsymbol{\theta}} L$ evaluation is performed once and immediately accumulated, requiring no trajectory storage for this term. The difference $(\partial_{\dot{\boldsymbol{s}}} L_\beta - \partial_{\dot{\boldsymbol{s}}} L_0)$ and the state difference $(\boldsymbol{s}_T^\beta - \boldsymbol{s}_T^0)$ are both $\mathcal{O}(d_s)$ to compute. However, the term $d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0$ is equally problematic: by the chain rule, $d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0 = \partial^2_{\dot{\boldsymbol{s}},\dot{\boldsymbol{s}}} L_0 \cdot d_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_T^0 + \partial^2_{\boldsymbol{\theta},\dot{\boldsymbol{s}}} L_0$, which involves the Jacobian $d_{\boldsymbol{\theta}} \dot{\boldsymbol{s}}_T^0 \in \mathbb{R}^{d_s \times d_\theta}$—the sensitivity of the final velocity to all parameters. Computing this Jacobian incurs the same $\mathcal{O}(N \cdot d_s^3)$ cost as $\partial_{\boldsymbol{\theta}} \boldsymbol{s}_T^0$.

This memory cost, which scales linearly with the sequence length N, constitutes the fundamental limitation of CIVP. It negates the primary advantage of Equilibrium Propagation, which aims to avoid storing trajectories for gradient computation.

Forward-only property. CIVP is not forward-only. The gradient computation requires an explicit backward pass through the stored computational graph. The system cannot compute gradients by running forward dynamics alone; it must differentiate through the ODE solver, necessitating either trajectory storage with backpropagation or forward propagation of a $d_s \times d_\theta$ Jacobian at each step (the RTRL algorithm, which incurs even greater time complexity).

O.4. CBPVP (constant boundary position value problem)

The CBPVP formulation is defined in section 3.3.1. In CBPVP, all trajectories satisfy fixed position boundary conditions at both temporal endpoints: $\boldsymbol{s}_0 = \boldsymbol{\alpha}$ and $\boldsymbol{s}_T = \boldsymbol{\gamma}$, independent of $\boldsymbol{\theta}$ and β. The velocities $\dot{\boldsymbol{s}}_0$ and $\dot{\boldsymbol{s}}_T$ remain free to vary.

Dynamics computation. Unlike CIVP, the CBPVP formulation defines a two-point BVP that cannot be solved by simple forward integration. Instead, one solves it via gradient descent on the action functional, as described in equation 26:

$$\partial_\tau \boldsymbol{s}_t = -\delta_{\boldsymbol{s}} \mathcal{A}_\beta = -\mathrm{EL}\left(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t, \boldsymbol{s}_{t+1}, \boldsymbol{\theta}, \beta\right), \quad t = 1, \ldots, N-1,$$

with fixed boundaries $\boldsymbol{s}_0 = \boldsymbol{\alpha}$ and $\boldsymbol{s}_T = \boldsymbol{s}_N = \boldsymbol{\gamma}$. Here τ represents an artificial relaxation time, while the physical time t becomes a spatial index. The procedure initializes a trajectory guess satisfying the boundary conditions, then iteratively updates the interior points according to the Euler–Lagrange residual until convergence.

Each relaxation iteration requires evaluating the Euler–Lagrange expression at all N time points, with each evaluation costing $\mathcal{O}(C_f) = \mathcal{O}(d_s^2)$. A single iteration therefore costs $\mathcal{O}(N \cdot d_s^2)$. Convergence typically requires K iterations, where K depends on the problem conditioning and initialization quality. The total dynamics computation thus requires $\mathcal{O}(K \cdot N \cdot d_s^2)$ operations.

The iterative nature of the BVP solver requires storing the entire trajectory $\{\boldsymbol{s}_t : t = 0, \ldots, N\}$ simultaneously, as all points are updated together in each iteration. The dynamics memory is therefore $\mathcal{O}(N \cdot d_s)$.

Gradient computation. The CBPVP gradient estimator, given by Corollary 2 (equation 24), simplifies considerably:

$$\Delta^{\mathrm{CBPVP}}(\beta) = \frac{1}{\beta} \int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta - \partial_{\boldsymbol{\theta}} L_0\right] \mathrm{d}t.$$

The boundary residuals vanish entirely because both endpoint positions are fixed. This is the principal advantage of the CBPVP formulation.

Computing this estimator requires evaluating the Lagrangian parameter derivatives $\partial_{\boldsymbol{\theta}} L$ at each of the N time points along both converged trajectories, after the K relaxation iterations have completed. For the Hopfield model, the dominant cost arises from $\partial_W L$, which involves outer products of dimension $d_s \times d_s$. Since $d_\theta = \mathcal{O}(d_s^2)$, the gradient computation requires $\mathcal{O}(N \cdot d_\theta)$ operations.

The gradient estimator only requires accumulating a running sum of dimension d_θ, resulting in $\mathcal{O}(d_\theta)$ memory for the gradient computation.

Forward-only property. CBPVP is forward-only in the sense that no backward pass through a computational graph is required. The gradient estimator does not require computing complex boundary residuals and the iterative solver only requires forward dynamics evaluations. However the iterative solver is much more expensive than the forward dynamics, requiring K iterations and $\mathcal{O}(N \cdot d_s)$ memory to store all time points simultaneously. These constraints preclude online or streaming processing of temporal sequences.

O.5. PFVP/RHEL (parametric final value problem)

The PFVP formulation is introduced in section 5.1.1 and its equivalence to RHEL (section 4) is established in section 5. In PFVP, the nudged trajectory shares its final conditions with the free trajectory's final state, but with reversed velocity: $(\boldsymbol{s}^{\beta}_{\leftarrow,T}, \dot{\boldsymbol{s}}^{\beta}_{\leftarrow,T}) = (\boldsymbol{s}^0_T, -\dot{\boldsymbol{s}}^0_T)$. These boundary conditions depend on $\boldsymbol{\theta}$ through the free trajectory, which distinguishes PFVP from the constant boundary conditions of CIVP and CBPVP.

Key insight: exploiting reversibility. For time-reversible Lagrangians satisfying $L(\boldsymbol{s}, \dot{\boldsymbol{s}}) = L(\boldsymbol{s}, -\dot{\boldsymbol{s}})$, Proposition 2 establishes that the final value problem can be converted to an initial value problem:

$$\boldsymbol{s}^{\beta}_{\leftarrow,t}\left(\boldsymbol{\theta}, \left(\boldsymbol{s}^0_T, \dot{\boldsymbol{s}}^0_T\right)\right) = \boldsymbol{s}^{\beta}_{T-t}\left(\boldsymbol{\theta}, \left(\boldsymbol{s}^0_T, -\dot{\boldsymbol{s}}^0_T\right)\right).$$

Rather than solving a difficult final value problem, one simply performs forward integration from the velocity-reversed final state while playing the input sequence backward. This transformation is the key enabler of PFVP's computational efficiency.

Dynamics computation. The free phase proceeds by standard forward integration from initial conditions $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ over the interval $t \in [0, T]$, storing only the final state $(\boldsymbol{s}^0_T, \dot{\boldsymbol{s}}^0_T)$ upon completion. This requires $\mathcal{O}(N \cdot d_s^2)$ operations.

The echo phase initializes from $(\boldsymbol{s}^0_T, -\dot{\boldsymbol{s}}^0_T)$ and integrates forward over $t \in [0, T]$, using the time-reversed input sequence $\boldsymbol{x}_{T-t}$ and targets $\boldsymbol{y}_{T-t}$. This also requires $\mathcal{O}(N \cdot d_s^2)$ operations.

The total dynamics computation is therefore $\mathcal{O}(N \cdot d_s^2)$, identical to CIVP. Note, however, that the two phases must be executed sequentially: the echo phase requires the final state $(\boldsymbol{s}^0_T, \dot{\boldsymbol{s}}^0_T)$ from the free phase to initialize its dynamics. This contrasts with CIVP, where the free and nudged trajectories are independent initial value problems that can be computed in parallel.

Regarding memory, each phase requires only the current state for the Euler integrator, consuming $\mathcal{O}(d_s)$ memory. The only additional storage is the final state of the free phase, which is needed to initialize the echo phase, also $\mathcal{O}(d_s)$. The dynamics memory is therefore $\mathcal{O}(d_s)$, independent of the sequence length N.

Gradient computation. The PFVP gradient estimator, given by Theorem 3 (equation (43)), takes the form:

$$\Delta^{\mathrm{PFVP}}(\beta) = \frac{1}{\beta}\left[\int_0^T \left[\partial_{\boldsymbol{\theta}} L_\beta - \partial_{\boldsymbol{\theta}} L_0\right] \mathrm{d}t + \left(\mathrm{d}_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0\right)^\top \left(\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha}\right) - \left(\partial_{\boldsymbol{\theta}} \boldsymbol{\alpha}\right)^\top \left(\partial_{\dot{\boldsymbol{s}}} L_\beta - \partial_{\dot{\boldsymbol{s}}} L_0\right)\right].$$

Unlike CIVP, no trajectory sensitivities such as $\partial_{\boldsymbol{\theta}} \boldsymbol{s}^0_T$ appear in this estimator, which eliminates the need for backpropagation.

As in CIVP, the integral term can be computed by maintaining two accumulators that are updated at each time step during the forward integration, requiring $\mathcal{O}(N \cdot d_\theta)$ operations. The boundary terms are computed as follows: the state difference $(\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha})$ and momentum difference $(\partial_{\dot{\boldsymbol{s}}} L_\beta - \partial_{\dot{\boldsymbol{s}}} L_0)$ are both $\mathcal{O}(d_s)$ to compute. When $\partial_{\boldsymbol{\theta}} \boldsymbol{\alpha} = 0$, the second boundary term vanishes. The first boundary term $(d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0)^\top (\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha})$ involves $d_{\boldsymbol{\theta}} \partial_{\dot{\boldsymbol{s}}} L_0 = \partial^2_{\boldsymbol{\theta},\dot{\boldsymbol{s}}} L_0$, which for the Hopfield model is $\mathcal{O}(d_s \times d_\theta)$; however, the contraction with the $\mathcal{O}(d_s)$ vector $(\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha})$ yields an $\mathcal{O}(d_\theta)$ result. For the Hopfield Lagrangian where $L = \frac{1}{2}\dot{\boldsymbol{s}}^\top \mathrm{diag}(\tau)\dot{\boldsymbol{s}} - V(\boldsymbol{s}, \boldsymbol{\theta})$, we have $\partial_{\dot{\boldsymbol{s}}} L = \mathrm{diag}(\tau)\dot{\boldsymbol{s}}$. The only $\boldsymbol{\theta}$-dependence is through τ, so $\partial^2_{\tau,\dot{\boldsymbol{s}}} L = \mathrm{diag}(\dot{\boldsymbol{s}})$, which is diagonal and $\mathcal{O}(d_s)$. The contraction $(\partial^2_{\tau,\dot{\boldsymbol{s}}} L)^\top (\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha}) = \dot{\boldsymbol{s}} \odot (\boldsymbol{s}^{\beta}_{\leftarrow,0} - \boldsymbol{\alpha})$ is therefore $\mathcal{O}(d_s)$ to compute.

The total gradient computation requires $\mathcal{O}(N \cdot d_\theta)$ operations, dominated by the integral term. The memory requirement is $\mathcal{O}(d_\theta)$ for the two accumulators plus $\mathcal{O}(d_s)$ for the boundary quantities.

Forward-only property. PFVP/RHEL satisfies the forward-only property in the strongest sense. Both the free and echo phases consist of pure forward integration. No iterative solving is required, in contrast to CBPVP. No backward pass through a computational graph is needed, in contrast to CIVP. The gradients are computed from Lagrangian derivatives accumulated along the trajectories during the forward passes.

It is important to note that the echo phase is not a backward pass in the algorithmic sense. It is a forward pass with reversed initial velocity and reversed input sequence. The physical system runs forward in time during both phases. This property makes PFVP/RHEL well suited to implementation in physical hardware, because both phases reuse the system's natural forward dynamics.

O.6. Summary

The analysis reveals a clear hierarchy among the three instantiations, as summarized in table 2 in the main text. CIVP achieves efficient dynamics computation but requires BPTT for gradient estimation, incurring $\mathcal{O}(N \cdot d_s)$ memory to store the trajectory and $\mathcal{O}(N \cdot d_s^3)$ in time complexity.

CBPVP eliminates the boundary residuals entirely, yielding a clean gradient estimator, and maintains the forward-only property. However, solving the BVP requires K iterations and $\mathcal{O}(N \cdot d_s)$ memory to store all time points simultaneously. These constraints preclude online or streaming processing and may result in slow convergence for challenging problems.

PFVP/RHEL achieves optimal scaling across all metrics. The dynamics computation matches CIVP's efficiency through pure forward integration. The gradient computation requires only local Lagrangian derivatives accumulated during the forward passes. Both time and memory complexities are independent of the sequence length N in terms of trajectory storage, with memory scaling only as $\mathcal{O}(d_s + d_\theta)$. Among the three LEP instantiations analyzed here, PFVP/RHEL is the only one that combines forward-only gradient computation with streaming trajectory processing.

Appendix P. Experimental Details

P.1. Hopfield-inspired system (figure 5)

This experiment trains a $d = 6$ Hopfield-inspired system over 100 epochs in a teacher-student setup. Both RHEL and LEP training runs use the Adam optimizer with learning rate $\eta = 0.005$, nudging strength $\beta = 0.01$, Euler integration with $\mathrm{dt} = 0.001$, total duration $T = 10$, and random seed 50. Gradients are saved every 10 epochs.

Weight initialization. The symmetric weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is initialized via QR decomposition with controlled eigenvalues. A random matrix is drawn from $\mathcal{N}(0,1)$ and its QR factorization yields an orthogonal matrix $\boldsymbol{U}$. A diagonal matrix $\boldsymbol{S} = \mathrm{diag}(\boldsymbol{\lambda})$ is formed with eigenvalues $\lambda_i \sim \mathrm{Uniform}(0.1, 1.0)$. The weight matrix is then constructed as $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^\top$, ensuring symmetry and bounded eigenvalues for stable dynamics.

Time constants. Each τ_i is sampled independently from $\mathrm{Uniform}(0.5, 1.0)$.

Initial conditions. The Hamiltonian initial conditions are fixed:

$$\boldsymbol{s}_0 = (0.10,\ 0.15,\ 0.08,\ 0.12,\ 0.10,\ 0.11)^\top,$$
$$\boldsymbol{p}_0 = (0.0,\ 0.4,\ -0.6,\ 0.45,\ 0.5,\ 0.0)^\top.$$

In the LEP training run, the Lagrangian initial velocity is $\dot{\boldsymbol{s}}_0 = \mathrm{diag}(\boldsymbol{\tau})^{-1}\boldsymbol{p}_0$, which changes across training epochs as $\boldsymbol{\tau}$ evolves.

Input signal. The input x_t is a superposition of $n_{\mathrm{waves}} = 10$ random sine waves injected into neuron 0 (with input scaling 1.0). The activation function is $\rho = \tanh$.

P.2. Dissipative harmonic oscillators (figure 6)

This experiment validates the dissipative LEP gradient estimator on a $d = 6$ system of coupled damped harmonic oscillators. No training is performed for this gradient check (Panels A–C): the gradient comparison is computed at a single epoch from the randomly initialized parameters, comparing the dissipative LEP gradient estimate against the autodiff/BPTT ground truth. Integration uses $\mathrm{dt} = 0.001$, total duration $T = 15$, nudging strength $\beta = 0.01$, and random seed 50.

Mass and stiffness initialization. Masses m_i are sampled from $\mathrm{Uniform}(0.8, 1.2)$. The stiffness matrix $\boldsymbol{K}$ is constructed to be symmetric positive semi-definite: diagonal self-coupling terms are scaled by 0.5, off-diagonal coupling terms by 1.0, and the matrix is symmetrized via $\boldsymbol{K} = \frac{1}{2}(\boldsymbol{K} + \boldsymbol{K}^\top)$, with diagonal entries adjusted to include the row sums of the coupling matrix.

Dissipation. The damping coefficient is $\zeta = 0.2$, giving per-dimension damping forces $\gamma_i = \zeta \cdot m_i$.

Initial conditions. All positions are initialized to $s_{i,0} = 1.0$ and all velocities to $\dot{s}_{i,0} = 0$, ensuring that boundary terms in the gradient estimator vanish (remark 3).

Input signal. The external drive is injected into oscillator 1 with input scaling 5.0. The output is measured from oscillator d (the last one). The cost function is $c(\boldsymbol{s}_t, y_t) = \frac{1}{2}(s_{d,t} - y_t)^2$.

Training experiment (Panel D). To demonstrate end-to-end learning rather than a single gradient check, we additionally *train* the same $d = 6$ damped-oscillator system in a teacher–student setup. Teacher and student are drawn independently from the initialization described above, so that they differ in all parameters: the masses differ by 15.5% and the stiffness matrix by 31.8% in relative Frobenius norm, while their damping coefficients differ by a factor of two (the teacher uses $\zeta = 0.2$ and the student is initialized at $\zeta_0 = 0.1$). During training all of the student's parameters $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{K}, \zeta\}$ are updated to reproduce the teacher's output trajectory $y_t = s_{d,t}$. Training uses the Adam optimizer with learning rate $\eta = 0.005$, nudging strength $\beta = 0.01$, Euler integration with dt $= 0.001$ and total duration $T = 15$ (matching Panels A–C), for 200 epochs. We compare three learning rules: (i) dissipative LEP (with the exponential weighting); (ii) BPTT/autodiff; and (iii) the classical LEP baseline (sign-flipped damping but no exponential weighting). Dissipative LEP follows the BPTT learning curve closely throughout training: the loss falls from approximately 1.39 to 3.11×10^{-3} for dissipative LEP and to 3.06×10^{-3} for BPTT—a final relative difference of 1.8%—and both recover the damping coefficient $\zeta \to 0.2$. Because the estimator is evaluated at the finite nudging strength $\beta = 0.01$, it retains the expected $O(\beta)$ error of the one-sided finite difference. Forward Euler discretization and the optimizer may introduce additional numerical differences, but their contributions have not been separated in this experiment. The classical baseline can still reduce the loss by updating $\boldsymbol{m}$ and $\boldsymbol{K}$. However, these potential-parameter gradients omit the exponential weighting required by dissipative LEP, and its ζ-gradient is identically zero because, without the factor $e^{\zeta t}$, the damping coefficient does not enter the classical estimator (see Panel C). It therefore cannot learn the damping coefficient and eventually plateaus near 0.238, approximately two orders of magnitude above dissipative LEP. The result demonstrates effective learning with dissipative LEP and a substantial improvement over the classical estimator, without implying exact agreement with BPTT at finite β and finite time step.

Appendix Q. Learning Performance on the Teacher–Student Task

This appendix complements the gradient-agreement analysis of section 5.4 with two additional training runs on the same teacher–student task and with the same hyperparameters (LEP and RHEL, 100 epochs, protocol of appendix P).

Learning curves and trajectory reconstruction. Figure 11(a) shows the training loss of both runs. The loss decreases from 0.38 to 5.6×10^{-4}—nearly three orders of magnitude—and the LEP and RHEL learning curves are indistinguishable throughout training, as predicted by the gradient equivalence of theorem 4. Figure 11(b) overlays the student's output trajectory on the teacher's target: before training, the output clearly deviates from the target, whereas after 100 epochs it reproduces it, the output mean squared error having decreased from 3.8×10^{-2} to 5.6×10^{-5}.

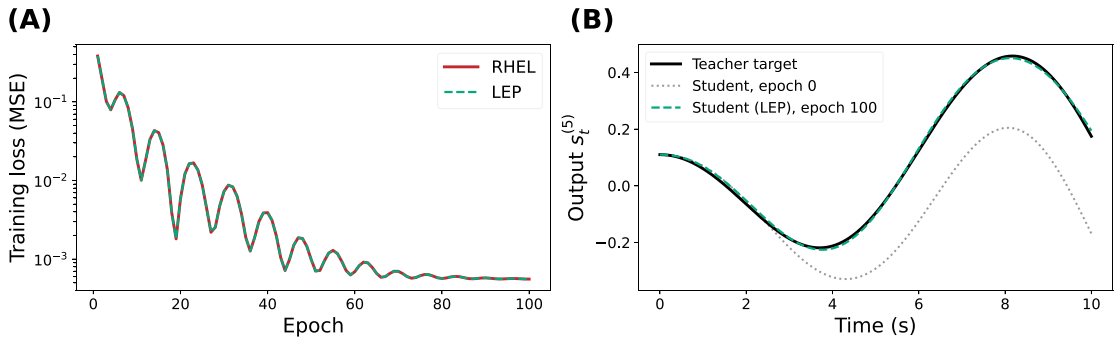


Figure 11. Learning performance on the teacher–student task of figure 5. (a) Training loss (MSE, log scale) of the two independent training runs of section 5.4: RHEL (red, solid) and LEP (green, dashed). The two learning curves coincide throughout training and decrease by nearly three orders of magnitude. (b) Output trajectory $s_t^{(5)}$ of the LEP-trained student before (grey, dotted) and after (green, dashed) training, overlaid on the teacher target (black). After training, the student reproduces the target trajectory.

These results should be read within the scope of our claims, delimited in the Discussion paragraph *Relation to BPTT*: the theory specifies how the gradient is computed, while the learning process built on top of it (optimizer, learning rate, schedule) is orthogonal and carries over unchanged from standard deep learning. The learning curves above are therefore an end-to-end consistency check of the estimator inside a complete optimization loop, not a claim of a novel learning behavior; empirical evidence at larger scale exists for RHEL [PE25] and transfers to LEP through theorem 4, as discussed in the same paragraph.

Appendix R. Generalization to Anisotropic Damping

In section 6, we introduced the dissipative Lagrangian $L_\beta^{\text{diss}} = L_0 \cdot \exp(\zeta t)$ with a *scalar* damping coefficient $\zeta > 0$, which produces uniform proportional damping $\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \zeta\, \boldsymbol{m} \odot \dot{\boldsymbol{s}}_t + \boldsymbol{K}\boldsymbol{s}_t = \boldsymbol{f}(t)$ where all oscillators share the same damping ratio. Here we present a generalization that allows *anisotropic* (per-dimension) damping rates while maintaining a variational structure.

R.1. Anisotropic exponential integrating-factor Lagrangian

Let $\boldsymbol{s}(t) \in \mathbb{R}^d$, $\boldsymbol{m} = (m_1, \ldots, m_d)^\top \in \mathbb{R}^d$ be the mass vector, and define the per-dimension damping coefficients $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_d)^\top \in \mathbb{R}^d$ with $\gamma_i > 0$ (not necessarily equal). Define the elementwise exponential:

$$\boldsymbol{e}(t) := \exp(\boldsymbol{\gamma} \odot t) = (e^{\gamma_1 t}, \ldots, e^{\gamma_d t})^\top \in \mathbb{R}^d,$$

where $\odot$ denotes elementwise multiplication.

Pick *any symmetric matrix function* $B(t) = B(t)^\top \in \mathbb{R}^{d\times d}$. The time-dependent Lagrangian is:

$$L(t, \boldsymbol{s}, \dot{\boldsymbol{s}}) = \frac{1}{2}\sum_{i=1}^{d} e^{\gamma_i t} m_i \dot{s}_i^2 - \frac{1}{2}\boldsymbol{s}^\top B(t)\, \boldsymbol{s}.$$

Euler–Lagrange equations. For each dimension i, we have $\partial_{\dot{s}_i} L = e^{\gamma_i t} m_i \dot{s}_i$ and $\frac{\mathrm{d}}{\mathrm{d}t}(e^{\gamma_i t}) = \gamma_i e^{\gamma_i t}$. The Euler–Lagrange equation gives:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{\gamma_i t} m_i \dot{s}_i\right) + [B(t)\, \boldsymbol{s}]_i = 0 \quad \Longleftrightarrow \quad e^{\gamma_i t} m_i \ddot{s}_i + \gamma_i e^{\gamma_i t} m_i \dot{s}_i + [B(t)\, \boldsymbol{s}]_i = 0.$$

Dividing by $e^{\gamma_i t}$ and writing in vector form:

$$\boldsymbol{m} \odot \ddot{\boldsymbol{s}}_t + \boldsymbol{\gamma} \odot \boldsymbol{m} \odot \dot{\boldsymbol{s}}_t + \boldsymbol{k}_{\text{eff}}(t) = \boldsymbol{0}, \qquad \boldsymbol{k}_{\text{eff}}(t) := \exp(-\boldsymbol{\gamma} \odot t) \odot (B(t)\, \boldsymbol{s}_t).$$

Thus, *anisotropic damping* with per-dimension damping rates γ_i is generated exactly. The price is an induced, generally *time-varying*, effective force $\boldsymbol{k}_{\text{eff}}(t)$ determined by the choice of $B(t)$.

R.2. Physical interpretation: time-varying coupling

The effective force $\boldsymbol{k}_{\text{eff}}(t) = \exp(-\boldsymbol{\gamma} \odot t) \odot (B(t)\boldsymbol{s}_t)$ has a natural interpretation: *the coupling between oscillators switches off with time.* Each oscillator i has its own exponential decay coefficient $e^{-\gamma_i t}$, so the coupling from oscillator i to the rest of the network decays according to *its own damping rate* γ_i. Different oscillators can 'disconnect' from the network at different rates, leading to time-dependent coupling encoded in $\boldsymbol{k}_{\text{eff}}(t)$.

If one wishes the physical coupling to remain *time-independent* in the sense that $\boldsymbol{k}_{\text{eff}}(t) = \boldsymbol{K}\boldsymbol{s}_t$ for some constant matrix $\boldsymbol{K}$, one must choose $B(t)$ such that $\exp(-\boldsymbol{\gamma} \odot t) \odot (B(t)\boldsymbol{s}_t) = \boldsymbol{K}\boldsymbol{s}_t$ for all $\boldsymbol{s}_t$. This requires $B(t)_{ij} = e^{(\gamma_i + \gamma_j)t/2} K_{ij}$ (assuming a symmetric construction). However, for $B(t)$ to be symmetric (as required for a proper mechanical potential), we need additional structure.

The simplest cases where time-independent coupling is achievable are:

- All γ_i equal (scalar damping)—this is the case in section 6;
- $\boldsymbol{K}$ is diagonal (uncoupled oscillators);
- Special damping structures where the per-dimension rates align with the coupling structure.

When these conditions fail (generic coupling with different γ_i), maintaining time-independent physical coupling within the variational framework is not possible—one must either restrict the damping structure or accept time-varying $\boldsymbol{k}_{\text{eff}}(t)$ in the learning dynamics.

R.3. Comparison with alternative approaches

One might consider using Rayleigh dissipation functions $\mathcal{R}(\dot{\boldsymbol{s}}) = \frac{1}{2}\sum_{ij} C_{ij}\dot{s}_i\dot{s}_j$ (separate from the Lagrangian L), which handle arbitrary damping matrices elegantly in classical mechanics via the modified Euler–Lagrange equation $d_t\partial_{\dot{\boldsymbol{s}}}L - \partial_{\boldsymbol{s}}L + \partial_{\dot{\boldsymbol{s}}}\mathcal{R} = 0$. However, this approach is *incompatible* with the variational gradient estimator framework presented in this work (theorem 5), which requires all system dynamics to be encoded within the Lagrangian L_β^{diss} itself. The gradient estimator depends on $\partial_{\boldsymbol{\theta}} L_0$, not on a separate dissipation function.

More broadly, one could perform gradient descent directly on the action functional. However, as discussed in the CBPVP instantiation (section 5), the *converging phase* of such optimization is dissipative (evolving in the artificial relaxation time τ), while the *fixed Hamiltonian system* implemented after convergence corresponds to a non-dissipative system on the physical time axis. The value of maintaining a *variational principle within the Lagrangian itself* is that it guides the construction of learning algorithms systematically, enabling principled extensions like the dissipative LEP framework, rather than relying on ad-hoc inspired guesses as was done in prior work (e.g. RHEL before its variational foundation was established in theorem 4).

Appendix S. Unconstrained action minimization

In the main text (section 3.3), we noted that if one is willing to accept iterative optimization rather than forward Euler–Lagrange integration, boundary conditions need not be imposed at all. We elaborate on this observation here.

Consider minimizing the action functional without any boundary constraints:

$$\boldsymbol{s}^\beta = \arg\min_{\boldsymbol{s}} A_\beta[\boldsymbol{s}]\,. \tag{55}$$

Since the initial and final values $\boldsymbol{s}_0^\beta$ and $\boldsymbol{s}_T^\beta$ are determined implicitly as part of the optimization, the variational principle is no longer partial: the boundary terms in the first variation of the action vanish by the natural boundary conditions (which require $\partial_{\dot{\boldsymbol{s}}}L_\beta = 0$ at both endpoints). Consequently, boundary residuals vanish entirely in theorem 1, and the gradient estimator reduces to the simple form of CBPVP (equation (8)).

However, this formulation inherits the same non-causal drawbacks as CBPVP—and is in fact more expensive. In CBPVP, the $2d_s$ boundary values $(\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_T)$ are fixed, so the optimization runs over the interior of the trajectory: a space of dimension $(N-2)\times d_s$. In the unconstrained formulation, the full trajectory including its endpoints becomes part of the optimization, increasing the search space to $N\times d_s$. The iterative solver cost remains $\mathcal{O}(KNd_s^2)$ with a potentially larger K due to the additional degrees of freedom.

In summary, unconstrained action minimization yields a 'perfect' variational formulation—analogous to standard EP—where the gradient estimator is free of boundary residuals. Yet this comes at the price of a non-causal, iterative computation that is at least as expensive as CBPVP, making it equally impractical for forward-only hardware implementations.

ORCID iDs

Guillaume Pourcel 0000-0002-7147-3652
Debabrota Basu 0000-0002-3204-2884
Aditya Gilra 0000-0002-8628-1864

References

[ABB+25] Aifer M *et al* 2025 Solving the compute crisis with physics-based asics (arXiv:2507.10463)
[AdHLG24] Aceituno P V, de Haan S, Loidl R and Grewe B F 2024 Target learning rather than backpropagation explains learning in the mammalian neocortex bioRxiv (https://doi.org/10.1101/2024.04.10.588837)
[Alm89] Almeida L B 1989 Backpropagation in perceptrons with feedback *Neural Computers* (Springer) pp 199–208
[AWH+19] Akrout M, Wilson C, Humphreys P, Lillicrap T and Tweed D B 2019 Deep learning without weight transport *Advances in Neural Information Processing Systems* (vol 32)
[BB08] Bell B M and Burke J V 2008 Algorithmic differentiation of implicit functions and optimal values *Advances in Automatic Differentiation* (Springer) pp 67–77
[BH25] Berneman M and Hexner D 2025 Equilibrium propagation for periodic dynamics (arXiv:2506.20402)

[BHM+16] Ba J, Hinton G E, Mnih V, Leibo J Z and Ionescu C 2016 Using fast weights to attend to the recent past *Advances in Neural Information Processing Systems* vol 29
[BKK19] Bai S, Kolter J Z and Koltun V 2019 Deep equilibrium models *Advances in Neural Information Processing Systems* vol 32
[BRR25] Boyer J, Rusch T K and Rus D 2025 Learning to dissipate energy in oscillatory state-space models (arXiv:2505.12171)
[CRBD18] Chen R T Q, Rubanova Y, Bettencourt J and Duvenaud D 2018 Neural ordinary differential equations *Advances in Neural Information Processing Systems* (vol 31)
[CWC+22] Cao W, Wang C, Chen W, Hu S, Wang H, Yang L and Zhang X 2022 Fully integrated parity-time-symmetric electronics *Nat. Nanotechnol.* **17** 262–8
[DBS+24] Dillavou S, Beyer B D, Stern M, Liu A J, Miskin M Z and Durian D J 2024 Machine learning without a processor: emergent learning in a nonlinear analog network *Proc. Natl Acad. Sci.* **121** e2319718121
[DFE+22] Dao T, Fu D, Ermon S, Rudra A and Ré C 2022 FlashAttention: fast and memory-efficient exact attention with IO-awareness *Advances in Neural Information Processing Systems* vol 35 pp 16344–59
[DP25] Dauphin A S and Pourcel G 2025 Recurrent Hamiltonian Echo Learning Enables Biologically Plausible Training of Recurrent Neural Networks *Women in Machine Learning Workshop @ NeurIPS 2025*
[EGQ+19] Ernoult M, Grollier J, Querlioz D, Bengio Y and Scellier B 2019 Updates of equilibrium prop match gradients of backprop through time in an RNN with static input *Advances in Neural Information Processing Systems* vol 32
[EM20] Erickson R W and Maksimović D 2020 *Fundamentals of Power Electronics* 3 edn (Springer)
[Ern20] Ernoult M 2020 Rethinking biologically inspired learning algorithms towards better credit assignment for on-chip learning *PhD Thesis* (Sorbonne Université)
[FFS07] Fiete I R, Fee M S and Seung H S 2007 Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances *J. Neurophysiol.* **98** 2038–57
[GG17] Gilra A and Gerstner W 2017 Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network *eLife* **6** e28295
[HH15] Horowitz P and Hill W 2015 *The Art of Electronics* 3 edn (Cambridge University Press)
[Hin22] Hinton G 2022 The forward-forward algorithm: Some preliminary investigations (arXiv:2212.13345)
[Hoo20] Hooker S 2020 The hardware lottery (arXiv:2009.06489)
[Hop82] Hopfield J J 1982 Neural networks and physical systems with emergent collective computational abilities *Proc. Natl Acad. Sci.* **79** 2554–8
[JGP+17] Jing L, Gulcehre C, Peurifoy J, Shen Y, Tegmark M, Soljačić M and Bengio Y 2017 Gated orthogonal recurrent units: on learning to forget (arXiv:1706.02761)
[JNv23] Jaeger H, Noheda B and van der Wiel W G 2023 Toward a formal theory for computing machines made out of whatever physics offers *Nat. Commun.* **14** 4911
[JYP+17] Jouppi N P *et al* 2017 In-datacenter performance analysis of a tensor processing unit *Proc. 44th Annual International Symposium on Computer Architecture* pp 1–12
[Ken21] Kendall J 2021 A gradient estimator for time-varying electrical networks with non-linear dissipation (arXiv:2103.05636)
[KPM+20] Kendall J, Pantone R, Manickavasagam K, Bengio Y and Scellier B 2020 Training end-to-end analog neural networks with equilibrium propagation (arXiv:2006.01981)
[LBH15] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44
[LCTA16] Lillicrap T P, Cownden D, Tweed D B and Akerman C J 2016 Random synaptic feedback weights support error backpropagation for deep learning *Nat. Commun.* **7** 13276
[LES+21] Laborieux A, Ernoult M, Scellier B, Bengio Y, Grollier J and Querlioz D 2021 Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias *Frontiers in Neuroscience* **15** 633674
[LPM23] López-Pastor V and Marquardt F 2023 Self-learning machines based on Hamiltonian echo backpropagation *Phys. Rev. X* **13** 031020
[LSM+20] Lillicrap T P, Santoro A, Marris L, Akerman C J and Hinton G 2020 Backpropagation and the brain *Nat. Rev. Neurosci.* **21** 335–46
[LWWM24] Laydevant J, Wright L G, Wang T and McMahon P L 2024 The hardware is the software *Neuron* **112** 180–3
[LZ22] Laborieux A and Zenke F 2022 Holomorphic equilibrium propagation computes exact gradients through finite size oscillations *Advances in Neural Information Processing Systems* vol 35 pp 12950–63
[LZC+22] Lin J, Zhu L, Chen W-M, Wang W-C, Gan C and Han S 2022 On-device training under 256kb memory *Advances in Neural Information Processing Systems* vol 35 pp 22941–54
[Mas25] Massar S 2025 Equilibrium propagation for learning in Lagrangian dynamical systems *Phys. Rev. E* **112** 035304
[MRS+24] Momeni A *et al* 2024 Training of physical neural networks (arXiv:2406.03372)
[MZK+22] Meulemans A, Zucchet N, Kobayashi S, von Oswald J and Sacramento J 2022 The least-control principle for local learning at equilibrium *Advances in Neural Information Processing Systems* vol 35 pp 33603–17
[NE24] Nest T and Ernoult M 2024 Towards training digitally-tied analog blocks via hybrid gradient computation *Advances in Neural Information Processing Systems* vol 37 pp 83877–914
[Olv22] Olver P J 2022 The Calculus of Variations
[OSG+23] Orvieto A, Smith S L, Gu A, Fernando A, Gulcehre C, Pascanu R and De S 2023 Resurrecting recurrent neural networks for long sequences (arXiv:2303.06349)
[PCG+23] Pogodin R, Cornford J, Ghosh A, Gidel G, Lajoie G and Richards B 2023 Synaptic weight distributions depend on the geometry of plasticity (arXiv:2305.19394)
[PE25] Pourcel G and Ernoult M 2025 Learning long range dependencies through time reversal symmetry breaking (arXiv:2506.05259)
[Pin89] Pineda F J 1989 Recurrent backpropagation and the dynamical approach to adaptive neural computation *Neural Comput.* **1** 161–72
[RHW86] Rumelhart D E, Hinton G E and Williams R J 1986 Learning representations by back-propagating errors *Nature* **323** 533–6
[Rie96] Riewe F 1996 Nonconservative Lagrangian and Hamiltonian mechanics *Phys. Rev. E* **53** 1890–9
[RKLH22] Ren M, Kornblith S, Liao R and Hinton G 2022 Scaling forward gradient with local losses (arXiv:2210.03310)

[RLB+19] Richards B A *et al* 2019 A deep learning framework for neuroscience *Nat. Neurosci.* **22** 1761–70
[RM21] Rusch T K and Mishra S 2021 UnICORNN: a recurrent model for learning very long time dependencies (arXiv:2103.05487)
[Ros60] Rosenblatt F 1960 Perceptual generalization over transformation groups *Self-Organizing Systems* ed M C Yovits and S Cameron (*International Tracts in Computer Science and Technology and Their Application* vol 2) (Pergamon) pp 63–96
[RR25] Rusch T K and Rus D 2025 Oscillatory State-Space Models (arXiv:2410.03943)
[S+86] Smolensky P *et al* Information processing in dynamical systems: Foundations of harmony theory 1986
[SB17] Scellier B and Bengio Y 2017 Equilibrium propagation: bridging the gap between energy-based models and backpropagation *Front. Comput. Neurosci.* **11** 24
[SB19] Scellier B and Bengio Y 2019 Equivalence of equilibrium propagation and recurrent backpropagation *Neural Comput.* **31** 312–29
[SBC19] Schmidt C, Buchholz M and Chandra M 2019 Optimization of loosely coupled inductive data transfer systems by non-Foster impedance matching *Adv. Radio Sci.* **17** 151–60
[Sce21] Scellier B 2021 A deep learning theory for neural networks grounded in physics (arXiv:2103.09985 [cs])
[Sce24] Scellier B 2024 A fast algorithm to simulate nonlinear resistive networks (arXiv:2402.11674)
[SEKK23] Scellier B, Ernoult M, Kendall J and Kumar S 2023 Energy-based learning algorithms for analog computing: a comparative study *Advances in Neural Information Processing Systems* (vol 36) pp 52705–31
[SLZ+11] Schindler J, Li A, Zheng M C, Ellis F M and Kottos T 2011 Experimental study of active LRC circuits with PT symmetries *Phys. Rev. A* **84** 040101
[SMBO22] Scellier B, Mishra S, Bengio Y and Ollivier Y 2022 Agnostic physics-driven deep learning (arXiv:2205.15021)
[SMS+24] Song Y, Millidge B, Salvatori T, Lukasiewicz T, Xu Z and Bogacz R 2024 Inferring neural activity before plasticity as a foundation for learning beyond backpropagation *Nat. Neurosci.* **27** 348–58
[Spa92] Spall J C 1992 Multivariate stochastic approximation using a simultaneous perturbation gradient approximation *IEEE Trans. Autom. Control* **37** 332–41
[vdSJ14] van der Schaft A J and Jeltsema D 2014 *Port-Hamiltonian Systems Theory: An Introductory Overview* (*Foundations and Trends in Systems and Control*)
[WZ89] Williams R J and Zipser D 1989 A learning algorithm for continually running fully recurrent neural networks *Neural Comput.* **1** 270–80
[YKWK23] Yi S-i, Kendall J D, Williams R S and Kumar S 2023 Activity-difference training of deep neural networks using memristor crossbars *Nat. Electron.* **6** 45–51
[ZS22] Zucchet N and Sacramento J 2022 Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation *Neural Comput.* **34** 2309–46
[ioppcitedata] Pourcel G 2026 Lagrangian-Equilibrium-Propagation (available at: https://github.com/guillaumepourcel/Lagrangian-Equilibrium-Propagation)