A PHYSICAL THEORY OF BACKPROPAGATION: EXACT GRADIENTS FROM THE LEAST-ACTION PRINCIPLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Backpropagation (BP) computes gradients through successive forward and backward passes. We derive the same gradients from an extension of Hamilton's action principle that describes dissipative systems using two coupled states. Their mean represents network activations, while their difference carries the sensitivities needed to compute gradients. The task loss supplies the output error, and the two states evolve together through interactions between neighboring computational stages. These interactions use the same transposed local derivatives as BP. At equilibrium the gradient is exactly BP; updating all stages together with unit-step Euler reaches this equilibrium within $2L$ updates for a feedforward chain of L stages. The formulation also lets us change how the gradient is reached. Grouping operations into larger blocks reduces the number of update rounds. Adding inertia and choosing its settings from the system's decay rates speeds up the tested recurrent gradient computations while preserving their equilibrium. The formulation opens the door to using tools from physics to study and design learning dynamics.

1 INTRODUCTION

For nearly four decades, backpropagation has been the engine of deep learning, yet it has resisted a physical derivation. As currently formulated, it is a symbolic procedure: a backward pass topologically distinct from inference, transmitting non-local error signals under synchronous global clocking (Crick, 1989). None of these features has a clear analog in physical reality, and this gap has motivated a long line of attempts to recover gradient computation from physically plausible dynamics.

Prior approaches derive gradients from energy models, constraints, and adjoint equations (Section 6). Holomorphic EP and exact predictive-coding variants also recover exact gradients under distinct assumptions (Laborieux & Zenke, 2022; Salvatori et al., 2022).

In this work we address this tension. We show that backpropagation is the equilibrium of a least-action flow on a doubled phase space, and that standard backpropagation is recovered exactly under the unit-step Euler discretization of this flow. Specializing the Bateman–Galley formalism for non-conservative classical mechanics (Bateman, 1931; Galley, 2013; Aykroyd et al., 2025) to the feedforward network flow, we construct a single global Lagrangian whose two conjugate fields jointly encode activations and sensitivities. The Euler–Lagrange equations of this Lagrangian, with the standard Rayleigh dissipation, generate a continuous flow whose unique stable equilibrium encodes the exact backpropagation gradient as the stress between the conjugate states. The task loss enters the action as a finite symmetry-breaking perturbation—no vanishing-nudge limit, no weight symmetry, no reverse-time integration, with exact transpose-Jacobian coupling retained. Inference and credit assignment unfold simultaneously through local interactions, as two facets of one variational dynamics.

The formulation opens the door to using tools from physics to study and design learning dynamics. We demonstrate two practical consequences here. Grouping operations into larger blocks reduces the number of rounds in which stages exchange information. Adding inertia lets states retain part of their previous motion; damping controls how that motion decays. Choosing these settings changes the time needed to obtain an accurate gradient while preserving the gradient at equilibrium. These results provide a starting point for broader use of the physics formalism and for analog or neuromorphic implementations.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Contributions.

- **Variational derivation.** The first derivation of exact backpropagation from Hamilton's least-action principle, obtained by specializing the Bateman–Galley formalism to the feedforward flow. The construction extends to the full second-order action (Appendix B).
- **Exact gradient, exact discrete recovery.** The unique stable equilibrium encodes the exact BP gradient as the stress between the two conjugate fields; discretization recovers standard BP digital algorithm. Empirically, on CIFAR-10 (Krizhevsky, 2009) (released by the authors for non-commercial research use) with a 9-layer VGG, gradients match BP to float32 precision ($\sim 10^{-7}$) at parity accuracy ($\sim 93\%$).
- **Control of convergence.** We demonstrate two ways to change how an exact gradient is reached: grouping operations to reduce update rounds, and adding inertia to reach accurate gradients sooner. We also identify cases where the added inertia requires more damping to remain stable.

2 Classical Backpropagation

We fix notation. Consider a standard L-layer feedforward network with input $\boldsymbol{x}_0 \in \mathbb{R}^{n_0}$ and output $\boldsymbol{a}_L \in \mathbb{R}^{n_L}$. For each layer $\ell = 1, \dots, L$, the pre-activation $\boldsymbol{z}_\ell \in \mathbb{R}^{n_\ell}$ and activation $\boldsymbol{a}_\ell \in \mathbb{R}^{n_\ell}$ are given by

$$\boldsymbol{z}_\ell = \boldsymbol{W}_\ell \boldsymbol{a}_{\ell-1} + \boldsymbol{b}_\ell, \qquad \boldsymbol{a}_\ell = \sigma_\ell(\boldsymbol{z}_\ell), \tag{1}$$

where $\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $\boldsymbol{b}_\ell \in \mathbb{R}^{n_\ell}$, and σ_ℓ is an elementwise nonlinearity; for $\ell = L$, $\boldsymbol{a}_L$ is the network output. We collect all parameters into $\boldsymbol{\theta} = \{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{L}$, and denote the supervised loss by $C(\boldsymbol{a}_L, \boldsymbol{y})$. Backpropagation computes gradients via the recursion

$$\boldsymbol{\delta}_\ell = (\boldsymbol{W}_{\ell+1}^\top \boldsymbol{\delta}_{\ell+1}) \odot \sigma_\ell'(\boldsymbol{z}_\ell), \qquad \boldsymbol{\delta}_L = \nabla_{\boldsymbol{a}_L} C \odot \sigma_L'(\boldsymbol{z}_L),$$

yielding $\partial C/\partial \boldsymbol{W}_\ell = \boldsymbol{\delta}_\ell \boldsymbol{a}_{\ell-1}^\top$ and $\partial C/\partial \boldsymbol{b}_\ell = \boldsymbol{\delta}_\ell$. The layer-wise structure induces a strict backward ordering, which we reinterpret as a global dynamical system in what follows.

3 From Layers to Global Flow

We now move from the layer-wise discrete dynamics of classical backpropagation (Section 2) to a *global* description. The key step is to reinterpret the layer index as a discrete time coordinate and then embed this into a continuous-time vector field defined on a single stacked state. This will allow us to apply the Lagrangian theory of non-conservative systems to the network as a whole in Section 3.4.

3.1 Global Activation Vector and Weight Matrix

We stack all layer activations and bias/input drives into single global vectors, and collect the inter-layer weights into a single global matrix:

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_L \end{bmatrix}, \quad \boldsymbol{\beta}(\boldsymbol{x}_0) = \begin{bmatrix} \boldsymbol{W}_1 \boldsymbol{x}_0 + \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_L \end{bmatrix}, \quad \boldsymbol{W} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{W}_2 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{W}_L & \boldsymbol{0} \end{bmatrix}, \tag{2}$$

with $\boldsymbol{a}, \boldsymbol{\beta} \in \mathbb{R}^n$ and $n = \sum_{\ell=1}^{L} n_\ell$. The matrix $\boldsymbol{W}$ is strictly lower block-triangular: information flows only from layer $\ell - 1$ to layer ℓ. The forward pass over all layers can then be written compactly as the unique fixed point of

$$\boldsymbol{a} = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{a} + \boldsymbol{\beta}(\boldsymbol{x}_0)), \tag{3}$$

where $\boldsymbol{\sigma}$ denotes the stacked nonlinearity acting elementwise.

3.2 The Forward Vector Field

In preparation for the physical formulation, we then proceed to generalize the feedforward dynamics from discrete to continuous time as the fixed point of a global relaxation dynamics. We introduce a time-dependent global activation state $\boldsymbol{a}(t)$ evolving under the vector field

$$\frac{d\boldsymbol{a}(t)}{dt} = \boldsymbol{\sigma}\big(\boldsymbol{W}\boldsymbol{a}(t) + \boldsymbol{\beta}(\boldsymbol{x}_0)\big) - \boldsymbol{a}(t) =: F(\boldsymbol{a}(t)). \tag{4}$$

By construction, the equilibria of (4) are exactly the solutions of the fixed-point equation (3). In particular, the standard forward pass configuration corresponds to the unique stable equilibrium $\bar{\boldsymbol{a}}$ satisfying $\bar{\boldsymbol{a}} = \boldsymbol{\sigma}(\boldsymbol{W}\bar{\boldsymbol{a}} + \boldsymbol{\beta}(\boldsymbol{x}_0))$.

3.3 NILPOTENCY

It is useful to notice that in this global framework, the global connection matrix $\boldsymbol{W}$ is strictly block lower-triangular and this implies a strong algebraic property.

Lemma 3.1 (Nilpotency of the Global Weight Matrix)**.** *For an L-layer feedforward network, the global matrix $\boldsymbol{W}$ satisfies:*

$$\boldsymbol{W}^L = \boldsymbol{0}.$$

Proof. Because $\boldsymbol{W}$ has nonzero blocks only immediately below the diagonal, each multiplication by $\boldsymbol{W}$ shifts nonzero contributions at most one block row down. After L such shifts, all contributions have moved past the last block row and vanish, hence $\boldsymbol{W}^L = \boldsymbol{0}$. □

Nilpotency captures the acyclic structure of feedforward networks: information propagates strictly layer by layer with no feedback cycles. This property will imply finite-time convergence for the relaxation dynamics constructed in later sections.

3.4 THE LEAST ACTION PATH

The first step is to build a Lagrangian for our formulation. To do this we first have to address the *non-reciprocal* nature of F: information flows from layer ℓ to $\ell + 1$ without symmetric feedback. This violation of Newton's third law implies that the network vector field F previously defined in Eq. 4 is *non-conservative* and cannot be derived from a standard scalar potential Goldstein et al. (2002).

To resolve this, we apply the Lagrangian theory of non-conservative systems. While inspired by the work of Bateman (1931), Keldysh (1964), and Galley (2013), we modify the formalism to enable learning through equilibria including dissipation (see Appendix A). Crucially, to variationally encode non-reciprocal interactions, this theory entails a doubling of the phase space (see also Scurria et al. (2026) for a related doubled-state construction in the equilibrium-propagation setting). We therefore introduce a conjugate pair of global state vectors: a "backward" state $\boldsymbol{x} \in \mathbb{R}^n$ and a "forward" state $\boldsymbol{z} \in \mathbb{R}^n$.

Starting from the definition of the force F, we can thus build the Lagrangian of our system (see Appendix A.3 for a detailed explanation):

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}) = \underbrace{(\boldsymbol{x} - \boldsymbol{z})^\top F\left(\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}\right)}_{\text{Lagrangian Lifting}} + \underbrace{C\left(\left[\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}\right]_L, \boldsymbol{y}\right)}_{\text{Cost}}. \tag{5}$$

Notice that in this Lagrangian we do not have a mass term (usually termed 'overdamped regime') to adapt to the dynamics of the previous section in which we do not have an acceleration term; this can be however included obtain the same equilibria and stability properties (see Appendix B). The inclusion of the cost directly in the Lagrangian breaks the symmetry of the two states $\boldsymbol{x}$ and $\boldsymbol{z}$ with a perturbation that will encode the gradient (see Appendix A.3).

The equations of motion for the doubled system are then obtained through the least action principle:

$$\delta\left(\int_{t_1}^{t_2} \mathcal{L}(x, \dot{x}, z, \dot{z})dt\right) = 0 \tag{6}$$

This gives us the equations

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = -\frac{\partial R}{\partial \dot{x}} \implies \dot{x} = +\frac{\partial \mathcal{L}}{\partial x} \\ -\left(\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{z}}\right) - \frac{\partial \mathcal{L}}{\partial z}\right) = -\frac{\partial R}{\partial \dot{z}} \implies \dot{z} = -\frac{\partial \mathcal{L}}{\partial z} \end{aligned} \tag{7}$$

where $R = \frac{1}{2}(\dot{x}^2 + \dot{z}^2)$ is the so called Rayleigh dissipation function. Thus we have:

$$\begin{aligned}
\frac{d\boldsymbol{x}}{dt} &= \underbrace{F\big(\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}\big)}_{\text{Network Relaxation}} + \underbrace{\tfrac{1}{2}\left[\nabla F(\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2})\right]^\top(\boldsymbol{x}-\boldsymbol{z})}_{\text{Backward Signal}} + \underbrace{\frac{1}{2}\begin{bmatrix}\mathbf{0}\\ \vdots \\ \nabla_{\boldsymbol{a}_L} C\big([\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}]_L, \boldsymbol{y}\big)\end{bmatrix}}_{\text{Cost}}, \\
\frac{d\boldsymbol{z}}{dt} &= \underbrace{F\big(\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}\big)}_{\text{Network Relaxation}} - \underbrace{\tfrac{1}{2}\left[\nabla F(\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2})\right]^\top(\boldsymbol{x}-\boldsymbol{z})}_{\text{Backward Signal}} - \underbrace{\frac{1}{2}\begin{bmatrix}\mathbf{0}\\ \vdots \\ \nabla_{\boldsymbol{a}_L} C\big([\tfrac{\boldsymbol{x}+\boldsymbol{z}}{2}]_L, \boldsymbol{y}\big)\end{bmatrix}}_{\text{Cost}}.
\end{aligned} \tag{8}$$

These dynamics consist of three distinct physical forces. **Network Relaxation** (first term in both equations) is identical for both states, driving $\boldsymbol{x}$ and $\boldsymbol{z}$ toward the forward pass configuration. **Backward Signal** uses the stress $\boldsymbol{x} - \boldsymbol{z}$, to propagate error information. **Cost** acts as an external force applied only to the output block.

Crucially, this formalism requires identical initial conditions for both x and z (also called "the physical limit", see Galley (2013)). In this scenario, absent the cost term, both states collapse to the same dynamics, corresponding to the forward pass dynamics previously defined in section 3.2. Thus the cost is perturbing the system in order to encode the gradients in the stress term $(\boldsymbol{x} - \boldsymbol{z})$ (see Appendix A.4).

The Euler discretization of the flow (8) yields a joint update in which both states evolve on the same lattice at every step providing, as we will prove later, exact gradients with only local computations in both space and time; we refer to this scheme as Dyadic Backpropagation (Algorithm 1, deferred to Appendix C). A simplified dynamical scheme (Appendix G), tested with identical accuracy, is also provided to facilitate hardware implementation.

3.5 Mean-Stress Variables

To better inspect the interaction between $\boldsymbol{x}$ and $\boldsymbol{z}$, we work with the mean–stress variables $\boldsymbol{m} = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{z})$ and $\boldsymbol{s} = \boldsymbol{x} - \boldsymbol{z}$ already introduced above. Differentiating these relations yields the transformed system:

$$\begin{aligned}
\frac{d\boldsymbol{m}}{dt} &= \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m} + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{m}, \\
\frac{d\boldsymbol{s}}{dt} &= \big(\boldsymbol{W}^\top \boldsymbol{D}(\boldsymbol{m}) - \boldsymbol{I}\big)\boldsymbol{s} + \nabla_{\boldsymbol{m}_L} C(\boldsymbol{m}_L, \boldsymbol{y}),
\end{aligned} \tag{9}$$

where $\boldsymbol{D}(\boldsymbol{m}) = \operatorname{diag}\big(\boldsymbol{\sigma}'(\boldsymbol{W}\boldsymbol{m} + \boldsymbol{\beta}(\boldsymbol{x}_0))\big)$ is the diagonal matrix of local activation derivatives evaluated at the current mean state.

In this representation, the physical roles of the two variables become clear: $\boldsymbol{m}$ relaxes toward a consistent forward activation pattern, while $\boldsymbol{s}$ records the tension that enforces this consistency and channels sensitivity information backward through the network.

Locality in space and relaxation time. Each stage reads the current activations and sensitivities of its neighbors. For a block f, it sends sensitivity backward by multiplying by the transposed local derivative, $J_f^\top \boldsymbol{s}$, called a vector–Jacobian product (VJP). No stored sequence of relaxation states is replayed backward, and no two separately reached equilibria need to be compared. Thus locality refers to the chosen graph and to the current relaxation time. The exact backward derivative operation is still required. Inputs and parameters remain fixed while each gradient is computed.

3.6 The Equilibrium States

We now inspect the equilibria of the system in the $\boldsymbol{m}$, $\boldsymbol{s}$ variables to better understand the configuration to which the system converges. The equilibria $\bar{\boldsymbol{m}}$ and $\bar{\boldsymbol{s}}$ satisfy

$$\boldsymbol{\sigma}(\boldsymbol{W}\bar{\boldsymbol{m}} + \boldsymbol{\beta}(\boldsymbol{x}_0)) = \bar{\boldsymbol{m}}, \qquad (\boldsymbol{I} - \boldsymbol{W}^\top \bar{\boldsymbol{D}})\,\bar{\boldsymbol{s}} = \nabla_{\bar{\boldsymbol{m}}_L} C(\bar{\boldsymbol{m}}_L, \boldsymbol{y}). \tag{10}$$

The first equation enforces relaxation to the forward pass configuration. The second yields exactly the same result as backpropagation (see Appendix F.1): $\bar{\boldsymbol{s}}$ collects the total derivatives of the cost with respect to the equilibrium activations, $\bar{\boldsymbol{s}}_\ell = \nabla_{\bar{\boldsymbol{m}}_\ell} C$. Stacking everything into a single vector and applying the diagonal derivative $\bar{\boldsymbol{D}}$ recovers the classical pre-activation errors:

$$\bar{\boldsymbol{s}} = \begin{bmatrix} \nabla_{\bar{\boldsymbol{m}}_1} C \\ \nabla_{\bar{\boldsymbol{m}}_2} C \\ \vdots \\ \nabla_{\bar{\boldsymbol{m}}_L} C \end{bmatrix}, \qquad \boldsymbol{\Delta} = \bar{\boldsymbol{D}}\, \bar{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2 \\ \vdots \\ \boldsymbol{\delta}_L \end{bmatrix}, \qquad \boldsymbol{\delta}_\ell = D_\ell(\bar{\boldsymbol{m}}_\ell)\, \bar{\boldsymbol{s}}_\ell, \tag{11}$$

where $D_\ell(\bar{\boldsymbol{m}}_\ell)$ is the diagonal derivative matrix at layer ℓ. Finally, this equilibrium configuration $(\bar{\boldsymbol{m}}, \bar{\boldsymbol{s}})$ is globally exponentially stable (see Appendix F.2).

3.7 Gradient Computation

Now we use the equilibrium configuration of our system to compute the total derivative of the cost with respect to the weights, denoted $\nabla_{\boldsymbol{W}} C$. Using the chain rule, the total gradient can be expressed in terms of the equilibrium variables:

$$\nabla_{\boldsymbol{W}} C = \sum_j \nabla_{\bar{\boldsymbol{m}}_j} C \frac{\partial \bar{\boldsymbol{m}}_j}{\partial \boldsymbol{W}} = \sum_j \bar{s}_j \frac{\partial (\boldsymbol{\sigma}(\boldsymbol{W}\bar{\boldsymbol{m}} + \boldsymbol{\beta}))_j}{\partial \boldsymbol{W}}. \tag{12}$$

Evaluating the partial derivative of the update rule $\bar{m}_i = \sigma_i(\sum_j W_{ij}\bar{m}_j + \beta_i)$ yields:

$$\frac{\partial \bar{m}_i}{\partial W_{ij}} = D_{ii}(\bar{\boldsymbol{m}})\, \bar{m}_j, \tag{13}$$

where $D_{ii}(\bar{\boldsymbol{m}})$ denotes the i-th block of $\boldsymbol{D}(\bar{\boldsymbol{m}})$. Substituting this into (12), we obtain the standard gradient outer product:

$$\nabla_{\boldsymbol{W}} C = \boldsymbol{\Delta}\, \bar{\boldsymbol{m}}^\top, \qquad \boldsymbol{\Delta} = \boldsymbol{D}(\bar{\boldsymbol{m}})\, \bar{\boldsymbol{s}}. \tag{14}$$

In layer form, this recovers the familiar local gradients:

$$\nabla_{\boldsymbol{W}_\ell} C = \boldsymbol{\delta}_\ell\, \bar{\boldsymbol{m}}_{\ell-1}^\top, \tag{15}$$

$$\nabla_{\boldsymbol{b}_\ell} C = \boldsymbol{\delta}_\ell. \tag{16}$$

These expressions highlight the two-factor structure of gradient computation: a pre-synaptic activation $\bar{\boldsymbol{m}}_{\ell-1}$ and a post-synaptic variable $\boldsymbol{\delta}_\ell$.

4 The Discrete Trace: Recovering Backpropagation

We now study the discretized version of the continuous-time $(\boldsymbol{m}, \boldsymbol{s})$ dynamics Eq. (9) to explicitly show equivalence to the backpropagation dynamics.

4.1 Canonical Discretization and the Natural Time Step

Applying forward Euler with step size $\Delta t > 0$ to the equation 9 gives the discrete updates:

$$\boldsymbol{m}^{k+1} = \boldsymbol{m}^k + \Delta t \left[\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{m}^k\right], \tag{17}$$

$$\boldsymbol{s}^{k+1} = \boldsymbol{s}^k + \Delta t \left[\left(\boldsymbol{W}^\top \boldsymbol{D}(\boldsymbol{m}^k) - \boldsymbol{I}\right)\boldsymbol{s}^k + \nabla_{\boldsymbol{m}_L} C(\boldsymbol{m}_L^k, \boldsymbol{y})\right]. \tag{18}$$

Standard layer-by-layer backpropagation corresponds to choosing a *unit* Euler step, $\Delta t = 1$. This specific choice exactly cancels the inertial term $(1 - \Delta t)\boldsymbol{m}^k$. In that case, the equations simplify to the coupled map:

$$\boldsymbol{m}^{k+1} = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0)), \tag{19}$$

$$\boldsymbol{s}^{k+1} = \boldsymbol{W}^\top \boldsymbol{D}(\boldsymbol{m}^k)\boldsymbol{s}^k + \nabla_{\boldsymbol{m}_L} C(\boldsymbol{m}_L^k, \boldsymbol{y}). \tag{20}$$

4.2 NILPOTENCY AND EXACT CONVERGENCE IN 2L STEPS

We now show that we obtain finite time convergence of the discretized dynamical equations (19)(20) in exactly $2L$ steps, where L corresponds to the depth of the network. This can be easily seen because $\boldsymbol{W}$ is strictly lower block-triangular. Then each block $\boldsymbol{m}_\ell$ depends only on upstream layers $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\ell-1}$, ensuring strictly sequential convergence.

Lemma 4.1 (Forward Layer Freezing)**.** *For the iteration* (19) *with initial condition $\boldsymbol{m}^0$, the ℓ-th block satisfies*

$$\boldsymbol{m}_\ell^k = \boldsymbol{a}_\ell \quad \textit{for all} \quad k \geq \ell,$$

where $\boldsymbol{a}_\ell$ denotes the standard forward activation at layer ℓ.

Proof. By induction on ℓ, using the strict block lower-triangularity of $\boldsymbol{W}$. See Appendix D for the full argument. □

Let $\bar{\boldsymbol{m}}$ denote the stacked forward activations at equilibrium. It follows that for all $k \geq L$, $\boldsymbol{m}^k = \bar{\boldsymbol{m}}$. Thus the forward relaxation converges in exactly L iterations.

We now need to inspect the convergence of the $\boldsymbol{s}$ variable.

Crucially, the $\boldsymbol{s}$ update in Eq. (20) uses the matrix $\boldsymbol{D}(\boldsymbol{m}^k)$. We can then use the fact that for $k \geq L$ we get

$$\boldsymbol{D}(\boldsymbol{m}^k) = \boldsymbol{D}(\bar{\boldsymbol{m}}) =: \bar{\boldsymbol{D}}, \quad \text{for all } k \geq L.$$

Consequently, for all subsequent steps $k \geq L$, the update for $\boldsymbol{s}$ becomes a linear dynamical system with a *constant* transition matrix $\bar{\boldsymbol{M}} = \boldsymbol{W}^\top \bar{\boldsymbol{D}}$:

$$\boldsymbol{s}^{k+1} = \bar{\boldsymbol{M}} \boldsymbol{s}^k + \nabla C, \quad \text{for } k \geq L.$$

The convergence of this phase relies on the algebraic properties of $\bar{\boldsymbol{M}}$. Because the network is feedforward, $\bar{\boldsymbol{M}}$ is strictly upper block-triangular and thus *nilpotent* with index L (i.e., $\bar{\boldsymbol{M}}^L = \boldsymbol{0}$).

This nilpotency guarantees that any transient (or residual) information accumulated in $\boldsymbol{s}$ during the initial transient phase (steps 0 to $L-1$, when $\boldsymbol{D}$ is changing) is progressively shifted out of the network. After exactly L applications of the constant operator $\bar{\boldsymbol{M}}$ (i.e., at step $k = L + L = 2L$), the memory of the initial transient is completely annihilated, and the state settles exactly into the fixed point previously discussed (see Eq. 11).

More formally:

Theorem 4.2 (Discrete Equivalence to Standard Backpropagation)**.** *Let the coupled system evolve simultaneously through Eqs. 19 and 20. Then:*

1. ***Forward Settling ($k < L$):*** *The primal state converges to the valid network activations at $k = L$, i.e., $\boldsymbol{m}^k = \bar{\boldsymbol{m}}$ for all $k \geq L$.*
2. ***Backward Flushing ($L \leq k < 2L$):*** *With $\boldsymbol{m}$ fixed, the dual state flushes out transient errors and converges to the exact backpropagation gradients at $k = 2L$, i.e., $\boldsymbol{s}^k = \bar{\boldsymbol{s}}$ for all $k \geq 2L$.*

The system thus reproduces the backpropagation gradient autonomously in $2L$ steps, without external mode switching.

The corresponding pseudocode (Algorithm 2) is deferred to Appendix E.

Together, Theorem 4.2 and Algorithm 2 establish that standard backpropagation is the exact discrete trace of the continuous $\boldsymbol{x}$–$\boldsymbol{z}$ dynamics in Eq. 8. The conventional two-phase procedure of the forward and backward passes is thus revealed to be a computational optimization of this single physical process, skipping redundant updates during the system's natural settling phases (we stop updating $\boldsymbol{m}$ as soon as it settles to its equilibrium).

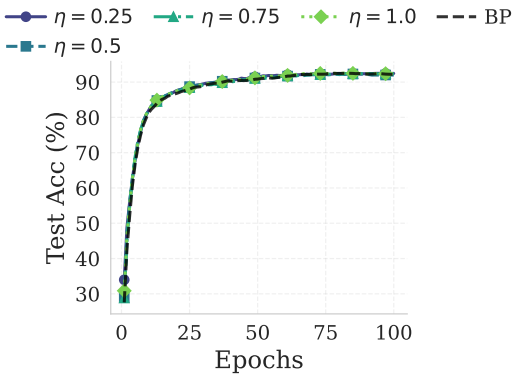

Figure 1: **Generalization Performance.** Test accuracy evolution for varying step sizes η matches the BP baseline (dashed black), reaching $\sim 93\%$.

5 Empirical Validation

We empirically validate the $(\boldsymbol{x}, \boldsymbol{z})$ dynamics Eq. (8) on CIFAR-10, comparing performance and internal dynamics against standard backpropagation (BP). The dynamics are integrated via forward Euler as detailed in Algorithm 1 (Appendix C).

5.1 Setup

We use a 9-layer VGG-style CNN ($\sim$5M parameters) with four convolutional blocks (Conv-ReLU-Conv-ReLU-Stride2Pool) of channel dimensions $[64, 128, 256, 512]$ and a linear classifier. Models are trained for 100 epochs with batch size 64 using SGD with Nesterov momentum ($\mu = 0.9$), weight decay (5×10^{-4}), cosine learning-rate annealing ($0.035 \to 0.0002$), standard augmentation (random crops, horizontal flips, Cutout), and label smoothing ($\epsilon = 0.1$). The Euler step size is varied over $\eta \in \{0.25, 0.50, 0.75, 1.00\}$ with up to $K_{\max} = 1000$ relaxation iterations per training step; early stopping triggers when consecutive iterates differ by less than 10^{-6} in Euclidean norm, and all runs share random seeds. Gradient fidelity is measured against PyTorch 2.7.1 autodiff (Paszke et al., 2019) (BSD-3-Clause license) references via the global cosine similarity $\text{CosSim} = \langle \nabla_{\text{relax}}, \nabla_{\text{BP}} \rangle / (\|\nabla_{\text{relax}}\| \|\nabla_{\text{BP}}\|)$, the relative error $\|\nabla_{\text{relax}} - \nabla_{\text{BP}}\| / \|\nabla_{\text{BP}}\|$, and per-layer cosine similarity (additional metrics in Appendix H). Code with fixed seeds is provided in the supplementary material for reproducibility. The CNN9 training runs used a single NVIDIA RTX 6000 Ada (48 GB) with a 32-core CPU and 128 GB DDR5 RAM; each η run takes $\sim$2 GPU-hours ($\sim$8 GPU-hours).

5.2 Results

Our experiments confirm the theoretical predictions, demonstrating that the proposed method achieves robust stability across a wide range of step sizes, consistent with theoretical bounds.

Generalization Performance. As shown in Figure 1, the choice of the discretization step size η in Algorithm 1 does not negatively impact the learning trajectory. Algorithm 1 achieves test accuracy parity with standard Backpropagation (BP), reaching a peak of roughly 93%. This indicates that the discrete-time dynamics successfully approximate the learning signal of the ideal gradient.

Gradient Fidelity and Dynamics. Beyond generalization accuracy, we analyze the internal mechanics of the method. Figure 2 corroborates our core theoretical claims along three axes. *Precision* (left panel): the algorithm recovers the true gradient with low global relative error, almost entirely driven by a mismatch in Euclidean norm rather than directional deviation (see Appendix H.3). *Convergence speed* (center panel): as $\eta \to 1$, the number of required relaxation steps collapses toward the finite recovery bound $2L = 18$ (twice the network depth). *Alignment* (right panel): the layer-wise log-misalignment $\log_{10}(1 - \cos\theta)$ hovers near the IEEE 754 Float32 machine precision limit ($\approx 10^{-7}$), making the physical relaxation effectively indistinguishable from the symbolic algorithm (further details in Appendix H.6).

Further gradient-fidelity measurements appear in Appendix H.

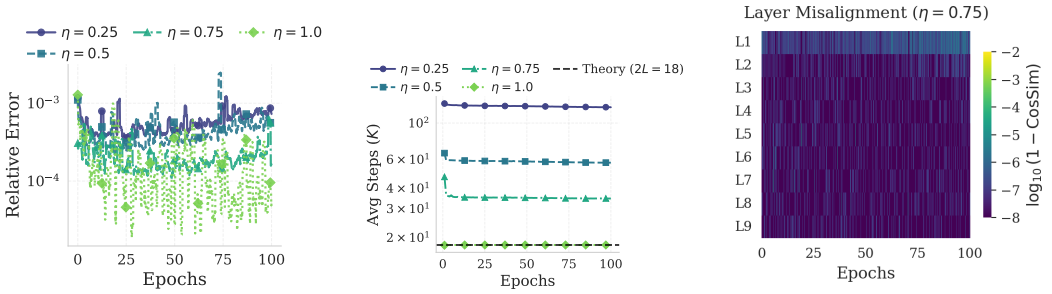

Figure 2: Empirical Validation of Dynamics. (left) **Relative gradient error** shows accurate recovery of ∇_{BP}. (center) **Relaxation steps** converge to the theoretical limit $2L = 18$ as $\eta \to 1$. (right) **Log-misalignment heatmap** ($\eta = 0.75$) plots $\log_{10}(1 - \text{CosSim})$ for each layer L_i. Values hover between -6 and -8, bounded by IEEE 754 Float32 machine precision ($\approx 10^{-7}$), rendering relaxation gradients basically indistinguishable from BP.

5.3 Applications of the Formalism

We use the formalism to accelerate gradient computation and to reduce communication rounds. In both cases, weights remain fixed while activation and sensitivity states are updated.

Accelerating gradient computation by adding mass. We add a mass term to the activation and sensitivity equations. In our variables, Eq. (9) becomes

$$\begin{aligned} M\ddot{\boldsymbol{m}} + \gamma\dot{\boldsymbol{m}} &= \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m} + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{m}, \\ M\ddot{\boldsymbol{s}} + \gamma\dot{\boldsymbol{s}} &= (\boldsymbol{W}^\top \boldsymbol{D}(\boldsymbol{m}) - \boldsymbol{I})\boldsymbol{s} + \boldsymbol{g}(\boldsymbol{m}), \end{aligned} \tag{21}$$

where $\boldsymbol{g}$ is the loss gradient embedded at the output, $M > 0$ is the mass, and $\gamma > 0$ controls damping. A dot denotes the rate of state change; two dots denote its acceleration. Mass lets the states retain velocity, while damping reduces that velocity. Setting $M = 0, \gamma = 1$ recovers the original equations. At equilibrium the time derivatives vanish, leaving the same forward and backward conditions and hence the same gradient (Appendix B).

To implement these equations, collect $\boldsymbol{q} = (\boldsymbol{m}, \boldsymbol{s})$ and their right-hand sides in $\boldsymbol{r}(\boldsymbol{q})$. Updating velocity before the state at each time step Δt gives the update rule

$$\boldsymbol{v}^{k+1} = \beta\boldsymbol{v}^k + \alpha\boldsymbol{r}(\boldsymbol{q}^k), \qquad \boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \boldsymbol{v}^{k+1}, \tag{22}$$

with $\alpha = \Delta t^2/M$, $\beta = 1 - \gamma\Delta t/M$, and $\boldsymbol{v}^0 = \boldsymbol{0}$. Here $\boldsymbol{v}$ stores the previous state increment. In the tested setting $0 \le \beta < 1$, each increment retains a fraction of the previous increment and adds the current correction. This is how the mass term becomes momentum on the internal states.

We test a linear recurrent model $\boldsymbol{h}^\star = \boldsymbol{W}\boldsymbol{h}^\star + \boldsymbol{B}\boldsymbol{u}$ with symmetric $\boldsymbol{W}$. At spectral radius $\rho = 0.99$ (the largest absolute eigenvalue of $\boldsymbol{W}$), the original relaxation needs many updates. Choosing α, β from the eigenvalue range reduces the median from 930 to 92, at relative gradient error 10^{-3} over five float32 seeds. Both methods use one local VJP per update; inertia also stores $\boldsymbol{v}$. A continuous control compares $M = 0$ with $M = 1$, using the same $1 + \rho$ normalization of the right-hand sides. Median model settling time decreases from 1857.71 to 184.78. Nonlinear tests show that damping suitable for symmetric connections can fail for nonsymmetric ones. A rule chosen after those tests restores convergence with smaller gains. Appendix H.1 gives the settings and full controls.

Reducing communication rounds by grouping operations. We choose where to place the evolving states in the same network. For example, a convolution and its ReLU can form one block: both operations are evaluated locally, and the activation and sensitivity states are kept at the block output. More generally, consecutive maps can be combined as $B_j = f_{\ell+1} \circ f_\ell$. With K blocks and unit-step Euler, all blocks update simultaneously:

$$\begin{aligned} \boldsymbol{m}_j^{k+1} &= B_j(\boldsymbol{m}_{j-1}^k), \\ \boldsymbol{s}_j^{k+1} &= J_{B_{j+1}}(\boldsymbol{m}_j^k)^\top \boldsymbol{s}_{j+1}^k, \qquad j < K, \end{aligned} \tag{23}$$

with input $\boldsymbol{m}_0 = \boldsymbol{x}_0$ and output sensitivity $\boldsymbol{s}_K^{k+1} = \nabla_{\boldsymbol{m}_K} C(\boldsymbol{m}_K^k, \boldsymbol{y})$. The first update passes activations forward; the second passes sensitivities backward. $J_B^\top$ applies the chain rule within a block, so grouping preserves the network function and its equilibrium gradient. Larger blocks do more local work but require fewer rounds of exchanges between blocks. This provides a trade-off when communication or synchronization is costly. For the same reduced CNN, operation, layer, and block partitions give 15, 7, and 4 stages. At $\eta = 1$, five seeds give median stopping counts of 30, 14, and 8 updates, with median relative gradient error below 4×10^{-8}. This is $3.75\times$ fewer update rounds; hardware runtime has not been measured. These are different stage boundaries in Eq. (9), not a test of the separate x/z derivative variant in Appendix G.

6 Related Work

Table 1 in Appendix I summarizes gradient guarantees and assumptions.

LeCun's constrained formulation. LeCun (LeCun, 1988) formulates BP as constrained optimization: each forward equation is an equality constraint, and its multiplier gives a backward sensitivity. His feedforward derivation obtains BP from stationarity; he explicitly distinguishes this Lagrange function from a physical Lagrangian (his footnote 2). These conditions alone do not specify a relaxation that reaches them. Our extension of Hamilton's action principle derives dissipative dynamics for activations and sensitivities together. The difference $\boldsymbol{x} - \boldsymbol{z}$ carries sensitivities, allowing gradients to be computed from current neighboring states. For feedforward chains, unit-step Euler recovers BP exactly (Section 4). The action thus describes how the gradient is reached, including the dissipation needed to settle. It also opens the door to tools from physics for studying and modifying learning dynamics, as demonstrated by mass and damping in Section 5.3.

CHL and equilibrium propagation. CHL (Movellan, 1991) contrasts free and clamped equilibria. Standard EP (Scellier & Bengio, 2017) uses a weak output nudge and recovers the energy-model gradient as the nudge tends to zero. These two-phase methods compare quantities across equilibria. Our states evolve together, using current neighboring states and a finite loss term, without storing a separate free-phase equilibrium. The feedforward couplings need not be reciprocal, although exact backward derivatives are required.

Dual propagation. DP uses two oppositely nudged states in one inference phase (Høier et al., 2023); its Lagrangian interpretation modifies LeCun's constraints (Høier & Zach, 2023). Our distinction is the derivation of both states' evolution from a dissipative action, with exact BP at equilibrium without a small-nudge limit. For a feedforward chain, unit-step Euler reaches this gradient within $2L$ updates.

Continuous adjoints. Trajectory adjoints integrate sensitivities backward along a forward solution (Chen et al., 2018). Our two states evolve forward in relaxation time to compute a fixed-point gradient. This avoids reversing that relaxation, but does not establish exact online BPTT for evolving inputs.

Recurrent backpropagation. Recurrent BP already computes fixed-point sensitivities through an auxiliary linear relaxation (Almeida, 1990; Pineda, 1987). We derive the activation and sensitivity equations together from one action, which also provides the mass extension tested in Section 5.3.

Limitation: weight transport. Eq. (8) retains $\boldsymbol{W}^\top$: the backward connections must use the forward weights. Using feedback alignment (Lillicrap et al., 2016; Nøkland, 2016) or the Kolen–Pollack scheme (Kolen & Pollack, 1994) would require reassessing gradient exactness.

7 Conclusion

We have derived backpropagation from an action principle for dissipative systems with two coupled states. Activations and sensitivities evolve using current information from neighboring stages and exact backward derivatives. Updating all stages together with unit-step Euler recovers BP within $2L$ updates for the feedforward chain. Grouping operations and adding inertia change how the gradient is computed and how quickly it is reached, while preserving its value at equilibrium. In ongoing work, we are combining this formalism with the replica method.

AI USE STATEMENT

Generative AI tools were used to assist with writing and editing the manuscript, with code development, and with preparation of the submission files. The authors are responsible for the final content, including all text, code, mathematical claims, and experimental results.

REPRODUCIBILITY STATEMENT

The theoretical derivations and assumptions are presented in Sections 3 and 4, with additional derivations, proofs, and pseudocode in Appendices A–G. Section 5 describes the experimental setup, and Appendix H reports additional results. The supplementary code includes the training and plotting scripts, fixed random seeds, dependency versions, and instructions for running the experiments.

REFERENCES

Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial neural networks: concept learning*, pp. 102–111. 1990.

Christopher Aykroyd, Adrien Bourgoin, and Christophe Le Poncin-Lafitte. Hamiltonian treatment of non-conservative systems. *arXiv preprint arXiv:2507.18658*, 2025.

Harry Bateman. On dissipative systems and related variational principles. *Physical Review*, 38(4): 815, 1931.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11:3625, 2020. doi: 10.1038/s41467-020-17236-y.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Francis Crick. The recent excitement about neural networks. *Nature*, 337, 1989.

Chad R Galley. Classical mechanics of nonconservative systems. *Physical review letters*, 110(17): 174301, 2013.

Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics*. Addison Wesley, 3rd edition, 2002.

Rasmus Høier, D. Staudt, and Christopher Zach. Dual propagation: accelerating contrastive hebbian learning with dyadic neurons. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Rasmus Kjær Høier and Christopher Zach. A Lagrangian perspective on dual propagation. In *NeurIPS Workshop on Machine Learning with New Compute Paradigms (MLNCP)*, 2023.

L. V. Keldysh. Diagram technique for nonequilibrium processes. *Zh. Eksp. Teor. Fiz.*, 47:1515, 1964. [Sov. Phys. JETP **20**, 1018 (1965)].

John F Kolen and Jordan B Pollack. Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN)*, volume 3, pp. 1375–1380, 1994.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Axel Laborieux and Friedemann Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Advances in Neural Information Processing Systems*, 35: 12950–12963, 2022.

Yann LeCun. A theoretical framework for back-propagation. 1988.

Timothy P. Lillicrap, Daniel Crownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, (7), 2016.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pp. 10–17. Elsevier, 1991.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 8024–8035, 2019.

Fernando Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987.

Tommaso Salvatori, Yuhang Song, Zhenghua Xu, Thomas Lukasiewicz, and Rafal Bogacz. Reverse differentiation via predictive coding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):8150–8158, 2022. doi: 10.1609/aaai.v36i7.20788.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Antonino Emanuele Scurria, Dimitri Vanden Abeele, Bortolo Matteo Mognetti, and Serge Massar. Equilibrium propagation for non-conservative systems. *International Conference on Machine Learning*, 2026.

A Overview on Out-of-Equilibrium Mechanics

This appendix provides a brief overview of the classical mechanics of non-conservative systems to motivate the definition of the global Lagrangian functional $\mathcal{L}(\boldsymbol{x}, \boldsymbol{z})$ presented in Eq. 5. Our construction draws on the formalism developed by Galley (2013), Keldysh (1964) and Bateman (1931), adapted here for the specific context of neural dynamics.

A.1 Lagrangian Mechanics for Dissipative Systems

We follow the logic of Galley's framework, and refer the reader to the recent review Aykroyd et al. (2025) for a detailed and pedagogical derivation. Our framework extends the Bateman–Galley formalism, in which an auxiliary system absorbs the energy lost by a dissipative one so as to guarantee total conservation, and adapts it to the relaxation dynamics relevant to feedforward networks.

Conservative, dissipative, and overdamped regimes. In a conservative system, the dynamics follow

$$m\ddot{x} = -\frac{\partial U(x)}{\partial x}, \tag{24}$$

where m is the mass, U is a scalar potential, and the overdot denotes the time derivative. A dissipative system carries an additional damping term,

$$m\ddot{x} + \gamma\dot{x} + \frac{\partial U(x)}{\partial x} = 0, \tag{25}$$

whose dynamics is recovered from the modified Lagrange equation

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} + \frac{\partial R}{\partial \dot{x}} = 0, \tag{26}$$

where $R = \frac{1}{2}\dot{x}^2$ is the Rayleigh dissipation function for linear damping (we set $\gamma = 1$). In machine learning, and in equilibrium-based theories such as ours, the natural setting is one of first-order dynamics. Physics calls this the *overdamped* regime (e.g. Brownian motion): the acceleration term is neglected and one recovers

$$\dot{x} = -\frac{1}{\gamma}\frac{\partial U(x)}{\partial x}. \tag{27}$$

We adopt this regime throughout the main text purely for clarity of exposition; the construction extends without difficulty to the full massive case, recovering the same equilibria and stability properties, as briefly shown in Appendix B. In this regime, (26) simplifies to

$$-\frac{\partial L}{\partial x} + \frac{\partial R}{\partial \dot{x}} = 0 \quad \Longleftrightarrow \quad \dot{x} = \frac{\partial L}{\partial x}. \tag{28}$$

Doubling and the action principle. To accommodate non-reciprocal forces, the Bateman–Galley formalism doubles the configuration space, replacing x by a conjugate pair (x, z). Because we work in the overdamped regime, the doubled Lagrangian carries no kinetic term, and the natural action takes the form

$$S = \int_{t_1}^{t_2} \mathcal{L}(x, \dot{x}, z, \dot{z})\, dt = \int_{t_1}^{t_2} \big[L(x, \dot{x}) - L(z, \dot{z})\big]\, dt, \tag{29}$$

in which the z-path is integrated backward in time relative to the x-path; this is the origin of the sign asymmetry that will appear in the equations of motion.

Antisymmetric coupling. Following Aykroyd et al. (2025) and the Galley construction, we introduce an antisymmetric coupling term $K(x, z)$ between the two paths,

$$S = \int_{t_1}^{t_2} \big[L(x, \dot{x}) - L(z, \dot{z}) + K(x, \dot{x}, z, \dot{z})\big]\, dt, \tag{30}$$

which breaks the path symmetry and injects non-conservative effects into the equations of motion. Applying (26) to (30) yields

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \dot{x} = \frac{\partial \mathcal{L}}{\partial x}, \qquad -\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{z}}\right) + \dot{z} = -\frac{\partial \mathcal{L}}{\partial z}. \tag{31}$$

The construction of the explicit form of K and the generalized Lagrangian $\mathcal{L}$ for the network flow F is carried out in Appendix A.3.

A.2 From a Non-Reciprocal Flow to a Variational Principle

The forward dynamics $\dot{\boldsymbol{a}} = F(\boldsymbol{a}) = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{a} + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{a}$ introduced in Sec. 3.2 is intrinsically *non-reciprocal*: information flows from layer ℓ to $\ell + 1$ but not in reverse, and the corresponding Jacobian ∇F is strictly block-lower-triangular and therefore non-self-adjoint. The Helmholtz conditions of the inverse problem of the calculus of variations state that a vector field admits a scalar potential—i.e., can be written as $F = -\nabla H$ for some H—if and only if its Jacobian is symmetric. Consequently, no scalar potential exists on the original n-dimensional state space whose gradient generates the feedforward flow, and the standard "energy minimisation" route to a variational principle is unavailable.

The Bateman–Galley formalism circumvents this obstruction by doubling the configuration space. The rotational (non-reciprocal) component of F, which has no scalar potential in the original n-dimensional space, is absorbed into a bilinear coupling between two conjugate fields $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n$ on the doubled $2n$-dimensional space, where it admits a well-defined Lagrangian description. The physical motion of the original system is recovered on the diagonal submanifold $\boldsymbol{x} = \boldsymbol{z}$, while the off-diagonal direction $\boldsymbol{x} - \boldsymbol{z}$ provides the additional degree of freedom needed to encode the non-reciprocity variationally. We adapt this construction to the network flow F in Appendix A.3 below.

A.3 Constructing the Lagrangian

We now make explicit how the global Lagrangian (5) is obtained from the bare forward vector field F. The construction follows the inverse-problem recipe presented in Section V of Aykroyd et al. (2025) for arbitrary second-order ODEs, specialised here to the first-order setting that is natural for feedforward relaxation dynamics.

Notation In this section we are briefly describing how to construct the Lagrangian in our specific case; we will map our framework (even if we initially obtained the formulation through a geometrical argument) onto the construction in section V of Aykroyd et al. (2025) to give the reader more material and better clarity. In our formalism the conjugate pair $(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{R}^{2n}$, which corresponds to the $(\boldsymbol{q}_\uparrow, \boldsymbol{q}_\downarrow)$ parametrisation of Aykroyd et al. (2025). The change of basis to $\left(\frac{1}{2}(\boldsymbol{x} + \boldsymbol{z}),\, \boldsymbol{x} - \boldsymbol{z}\right)$ then corresponds to their $(\boldsymbol{q}_+, \boldsymbol{q}_-)$ parametrisation, with $\frac{1}{2}(\boldsymbol{x} + \boldsymbol{z})$ playing the role of the physical (plus) variable and $\boldsymbol{x} - \boldsymbol{z}$ the role of the virtual (minus) variable. The physical limit corresponds to $\boldsymbol{x} = \boldsymbol{z}$.

Reconstruction from a target ODE. For a generic second-order target ODE $\ddot{\boldsymbol{q}} = \boldsymbol{U}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$, Aykroyd et al. (2025) show that the doubled-variable Lagrangian

$$\mathcal{L} = \boldsymbol{q}_- \cdot \left(\ddot{\boldsymbol{q}}_+ - \boldsymbol{U}(\boldsymbol{q}_+, \dot{\boldsymbol{q}}_+, t)\right) \tag{32}$$

trivially reproduces the equations of motion as its physical-limit Euler–Lagrange equation. Integrating (32) by parts removes the explicit acceleration and yields the equivalent linear (Lie) form

$$\mathcal{L}(\boldsymbol{q}_\pm, \dot{\boldsymbol{q}}_\pm, t) = \dot{\boldsymbol{q}}_- \cdot \dot{\boldsymbol{q}}_+ + \boldsymbol{q}_- \cdot \boldsymbol{U}(\boldsymbol{q}_+, \dot{\boldsymbol{q}}_+, t), \tag{33}$$

which is Eq. (121) of Aykroyd et al. (2025). The first term is the canonical kinetic coupling associated with the standard momentum choice $\boldsymbol{P} = \dot{\boldsymbol{q}}_+$ (see Eq. (126) of Aykroyd et al. (2025)); the second encodes the force $\boldsymbol{U}$.

Specialisation to the network flow. The forward relaxation defined in Sec. 3.2,

$$\frac{d\boldsymbol{a}}{dt} = F(\boldsymbol{a}) = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{a} + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{a}, \tag{34}$$

is first-order in time. Adapting (33) to this regime amounts to neglecting the acceleration (kinetic) term $\dot{\boldsymbol{q}}_- \cdot \dot{\boldsymbol{q}}_+$ and replacing the second-order acceleration field $\boldsymbol{U}$ by the first-order velocity field F. This is no loss of generality: the analogous derivation retaining a mass term reaches the same equilibria and stability properties, and is treated separately in Appendix B. Translating to the $(\boldsymbol{x}, \boldsymbol{z})$ notation, the remaining contribution is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x} - \boldsymbol{z})^\top F\left(\frac{\boldsymbol{x} + \boldsymbol{z}}{2}\right), \tag{35}$$

which is precisely the Lagrangian appearing in (5), before the symmetry-breaking cost is added. The auxiliary variable $\boldsymbol{z}$ then plays exactly the role of the virtual copy in the Galley framework: it absorbs the rotational (non-conservative) component of F off the physical submanifold $\boldsymbol{x} = \boldsymbol{z}$, while leaving the on-shell motion unchanged. Adding the task loss $C\big([\frac{1}{2}(\boldsymbol{x}+\boldsymbol{z})]_L, \boldsymbol{y}\big)$ to (35) as a symmetry-breaking perturbation yields the full global Lagrangian (5) used throughout the paper.

A.4 Symmetry Breaking as Credit Assignment

The variational principle established above successfully generates the forward dynamics on the diagonal manifold $\boldsymbol{x} = \boldsymbol{z}$. On this manifold, the system possesses a gauge symmetry: the forward state $\boldsymbol{z}$ is redundant, and the stress $(\boldsymbol{x} - \boldsymbol{z})$ is zero.

We propose to exploit this symmetry to perform credit assignment. Specifically, we introduce the task loss $C(\boldsymbol{m})$ as a symmetry-breaking potential acting on the output block. By adding this cost to the Lagrangian, we intentionally perturb the gauge symmetry of the system.

This perturbation exerts opposite forces on the pair: it drives $\boldsymbol{z}$ to descend the loss landscape while forcing $\boldsymbol{x}$ to ascend.

This "tug-of-war" induces a physical separation $\boldsymbol{s} = \boldsymbol{x} - \boldsymbol{z}$. The evolution of this stress variable naturally recovers the exact gradients. Thus, in our framework, backpropagation emerges not as a symbolic operation, but as the physical relaxation of the stress induced by explicitly breaking the symmetry of the forward pass.

B Extension to Second-Order Dynamics

In the main text we worked in the first-order regime, in which the acceleration term of the Aykroyd–Galley Lagrangian is dropped. We show here that this is purely an expository choice: the construction extends without modification to the full second-order (massive) case, and recovers the same equilibria and the same gradient signal.

Massive Lagrangian. Endowing the doubled state $(\boldsymbol{x}, \boldsymbol{z})$ with a mass m and reinstating the kinetic coupling of Eq. (121) in Aykroyd et al. (2025) yields the global Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \dot{\boldsymbol{x}}, \dot{\boldsymbol{z}}) = m\,(\dot{\boldsymbol{x}} - \dot{\boldsymbol{z}})^\top \left(\frac{\dot{\boldsymbol{x}} + \dot{\boldsymbol{z}}}{2}\right) + (\boldsymbol{x} - \boldsymbol{z})^\top F\left(\frac{\boldsymbol{x} + \boldsymbol{z}}{2}\right) + C\left(\left[\frac{\boldsymbol{x} + \boldsymbol{z}}{2}\right]_L, \boldsymbol{y}\right), \tag{36}$$

which differs from (5) only by the leading kinetic term. The remaining ingredients — the bilinear lifting of the network flow F and the symmetry-breaking cost — are unchanged.

Equations of motion. Extremising the action of (36) together with the Rayleigh dissipation function $R = \frac{1}{2}(\dot{\boldsymbol{x}}^2 + \dot{\boldsymbol{z}}^2)$ gives

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{x}}}\right) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = -\frac{\partial R}{\partial \dot{\boldsymbol{x}}} \implies m\ddot{\boldsymbol{x}} + \dot{\boldsymbol{x}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}, \\ -\left(\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{z}}}\right) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}}\right) = -\frac{\partial R}{\partial \dot{\boldsymbol{z}}} \implies m\ddot{\boldsymbol{z}} + \dot{\boldsymbol{z}} = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}}. \end{aligned} \tag{37}$$

In the limit $m \to 0$ these collapse exactly to the first-order flow used in the main text. For $m > 0$, (37) is a damped second-order system whose first-order limit recovers our dynamics verbatim.

Equilibria and gradient signal. At any equilibrium $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$ of (37) we have $\ddot{\boldsymbol{x}} = \dot{\boldsymbol{x}} = \boldsymbol{0}$ and $\ddot{\boldsymbol{z}} = \dot{\boldsymbol{z}} = \boldsymbol{0}$, so the kinetic and damping terms drop out and the equilibrium conditions reduce to

$$\left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\right|_{\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}} = \boldsymbol{0}, \qquad \left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}}\right|_{\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}} = \boldsymbol{0}, \tag{38}$$

which are exactly the equilibrium conditions analysed in Sec. 3.5. Consequently the equilibrium configuration $(\bar{\boldsymbol{m}}, \bar{\boldsymbol{s}})$ is unchanged, and the gradient extracted from the stress variable $\bar{\boldsymbol{s}} = \bar{\boldsymbol{x}} - \bar{\boldsymbol{z}}$ remains the exact backpropagation gradient (11).

Stability. The Jacobian of the linearised second-order system inherits the spectral structure of the first-order analysis of Appendix F.2, augmented by the standard damped-oscillator eigenvalue pattern: each eigenvalue $\lambda = -1$ of the first-order Jacobian becomes a pair of roots of the characteristic polynomial $m\mu^2 + \mu + 1 = 0$, whose real parts are strictly negative for any $m > 0$. The equilibrium therefore remains globally exponentially stable, with a relaxation rate that interpolates smoothly between the overdamped and underdamped regimes as m is varied.

In summary, the massive extension preserves both the forward-pass equilibria and the backpropagation gradients, and merely modifies the transient relaxation profile. The first-order treatment of the main text is therefore a notational simplification rather than a structural restriction.

C ALGORITHM

The pseudocode for the integration of the Dyadic Backpropagation scheme is given below.

Algorithm 1 Dyadic Backpropagation (DBP)

Input: $\boldsymbol{x}_0, \boldsymbol{y}$, parameters $\{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{L}$, step size $\eta > 0$, tolerance $\varepsilon > 0$
Initialize $\boldsymbol{x}^0 = \boldsymbol{z}^0$
for $k = 0, 1, 2, \dots$ **do**
  $\boldsymbol{m}^k \leftarrow \frac{1}{2}(\boldsymbol{x}^k + \boldsymbol{z}^k)$
  $\boldsymbol{D}^k \leftarrow \mathrm{diag}\big(\boldsymbol{\sigma}'(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0))\big)$
  Compute loss gradient $\boldsymbol{g}^k \leftarrow \nabla_{\boldsymbol{m}} C(\boldsymbol{m}^k, \boldsymbol{y})$, embedded on the output block
  Compute velocity fields from Eqs. (8)

$$\begin{aligned}\dot{\boldsymbol{x}}^k &= \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{x}^k \\ &\quad + \tfrac{1}{2}\boldsymbol{W}^\top \boldsymbol{D}^k(\boldsymbol{x}^k - \boldsymbol{z}^k) + \tfrac{1}{2}\boldsymbol{g}^k, \\ \dot{\boldsymbol{z}}^k &= \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0)) - \boldsymbol{z}^k \\ &\quad - \tfrac{1}{2}\boldsymbol{W}^\top \boldsymbol{D}^k(\boldsymbol{x}^k - \boldsymbol{z}^k) - \tfrac{1}{2}\boldsymbol{g}^k\end{aligned}$$

  Update states by Euler integration:

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \eta\, \dot{\boldsymbol{x}}^k, \qquad \boldsymbol{z}^{k+1} \leftarrow \boldsymbol{z}^k + \eta\, \dot{\boldsymbol{z}}^k$$

  if $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|_2 + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|_2 < \varepsilon$ **then**
    break
  end if
end for
Equilibrium variables:

$$\bar{\boldsymbol{x}} = \boldsymbol{x}^k, \quad \bar{\boldsymbol{z}} = \boldsymbol{z}^k, \quad \bar{\boldsymbol{m}} = \tfrac{1}{2}(\bar{\boldsymbol{x}} + \bar{\boldsymbol{z}}), \quad \bar{\boldsymbol{s}} = \bar{\boldsymbol{x}} - \bar{\boldsymbol{z}}.$$

Gradient (equivalent to standard BP):

$$\frac{dC}{d\boldsymbol{W}} = \boldsymbol{D}(\bar{\boldsymbol{m}})\, \bar{\boldsymbol{s}}\, \bar{\boldsymbol{m}}^\top = \boldsymbol{D}\big(\tfrac{\bar{\boldsymbol{x}}+\bar{\boldsymbol{z}}}{2}\big)\, (\bar{\boldsymbol{x}} - \bar{\boldsymbol{z}})\, \big(\tfrac{\bar{\boldsymbol{x}}+\bar{\boldsymbol{z}}}{2}\big)^\top.$$

D PROOF OF FORWARD LAYER FREEZING

We provide the deferred proof of the Forward Layer Freezing lemma stated in Section 4.

Proof. The proof proceeds by induction on ℓ. For $\ell = 1$, the update reads

$$\boldsymbol{m}_1^{k+1} = \sigma_1(\boldsymbol{W}_1\boldsymbol{x}_0 + \boldsymbol{b}_1),$$

which is independent of k; hence $\boldsymbol{m}_1^k = \boldsymbol{a}_1$ for all $k \geq 1$. Now assume the claim holds for layers $1, \dots, \ell - 1$. The ℓ-th block update uses only $\boldsymbol{m}_{\ell-1}^k$, which equals $\boldsymbol{a}_{\ell-1}$ for all $k \geq \ell - 1$. Thus for $k \geq \ell - 1$,

$$\boldsymbol{m}_\ell^{k+1} = \sigma_\ell(\boldsymbol{W}_\ell \boldsymbol{m}_{\ell-1}^k + \boldsymbol{b}_\ell) = \sigma_\ell(\boldsymbol{W}_\ell \boldsymbol{a}_{\ell-1} + \boldsymbol{b}_\ell) = \boldsymbol{a}_\ell,$$

and the claim follows. □

E ALGORITHMIC FORM OF THE DISCRETE DYNAMICS

We give here the explicit pseudocode for the discrete-time projection of the $(\boldsymbol{m}, \boldsymbol{s})$ dynamics referenced in Section 4.

Algorithm 2 Dynamical Backpropagation

Input: $\boldsymbol{x}_0, \boldsymbol{y}, \{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{L}$
Initialize: $\boldsymbol{m}^0$, $\boldsymbol{s}^0$
// Simultaneous evolution loop
for $k = 0$ to $2L - 1$ **do**
  $\boldsymbol{m}^{k+1} \leftarrow \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0))$
  $\boldsymbol{D}^k \leftarrow \mathrm{diag}\big(\boldsymbol{\sigma}'(\boldsymbol{W}\boldsymbol{m}^k + \boldsymbol{\beta}(\boldsymbol{x}_0))\big)$
  $\boldsymbol{s}^{k+1} \leftarrow \boldsymbol{W}^\top \boldsymbol{D}^k \boldsymbol{s}^k + \nabla_{\boldsymbol{m}_L} C(\boldsymbol{m}_L^k, \boldsymbol{y})$
end for
Gradient: $\dfrac{\partial C}{\partial \boldsymbol{W}} = \boldsymbol{D}(\boldsymbol{m}^{2L})\, \boldsymbol{s}^{2L}\, (\boldsymbol{m}^{2L})^\top$

F PROPERTIES OF THE EQUILIBRIUM

In this section we analyze some claims of the main text related to the properties of the equilibrium configuration of our relaxation dynamics.

F.1 EXPLICIT EQUIVALENCE TO BACKPROPAGATION

We derive the explicit form of the equilibrium state $\bar{\boldsymbol{s}}$ by directly inverting the linear system governing the backward phase. Recall the equilibrium condition for the stress variable from Eq. (10):

$$(\boldsymbol{I} - \boldsymbol{W}^\top \bar{\boldsymbol{D}})\, \bar{\boldsymbol{s}} = \nabla_{\boldsymbol{m}} C, \tag{39}$$

where $\nabla_{\boldsymbol{m}} C$ denotes the gradient of the loss with respect to all activations. Due to the network structure, this vector is sparse, containing non-zero entries only in the block corresponding to the output layer L.

Let $\boldsymbol{M} = \boldsymbol{W}^\top \bar{\boldsymbol{D}}$. Since $\boldsymbol{W}$ is strictly lower block-triangular (representing feedforward connectivity), its transpose $\boldsymbol{W}^\top$ is strictly upper block-triangular. The multiplication by the diagonal matrix $\bar{\boldsymbol{D}}$ preserves this structure.

Consequently, $\boldsymbol{M}$ acts as a "backward shift" operator, moving signals from layer ℓ to $\ell - 1$. Crucially, this implies that $\boldsymbol{M}$ is nilpotent with index L:

$$\boldsymbol{M}^L = \boldsymbol{0}. \tag{40}$$

Physically, this reflects the fact that error signals cannot propagate backward through more than L layers.

Because $\boldsymbol{M}$ is nilpotent, the matrix $(\boldsymbol{I} - \boldsymbol{M})$ is invertible, and its inverse is given exactly by the finite Neumann series:

$$(\boldsymbol{I} - \boldsymbol{M})^{-1} = \sum_{k=0}^{L-1} \boldsymbol{M}^k = \boldsymbol{I} + \boldsymbol{M} + \boldsymbol{M}^2 + \cdots + \boldsymbol{M}^{L-1}. \tag{41}$$

Substituting this into the equilibrium equation yields the closed-form solution:

$$\bar{\boldsymbol{s}} = \left(\sum_{k=0}^{L-1} (\boldsymbol{W}^\top \bar{\boldsymbol{D}})^k \right) \nabla_{\boldsymbol{m}} C. \tag{42}$$

Interpretation. This series expansion recovers the standard backpropagation of errors. The term $k = 0$ captures the direct loss gradient at the output. The term $k = 1$ represents the loss gradient backpropagated by one layer ($\boldsymbol{W}^\top \bar{\boldsymbol{D}} \nabla_{\boldsymbol{m}} C$), and the general term k represents the contribution of the loss backpropagated through k layers. Since $\nabla_{\boldsymbol{m}} C$ is non-zero only at layer L, the vector $\bar{\boldsymbol{s}}$ correctly aggregates the total derivative of the loss with respect to the activations at every layer ℓ.

F.2 Exponential Stability of the Equilibrium

We analyze the local stability of the coupled dynamical system by inspecting the spectrum of its Jacobian matrix at the equilibrium point $(\bar{\boldsymbol{m}}, \bar{\boldsymbol{s}})$.

Let $\boldsymbol{u} = [\boldsymbol{m}^\top, \boldsymbol{s}^\top]^\top$ denote the full state vector. The total Jacobian $\mathcal{J} \in \mathbb{R}^{2n \times 2n}$ is given by the block matrix:

$$\mathcal{J} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{m}}}{\partial \boldsymbol{m}} & \frac{\partial \dot{\boldsymbol{m}}}{\partial \boldsymbol{s}} \\ \frac{\partial \dot{\boldsymbol{s}}}{\partial \boldsymbol{m}} & \frac{\partial \dot{\boldsymbol{s}}}{\partial \boldsymbol{s}} \end{bmatrix}. \tag{43}$$

From the system definition in Eq. (9), we observe that the evolution of the primal variable $\boldsymbol{m}$ is independent of the dual variable $\boldsymbol{s}$. Consequently, the upper-right block vanishes:

$$\frac{\partial \dot{\boldsymbol{m}}}{\partial \boldsymbol{s}} = \mathbf{0}.$$

This block-triangular structure implies that the spectrum of $\mathcal{J}$ is the union of the spectra of its diagonal blocks. We analyze them individually:

1. Primal Block ($\boldsymbol{m}$). The linearization of the forward dynamics is

$$\boldsymbol{J}_{\boldsymbol{mm}} = \frac{\partial \dot{\boldsymbol{m}}}{\partial \boldsymbol{m}} = \mathrm{diag}(\boldsymbol{\sigma}'(\boldsymbol{m}))\boldsymbol{W} - \boldsymbol{I}.$$

Since $\boldsymbol{W}$ is strictly lower block-triangular (nilpotent), the product $\mathrm{diag}(\dots)\boldsymbol{W}$ retains this strictly lower triangular structure and has zeros on its main diagonal. Therefore, the eigenvalues are determined solely by the identity term:

$$\mathrm{spec}(\boldsymbol{J}_{\boldsymbol{mm}}) = \{-1\}.$$

2. Dual Block ($\boldsymbol{s}$). The linearization of the backward dynamics with respect to $\boldsymbol{s}$ is

$$\boldsymbol{J}_{\boldsymbol{ss}} = \frac{\partial \dot{\boldsymbol{s}}}{\partial \boldsymbol{s}} = \boldsymbol{W}^\top \mathrm{diag}(\boldsymbol{\sigma}'(\boldsymbol{m})) - \boldsymbol{I}.$$

Since $\boldsymbol{W}$ is strictly lower triangular, its transpose $\boldsymbol{W}^\top$ is strictly upper triangular. The product $\boldsymbol{W}^\top \mathrm{diag}(\dots)$ is thus strictly upper triangular with a zero diagonal. Similarly, the eigenvalues are determined by the identity term:

$$\mathrm{spec}(\boldsymbol{J}_{\boldsymbol{ss}}) = \{-1\}.$$

Conclusion. The Jacobian $\mathcal{J}$ has a single eigenvalue $\lambda = -1$ with algebraic multiplicity $2n$. Since $\mathrm{Re}(\lambda) < 0$, the equilibrium $(\bar{\boldsymbol{m}}, \bar{\boldsymbol{s}})$ is **globally exponentially stable**. The system relaxes with a natural time constant $\tau = 1$, ensuring rapid convergence to the backpropagation solution regardless of initialization. Finally, the $(\boldsymbol{x}, \boldsymbol{z})$ system also possesses globally exponentially stable equilibria, given that these dynamics are obtained through a linear change of variable.

G Alternative Formulation: Split-Dynamics

To facilitate implementation on hardware, we present an alternative dynamical formulation where the forward drive and Jacobian operators are evaluated individually at $\boldsymbol{x}$ and $\boldsymbol{z}$ rather than at the shared midpoint, neglecting all non-linear orders of the stress term $(x - z)$ in Eq.(8). One can easily see that only the linear term of the stress term $(x - z)$ will contribute to the gradient.

G.1 Split-Dynamics Equations

Using the same notation of the main text, neglecting all non-linear orders of $(x - z)$ in Eq.(8), we get

$$\frac{d\boldsymbol{x}}{dt} = \frac{1}{2}\Big[F(\boldsymbol{x}) + F(\boldsymbol{z})\Big] + \frac{1}{2}\left(\frac{\partial F}{\partial \boldsymbol{x}}\right)^T (\boldsymbol{x} - \boldsymbol{z}) + \frac{1}{2}\nabla_{\boldsymbol{x}} C, \tag{44}$$

$$\frac{d\boldsymbol{z}}{dt} = \frac{1}{2}\Big[F(\boldsymbol{x}) + F(\boldsymbol{z})\Big] - \frac{1}{2}\left(\frac{\partial F}{\partial \boldsymbol{z}}\right)^T (\boldsymbol{x} - \boldsymbol{z}) - \frac{1}{2}\nabla_{\boldsymbol{z}} C. \tag{45}$$

Algorithm 3 Decoupled Split-Jacobian $\boldsymbol{x}$–$\boldsymbol{z}$ Relaxation

Input: $\boldsymbol{x}_0, \boldsymbol{y}$, parameters $\{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{L}$, step size $\eta > 0$, tolerance $\varepsilon > 0$
Initialize $\boldsymbol{x}^0 = \mathbf{0}$, $\boldsymbol{z}^0 = \mathbf{0}$
for $k = 0, 1, 2, \dots$ **do**
  Define local variables:

$$\begin{aligned}
\boldsymbol{s}^k &= \boldsymbol{x}^k - \boldsymbol{z}^k, \\
\boldsymbol{\Sigma}^k &= \tfrac{1}{2}\big[\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{x}^k + \boldsymbol{\beta}) + \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{z}^k + \boldsymbol{\beta})\big], \\
\boldsymbol{D}_{\boldsymbol{x}}^k &= \operatorname{diag}\big(\boldsymbol{\sigma}'(\boldsymbol{W}\boldsymbol{x}^k + \boldsymbol{\beta})\big), \quad \boldsymbol{g}_{\boldsymbol{x}}^k = \nabla_{\boldsymbol{x}} C(\boldsymbol{x}^k, \boldsymbol{y}), \\
\boldsymbol{D}_{\boldsymbol{z}}^k &= \operatorname{diag}\big(\boldsymbol{\sigma}'(\boldsymbol{W}\boldsymbol{z}^k + \boldsymbol{\beta})\big), \quad \boldsymbol{g}_{\boldsymbol{z}}^k = \nabla_{\boldsymbol{z}} C(\boldsymbol{z}^k, \boldsymbol{y}).
\end{aligned}$$

  Compute velocity fields (fully decoupled dynamics):

$$\begin{aligned}
\dot{\boldsymbol{x}}^k &= \boldsymbol{\Sigma}^k - \boldsymbol{x}^k + \tfrac{1}{2}\boldsymbol{W}^\top \boldsymbol{D}_{\boldsymbol{x}}^k \boldsymbol{s}^k + \tfrac{1}{2}\boldsymbol{g}_{\boldsymbol{x}}^k, \\
\dot{\boldsymbol{z}}^k &= \boldsymbol{\Sigma}^k - \boldsymbol{z}^k - \tfrac{1}{2}\boldsymbol{W}^\top \boldsymbol{D}_{\boldsymbol{z}}^k \boldsymbol{s}^k - \tfrac{1}{2}\boldsymbol{g}_{\boldsymbol{z}}^k
\end{aligned}$$

  Update states:

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \eta\, \dot{\boldsymbol{x}}^k, \qquad \boldsymbol{z}^{k+1} \leftarrow \boldsymbol{z}^k + \eta\, \dot{\boldsymbol{z}}^k$$

  if $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|_2 + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|_2 < \varepsilon$ **then**
    break
  end if
end for
Equilibrium variables:

$$\bar{\boldsymbol{m}} = \tfrac{1}{2}(\boldsymbol{x}^k + \boldsymbol{z}^k), \quad \bar{\boldsymbol{s}} = \boldsymbol{x}^k - \boldsymbol{z}^k.$$

Gradient computation:

$$\frac{dC}{d\boldsymbol{W}} = \boldsymbol{D}(\bar{\boldsymbol{m}})\, \bar{\boldsymbol{s}}\, \bar{\boldsymbol{m}}^\top.$$

Here, $\nabla_{\boldsymbol{u}} C$ denotes the gradient of the loss injected at the output block of the state $\boldsymbol{u}$.

This formulation avoids the computation of the midpoint state $\boldsymbol{m}$ for derivative evaluations.

G.2 Split-Jacobian Relaxation Algorithm

The corresponding discrete-time implementation of the Split-Dynamics is provided in Algorithm 3. This reformulation of the algorithm, tested on the same tasks, matches the performance of algorithm 1 as predicted from the theory.

H Additional Experimental Results

H.1 Block Grouping and Inertial Relaxation Protocols

Block grouping (E6). We use five fixed CIFAR-10 batches of size 128, one per seed, and a reduced CNN with five convolution–ReLU stages, a pooled hidden linear–ReLU stage, and a classifier. Operation-level states are placed after individual operations; layer-level states follow the seven logical stages; block-level states group pairs of hidden stages, retaining the classifier separately. Matched partitions share parameters and have identical forward-output hashes. Means and stresses start at zero, weights remain fixed, and updates use simultaneous local VJPs in float32. The stopping statistic is the relative change of the full reconstructed state $\boldsymbol{q}_{xz} = (\boldsymbol{m} + \boldsymbol{s}/2, \boldsymbol{m} - \boldsymbol{s}/2)$, with tolerance 10^{-5}; the main text reports $\eta = 1$. Reference gradients use autodiff of the same network function. These controls are separate from the CNN9 training runs.

Discrete and continuous inertia (E12–E13). We test $\boldsymbol{h}^\star = \boldsymbol{W}\boldsymbol{h}^\star + \boldsymbol{B}\boldsymbol{u}$ with symmetric $\boldsymbol{W}$, spectral radius $\rho \in \{0.5, 0.75, 0.9, 0.95, 0.99\}$, state dimension 16, input dimension 8, and batch size 32. The objective is half the mean squared error to a fixed target. For one sample, the residuals used

in Eq. (22) are

$$\begin{aligned} \boldsymbol{r}_m(\boldsymbol{m}, \boldsymbol{s}) &= \boldsymbol{W}\boldsymbol{m} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{m}, \\ \boldsymbol{r}_s(\boldsymbol{m}, \boldsymbol{s}) &= (\boldsymbol{W}^\top - \boldsymbol{I})\boldsymbol{s} + \nabla_{\boldsymbol{m}} C(\boldsymbol{m}, \boldsymbol{y}), \end{aligned} \tag{46}$$

where the loss gradient includes the batch and state-dimension averaging of the mean squared error. The equilibrium gradients are $\nabla_{\boldsymbol{W}} C = \sum_i \boldsymbol{s}_i^\star (\boldsymbol{m}_i^\star)^\top$ and $\nabla_{\boldsymbol{B}} C = \sum_i \boldsymbol{s}_i^\star \boldsymbol{u}_i^\top$, summed over samples; the averaging is already included in $\boldsymbol{s}_i^\star$. Means, sensitivities, and stored updates start at zero. For $a = 1 - \rho$ and $b = 1 + \rho$, the momentum settings are $\alpha = 4/(\sqrt{b} + \sqrt{a})^2$ and $\beta = ((\sqrt{b} - \sqrt{a})/(\sqrt{b} + \sqrt{a}))^2$; the overdamped baseline uses $\alpha = 1, \beta = 0$. Gradients are compared with a CPU-float64 implicit reference at relative error 10^{-3} for five float32 seeds and 10^{-10} for one float64 control per radius. At $\rho = 0.99$, the latter requires $2625/223$ overdamped/inertial updates. For the inertial runs, choosing $\Delta t = 1$ identifies the corresponding mass and damping in Eq. (21) as $M = 1/\alpha$ and $\gamma = (1 - \beta)/\alpha$. These are standard momentum settings applied to the internal state updates.

For the continuous control, let $\boldsymbol{H} = (\boldsymbol{I} - \boldsymbol{W})/\lambda_{\max}(\boldsymbol{I} - \boldsymbol{W})$. Let $\widetilde{\boldsymbol{r}} = \boldsymbol{r}/\lambda_{\max}(\boldsymbol{I} - \boldsymbol{W})$, where $\boldsymbol{r}$ is given by Eq. (46). For our matrices, whose eigenvalues are $\pm\rho$, the denominator is $1 + \rho$. We compare

$$\dot{\boldsymbol{q}} = \widetilde{\boldsymbol{r}}(\boldsymbol{q}) \quad \text{and} \quad \ddot{\boldsymbol{q}} + \gamma \dot{\boldsymbol{q}} = \widetilde{\boldsymbol{r}}(\boldsymbol{q}), \qquad \gamma = 2\sqrt{\lambda_{\min}(\boldsymbol{H})}. \tag{47}$$

The acceleration term lets the state retain velocity, while γ damps that velocity. Both equations have the same equilibrium. We propagate the coupled state and sensitivities exactly in the eigenvector basis of $\boldsymbol{H}$. Settling is measured after the last time the gradient error exceeds its target during the simulated interval; the time grid is 0.05 with bisection refinement. The normalization fixes the fastest decay rate without inertia to one. At $\rho = 0.99$, the float64 settling times are $5247.74/447.25$. These are normalized model times, not hardware measurements.

Nonlinear stability controls (E14). The fixed point is $\boldsymbol{h}^\star = \tanh(\boldsymbol{W}\boldsymbol{h}^\star + \boldsymbol{B}\boldsymbol{u})$ with $\|\boldsymbol{W}\|_2 = \rho < 1$. We test symmetric matrices and non-normal matrices ($\boldsymbol{W}\boldsymbol{W}^\top \neq \boldsymbol{W}^\top\boldsymbol{W}$), $\rho \in \{0.5, 0.75, 0.9, 0.95\}$, drive scales $0.5/1.5$, five float32 seeds, and one float64 control per configuration. Fourth-order Runge–Kutta (RK4) integration uses $dt = 0.1/0.05$ in float32/float64 and implicit-gradient targets $10^{-3}/10^{-9}$. We continue integrating after the error reaches its target to check that it remains below it. Of 192 original runs, 191 settle. For the symmetric float32 cases, median speedups across seeds at each parameter setting are 1.21–2.36×. The unsuccessful non-normal run also fails when the time step is halved. To choose damping before finding the equilibrium, we use the eigenvalues $a_i \in \sigma((\boldsymbol{I} - \boldsymbol{W})/(1 + \rho))$, the linear stability condition is $\gamma^2 \operatorname{Re} a_i > (\operatorname{Im} a_i)^2$. The damping rule chosen after the initial experiments is

$$\gamma = \max\left\{ 2\sqrt{\min_i \operatorname{Re} a_i},\ 1.05 \max_i \frac{|\operatorname{Im} a_i|}{\sqrt{\operatorname{Re} a_i}} \right\}.$$

All 48 added non-normal controls settle; at $\rho \geq 0.9$, inertia wins all 20 float32 pairs against their original overdamped baselines. This rule is empirical for the nonlinear system, not a general nonlinear stability guarantee. The added controls use PyTorch 2.7.1, Apple MPS for float32, and CPU float64 references and controls.

The following CNN9 analyses compare Algorithm 1 with standard backpropagation on CIFAR-10.

H.2 Training Stability and Loss Trajectory

To validate that the relaxation dynamics do not introduce optimization artifacts, we tracked the training loss across different Euler step sizes η. As illustrated in Figure 3, the learning trajectories for $\eta \in \{0.25, 0.5, 0.75, 1.0\}$ indistinguishably overlap with the ideal case $\eta = 1$. This further confirms that the our formulation effectively solves the credit assignment problem without degrading optimization stability.

H.3 Gradient Norm Consistency

A critical requirement for replacing symbolic differentiation with physical relaxation is the preservation of gradient magnitude. We measured the norm ratio $\|\nabla_{\text{relax}}\|/\|\nabla_{\text{BP}}\|$ throughout training.

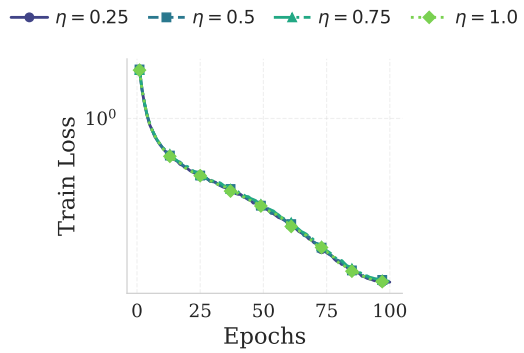

Figure 3: **Training Loss Consistency.** The decay of training loss for the relaxation algorithm matches exactly across all step sizes. Axes indicate Epochs (0–100) versus Loss (log scale).

Figure 4 demonstrates that the proposed method maintains near-perfect scale consistency. The deviation from unity is negligible, with fluctuations bounded within a minute range of 0.0001 to -0.0003. This confirms that the stress variable $\boldsymbol{s} = \boldsymbol{x} - \boldsymbol{z}$ correctly accumulates the total sensitivity of the loss with respect to activations.

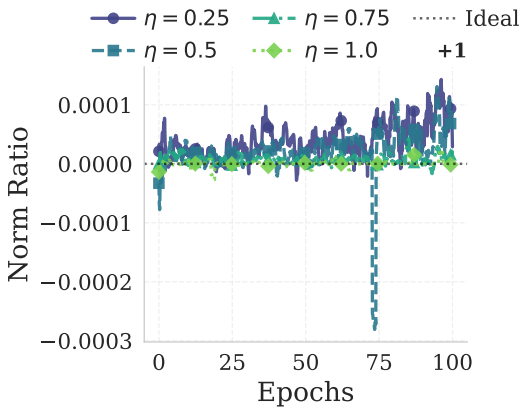

Figure 4: **Gradient Norm Fidelity.** The ratio between the norms of relaxation gradients and backpropagation gradients remains centered at 1.0, with deviations smaller than 10^{-4}. Legend indicates step sizes $\eta = 0.25$ through $\eta = 1.0$.

H.4 Directional Alignment

We can also observe that the directional accuracy of the gradient update is paramount. We present here a global quantification of this using the global cosine misalignment metric, defined as $1 - \cos(\theta)$, where θ is the angle between ∇_{relax} and ∇_{BP} as measured in the main text.

Figure 5 reveals that the misalignment is practically zero. The metric consistently falls below 10^{-5} oscillating around 10^{-7}, confirming to be almost indistinguishable from standard backpropagation (the accuracy used in the code is IEEE 754 Float32single point precision corresponding to a decimal precision $\approx 10^{-7}$).

H.5 Signal-to-Noise Ratio (SNR)

Finally, we assessed the quality of the gradient estimator by computing the Signal-to-Noise Ratio (SNR), defined as $\|\nabla_{\text{BP}}\|^2 / \|\nabla_{\text{relax}} - \nabla_{\text{BP}}\|^2$.

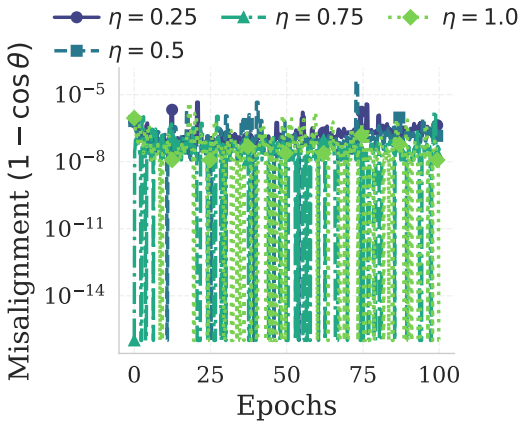

Figure 5: **Directional Alignment.** The cosine misalignment $(1 - \cos\theta)$ between the relaxation algorithm and standard BP.

As shown in Figure 6, the SNR remains exceptionally high throughout the training process, consistently exceeding 10^6 (60 dB) and peaking near 10^9. We observe a non-linear relationship where larger step sizes (e.g., $\eta = 1.0$) yield much higher SNR.

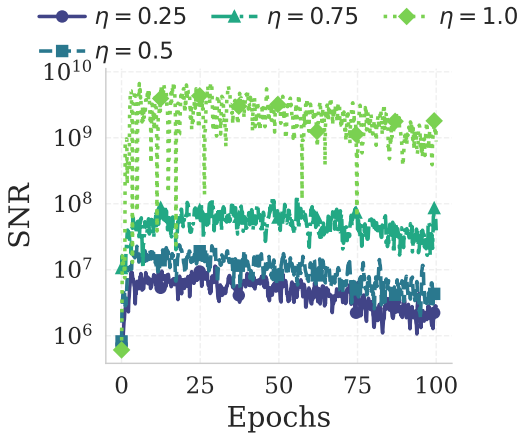

Figure 6: **Gradient Signal-to-Noise Ratio.** The SNR consistently exceeds 10^6, confirming that the relaxation error is negligible compared to the gradient signal.

H.6 Layer-wise Misalignment and Time Step

In our experimental analysis, we observed that while the overall gradient accuracy remains exceptionally high, a subtle degradation in alignment emerges as signals propagate backward to earlier layers (e.g., Layer 1) over the course of training. This phenomenon is visualized in Figure 7, which displays the log-misalignment for a step size of $\eta = 0.25$. This slight drift is attributable to the propagation of numerical errors inherent to the finite-time step approximation of the continuous dynamics.

Conversely, using the theoretically optimal step size $\eta = 1.0$ (Figure 8) recovers perfect alignment across all layers. We stress that digital implementation is not our goal; these experiments serve simply to demonstrate consistency and experimentally validate the theory.

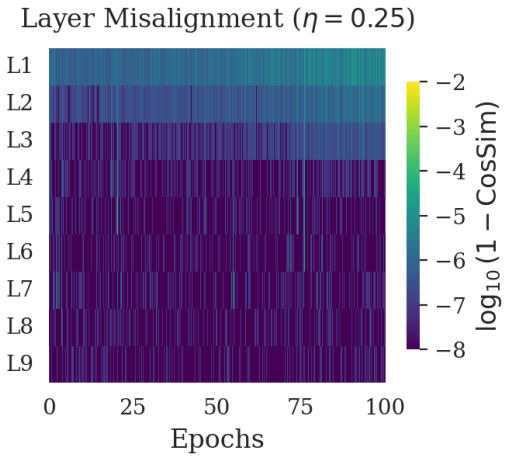

Figure 7: **Log-misalignment heatmap ($\eta = 0.25$).** The plot displays $\log_{10}(1 - \text{CosSim})$ for each layer. A minor accumulation of numerical error is visible in earlier layers, consistent with finite-step approximations.

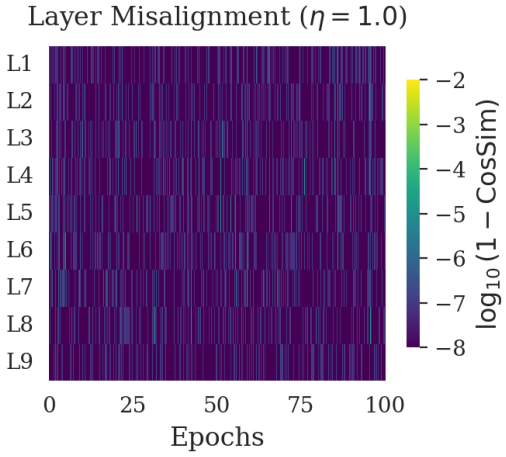

Figure 8: **Log-misalignment heatmap ($\eta = 1.0$).** The plot displays $\log_{10}(1 - \text{CosSim})$ for each layer. The alignment is effectively perfect (bounded only by machine precision), confirming that the unit-step discretization recovers the exact gradient.

I Comparison with Related Methods

Table 1 summarizes the gradient guarantees, assumptions, and update structures discussed in Section 6.

Table 1: Gradient guarantees of related methods. Each guarantee applies to the method's stated objective and assumptions.

Method	Gradient statement	Dynamics / phases	Coupling and scope
Standard EP (Scellier & Bengio, 2017)	Exact energy-model gradient as nudging tends to zero	Free and nudged equilibria	Reciprocal energy-based interactions
Holomorphic EP (Laborieux & Zenke, 2022)	Exact gradient from finite oscillations	Continuous oscillations	Complex-analytic energy model
Exact predictive coding (Salvatori et al., 2022)	Exact BP with specified initialization and update schedule	Local inference with prescribed timing	Differentiable computational graphs
Dual propagation (Høier et al., 2023)	Second-order approximation in the perturbation	One phase with two states per neuron	Feedforward networks; weight transport studied separately
E-prop (Bellec et al., 2020)	Online approximation to BP through time (BPTT)	Eligibility traces and learning signals	Recurrent spiking networks; feedback variants
Our two-state dynamics	Exact equilibrium BP; recovery by $2L$ for unit-step Euler	Two states evolve together	Feedforward stages; exact local VJPs and weight transport