Towards a Modern Theory of Time Parallelization for Nonlinear ODEs

Zhouyang Lu[1] **Arvind R. Venkatakrishnan**[2] **Emmie Fitz-Gibbon**[1]
Xavier Gonzalez[3] **Yu Kawano**[4] **Ian R. Manchester**[5] **Eduardo D. Sontag**[6]
Jean-Jacques Slotine[7] **Francesco Bullo**[2] **Leo Kozachkov**[1]

[1]*Brown University* [2]*University of California, Santa Barbara*
[3]*Unconventional AI* [4]*Hiroshima University* [5]*The University of Sydney*
[6]*Northeastern University* [7]*MIT*

zhouyang_lu@brown.edu leokoz8@brown.edu

Abstract

Nonlinear ordinary differential equations (ODEs) are fundamental to science and engineering, but standard numerical solvers are inherently sequential and therefore poorly suited to parallel hardware such as GPUs. Parallel-in-time (PiT) methods offer a promising approach to overcome this limitation, but a firm theoretical understanding of when they can be efficiently applied to nonlinear ODEs is lacking. This work provides such a theory, by connecting intrinsic properties of the ODE, such as stability, to the conditioning of the associated parallel-in-time optimization problem, as measured by a Polyak–Łojasiewicz constant. We use our theory to provide several practical strategies for improving the conditioning of the PiT problem, and we validate our theory and these strategies on several real-world examples, including contraction-constrained imitation learning for robot motion, where we achieve a $\sim 35\times$ speedup in total training time. We provide code to reproduce our experiments and formal Lean proofs of our two main bounds here:

github.com/dynamic-intelligence-lab/modern_parallel_ode

Keywords: parallel-in-time integration, nonlinear ODEs, contraction analysis, Polyak–Łojasiewicz inequality, GPU acceleration

1. Introduction and Background

Parallel hardware is becoming cheaper, more accessible, and increasingly powerful (Juniewicz and You, 2026). This improvement has helped drive the dramatic progress of artificial intelligence, whose neural network architectures rely heavily on matrix multiplications that graphics processing units (GPUs) execute efficiently. However, many important computational tasks cannot be naively parallelized in the same way matrix multiplication can, by virtue of being "inherently sequential" (i.e., causal). A prominent example is the numerical integration of nonlinear ordinary differential equations (ODEs). Traditional time-stepping methods for ODEs, such as Euler's method or Runge–Kutta, proceed one step at a time, limiting their ability to exploit massively parallel hardware.

However, recent work on parallel-in-time (PiT) methods has shown that certain nonlinear dynamical systems[1] can be parallelized across time efficiently on GPUs. These techniques reformulate numerical integration as a gigantic optimization problem, which can then be solved in parallel. Prominent approaches include DEER (Lim et al., 2024; Danieli et al., 2023), which uses Newton's method as the solver, and ELK (Gonzalez et al., 2024), which uses Levenberg–Marquardt. Moreover, Gonzalez et al. (2025) show that a key factor in whether or not a system can be parallelized in time is

1. Sometimes referred to as nonlinear state-space models (SSMs).

its underlying *stability*. Although this discrete-time result applies to discretized ODEs, we seek a theory of parallelization that applies to the continuous-time dynamics themselves, before discretization. This is important because discretization introduces a tradeoff between numerical accuracy and parallel efficiency, and different discretizations can yield different stability properties. In particular, the choice of discretization scheme and step size can affect whether a system is stable or unstable, and thus whether it can be efficiently parallelized. Appendix A discusses related work in detail.

In this work, we develop a continuous-time theory that directly connects the stability of nonlinear ODEs to the conditioning of the optimization problems used to solve them in parallel. These problems are generally nonconvex, but we show that they satisfy a Polyak–Łojasiewicz (PL) inequality, a cornerstone of modern nonconvex optimization theory (Karimi et al., 2016). The PL constant measures the conditioning of these optimization problems and, in turn, the difficulty of solving them using parallel-in-time approaches. We also develop a common framework for updating candidate trajectories, which recovers Picard, Jacobi, and Newton-type methods as special cases. This framework gives convergence guarantees for Jacobi under contraction and suggests how to stabilize parallel updates when the current trajectory guess is unstable. One such update, LASER (Section 2.2), is a damped, adaptive Newton-style method whose damping is set at every iteration from the finite-time Lyapunov exponent of the current guess. Finally, we demonstrate substantial speedups in applications such as robotic imitation learning, where parallelization accelerates the entire training pipeline by $\sim 35\times$. On a bistable Lorenz system solved from an unstable initial guess, the undamped Newton update diverges, whereas LASER converges without tuning and is the fastest solver at every time horizon tested.

Initial value problems. We are interested in solving the initial value problem (IVP)

$$\dot{x} = f(x,t), \qquad x(0) = x_0 \in \mathbb{R}^D, \qquad t \in [0,T], \tag{1}$$

with final time $T > 0$. In most cases, (1) cannot be solved analytically and must instead be numerically integrated. Numerical integration methods typically start at the initial state x_0 and advance the solution a short interval at a time.

A different approach is to reformulate (1) as an optimization problem, and then seek parallelism while iteratively solving that problem. Informally, we may *guess* a candidate trajectory $s(t)$ satisfying $s(0) = x_0$, measure how much it violates the ODE at each time by its *residual* $r(t) \coloneqq \dot{s}(t) - f(s(t),t)$, and evaluate the quality of this guess using the merit functional

$$\mathcal{L}[s(t)] \;=\; \frac{1}{2}\int_0^T \|r(t)\|_2^2\, dt \;=\; \frac{1}{2}\int_0^T \|\dot{s}(t) - f(s(t),t)\|_2^2\, dt. \tag{2}$$

Throughout, we will assume that f and the initial trajectory guess are sufficiently "nice" so that $\mathcal{L}$ exists and is finite. Notice that if $\mathcal{L}[s(t)] = 0$, then $s(t)$ satisfies the IVP (1) exactly. Consequently, under standard existence and uniqueness assumptions on the IVP, the global minimizer $s^*(t)$ coincides with the solution of the IVP. For discrete-time IVPs, this strategy has recently been shown to produce significant speedups on GPUs (Lim et al., 2024; Gonzalez et al., 2025; Zoltowski et al., 2025; Danieli et al., 2026), because each optimization step can be parallelized across time (Section 2.2). Our aim in this paper is to extend the theory behind this approach to continuous-time systems. Note that because the IVP is assumed to be nonlinear, the merit functional (2) is nonconvex in $s(t)$. However, as we will see, it satisfies a Polyak–Łojasiewicz inequality,[2] which will give us a means of assessing how ill-conditioned the merit functional is and therefore how "easy" it is to optimize.

2. Sometimes referred to as gradient dominance (e.g., Fazel et al., 2018).

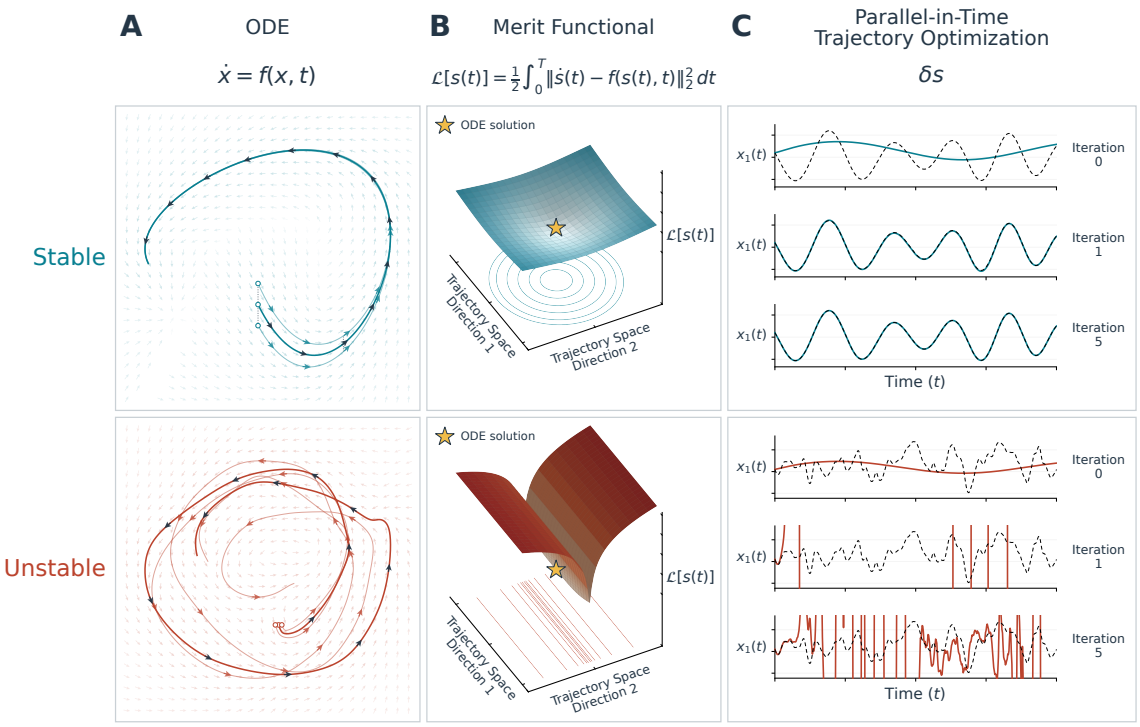


Figure 1: Stable ODEs induce well-conditioned optimization problems that can be solved in parallel, yielding substantial evaluation speedups on modern hardware. Unstable ODEs induce ill-conditioned problems that are difficult to solve efficiently. **A**: state-space trajectories over time. Top row shows incremental stability (i.e., contraction); bottom row shows incremental instability (i.e., chaos). **B**: contours of the merit functional on a two-dimensional slice spanned by the smallest and largest singular vectors at the solution, showing ill-conditioning of the unstable case. **C**: ODE evaluation via Gauss–Newton iterations.

Polyak–Łojasiewicz inequalities. Briefly, a functional $\mathcal{L}$ is said to be μ-PL if there exists some constant $\mu > 0$ such that, for every candidate trajectory $s(t)$,

$$\frac{1}{2}\|\nabla_{s(t)}\mathcal{L}\|^2 \geq \mu\big(\mathcal{L} - \mathcal{L}^*\big) \qquad \text{where} \qquad \mathcal{L}^* := \inf_{s(t)} \mathcal{L}[s(t)]. \tag{3}$$

Intuitively, the PL condition ensures that the gradient vanishes if and only if the iterate is at a global minimizer. For gradient flows, the PL condition is exactly equivalent to imposing exponential convergence of the residual cost (Wensing and Slotine, 2020; Kozachkov et al., 2023). Since the PL inequality may hold for a range of constants, we write $\mu_\star$ for the optimal PL constant, i.e., the supremum over all $\mu \geq 0$ for which the PL inequality holds. Holding everything else fixed, a larger $\mu_\star$ corresponds to an easier optimization problem. For more details on the finite-dimensional PL inequality, we refer readers to Karimi et al. (2016) or Section 3.1 of Gonzalez et al. (2025). We note that the merit functional (2) is by no means unique: one could also consider the L^1 norm of the residual instead of the L^2 norm, or one could consider a weighted L^2 norm. The PL lower bounds we will derive extend to other differentiable residual penalties that themselves satisfy a PL inequality. We elaborate on these two points in Appendix C.

Incremental stability. To estimate the PL constant $\mu_\star$, we need the gradient of the merit functional (2). This in turn requires understanding how small perturbations to a candidate trajectory $s(t)$ affect its violation of the ODE. This sensitivity is characterized by the variational dynamics associated with the IVP along a trajectory $x(t)$

$$\delta\dot{x} = J(x(t),t)\,\delta x, \qquad J(x(t),t) := \frac{\partial f}{\partial x}(x(t),t), \tag{4}$$

which describe the evolution of infinitesimal perturbations and thereby characterize the stability of nearby trajectories. If we fix an initial time τ, and consider how a perturbation $\delta x(\tau)$ evolves along a candidate trajectory $s(t)$ to a later time t, we find that this evolution is linear and is described by the state-transition matrix $\Phi_s(t,\tau)$. The subscript s denotes that the state-transition matrix is computed along a particular trajectory $s(t)$. A direct calculation[3] shows that $\Phi_s(t,\tau)$ satisfies the linear matrix differential equation

$$\frac{d}{dt}\Phi_s(t,\tau) = J(s(t),t)\,\Phi_s(t,\tau), \qquad \Phi_s(\tau,\tau) = I. \tag{5}$$

Thus, $\Phi_s(t,\tau)$ encodes the cumulative effect of the Jacobian along the trajectory $s(\cdot)$, and thereby determines the growth or decay of perturbations.

2. Results

As noted in the introduction, a larger PL constant $\mu_\star$ corresponds to a better-conditioned optimization problem. Our goal is therefore twofold: (i) to establish positive conditions under which $\mu_\star$ remains bounded away from zero, characterizing regimes where optimization is tractable; and (ii) to identify negative conditions under which $\mu_\star$ necessarily approaches zero, revealing fundamental limitations on optimization efficiency.

2.1. Stability and the PL Constant

We consider two major types of behavior for the variational dynamics (4), which we refer to as CASE 1 and CASE 2. In CASE 1, perturbations decay exponentially along every admissible trajectory, which is a strong form of incremental stability. This can be seen in Figure 1, top row. In CASE 2, perturbations grow exponentially along some admissible trajectory, which is a strong form of incremental instability, typical of chaotic systems. This can be seen in Figure 1, bottom row.

CASE 1: INCREMENTALLY STABLE SYSTEMS

In this case, suppose there exists an overshoot constant $a \geq 1$ and a contraction rate $c > 0$ such that, for every admissible trajectory $s(\cdot)$ and all $0 \leq \tau \leq t \leq T$, the state-transition matrix (5) satisfies

$$\|\Phi_s(t,\tau)\| \leq a e^{-c(t-\tau)}. \tag{6}$$

This condition resembles the standard notion of contraction in nonlinear control theory (Lohmiller and Slotine, 1998; Bullo, 2026). The subtlety here is that the exponential decay estimate must hold uniformly along *every candidate trajectory* $s(\cdot)$ encountered during optimization, including

3. Fix τ and define $z(t) := \Phi_s(t,\tau)\delta x(\tau)$. Then $\dot{z}(t) = J(s(t),t)z(t)$ and $z(\tau) = \delta x(\tau)$. Since $\delta x(t)$ satisfies the same linear IVP, uniqueness implies $\delta x(t) = \Phi_s(t,\tau)\delta x(\tau)$.

trajectories that do not satisfy the ODE. It is therefore a much stronger requirement than imposing the same stability condition only along solutions of the IVP. This uniformity is needed because the PL inequality is a global statement about the optimization landscape and therefore requires a bound on perturbation growth along all candidate trajectories. We return to this requirement in Sections 2.2 and 3.

Theorem 1 (Incremental stability implies a PL inequality) *Under the decay estimate (6), the merit functional (2) satisfies the Polyak–Łojasiewicz inequality (3) with $\mathcal{L}^* = 0$, and its optimal PL constant obeys*

$$\sqrt{\mu_\star} \;\geq\; \frac{c}{a} \cdot \frac{1}{1 - e^{-cT}} \;\geq\; \frac{c}{a} \;>\; 0. \tag{7}$$

Proof The proof is given in Appendix B.1. ■

The key consequence of Theorem 1 is that the PL constant remains bounded away from zero independently of the time horizon T. Thus, extending the trajectory does not weaken the PL guarantee below c^2/a^2. The bound also separates the roles of the two stability constants: faster contraction strengthens the guarantee, whereas larger transient amplification, represented by the overshoot a, weakens it.

CASE 2: EXPONENTIALLY UNSTABLE SYSTEMS

We now consider the case of exponential growth along the solution. Here, an upper bound on perturbation growth is not enough: we need a lower bound that guarantees that some perturbations actually grow. Unlike the stable case, this condition only needs to hold along the exact solution of the IVP. Let $x(\cdot)$ solve (1). Suppose there are constants $a \geq 1$, $0 < b \leq 1$, and a growth rate $\lambda > 0$ such that, for all $0 \leq \tau \leq t \leq T$,

$$b e^{\lambda(t-\tau)} \;\leq\; \|\Phi_x(t,\tau)\| \;\leq\; a e^{\lambda(t-\tau)}. \tag{8}$$

The lower bound is on the operator norm of Φ_x, so it only requires growth in some direction; other directions may contract. The upper bound controls how unevenly this growth can occur over time. Together, the two bounds allow us to identify a perturbation that grows throughout the interval, which is the key ingredient in the proof. There is also an important difference in the scope of the two cases. To lower bound the global PL constant, we needed control along every candidate trajectory. To upper bound it, it is enough to find trajectories near the solution for which the gradient is small relative to the loss. This is why (8) only concerns $x(\cdot)$, as opposed to all possible candidate trajectories $s(t)$.

Theorem 2 (Instability implies a vanishing PL constant) *Suppose (8) holds. Then, for $\lambda T \geq 2$, the optimal PL constant of the merit functional (2) obeys*

$$\sqrt{\mu_\star} \;\leq\; \frac{e\lambda a^2}{b} e^{-\lambda T}. \tag{9}$$

Note that, for fixed a, b, and λ, the PL constant $\mu_\star$ goes to zero as $T \to \infty$.

Proof The proof is given in Appendix B.2. ■

The interpretation of this result is that, if a system is exponentially unstable, then the PL constant can be made arbitrarily small by taking a long enough time horizon. From the optimization point of view, this means that the problem becomes exponentially ill-conditioned and therefore increasingly difficult to solve efficiently as the time horizon T grows.

2.2. Modifying the Optimization Step

So far we have focused on the merit functional and its PL constant, which roughly controls the difficulty of the optimization landscape. However, we have not discussed the actual optimization algorithm itself. We now consider how the trajectory guess is updated at each iteration. The aim is to control how errors propagate through the update while keeping the original IVP as its fixed-point condition. This allows us to study updates without requiring them to minimize a particular merit functional. A general approach is to consider the instantaneous loss $\ell(t) \coloneqq \frac{1}{2}\|r(t)\|^2$ of the residual $r(t) = \dot{s}(t) - f(s(t), t)$ at some time t. A small perturbation (formally, a variation) of the trajectory $s(t)$ will change the residual $r(t)$, and therefore change the instantaneous loss. Going forward, we will drop the explicit dependence on time t whenever it is clear from context. The linearized change in the loss is

$$\delta\ell \;=\; r^\top \delta r \;=\; r^\top \big(\delta\dot{s} - J(s,t)\,\delta s\big). \tag{10}$$

We can choose a perturbation δs that reduces the instantaneous loss. A natural choice is

$$\delta\dot{s} = J(s,t)\delta s - r(t), \quad \implies \quad \delta\ell = -r^\top r \;\le\; 0.$$

The actual update to $s(t)$ at optimization step i is then given by

$$s^{(i+1)}(t) = s^{(i)}(t) + \delta s, \quad \text{where} \quad \delta\dot{s} = J(s^{(i)}, t)\,\delta s - r^{(i)}(t). \tag{11}$$

This turns out to correspond to a Newton correction, the continuous-time counterpart of DEER (Lim et al., 2024). Unlike (1), this ODE is *linear* in δs, and this is the source of parallelism. On a grid of N time steps it becomes a linear recurrence $\delta s_{n+1} = \Gamma_n\,\delta s_n + c_n$, whose coefficients depend only on the current guess and are computed for all n at once. Because composing affine maps is associative, a parallel scan (Blelloch, 1990) then solves the recurrence in $O(\log N)$ time (Appendix J). Of course, this is not the only update that reduces the instantaneous loss. We can also consider a more general linearized update of the form

$$\delta\dot{s} = A(s,\delta s,t)\delta s - B(s,\delta s,t)r(t), \;\; \implies \;\; \delta\ell = r^\top\big(A(s,\delta s,t) - J(s,t)\big)\delta s - r^\top B(s,\delta s,t)r,$$

where $A(s,\delta s,t)$ and $B(s,\delta s,t)$ are arbitrary matrices. Thus, as long as the symmetric part of $B(s,\delta s,t)$ is positive definite and $\|A(s,\delta s,t) - J(s,t)\|$ is sufficiently small, the instantaneous loss will be reduced. When A and B do not depend on δs, the update is still a linear ODE in δs, and it parallelizes in the same way as the Newton update. Of course, different choices of instantaneous loss functions are possible, which lead to different linearized updates.

Damped Newton and LASER. One particularly simple choice is to pick $A(s,\delta s,t) = J(s,t) - \alpha I$ and $B = I$, which damps the Newton correction in case the system, or the current guess, is unstable:

$$\delta\dot{s} = \big(\; -\alpha I + J(s,t)\;\big)\,\delta s - r(t).$$

The transition matrix of this update is $e^{-\alpha(t-\tau)}\Phi_s(t,\tau)$. Note that this update cannot be achieved via an exponential weighting of the merit functional, because that also requires modifying the residual $r(t)$. Weighting the residual by $e^{-2\alpha t}$ gives a horizon-independent PL bound once α exceeds the growth rate (Theorem 4), but it does not change the Newton correction (Appendix C.2). With a constant α, the discrete analogue scales the Jacobians that propagate corrections, as in scale-ELK (Gonzalez et al., 2024). We instead set α adaptively at every optimization step, using the finite-time Lyapunov exponent of the current guess,

$$\lambda(s) = \frac{1}{T}\log\|\Phi_s(T,0)\|, \qquad \alpha = \max\{0,\;\lambda(s) + \epsilon\}, \qquad \epsilon > 0, \tag{12}$$

so that the damped transition matrix over the whole horizon has norm at most $e^{-\epsilon T}$. We call our novel method LASER (**L**yapunov-**A**ware **S**cale **E**LK **R**egularization; Appendix E). To avoid a sequential bottleneck, we rely on a recently introduced PiT algorithm for computing the finite-time largest Lyapunov exponent (Heinsen and Kozachkov, 2025).

Picard and Jacobi. We recover two familiar methods by taking $B = I$. Picking $A = 0$, with zero correction at $t = 0$, gives standard Picard iteration. Jacobi iteration needs an A that also depends on the correction, $A(s, \delta s, t) = \int_0^1 J(s + \theta\delta s, t)\, d\theta$. The update is then no longer linear in δs, and it is parallelized across time intervals instead: solving the correction independently on each interval, with zero correction at its left endpoint, gives exact-flow Jacobi iteration. Under CASE 1, the latter converges geometrically. Both derivations and the Jacobi convergence proof are given in Appendix D. Jacobi also converges geometrically under contraction on average, even when individual time steps are expansive (Appendix L).

Remark 1 (Choosing the step size for a fixed iteration budget) *Let $\phi(t; x_0)$ be the solution of (1). On a uniform grid with step size $h = T/K$, let $\hat{s}$ be the piecewise-linear interpolant of the exact discrete solution. Furthermore, let $\hat{p}_M$ be the piecewise-linear interpolant after M parallel iterations. Integrating the pointwise triangle inequality gives*

$$E(M, h; T, x_0) := \int_0^T \|\hat{p}_M(t) - \phi(t; x_0)\|\, dt \;\le\; \underbrace{\|\hat{p}_M - \hat{s}\|_{L^1}}_{E_M^{\mathrm{P}}\ \textit{(parallelization)}} + \underbrace{\|\hat{s} - \phi\|_{L^1}}_{E^{\mathrm{S}}\ \textit{(discretization)}}. \tag{13}$$

Refining the grid typically reduces E^{S} but can increase E_M^{P}. For example, with Jacobi, M sweeps recover the exact discrete trajectory through time $\min\{Mh, T\}$, so a smaller h can shorten the interval of guaranteed agreement. Appendix F derives an upper bound on $E_M^{\mathrm{P}} + E^{\mathrm{S}}$ for Jacobi and shows how to choose h by minimizing this bound for fixed M (Figure 4).

3. Experiments

Our experiments demonstrate the speedup of PiT solvers, and test their reliability when the bound (6) is not guaranteed along every candidate trajectory. In Section 3.1, we train a contracting model from demonstrations on long rollouts, where PiT solvers are $\sim 35\times$ faster than a sequential solver. In Section 3.2, we solve a bistable Lorenz system from an unstable initial guess, where the undamped Newton update diverges. In both, the adaptively damped update LASER (Section 2.2 and Appendix E) is the most reliable solver.

3.1. Stable Imitation Learning

There has been much work in machine learning and robotics on enforcing stability of learned dynamical models (Khansari-Zadeh and Billard, 2011; Ijspeert et al., 2013; Kozachkov et al., 2020; Revay and Manchester, 2020; Jafarpour et al., 2021; Kozachkov et al., 2022; Jaffe et al., 2024; Beik-Mohammadi et al., 2024). One major reason for this interest is that stable systems can be robust and safe in the presence of bounded disturbances. Many of the stability conditions for these models are stated in continuous time, making them an ideal target for applying our theory. In particular, Jaffe et al. (2024) and Beik-Mohammadi et al. (2024) recently developed a continuous-time framework for learning contracting dynamics from demonstrations through latent dynamics of the form

$$\dot{z} = A_\theta(z)\,(z - z^\star), \qquad A_\theta(z) + A_\theta(z)^\top \preceq -2\alpha I, \tag{14}$$

composed with a learned diffeomorphism $z = \varphi(x)$ to the data space (Figure 2A). Here θ contains the learnable parameters of a neural network.

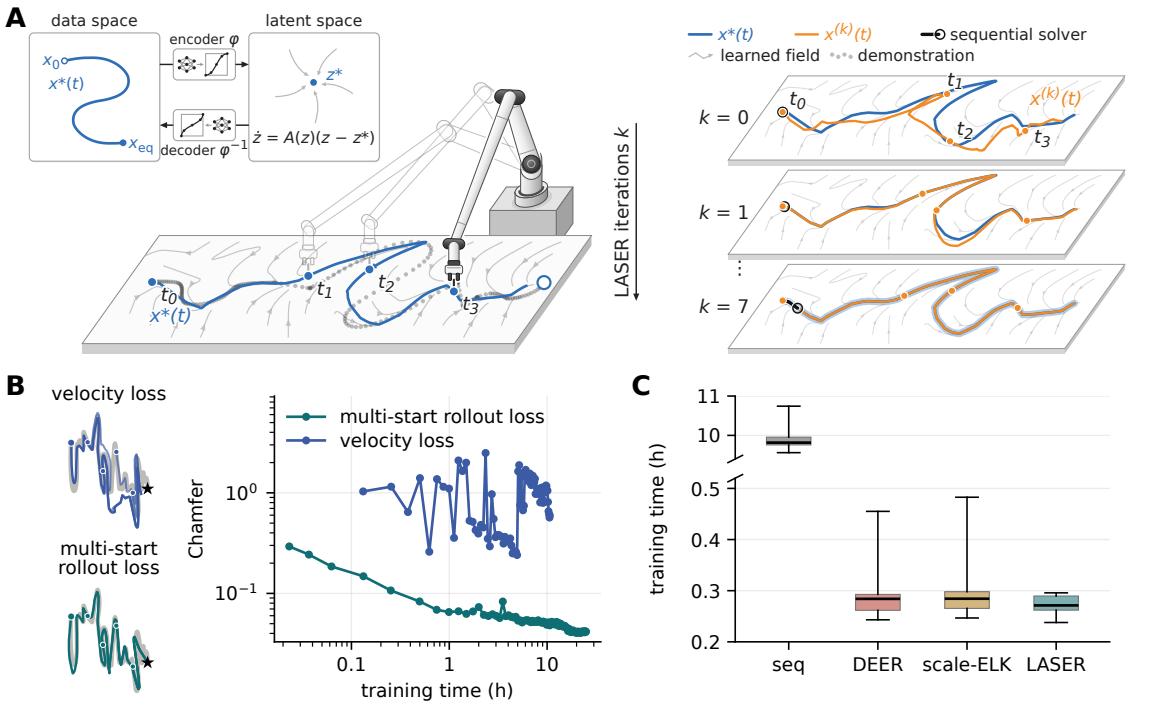


Figure 2: Training ELCD on long rollouts. **A**: left, schematic of the ELCD model (inset) and a rollout $x^\star(t)$ of the learned field; right, a PiT solver finds the trajectory via fixed-point iterations. **B**: left, rollouts of the best seed for each loss (•: starts; ★: goal); right, Chamfer distance between rollouts and training demonstrations during training (one seed). With the same training time, the model trained with the multi-start rollout loss and a PiT solver reaches a lower Chamfer distance than the velocity-trained model. **C**: training time of different solvers over 10,000 optimizer steps (10 seeds). The PiT solvers are $\sim 35\times$ faster than the sequential solver. LASER has the lowest median wall-clock time and the smallest spread. Details in Appendix G.6.

Jaffe et al. (2024) show that (14) is contracting in a state-dependent metric, and hence incrementally stable; they call this model Extended Linearized Contracting Dynamics (ELCD). This guarantee holds along the solutions of (14), whereas our bound (6) must hold along every candidate trajectory, which ELCD does not guarantee (Appendix G.3). ELCD is therefore a natural test case for the damped updates of Section 2.2.

To demonstrate the utility of time parallelization in this setting, we first extend ELCD to include longer multi-start rollouts during training on chained LASA handwriting demonstrations (Appendices G.1 and G.4), whereas previously the model was only trained to predict the next time instant (velocity loss). We then show that this significantly improves performance (Figure 2B). Every optimizer step must now integrate the learned ODE over a long rollout, which is slow sequentially. A PiT solver recovers faster training times while retaining this improved performance. We test three different choices for the matrix $A(s, \delta s, t)$ in Section 2.2 (DEER, scale-ELK, and LASER), and detail

these in Appendix G.5. Overall, all three achieve a $\sim 35\times$ speedup over the sequential solver in total training wall-clock time for the multi-start rollout loss, and LASER is the fastest and most consistent (Figure 2C): DEER and scale-ELK occasionally fail to converge and fall back to the sequential solver, whereas LASER never does (Appendix G.7).

3.2. Bistable Dynamics with an Unstable Initial Guess

We show empirically that the "global" requirement of (6) can sometimes be relaxed to a "local" one when using damped PiT solvers such as LASER. We use the Lorenz system with parameters $\rho = 10$, $\sigma = 10$, $\beta = 8/3$, which is *bistable and non-chaotic*: solutions settle at one of two stable equilibria $C_\pm$, and the origin is a saddle. We choose this system because, even though its solutions are not chaotic, a smooth curve through the vector field that is not a solution can still have a positive finite-time Lyapunov exponent (12), computed *as if* the curve were a solution to the Lorenz ODE. Undamped Gauss–Newton, whose iterates are such curves, therefore struggles to parallelize bistable Lorenz efficiently (Figure 3).

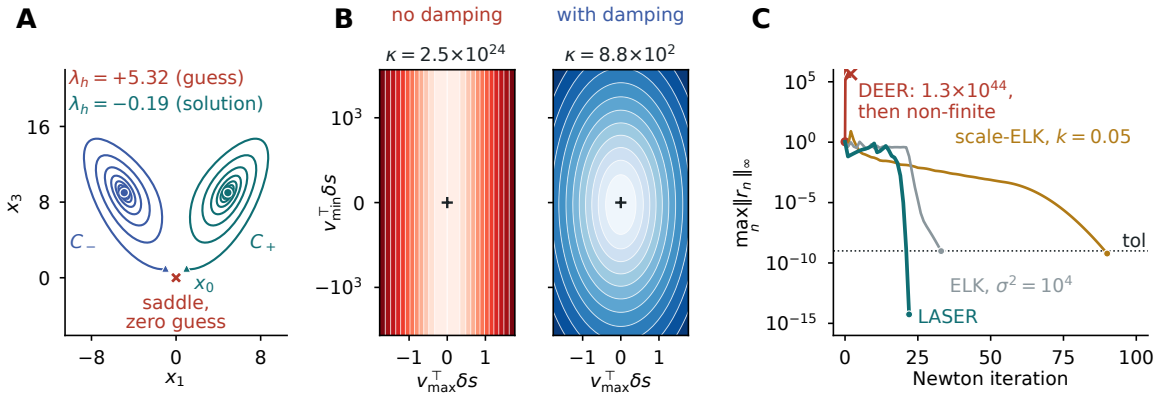


Figure 3: Solving a bistable Lorenz system from an unstable initial guess ($N = 10^3$ time steps). **A**: the trajectory from x_0 settles at C_+ and its mirror image at C_- (x_1–x_3 plane); the zero initial guess is the constant trajectory at the saddle. λ_h is the finite-time Lyapunov exponent (37). **B**: contours of the Gauss–Newton model of the merit functional around the initial guess ($+$), without and with the damping (38) of LASER. The axes are the components of the step δs along the right singular vectors $v_{\max}$ and $v_{\min}$ of the Jacobian of the residual with the largest and smallest singular value; κ is the condition number of this Jacobian. **C**: residual against iteration (dotted line: tolerance 10^{-9}). DEER returns a non-finite iterate at its second step. Scale-ELK and ELK reach the tolerance in 90 and 33 iterations, LASER in 22 (its 10 rejected trial steps are not drawn).

The zero initial guess, the constant trajectory at the saddle, has a finite-time Lyapunov exponent of $\lambda_h = +5.32$, against -0.19 along the solution (Figure 3A). Undamped Gauss–Newton (DEER) therefore takes a first step of order $e^{\lambda_h T}$ and diverges, while the damped variants ELK, scale-ELK (Gonzalez et al., 2024), and LASER converge (Figure 3C). LASER, which adapts its damping at every iteration, makes the Gauss–Newton model at this guess well conditioned (Figure 3B) and needs no tuning. Its iteration count does not grow with the horizon, and at $N \geq 10^4$ it needs the fewest iterations. It is also the fastest in wall-clock time at every horizon, about $4\times$ faster than the fastest PiT baseline, and over $9\times$ faster than the sequential solver at $N \geq 10^4$ (Appendix H).

Additional experiments. Appendix I.1 reports single- and multi-GPU scaling for a frozen contracting RNN, where Jacobi remains competitive without storing Jacobians. Appendix I.2 studies end-to-end training of a globally contracting nonlinear SSM, including a contractivity ablation and an accuracy–speed tradeoff. Appendix I.3 examines neural and adaptive controllers for a two-link robot, where stabilizing feedback enables parallel simulation and training. The DEER variants used there are described in Appendix K.

4. Discussion

We have shown that the stability of nonlinear ODEs is directly tied to our ability to parallelize them in time. This connection is not an artifact of a particular numerical discretization or optimizer, but rather an intrinsic property of the ODEs themselves. We establish this by connecting dynamical systems theory with modern optimization theory, showing that solutions to ODEs can be viewed as global minimizers of a merit functional that satisfies a Polyak–Łojasiewicz (PL) condition. We then relate the stability of the ODE to the PL constant, showing that incrementally stable systems yield PL bounds that do not deteriorate as the time horizon grows, whereas incrementally unstable systems (including chaotic systems) correspond to PL constants that decay exponentially with the time horizon.

Our theory provides a blueprint for co-designing neural networks jointly with PiT algorithms to achieve faster training. Stability influences the number of iterations required for a parallel rollout, while the structure of the state Jacobian determines the cost of each iteration. For example, a recurrent model with small block-diagonal Jacobians admits inexpensive parallel Newton updates (Danieli et al., 2026). On the other hand, global stability is a strong and sometimes restrictive constraint on neural network expressivity. Our experiments indicate that global incremental stability is sufficient but not necessary for efficient parallelization in time. This motivates future theoretical work that relaxes these global conditions to local ones, potentially enabling the design of learned dynamical systems that are both expressive and parallelizable in time. Another direction for future work is extending our analysis to hybrid systems (Lohmiller and Slotine, 2000), which are not always differentiable. This will require careful approximations and new optimizers, as was done for non-differentiable MCMC updates in Zoltowski et al. (2025). Extending our convergence guarantees to account for discrete transitions remains an interesting open direction. Additionally, we will investigate whether contraction metrics and norms can inform the design of better optimizers for PiT algorithms, for example through novel preconditioners.

Building on our imitation learning experiments, model predictive control (MPC) with world models is a natural application for PiT methods, particularly when a learned world model is used to evaluate many candidate action sequences (Hansen et al., 2024). Parallelizing each rollout across time could reduce the delay between observing the state and choosing an action. Our weighted merit functional (Appendix C.2) is particularly relevant in this setting. Since the controller executes only the first action before replanning, it may be advantageous to resolve the early part of each rollout more accurately under a fixed compute budget. Further applications include energy-based models and diffusion models, both of which often exhibit stability properties favorable to the PiT perspective. For diffusion models, parallel sampling across denoising steps has already demonstrated practical speedups (Shih et al., 2023; Tang et al., 2024). Studying perturbation growth along the probability flow ODE could help identify which portions of the denoising trajectory can be solved efficiently in parallel.

Acknowledgment

ZL, EFG, and LK thank the Carney Institute for Brain Science computing condo, as well as the Center for Computation and Visualization, Brown University. LK also thanks Scott Linderman (Stanford) for many helpful conversations about PiT methods. IM's contribution was supported in part by the Australian Research Council through grants DP240101820 and IH210100030. EDS research was supported in part by grant ONR N00014-21-1-2431. ARV and FB's contribution was supported in part by the ARO NeuralSync MURI grant W911NF-24-1-0228.

Generative AI Usage

We used large language models (LLMs) and AI agents to assist with the research, coding, and writing of this paper, and we disclose this usage in accordance with the L4DC policy.

Writing. We used LLMs to simplify and streamline text that we had initially written ourselves, and to assist with parts of the literature review. We have reviewed the full text of this paper, verified its accuracy, and take full responsibility for all of its claims.

Research. We used LLMs and AI agents extensively as coding assistants for writing and validating research code and for generating plots. We tested the code extensively, ran sanity checks on its outputs, and manually reviewed it to confirm its correctness. We take full responsibility for all code and results reported in this paper.

References

Giancarlo Antonino Antonucci, Raphael Andreas Hauser, Debasmita Samaddar, and James Buchanan. Finite-time convergence guarantees for time-parallel methods. *arXiv preprint arXiv:2604.00855*, 2026.

Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Nadia Figueroa, Gerhard Neumann, and Leonel Rozo. Neural contractive dynamical systems. *arXiv preprint arXiv:2401.09352*, 2024.

Alfredo Bellen and Marino Zennaro. Parallel algorithms for initial-value problems for difference and differential equations. *Journal of Computational and Applied Mathematics*, 25(3):341–350, 1989.

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

F. Bullo. *Contraction Theory for Dynamical Systems*. Kindle Direct Publishing, 1.3 edition, 2026. ISBN 979-8836646806.

Philippe Chartier and Bernard Philippe. A parallel shooting technique for solving dissipative ODE's. *Computing*, 51:209–236, 1993.

Federico Danieli, Miguel Sarabia, Xavier Suau Cuadros, Pau Rodriguez, and Luca Zappella. DeepPCR: Parallelizing sequential operations in neural networks. *Advances in Neural Information Processing Systems*, 36:47598–47625, 2023.

Federico Danieli, Pau Rodriguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. ParaRNN: Unlocking parallel training of nonlinear RNNs for large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Alexander Davydov, Saber Jafarpour, Anton V. Proskurnikov, and Francesco Bullo. Non-Euclidean monotone operator theory and applications. *Journal of Machine Learning Research*, 25(307):1–33, 2024. URL http://jmlr.org/papers/v25/23-0805.html.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32, pages 7511–7522, 2019.

Matthew Emmett and Michael L. Minion. Toward an efficient parallel in time method for partial differential equations. *Communications in Applied Mathematics and Computational Science*, 7(1): 105–132, 2012. doi: 10.2140/camcos.2012.7.105.

Robert D. Falgout, Stefanie Friedhoff, Tzanio V. Kolev, Scott P. MacLachlan, and Jacob B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

Mónika Farsang, Ramin Hasani, Daniela Rus, and Radu Grosu. Parallelization of Non-linear State-Space Models: Scaling Up Liquid-Resistance Liquid-Capacitance Networks for Efficient Sequence Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Mónika Farsang, Ramin Hasani, Daniela Rus, and Radu Grosu. Riccati state space models: Non-iterative parallelization for nonlinear sequence modeling. *arXiv preprint arXiv:2609.35441*, 2026.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning (ICML)*, 2018.

Martin J. Gander. 50 years of time parallel time integration. In *Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, Heidelberg, May 6-8, 2013*, pages 69–113. Springer, 2015.

Martin J. Gander and Stefan Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

Xavier Gonzalez. *Unifying Optimization and Dynamics to Parallelize Sequential Computation: A Guide to Parallel Newton Methods for Breaking Sequential Bottlenecks*. PhD thesis, Stanford University, 2026. URL https://purl.stanford.edu/vf943fc9855.

Xavier Gonzalez, Andrew Warrington, Jimmy T.H. Smith, and Scott W. Linderman. Towards Scalable and Stable Parallelization of Nonlinear RNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Xavier Gonzalez, Leo Kozachkov, David M. Zoltowski, Kenneth L. Clarkson, and Scott W. Linderman. Predictability Enables Parallelization of Nonlinear State Space Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xavier Gonzalez, E. Kelly Buchanan, Hyun Dong Lee, Jerry Weihong Liu, Ke Alexander Wang, David M. Zoltowski, Leo Kozachkov, Christopher Re, and Scott Linderman. A unifying framework for parallelizing sequential models with linear dynamical systems. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=fw6GgAIGur.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Conference on Language Modeling (COLM)*, 2024.

Wolfgang Hackbusch. Parabolic multi-grid methods. In Roland Glowinski and Jacques-Louis Lions, editors, *Computing Methods in Applied Sciences and Engineering VI*, pages 189–197. North-Holland, Amsterdam, 1984.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, 2nd revised edition, 1993. doi: 10.1007/978-3-540-78862-1.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Franz A. Heinsen and Leo Kozachkov. Generalized orders of magnitude for scalable, parallel, high-dynamic-range computation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=SUuzb0SOGu.

Florian Hess, Florian Götz, and Daniel Durstewitz. Parallel-in-time training of recurrent neural networks for dynamical systems reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

Casian Iacob, Hassan Razavi, and Simo Särkkä. A parallel-in-time Newton's method-based ODE solver. *arXiv preprint arXiv:2511.01465*, 2025.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25 (2):328–373, February 2013. doi: 10.1162/NECO_a_00393. URL https://doi.org/10.1162/NECO_a_00393.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

Saber Jafarpour, Alexander Davydov, Anton Proskurnikov, and Francesco Bullo. Robust implicit networks via non-Euclidean contractions. *Advances in Neural Information Processing Systems*, 34: 9857–9868, 2021.

Sean Jaffe, Alexander Davydov, Deniz Lapsekili, Ambuj K. Singh, and Francesco Bullo. Learning neural contracting dynamics: Extended linearization and global guarantees. *Advances in Neural Information Processing Systems*, 37:66204–66225, 2024.

Isabel Juniewicz and Josh You. New global AI computing capacity is doubling every 9 months, 2026. URL https://epoch.ai/data-insights/ai-computing-capacity-additions-doubling. Accessed: 2026-11-11.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.

Shiva Kaul. Linear dynamical systems as a core computational primitive. In *Advances in Neural Information Processing Systems*, 2020.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Leo Kozachkov, Mikael Lundqvist, Jean-Jacques Slotine, and Earl K. Miller. Achieving stable dynamics in neural circuits. *PLoS Computational Biology*, 16(8):e1007659, 2020.

Leo Kozachkov, Michaela Ennis, and Jean-Jacques Slotine. RNNs of RNNs: Recursive construction of stable assemblies of recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Leo Kozachkov, Patrick Wensing, and Jean-Jacques Slotine. Generalization as dynamical robustness—the role of Riemannian contraction in supervised learning. *Transactions on Machine Learning Research*, 2023.

E. Lelarasmee, Albert E. Ruehli, and Alberto Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1(3):131–145, 1982.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations*, 2024.

Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. A "parareal" in time discretization of PDEs. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(7):661–668, 2001.

Winfried Lohmiller and Jean-Jacques E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

Winfried Lohmiller and Jean-Jacques E. Slotine. Nonlinear process control using contraction theory. *AIChE Journal*, 46(3):588–596, 2000. doi: 10.1002/aic.690460317.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, pages 7829–7838. PMLR, 2021.

J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733, December 1964.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

Benjamin W. Ong and Jacob B. Schroder. Applications of time parallelization. *Computing and Visualization in Science*, 23:1–15, 2020.

James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York and London, 1970. Republished by SIAM in 2000.

A. Y. Pogromsky and A. S. Matveev. A non-quadratic criterion for stability of forced oscillations. *Systems & Control Letters*, 62(5):408–412, 2013. doi: 10.1016/j.sysconle.2013.02.002.

A. Y. Pogromsky and A. S. Matveev. Stability analysis via averaging functions. *IEEE Transactions on Automatic Control*, 61(4):1081–1086, 2016. doi: 10.1109/tac.2015.2459152.

Boris T. Polyak. Gradient methods for the minimisation of functionals. *Zh. Vychisl. Mat. Mat. Fiz.*, 3 (4):643–653, 1963.

Max Revay and Ian Manchester. Contracting implicit recurrent neural networks: Stable models with improved trainability. In *Learning for Dynamics and Control*, pages 393–403. PMLR, 2020.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Troy Shinbrot, Celso A. Grebogi, Jack Wisdom, and James A. Yorke. Chaos in a double pendulum. *American Journal of Physics*, 60:491–499, 1992. doi: 10.1119/1.16860. URL https://hal.science/hal-01403609.

Jean-Jacques E. Slotine and Weiping Li. On the adaptive control of robot manipulators. *The International Journal of Robotics Research*, 6(3):49–59, 1987.

Jimmy T.H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, 2021.

Simo Särkkä and Lennart Svensson. Levenberg-Marquardt and line-search extended Kalman smoothers. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5875–5879. IEEE, 2020.

Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

Zhiwei Tang, Jiasheng Tang, Hao Luo, Fan Wang, and Tsung-Hui Chang. Accelerating parallel sampling of diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

Gerald Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Patrick M. Wensing and Jean-Jacques Slotine. Beyond convexity—contraction and global convergence of gradient descent. *PLoS ONE*, 15(8):e0236661, 2020.

Min Yan, Can Huang, Peter Bienstman, Peter Tino, Wei Lin, and Jie Sun. Emerging opportunities and challenges for the future of reservoir computing. *Nature Communications*, 15:2056, 2024.

Eviatar Yemini, Tadas Jucikas, Laura J. Grundy, André E. X. Brown, and William R. Schafer. A database of Caenorhabditis elegans behavioral phenotypes. *Nature Methods*, 10(9):877–879, 2013.

David M. Zoltowski, Skyler Wu, Xavier Gonzalez, Leo Kozachkov, and Scott W. Linderman. Parallelizing MCMC Across the Sequence Length. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Appendix

A Related Work **19**

B Proofs of the Continuous-Time PL Bounds **20**
B.1 Proof of Theorem 1 . . . 20
B.2 Proof of Theorem 2 . . . 22
B.3 Lower Bound for a General Growth Rate . . . 24

C Choice of Merit Functional **25**
C.1 Penalties That Satisfy a PL Inequality . . . 25
C.2 Exponential Weighting of the Merit Functional . . . 26

D Picard and Jacobi Updates **28**
D.1 Derivation of Picard Iteration . . . 29
D.2 Derivation of the Exact-Flow Jacobi Correction . . . 29
D.3 Jacobi Convergence under Incremental Stability . . . 30

E The LASER Update **32**

F Jacobi-Specific Bounds for the Discretization–Parallelization Tradeoff **32**
F.1 Proof of Lemma 6 . . . 35
F.2 Estimates of the Sequential Error . . . 36
F.3 Estimate of Critical Step Size . . . 38

G Training ELCD on Long Rollouts **38**
G.1 Tasks . . . 39
G.2 Model . . . 39
G.3 Contraction of ELCD . . . 39
G.4 Training . . . 40
G.5 Solvers and Timing Protocol . . . 40
G.6 Details of Figure 2 . . . 41
G.7 Further Results on chain8z . . . 42
G.8 Eight-Link Pendulum . . . 45
G.9 The Original Tasks of Jaffe et al. (2024) . . . 46

H Details of the Bistable Lorenz Experiment **48**

I Supplementary Experiments **48**
I.1 GPU Acceleration of Reservoir Computing . . . 48
I.1.1 Experimental Protocol . . . 51
I.2 Parallel Training of Contractive State-Space Models . . . 52
I.2.1 Architecture and Training Details . . . 54
I.3 Parallel-in-Time Robotic Control and Simulation . . . 56
I.3.1 Neural Controller Details . . . 57
I.3.2 Adaptive Controller Details . . . 58

J Parallel Scan for Linear ODEs **59**

K DEER, quasi-DEER, and DEER-exp **61**

L Contraction on Average for Incremental Exponential Stability with Overshoot and Global Jacobi Convergence **62**

Appendix A. Related Work

Parallel-in-time methods have been studied for more than half a century, beginning with the work of Nievergelt (1964). Although their implementations have changed with the available hardware, the underlying motivation remains the same: to compute different parts of a trajectory simultaneously, despite the sequential dependence imposed by the dynamics. For broader accounts of this literature, we refer readers to Gander (2015) and Ong and Schroder (2020).

Classical parallel-in-time methods. Several major approaches have emerged from this early work. Multiple shooting methods divide the time interval into smaller intervals and iteratively enforce agreement between their solutions (Bellen and Zennaro, 1989; Chartier and Philippe, 1993). Waveform relaxation instead iterates over entire trajectories of coupled subsystems (Lelarasmee et al., 1982). Multigrid methods introduce corrections at different temporal resolutions (Hackbusch, 1984). A particularly influential method is Parareal (Lions et al., 2001), which combines inexpensive sequential coarse integration with more accurate integrations performed in parallel. Its connections to multiple shooting and multigrid, together with convergence results, were developed by Gander and Vandewalle (2007). Other major developments include multigrid reduction in time (MGRIT; Falgout et al., 2014) and the parallel full approximation scheme in space and time (PFASST; Emmett and Minion, 2012). Finally, certain nonlinear ODEs admit an analytic transformation into linear ODEs, thus allowing for direct parallelization (Farsang et al., 2026). These methods provide several ways of organizing parallel computation; our work focuses on how the underlying dynamics affect the conditioning of trajectory optimization and the convergence of particular iterations.

Parallel nonlinear equation solving. Recasting a sequential computation as a system of nonlinear equations also has a long history. Bellen and Zennaro (1989) developed parallel iterations for nonlinear recurrences, including those obtained from numerical ODE solvers. Chartier and Philippe (1993) studied related shooting methods for dissipative ODEs, already identifying dissipation as an important condition for effective parallel solution. More recently, Song et al. (2021) applied nonlinear Jacobi and Gauss–Seidel iterations to accelerate feedforward computations in machine learning. DEER (Lim et al., 2024) and DeepPCR (Danieli et al., 2023) use Newton-type iterations, solving the resulting linear systems using associative scans, also known as parallel cyclic reduction. Iacob et al. (2025) similarly combine Newton's method with parallel scans to solve discretized nonlinear ODEs. Our analysis is motivated by these methods and by the need to understand when their parallel iterations converge quickly enough to yield a computational advantage.

Stability and optimization conditioning. The closest theoretical work is Gonzalez et al. (2025), who connect the stability of discrete-time nonlinear systems to the conditioning of the optimization problems used to evaluate them. Their results distinguish stable systems from unstable systems whose conditioning deteriorates exponentially with sequence length. We develop this connection directly for continuous-time ODEs, before selecting a numerical discretization. Our analysis uses the Polyak–Łojasiewicz inequality (Polyak, 1963; Karimi et al., 2016) to relate the conditioning of the residual functional to the growth or decay of perturbations. This separates properties of the continuous-time dynamics from those introduced by the numerical scheme, and allows us to examine how discretization error and incomplete parallel convergence jointly affect solution accuracy.

Common frameworks for iterative solvers. Gonzalez et al. (2026) give a common description of Newton, quasi-Newton, Picard, and Jacobi iterations for discrete sequential models using linear

dynamical systems. Different choices of the approximate linearization lead to different algorithms and convergence properties. Our trajectory-update formulation provides a continuous-time counterpart to this perspective. It recovers Picard iteration, Newton-type corrections, and exact-flow Jacobi updates, and connects their behavior to the stability of the underlying ODE. In particular, our Jacobi analysis gives geometric convergence under incremental stability and makes explicit how the convergence rate depends on the length of each time interval.

Stabilization and weighted errors. Instability can make parallel updates unreliable even when the desired solution is stable, because the linearization is evaluated along the current trajectory guess. Gonzalez et al. (2024) address numerical instability through regularized Newton-type methods (Nocedal and Wright, 2006; Särkkä and Svensson, 2020), including ELK and scale-ELK. Our LASER method builds on this direction by adapting the damping to estimated perturbation growth along the current iterate. A related approach is to change how errors at different times are weighted. Antonucci et al. (2026) study convergence and stopping criteria for time-decomposition methods using weighted interface defects. We instead analyze exponential weighting of the continuous-time ODE residual and its effect on the PL bound in the corresponding weighted norm. This distinguishes improving conditioning in a weighted metric from recovering an unstable trajectory accurately in the original coordinates. Another related approach is GTF-DEER (Hess et al., 2026), which is a technique for learning chaotic dynamics by using teacher forcing to reduce the spectral norm of the dynamics Jacobians.

State-space models and learned stable dynamics. Modern state-space architectures such as S5 (Smith et al., 2023) and Mamba (Gu and Dao, 2024) exploit recurrences that are linear in the hidden state, allowing direct evaluation by parallel scans (see also Martin and Cundy, 2018; Kaul, 2020). Nonlinear state transitions generally require additional iteration, making their stability relevant to parallel efficiency. Separately, contraction theory (Lohmiller and Slotine, 1998) has motivated learned dynamical models with stability and robustness guarantees, including neural contractive dynamical systems (Beik-Mohammadi et al., 2024) and extended linearized contracting dynamics (Jaffe et al., 2024). Our work connects these modeling choices to the computational cost of trajectory evaluation. The imitation-learning experiments also examine the distinction between stability along solutions and stability along the intermediate trajectory guesses encountered by a parallel solver.

Appendix B. Proofs of the Continuous-Time PL Bounds

B.1. Proof of Theorem 1

Theorem 1 (Incremental stability implies a PL inequality) *Under the decay estimate* (6)*, the merit functional* (2) *satisfies the Polyak–Łojasiewicz inequality* (3) *with* $\mathcal{L}^* = 0$*, and its optimal PL constant obeys*

$$\sqrt{\mu_\star} \;\geq\; \frac{c}{a} \cdot \frac{1}{1 - e^{-cT}} \;\geq\; \frac{c}{a} \;>\; 0. \tag{7}$$

Proof The idea is to find a trajectory correction that reduces the residual at a known rate. Stability will ensure that this correction is not too large. The loss therefore cannot have a small gradient unless the residual is also small. Fix a candidate trajectory s and write

$$r_s(t) = \dot{s}(t) - f(s(t), t), \qquad A_s(t) = J(s(t), t).$$

Recall that $\mathcal{L}[s] = \frac{1}{2}\|r_s\|_{L^2}^2$, where $\|r_s\|_{L^2}^2 = \int_0^T \|r_s(t)\|_2^2\, dt$. The exact solution has zero residual, so $\mathcal{L}^* = 0$.

Step 1: Choose a correction that reduces the residual. Let η solve the linear ODE

$$\dot{\eta}(t) = A_s(t)\eta(t) + r_s(t), \qquad \eta(0) = 0.$$

This is the linearized dynamics driven by the current residual. To see why it is useful, consider the corrected trajectory $s - \varepsilon\eta$. It has the same initial condition as s, and a first-order expansion gives

$$r[s - \varepsilon\eta] = r_s - \varepsilon(\dot{\eta} - A_s\eta) + o(\varepsilon) = (1 - \varepsilon)r_s + o(\varepsilon).$$

Thus, subtracting a small multiple of η removes that same fraction of the residual, to first order.

Step 2: Use stability to bound the size of the correction. The solution of the linear ODE is

$$\eta(t) = \int_0^t \Phi_s(t, \tau) r_s(\tau)\, d\tau.$$

The decay estimate (6) therefore gives

$$\|\eta(t)\|_2 \le \int_0^t a e^{-c(t-\tau)} \|r_s(\tau)\|_2\, d\tau.$$

Each residual value drives the correction at later times, but its effect decays exponentially. The total weight of this decay is at most

$$B_T := \int_0^T a e^{-cu}\, du = \frac{a(1 - e^{-cT})}{c}.$$

We can turn this observation into an L^2 bound using Cauchy–Schwarz:

$$\begin{aligned}\|\eta(t)\|_2^2 &\le \left(\int_0^t a e^{-c(t-\tau)}\, d\tau\right)\left(\int_0^t a e^{-c(t-\tau)} \|r_s(\tau)\|_2^2\, d\tau\right) \\ &\le B_T \int_0^t a e^{-c(t-\tau)} \|r_s(\tau)\|_2^2\, d\tau.\end{aligned}$$

Integrating over t and switching the order of integration then gives

$$\|\eta\|_{L^2}^2 \le B_T \int_0^T \|r_s(\tau)\|_2^2 \left(\int_\tau^T a e^{-c(t-\tau)}\, dt\right) d\tau \le B_T^2 \|r_s\|_{L^2}^2.$$

In other words, the linear system has L^2 input-output gain at most B_T:

$$\|\eta\|_{L^2} \le B_T \|r_s\|_{L^2}. \tag{15}$$

Step 3: Compare the decrease in loss with the gradient. Step 1 shows that

$$\frac{d}{d\varepsilon}\mathcal{L}[s-\varepsilon\eta]\Big|_{\varepsilon=0} = -\|r_s\|_{L^2}^2.$$

By the definition of the gradient, the same derivative is $-\langle\nabla\mathcal{L}[s],\eta\rangle_{L^2}$.[4] Using Cauchy–Schwarz once more,

$$\begin{aligned}\|r_s\|_{L^2}^2 &= \langle\nabla\mathcal{L}[s],\eta\rangle_{L^2}\\ &\le \|\nabla\mathcal{L}[s]\|_{L^2}\|\eta\|_{L^2}\\ &\le B_T\|\nabla\mathcal{L}[s]\|_{L^2}\|r_s\|_{L^2}.\end{aligned} \tag{16}$$

If the residual is zero, the PL inequality already holds. Otherwise, cancel $\|r_s\|_{L^2}$ and square to obtain

$$\frac{1}{2}\|\nabla\mathcal{L}[s]\|_{L^2}^2 \ge \frac{1}{B_T^2}\frac{1}{2}\|r_s\|_{L^2}^2 = \frac{c^2}{a^2(1-e^{-cT})^2}\big(\mathcal{L}[s]-\mathcal{L}^*\big).$$

This holds for every candidate trajectory s, proving the claimed bound on $\mu_\star$. Since $1-e^{-cT}\le 1$, it also gives $\mu_\star \ge c^2/a^2$. ■

B.2. Proof of Theorem 2

Theorem 2 (Instability implies a vanishing PL constant) *Suppose (8) holds. Then, for $\lambda T \ge 2$, the optimal PL constant of the merit functional (2) obeys*

$$\sqrt{\mu_\star} \;\le\; \frac{e\lambda a^2}{b}e^{-\lambda T}. \tag{9}$$

Note that, for fixed a, b, and λ, the PL constant $\mu_\star$ goes to zero as $T\to\infty$.

Proof The idea is to introduce a perturbation over a short initial interval and then let it follow the linearized dynamics. Its residual is confined to that initial interval, but its size continues to grow. We then use a backward linear ODE to turn this small residual-to-trajectory ratio into a small PL ratio.

Write $A(t)=J(x(t),t)$, where x is the exact solution, and use L^2 norms on $[0,T]$ throughout.

Step 1: Find a perturbation that keeps growing. Choose a unit vector v that realizes the largest amplification at the final time:

$$\|\Phi_x(T,0)v\|_2 = \|\Phi_x(T,0)\| \ge be^{\lambda T}.$$

Set $z(t)=\Phi_x(t,0)v$. Since $z(T)=\Phi_x(T,t)z(t)$, the upper bound in (8) gives

$$be^{\lambda T} \le \|z(T)\|_2 \le ae^{\lambda(T-t)}\|z(t)\|_2.$$

Thus this same initial direction satisfies

$$\frac{b}{a}e^{\lambda t} \le \|z(t)\|_2 \le ae^{\lambda t}, \qquad 0\le t\le T. \tag{17}$$

This is where both growth bounds are used: growth at the final time cannot come entirely from an arbitrarily large last burst of amplification.

4. Here the gradient norm means the largest absolute directional derivative per unit L^2 trajectory perturbation, with perturbations in H^1 that vanish at $t=0$. When an L^2 gradient exists, this is its usual norm. If these directional derivatives are unbounded, we take the norm to be $+\infty$, and the PL inequality is automatic. The calculation therefore only needs the case of a finite gradient norm.

Step 2: Introduce the perturbation near the beginning. We cannot use z directly, because a trajectory perturbation must vanish at $t = 0$. Instead, fix $0 < \delta < T$ and gradually turn it on:

$$\eta(t) = \chi(t)z(t), \qquad \chi(t) = \begin{cases} t/\delta, & 0 \le t \le \delta, \\ 1, & \delta < t \le T. \end{cases}$$

Because $\dot{z} = Az$, the residual of this perturbation under the linearized dynamics is

$$u(t) := \dot{\eta}(t) - A(t)\eta(t) = \dot{\chi}(t)z(t).$$

In particular, u is zero after time δ: once the perturbation has been introduced, it follows the dynamics without further forcing. Using (17),

$$\|u\|_{L^2}^2 \le \frac{a^2}{\delta^2} \int_0^\delta e^{2\lambda t}\, dt = \frac{a^2}{2\lambda\delta^2}\big(e^{2\lambda\delta} - 1\big),$$
$$\|\eta\|_{L^2}^2 \ge \frac{b^2}{a^2} \int_\delta^T e^{2\lambda t}\, dt = \frac{b^2}{2\lambda a^2}\big(e^{2\lambda T} - e^{2\lambda\delta}\big).$$

Therefore,

$$\frac{\|u\|_{L^2}^2}{\|\eta\|_{L^2}^2} \le \frac{a^4}{b^2\delta^2} \frac{e^{2\lambda\delta} - 1}{e^{2\lambda T} - e^{2\lambda\delta}}. \tag{18}$$

For fixed δ, the residual cost stays bounded while the trajectory perturbation grows exponentially with T.

Step 3: Turn this ratio into an upper bound on the PL constant. The ratio in (18) compares a residual with a trajectory perturbation. The PL inequality instead compares a gradient with a residual. To connect the two, solve the backward linear ODE

$$-\dot{p}(t) - A(t)^\top p(t) = \eta(t), \qquad p(T) = 0.$$

This is the adjoint equation for the linearized dynamics. Integration by parts, using $\eta(0) = 0$ and $p(T) = 0$, gives

$$\|\eta\|_{L^2}^2 = \int_0^T (-\dot{p} - A^\top p)^\top \eta\, dt = \int_0^T p^\top(\dot{\eta} - A\eta)\, dt = \langle p, u\rangle_{L^2}.$$

Cauchy–Schwarz therefore implies

$$\frac{\|\eta\|_{L^2}^2}{\|p\|_{L^2}^2} \le \frac{\|u\|_{L^2}^2}{\|\eta\|_{L^2}^2}. \tag{19}$$

In other words, the backward equation turns our perturbation with a small residual into a residual p whose gradient is small by comparison.

To see this for the nonlinear merit functional, let s_ε solve the slightly forced IVP

$$\dot{s}_\varepsilon(t) = f(s_\varepsilon(t), t) + \varepsilon p(t), \qquad s_\varepsilon(0) = x_0.$$

For sufficiently small ε, this trajectory stays close to x. Its residual is exactly εp, so $2\mathcal{L}[s_\varepsilon] = \varepsilon^2\|p\|_{L^2}^2$. The gradient follows by differentiating the loss and integrating by parts:

$$\begin{aligned} \nabla\mathcal{L}[s_\varepsilon] &= \varepsilon\big(-\dot{p} - J(s_\varepsilon(t), t)^\top p\big) \\ &= \varepsilon\eta + o(\varepsilon). \end{aligned}$$

There is no terminal boundary term because $p(T) = 0$, and the last equality uses $s_\varepsilon \to x$ and continuity of the Jacobian. Since $\mathcal{L}^* = 0$, every valid PL constant must then satisfy

$$\mu_\star \le \lim_{\varepsilon \to 0} \frac{\|\nabla \mathcal{L}[s_\varepsilon]\|_{L^2}^2}{2\mathcal{L}[s_\varepsilon]} = \frac{\|\eta\|_{L^2}^2}{\|p\|_{L^2}^2} \le \frac{\|u\|_{L^2}^2}{\|\eta\|_{L^2}^2}.$$

Together with (18), this gives an upper bound for any $0 < \delta < T$.

Finally, if $\lambda T \ge 2$, choose $\delta = 1/\lambda$. Then

$$\mu_\star \le \frac{a^4}{b^2}\lambda^2 \frac{e^2 - 1}{e^{2\lambda T} - e^2} \le e^2 \frac{a^4}{b^2}\lambda^2 e^{-2\lambda T},$$

where the last inequality uses $1 - e^{2-2\lambda T} \ge 1 - e^{-2}$. For fixed a, b, and $\lambda > 0$, the right-hand side tends to zero as $T \to \infty$. ■

B.3. Lower Bound for a General Growth Rate

Suppose that, for some $a \ge 1$ and $\lambda \in \mathbb{R}$,

$$\|\Phi_s(t, \tau)\| \le a e^{\lambda(t-\tau)} \tag{20}$$

for every admissible candidate trajectory s and all $0 \le \tau \le t \le T$. Then

$$\sqrt{\mu_\star} \ge \frac{1}{a} \frac{\lambda}{e^{\lambda T} - 1}, \tag{21}$$

with limiting value $1/(aT)$ when $\lambda = 0$. This recovers Theorem 1 by taking $\lambda = -c$.

The bound follows from the same argument as Appendix B.1, with $ae^{\lambda(t-\tau)}$ in place of $ae^{-c(t-\tau)}$. To also record the estimate for a general forcing, fix a candidate trajectory s and let η solve

$$\dot\eta(t) = J(s(t), t)\eta(t) + u(t), \qquad \eta(0) = 0,$$

where $u \in L^2([0, T]; \mathbb{R}^D)$. The solution formula and (20) give

$$\eta(t) = \int_0^t \Phi_s(t, \tau)u(\tau)\, d\tau, \qquad \|\eta(t)\|_2 \le \int_0^t a e^{\lambda(t-\tau)} \|u(\tau)\|_2\, d\tau.$$

Define the total weight

$$G_T := a \int_0^T e^{\lambda v}\, dv = \begin{cases} a\dfrac{e^{\lambda T} - 1}{\lambda}, & \lambda \ne 0, \\ aT, & \lambda = 0. \end{cases}$$

Repeating the Cauchy–Schwarz estimate in Step 2 of Appendix B.1 gives

$$\|\eta\|_{L^2} \le G_T \|u\|_{L^2}. \tag{22}$$

Equivalently, for the linearized residual map $\mathcal{J}_s\eta = \dot\eta - J(s(t), t)\eta$, this says $\|\mathcal{J}_s^{-1}u\|_{L^2} \le G_T\|u\|_{L^2}$. Taking $u = r_s$ and applying Step 3 of the same proof yields

$$\frac{1}{2}\|\nabla \mathcal{L}[s]\|_{L^2}^2 \ge \frac{1}{G_T^2}\big(\mathcal{L}[s] - \mathcal{L}^*\big).$$

Thus $\sqrt{\mu_\star} \ge 1/G_T$, which is (21); when $\lambda = 0$, this becomes $\sqrt{\mu_\star} \ge 1/(aT)$.

Appendix C. Choice of Merit Functional

This appendix considers two alternatives to the merit functional (2): a general differentiable penalty on the residual (Appendix C.1) and an exponentially weighted L^2 norm (Appendix C.2).

C.1. Penalties That Satisfy a PL Inequality

In the main text, we focus on the squared-residual merit functional $\mathcal{L}[s] = \frac{1}{2}\|r[s]\|_{L^2}^2$. A natural question is whether the optimization behavior depends on this particular quadratic choice, or whether different penalties on the residual can improve the Polyak–Łojasiewicz constant. The following result extends the PL lower-bound guarantee to a broad class of residual merit functionals: any continuously Fréchet differentiable functional of the residual that itself satisfies a PL inequality inherits a PL lower bound when composed with the trajectory residual map.

Proposition 3 (PL bound under a change of merit functional) *Let $V : L^2([0,T];\mathbb{R}^D) \to \mathbb{R}$ be continuously Fréchet differentiable, and suppose that V satisfies the Polyak–Łojasiewicz inequality*

$$\frac{1}{2}\|\nabla V(u)\|_{L^2}^2 \geq \mu_V\big(V(u) - V^\star\big) \qquad \text{for all } u \in L^2([0,T];\mathbb{R}^D),$$

for some $\mu_V > 0$, where $V^\star = \inf_{u\in L^2} V(u)$, and that $\nabla V(u) = 0$ if and only if $u = 0$. Define $\mathcal{L}_V[s] \coloneqq V(r[s])$, and suppose that $\sigma_{\min}(\mathcal{J}_s)^2 \geq \gamma$ for all admissible s, for some $\gamma > 0$. Then $\mathcal{L}_V$ satisfies the PL inequality

$$\frac{1}{2}\|\nabla \mathcal{L}_V[s]\|_{L^2}^2 \geq \mu_V\gamma\big(\mathcal{L}_V[s] - V^\star\big) \qquad \text{for all admissible } s.$$

In particular, $\mathcal{L}_V[s] = V^\star$ if and only if $r[s] = 0$.

Under the upper bound (20), the inverse estimate (22) shows that the hypothesis on $\sigma_{\min}(\mathcal{J}_s)$ holds with $\sqrt{\gamma} = \frac{1}{a}\,\lambda/(e^{\lambda T} - 1)$ (with value $1/(aT)$ when $\lambda = 0$), so the proposition applies to every system covered by our lower bound.

Remark 2 *Any μ-strongly convex, continuously Fréchet differentiable functional V on $L^2([0,T];\mathbb{R}^D)$ satisfies a PL inequality with constant $\mu_V \geq \mu$. A basic example is the weighted quadratic residual functional*

$$V(u) = \frac{1}{2}\int_0^T u(t)^\top M(t) u(t)\, dt,$$

where $M(t) = M(t)^\top \succ 0$ uniformly in t, i.e., there exists $m > 0$ such that $u^\top M(t) u \geq m\|u\|_2^2$ for all $u \in \mathbb{R}^D$ and $t \in [0,T]$. In this case, V is strongly convex on L^2, and therefore satisfies the hypotheses of Proposition 3.

Proof [Proof of Proposition 3] By the chain rule for Fréchet derivatives, $\nabla \mathcal{L}_V[s] = \mathcal{J}_s^* \nabla V(r[s])$. Since the nonzero singular values of $\mathcal{J}_s$ and $\mathcal{J}_s^*$ coincide,

$$\|\mathcal{J}_s^* v\|_{L^2} \geq \sigma_{\min}(\mathcal{J}_s)\|v\|_{L^2} \geq \sqrt{\gamma}\,\|v\|_{L^2} \qquad \text{for all } v \in L^2([0,T];\mathbb{R}^D).$$

Applying this with $v = \nabla V(r[s])$, squaring, and using the PL inequality for V, we obtain

$$\frac{1}{2}\|\nabla \mathcal{L}_V[s]\|_{L^2}^2 \geq \frac{\gamma}{2}\|\nabla V(r[s])\|_{L^2}^2 \geq \mu_V\gamma\big(V(r[s]) - V^\star\big) = \mu_V\gamma\big(\mathcal{L}_V[s] - V^\star\big).$$

Finally, the PL inequality for V implies that stationary points of V are global minimizers, and by assumption $\nabla V(u) = 0$ if and only if $u = 0$. Hence $\mathcal{L}_V[s] = V^\star \Leftrightarrow \nabla V(r[s]) = 0 \Leftrightarrow r[s] = 0$, which completes the proof. ■

C.2. Exponential Weighting of the Merit Functional

In an unstable system, small perturbations are amplified exponentially in time, and Theorem 2 shows that the PL constant of (2) vanishes as T grows. Often the early part of a trajectory matters more than the late part, as in a weather forecast, where the first few days matter more than a day one month ahead. We can then give less weight to late-time residuals,

$$\mathcal{L}_\alpha[s] = \frac{1}{2}\int_0^T e^{-2\alpha t}\|\dot{s}(t) - f(s(t),t)\|_2^2\,dt = \frac{1}{2}\|r[s]\|_{L^2_\alpha}^2, \qquad \alpha \geq 0, \tag{23}$$

where $\|u\|_{L^2_\alpha}^2 := \int_0^T e^{-2\alpha t}\|u(t)\|_2^2\,dt$. We measure perturbations of s in the same norm and write ∇_α for the gradient with respect to the L^2_α inner product. A *weighted PL inequality* is (3) with $\|\nabla_\alpha \mathcal{L}_\alpha[s]\|_{L^2_\alpha}$ as the gradient norm, and $\mu_{\alpha,\star}$ denotes its optimal constant.

Weighting as a change of coordinates. Let $(Eu)(t) := e^{-\alpha t}u(t)$ and map a candidate trajectory s to $\tilde{s} := Es$. The transformed trajectory is a candidate for the IVP

$$\dot{x} = \tilde{f}(x,t), \qquad \tilde{f}(x,t) := -\alpha x + e^{-\alpha t} f(e^{\alpha t}x, t), \qquad x(0) = x_0. \tag{24}$$

We write $\tilde{r}$, $\tilde{\mathcal{L}}$ and $\tilde{\Phi}$ for the residual, the merit functional (2) and the state-transition matrix of this transformed IVP.

Lemma 3 (Weighting is a change of coordinates) *For every candidate trajectory s and $\tilde{s} = Es$,*

$$\tilde{r}[\tilde{s}] = E\,r[s], \qquad \tilde{\mathcal{L}}[\tilde{s}] = \mathcal{L}_\alpha[s], \qquad \|\nabla\tilde{\mathcal{L}}[\tilde{s}]\|_{L^2} = \|\nabla_\alpha\mathcal{L}_\alpha[s]\|_{L^2_\alpha}, \tag{25}$$

and $\tilde{\Phi}_{\tilde{s}}(t,\tau) = e^{-\alpha(t-\tau)}\Phi_s(t,\tau)$ for all $0 \leq \tau \leq t \leq T$.

Proof Since $\dot{\tilde{s}} = -\alpha\tilde{s} + e^{-\alpha t}\dot{s}$ and $e^{\alpha t}\tilde{s} = s$,

$$\dot{\tilde{s}}(t) - \tilde{f}(\tilde{s}(t),t) = e^{-\alpha t}\big(\dot{s}(t) - f(s(t),t)\big),$$

which is the first identity. The second follows because $\|Eu\|_{L^2} = \|u\|_{L^2_\alpha}$. For the third, E maps the perturbations of s one-to-one onto the perturbations of $\tilde{s}$ and preserves their size, $\|E\eta\|_{L^2} = \|\eta\|_{L^2_\alpha}$, while $\mathcal{L}_\alpha[s+\eta] = \tilde{\mathcal{L}}[\tilde{s} + E\eta]$. The two functionals therefore have the same directional derivatives per unit perturbation, and the same gradient norm. Finally, $\partial_x\tilde{f}(\tilde{s}(t),t) = J(s(t),t) - \alpha I$, and $e^{-\alpha(t-\tau)}\Phi_s(t,\tau)$ solves (5) with this Jacobian. ■

Theorem 4 (PL bound for the weighted merit functional) *Suppose the upper bound (20) holds for some $a \geq 1$ and $\lambda \in \mathbb{R}$. Then, for every $\alpha \geq 0$, the weighted merit functional (23) satisfies a weighted PL inequality, and its optimal constant obeys*

$$\sqrt{\mu_{\alpha,\star}} \;\geq\; \frac{1}{a}\,\frac{\lambda - \alpha}{e^{(\lambda-\alpha)T} - 1}, \tag{26}$$

with value $1/(aT)$ when $\alpha = \lambda$. For $\alpha > \lambda$, the right-hand side is at least $(\alpha - \lambda)/a$ for every T.

Proof Every candidate trajectory of the transformed IVP (24) is Es for a candidate s of the original one. By Lemma 3, the transformed IVP therefore satisfies (20) with $\lambda - \alpha$ in place of λ, and (21) gives the stated bound for the optimal PL constant of $\tilde{\mathcal{L}}$. By (25), a PL inequality for $\tilde{\mathcal{L}}$ at $\tilde{s}$ is the weighted PL inequality for $\mathcal{L}_\alpha$ at s with the same constant. ■

For an unstable system, where $\lambda > 0$, any $\alpha > \lambda$ places the transformed IVP in CASE 1 with contraction rate $\alpha - \lambda$, and the weighted PL constant stays bounded away from zero as T grows, unlike the unweighted one in Theorem 2.

The PL estimate of Theorem 4 gives a decay rate for gradient flow in the weighted norm. Exact Newton updates are unchanged by the weighting. We make this distinction explicit below.

Decay along gradient flow. Let $\vartheta \mapsto s^{(\vartheta)}$ be the gradient flow $\partial_\vartheta s^{(\vartheta)} = -\nabla_\alpha \mathcal{L}_\alpha[s^{(\vartheta)}]$ started from an initial guess $s^{(0)}$, and let $\mu_\alpha > 0$ be a weighted PL constant. Then $\frac{d}{d\vartheta}\mathcal{L}_\alpha = -\|\nabla_\alpha \mathcal{L}_\alpha\|^2_{L^2_\alpha} \leq -2\mu_\alpha \mathcal{L}_\alpha$, and since $e^{-2\alpha T}\mathcal{L} \leq \mathcal{L}_\alpha \leq \mathcal{L}$,

$$\mathcal{L}[s^{(\vartheta)}] \;\leq\; e^{2\alpha T}\mathcal{L}_\alpha[s^{(\vartheta)}] \;\leq\; e^{2\alpha T - 2\mu_\alpha \vartheta}\,\mathcal{L}[s^{(0)}].$$

For $\alpha > \lambda$ we may take $\mu_\alpha = (\alpha - \lambda)^2/a^2$ for every T, so the flow reaches $\mathcal{L} \leq \varepsilon$ by the flow time $\big(2\alpha T + \log(\mathcal{L}[s^{(0)}]/\varepsilon)\big)/(2\mu_\alpha)$, which is linear in T. Without weighting, (9) bounds $\mu_\star$ by a multiple of $e^{-2\lambda T}$ for an exponentially unstable system, so the rate guaranteed by the PL inequality decays exponentially in T. These are statements about the continuous flow; the number of steps of a numerical optimizer also depends on its step size.

The weighted norm on trajectories is needed as well as the weighted loss. If $\mathcal{L}_\alpha$ is minimized in the unweighted L^2 metric, the linearized problem $\min_\eta \frac{1}{2}\|W^{1/2}(r[s] + \mathcal{J}_s\eta)\|^2_{L^2}$, with $W := e^{-2\alpha t} \leq 1$ and $\mathcal{J}_s$ the linearized residual map of Appendix B.3, has PL constant $\sigma_{\min}(W^{1/2}\mathcal{J}_s)^2 \leq \sigma_{\min}(\mathcal{J}_s)^2$, which is no larger than without weighting.

Exact Newton updates. A Gauss–Newton step for $\mathcal{L}_\alpha$ minimizes the local model $\frac{1}{2}\|r[s] + \mathcal{J}_s\eta\|^2_{L^2_\alpha}$ over perturbations η with $\eta(0) = 0$. By variation of constants, $\mathcal{J}_s$ is a bijection from $\{\eta \in H^1 : \eta(0) = 0\}$ onto L^2, so $\eta_{\mathrm{N}} := -\mathcal{J}_s^{-1} r[s]$ makes the model vanish. It is therefore the minimizer for every $\alpha \geq 0$, and for any other positive weight. The transformed IVP has the linearized residual map $\tilde{\mathcal{J}}_{\tilde{s}} = E\mathcal{J}_s E^{-1}$ and the Newton step $-\tilde{\mathcal{J}}_{\tilde{s}}^{-1}\tilde{r}[\tilde{s}] = -E\mathcal{J}_s^{-1} r[s] = E\eta_{\mathrm{N}}$, so the two iterations coincide in exact arithmetic. The discrete DEER and quasi-DEER updates of Appendix K behave in the same way with $\mathbf{E} = \mathrm{diag}(e^{-\alpha t_n} I_D)$. The residual becomes $\mathbf{Er}$, the step $\mathbf{E}\Delta\mathbf{s}$, and the Jacobian $\mathbf{E}J\mathbf{E}^{-1}$, in which each $\partial_x g_n$ is multiplied by $e^{-\alpha h}$, an operation that commutes with taking its diagonal. From corresponding initial guesses, these methods therefore produce the same trajectories in the original coordinates, in exact arithmetic. Working in the transformed coordinates can still reduce intermediate magnitudes and the risk of overflow in an implementation, and a weighted stopping criterion places less emphasis on late-time errors.

Convergence radius of Newton's method. We now ask how close to the solution s^* a candidate trajectory must be for Newton's method to converge. Suppose (20) holds and $x \mapsto J(x,t)$ is L_J-Lipschitz for every t. Then the error $\xi := s - s^*$ of a candidate trajectory and the error ξ^+ after one Newton update satisfy

$$\|\xi^+\|_{L^\infty} \leq \frac{L_J G_T}{2}\,\|\xi\|^2_{L^\infty}, \tag{27}$$

with the constant G_T of Appendix B.3. Newton's method therefore converges quadratically from any initial guess with $\|s^{(0)} - s^*\|_{L^\infty} < \rho := 2/(L_J G_T)$. Because $1/G_T$ is the lower bound on $\sqrt{\mu_\star}$

in (21), the radius ρ is $2/L_J$ times that bound, so the same constant controls both the PL constant and the region in which Newton's method is guaranteed to converge. In CASE 1, $G_T \le a/c$ and $\rho \ge 2c/(aL_J)$ for every T, whereas for $\lambda > 0$ the radius decays like $e^{-\lambda T}$. For a linear IVP, $L_J = 0$ and one step suffices however small $\mu_\star$ is.

To derive (27), write the Newton update as $s^+ = s + \eta_{\mathrm{N}}$ with $\eta_{\mathrm{N}} = -\mathcal{J}_s^{-1} r[s]$ as above. Since $r[s^*] = 0$, a Taylor expansion of f around s gives $r[s] = \mathcal{J}_s \xi - R$, where the remainder $R := f(s,t) - f(s^*,t) - J(s,t)\,\xi$ satisfies $\|R(t)\| \le \frac{L_J}{2}\|\xi(t)\|^2$ at every t. The part of the error that is linear in ξ cancels in the update, which leaves $\xi^+ = \xi - \mathcal{J}_s^{-1}(\mathcal{J}_s\xi - R) = \mathcal{J}_s^{-1} R$. By the solution formula of Appendix B.3,

$$\xi^+(t) = \int_0^t \Phi_s(t,\tau)\, R(\tau)\, d\tau,$$

so the new error is the remainder propagated by the linearized dynamics. Because the bound on R is pointwise in time, we estimate this integral in the supremum norm. By (20), $\|\xi^+(t)\| \le \|R\|_{L^\infty}\, a \int_0^t e^{\lambda(t-\tau)}\, d\tau \le G_T \|R\|_{L^\infty}$, which gives (27).

The same computation applies to the transformed IVP. There the constant G_T improves to $a(1 - e^{-(\alpha-\lambda)T})/(\alpha - \lambda)$, the reciprocal of the bound (26), but the Lipschitz constant of the Jacobian grows:

$$\partial_x^2 \tilde{f}(x,t) = e^{\alpha t}\, \partial_x^2 f\big(e^{\alpha t} x, t\big), \qquad \tilde{L}_J = e^{\alpha T} L_J.$$

Thus the smaller G_T comes with a potentially larger remainder. A larger weighted PL constant alone does not give a larger Newton convergence region or fewer DEER iterations. This is consistent with the previous paragraph, in which the Newton iterates were the same in both coordinates.

Appendix D. Picard and Jacobi Updates

We derive the Picard and exact-flow Jacobi updates in Section 2.2 and prove convergence of the latter under CASE 1. Table 1 lists the updates of that section. Throughout, f is continuously differentiable, and the exact flows are assumed to exist on the intervals and for the initial states under consideration. We write $\phi(t, \tau; y)$ for the exact ODE solution at time t starting from y at time τ.

At optimization step i, let $s^{(i)}(t)$ be a differentiable interpolation of the current nodal guesses, with $s^{(i)}(0) = x_0$, and write $r^{(i)} = \dot{s}^{(i)} - f(s^{(i)}, t)$.

Table 1: Choices of A in the update $\delta\dot{s} = A\,\delta s - r$ of Section 2.2 (all with $B = I$), at optimization step i. With a constant α, the discrete analogue of the damped update is scale-ELK; LASER sets α at every step by (12).

Method	A	Linear in δs?	Parallelized by
Newton / DEER	$J(s^{(i)}, t)$	yes	scan
Damped / LASER	$J(s^{(i)}, t) - \alpha I$	yes	scan
Picard	0	yes	prefix sum
Exact-flow Jacobi	$\int_0^1 J(s^{(i)} + \theta\,\delta s, t)\, d\theta$	no	independent intervals

D.1. Derivation of Picard Iteration

Picking $A(s, \delta s, t) = 0$ and $B(s, \delta s, t) = I$ gives the correction

$$\delta \dot{s}^{(i)} = -r^{(i)}, \qquad \delta s^{(i)}(0) = 0.$$

Integrating and using $s^{(i)}(0) = x_0$ yields

$$\begin{aligned} s^{(i+1)}(t) &= s^{(i)}(t) + \delta s^{(i)}(t) = s^{(i)}(t) - \int_0^t r^{(i)}(\tau)\, d\tau \\ &= s^{(i)}(t) - \int_0^t \left[\dot{s}^{(i)}(\tau) - f\Big(s^{(i)}(\tau), \tau\Big)\right] d\tau \\ &= x_0 + \int_0^t f\Big(s^{(i)}(\tau), \tau\Big)\, d\tau, \end{aligned}$$

which is standard Picard iteration (cf. Teschl, 2012; Shih et al., 2023).

D.2. Derivation of the Exact-Flow Jacobi Correction

Choose

$$A(s, \delta s, t) = \int_0^1 J(s + \theta \delta s, t)\, d\theta, \qquad B(s, \delta s, t) = I. \tag{28}$$

By the fundamental theorem of calculus, these matrices satisfy

$$A(s, \delta s, t)\, \delta s = f(s + \delta s, t) - f(s, t).$$

On each interval $[t_{n-1}, t_n]$, independently solve

$$\delta \dot{s}_n^{(i)} = A\Big(s^{(i)}, \delta s_n^{(i)}, t\Big)\, \delta s_n^{(i)} - r^{(i)}, \qquad \delta s_n^{(i)}(t_{n-1}) = 0. \tag{29}$$

The corrected local trajectory then obeys

$$\begin{aligned} \frac{d}{dt}\left(s^{(i)} + \delta s_n^{(i)}\right) &= \dot{s}^{(i)} + f\Big(s^{(i)} + \delta s_n^{(i)}, t\Big) - f(s^{(i)}, t) - r^{(i)} \\ &= f\Big(s^{(i)} + \delta s_n^{(i)}, t\Big). \end{aligned}$$

Its value at t_{n-1} is $s^{(i)}(t_{n-1})$. Uniqueness of the local IVP therefore gives

$$s^{(i+1)}(t_n) = s^{(i)}(t_n) + \delta s_n^{(i)}(t_n) = \phi\Big(t_n, t_{n-1}; s^{(i)}(t_{n-1})\Big). \tag{30}$$

We recognize this as a nonlinear Jacobi iteration (cf. Ortega and Rheinboldt, 1970; Song et al., 2021; Gonzalez et al., 2026), where the solution ϕ of the differential equation on the interval $[t_{n-1}, t_n]$ serves as the update. The endpoint update is independent of the chosen interpolation. Adjacent corrected segments need not agree at their interfaces, since each interval starts from the old nodal value. The new endpoints define the next nodal iterate and can be interpolated before the next sweep. Because A depends on the unknown correction, each interval problem is nonlinear and equivalent to solving the original ODE on that interval.

D.3. Jacobi Convergence under Incremental Stability

Choose equally spaced times $t_n = nh$, with $Nh = T$. To cover both exact flows and numerical integrators, let g_n denote a generic discrete step map. The corresponding discrete trajectory satisfies

$$x_n = g_n(x_{n-1}), \qquad x_0 \in \mathbb{R}^D, \qquad n = 1, \dots, N. \tag{31}$$

For exact-flow Jacobi,

$$g_n(y) = \phi(t_n, t_{n-1}; y),$$

and $x_n = \phi(t_n, 0; x_0)$. In this case, h is simply the length of a time interval; no numerical approximation has been introduced.

For a trajectory guess with $s_0 = x_0$, write $R_n = s_n - g_n(s_{n-1})$. Linearizing the flow relation above gives the Newton correction

$$\Delta s_n = Dg_n(s_{n-1})\Delta s_{n-1} - R_n, \qquad \Delta s_0 = 0.$$

This has the same two ingredients as the continuous-time correction: propagate a correction from the preceding time point, and subtract the local residual. Newton-type methods such as DEER retain the propagated correction (Lim et al., 2024; Gonzalez et al., 2024). Jacobi drops it, giving

$$\Delta s_n = -R_n \qquad \Longrightarrow \qquad s_n^+ = s_n - R_n = g_n(s_{n-1}).$$

Thus each new state is obtained by evolving the preceding state from the *previous iterate* through one time interval:

$$s_n^{(i+1)} = g_n\left(s_{n-1}^{(i)}\right), \qquad s_0^{(i)} = x_0. \tag{32}$$

All intervals can be processed in parallel, and the update requires no Jacobians (Ortega and Rheinboldt, 1970; Song et al., 2021; Chartier and Philippe, 1993). For later use, write $\mathbf{s} = (s_1, \dots, s_N)$, $G_n(\mathbf{s}) = g_n(s_{n-1})$, and $\mathbf{r}(\mathbf{s}) = \mathbf{s} - \mathbf{G}(\mathbf{s})$. Then Jacobi is simply $\mathbf{s}^{(i+1)} = \mathbf{G}\left(\mathbf{s}^{(i)}\right)$, and its fixed points are exactly the trajectories satisfying (31).

Stability gives a simple explanation for fast convergence. Each sweep propagates the initial guess through one more interval. After i sweeps, an error has therefore evolved through i intervals, so stability of the flow reduces that error. We express this using the following bound.

Assumption 1 (Incremental exponential stability with overshoot) *Let $\Psi_{\ell,n} := g_{\ell+n} \circ \cdots \circ g_{\ell+1}$ denote the n-step flow map of (31). There exist $\kappa \geq 1$ and $\rho \in [0, 1)$, independent of N, such that for all $0 \leq \ell \leq N-1$, $1 \leq n \leq N-\ell$, and all $x, y \in \mathbb{R}^D$,*

$$\|\Psi_{\ell,n}(x) - \Psi_{\ell,n}(y)\| \leq \kappa\rho^n \|x - y\|.$$

CASE 1 implies the discrete stability bound. For the exact flow $\phi(t, \tau; y)$, the derivative with respect to y is the state-transition matrix along that solution. Hence CASE 1 gives

$$\|D_y\phi(t, \tau; y)\| \leq ae^{-c(t-\tau)}.$$

Integrating this derivative along the segment joining two initial states u and v yields

$$\begin{aligned} \|\phi(t, \tau; u) - \phi(t, \tau; v)\| &= \left\| \int_0^1 D_y\phi(t, \tau; v + \theta(u - v))(u - v)\, d\theta \right\| \\ &\leq ae^{-c(t-\tau)}\|u - v\|. \end{aligned} \tag{33}$$

The flow-composition property gives $\Psi_{\ell,n}(y) = \phi(t_{\ell+n}, t_\ell; y)$, so Assumption 1 holds with $\kappa = a$ and $\rho = e^{-ch}$. The overshoot a appears only once because the estimate applies to the complete flow over time nh. This argument needs the variational bound only along actual solution segments, which is weaker than CASE 1's requirement along every candidate trajectory.

Theorem 5 (Global convergence of nonlinear Jacobi) *Suppose Assumption 1 holds. Let* $\mathbf{s}^* = (x_1, \dots, x_N) \in \mathbb{R}^{ND}$ *be the exact trajectory of* (31)*, and let* $\{\mathbf{s}^{(i)}\}$ *be generated by* (32)*. Then, for every* $i \geq 0$*,*

$$\|\mathbf{s}^{(i)} - \mathbf{s}^*\|_\infty \leq \kappa\rho^i \|\mathbf{s}^{(0)} - \mathbf{s}^*\|_\infty. \tag{34}$$

In particular, $\mathbf{s}^{(i)} \to \mathbf{s}^*$ *geometrically in the block maximum norm* $\|\mathbf{s}\|_\infty = \max_{1\leq n\leq N} \|s_n\|$.

Proof Each Jacobi sweep moves information forward by one interval. The first sweep sets $s_1^{(1)} = g_1(x_0) = x_1$; the next makes $s_2^{(2)} = x_2$, and so on. Thus the first i states are exact after i sweeps. At a later time point, repeatedly substituting the Jacobi update gives

$$s_m^{(i)} = \begin{cases} x_m, & m \leq i, \\ \Psi_{m-i,i}\left(s_{m-i}^{(0)}\right), & m > i, \end{cases} \tag{35}$$

where $\Psi_{\ell,i} = g_{\ell+i} \circ \cdots \circ g_{\ell+1}$ and $\Psi_{\ell,0} = \mathrm{Id}$. In words, the state at time m is the initial guess from i intervals earlier, propagated through those i intervals.

The exact trajectory passes through the same maps: $x_m = \Psi_{m-i,i}(x_{m-i})$. Therefore, for $m > i \geq 1$,

$$\begin{aligned} \|s_m^{(i)} - x_m\| &= \left\|\Psi_{m-i,i}(s_{m-i}^{(0)}) - \Psi_{m-i,i}(x_{m-i})\right\| \\ &\leq \kappa\rho^i \|s_{m-i}^{(0)} - x_{m-i}\| \\ &\leq \kappa\rho^i \|\mathbf{s}^{(0)} - \mathbf{s}^*\|_\infty. \end{aligned}$$

For $m \leq i$ the error is zero, so taking the maximum over m proves (34). The case $i = 0$ follows from $\kappa \geq 1$. For exact flow maps, the composition above is simply the ODE flow over time ih, and the factor $\kappa\rho^i$ becomes ae^{-cih}. ■

Remark 4 *For* $0 < \rho < 1$*,* (34) *shows that it suffices to take*

$$i \geq \frac{\log\left(\kappa \|\mathbf{s}^{(0)} - \mathbf{s}^*\|_\infty / \epsilon\right)}{\log(1/\rho)} \tag{36}$$

iterations to reach error at most $\epsilon > 0$*, unless the initial guess already meets this tolerance. If* $\rho = 0$*, one sweep is sufficient. For exact flow maps the decay factor is* e^{-cih}*: at fixed* h*, initial-error bound, and tolerance, the iteration bound is independent of the horizon* $T = Nh$*. Refining the grid at fixed* T *decreases* h *and weakens the per-sweep factor* e^{-ch}*. More generally, the horizon independence holds whenever* κ*,* ρ*, the initial-error bound, and the tolerance remain fixed. The first* i *states are also exact after* i *sweeps, so Jacobi terminates after at most* N *sweeps even without stability; stability replaces this worst case with the bound above.*

Remark 5 *It is possible to find a metric in which the Jacobi steps are contractive at every time point; see Appendix L.*

Numerical flow maps. In practice, the step map g_n is supplied by a numerical integrator approximating $\phi(t_n, t_{n-1}; \cdot)$. The same theorem applies whenever these maps satisfy Assumption 1, with $\mathbf{s}^*$ now denoting the numerical trajectory. For a uniformly contracting ODE with rate c, backward Euler inherits the bound with $\kappa = 1$ and $\rho(h) = 1/(1 + ch)$. Forward Euler also inherits contraction for sufficiently small h under a uniform Lipschitz bound, with $\rho(h) = 1 - ch + O(h^2)$ (Davydov et al., 2024). A sufficient condition that allows individual steps to be expansive is given in Appendix L. In Appendix I.1, we observe that at fixed tolerance the Jacobi iteration count stays essentially constant as N ranges from 10^3 to 10^7.

Appendix E. The LASER Update

The stability bounds of Section 2 concern perturbation growth along candidate trajectories, so a stable solution alone does not guarantee a well-conditioned problem at the initial guess. For a discretized system $x_n = g_n(x_{n-1})$, $n = 1, \ldots, N$, with $T = N\Delta t$, let X be the current iterate with $X_0 = x_0$ fixed, $J_n = Dg_n(X_{n-1})$ its step Jacobians and $r_n = X_n - g_n(X_{n-1})$ its residual. The DEER correction (Appendix K) solves $\Delta X_n = J_n \, \Delta X_{n-1} - r_n$ with $\Delta X_0 = 0$, so it propagates through products of these Jacobians. We measure their growth by the finite-time Lyapunov exponent of the iterate,

$$\lambda_h(X) = \frac{1}{N\Delta t} \log \left\| J_N \cdots J_1 \right\|. \tag{37}$$

The exponent is computed in $O(\log N)$ depth by a parallel reduction of the Jacobian product, in which every partial product is renormalized and the logarithm of its scale is accumulated, so that it does not overflow (cf. Heinsen and Kozachkov, 2025).

The damped update of Section 2.2, $\delta\dot{s} = (-\alpha I + J)\, \delta s - r$, multiplies the transition matrix of the undamped update by $e^{-\alpha \Delta t}$ over one time step. LASER applies the same factor to the step Jacobians, with a rate γ set by the exponent:

$$\Delta X_n = e^{-\gamma \Delta t} J_n \, \Delta X_{n-1} - r_n, \qquad \gamma = \max\{0, \lambda_h(X) + \epsilon\}, \qquad \epsilon = 1/T. \tag{38}$$

The recurrence is solved by the same parallel scan, and $X \leftarrow X + \Delta X$. Setting $\gamma = 0$ gives DEER, and $\gamma \to \infty$ gives the Jacobi update $X_n \leftarrow g_n(X_{n-1})$. For every γ, $\Delta X = 0$ if and only if $r = 0$, so the trajectory of the original system is still the only fixed point. With the γ of (38), the damped product $e^{-\gamma T} J_N \cdots J_1$ has norm at most e^{-1}.

Safeguards. This bounds only the end-to-end product of the linearization, so every step is checked before it is accepted. The first trial of an iteration uses the larger of the γ of (38) and a quarter of the rate accepted at the previous iteration. The trial step is accepted if its residual $\max_n \|r_n\|_\infty$ is finite, at most twice the current one, and at most 100 times the smallest so far. Otherwise $\gamma \leftarrow \max\{4\gamma, \epsilon\}$ and the step is recomputed; after 8 rejections a Jacobi step is taken. The experiments of Section 3 use these settings.

Appendix F. Jacobi-Specific Bounds for the Discretization–Parallelization Tradeoff

In this appendix, we specialize the error decomposition in Remark 1 to the case where the discrete problem is solved by a finite number of Jacobi iterations. The main text only uses the abstract decomposition

$$E(M, h; T, x_0) \le E_M^{\mathrm{P}}(h; T, x_0) + E^{\mathrm{S}}(h; T, x_0),$$

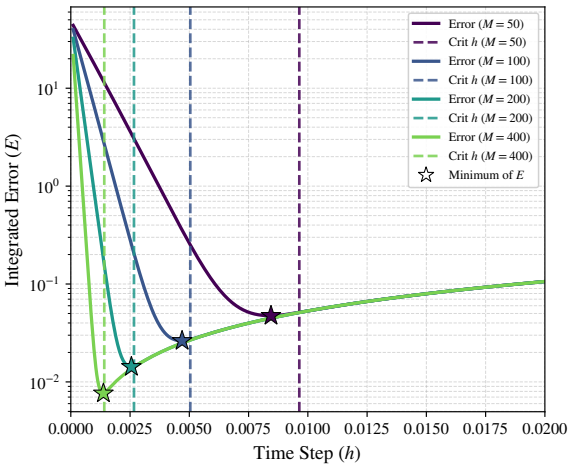


Figure 4: Total error for the system $\dot{x} = -20x(t) + 20\sin(2t)$, $x(0) = 10$, $T = 5$, as the step size varies for different values of the parallel iteration budget M. Dashed verticals mark the step-size estimate (46). Each curve has an interior minimum. For large h the discretization error dominates, for small h the M Jacobi sweeps have not converged, and the predicted step size follows the empirical minimizer as M varies. The predicted step sizes lie 2.6–14.2% above the empirical minimizers, with the gap shrinking as M grows. ★: the minimum of each curve. The error at the estimated step size exceeds this minimum by at most 9%.

where E^{S} is the discretization error of the numerical method and E_M^{P} is the additional error incurred by solving the discrete problem only approximately. Here we give one concrete bound on E_M^{P}, and use it to obtain the step-size selection rule used in Figure 4.

Fix $T > 0$, an integer $K \geq 1$, and $M \in \mathbb{N}_0$. Consider step maps $g_k : \mathbb{R}^D \to \mathbb{R}^D$ with constant step size $h = T/K$,

$$s_k = g_k(s_{k-1}), \qquad k = 1, \ldots, K, \tag{39}$$

with initial condition $s_0 = x_0$. The method may be explicit or implicit: for an implicit method, g_k denotes the solved one-step map, for instance $g_k = \big(\mathrm{Id} - hf(\cdot, t_k)\big)^{-1}$ for backward Euler. This is the step map of (31), so the Jacobi iteration (32) applies unchanged. Let $\mathbf{s}^\star = (s_1^\star, \ldots, s_K^\star)$ denote the exact solution of the resulting discrete problem, and let $\mathbf{p}^{(M)} = (p_1^{(M)}, \ldots, p_K^{(M)})$ denote the approximate trajectory obtained after M Jacobi iterations. The first states are fixed, $s_0^\star = p_0^{(M)} = x_0$, and $\hat{s}$, $\hat{p}_M$ are the interpolants of Remark 1. Trajectories are measured in the block maximum norm $\|\mathbf{p}\|_\infty = \max_{1 \leq k \leq K} \|p_k\|$ of Theorem 5.

Lemma 6 (Upper bound on parallelization error after M Jacobi steps) *Let the Jacobi iterates be written as*

$$\mathbf{p}^{(m+1)} = \mathbf{G}(\mathbf{p}^{(m)}), \qquad \mathbf{G}(\mathbf{p})_k = g_k(p_{k-1}), \quad k = 1, \ldots, K, \qquad p_0^{(m)} = x_0 \text{ for all } m,$$

and suppose that the iterates converge geometrically:

$$\left\|\mathbf{p}^{(m)} - \mathbf{s}^\star\right\|_\infty \leq \rho(h)^m \left\|\mathbf{p}^{(0)} - \mathbf{s}^\star\right\|_\infty, \qquad 0 \leq \rho(h) < 1. \tag{40}$$

This holds, for example, if every step map g_k is $\rho(h)$-Lipschitz, since then $\mathbf{G}$ is a $\rho(h)$-contraction in $\|\cdot\|_\infty$ with fixed point $\mathbf{s}^\star$. This is Theorem 5 with $\kappa = 1$. Then the parallelization error satisfies

$$E_M^{\mathrm{P}}(h; T, x_0) \le \max\{T - Mh, 0\}\, \rho(h)^M \left\|\mathbf{p}^{(0)} - \mathbf{s}^\star\right\|_\infty . \tag{41}$$

The factor $\rho(h)^M$ comes from the contraction of the Jacobi iteration, while the prefactor $\max\{T - Mh, 0\}$ is the length of the time interval over which the finite Jacobi iteration has not yet propagated exact information. Note that this prefactor stays bounded as $h \to 0$, whereas $\max\{T/h - M, 0\}$ would diverge. If instead the iterates satisfy the bound (34) of Theorem 5 with overshoot $\kappa \ge 1$ in the same trajectory norm, the same proof gives (41) with an extra factor κ. The precise form of $\rho(h)$ depends on the discretization scheme. For example, for forward and backward Euler applied to a vector field that satisfies assumption (A1) of Appendix F.2, the corresponding contraction factors are

$$\rho_{\mathrm{FE}}(h) = e^{-ch} + e^{\ell_x h} - 1 - \ell_x h, \qquad \rho_{\mathrm{BE}}(h) = \frac{1}{1 + ch}, \tag{42}$$

respectively (Davydov et al., 2024, Lemma 19(i) and Lemma 22). The factor $\rho_{\mathrm{BE}}(h)$ is less than 1 for every $h > 0$, while $\rho_{\mathrm{FE}}(h) < 1$ only for h small enough. Under the assumptions of Lemma 7, with d defined just before it, for both schemes the discretization error satisfies

$$E^{\mathrm{S}}(h; T, x_0) \le \frac{Td}{2c}\, h + O(h^2). \tag{43}$$

Thus the sequential-error bound tends to zero as $h \to 0$, whereas both Euler factors tend to one. At fixed M, the resulting parallel-error bound need not decrease with h. These bounds motivate a tradeoff; they do not prove monotonicity of either actual error. Table 2 summarizes both schemes.

Table 2: Forward and backward Euler as Jacobi step maps. The columns give the step map g_k of (39), the bound on the sequential error E^{S} from Lemma 7, and the contraction factor $\rho(h)$ of the induced Jacobi iteration for a vector field that satisfies assumption (A1). The factors are from Davydov et al. (2024, Lemma 19(i) and Lemma 22).

Method	Step map $g_k(s)$	Sequential error bound	Jacobi factor $\rho(h)$
Forward Euler	$s + hf(s, t_{k-1})$	$\frac{Td}{2c}\, h + O(h^2)$	$e^{-ch} + e^{\ell_x h} - 1 - \ell_x h$
Backward Euler	$\big(\mathrm{Id} - hf(\cdot, t_k)\big)^{-1}(s)$	$\frac{Td}{2c}\, h + O(h^2)$	$\frac{1}{1 + ch}$

Combining (41) with an upper bound $B^{\mathrm{S}}(h; T, x_0)$ on the discretization error gives the Jacobi-specific upper-bound minimization problem

$$h^\star \in \operatorname*{argmin}_{h>0,\ \rho(h)<1} \max\{T - Mh, 0\}\, \rho(h)^M \|\mathbf{p}^{(0)} - \mathbf{s}^\star\|_\infty + B^{\mathrm{S}}(h; T, x_0). \tag{44}$$

The minimization is over $h > 0$. In practice, the minimizer is then rounded to the nearest admissible step size T/K, $K \in \mathbb{N}$. For forward and backward Euler, B^{S} is the right-hand side of the bound on E^{S} in the corresponding statement of Lemma 7, (49) or (50). In practice, $\mathbf{s}^\star$ is not known a priori, since it is the discrete trajectory we are trying to compute. A simple choice is to initialize the trajectory by the constant path $\mathbf{p}^{(0)} \equiv x_0$. Since neither the actual error nor $\mathbf{s}^\star$ is available, we use two computable

choices based on a leading-order surrogate of (44). If $\|f(x,t)\| \le \bar{v}$ for all $t \in [0,T]$ and all x in a set that contains the discrete trajectory $\mathbf{s}^\star$, a bound that assumption (A2) does not provide, then, since $\|s^\star_{k+1} - s^\star_k\| \le h\bar{v}$ for both Euler schemes, $\|\mathbf{p}^{(0)} - \mathbf{s}^\star\|_\infty \le T\bar{v}$. Assume $M \ge 1$, $\bar{v} > 0$ and $d > 0$, and fix a compact interval $I \subset (0,\infty)$ on which ρ is continuous and $\rho(h) < 1$. We use

$$h^\star_{\mathrm{asy}} = \frac{1}{cM}\log\Big(\frac{2c^2MT\bar{v}}{d}\Big), \qquad h^\star_{\mathrm{num}} \in \operatorname*{argmin}_{h\in I} \max\{T - Mh, 0\}\,\rho(h)^M + \frac{d}{2c\bar{v}}\,h. \tag{45}$$

The minimum over I is attained since the objective is continuous. Here $h^\star_{\mathrm{asy}}$ is an asymptotic stationary-point estimate for the approximate objective, derived in Appendix F.3 for small h and large T (precisely, $cT \gg 1$ and $Mh \ll T$). It is positive when $2c^2MT\bar{v} > d$, and for forward Euler it must also satisfy $\rho_{\mathrm{FE}}(h^\star_{\mathrm{asy}}) < 1$. Both hold in the example below. The estimate $h^\star_{\mathrm{num}}$ minimizes numerically the leading-order approximation of (44). In that approximation, $\|\mathbf{p}^{(0)} - \mathbf{s}^\star\|_\infty$ is replaced by its bound $T\bar{v}$, the bound B^{S} by its leading term $\frac{Td}{2c}h$, and the objective is divided by $T\bar{v}$. In the experiments, we use their midpoint as a heuristic step-size estimate,

$$h_{\mathrm{mid}} := \frac{h^\star_{\mathrm{asy}} + h^\star_{\mathrm{num}}}{2}. \tag{46}$$

In practice, T/h_{mid} is rounded to the nearest integer K. We do not claim that $h^\star_{\mathrm{asy}}$ and $h^\star_{\mathrm{num}}$ bracket the minimizer of the actual error. In the numerical search for (47), both lie at or above the empirical minimizer, by at most 20%. Their relative separation, $h^\star_{\mathrm{num}}/h^\star_{\mathrm{asy}} - 1$, is at most 10%.

Figure 4 uses the scalar contracting system

$$\dot{x} = -20x(t) + 20\sin(2t), \qquad x(0) = 10, \tag{47}$$

over the horizon $T = 5$. We discretize the system using forward Euler and solve the resulting discrete problem using $M \in \{50, 100, 200, 400\}$ Jacobi iterations. The interval $X = [-10, 10]$ is forward invariant for (47) and, for $h \le 1/11$, for its forward Euler discretization: the vector field points inward at both endpoints, and a forward Euler step satisfies $|x_{k+1}| \le 10|1 - 20h| + 20h \le 10$ for $0 < h \le 1/11$. The assumptions (A1)–(A2) hold with $c = \ell_x = 20$, $\ell_t = 40$ and $\bar{v} = 220$, so that $d = \ell_x\bar{v} + \ell_t = 4{,}440$, and $|f(x,t)| \le \bar{v}$ for every $x \in X$. For each fixed M, we vary the step size over a grid of values $h = T/K$, $K \in \mathbb{N}$, in $[10^{-4}, 0.02]$, refined locally around the smallest error, compute the approximate trajectory, and evaluate the total error

$$E(M, h; T, x_0) = \int_0^T \|\hat{p}_M(t) - \phi(t; x_0)\|\, dt.$$

For (45), I is the interval of stable $h < T/M$. The resulting curves are shown in Figure 4. The vertical dashed lines correspond to the step-size estimates (46). In these computations, the smallest error occurs at an intermediate step size. For larger h the discretization error dominates, and for smaller h the Jacobi parallelization error dominates.

F.1. Proof of Lemma 6

Proof Write $e_k := p^{(M)}_k - s^\star_k$ for $k = 0, \dots, K$. Since $\mathbf{G}(\mathbf{p})_k$ depends only on p_{k-1} and $p^{(m)}_0 = x_0 = s^\star_0$, induction on m gives $p^{(m)}_j = s^\star_j$ for $0 \le j \le \min\{m, K\}$: each sweep makes one more state exact, as in the proof of Theorem 5. Hence $e_j = 0$ for $j \le \min\{M, K\}$, so $\hat{p}_M - \hat{s}$ vanishes on

$[0, \min\{Mh, T\}]$. On each interval $[t_k, t_{k+1}]$, the difference $\hat{p}_M(t) - \hat{s}(t)$ is a convex combination of e_k and e_{k+1}, so $\|\hat{p}_M(t) - \hat{s}(t)\| \le \max_i \|e_i\| = \left\|\mathbf{p}^{(M)} - \mathbf{s}^\star\right\|_\infty$. Therefore

$$E_M^{\mathrm{P}}(h; T, x_0) = \int_{\min\{Mh,T\}}^{T} \|\hat{p}_M(t) - \hat{s}(t)\| \, dt \le \max\{T - Mh, 0\} \left\|\mathbf{p}^{(M)} - \mathbf{s}^\star\right\|_\infty$$
$$\overset{(40)}{\le} \max\{T - Mh, 0\} \, \rho(h)^M \left\|\mathbf{p}^{(0)} - \mathbf{s}^\star\right\|_\infty,$$

which is the bound (41). If $M \ge K$, the error is zero by prefix propagation alone; this exact completion does not require contraction. ■

F.2. Estimates of the Sequential Error

We bound the sequential error E^{S} of forward and backward Euler. The step map of either method is a forward step or a resolvent of the operator $-f(\cdot, t)$, and Davydov et al. (2024) bound the Lipschitz constants of both. The errors at the nodes accumulate as in the convergence proof of Hairer et al. (1993, Section II.3), and a bound on linear interpolation carries them from the nodes to the interpolant. Throughout this subsection, $f : \mathbb{R}^D \times [0, T] \to \mathbb{R}^D$ and $x_0 \in \mathbb{R}^D$, with a fixed state norm. We assume that:

- (A1) (*contractivity*) for every $t \in [0, T]$, the map $f(\cdot, t)$ is globally Lipschitz on $\mathbb{R}^D$ with constant ℓ_x and strongly infinitesimally contracting with rate $c > 0$, that is, $\mathrm{osL}\big(f(\cdot, t)\big) \le -c$, with respect to a norm $\|\cdot\|$; and
- (A2) (*bounded velocity*) the exact solution satisfies $\|f(\phi(t; x_0), t)\| \le \bar{v}$ for every $t \in [0, T]$, and for every state x the map $f(x, \cdot)$ is Lipschitz with constant ℓ_t.

Under (A1), the backward Euler solved map is defined uniquely on all of $\mathbb{R}^D$ for every $h > 0$: apply Davydov et al. (2024, Theorem 17 and Lemma 22) to the continuous strongly monotone operator $-f(\cdot, t_k)$. If f is continuously differentiable in the state, (A1) gives $\mu(D_x f(x, t)) \le -c$ for all x and t (Davydov et al., 2024, Lemma 7). By Coppel's inequality the decay estimate (6) of CASE 1, measured in the norm of (A1), then holds along every trajectory with $a = 1$, so c is the same rate as in Section 2.1. We set $d := \ell_x \bar{v} + \ell_t$. The velocity $\phi'(t; x_0) = f(\phi(t; x_0), t)$ is Lipschitz with constant d: for all $t, \tau \in [0, T]$,

$$\left\|\phi'(t; x_0) - \phi'(\tau; x_0)\right\| \overset{\text{(A1), (A2)}}{\le} \ell_x \|\phi(t; x_0) - \phi(\tau; x_0)\| + \ell_t |t - \tau| \overset{\text{(A2)}}{\le} d \, |t - \tau|. \tag{48}$$

On the grid $t_k = kh$, $h = T/K$, we write $\phi_k := \phi(t_k; x_0)$ for the *exact states* and $\varepsilon_k := s_k - \phi_k$ for the *nodal errors* of a numerical sequence $\{s_k\}_{k=0}^K$ with $s_0 = x_0$.

Lemma 7 (Sequential error of the Euler methods) *Suppose (A1)–(A2) hold, and let $\{s_k\}_{k=0}^K$ be the forward or the backward Euler sequence with step size $h = T/K$ and interpolant $\hat{s}$. Then the following statements hold:*

(i) (forward Euler) *if $\rho := \rho_{\mathrm{FE}}(h) < 1$, then $\|\varepsilon_k\| \le h^2 d / \big(2(1 - \rho)\big)$ for every k, and*

$$E^{\mathrm{S}}(h; T, x_0) \le \frac{Td}{2c}\Big(\frac{c\,h^2}{1 - \rho} + \frac{c\,h^2}{6}\Big) = \frac{Td}{2c}\Big(h + \Big(\frac{2c}{3} + \frac{\ell_x^2}{2c}\Big)h^2 + O(h^3)\Big); \text{ and} \tag{49}$$

(ii) (backward Euler) *for every* $h > 0$, $\|\varepsilon_k\| \le dh/(2c)$ *for every* k, *and*

$$E^{\mathrm{S}}(h;T,x_0) \le \frac{Td}{2c}\Big(h + \frac{c}{6}h^2\Big). \tag{50}$$

Proof Both statements use two estimates, on the local errors and on the interpolation error. The local errors of forward and backward Euler along the exact solution are

$$r_k^{\mathrm{FE}} := \phi_{k+1} - \phi_k - hf(\phi_k, t_k) = \int_{t_k}^{t_{k+1}} \big(\phi'(\tau) - \phi'(t_k)\big)\, d\tau,$$
$$r_k^{\mathrm{BE}} := \phi_{k+1} - \phi_k - hf(\phi_{k+1}, t_{k+1}) = \int_{t_k}^{t_{k+1}} \big(\phi'(\tau) - \phi'(t_{k+1})\big)\, d\tau,$$

where $\phi'(\tau)$ stands for $\phi'(\tau; x_0)$. By the velocity bound (48), the integrands are at most $d(\tau - t_k)$ and $d(t_{k+1} - \tau)$ in norm, so

$$\big\|r_k^{\mathrm{FE}}\big\| \le \frac{h^2}{2}d, \qquad \big\|r_k^{\mathrm{BE}}\big\| \le \frac{h^2}{2}d. \tag{51}$$

On $[t_k, t_{k+1}]$, let $\hat\phi$ be the linear interpolant of ϕ_k and ϕ_{k+1}. Then $\hat s - \hat\phi$ is the linear interpolant of ε_k and ε_{k+1}, and

$$\phi(t;x_0) - \hat\phi(t) = \frac{1}{h}\int_{t_k}^{t}\int_{t}^{t_{k+1}} \big(\phi'(\tau) - \phi'(\sigma)\big)\, d\sigma\, d\tau,$$

which gives us

$$\Big\|\phi(t;x_0) - \hat\phi(t)\Big\| \overset{(48)}{\le} \frac{d}{2}(t - t_k)(t_{k+1} - t).$$

Integrating over $[t_k, t_{k+1}]$ and summing over the $K = T/h$ intervals gives

$$E^{\mathrm{S}}(h;T,x_0) \le \sum_{k=0}^{K-1}\Big(\frac{h}{2}\big(\|\varepsilon_k\| + \|\varepsilon_{k+1}\|\big) + \frac{dh^3}{12}\Big) \le T \max_{0\le k\le K}\|\varepsilon_k\| + \frac{Tdh^2}{12}. \tag{52}$$

Regarding statement (i), the forward Euler step map $g_{k+1}(x) = x + hf(x, t_k)$ is the forward step operator $\mathrm{Id} - hF$ of $F = -f(\cdot, t_k)$. By (A1), F is Lipschitz with constant ℓ_x and strongly monotone with parameter c (Davydov et al., 2024, Remark 13). Davydov et al. (2024, Lemma 19(i)) show that then g_{k+1} is Lipschitz with constant $\rho_{\mathrm{FE}}(h) = \rho$. Since $s_{k+1} = g_{k+1}(s_k)$ and $\phi_{k+1} = g_{k+1}(\phi_k) + r_k^{\mathrm{FE}}$,

$$\|\varepsilon_{k+1}\| \le \|g_{k+1}(s_k) - g_{k+1}(\phi_k)\| + \big\|r_k^{\mathrm{FE}}\big\| \overset{(51)}{\le} \rho\,\|\varepsilon_k\| + \frac{h^2}{2}d.$$

Because $\varepsilon_0 = 0$ and $\rho < 1$, induction on k gives $\|\varepsilon_k\| \le \frac{h^2 d}{2}\sum_{i=0}^{k-1}\rho^i \le \frac{h^2 d}{2(1-\rho)}$. Inserting this into the interpolation bound (52) gives (49). The expansion uses $1 - \rho_{\mathrm{FE}}(h) = ch - \frac{1}{2}(c^2 + \ell_x^2)h^2 + O(h^3)$.

Regarding statement (ii), the backward Euler step map $g_{k+1} = \big(\mathrm{Id} - hf(\cdot, t_{k+1})\big)^{-1}$ is the resolvent $(\mathrm{Id} + hF)^{-1}$ of $F = -f(\cdot, t_{k+1})$. Since F is strongly monotone with parameter c, Davydov et al. (2024, Lemma 22) show that g_{k+1} is Lipschitz with constant $1/(1+ch)$. Since $s_{k+1} = g_{k+1}(s_k)$ and $\phi_{k+1} = g_{k+1}(\phi_k + r_k^{\mathrm{BE}})$,

$$\|\varepsilon_{k+1}\| \le \frac{\|\varepsilon_k\| + \big\|r_k^{\mathrm{BE}}\big\|}{1+ch} \overset{(51)}{\le} \frac{1}{1+ch}\Big(\|\varepsilon_k\| + \frac{h^2}{2}d\Big).$$

Because $\varepsilon_0 = 0$, induction on k gives $\|\varepsilon_k\| \le \frac{h^2 d}{2} \sum_{i=1}^{k} (1+ch)^{-i} \le \frac{h^2 d}{2} \cdot \frac{1}{ch} = \frac{dh}{2c}$. Inserting this into the interpolation bound (52) gives (50). ∎

F.3. Estimate of Critical Step Size

We now simplify the objective of (44). Recall $d = \ell_x \bar{v} + \ell_t$ from Appendix F.2. If $\|f(x,t)\| \le \bar{v}$ along the discrete trajectory $\mathbf{s}^\star$, as in the example (47), then $\left\|\mathbf{p}^{(0)} - \mathbf{s}^\star\right\|_\infty \le T\bar{v}$. Since $\rho_{\mathrm{FE}}(h) = 1 - ch + \frac{1}{2}(c^2 + \ell_x^2)h^2 + O(h^3)$ and $\rho_{\mathrm{BE}}(h) = (1+ch)^{-1}$, both methods satisfy $\log \rho(h) = -ch + \beta h^2 + O(h^3)$, with $\beta = \ell_x^2/2$ for forward Euler and $\beta = c^2/2$ for backward Euler. Hence

$$\rho(h)^M = e^{M \log \rho(h)} = e^{-cMh}\, e^{\beta M h^2 + O(Mh^3)}.$$

A binomial expansion of $\rho(h)^M$ in powers of h would require $cMh \ll 1$. This fails at the optimum, where $cMh^\star_{\mathrm{asy}} = \log\bigl(2c^2 MT\bar{v}/d\bigr)$ is about 9 to 11 in the example of (47). Replacing $\rho(h)^M$ by e^{-cMh} changes the exponent $-cMh$ by $\beta Mh^2 + O(Mh^3)$, a relative change of order $\beta h/c$, which is small even where βMh^2 is not. Using it, together with the leading term $\frac{Td}{2c}h$ of the sequential-error bound, the objective of (44) for $h < T/M$ is approximated by the function

$$\psi(h) := T\bar{v}\,(T - Mh)\, e^{-cMh} + \frac{Td}{2c}h.$$

Setting $\psi'(h) = 0$ yields the stationarity condition $M\bar{v}\, e^{-cMh}\bigl(1 + c(T - Mh)\bigr) = d/(2c)$. In the regime $cT \gg 1$ and $Mh \ll T$, we have $1 + c(T - Mh) \approx cT$, so $e^{-cMh} \approx d/(2c^2 MT\bar{v})$, and taking logarithms gives the closed form $h^\star_{\mathrm{asy}}$ in (45). On $(0, T/M)$ the function ψ is strictly convex, since $\psi''(h) = cM^2 T\bar{v}\, e^{-cMh}\bigl(2 + c(T - Mh)\bigr) > 0$. Hence, whenever $\psi'(0) < 0 < \psi'(T/M)$, the stationary point is the unique minimizer of ψ on that interval. In the example of (47), $h^\star_{\mathrm{asy}}$ lies within 1% of the exact stationary point for $M \in \{50, 100, 200, 400\}$, where $Mh \approx 0.1\,T$. The factor $\rho(h)^M$ itself is approximated only roughly for small M: at $h^\star_{\mathrm{asy}}$, the term $\beta Mh^2 = \ell_x^2 Mh^2/2$ decreases from 0.85 for $M = 50$ to 0.16 for $M = 400$, and $\rho_{\mathrm{FE}}(h)^M / e^{-cMh}$ is about 2.9, 1.7, 1.35 and 1.18 for $M = 50$, 100, 200, 400. What matters for the minimizer is the exponent. Keeping the term βMh^2 in the stationarity condition, with the same approximations as above, gives $cMh(1 - \beta h/c) \approx cMh^\star_{\mathrm{asy}}$, that is, $h \approx h^\star_{\mathrm{asy}}(1 + \beta h/c)$ to first order. In the example, $\beta h/c = \ell_x^2 h/(2c)$ is about 0.09, 0.05, 0.03 and 0.01, and $h^\star_{\mathrm{num}}$ in (45), which keeps $\rho(h)^M$ exact, lies above the stationary point of ψ by about 11%, 5%, 2% and 1%, respectively. This shift is the source of the gap of at most 10% between the two estimates, and the reason we also use $h^\star_{\mathrm{num}}$. At fixed T, the condition $Mh \ll T$ also limits the logarithmic estimate as M grows, since $Mh^\star_{\mathrm{asy}}$ grows like $\log(M)/c$. The objective that defines $h^\star_{\mathrm{num}}$ is not an upper bound on the objective of (44): replacing the unknown initial error by $T\bar{v}$ increases one term, while by Lemma 7 the full bound B^{S} exceeds its leading term $\frac{Td}{2c}h$ by a term of order h^2.

Appendix G. Training ELCD on Long Rollouts

This appendix gives the details of the ELCD experiment in Section 3.1. We describe the two tasks (Appendix G.1) and the model (Appendix G.2), and explain why the PiT solvers do not see a contracting system even though ELCD is contracting by construction (Appendix G.3). We then give the two losses and the metrics (Appendix G.4), the solvers and the timing protocol (Appendix G.5), and the details of Figure 2 (Appendix G.6). The last three subsections report further results on chain8z (Appendix G.7), the pendulum (Appendix G.8), and the original, shorter tasks of Jaffe et al. (2024) (Appendix G.9).

G.1. Tasks

Chained LASA shapes (chain8z). The LASA dataset (Khansari-Zadeh and Billard, 2011) contains 7 demonstrations of each of 30 planar handwriting shapes, each about 1,000 samples long. To obtain a long and complex motion, we resample eight shapes[5] to $\Delta t = 0.005\,\mathrm{s}$ and join them by short straight segments, the i-th demonstration of every shape into the i-th chained demonstration, giving 7 chained demonstrations of $N = 7{,}689$ samples (38.4 s) each. The order and vertical offsets of the shapes are chosen so that no chained demonstration crosses itself, since an autonomous vector field cannot follow a self-crossing path. As in Jaffe et al. (2024), we train on the first 3 demonstrations, hold out the other 4, and divide positions by the per-coordinate standard deviation of the training demonstrations.

Eight-link pendulum. The eight-link pendulum data of Jaffe et al. (2024) have 1,000 samples at $\Delta t = 0.1\,\mathrm{s}$. We fit a damped point-mass model to these data and re-simulate it at $\Delta t = 0.005\,\mathrm{s}$ for $N = 5{,}000$ samples (25 s), from 7 initial conditions drawn as in their data, 3 for training and 4 held out. The state is $(q, \dot{q}) \in \mathbb{R}^{16}$ and the goal is the rest state.

G.2. Model

We use the architecture of Jaffe et al. (2024), ported from their PyTorch code to JAX. The latent dynamics are (14) with

$$A(z) = -P_s(z)^\top P_s(z) - \alpha I + P_a(z) - P_a(z)^\top, \tag{53}$$

where $P_s(z), P_a(z) \in \mathbb{R}^{D\times D}$ are the outputs of two networks with one hidden layer of 16 $\tanh$ units, and $\alpha = 0.01$. Then $A + A^\top \preceq -2\alpha I$ for any weights. The diffeomorphism φ is their normalizing flow built from rational-quadratic spline couplings (Durkan et al., 2019), and the goal is $z^\star = \varphi(x_{\mathrm{eq}})$, with x_{eq} the common endpoint of the demonstrations. The model has 9,608 parameters on chain8z and 32,428 on the pendulum.

A rollout is solved in latent coordinates: $z_0 = \varphi(x_0)$, fourth-order Runge–Kutta (RK4) steps of (14) at the demonstration's Δt, and $x_n = \varphi^{-1}(z_n)$. The flow enters only at the two ends, so the PiT solvers see only the Jacobians of the latent RK4 step.

G.3. Contraction of ELCD

Along any solution of (14),

$$\tfrac{d}{dt}\,\tfrac{1}{2}\|z - z^\star\|^2 = (z - z^\star)^\top A(z)\,(z - z^\star) \le -\alpha\|z - z^\star\|^2,$$

so every solution converges to the goal. This bounds the distance to one point, not the distance between two solutions, which is governed by $Df(z) = A(z) + \frac{\partial A}{\partial z}(z)\,(z - z^\star)$. The constraint bounds only the first term; the second grows with $\|z - z^\star\|$ at a rate set by how fast A varies, which nothing in (53) limits.

Trained models are indeed not contracting in the Euclidean norm: along the training demonstrations of chain8z, in latent coordinates, the symmetric part of Df has a positive eigenvalue at 22% to 85% of the 23,067 sample points, across the ten models trained with LASER in single precision (fp32).

Jaffe et al. (2024) also show, by a converse contraction theorem, that ELCD is contracting in some state-dependent metric on any compact set containing the goal. The metric is defined by an integral

5. In chaining order: `CShape`, `Saeghe`, `JShape`, `Snake`, `Angle`, `Worm`, `Sshape` and `Multi_Models_1`.

along solutions and is never computed, so its condition number and contraction rate are unknown. It gives a bound of the form (6) along solutions of the model, but not the uniform bound along every candidate trajectory that Theorem 1 requires.

G.4. Training

Multi-start rollout loss. At every optimizer step, for each training demonstration b with samples $x_0^b, \dots, x_{N-1}^b$, we draw a set K_b of 64 start indices uniformly without replacement from $\{0, \dots, N-2\}$. From each start $k \in K_b$ the model is rolled out (Appendix G.2), giving $X_1^{b,k}, X_2^{b,k}, \dots$, and compared with the demonstration at matched times:

$$\mathcal{L}(\theta) = \frac{1}{D\,|\mathcal{P}|} \sum_{(b,k,j)\in\mathcal{P}} \left\| X_j^{b,k} - x_{k+j}^b \right\|^2, \qquad \mathcal{P} = \big\{(b,k,j) : k \in K_b,\ 1 \le j \le N-1-k\big\}. \tag{54}$$

All 192 rollouts are run for $N-1$ steps, so that one compiled program solves them; steps past the end of the demonstration are left out of (54). The seed sets both the initial weights and the draws of the starts, so all solvers of a seed train from the same initial model on the same starts.

Optimizer and gradients. We train for 10,000 steps with Adam (Kingma and Ba, 2015), a learning rate decaying from 10^{-3} to 10^{-5} by a cosine schedule, and gradient clipping at global norm 1, starting from the initialization in the code of Jaffe et al. (2024). For a PiT solver, the gradient of the converged rollout follows from the implicit function theorem: the adjoint is a linear recurrence in the transposed step Jacobians, solved by the same parallel scan. The sequential solver backpropagates through its steps.

Velocity loss. Jaffe et al. (2024) fit the data-space velocity $F(x) = D\varphi(x)^{-1} f(\varphi(x))$ to the recorded velocities by mean squared error over mini-batches of 100 samples, for 100 epochs at a constant learning rate of 10^{-3}. Besides this recipe, we train for 1,000 epochs at the constant rate (about the training time of our rollout runs), for 1,000 epochs with our cosine schedule, and, on seed 1, for 43,290 epochs ($\approx 10^7$ steps) with the cosine schedule. The recorded velocities are forward differences of the positions at the demonstration's Δt.

Evaluation. We roll the model out from the first sample of each demonstration as in training, for as many steps as the demonstration has, and compare the rollout $X = (X_i)$ with the demonstration $Y = (y_j)$ by the Chamfer distance

$$d_{\mathrm{C}}(X, Y) = \operatorname*{mean}_j \min_i \|X_i - y_j\| + \operatorname*{mean}_i \min_j \|X_i - y_j\|,$$

which Jaffe et al. (2024) report as DTWD, averaged over the 3 training or the 4 held-out demonstrations. We also report the multi-start RMSE, the root mean square of the errors in (54) over evenly spaced starts, which is the quantity the rollout loss minimizes.

G.5. Solvers and Timing Protocol

Solvers. Every PiT solver applies Gauss–Newton to the latent RK4 rollout, with dense step Jacobians and the parallel scan of Appendix K. DEER (Lim et al., 2024) takes the full Newton step. Scale-ELK (Gonzalez et al., 2024) multiplies every step Jacobian by $e^{-g\Delta t}$ for a constant rate g. LASER chooses the rate at every iteration by (38), with the safeguards of Appendix E. In the notation of Section 2.2,

with $B = I$, DEER is $A = J$, scale-ELK is $A = J - gI$, and LASER is $A = J - \gamma I$ with γ chosen at every iteration; subtracting γI from A corresponds to multiplying each step Jacobian by $e^{-\gamma \Delta t}$ (Appendix E).

Tuning. We tuned g on seeds 10 and 11, outside the reported seeds 0 to 9, by the training time of their first 2,000 steps (53 GPU hours in total). In every setting the smallest rate tried was the fastest (Table 3). Figure 2C shows $g = 0.03$, which in fp32 has about the same median training time as the tuned $g = 0.01$ (0.284 against 0.289 h) and fewer fallback steps; Tables 4 and 5 report both. ELK (Gonzalez et al., 2024) took a median of at least 2.6 s per step over the first 200 steps of chain8z in fp32 ($\sigma^2 \in \{1, 10^2, 10^4\}$), against 0.45 s for LASER, so we did not train with it.

Convergence and fallback. A solve stops when its maximum residual $\max_n \|r_n\|_\infty$ is at most τ (Table 3), when an iterate is not finite, when the update stalls, or after $N - 1$ iterations, the length of the longest rollout, by which DEER is exact in exact arithmetic (Appendix K). It has converged if its final residual is finite and at most τ. If any of the 192 solves of an optimizer step has not converged, the sequential solver re-solves that step's rollouts, and its time is charged to the PiT solver.

Warm starts. Each start index keeps its own initial guess. The first time a start k is drawn, its guess is the encoded demonstration $\varphi(x_{k+1}), \ldots, \varphi(x_{N-1})$, padded with $z^\star$ (a *cold* guess); afterwards it is that start's last converged rollout (a *warm* guess). On chain8z every guess is warm from about step 1,000 on, and a warm guess comes on average from the model of 120 optimizer steps earlier.

Timing. Every run used one NVIDIA B200, with JAX's matrix-multiply precision set to the highest in fp32. The training time of a run is the sum of the wall-clock times of its 10,000 optimizer steps, including the solves, any fallback, the adjoint and the Adam update, and excluding evaluations. The 192 solves of a step are processed in vmapped chunks of C solves, chosen per task and precision by short pilot runs (Table 3); a chunk runs until its slowest solve has finished. The sequential solver integrates all 192 rollouts of a step in one vmapped call and was timed in the same way over all 10,000 steps. LASER was run in a later job than the other solvers, with a later version of the code; a 20-step calibration run of DEER gives identical losses under both versions.

Table 3: Settings per task and precision (fp32: single; fp64: double). τ: residual tolerance. C: solves per vmapped chunk of the PiT solvers. g: the rate of scale-ELK selected by tuning. Every solver has the iteration cap $N - 1$; LASER has no tuned constant.

Task	Precision	τ	C	g
chain8z	fp32	10^{-5}	32	0.01
chain8z	fp64	10^{-7}	2	0.01
pendulum	fp32	10^{-5}	8	0.1
pendulum	fp64	10^{-7}	4	0.03

G.6. Details of Figure 2

Panel A. Left: the model of Appendix G.2. The diffeomorphism φ is drawn as the encoder and φ^{-1} as the decoder, with the latent dynamics (14) between them. Right: one LASER solve of a rollout of a trained model on chain8z. It converges in $K = 7$ iterations, and the cards show the iterates

$k = 0$, 1 and 7. The iterates are those of the solve; only the deviation of the initial guess from the solution was doubled before the solve, to make it visible. The black marker on each card is the step a sequential solver reaches in the same time: one LASER iteration of this rollout costs about as much as 33 sequential RK4 steps, so after the 7 iterations the sequential solver is at step 232 of 2,737. Solves during training are shorter: with warm guesses (Appendix G.5), a solve takes one or two iterations after the first 2,100 optimizer steps (Figure 5).

Panel B, trajectories. The first training demonstration of chain8z in normalized coordinates (gray) and the rollouts from its first sample and from four later samples (1,513, 3,087, 4,600 and 6,174), each run to the end of the demonstration. The starts are marked • and the goal ★. Top: the model trained with the velocity loss for 1,000 epochs at the constant learning rate, ten times the epochs of Jaffe et al. (2024) and about the same training time as the rollout runs. Bottom: the model trained with the multi-start rollout loss and LASER for 10,000 steps in fp32. For each loss we show the seed with the lowest final training Chamfer distance of the ten: seed 7 (0.30) for the velocity loss and seed 1 (0.09) for the rollout loss.

Panel B, curves. The training Chamfer distance (mean over the 3 training demonstrations) of two long runs on chain8z, seed 1, fp32, against the training time: the wall-clock time of the optimizer steps before each evaluation, evaluations excluded as in Appendix G.5. Both use the cosine schedule over the whole run. The rollout loss with LASER runs for 10^6 steps (25.0 h) and is evaluated at steps 500, 1,000, 2,000 and 5,000, and then every 10,000 steps. The velocity loss runs for 43,290 epochs ($\approx 10^7$ steps, 10.7 h, an epoch being 231 mini-batches on chain8z) and is evaluated every 500 epochs (115,500 steps).

Panel C. The training time of 10,000 steps on chain8z in fp32, over seeds 0 to 9, as box plots: the median (black line), the quartiles (box) and the minimum and maximum (whiskers). Scale-ELK is shown at $g = 0.03$. All times are measured, the sequential solver's included (Appendix G.5). The axis is broken between the sequential solver and the PiT solvers. The per-seed values are in Table 4.

G.7. Further Results on chain8z

Tables 4 and 5 give the training time and the fallback steps of every seed. A *fallback step* is an optimizer step that the sequential solver re-solved because a PiT solve had not converged (Appendix G.5). LASER had none on any seed in either precision.

Time. In fp32, DEER trains 24 to 40× faster than the sequential solver and LASER 33 to 41×. LASER's training time varies least over the seeds, 0.238 to 0.296 h against 0.243 to 0.455 h for DEER, whose slow seeds are the ones with many fallback steps. In fp64, fallback steps are rare (at most 24 per seed for DEER) and every PiT solver is 2.6 to 4.0× faster than the sequential solver. LASER is 2 to 12% slower than DEER on 9 of the 10 seeds, because it computes the finite-time Lyapunov exponent at every iteration, which gains nothing when the plain Newton steps already converge.

Where the failures happen. Figure 5 shows seed 0 in fp32. DEER's fallback steps and the long solves of both solvers fall in the first 2,100 steps, when the guesses are cold or come from a model that still changes quickly (Appendix G.5). LASER's slowest solve there takes 213 iterations against 263 for DEER, but none of its solves fails. After that, the slowest of a step's 192 solves takes 2 iterations (median) and never more than 6 with either solver, and a step takes the same time with both solvers.

Table 4: Training time of the 10,000 optimizer steps and number of fallback steps on chain8z in fp32, by seed. The last three rows give the median, the minimum and the maximum over the ten seeds; bold marks the shortest training time in each of them.

	Training time (h) (↓)					Fallback steps (↓)			
			scale-ELK				scale-ELK		
Seed	sequential	DEER	$g = 0.01$	$g = 0.03$	LASER	DEER	$g = 0.01$	$g = 0.03$	LASER
0	10.74	0.455	0.513	0.483	0.292	515	646	554	0
1	9.75	0.243	0.243	0.247	0.250	0	0	0	0
2	9.80	0.293	0.298	0.300	0.295	10	22	8	0
3	10.15	0.291	0.298	0.287	0.261	179	193	164	0
4	9.96	0.270	0.276	0.279	0.275	11	15	16	0
5	9.94	0.289	0.289	0.281	0.267	86	79	72	0
6	9.77	0.248	0.249	0.252	0.238	24	21	20	0
7	9.84	0.353	0.378	0.361	0.296	464	473	397	0
8	9.56	0.259	0.259	0.261	0.264	13	11	9	0
9	9.75	0.279	0.289	0.293	0.284	81	87	84	0
median	9.82	0.284	0.289	0.284	**0.271**	52.5	50.5	46	0
min	9.56	0.243	0.243	0.247	**0.238**	0	0	0	0
max	10.74	0.455	0.513	0.483	**0.296**	515	646	554	0

Table 5: Training time and fallback steps on chain8z in fp64, by seed. Columns, summary rows and bold as in Table 4.

	Training time (h) (↓)					Fallback steps (↓)			
			scale-ELK				scale-ELK		
Seed	sequential	DEER	$g = 0.01$	$g = 0.03$	LASER	DEER	$g = 0.01$	$g = 0.03$	LASER
0	9.54	3.064	3.273	3.577	3.044	18	18	16	0
1	9.84	2.541	2.758	3.102	2.620	0	0	0	0
2	9.89	2.789	2.991	3.581	2.981	0	0	0	0
3	9.82	2.644	2.838	3.284	2.734	0	2	0	0
4	9.39	2.762	2.927	3.327	2.903	1	1	1	0
5	9.24	2.768	2.879	3.263	2.824	6	5	1	0
6	9.97	2.523	2.618	2.873	2.640	0	0	0	0
7	9.19	2.713	3.138	3.600	3.043	24	34	19	0
8	9.84	2.603	2.806	3.201	2.725	2	0	0	0
9	9.31	2.681	2.854	3.252	2.790	3	8	8	0
median	9.68	**2.697**	2.867	3.274	2.807	1.5	1.5	0.5	0
min	9.19	**2.523**	2.618	2.873	2.620	0	0	0	0
max	9.97	3.064	3.273	3.600	**3.044**	24	34	19	0

Quality is the same across solvers. Every PiT solver stops by the same residual test (Appendix G.5), and the final training Chamfer distance does not depend on the solver in a systematic way. Over the ten seeds it is 0.186 ± 0.032 (DEER), 0.198 ± 0.039 (LASER) and 0.170 ± 0.021 (sequential solver)



Figure 5: DEER and LASER over the 10,000 optimizer steps on chain8z in fp32, seed 0. **A**: time of each step; the lines are medians over 51 consecutive steps. **B**: Newton iterations of the slowest of the step's 192 solves. Dark gray: the steps re-solved by the sequential solver, all of them DEER's (515, the last at step 1,882). The first step, which includes compilation, is not drawn. Where the two solvers coincide, LASER's dots cover DEER's.

in fp32, and 0.174 ± 0.021, 0.172 ± 0.022 and 0.177 ± 0.022 in fp64 (mean $\pm$ standard error). The fp32 means of DEER and LASER each include one seed on which training stalls, at 0.415 (DEER, seed 9) and 0.502 (LASER, seed 7).

Rollout loss against the velocity loss. Table 6 compares the final models of the two losses on chain8z in fp32. The rollout loss gives the lowest median on every metric, at the training time of the 1,000-epoch velocity runs, and gives lower values on all three metrics in every seed than the recipe of Jaffe et al. (2024) and than the cosine-schedule run. Against 1,000 epochs at the constant learning rate it gives lower values on the multi-start RMSE in every seed and on both Chamfer distances in 9 of 10. On seed 7 the rollout run stalls (0.50 on the training demonstrations) while the velocity loss has its best seed (0.30, the one drawn in Figure 2B). Figure 6 shows the five best seeds of each loss.

Table 6: Final models of the velocity loss and of the multi-start rollout loss (Appendix G.4) on chain8z in fp32, medians over seeds 0 to 9. Bold: the lowest value of each metric.

	Chamfer distance (↓)		Multi-start	Training
	Training	Held-out	RMSE (↓)	time (h)
initial model	1.26	1.27	1.31	
velocity loss, 100 epochs (Jaffe et al., 2024)	1.51	1.33	1.73	0.04
velocity loss, 1,000 epochs	0.73	0.77	1.17	0.26
velocity loss, 1,000 epochs, cosine schedule	1.11	1.09	1.48	0.26
rollout loss, LASER, 10,000 steps	**0.17**	**0.23**	**0.28**	0.27

Long runs. The curves of Figure 2B are two long runs of seed 1 in fp32: 10^6 steps of the rollout loss with LASER and 43,290 epochs of the velocity loss (Appendix G.6). The rollout run levels off at a Chamfer distance of 0.042 on the training and 0.102 on the held-out demonstrations (median of the

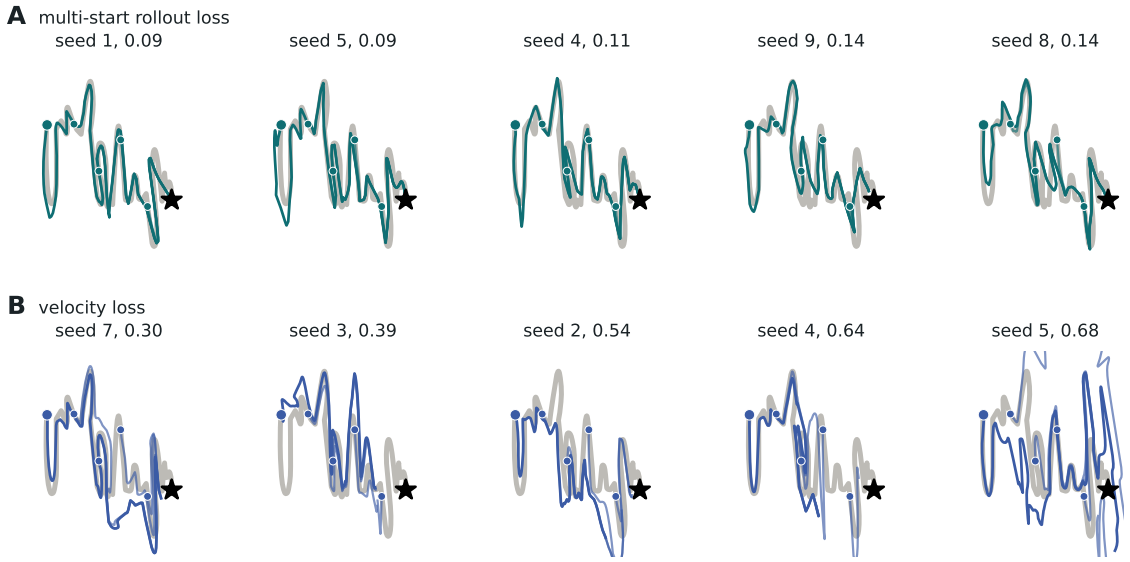


Figure 6: On chain8z in fp32, the five seeds of each loss with the lowest final training Chamfer distance of the ten, the best at the left, drawn as in Figure 2B (demonstration in gray, rollouts from five of its samples in color). **A**: multi-start rollout loss with LASER. **B**: velocity loss, 1,000 epochs at the constant learning rate. Above each panel: the seed and its training Chamfer distance. The other five seeds are between 0.19 and 0.50 (A) and between 0.78 and 5.69 (B).

last ten evaluations). The 10,000-step run of the same seed already reaches 0.089 and 0.124. This run is not a point of the curve: after 10,000 steps the long run is at 0.107 and 0.154, because its schedule spans the 10^6 steps and its learning rate is still close to 10^{-3}. Over the 10^6 steps LASER had no fallback step and took at most 5 iterations per solve. The velocity loss falls throughout, to 0.013, but the training Chamfer distance of its model does not level off. It is lowest at 0.24, after 4.6×10^6 steps, and 0.60 at the end, and the final model's multi-start RMSE (1.69) is above the initial model's (0.87).

G.8. Eight-Link Pendulum

Table 7 gives the results on the pendulum (3 seeds per precision). LASER has no fallback step and the shortest time in all 6 rows, and all solvers reach about the same final training Chamfer distance (0.023 to 0.031). LASER's speedup over the sequential solver is smaller than on chain8z: 1.7 to 1.8× in fp32 and 1.1 to 1.2× in fp64. Even an optimizer step in which every solve converges in its first iteration takes 1.43 s in fp32, against 3.0 s for the sequential solver (0.08 against 3.4 to 3.9 s on chain8z). In fp32, 30 to 48% of LASER's steps are of this kind (Figure 7); in fp64 the slowest solve of a step takes two iterations (median) and a step 2.5 s.

DEER needs more iterations than LASER in the first 1,000 steps. These contain all of DEER's fallback steps and take it 0.9 to 1.7 h, against 0.7 to 1.2 h for LASER. Afterwards the two solvers run the same number of iterations (within 3%).

Table 7: Training time of the 10,000 optimizer steps and number of fallback steps on the eight-link pendulum ($N = 5{,}000$, $D = 16$), by seed, with scale-ELK at its tuned rate ($g = 0.1$ in fp32, 0.03 in fp64). Bold: the shortest training time of each row.

		Training time (h) (↓)				Fallback steps (↓)		
Precision	Seed	sequential	DEER	scale-ELK	LASER	DEER	scale-ELK	LASER
fp32	0	8.34	5.694	6.415	**4.900**	311	352	0
fp32	1	8.50	4.902	5.623	**4.738**	91	75	0
fp32	2	8.33	4.937	5.853	**4.761**	112	90	0
fp64	0	8.46	8.330	8.621	**7.433**	1	3	0
fp64	1	8.43	8.076	7.766	**7.295**	3	1	0
fp64	2	8.40	7.344	8.250	**7.223**	1	5	0

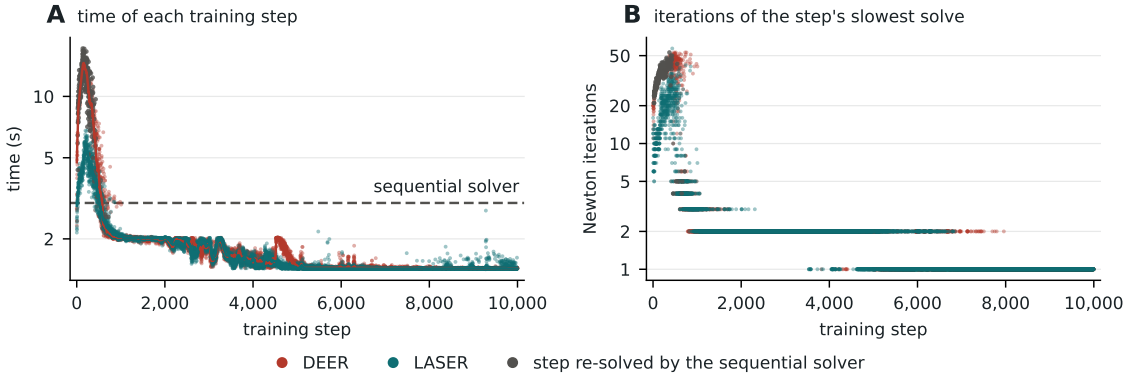


Figure 7: Eight-link pendulum in fp32, seed 0: DEER and LASER over the 10,000 optimizer steps, drawn as in Figure 5. Dashed line in **A**: the median step time of the sequential solver (3.0 s). Dark gray: DEER's 311 fallback steps, the last at step 719.

G.9. The Original Tasks of Jaffe et al. (2024)

To check that the results do not depend on the tasks we built, we repeat the comparison on the original, shorter tasks of Jaffe et al. (2024): the 30 LASA shapes (985 samples, 3 training and 4 held-out demonstrations) and their eight-link pendulum (1,000 samples, $\Delta t = 0.1$ s, 3 training and 7 held-out demonstrations). For seeds 0 to 2 in both precisions, we train two models from the same initial weights: one with their velocity loss and recipe (100 epochs) and one with the rollout loss and LASER (10,000 steps).

Loss. In fp32 the rollout loss gives the lower training Chamfer distance on 29 of the 30 shapes and the lower held-out Chamfer distance on 19 (Figure 8); in fp64 the counts are 30 and 20. On the pendulum it gives the lower value on every metric in every seed. The difference is smaller than on chain8z: the median training Chamfer distance over the 90 fp32 runs on the LASA shapes is 0.124 against 0.205 for the velocity loss, compared with 0.17 against 1.51 in Table 6.

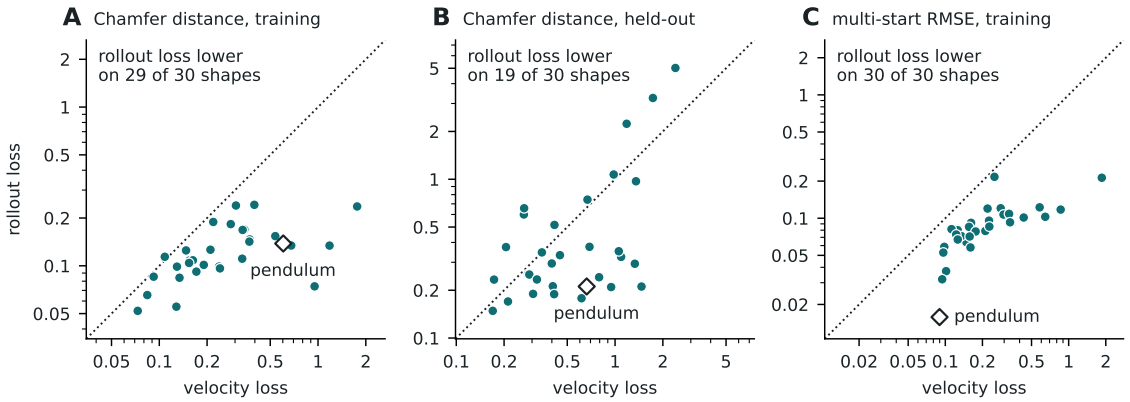


Figure 8: The original tasks of Jaffe et al. (2024) in fp32: the rollout loss against the velocity loss. Each point is one task, the mean over seeds 0 to 2: the 30 LASA shapes (•) and the pendulum (◊). Below the dotted line the rollout loss gives the lower value. **A**: Chamfer distance on the training demonstrations. **B**: on the held-out demonstrations. **C**: multi-start RMSE on the training demonstrations.

Speed. LASER had no fallback step in any of the 186 runs. Table 8 gives the training time on three LASA shapes (`Sshape`, `JShape` and `Angle`) and on the pendulum. On the LASA shapes DEER and LASER are about 30× faster than the sequential solver in fp32 and 4 to 6× in fp64. On the pendulum, LASER is 1.9× faster in fp32 and 1.15× in fp64; DEER is 1.7× faster in fp32 and slower than the sequential solver in fp64. DEER has no fallback step on the three LASA shapes, so these tasks do not distinguish it from LASER as chain8z does.

Table 8: Training time on the original tasks (one NVIDIA B200): median over seeds 0 to 2 of the time of 10,000 steps, and the largest number of fallback steps in any seed. The time of the sequential solver is extrapolated from a 200-step run. Bold: the shortest training time of each row.

		Training time (h) (↓)			Fallback steps (↓)	
Task	Precision	sequential	DEER	LASER	DEER	LASER
`Sshape`	fp32	1.277	**0.042**	0.044	0	0
`JShape`	fp32	1.251	**0.040**	0.045	0	0
`Angle`	fp32	1.255	**0.040**	0.042	0	0
pendulum	fp32	1.730	1.028	**0.893**	121	0
`Sshape`	fp64	1.283	**0.215**	0.254	0	0
`JShape`	fp64	1.213	**0.233**	0.322	0	0
`Angle`	fp64	1.206	**0.234**	0.243	0	0
pendulum	fp64	1.718	1.813	**1.495**	3	0

Appendix H. Details of the Bistable Lorenz Experiment

Setup. The system of Section 3.2 is discretized by explicit Euler with $\Delta t = 0.01$ and solved in double precision over horizons of $N = 10^3$, 10^4 and 10^5 steps, from the zero initial guess, the constant trajectory at the saddle. Its finite-time Lyapunov exponent (37) is $\lambda_h = +5.32$, against -0.19 to -0.40 along the solution. The first DEER step is therefore of order $e^{\lambda_h T}$: at $N = 10^3$ the iterate reaches 1.5×10^{23}, and the quadratic term of the Lorenz system makes its residual 1.3×10^{44} (Figure 3C). The next step is non-finite. A solve has converged when its residual $\max_n \|r_n\|_\infty$ is below 10^{-9}. Every solver is given the budget of 200 iterations used for Figure 3 ($N = 10^3$); a baseline that has not converged by then is run again with 2,000, and ELK with $\sigma^2 = 10^4$ at $N = 10^5$, which needs more, with 5,000. Times are measured on one NVIDIA B200, as the median of ten solves after a compilation run.

Solvers. DEER (Lim et al., 2024) takes the full Newton step. Scale-ELK (Gonzalez et al., 2024) multiplies every step Jacobian by $1 - k$ for a constant k, which is $e^{-g\Delta t}$ with the rate $g = -\log(1 - k)/\Delta t$ in the notation of Appendix G.5. ELK (Gonzalez et al., 2024) takes a Levenberg–Marquardt step with damping parameter $1/\sigma^2$ and is run with a constant σ^2. A smaller σ^2 gives a smaller trust region and more damping, and $\sigma^2 \to \infty$ recovers DEER. LASER uses (38) with the safeguards of Appendix E.

Tuning. We ran scale-ELK at 16 values of k from 0.03 to 0.5 and ELK at 11 values of σ^2 from 1 to 10^5, at every horizon (Figure 9). Table 9 reports the constants of Figure 3 ($k = 0.05$ and $\sigma^2 = 10^4$) and the best ones found. Scale-ELK returns a non-finite iterate when k is too small, at $N \geq 10^4$ for every $k \leq 0.052$. This boundary is close to $k = 1 - e^{-\lambda_h \Delta t} = 0.0518$, above which the damped Jacobian product $(1 - k)^N J_N \cdots J_1$ of the guess stops growing. Among the values that converge, its iteration count increases with k, so the best constant is the smallest that converges: $k = 0.05$ at $N = 10^3$ and $k = 0.053$ at $N \geq 10^4$. ELK needs the fewest iterations at $\sigma^2 = 3 \times 10^4$ at every horizon and returns a non-finite iterate at $\sigma^2 = 10^5$. LASER has no constant to tune and is the fastest solver at every horizon, $1.3\times$, $4.1\times$ and $4.0\times$ faster in wall-clock time than the fastest baseline at $N = 10^3$, 10^4 and 10^5 (Table 9). It is also $2.0\times$, $9.8\times$ and $28\times$ faster than the sequential solver. Figure 10 shows the residual at every iteration for the best constants at $N = 10^4$ and 10^5.

Appendix I. Supplementary Experiments

We complement the experiments in Section 3 with three settings: a fixed contractive reservoir, a nonlinear state-space model whose contractivity is enforced throughout training, and robotic systems stabilized by feedback. Each subsection presents the results together with the associated model, solver, and training details.

I.1. GPU Acceleration of Reservoir Computing

By Theorem 5, Jacobi iteration reaches precision ϵ in a number of iterations independent of N; each iteration is embarrassingly parallel across time steps, with no inter-step communication and no Jacobian storage. Parallel-scan methods like DEER (Lim et al., 2024) and quasi-DEER (Gonzalez et al., 2024) take fewer but more expensive iterations: each Newton step incurs an $O(\log N)$ parallel-scan depth and stores Jacobians at $O(ND^2)$ memory for Full-DEER ($O(ND)$ for quasi-DEER).

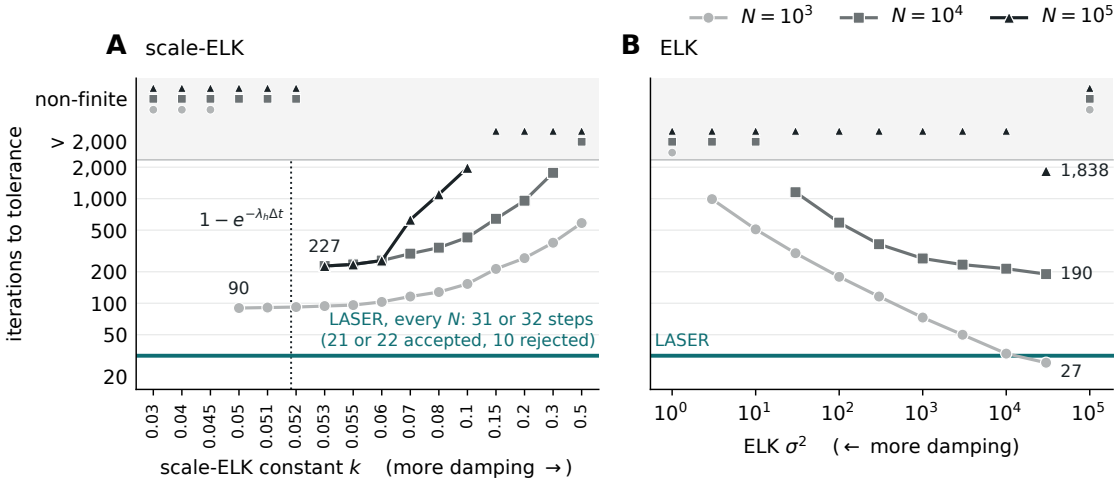


Figure 9: Iterations to reach the tolerance on the bistable Lorenz problem against the constant of each baseline, at three horizons. **A**: scale-ELK; the tested values of k are drawn equally spaced, and the dotted line marks $k = 1 - e^{-\lambda_h \Delta t}$ for the initial guess. **B**: ELK. Markers in the shaded band are runs that returned a non-finite iterate or did not converge in 2,000 iterations. Numbers give the fewest iterations at each horizon. Teal line: LASER, counting its rejected trial steps.

Table 9: Iterations to reach the tolerance on the bistable Lorenz problem and wall-clock time of the solve, for the constants of Figure 3 and the best constants found, at $N = 10^3$, 10^4 and 10^5. LASER: accepted iterations, with rejected trial steps in parentheses; its time includes them. Sequential: the N Euler steps one after another (no iterations). nf: the iterate became non-finite, so the solve has no time. ELK with $\sigma^2 = 10^4$ at $N = 10^5$ was given 5,000 iterations, every other solve at most 2,000. Bold: the shortest time at each N.

	Iterations (↓)			Time (ms) (↓)		
Solver	10^3	10^4	10^5	10^3	10^4	10^5
sequential	–	–	–	14.4	123	1,280
DEER	nf	nf	nf	nf	nf	nf
scale-ELK, $k = 0.05$	90	nf	nf	9.3	nf	nf
scale-ELK, $k = 0.053$	94	227	227	9.6	51.3	179
ELK, $\sigma^2 = 10^4$	33	214	2,063	62.2	682	16,700
ELK, $\sigma^2 = 3 \times 10^4$	27	190	1,838	51.0	607	15,000
LASER	22 (10)	21 (10)	21 (10)	**7.3**	**12.6**	**45.2**

Wall-clock time, however, is not determined by iteration count alone. For sufficiently long sequences, even an embarrassingly parallel iteration saturates the compute units of a single GPU, at which point the per-iteration cost grows linearly in N. Figure 11 reports the resulting speedups over sequential rollout, with all solvers run to the same stopping tolerance (Appendix I.1.1): newer GPUs

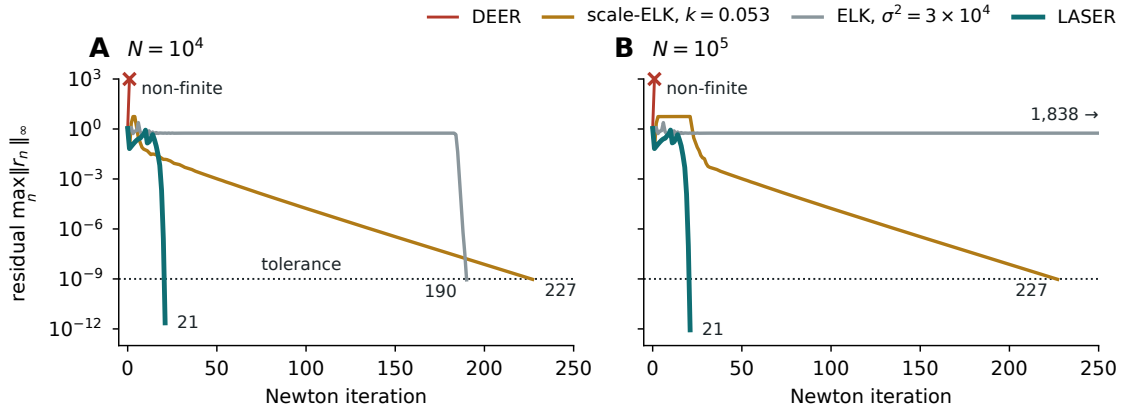


Figure 10: Residual against iteration on the bistable Lorenz problem with the best constants of Table 9. **A**: $N = 10^4$. **B**: $N = 10^5$. Dotted line: the tolerance. Numbers: the iteration at which a solver reaches it. The axis ends at 250 iterations; ELK at $N = 10^5$ reaches the tolerance at iteration 1,838. DEER's residual is non-finite after its first step. LASER's 10 rejected trial steps are not drawn.

deliver larger speedups for all three solvers, Full-DEER runs out of memory at large N, and Jacobi continues to scale to $N = 10^7$.

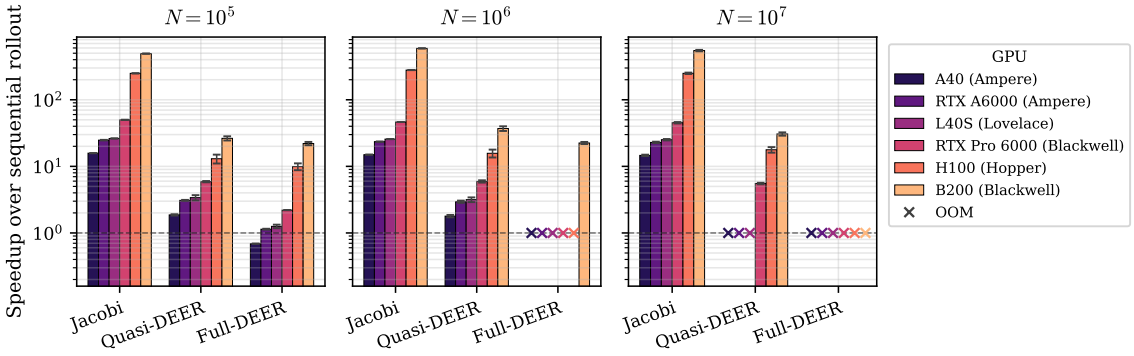


Figure 11: Wall-clock speedup over sequential rollout for three parallel-in-time solvers (Jacobi, quasi-DEER, Full-DEER) across six GPU classes (bar colors), with sequence length increasing left to right (N=10^5, 10^6, 10^7); $\times$ marks indicate out-of-memory, and all solvers run to the same stopping tolerance. Speedups grow with both sequence length and GPU generation. On these dense-weight dynamics Jacobi is competitive with the Newton-based methods at every N while storing no Jacobians; Full-DEER runs out of memory at large N, quasi-DEER at N=10^7 on 48 GB GPUs. See Table 10 for multi-GPU scaling on the same dynamics.

We evaluate on a simple contractive recurrent system, representative of the fixed-reservoir dynamics used in reservoir computing and echo-state networks, where a nonlinear recurrent system

is driven by an input sequence and only a downstream readout is typically trained (Jaeger, 2001; Lukoševičius and Jaeger, 2009; Lukoševičius, 2012; Tanaka et al., 2019; Yan et al., 2024). Concretely, we consider the non-autonomous contractive RNN

$$\dot{x} = -x + W \tanh(x) + c(t), \tag{55}$$

discretized with explicit Euler at step size $h = 1$,

$$x_{n+1} = W \tanh(x_n) + c_n, \qquad n = 0, \dots, N-1, \tag{56}$$

with $W \in \mathbb{R}^{D\times D}$ rescaled to have operator norm $\rho < 1$ so that the hypotheses of Theorem 5 are met. We report results at $D = 64$, $\rho = 0.6$; full setup in Appendix I.1.1.

Multi-GPU partitioning is also cheap: each iteration is local except for a single boundary exchange between adjacent devices. Table 10 reports the resulting scaling. Scaling improves with sequence length: at $N = 10^7$, going from one to eight GPUs raises the speedup from $548\times$ to $3{,}209\times$ (a factor of 5.9, close to the ideal 8), while at $N = 10^6$ the gain saturates near four GPUs (a factor of 2.5 at eight). At $N = 10^5$ the per-device workload no longer saturates a B200 and inter-device latency dominates, so multi-GPU is slower than single-GPU.

Table 10: Wall-clock speedup of Jacobi iteration over sequential rollout as the number of GPUs increases (rows) and as sequence length grows (columns); D=64, ρ=0.6, fp64, mean $\pm$ std over 5 seeds. Scaling approaches linear as sequence length grows, while short sequences are slower on multiple GPUs because inter-device latency dominates. Bold: the highest speedup at each N.

GPUs	Wall-clock speedup (↑)		
	N=10^5	N=10^6	N=10^7
1× B200	$\mathbf{492.7} \pm 0.2$	591.0 ± 0.1	547.6 ± 13.6
2× B200	419.3 ± 0.7	$1{,}003.9 \pm 0.6$	$1{,}058.1 \pm 25.5$
4× B200	342.3 ± 2.9	$1{,}428.9 \pm 1.8$	$2{,}053.2 \pm 49.3$
8× B200	168.4 ± 5.9	$\mathbf{1{,}457.7} \pm 21.7$	$\mathbf{3{,}209.1} \pm 76.7$

I.1.1. EXPERIMENTAL PROTOCOL

We use the contractive RNN (55)–(56) with $D = 64$ and $\rho = 0.6$ for both the single- and multi-GPU experiments.

Weight matrix. $W \in \mathbb{R}^{D\times D}$ is constructed to have operator norm exactly ρ: we draw a matrix with i.i.d. $\mathcal{N}(0, 1)$ entries, compute its singular value decomposition $U\Sigma V^\top$, and set $W = \rho UV^\top$, so that every singular value of W equals ρ. Since $\tanh$ is 1-Lipschitz, the resulting discrete map $x_{n+1} = W \tanh(x_n) + c_n$ is globally contractive with rate $\rho < 1$, satisfying the hypotheses of Theorem 5. For results in Appendix I.1, D=64, ρ=0.6.

Inputs and initialization. The inputs c_n and the initial state x_0 are drawn i.i.d. from $\mathcal{N}(0, I_D)$, independently for each seed. Jacobi starts from a random trajectory guess $\mathbf{s}^{(0)} \in \mathbb{R}^{ND}$ with i.i.d. standard normal entries, and Full-DEER and quasi-DEER from zeros.

Solver settings. All arithmetic is in fp64, and all three solvers are run to the same stopping criterion, the mixed absolute/relative tolerance test of Gonzalez et al. (2024): iteration stops when $\max_{n,i}\big(|\Delta s_{n,i}| - r_{\mathrm{tol}}\,|s_{n,i}|\big) < \tau_{\mathrm{tol}}$ with $\tau_{\mathrm{tol}} = 10^{-7}$ and $r_{\mathrm{tol}} = 10^{-4}$, where $\Delta\mathbf{s}$ is the most recent update. For Jacobi the update equals the negative residual, so this criterion is a relative residual tolerance. Full-DEER and quasi-DEER run as early-stopping while-loops with an iteration cap of 2,000, which is never reached. Jacobi wall-clock time is measured with a fixed-length scan of 120 iterations and rescaled by (iterations to tolerance)/120, which is equivalent to a scan of exactly the required length; we use this scan-based measurement because a while-loop inside a sharded multi-GPU program cannot overlap the boundary exchange with computation, and the rescaling keeps the methodology identical across single- and multi-GPU runs. At this tolerance the measured iteration counts are essentially independent of N across $N = 10^3$ to 10^7: 6 for Full-DEER, 86–106 for quasi-DEER, and 20–22 for Jacobi under the criterion above (22–23 for the scan-based Jacobi runner, which stops on the stricter condition that the error to the sequential ground-truth trajectory falls below τ_{tol}).

Hardware and timing. Single-GPU experiments (Figure 11) sweep $N \in \{10^5, 10^6, 10^7\}$ across six GPU classes: A40, RTX A6000, L40S, RTX Pro 6000, H100, and B200. Multi-GPU experiments (Table 10) use $\{1, 2, 4, 8\}\times$ B200. Each reported wall-clock time is the median of 3 timed runs, taken after one untimed run that triggers JIT compilation; results are aggregated over 5 seeds. The sequential baseline is a JIT-compiled scan rollout timed with the same protocol on a single GPU of the same type. The out-of-memory marks in Figure 11 arise as follows: Full-DEER exceeds memory for $N \geq 3 \times 10^5$ on the 48 GB GPUs (A40, RTX A6000, L40S), for $N \geq 10^6$ on H100 and RTX Pro 6000, and for $N \geq 3 \times 10^6$ on B200; quasi-DEER exceeds memory at $N = 10^7$ on the 48 GB GPUs.

I.2. Parallel Training of Contractive State-Space Models

We test whether parallel-in-time solvers accelerate *training* a nonlinear sequence model end-to-end. We consider a continuous-time, diagonal SSM cell similar to Farsang et al. (2025), modified so that contractivity is enforced *globally* rather than locally, satisfying the hypotheses of Theorem 1. Each layer evolves a hidden state $h(t) \in \mathbb{R}^D$ under input u_n:

$$\dot{h} = -\gamma_\theta(u_n) \odot h + A_\theta(u_n) \odot \tanh(h) + b_\theta(u_n). \tag{57}$$

Here $\gamma_\theta,\ A_\theta \in \mathbb{R}^D$ are diagonal vectors produced by shallow MLPs with sigmoid heads constraining $A_\theta \in [-\alpha_{\max}, -\alpha_{\min}]^D$ and $\gamma_\theta \in [\gamma_{\min}, \gamma_{\max}]^D$ with $\gamma_{\min} > 0$, and b_θ is an input-dependent forcing. By construction, the explicit-Euler propagator is a strict contraction with worst-case rate $\rho \leq 0.99$ at our defaults, independently of the trained weights. Cells are stacked into an L-layer pre-norm residual model; full derivation, parametrization, and training details are in Appendix I.2.1.

We test on EigenWorms (Yemini et al., 2013), a 5-class *C. elegans* genotype-classification benchmark with $N = 17{,}984$ time steps. Figure 12B shows that the contractivity constraint is necessary: with γ_θ removed, both Jacobi and quasi-DEER fail to converge and downstream accuracy collapses.[6] This observed failure is consistent with the conditioning picture of Theorem 2. Without the leak term the cell admits no uniform contraction rate (Appendix I.2.1); applying the theorem would additionally require the growth bounds (8). With contractivity enforced, classification accuracy

6. Accuracy does not fall all the way to chance because, even when the parallel solver fails to find the cell's fixed point, training and evaluation both use the same partially converged solver iterate as the effective forward map; the network learns to make that perturbed map informative for the labels.

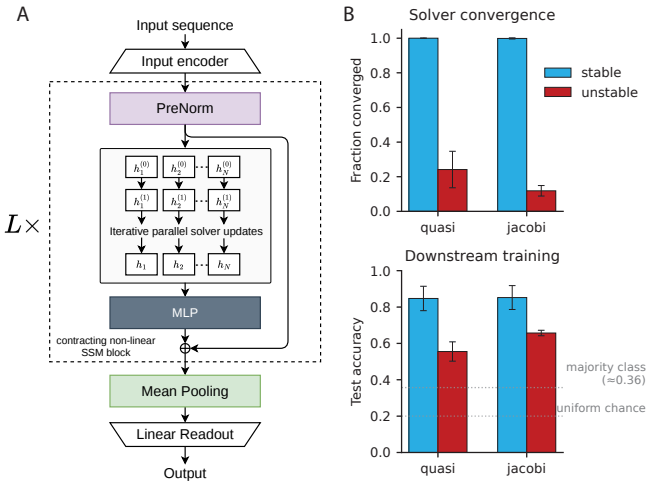


Figure 12: **A**: contracting nonlinear SSM architecture. The input sequence is first passed through an input encoder, followed by L contracting SSM blocks. Each block runs a pre-norm followed by the cell ODE $\dot{h} = -\gamma \odot h + A \odot \tanh(h) + b$ solved by a parallel-in-time solver and an MLP. A mean-pooled linear readout produces the output. **B**: ablation in which the contractivity constraint (γ_θ head) on the nonlinear SSM cell is either enforced (blue) or removed (red). Without the contractivity constraint, both solvers (quasi-DEER and Jacobi) fail to converge and downstream accuracy collapses.

is roughly flat (83–85%) across a $20\times$ range of $\gamma_{\min}$, while the Jacobi speedup grows monotonically from $4\times$ to $36\times$ (Table 11). The speedup growth matches Theorem 5, since tighter contractivity gives a smaller ρ and so fewer iterations to fixed precision.

Table 11: EigenWorms classification with the contracting nonlinear SSM (L=4, D=32). Mean $\pm$ std over n=5 seeds. We sweep the contraction floor $\gamma_{\min}$ (rows) from 10^{-2} to 0.2 and report test accuracy (averaged over the three solvers) and the speedups of Jacobi and quasi-DEER over the sequential baseline. Test accuracy is roughly flat across this range, while Jacobi's speedup grows monotonically from $4\times$ to $36\times$, matching Theorem 5, since tighter contractivity gives a smaller ρ and so fewer iterations to fixed precision.

		Wall time per epoch (s) (↓)			Speedup vs. sequential (↑)	
$\gamma_{\min}$	Test acc. (%) (↑)	sequential	quasi-DEER	Jacobi	quasi-DEER	Jacobi
0.01	84.8 ± 5.2	51.2 ± 5.9	0.56 ± 0.04	12.12 ± 1.02	$91\times \pm 8.7$	$4.2\times \pm 0.4$
0.02	85.4 ± 6.8	45.8 ± 1.5	0.52 ± 0.03	7.83 ± 0.38	$89\times \pm 6.0$	$5.9\times \pm 0.3$
0.05	85.0 ± 6.6	49.5 ± 3.6	0.46 ± 0.02	4.06 ± 0.07	$108\times \pm 9.2$	$12.2\times \pm 1.0$
0.10	84.8 ± 5.5	50.2 ± 6.0	0.42 ± 0.02	2.25 ± 0.06	$121\times \pm 17.2$	$22.3\times \pm 3.1$
0.20	83.1 ± 8.2	45.0 ± 1.0	0.39 ± 0.01	1.25 ± 0.02	$117\times \pm 4.2$	$36.0\times \pm 1.0$

Quasi-DEER is fastest in absolute terms on this task because the cell's state Jacobian is diagonal by construction (Appendix I.2.1), so its diagonal Jacobian is exact and the method collapses to Full-DEER's Newton-type convergence at $O(ND)$ rather than $O(ND^2)$ memory. This also explains

Table 12: Hyperparameter defaults for the contracting-SSM EigenWorms experiments. All entries are shared across the $\gamma_{\min}$ sweep of Table 11; only $\gamma_{\min}$ varies across rows.

Group	Hyperparameter	Value
Architecture	layers L	4
	hidden dim D	32
	MLP backbone width H	32
	FFN expansion ratio r	2
Cell	A_θ parametrization	sigmoid, $A_\theta \in [-\alpha_{\max}, 0]$
	γ_θ parametrization	sigmoid, $\gamma_\theta \in [\gamma_{\min}, \gamma_{\max}]$
	$\alpha_{\max}$ / $\alpha_{\min}$	$1.5 \,/\, 0$
	$\gamma_{\max}$	0.4
	$\gamma_{\min}$	$\{0.01, 0.02, 0.05, 0.10, 0.20\}$
	forcing	$\tanh\big(\mathrm{MLP}_2\big)$
Init	weights	Xavier-normal
	A-head bias $b_2^{(A)}$	$\mathcal{U}(-2, 2)$
	γ-head bias $b_2^{(\gamma)}$	$\mathcal{U}(-2, 2)$
	other biases, LN scale/shift	0, 1, 0
Solver	discretization	explicit Euler, $\Delta t = 1$
	residual tolerance	10^{-4}
	max iters (quasi-DEER / Jacobi)	$200 \,/\, 1{,}000$
Optimizer	algorithm	AdamW
	peak LR $\eta_{\max}$	3×10^{-4}
	schedule	warmup–cosine, warmup $= \min(500, T_{\text{steps}}/4)$
	weight decay	10^{-4}
	gradient clip (global norm)	1.0
	batch size	8
	epochs / patience	$60 \,/\, 30$
Hardware	device	$1\times$ NVIDIA B200
	precision	fp32

the reversed ordering relative to Appendix I.1: there the recurrent weights are dense, the diagonal approximation is inexact, and Jacobi's matrix-free profile becomes the relevant alternative.

All solvers stop at residual tolerance 10^{-4} (iteration caps 200 for quasi-DEER, 1,000 for Jacobi) and solve the same discrete system; their test accuracies coincide in 23 of the 25 ($\gamma_{\min}$, seed) pairs. The sequential baseline uses the same parallel backward pass as the other two, so the speedups reflect the forward solve. They are ratios of the wall time of the final epoch (training and validation), paired by seed. Full hyperparameters are in Table 12.

I.2.1. ARCHITECTURE AND TRAINING DETAILS

The EigenWorms experiments use the following cell parametrization, initialization, residual-block architecture, and training protocol.

State Jacobian and contraction rate. The cell (57) has diagonal state Jacobian

$$\frac{\partial \dot{h}}{\partial h} = -\operatorname{diag}\big(\gamma_\theta(u_n)\big) + \operatorname{diag}\big(A_\theta(u_n) \odot \operatorname{sech}^2(h)\big),$$

with entries in $[-(\gamma_{\max}+\alpha_{\max}),\ -\gamma_{\min}]$ uniformly in h, u_n. For explicit Euler with step Δt satisfying $\Delta t(\gamma_{\max} + \alpha_{\max}) < 2$, every diagonal entry of the one-step propagator lies in $[1 - \Delta t(\gamma_{\max} + \alpha_{\max}),\ 1 - \Delta t\gamma_{\min}]$, giving the contraction rate

$$\rho \ \le\ \max\big\{\, 1 - \Delta t\, \gamma_{\min},\ |1 - \Delta t(\gamma_{\max} + \alpha_{\max})| \,\big\}.$$

At our defaults $(\Delta t, \alpha_{\max}, \gamma_{\max}, \gamma_{\min}) = (1, 1.5, 0.4, 10^{-2})$, this evaluates to $\rho \le 0.99$.

Why the leak term is essential. Without it, $\operatorname{sech}^2(h)$ vanishes in saturation and the discrete propagator approaches the identity. The simpler cell $\dot{h} = A_\theta(u_n) \odot \tanh(h) + b_\theta(u_n)$ therefore admits no uniform contraction rate, and so does not satisfy the hypotheses of Theorem 5.

Layer stacking. We stack L such cells as pre-norm residual blocks using the residual-block layout of Transformers and Mamba (Vaswani et al., 2017; Gu and Dao, 2024), with a two-layer MLP encoder and a mean-pooled linear head.

Parametrization of the contracting dynamics. The diagonal vectors $A_\theta(u)$ and $\gamma_\theta(u)$ in (57) share a single hidden layer $\operatorname{pre}(u) = \tanh(W_1^{(g)}u + b_1^{(g)}) \in \mathbb{R}^H$ with $H = D$, and split into two independent sigmoid heads producing $A_\theta \in [-\alpha_{\max}, -\alpha_{\min}]^D$ and $\gamma_\theta \in [\gamma_{\min}, \gamma_{\max}]^D$. Both heads modulate the same diagonal Jacobian $\partial\dot{h}/\partial h$, so coupling them through one feature stage halves the head parameter count without observable loss in accuracy. The forcing $b_\theta(u) = \tanh(\mathrm{MLP}_2(u)) \in (-1, 1)^D$ uses an independent two-layer MLP, since it enters the dynamics additively rather than multiplicatively.

Initialization. All weight matrices are Xavier-normal. We set $b_2^{(A)}, b_2^{(\gamma)} \sim \mathcal{U}(-2, 2)$, which spreads the initial $|A_{ii}(0)|$ and $\gamma_i(0)$ across most of their respective ranges. Other biases are zero; LayerNorm is initialized to scale 1, shift 0.

Residual block. Each of the L blocks ($\ell = 1, \dots, L$ indexing layers throughout this appendix) is a pre-norm residual block with the ODE solve in the residual branch:

$$z_t^{(\ell)} = \mathrm{LN}(h_t^{(\ell)}), \qquad \bar{h}_t^{(\ell)} = \mathrm{ODESolve}(z_{1:T}^{(\ell)}; \theta^{(\ell)})_t,$$
$$h_t^{(\ell+1)} = h_t^{(\ell)} + W_2^{\mathrm{ffn}}\, \mathrm{GELU}(W_1^{\mathrm{ffn}}\bar{h}_t^{(\ell)} + b_1) + b_2,$$

with FFN expansion $r = 2$ and LN $\varepsilon = 10^{-5}$. The cell consumes the post-LN signal z_t as its input u_t, keeping u on a controlled scale throughout depth. The encoder $\mathrm{enc}(x_t) = W_2^{\mathrm{enc}} \tanh(W_1^{\mathrm{enc}}x_t + b_1) + b_2$ lifts $x_t \in \mathbb{R}^{n_{\mathrm{inp}}}$ to the residual stream; the readout mean-pools the final layer over time and applies a linear classifier.

Training. Cross-entropy with AdamW (Loshchilov and Hutter, 2017), weight decay 10^{-4}, peak LR 3×10^{-4} on a warmup–cosine schedule (warmup $= \min(500, T_{\mathrm{steps}}/4)$), global gradient clipping at 1.0, batch size 8, 60 epochs. The backward pass is implemented via a parallel scan on the implicit-adjoint equation. We use the EigenWorms split in Morrill et al. (2021); channels are z-scored using train-set statistics. All runs use a single NVIDIA B200 in fp32. Hyperparameters are summarized in Table 12; only $\gamma_{\min}$ varies across the rows of Table 11.

I.3. Parallel-in-Time Robotic Control and Simulation

Setup. We study trajectory tracking for a planar two-link manipulator, a nonlinear robotic system whose unforced dynamics are the classical double pendulum. In this regime, the dynamics are not uniformly contractive: the system has multiple gravity equilibria and can be chaotic at sufficiently high energy (Shinbrot et al., 1992). Thus, without feedback, the residual merit functional (2) need not admit a T-independent PL constant. The plant has state $x = (q, \dot{q}) \in \mathbb{R}^4$ and dynamics

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + b\dot{q} = u,$$

where $M(q) \succ 0$ is the inertia matrix, $C(q, \dot{q})\dot{q}$ contains Coriolis and centrifugal terms, $G(q)$ is the gravity torque, and $b > 0$ is joint friction.

Neural optimal controller. We evaluate parallel trajectory solvers in the stabilizing-feedback regime. A weak gravity-compensated PD controller serves as a baseline stabilizer, and a neural residual controller supplies the remaining torque, $u(t, x) = u_{\mathrm{PD}}(t, x) + \pi_\theta\big(x(t), q_{\mathrm{ref}}(t), \dot{q}_{\mathrm{ref}}(t)\big)$.

For bounded reference trajectories, the PD-stabilized system is locally incrementally exponentially stable around the tracking trajectory (Lohmiller and Slotine, 1998; Bullo, 2026); hence the corresponding residual minimization problem is locally well-conditioned in the sense of Theorem 1. Details of the controller, objective, and contractivity argument are given in Appendix I.3.1.

We compare a memoryless MLP residual controller against a recurrent controller whose hidden state evolves according to the contractive nonlinear SSM cell of Appendix I.2; the recurrent controller can use trajectory history to compensate for effects not identifiable from the instantaneous state alone. All three learned residuals cut tracking RMSE by 5–6$\times$ over the weak PD baseline (from 0.58 to $\sim$0.10), and the 2.3k-parameter SSM outperforms the 2.7k MLP while trailing the 17.9k MLP by 0.012. Parallel training accelerates the full differentiable training step (forward simulation, implicit-adjoint backward pass, and optimizer update) by 14–28$\times$ with Euler-DEER and 50–74$\times$ with DEER-exp (Appendix K) across the three controllers, at stopping tolerance 10^{-5} with a cap of 50 Newton iterations (converged runs use 6–32), each compared against the sequential integrator of the same discretization. Across five seeds, the contractive SSM controller trains and solves reliably in every run, with the DEER-exp iteration count constant at seven, whereas the MLP controllers are seed sensitive: one large-MLP and three small-MLP seeds either diverge at initialization or drive the closed loop past the solver's iteration cap. Full numbers are in Table 13; training details are in Appendix I.3.1.

Adaptive controller (online gradient descent). Finally, we evaluate parallel-in-time rollouts on a two-link manipulator with an unknown payload under an adaptive controller, where the payload parameter is updated online using a Slotine–Li-style gradient rule (Slotine and Li, 1987). This setting is not globally contractive, but the Lyapunov-stable closed-loop dynamics become locally contractive near the limiting trajectory under standard hyperbolicity assumptions. Figure 13 shows Jacobi rollouts of the closed-loop system with $N = 9{,}000$ integration points: as the iteration proceeds, the rollout converges to the reference path while the online payload-mass estimate is refined, and after about $0.2N$ sweeps it coincides with the sequential rollout. Full details are given in Appendix I.3.2.

Table 13: Neural controller training on a two-link arm tracking a Lissajous reference: $k_p = 3$, $k_d = 0.6$, $dt = 10^{-2}$, $T = 10\,\mathrm{s}$, $N = 1{,}000$, 1,200 AdamW epochs on a single B200. Mean $\pm$ std over the converged seeds out of 5 (SSM $5/5$, MLP large $4/5$, MLP small $2/5$; the same seeds fail for both solvers); failed seeds diverge at initialization or exceed the Newton-iteration cap. Final RMSE is for the DEER-exp-trained controllers. Each learned residual cuts tracking error by 5–6$\times$ over the PD baseline, the SSM matches the MLPs at the smallest parameter count and is the only controller that converges for every seed, and parallel training accelerates all three controllers by 14–74$\times$.

Controller	Final RMSE (↓)	Speedup (DEER-exp) (↑)	Speedup (Euler-DEER) (↑)
PD only	0.584	–	–
MLP large (17.9k params)	0.096 ± 0.001	$66.6\times \pm 20.6$	$13.6\times \pm 1.8$
MLP small (2.7k params)	0.117 ± 0.001	$73.5\times \pm 20.6$	$19.8\times \pm 7.3$
SSM (2.3k params)	0.108 ± 0.001	$50.4\times \pm 0.8$	$28.1\times \pm 2.5$

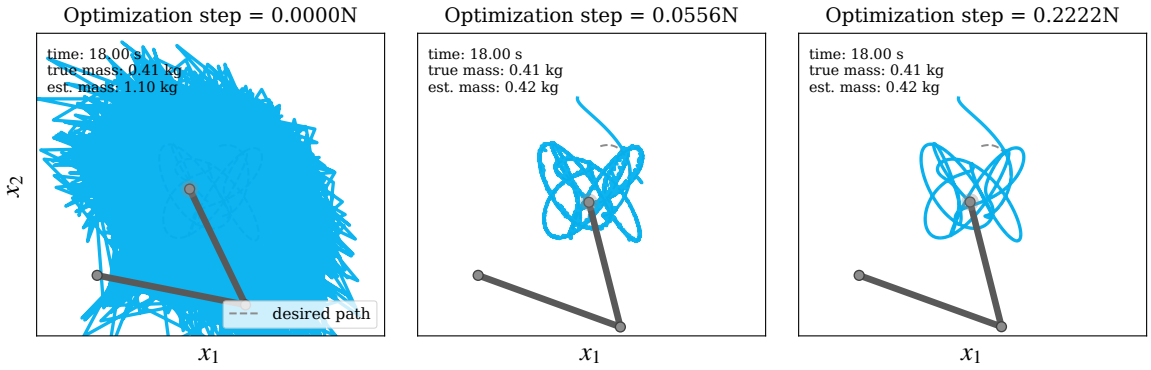


Figure 13: Parallel-in-time Jacobi rollouts for the Slotine–Li adaptive controller (Slotine and Li, 1987) at increasing progress fractions k/N, with $N = 9{,}000$ temporal integration points. From left to right, the rollout trajectory (blue) converges to the reference path (dashed gray), while the terminal robot configuration and adaptive payload-mass estimate improve. All panels share workspace coordinates (x_1, x_2).

I.3.1. NEURAL CONTROLLER DETAILS

The two-link arm obeys

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + b\dot{q} = u.$$

We use the gravity-compensated, mass-scaled PD baseline

$$u_{\mathrm{PD}}(t,x) = G(q) + M(q)\Big[-K_p\big(q - q_{\mathrm{ref}}(t)\big) - K_d\big(\dot{q} - \dot{q}_{\mathrm{ref}}(t)\big)\Big].$$

Writing $\tilde{q} = q - q_{\mathrm{ref}}$ gives

$$\ddot{\tilde{q}} = -K_p\tilde{q} - K_d\dot{\tilde{q}} - M(q)^{-1}\big[C(q,\dot{q})\dot{q} + b\dot{q}\big].$$

For bounded reference trajectories, the leading terms define a damped oscillator and the remaining velocity-product terms are locally bounded perturbations. Hence, for $K_p, K_d \succ 0$, the closed loop is locally incrementally exponentially stable around the reference trajectory (Lohmiller and Slotine, 1998; Bullo, 2026). The weak PD gains used in the experiments stabilize the rollout but intentionally leave substantial tracking error, allowing the learned residual controller to improve performance.

The learned controller is

$$u(t,x) = u_{\mathrm{PD}}(t,x) + \pi_\theta\big(x(t), q_{\mathrm{ref}}(t), \dot{q}_{\mathrm{ref}}(t)\big),$$

and is trained to minimize

$$\mathcal{J}(\theta) = \int_0^T \|q(t) - q_{\mathrm{ref}}(t)\|_2^2 \, dt + \alpha \int_0^T \|\pi_\theta(t)\|_2^2 \, dt.$$

We consider both memoryless MLP policies and recurrent policies. In the recurrent case, a hidden state evolves according to the contractive nonlinear SSM cell of Appendix I.2, driven by $(x, q_{\mathrm{ref}}, \dot{q}_{\mathrm{ref}})$, and the torque is produced by a small MLP readout. All parallel speedups are measured against the corresponding sequential solver; for example, DEER-exp is compared against a sequential exponential integrator.

I.3.2. ADAPTIVE CONTROLLER DETAILS

For the adaptive experiment, suppose the gravity term depends on an unknown payload parameter θ and the robot dynamics are linearly parameterized:

$$M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q,\theta) = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\theta.$$

Given a desired trajectory $q_d(t)$, define

$$\tilde{q} = q - q_d, \qquad \dot{q}_r = \dot{q}_d - \Lambda\tilde{q}, \qquad s = \dot{q} - \dot{q}_r,$$

with $\Lambda \succ 0$. The Slotine–Li controller is

$$\tau = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\theta} - K_D s, \qquad \dot{\hat{\theta}} = -\Gamma Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)^\top s,$$

where $K_D \succ 0$ and $\Gamma \succ 0$. The parameter update is an online gradient-style rule driven by the instantaneous tracking error s.

Let

$$\tilde{\theta} = \hat{\theta} - \theta, \qquad V = \frac{1}{2}s^\top M(q)s + \frac{1}{2}\tilde{\theta}^\top\Gamma^{-1}\tilde{\theta}.$$

Using the standard identity that $\dot{M}(q) - 2C(q,\dot{q})$ is skew-symmetric, the closed-loop derivative satisfies

$$\dot{V} = -s^\top K_D s \le 0.$$

Thus the adaptive controller yields a nonincreasing Lyapunov function. Under persistent excitation, the parameter estimate converges and the tracking error vanishes asymptotically.

This example illustrates that global uniform contractivity is not required for parallel trajectory iteration. Near the limiting closed-loop trajectory, the adaptive robot dynamics can be viewed as a fixed point of the rollout operator. If this fixed point is hyperbolically stable, then the nonlinear rollout is locally stable by the linearization theorem for hyperbolic fixed points (Bullo, 2026, Lemma 1.10). Hence Jacobi-style trajectory iteration can converge rapidly once initialized inside the local region of attraction, even when no global contraction bound is available.

Appendix J. Parallel Scan for Linear ODEs

In this section, we provide a gentle discussion of why the techniques in this paper are, in fact, parallel-in-time.

Our original goal was to solve the differential equation given by (1) to find the solution $x(t)$. To leverage parallel computation over time, we instead make some guess about a candidate trajectory $s^{(0)}(t)$ over the entire time interval, subject only to the constraint given by the initial condition, i.e., $s^{(0)}(0) = x_0$. We can evaluate the quality of this guess by our merit functional given by (2). If we can obtain candidate updates $\delta s^{(i)}(t)$ in a manner that is parallel-in-time, then clearly we can update our trajectories in an embarrassingly parallel manner as

$$s^{(i+1)}(t) = s^{(i)}(t) + \delta s^{(i)}(t).$$

In (11), we propose one such way to obtain the candidate updates: solve the ODE

$$\delta \dot{s} = J(s^{(i)}, t)\, \delta s - r^{(i)}(t).$$

However, the astute reader might ask: if our original goal was to solve the differential equation (1), how have we made our lives any easier by transforming the problem into the repeated solution of the differential equation (11)?

The key distinction is that the differential equation in (1) in general permits *nonlinear* dynamics. In contrast, the differential equation (11) is a *linear time-varying* (LTV) system. Crucially, both $J(s^{(i)}, t)$ and $r^{(i)}(t)$ can be computed in an embarrassingly parallel manner over the entire sequence length. So solving (11) reduces to solving a *linear* dynamical system. Linear dynamical systems can be parallelized with a parallel associative scan (Blelloch, 1990; Kaul, 2020; Smith et al., 2023; Gu and Dao, 2024; Iacob et al., 2025; Gonzalez et al., 2026).

To provide some intuition for what the parallel associative scan is, and how it can parallelize linear dynamical systems, consider solving the simplified LTV system

$$\dot{x}(t) = A(t)x(t) + b(t), \quad x(0) = x_0.$$

If we solve this system numerically, what we seek is some solution $\hat{x}(t) \approx x(t)$ on some time grid $[0, t_1, t_2, \ldots, t_n]$. Thus, most numerical integrators end up discretizing the original continuous-time system into a discrete-time recurrence of the form

$$\hat{x}(t_{i+1}) = \bar{A}(t_i)\hat{x}(t_i) + \bar{b}(t_i), \tag{58}$$

where $\bar{A}(t_i)$ and $\bar{b}(t_i)$ can often be precomputed across the time grid. For instance, in the simplest case of Euler integration with step size h, the discretizations take the form

$$\begin{aligned}\bar{A}(t_i) &= I + hA(t_i)\\ \bar{b}(t_i) &= hb(t_i).\end{aligned}$$

Having obtained the discrete-time recurrence (58), how do we bring the parallel associative scan to bear on it? Well, we observe that (58) gives us "one-step ahead" maps from $\hat{x}(t_i)$ to $\hat{x}(t_{i+1})$. What if we wanted "two-step ahead" maps that took us from $\hat{x}(t_i)$ to $\hat{x}(t_{i+2})$? Well, we could iterate the recurrence to obtain

$$\begin{aligned}\hat{x}(t_{i+1}) &= \bar{A}(t_i)\hat{x}(t_i) + \bar{b}(t_i)\\ \hat{x}(t_{i+2}) &= \bar{A}(t_{i+1})\hat{x}(t_{i+1}) + \bar{b}(t_{i+1}).\end{aligned}$$

Or, we could expand out this recurrence to obtain the "two-step ahead" map directly as

$$\hat{x}(t_{i+2}) = \underbrace{\bar{A}(t_{i+1})\bar{A}(t_i)}_{\text{linear map}}\hat{x}(t_i) + \underbrace{\bar{b}(t_{i+1}) + \bar{A}(t_{i+1})\bar{b}(t_i)}_{\text{offset}}.$$

Crucially, the "two-step ahead" map has the same functional form as the "one-step ahead" map! Therefore, by induction, all n-step ahead maps must also share this functional form, for any n.

In general, we can adopt the notation $\left(\hat{A}(t_i \to t_j), \hat{b}(t_i \to t_j)\right)$ to represent the transition operator from time t_i to time t_j, i.e., our numerical solution $\hat{x}(t)$ satisfies

$$\hat{x}(t_j) = \hat{A}(t_i \to t_j)\hat{x}(t_i) + \hat{b}(t_i \to t_j).$$

Thus, if we obtain all the transition operators $\left\{\left(\hat{A}(0 \to t_i), \hat{b}(0 \to t_i)\right)\right\}_{i=1}^{n}$, then in an embarrassingly parallel manner we can obtain all the nodal values $\{\hat{x}(t_i)\}_{i=1}^{n}$, i.e., solve the linear dynamical system. **The parallel associative scan lets us start with the set of one-step maps $\{\bar{A}(t_i), \bar{b}(t_i)\}_{i=0}^{n-1}$ and, using an algorithm that can be parallelized across the time grid, obtain the full set of transition operators $\left\{\left(\hat{A}(0 \to t_i), \hat{b}(0 \to t_i)\right)\right\}_{i=1}^{n}$.**

To make our notation easier, let's define the transition map

$$\mathcal{T}(t_i \to t_j) \coloneqq \left(\hat{A}(t_i \to t_j), \hat{b}(t_i \to t_j)\right).$$

Notice that we can compose the transition maps according to

$$\begin{aligned}\mathcal{T}(t_j \to t_k) \circ \mathcal{T}(t_i \to t_j) &= \mathcal{T}(t_i \to t_k) \\ &= \left(\hat{A}(t_j \to t_k)\hat{A}(t_i \to t_j), \hat{b}(t_j \to t_k) + \hat{A}(t_j \to t_k)\hat{b}(t_i \to t_j)\right).\end{aligned}$$

With this notation defined, let's consider a time grid of length $n = 8$ (see Table 14). We start out with the one-step associative maps in each position, and then we repeatedly merge neighbors in a divide-and-conquer manner, thus obtaining all 2^k-ahead maps with $\log_2(n)$ iterations. This part is called the "up-sweep" of the parallel scan.

Table 14: **Up-sweep** of the parallel scan. Gray: the one-step maps given at the start; blue: the maps formed in that step.

Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(1 \to 2)$	$\mathcal{T}(2 \to 3)$	$\mathcal{T}(3 \to 4)$	$\mathcal{T}(4 \to 5)$	$\mathcal{T}(5 \to 6)$	$\mathcal{T}(6 \to 7)$	$\mathcal{T}(7 \to 8)$
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(0 \to 2)$	$\mathcal{T}(2 \to 3)$	$\mathcal{T}(2 \to 4)$	$\mathcal{T}(4 \to 5)$	$\mathcal{T}(4 \to 6)$	$\mathcal{T}(6 \to 7)$	$\mathcal{T}(6 \to 8)$
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(0 \to 2)$	$\mathcal{T}(2 \to 3)$	$\mathcal{T}(0 \to 4)$	$\mathcal{T}(4 \to 5)$	$\mathcal{T}(4 \to 6)$	$\mathcal{T}(6 \to 7)$	$\mathcal{T}(4 \to 8)$
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(0 \to 2)$	$\mathcal{T}(2 \to 3)$	$\mathcal{T}(0 \to 4)$	$\mathcal{T}(4 \to 5)$	$\mathcal{T}(4 \to 6)$	$\mathcal{T}(6 \to 7)$	$\mathcal{T}(0 \to 8)$

Finally, with all the 2^k-ahead maps formed, we can effectively use the binary representation of numbers to fill in all the other transition maps we want, again in a maximum of $\log_2(n)$ iterations (this is called the down-sweep, see Table 15). Thus, with n parallel processors, we can evaluate the trajectory over the time grid, i.e., $\{\hat{x}(t_i)\}_{i=1}^{n}$, in $O(\log(n))$ time, leveraging computation that is parallelized over the time grid.

For an alternative presentation of the parallel scan, see Section 2.2 of Gonzalez (2026). For a discussion of how the parallel scan arises in DEER-type methods, see Appendix K.

Table 15: **Down-sweep** of the parallel scan. Blue: the maps formed in that step.

Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(0 \to 2)$	$\mathcal{T}(2 \to 3)$	$\mathcal{T}(0 \to 4)$	$\mathcal{T}(4 \to 5)$	$\mathcal{T}(0 \to 6)$	$\mathcal{T}(6 \to 7)$	$\mathcal{T}(0 \to 8)$
$\mathcal{T}(0 \to 1)$	$\mathcal{T}(0 \to 2)$	$\mathcal{T}(0 \to 3)$	$\mathcal{T}(0 \to 4)$	$\mathcal{T}(0 \to 5)$	$\mathcal{T}(0 \to 6)$	$\mathcal{T}(0 \to 7)$	$\mathcal{T}(0 \to 8)$

Appendix K. DEER, quasi-DEER, and DEER-exp

We give a brief description of the three parallel-in-time Newton variants used in the experiments: Full-DEER (Lim et al., 2024; Danieli et al., 2023) (Figure 11), quasi-DEER (Gonzalez et al., 2024) (Figure 11 and Table 11), and DEER-exp (Table 13). All three apply Gauss–Newton to the discrete residual $\mathbf{r}(\mathbf{s}) = \mathbf{s} - \mathbf{G}(\mathbf{s})$ from Appendix D.3, whose Jacobian is block-bidiagonal,

$$J(\mathbf{s}) = \begin{pmatrix} I_D & & & \\ -\partial_x g_2(s_1) & I_D & & \\ & \ddots & \ddots & \\ & & -\partial_x g_N(s_{N-1}) & I_D \end{pmatrix},$$

so each Newton step $\Delta\mathbf{s} = -J(\mathbf{s})^{-1}\mathbf{r}(\mathbf{s})$ reduces to a linear recurrence in the time index, solved in $O(\log N)$ depth via a parallel associative scan over the per-step Jacobians $\partial_x g_n$. Storing these Jacobians explicitly is what we call *Full-DEER*, costing $O(ND^2)$ memory and $O(ND^3)$ work per Newton iteration; *quasi-DEER* replaces each $\partial_x g_n$ with its diagonal, estimated by a single Hutchinson probe, reducing memory and work to $O(ND)$ at the cost of a coarser linearization. Both variants recover the exact discrete trajectory in at most N Newton steps (Gonzalez et al., 2024, Prop. 1). The variant we call *DEER-exp* (used in Table 13) instead applies continuous-time Gauss–Newton to the merit functional (2) first, then discretizes the resulting linear update with an exponential integrator (Hochbruck and Ostermann, 2010); we derive it next.

Continuous linearization. At Newton iteration i, linearize f around the current trajectory iterate $s^{(i)}(t)$ with

$$A^{(i)}(t) := \partial_x f\big(s^{(i)}(t), t\big), \qquad b^{(i)}(t) := f\big(s^{(i)}(t), t\big) - A^{(i)}(t)\, s^{(i)}(t).$$

The next iterate $s^{(i+1)}$ then satisfies the affine linear ODE

$$\dot{s}^{(i+1)}(t) = A^{(i)}(t)\, s^{(i+1)}(t) + b^{(i)}(t), \qquad s^{(i+1)}(0) = x_0. \tag{59}$$

Discretizing (59) with forward Euler reproduces standard (Euler-)DEER exactly: Euler discretization and Gauss–Newton linearization commute because the Euler map is affine in f.

Exponential-integrator discretization. Freezing $A^{(i)}$ and $b^{(i)}$ at the left endpoint of each interval $[t_n, t_{n+1}]$, the affine ODE (59) admits the closed-form variation-of-constants solution

$$s_{n+1}^{(i+1)} = e^{hA_n}\, s_n^{(i+1)} + h\, \varphi_1(hA_n)\, b_n, \qquad \varphi_1(Z) := \int_0^1 e^{(1-\tau)Z}\, d\tau = \frac{e^Z - I}{Z}, \tag{60}$$

where $A_n := A^{(i)}(t_n)$, $b_n := b^{(i)}(t_n)$, and φ_1 is the first φ-function of exponential-integrator theory. Both e^{hA_n} and $\varphi_1(hA_n)\, b_n$ are recovered from a single matrix-exponential call via the augmented

identity

$$\exp\begin{pmatrix} hA_n & hb_n \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} e^{hA_n} & h\,\varphi_1(hA_n)\,b_n \\ 0 & 1 \end{pmatrix}, \tag{61}$$

which avoids the catastrophic cancellation in $A_n^{-1}(e^{hA_n} - I)\,b_n$ as $hA_n \to 0$.

Parallel scan. Setting $\Delta s_n := s_n^{(i+1)} - s_n^{(i)}$ and substituting into (60) yields the linear recurrence

$$\Delta s_{n+1} = e^{hA_n}\,\Delta s_n + c_n, \qquad c_n := e^{hA_n}\,s_n^{(i)} + h\,\varphi_1(hA_n)\,b_n - s_{n+1}^{(i)}, \tag{62}$$

with $\Delta s_0 = 0$. This has the same form as the Euler-DEER recurrence, with the propagator $I + hA_n$ replaced by e^{hA_n}, and is solved in $O(\log N)$ depth by the associative scan with combining operator

$$(\Phi_2, c_2) \oplus (\Phi_1, c_1) \;:=\; \big(\Phi_2\Phi_1,\; \Phi_2 c_1 + c_2\big).$$

The trajectory is updated as $s^{(i+1)} = s^{(i)} + \Delta s$ and the procedure repeats until $\|c\|_\infty < \varepsilon$. Each Newton iteration costs N matrix exponentials, N Jacobian evaluations, and one parallel scan.

When DEER-exp helps. The exponential propagator preserves the contraction of the underlying ODE: if the linearized dynamics have logarithmic norm $\mu(A_n) < 0$ in some inner product, then $\|e^{hA_n}\| \le e^{h\mu(A_n)} < 1$ for every $h > 0$. By contrast, $I + hA_n$ can have spectral radius greater than one even when A_n is contractive, whenever $h\|A_n\|$ is not small. DEER-exp is therefore preferable when the dynamics are stiff or the step size is coarse relative to $\|A_n\|^{-1}$, at the cost of one matrix exponential per time step per Newton iteration.

Appendix L. Contraction on Average for Incremental Exponential Stability with Overshoot and Global Jacobi Convergence

The convergence guarantee of Theorem 5 rests on the incremental exponential stability assumption $\|\Psi_{\ell,n}(x) - \Psi_{\ell,n}(y)\| \le \kappa\rho^n\,\|x - y\|$ introduced in Appendix D. This appendix presents a concrete, checkable sufficient condition for that assumption, namely a metric (linear matrix inequality, LMI) condition on the per-step Jacobians. The condition may hold even when some individual maps g_n are *expansive*, and therefore extends Theorem 5 to systems contracting only *on average*.

Throughout, $\{g_n\}_{n\ge1}$, $g_n(\cdot) : \mathbb{R}^D \to \mathbb{R}^D$, are the maps of the discrete system (31), assumed continuously differentiable, with Jacobians $J_n(x) := \frac{\partial g_n}{\partial x}(x) \in \mathbb{R}^{D\times D}$. For $P_0 \succ 0$, we write $\|z\|_{P_0} := \left\|P_0^{1/2} z\right\|_2 = \sqrt{z^\top P_0\, z}$ for the weighted Euclidean norm and $\|M\|_{P_0} := \left\|P_0^{1/2} M P_0^{-1/2}\right\|_2$ for its induced matrix norm, so that the standard equivalence $\|M\|_{P_0} \le c \iff M^\top P_0\, M \preceq c^2 P_0$ holds. As in Appendix D, $\Psi_{\ell,n} := g_{\ell+n} \circ \cdots \circ g_{\ell+1}$ (with $\Psi_{\ell,0} = \mathrm{Id}$) denotes the n-step flow map from time ℓ, and $\mathbf{s}^* = (x_1, \ldots, x_N)$ the exact trajectory.

Definition 8 (Contraction on average) *The discrete system* (31) *is* contracting on average with rate $\rho \in [0, 1)$ *if there exist a positive-definite matrix* $P_0 \succ 0$ *and a* bounded *weight sequence* $\{w_n\}_{n\ge0} \subset \mathbb{R}$, *with* $w_{\min} := \inf_{n\ge0} w_n$ *and* $w_{\max} := \sup_{n\ge0} w_n$ *finite, such that each time-varying metric* $P_n := e^{w_n} P_0$, $n \ge 1$, *satisfies*

$$J_n(x)^\top P_n\, J_n(x) \;\preceq\; \rho^2\, P_{n-1} \qquad \textit{for all } x \in \mathbb{R}^D. \tag{63}$$

The time-varying metric $P_n = e^{w_n} P_0$ is the discrete-time version of the time-dependent metric $P(t) = e^{w(t)} P_0$ proposed by Pogromsky and Matveev (2013, 2016). Inequality (63) is the discrete-time contraction condition in the metric P_n; dividing by $e^{w_n} > 0$ shows it is equivalent to the per-step gain bound

$$\|J_n(x)\|_{P_0} \;\le\; \rho_0(n) := \rho\, e^{(w_{n-1}-w_n)/2} \qquad \text{for all } x \in \mathbb{R}^D. \tag{64}$$

Boundedness of $\{w_n\}_{n\ge 0}$ is essential and has three consequences, developed in Lemma 9. First, it keeps the metrics uniformly equivalent, $e^{w_{\min}} P_0 \preceq P_n \preceq e^{w_{\max}} P_0$, so no single step expands by more than a fixed factor. Second, and this is the precise sense of "on average", although the gains $\rho_0(n)$ may exceed 1 at locally expansive steps (when $w_{n-1} > w_n$), the weights telescope, so their long-run geometric mean is at most ρ. Third, it makes the overshoot factor $\kappa := e^{(w_{\max}-w_{\min})/2}$ of the resulting guarantee finite and independent of the time horizon N.

Lemma 9 (Properties of systems contracting on average) *Suppose the discrete system (31) is contracting on average with rate $\rho \in [0,1)$, metric P_0, and bounded weights $\{w_n\}_{n\ge 0}$ with $w_{\min} = \inf_n w_n$, $w_{\max} = \sup_n w_n$, and overshoot factor $\kappa := e^{(w_{\max}-w_{\min})/2}$, as in Definition 8. Then:*

(i) (One-step gains and uniform metric equivalence.) *Each map g_n is globally Lipschitz in the norm $\|\cdot\|_{P_0}$ with constant $\rho_0(n) = \rho\, e^{(w_{n-1}-w_n)/2}$, that is, $\|g_n(x) - g_n(y)\|_{P_0} \le \rho_0(n)\, \|x-y\|_{P_0}$ for all $x, y \in \mathbb{R}^D$. Boundedness of $\{w_n\}_{n\ge 0}$ moreover yields the uniform bounds*

$$\rho\,\kappa^{-1} \le \rho_0(n) \le \rho\,\kappa \qquad \textit{and} \qquad e^{w_{\min}} P_0 \preceq P_n \preceq e^{w_{\max}} P_0, \qquad \textit{for all } n, \tag{65}$$

with constants independent of n and of the time horizon N.

(ii) (Contraction on average.) *The per-step gains have long-run geometric mean at most ρ: for every $m \ge 1$,*

$$\Big(\prod_{n=1}^{m} \rho_0(n)\Big)^{1/m} = \rho\, e^{(w_0-w_m)/(2m)} \le \rho\,\kappa^{1/m}, \tag{66}$$

so that, using only the boundedness of $\{w_n\}_{n\ge 0}$,

$$\limsup_{m\to\infty} \Big(\prod_{n=1}^{m} \rho_0(n)\Big)^{1/m} \le \rho, \qquad \textit{equivalently} \qquad \limsup_{m\to\infty} \frac{1}{m} \sum_{n=1}^{m} \log \rho_0(n) \le \log \rho. \tag{67}$$

Thus, although individual gains may exceed 1, their geometric mean over a long window does not exceed ρ.

(iii) (Incremental exponential stability with overshoot.) *The flow maps contract up to a uniform overshoot: for all $0 \le \ell \le N-1$, $1 \le n \le N-\ell$, and $x, y \in \mathbb{R}^D$,*

$$\|\Psi_{\ell,n}(x) - \Psi_{\ell,n}(y)\|_{P_0} \;\le\; \kappa\,\rho^n\, \|x-y\|_{P_0}\,. \tag{68}$$

In particular, contraction on average implies the hypothesis of Theorem 5 in the norm $\|\cdot\|_{P_0}$, with overshoot factor κ and the same rate ρ, both independent of N.

(iv) (Global Jacobi convergence.) *Let $\{\mathbf{s}^{(i)}\}$ be the iterates of the nonlinear Jacobi iteration (32). Writing $\|\mathbf{s}\|_{\infty,P_0} := \max_{1\le n\le N} \|s_n\|_{P_0}$,*

$$\left\|\mathbf{s}^{(i)} - \mathbf{s}^*\right\|_{\infty,P_0} \;\le\; \kappa\,\rho^i\, \left\|\mathbf{s}^{(0)} - \mathbf{s}^*\right\|_{\infty,P_0} \qquad \textit{for every } i \ge 0, \tag{69}$$

so $\mathbf{s}^{(i)} \to \mathbf{s}^*$ *geometrically; to reach precision* $\epsilon > 0$ *it suffices to take* $i \geq \big(\log\kappa + \log(\|\mathbf{s}^{(0)} - \mathbf{s}^*\|_{\infty,P_0}/\epsilon)\big)/\log(1/\rho)$ *iterations, with* $\log\kappa = (w_{\max} - w_{\min})/2$*, independently of the horizon* N*.*

Proof (i) Fix $n \geq 1$ and $x \in \mathbb{R}^D$. Since $e^{w_n} > 0$, (63) is equivalent to $J_n(x)^\top P_0\, J_n(x) \preceq \rho^2 e^{w_{n-1}-w_n} P_0 = \rho_0(n)^2 P_0$, which by the norm/LMI equivalence is exactly the gain bound (64), $\|J_n(x)\|_{P_0} \leq \rho_0(n)$. As g_n is continuously differentiable, the fundamental theorem of calculus along the segment $y + t(x-y)$, $t \in [0,1]$, gives $g_n(x) - g_n(y) = \big(\int_0^1 J_n(y + t(x-y))\, dt\big)(x-y)$, so by the triangle inequality for the integral and the uniform bound (64),

$$\|g_n(x) - g_n(y)\|_{P_0} \leq \Big(\int_0^1 \big\|J_n\big(y + t(x-y)\big)\big\|_{P_0}\, dt\Big)\, \|x-y\|_{P_0} \leq \rho_0(n)\, \|x-y\|_{P_0}\,.$$

The uniform bounds (65) follow from $w_{\min} \leq w_n \leq w_{\max}$, which gives $|w_{n-1} - w_n| \leq w_{\max} - w_{\min} = 2\log\kappa$ and $e^{w_{\min}} \leq e^{w_n} \leq e^{w_{\max}}$.

(ii) Taking logarithms in (64), $\log\rho_0(n) = \log\rho + \frac{1}{2}(w_{n-1} - w_n)$, so the average over $n = 1, \ldots, m$ telescopes:

$$\frac{1}{m}\sum_{n=1}^{m} \log\rho_0(n) = \log\rho + \frac{1}{2m}\sum_{n=1}^{m}(w_{n-1} - w_n) = \log\rho + \frac{w_0 - w_m}{2m},$$

which exponentiates to (66). Since $w_0 - w_m \leq w_{\max} - w_{\min} = 2\log\kappa$, the correction $(w_0 - w_m)/(2m)$ is at most $(\log\kappa)/m \to 0$ as $m \to \infty$, which gives (67); boundedness of $\{w_n\}_{n\geq 0}$ is precisely what forces this correction to vanish.

(iii) The flow map $\Psi_{\ell,n} = g_{\ell+n} \circ \cdots \circ g_{\ell+1}$ composes n maps, the j-th of which is $\rho_0(\ell + j)$-Lipschitz in $\|\cdot\|_{P_0}$ by part (i). Since the Lipschitz constant of a composition is the product of the factors, and the product telescopes as in part (ii),

$$\|\Psi_{\ell,n}(x) - \Psi_{\ell,n}(y)\|_{P_0} \leq \Big(\prod_{j=\ell+1}^{\ell+n} \rho_0(j)\Big)\, \|x-y\|_{P_0} = \rho^n\, e^{(w_\ell - w_{\ell+n})/2}\, \|x-y\|_{P_0}\,.$$

As $w_\ell - w_{\ell+n} \leq w_{\max} - w_{\min} = 2\log\kappa$, the exponential factor is at most κ, which gives (68). This is precisely the incremental exponential stability bound assumed in Theorem 5, with norm $\|\cdot\|_{P_0}$, overshoot κ, and rate ρ.

(iv) By part (iii) the hypotheses of Theorem 5 hold with $\|\cdot\| = \|\cdot\|_{P_0}$ and overshoot κ; its bound (34) is then exactly (69), because the block maximum norm built from $\|\cdot\|_{P_0}$ is $\|\cdot\|_{\infty,P_0}$. The iteration count is (36). Finally, the iterate formula (35) of Appendix D gives $s_n^{(i)} = x_n$ for all $n \leq i$, so $\mathbf{s}^{(i)} = \mathbf{s}^*$ once $i \geq N$. ■

Remark 10 *Two limiting cases recover familiar settings. If the weights are constant,* $w_n \equiv \bar{w}$*, then* $w_{\min} = w_{\max}$*,* $\kappa = 1$*, and* $\rho_0(n) \equiv \rho$*: the metric is time-invariant (up to the fixed scale* $e^{\bar{w}}$*), every* g_n *is a genuine contraction, and* (68) *reduces to the pointwise-contractive hypothesis of Theorem 5. For a linear time-varying system* $g_n(x) = A_n x + b_n$ *the Jacobians are constant,* $J_n(x) \equiv A_n$*, condition* (63) *becomes the finite family of LMIs* $A_n^\top P_n A_n \preceq \rho^2 P_{n-1}$*, and the Lipschitz bounds of parts (i) and (iii) hold with equality up to the metric, so the block error propagates exactly through the products* $A_{\ell+n} \cdots A_{\ell+1}$*.*