OVERCOMING STIFFNESS BARRIERS IN NEURAL ODE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vanishing sensitivities to fast decay rates can hinder shooting based training of stiff neural ODEs. We investigate how established representation and training techniques can be combined to address this difficulty. Specifically, we avoid differentiation through an initial value solve by combining a production rate parameterization with a derivative matching initialization and simultaneous collocation. Analytically, we show for a scalar decay model that transformed derivative matching retains a restoring gradient for overestimated rates and further that a joint collocation step maintains a nonvanishing log rate correction while the corresponding shooting step vanishes. Experimentally, we observe on six stiff benchmarks benefits from pairing the architecture and training procedure, including improved spectral recovery on Van der Pol and the Oregonator, although recovery remains mixed across systems. We also show via counterexample a separate identification limit: exact trajectory and derivative fits don't imply identification of fast Jacobian eigenvalues. Together, these results present an avenue for addressing vanishing sensitivities through training formulation, while showing why improvements in optimization must be evaluated separately from exact recovery of the underlying dynamics.

1 INTRODUCTION

Neural ordinary differential equations (neural ODEs) learn continuous-time dynamics by parameterizing the vector-field of an ODE with a neural network

$$\frac{du}{dt} = f_\theta(t, u(t)), \qquad u(t_0) = u_0, \tag{1}$$

and obtaining trajectories by numerical integration (Chen et al., 2018). Trajectories can be evaluated at arbitrary times, making neural ODEs well suited for scientific system identification, but learning stiff dynamics remains challenging (Shampine & Gear, 1979; Hairer & Wanner, 1996; Kim et al., 2021). Although there is no single agreed upon mathematical definition of stiffness, many characterize stiffness as the presence of rapidly damped perturbations occurring far shorter than the solution's relevant timescale (Söderlind et al., 2014). These rapidly decaying modes can impose restrictive stability requirements on numerical integration while having little influence on the trajectory after an initial transient. Their recovery therefore depends on how the dynamics are represented, how parameters are estimated, and what information the observations contain.

Standard shooting based training estimates parameters by solving an initial value problem and then differentiating the resulting predictions with respect to those parameters. In particular, for stiff systems, one typically uses stable solvers (often implicit methods) to relax the stability restrictions imposed by fast modes. However, stable numerical integration does not necessarily produce useful parameter sensitivities. Indeed, Fronk & Petzold (2025b) analyze how sensitivities to fast decay rates vanish through the stability functions of A-stable stiff integration methods. An excessively fast candidate rate can produce predictions that are nearly insensitive to further rate changes even when those predictions disagree with the observations. These findings motivate a careful examination of the training formulation to better understand which features make this an obstacle to learning. Can useful corrections to fast rates be retained by reformulating the optimization so that parameter updates no longer require differentiation through an initial value solve?

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

We investigate this question by addressing two related difficulties that stiffness creates for neural ODEs: a *representation barrier*, in what the vector-field must be able to express, and a *sensitivity barrier*, in whether training can reach the parameters that express it. The chosen parameterization couples the difficulties because it determines which rates can be expressed and how changing a parameter affects the rates. Our approach, therefore, pairs a representation suited for widely separated scales with training that acts through local dynamics residuals.

To address the sensitivity barriers, we initialize the model by derivative matching at observed states and then jointly optimize trajectory variables and network parameters through simultaneous collocation. The parameters enter the residuals through direct evaluations of the vector field and therefore provide parameter updates without differentiation through an initial value solve.

The representation barrier matters because the local rates that training seeks to recover may differ by orders of magnitude in stiff systems. Existing approaches improve the representation of widely separated scales through scaling (Kim et al., 2021), transformations of the stiff dynamics (Caldana & Hesthaven, 2025), or structured decompositions that isolate stiff components (Loya et al., 2025). Following the production-destruction view of stiff kinetics (Zinihi et al., 2026), we write

$$f_\theta(u) = P_\theta(u) + R_\theta(u) \odot u. \tag{2}$$

Here, P_θ is an additive production term and R_θ is a state-proportional rate that supplies damping or amplification. Data-derived scales and output transformations allow moderate raw network outputs to represent physical coefficients spanning many orders of magnitude. The rate term contributes directly to the local Jacobian, whose full spectrum also depends on the state derivatives of both terms.

We provide theoretical support for these training choices by examining the scalar decay model with an overestimated candidate rate. We find that the transformed derivative loss maintains a restoring gradient in the log-rate coordinate. With initialization from the observations and fixed positive damping, a joint collocation step also retains a nonvanishing proposed log rate correction where the corresponding damped shooting step vanishes. These results identify a mechanism through which the reformulation can address the sensitivity barrier. We also examine what successful fitting establishes about the dynamics. A counterexample shows that even exact trajectory and derivative fits can leave a fast Jacobian eigenvalue undetermined.

Our contributions are threefold. We pair established representation and training techniques to address vanishing shooting sensitivities and provide theoretical support for their correction mechanisms (cf. Section 3). We demonstrate, through a counterexample and numerical experiments, that accurate trajectory reconstruction can coexist with poor recovery of fast Jacobian eigenvalues. Finally, we compare the proposed combination with alternative architectures and training procedures on six stiff benchmarks, assessing benefits and limitations through trajectory, vector-field, and spectral errors (cf. Section 4).

2 Related Work

Work on stiff neural ODEs has addressed both numerical integration and the computation of parameter sensitivities. Kim et al. (2021) combine scaling with stabilized sensitivity calculations, and Fronk & Petzold (2024; 2025a) study training with implicit single step and explicit exponential integrators. Zhang & Zhao (2023) use discrete adjoints and check-pointing to compute consistent gradients efficiently, including through implicit integration methods. Pal et al. (2021) use local error and stiffness estimates to encourage dynamics that are cheaper to integrate. These approaches retain training through numerical trajectories. Fronk & Petzold (2025b) analyze how sensitivities to fast decay rates vanish through the stability functions of stiff methods. An accurately computed shooting gradient can therefore still provide little information for correcting those rates.

Derivative fitting and simultaneous collocation provide alternative ways to constrain the learned vector-field. Roesch et al. (2021) fit the vector-field to state–derivative pairs obtained from a collocation approximation. Huang et al. (2025) combine a related derivative fit with normalization and use the result to initialize solver-based neural-ODE training. Peng et al. (2025) similarly use derivative fitting as an intermediate stage. An initial neural ODE fit gives estimated derivatives for training a chemical reaction network with prescribed reaction pathways, followed by refinement with numerical integration. Shapovalova & Tsay (2025) instead optimize a polynomial trajectory and network

parameters jointly, while Langenkamp et al. (2025) extend direct collocation to physics-enhanced neural ODEs using a high-order Radau transcription. These simultaneous formulations allow parameters to enter local dynamics residuals directly.

Representation choices complement these training strategies. Kim et al. (2021) use deep networks with rectified activations and scale the network outputs and losses. Caldana & Hesthaven (2025) learn a time reparameterization from the adaptive steps of an implicit reference solver, producing a nonstiff surrogate that can be integrated explicitly. Loya et al. (2025) learn a linear–nonlinear decomposition, constrain the spectrum of the linear part, and use an exponential integrator. Related work in chemical kinetics transforms reaction rates with a latent asinh map (Döppel & Votsmeier, 2023) or imposes a gain–loss form with a positivity-preserving discretization (Zinihi et al., 2026). These approaches provide explicit mechanisms for handling separated scales and constraints. The recovery of fast dynamics also depends on the information present in the observations. Valorani (2026) combines scaling, excitation of fast directions, and Jacobian diagnostics to improve spectral recovery, emphasizing that trajectory accuracy alone can conceal errors in fast modes.

3 METHOD

We construct a production rate vector-field (3.1) and estimate its parameters in two stages (3.2). We are given observations

$$\mathcal{D} = \{(t_i, y_i)\}_{i=1}^{N}, \qquad y_i \in \mathbb{R}^d,$$

sampled from an unknown dynamical system at times $t_1 < \cdots < t_N$. Our goal is to learn a neural ODE f_θ defining:

$$\frac{du}{dt} = f_\theta(t, u).$$

For the autonomous systems considered in this work, we suppress the time argument and write $f_\theta(u)$, with θ being the trainable parameters of the neural network.

3.1 PRODUCTION–RATE PARAMETERIZATION

We parameterize the vector-field as

$$f_\theta(u) = P_\theta(u) + R_\theta(u) \odot u, \tag{3}$$

where P_θ is a signed additive term and R_θ is a state-dependent rate. The additive term can remain nonzero when a state coordinate vanishes, while the rate term provides a direct mechanism for the network to apply state proportional damping or amplification. Here $\odot$ and $\oslash$ denote coordinatewise multiplication and division.

The network receives two views of the state: a linear normalization $\widehat{u}$ that preserves variation over the observed coordinate ranges, and an asinh compressed view z that accommodates differences in magnitude. We concatenate these as $\phi(u) = \begin{bmatrix} \widehat{u} \\ z \end{bmatrix} \in \mathbb{R}^{2d}$. A shared network maps $\phi(u)$ to bounded production and rate outputs α_θ and β_θ, as shown in Figure 1. The exact input normalizations are given in Appendix B.1.

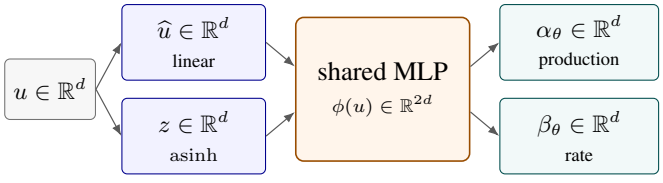


Figure 1: Two-view, two-head production–rate network.

To scale the outputs, let s_j be the largest absolute secant slope between consecutive observations and let $y_{\text{scale},j}$ denote the observed range of coordinate j. Following the data dependent scaling strategy of Kim et al. (2021), we use the following three scalings

$$s_j = \max_{1 \le i < N} \left| \frac{y_{i+1,j} - y_{i,j}}{t_{i+1} - t_i} \right|, \qquad \kappa = 10^{-3} s, \qquad \rho = s \oslash y_{\text{scale}},$$

with s as in Appendix B.1, along with numerical safeguards. The scales κ and ρ have units of state per unit time and inverse time respectively.

The default negative rate parameterization is

$$f_\theta(u) = \underbrace{\kappa \odot \sinh\big(\alpha_\theta(\phi(u))\big)}_{P_\theta(u)} + \underbrace{\big[-\rho \odot \exp\big(\beta_\theta(\phi(u))\big)\big] \odot u}_{R_\theta(u)\odot u}. \tag{4}$$

We refer to this production–rate architecture, together with the input and output transformations described above, as the StiffNet. These transformations allow moderate network outputs to represent coefficients spanning many orders of magnitude. In particular, β_θ parameterizes the logarithm of the rate magnitude relative to ρ, connecting the architecture to the log rate coordinate used in our scalar analysis. The signed rate variant uses $R_\theta(u) = \rho \odot \sinh(\gamma_\theta(\phi(u)))$, allowing either damping or amplification. Output bounds and details of the two variants are given in Appendices B.1-B.2.

Lastly, observe that the rate term contributes directly to the local linearization because

$$J_{f_\theta}(u) = J_{P_\theta}(u) + \operatorname{diag}\big(R_\theta(u)\big) + \operatorname{diag}(u) J_{R_\theta}(u).$$

The rate term therefore provides an explicit mechanism for large local decay rates. The Jacobian spectrum, however, depends on all three terms, so a negative state proportional rate alone does not guarantee stability of the learned dynamics.

3.2 Training

3.2.1 Derivative-matching warm start

Derivative matching provides an initial estimate of θ for the subsequent collocation solve (Roesch et al., 2021). Using finite-difference estimates $\widehat{y}$ of the observed derivatives, we minimize

$$L_D(\theta) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \left[\operatorname{asinh}\left(\frac{f_\theta(t_i, y_i)_j}{\kappa_j} \right) - \operatorname{asinh}\left(\frac{\widehat{y}_{i,j}}{\kappa_j} \right) \right]^2. \tag{5}$$

Here, Ω indexes the derivative targets retained after filtering. The loss compares the vector-field with estimated slopes at the observed states, so its parameter dependence does not pass through a predicted trajectory. We apply the same scaled asinh transformation to both predictions and targets which makes the comparison approximately linear near zero and logarithmic at large magnitudes. The scalar analysis below examines how this transformation interacts with the log rate parameterization to retain a restoring gradient for overestimated rates. The finite-difference estimator and target-filtering rule are given in Appendix C.1.

3.2.2 Radau IIA collocation

Collocation represents the trajectory on a finite mesh and enforces the ODE through algebraic residuals at selected points within each mesh interval. We use a two stage Radau IIA transcription on a mesh containing all observation times. With $D = \operatorname{diag}(y_{\text{scale}})$, we jointly optimize the normalized mesh states $\widehat{U}_i = D^{-1}U_i$, normalized stage increments $\widehat{K}_{i\ell}$, and network parameters θ. On an interval of width h_i, the stage increments are required to agree with $h_i D^{-1} f_\theta(t_{i\ell}, V_{i\ell})$, where $V_{i\ell}$ is the physical state at Radau stage ℓ. The stage-state construction and residual scalings are given in Appendices E.3 and C.3, respectively.

Stage residuals measure the disagreements between the normalized increments and the scaled vector-field evaluations f_θ. Continuity residuals connect adjacent mesh states, and data residuals anchor the trajectory to the observations. We collect these three residuals into $\mathcal{R}_{\text{coll}}$ and solve

$$\min_{U,K,\theta} \|\mathcal{R}_{\text{coll}}(U, K, \theta)\|_2^2. \tag{6}$$

with Levenberg-Marquardt (LM) Levenberg (1944); Marquardt (1963). We initialize the mesh states by interpolating the observations and stage increments from the changes between adjacent mesh states.

After each collocation solve, we check where the fitted trajectory is least consistent with the dynamics. We sum the squared stage and continuity residuals within each observation interval and add

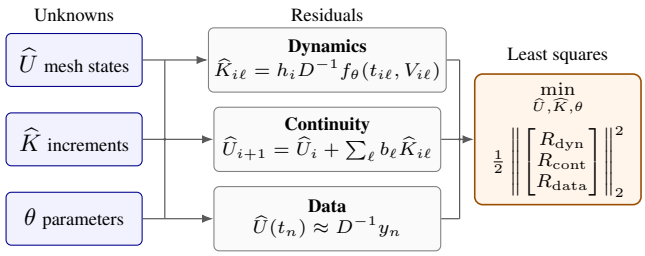


Figure 2: Collocation as neural ODE training.

mesh points in intervals with large totals, doubling their number of subdivisions. We then repeat the solve on the finer mesh, carrying forward the network parameters and reinitializing the trajectory variables from the observations. The refinement threshold and subdivision limits are given in Appendix C.4.

To choose the final model, we compare rollouts of the learned vector-fields from all refinement rounds and retain the parameters with the lowest trajectory error on the training interval. We note that the rollout errors are used for model selection and do not enter either training objective.

3.3 Scalar Analysis

Fronk & Petzold (2025b) use the scalar test equation $\frac{du}{dt} = -au$ to study the vanishing of parameter sensitivities as a increases. To examine the implications for training, fix a true rate $a_* > 0$, which generates the observations, and a reference scale $\rho > 0$. Write the candidate rate as $a := \rho e^{\beta}$, with $\beta \in \mathbb{R}$ and define $\beta_* = \log(a_*/\rho)$. An excessively large candidate rate can produce nearly zero predictions at the observation times even when the observed states remain nonzero. Further increases in the rate then have little effect on those predictions, leaving a poorly fitting model with a small shooting gradient. Holding the observations fixed, we examine how derivative matching and collocation retain a correction in this case.

Proposition 1. *For fixed states $\{y_i\}_{i=1}^m$ and exact targets $d_i = -a_* y_i$, let $L_D(\beta)$ denote the scalar version of equation 5. Assume $\kappa > 0$ and $y_i \neq 0$, and define*

$$\eta = \frac{1}{m} \sum_i \frac{d_i^2}{\kappa^2 + d_i^2}.$$

For $\beta \geq \beta_*$,

$$L_D''(\beta) \geq 2\eta \qquad L_D'(\beta) \geq 2\eta(\beta - \beta_*)$$

The transformed loss therefore retains a restoring gradient that grows at least linearly with the excess log rate. For $|d_i| \gg \kappa$, the corresponding squared residual is approximately $(\beta - \beta_*)^2$ on the overestimation half-line, so the loss measures the log-rate error directly in this regime.

Similar to derivative matching, collocation also evaluates local dynamics, however, it has extra degrees of freedom because its trajectory variables can move. We therefore should check whether a joint step still corrects the rate when it could instead use the gradient to adjust the states. We consider one two stage Radau IIA interval of fixed width $h > 0$, with a fixed normalized initial state u_0 and endpoint observation y, where $u_0 > y > 0$. Initialize the endpoint variable at y and both normalized stage increments at $y - u_0$. The residual weights are positive and fixed, and the LM step uses fixed damping μI, with $\mu > 0$. Let $\delta\beta_C$ denote the log rate component of the initial joint collocation proposal, and let $\delta\beta_S$ denote the corresponding shooting correction using the same Radau formula, observation weight, and damping.

Proposition 2. *At this initialization as $a \to \infty$,*

$$\delta\beta_C \to -c \qquad \delta\beta_S = O((ha)^{-1}),$$

for a constant $c \in (0, 1)$. The full joint collocation proposal also decreases the nonlinear collocation objective for all sufficiently large a.

Taking the full initial proposal gives $a_{\text{new}}/a = e^{\delta\beta_C} \to e^{-c} < 1$, so the rate is reduced by a fixed proportion even when the trajectory variables are free to move. Full proofs and the analysis of derivative target error are given in Appendix E.

The following example illustrates a separate identification limit. Consider the production rate family

$$x' = -x, \qquad y' = (a-1)x - ay,$$

with $a > 1$. From $(x(0), y(0)) = (1, 1)$, every member produces the same trajectory $x(t) = y(t) = e^{-t}$, with identical derivatives for every a, yet has Jacobian eigenvalues -1 and $-a$. Thus, the fast transverse rate is undetermined by these observations, so improving the optimization cannot resolve the ambiguity. Even exact trajectory and derivative fits, therefore, provide no general guaranty of spectral recovery (cf. Appendix E.4).

4 Numerical Experiments

Our experiments examine the practical benefit of the formulation motivated by the vanishing shooting sensitivities. We compare parameterizations and training procedures, examining trajectory accuracy, recovery of local dynamics and training reliability. We consider six autonomous stiff benchmarks (Hairer & Wanner, 1996): the Van der Pol oscillator (VDPOL) (van der Pol Jun., 1926), the air pollution model (POLLU) (Verwer, 1994), Robertson's reaction system (ROBER) (Robertson, 1966), the Oregonator (OREGO) (Field & Noyes, 1974), the high irradiance response (HIRES) (Schäfer, 1975), and the Davis–Skodje system (Davis & Skodje, 1999). The abbreviations VDPOL, ROBER, OREGO and HIRES follow Hairer & Wanner (1996), which also fixes the initial conditions and integration intervals used throughout; full definitions are given in Appendix A.

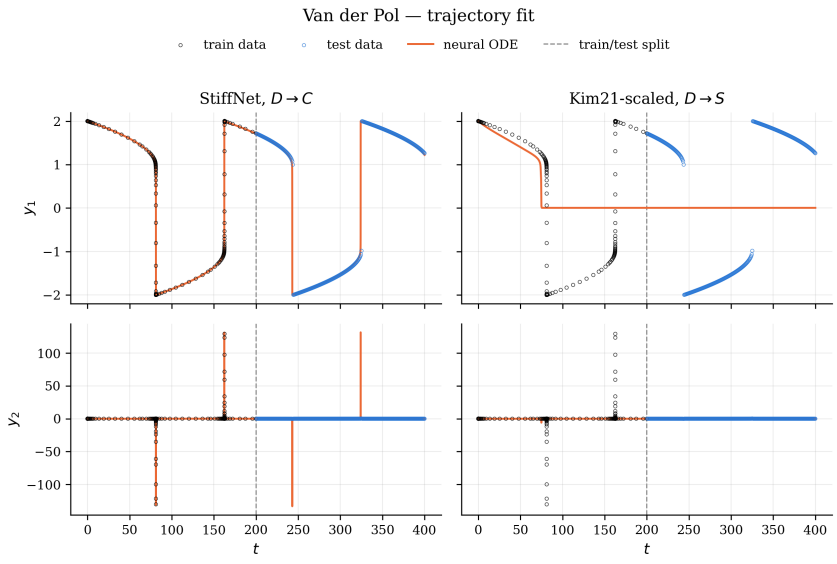


Figure 3: Van der Pol ($\mu = 100$) trajectory fits for StiffNet trained with derivative matching + collocation along with Kim21-scaled trained with derivative matching + shooting.

Each benchmark possesses complementary manifestations of stiffness (see Appendix A.7). ROBER tests widely separated time scales, while VDPOL and OREGO test oscillatory dynamics with rapid transitions. HIRES and POLLU test coupled reaction systems, with POLLU extending the comparison to 20 state variables with rates spanning roughly 10^{-3} to 10^{12}. Davis–Skodje has a fast decay rate controlled by $1/\varepsilon$. We use logarithmically spaced observations for ROBER, POLLU, HIRES, and Davis-Skodje, and observations drawn from adaptive stiff solver steps for VDPOL and OREGO. Sampling details are given in Appendix B.5.

We compare the StiffNet with a plain MLP baseline and the GELU-based scaled model described in Kim et al. (2021). We will refer to the latter as Kim21. For each benchmark, all three architectures

Table 1: Minimal trajectory error on the Rober training trajectory. Lower value is better. Best performance is highlighted in **bold**.

	Single			Combined	
Architecture	D	C	S	$D \to S$	$D \to C$
Kim21	1.177	4.682	6.993	1.063	1.177
Baseline MLP	1.081	1.641	1.212	1.29	1.081
StiffNet	1.293	1.207	1.117	.1423	**.00436**

have a similar number of trainable network parameters. Network dimensions and parameter counts are given in Appendix B.4.

To train the Neural ODEs we use combinations of three different methods denoted by D for derivative matching, C for the simultaneous Radau IIA collocation, and S for single shooting. The sequential combinations $D \to C$ and $D \to S$ use derivative matching to initialize the second training stage. We report a set of metrics and interpret them in Appendix D.1.

4.1 Results

We first examine whether the second training stage improves on derivative matching. Table 1 compares D, C, and S individually with $D \to C$ and $D \to S$ on the ROBER test trajectory. For StiffNet, $D \to C$ reduces the reported trajectory error from 1.293 after derivative matching to 0.113, while $D \to S$ remains at 1.293. Collocation without the warm start gives 1.207. The corresponding $D \to C$ results for the MLP and Kim21 networks show no reported improvement over the derivative matching. The lowest reported error therefore comes from combining the parameterization, initialization, and collocation stage, consistent with the scalar correction mechanism in Proposition 2.

Table 2: **Median** extrapolation errors across six stiff ODE benchmarks with 50 experiments per benchmark. We report trajectory error (Traj.), vector-field error (VF), and spectral error (Spec.). Lower values indicate better performance, and **best results are highlighted in bold**

		D → S			D → C		
Problem	**Metric**	**MLP**	**Kim21**	**StiffNet**	**MLP**	**Kim21**	**StiffNet**
VDPOL	Traj.	7.064e+03	7.408e+00	4.509e+00	5.327e+03	3.105e+00	**2.032e+00**
	Spec.	3.853e-01	7.180e-01	2.031e-01	3.753e-01	4.950e-01	**6.033e-02**
	VF	6.858e+00	1.149e+03	6.558e+00	3.879e+00	6.510e+00	**3.860e-01**
POLLU	Traj.	–	–	–	7.916e+36	1.272e+02	**1.451e-02**
	Spec.	–	–	–	X	X	X
	VF	–	–	–	6.351e+28	**2.303e+03**	6.251e+07
ROBER	Traj.	4.082e+03	5.918e+20	1.146e+01	1.420e+11	2.495e+21	**1.100e+00**
	Spec.	2.590e+29	**1.000e+00**	5.277e+23	9.300e+27	1.000e+00	2.148e+23
	VF	2.810e+17	**5.316e+05**	4.315e+09	5.663e+16	6.673e+05	5.717e+09
OREGO	Traj.	9.999e-01	9.983e-01	**9.821e-01**	1.500e+02	5.467e+00	1.202e+00
	Spec.	3.811e+00	1.644e+01	1.137e+00	3.167e+00	5.764e+00	**9.091e-01**
	VF	8.270e-01	1.730e+03	7.037e+00	8.079e-01	7.760e-01	**1.593e-05**
HIRES	Traj.	1.611e+04	8.290e+04	**1.332e+04**	1.022e+08	7.732e+05	4.836e+04
	Spec.	3.721e+29	7.764e+01	4.297e+27	1.935e+27	**2.451e+00**	7.397e+27
	VF	3.801e+10	9.389e+05	6.314e+07	2.891e+07	**3.215e+05**	1.434e+07
Davis–Skodje	Traj.	1.226e+08	6.274e+08	1.822e+06	2.999e+03	1.120e+05	**4.173e+01**
	Spec.	2.628e+00	9.570e-01	4.924e-01	4.526e+00	9.559e-01	**4.713e-01**
	VF	3.521e+12	1.953e+07	4.576e+06	5.577e+08	5.275e+01	**2.130e+01**

Note: '–' denotes that the run did not finish within the allotted eight-hour time limit, and 'X' denotes ill-defined Jacobian metrics due to null space degeneracies.

Table 3: Extrapolation errors across six stiff ODE benchmarks. We report **minimum** trajectory error (Traj.), vector-field error (VF), and spectral error (Spec.). Lower values indicate better performance, and **best results are highlighted in bold**.

		D → S			D → C		
Problem	**Metric**	**MLP**	**Kim21**	**StiffNet**	**MLP**	**Kim21**	**StiffNet**
VDPOL	Traj.	2.755e+00	1.058e+00	5.866e-01	1.063e+00	1.058e+00	**4.306e-02**
	Spec.	1.224e-01	1.801e-01	4.962e-02	1.292e-01	1.801e-01	**5.881e-03**
	VF	4.739e-01	2.532e-01	2.340e-01	3.970e-01	2.532e-01	**5.065e-02**
POLLU	Traj.	–	–	–	2.652e+25	5.326e+00	**1.803e-04**
	Spec.	–	–	–	X	X	X
	VF	–	–	–	1.780e+26	**1.372e+02**	3.572e+04
ROBER	Traj.	1.063e+00	1.29e+00	1.423e-01	1.081e+00	1.177e+00	**4.361e-02**
	Spec.	1.395e+27	**9.259e-01**	2.289e+21	1.290e+24	9.655e-01	5.026e+21
	VF	8.148e+14	2.912e+02	2.784e+07	2.000e+13	**1.223e+02**	5.232e+06
OREGO	Traj.	8.848e-01	8.697e-01	8.983e-01	9.252e-01	9.393e-01	**1.962e-01**
	Spec.	1.002e+00	1.196e+00	8.729e-01	9.897e-01	9.957e-01	**7.247e-01**
	VF	7.830e-01	1.039e+00	6.344e-04	7.603e-01	5.062e-02	**4.548e-07**
HIRES	Traj.	2.497e+02	1.020e+04	**9.132e+01**	1.873e+03	1.106e+04	4.834e+02
	Spec.	2.665e+27	**9.451e-01**	1.213e+25	4.459e+23	9.507e-01	1.394e+25
	VF	6.593e+07	5.935e+03	9.956e+03	1.951e+04	**8.006e+02**	4.681e+05
Davis–Skodje	Traj.	1.442e+06	4.277e+05	1.290e+02	1.775e+01	1.024e+00	**2.742e-01**
	Spec.	4.592e-02	2.882e-01	4.043e-01	**1.434e-02**	4.168e-01	3.869e-02
	VF	4.108e+08	1.314e+04	1.115e+02	3.869e+04	**6.487e-02**	3.274e-01

We next examine how trajectory reconstruction relates to recovery of the local dynamics. Figure 3 shows a Van der Pol example in which the StiffNet trained with $D \to C$ closely follows the observed *and* extrapolated trajectory, while visible errors remain in the Jacobian eigenvalue magnitudes (see figure 11 of Appendix D). A close trajectory fit therefore does not necessitate an equally accurate recovery of the local linearization. In the Kim21 $D \to S$ example (Figure 3), the predicted trajectory approaches a nearly constant state after the first transition and fails to reproduce both subsequent training and extrapolation oscillations.

Table 2 reports median extrapolation errors. All are first pretrained with derivative matching. Firstly, observe that the combination of the StiffNet architecture and collocation training renders the lowest trajectory error for four out of the six test problems. On HIRES, we note that none of the methods performed satisfactorily although the stiffnet with shooting had the lowest error. Furthermore, on OREGO the $D \to S$ StiffNet had the lowest median error, however, the $D \to C$ StiffNet had the minimum error. Hence, StiffNet has both the lowest median and minimum trajectory errors on all test problems.

Figure 4 examines reconstruction as the stiffness parameters vary. StiffNet with $D \to C$ achieves lower Davis Skodje reconstruction error throughout the tested range. For Van der Pol, its advantage narrows as μ increases compared to the relatively flat Kim21 error.

Additional trajectory plots are collected in Appendix A. Additional reliability and timing results are collected in Appendix D.

5 Discussion

Our results show how established representation and training techniques can be combined to address the vanishing gradient problem identified by Fronk & Petzold (2025b). The scalar analysis establishes a restoring derivative matching gradient for overestimated rates and a nonvanishing initial joint collocation correction. Across six stiff benchmarks, the experiments demonstrate practical benefits from the combined formulation on nonlinear stiff systems, with performance varying across benchmarks and metrics.

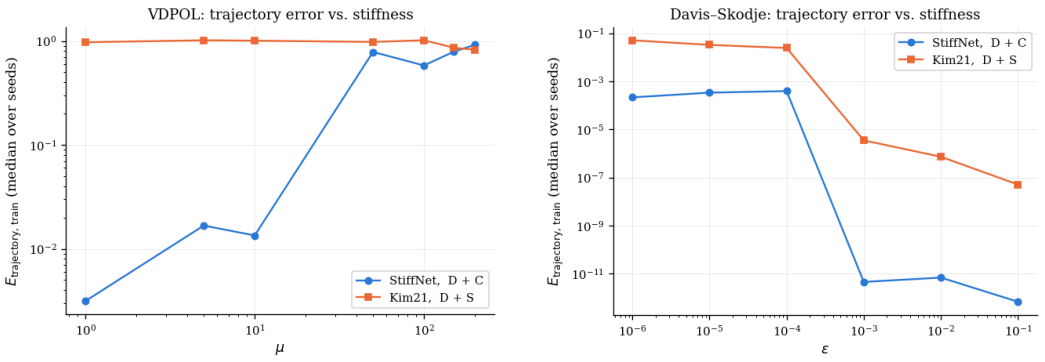


Figure 4: Trajectory error as stiffness varies for VDPOL (left) and Davis–Skodje (right). Stiffness increases with μ and with $1/\varepsilon$.

We observe in Section 4 that neither the training nor architecture alone provide substantial improvements, and we attribute the success in mitigating stiffness barriers to the combination of both methods. Indeed, the scalar analysis suggests a mechanism for the interaction seen in Table 1's ablations: the transformed derivative loss exposes errors in the log-rate coordinate, and joint collocation can retain a correction when shooting becomes insensitive.

Limitations: The warm start discards finite-difference targets that disagree across adjacent intervals, which stabilizes training but may remove the transitions where fast dynamics are most visible. Spectral recovery is also imperfect and uneven across problems. Indeed, accurate trajectories can coexist with unidentified fast modes, as illustrated by Appendix E.4. One must also choose between the damping and signed rate form of the architecture; training both and keeping the better fit is viable yet expensive. Finally, collocation itself is asymptotically expensive (see E.5), and so may scale poorly for very large systems. However, we observe in Appendix D.2 that the collocation objectives were consistently cheaper to train than shooting.

These limitations motivate future work on derivative estimates that remain reliable near rapid transitions and solvers that exploit the block-banded structure of the collocation Jacobian to improve training time. Additional trajectories or targeted perturbations could help distinguish optimization difficulties from insufficient information about fast modes. We hope that demonstrating the ability to retain useful gradient information when training on stiff problems motivates an effort to design simpler, more theoretically motivated, and generalizable approaches that successfully learn stiff dynamics.

Ultimately, we have shown that reformulating neural ODE training and parameterization can sidestep the vanishing sensitivity problem on stiff systems. However, accurate trajectory reconstruction is not consistently accompanied by accurate recovery of the vector field and its Jacobian spectrum.

AI Use Statement

Use of Generative AI tools was limited to assisting with checking and polishing this document, producing LaTeX-code for figures and tables, as well with code development.

Reproducibility Statement

Supporting code for the numerical experiments is available at `https://anonymous.4open.science/r/StiffNet-1C66/`

REFERENCES

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL https://doi.org/10.1137/141000671.

Matteo Caldana and Jan S. Hesthaven. Neural ordinary differential equations for model order reduction of stiff systems. *International Journal for Numerical Methods in Engineering*, 126(12), 2025. ISSN 0029-5981. doi: 10.1002/nme.70060. URL https://doi.org/10.1002/nme.70060.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Michael J. Davis and Rex T. Skodje. Geometric investigation of low-dimensional manifolds in systems approaching equilibrium. *The Journal of Chemical Physics*, 111:859–874, 1999. doi: 10.1063/1.479372. URL https://doi.org/10.1063/1.479372.

Felix A. Döppel and Martin Votsmeier. Efficient neural network models of chemical kinetics using a latent asinh rate transformation. *Reaction Chemistry Engineering*, 8(10):2620–2631, 10 2023. ISSN 2058-9883. doi: 10.1039/d3re00212h. URL https://doi.org/10.1039/d3re00212h.

Richard J. Field and Richard M. Noyes. Oscillations in chemical systems. iv. limit cycle behavior in a model of a real chemical reaction. *The Journal of Chemical Physics*, 60:1877–1884, 1974. doi: 10.1063/1.1681288. URL https://doi.org/10.1063/1.1681288.

Colby Fronk and Linda Petzold. Training stiff neural ordinary differential equations with implicit single-step methods. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(12):123147, 2024. doi: 10.1063/5.0243382. URL https://doi.org/10.1063/5.0243382.

Colby Fronk and Linda Petzold. Training stiff neural ordinary differential equations with explicit exponential integration methods. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 35(3):033154, 03 2025a. ISSN 1054-1500. doi: 10.1063/5.0251475. URL https://doi.org/10.1063/5.0251475.

Colby Fronk and Linda Petzold. The vanishing gradient problem for stiff neural differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 35(11):113113, 11 2025b. ISSN 1054-1500. doi: 10.1063/5.0294317. URL https://doi.org/10.1063/5.0294317.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, 2 edition, 1996. doi: 10.1007/978-3-642-05221-7. URL https://doi.org/10.1007/978-3-642-05221-7.

Xiangjun Huang, Kyriakos Kandris, and Evina Katsou. Training stiff neural ordinary differential equations in data-driven wastewater process modelling. *Journal of Environmental Management*, 373:123870, 2025. doi: 10.1016/j.jenvman.2024.123870. URL https://doi.org/10.1016/j.jenvman.2024.123870.

Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9):093122, 2021. ISSN 1054-1500. doi: 10.1063/5.0060697. URL https://doi.org/10.1063/5.0060697.

Linus Langenkamp, Philip Hannebohm, and Bernhard Bachmann. Efficient training of physics-enhanced neural odes via direct collocation and nonlinear programming. In *Linköping Electronic Conference Proceedings*, volume 218. Linköping University Electronic Press, 2025. doi: 10.3384/ecp218445. URL https://doi.org/10.3384/ecp218445.

Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

Allen Alvarez Loya, Daniel A. Serino, and Qi Tang. Structure-preserving neural ordinary differential equations for stiff systems. *arXiv preprint arXiv:2503.01775*, 2025. doi: 10.48550/arXiv.2503.01775. URL https://doi.org/10.48550/arXiv.2503.01775.

Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

Avik Pal. On Efficient Training & Inference of Neural Differential Equations, 2023a.

Avik Pal. Lux: Explicit Parameterization of Deep Neural Networks in Julia, April 2023b. URL https://doi.org/10.5281/zenodo.7808903. If you use this software, please cite it as below.

Avik Pal, Yingbo Ma, Viral Shah, and Christopher Rackauckas. Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics. *arXiv preprint arXiv:2105.03918*, 2021. doi: 10.48550/arXiv.2105.03918. URL https://doi.org/10.48550/arXiv.2105.03918.

Wenqing Peng, Zhi-Song Liu, and Michael Boy. Spin-ode: Stiff physics-informed neural ode for chemical reaction rate estimation, 2025. URL https://arxiv.org/abs/2505.05625.

Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1):15, 2017.

H. H. Robertson. The solution of a set of reaction rate equations. In J. Walsh (ed.), *Numerical Analysis: An Introduction*. Academic Press, 1966.

Elisabeth Roesch, Chris Rackauckas, and Michael Stumpf. Collocation based training of neural ordinary differential equations. *Statistical Applications in Genetics and Molecular Biology*, 20, 2021. doi: 10.1515/sagmb-2020-0025. URL https://doi.org/10.1515/sagmb-2020-0025.

E. Schäfer. A new approach to explain the high irradiance responses of photomorphogenesis on the basis of phytochrome. *Journal of Mathematical Biology*, 2:41–56, 1975. doi: 10.1007/BF00276015. URL https://doi.org/10.1007/bf00276015.

L. F. Shampine and C. W. Gear. A user's view of solving stiff ordinary differential equations. *SIAM Review*, 21:1–17, 1979. doi: 10.1137/1021001. URL https://doi.org/10.1137/1021001.

Mariia Shapovalova and Calvin Tsay. Training neural odes using fully discretized simultaneous optimization. *IFAC-PapersOnLine*, 59:469–474, 2025. doi: 10.1016/j.ifacol.2025.07.190. URL https://doi.org/10.1016/j.ifacol.2025.07.190.

Gustaf Söderlind, Laurent Jay, and Manuel Calvo. Stiffness 1952–2012: Sixty years in search of a definition. *BIT Numerical Mathematics*, 55:531–558, 2014. doi: 10.1007/s10543-014-0503-3. URL https://doi.org/10.1007/s10543-014-0503-3.

Mauro Valorani. Learning stiff dynamical operators: Scaling, fast-slow excitation, and eigen-consistent neural models, 2026. URL https://arxiv.org/abs/2601.01632.

Balth. van der Pol Jun. Lxxxviii. on "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926. doi: 10.1080/14786442608564127. URL https://doi.org/10.1080/14786442608564127.

J. G. Verwer. Gauss–seidel iteration for stiff odes from chemical kinetics. *SIAM Journal on Scientific Computing*, 15:1243–1250, 1994. doi: 10.1137/0915076. URL https://doi.org/10.1137/0915076.

Hong Zhang and Wenjun Zhao. A memory-efficient neural ode framework based on high-level adjoint differentiation, 2023. URL https://arxiv.org/abs/2206.01298.

Achraf Zinihi, Matthias Ehrhardt, and Moulay Rchid Sidi Ammi. Structure-preserving neural odes via nonstandard finite difference discretization, 2026. URL https://arxiv.org/abs/2607.10858.

APPENDIX

A PROBLEMS

A.1 VDPOL

(van der Pol Jun., 1926), with $(u_1, u_2) = (x, \dot{x})$, named VDPOL in Hairer & Wanner (1996):

$$\dot{u}_1 = u_2, \qquad \dot{u}_2 = \mu\,(1 - u_1^2)\,u_2 - u_1, \tag{7}$$

$\mathbf{u}(0) = (2,\ 0)^\top$, $t \in [0,\ 200]$. We use $\mu = 100$.

This problem models an oscillator. For large μ and when $|u_1| > 1$, $\mu(1 - u_1^2)$ is negative and of order μ, so perturbations of u_2 decay within a time of order $1/\mu$. However, the solution varies on a time scale of order μ. Therefore, these time scales differ by a factor of order μ^2, making the problem stiff.

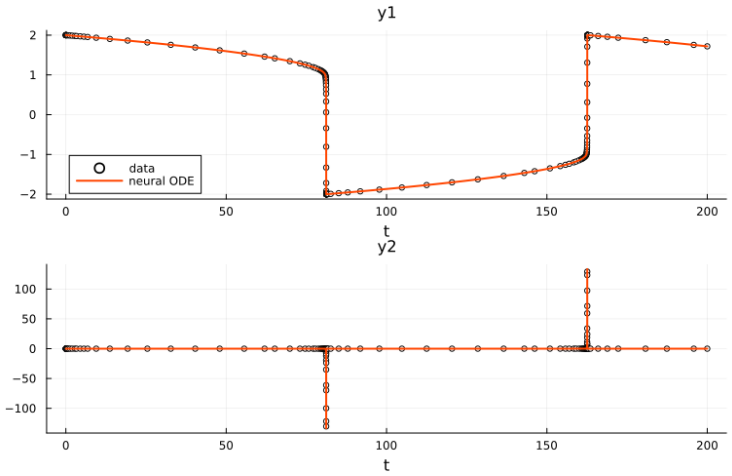


Figure 5: VDPOL: observed data and neural ODE fit for both state components. (StiffNet + Derivative Matching + Collocation)

A.2 ROBER

(Robertson, 1966), named ROBER in Hairer & Wanner (1996):

$$\begin{aligned} \dot{y}_1 &= -k_1 y_1 + k_3 y_2 y_3, \\ \dot{y}_2 &= \ \ k_1 y_1 - k_2 y_2^2 - k_3 y_2 y_3, \\ \dot{y}_3 &= \ \ k_2 y_2^2, \end{aligned} \tag{8}$$

$(k_1, k_2, k_3) = (0.04,\ 3 \times 10^7,\ 10^4)$, $\mathbf{y}(0) = (1, 0, 0)^\top$, $t \in [10^{-6},\ 10^3]$. The rate constants span nine orders of magnitude. The production rate $k_1 y_1$ is small compared to y_1 and y_3, so these species change slowly, but it is large compared with y_2 itself. Therefore, y_2 changes many orders of magnitude faster than the other species, which makes the problem stiff.

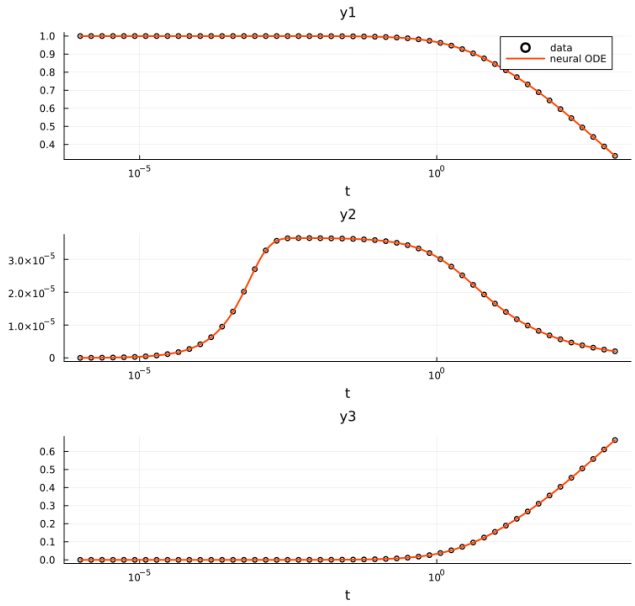


Figure 6: ROBER: observed data and neural ODE fit for all three species on a logarithmic time axis. (StiffNet + Derivative Matching + Collocation)

A.3 OREGO

Oregonator (Field & Noyes, 1974), in the scaled form of Hairer & Wanner (1996):

$$\begin{aligned} \dot{y}_1 &= s\,\big(y_2 + y_1(1 - q\,y_1 - y_2)\big), \\ \dot{y}_2 &= \tfrac{1}{s}\,\big(y_3 - (1 + y_1)\,y_2\big), \\ \dot{y}_3 &= w\,(y_1 - y_3), \end{aligned} \tag{9}$$

$s = 77.27$, $q = 8.375 \times 10^{-6}$, $w = 0.161$, $\mathbf{y}(0) = (1, 2, 3)^\top$, $t \in [0,\, 180]$. The factor s in the first equation makes perturbations of y_1 decay quickly, while y_2 and y_3, scaled by $1/s$ and w evolve slowly. This separation of time scales makes the problem stiff.

A.4 HIRES

(Schäfer, 1975), with parameters and name from Hairer & Wanner (1996):

$$\begin{aligned} \dot{y}_1 &= -1.71\,y_1 + 0.43\,y_2 + 8.32\,y_3 + 0.0007, \\ \dot{y}_2 &= 1.71\,y_1 - 8.75\,y_2, \\ \dot{y}_3 &= -10.03\,y_3 + 0.43\,y_4 + 0.035\,y_5, \\ \dot{y}_4 &= 8.32\,y_2 + 1.71\,y_3 - 1.12\,y_4, \\ \dot{y}_5 &= -1.745\,y_5 + 0.43\,y_6 + 0.43\,y_7, \\ \dot{y}_6 &= -280\,y_6 y_8 + 0.69\,y_4 + 1.71\,y_5 - 0.43\,y_6 + 0.69\,y_7, \\ \dot{y}_7 &= 280\,y_6 y_8 - 1.81\,y_7, \\ \dot{y}_8 &= -280\,y_6 y_8 + 1.81\,y_7, \end{aligned} \tag{10}$$

$\mathbf{y}(0) = (1,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0.0057)^\top$, $t \in [0,\, 321.8122]$. This system of 8 species contains reactions that act on time scales far shorter than the interval length, and this mismatch is the source of its stiffness.

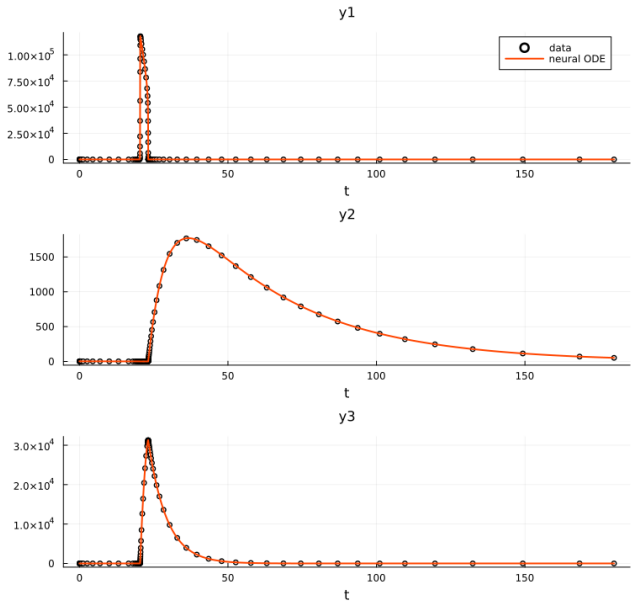


Figure 7: OREGO: observed data and neural ODE fit for all three species. (StiffNet + Derivative Matching + Collocation)

A.5 DAVIS–SKODJE

(Davis & Skodje, 1999):

$$\dot{y}_1 = -y_1, \qquad \dot{y}_2 = -\gamma\, y_2 + \frac{(\gamma - 1)\, y_1 + \gamma\, y_1^2}{(1 + y_1)^2}, \qquad \gamma = \varepsilon^{-1}. \tag{11}$$

ε can be varied for stiffness, $\mathbf{y}(0) = (1, 2)^\top$, $t \in [0,\, 15]$. We use $\epsilon = 0.01$. This classic example is stiff for small ε because the second coordinate decays at rate $1/\varepsilon$, while the first decays at rate 1.

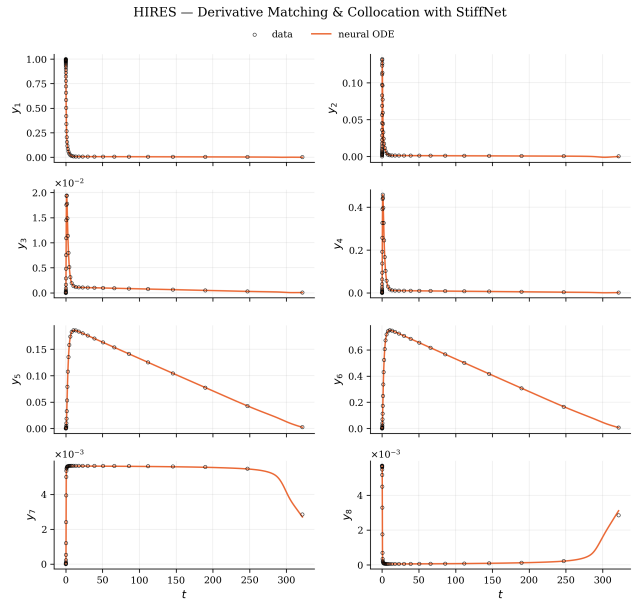


Figure 8: HIRES: observed data and neural ODE fit for all eight species. (StiffNet + Derivative Matching + Collocation)

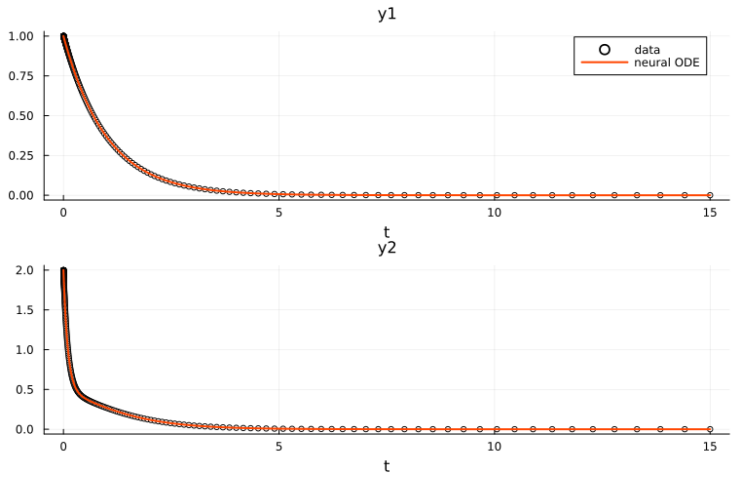


Figure 9: Davis–Skodje: observed data and neural ODE fit for both state components. (StiffNet + Derivative Matching + Collocation)

A.6 POLLU

(Verwer, 1994), with r_j in Table 4 and k_j in Table 5:

$$
\begin{aligned}
\dot{y}_1 &= -r_1 - r_{10} - r_{14} - r_{23} - r_{24} + r_2 + r_3 + r_9 + r_{11} + r_{12} + r_{22} + r_{25},\\
\dot{y}_2 &= -r_2 - r_3 - r_9 - r_{12} + r_1 + r_{21},\\
\dot{y}_3 &= -r_{15} + r_1 + r_{17} + r_{19} + r_{22},\\
\dot{y}_4 &= -r_2 - r_{16} - r_{17} - r_{23} + r_{15},\\
\dot{y}_5 &= -r_3 + 2r_4 + r_6 + r_7 + r_{13} + r_{20},\\
\dot{y}_6 &= -r_6 - r_8 - r_{14} - r_{20} + r_3 + 2r_{18},\\
\dot{y}_7 &= -r_4 - r_5 - r_6 + r_{13},\\
\dot{y}_8 &= r_4 + r_5 + r_6 + r_7,\\
\dot{y}_9 &= -r_7 - r_8,\\
\dot{y}_{10} &= -r_{12} + r_7 + r_9,\\
\dot{y}_{11} &= -r_9 - r_{10} + r_8 + r_{11},\\
\dot{y}_{12} &= r_9,\\
\dot{y}_{13} &= -r_{11} + r_{10},\\
\dot{y}_{14} &= -r_{13} + r_{12},\\
\dot{y}_{15} &= r_{14},\\
\dot{y}_{16} &= -r_{18} - r_{19} + r_{16},\\
\dot{y}_{17} &= -r_{20},\\
\dot{y}_{18} &= r_{20},\\
\dot{y}_{19} &= -r_{21} - r_{22} - r_{24} + r_{23} + r_{25},\\
\dot{y}_{20} &= -r_{25} + r_{24}.
\end{aligned}
\tag{12}
$$

$\mathbf{y}(0) = \mathbf{0}$ except $y_2 = 0.2$, $y_4 = 0.04$, $y_7 = 0.1$, $y_8 = 0.3$, $y_9 = 0.01$, $y_{17} = 0.007$; $t \in [0, 60]$. Species such as y_3 and y_{16} are removed at rates up to order 10^{12} while others evolve over the full interval.

A.7 Quantifying stiffness

To measure how stiff each benchmark is, we compute the stiffness indicator of Söderlind et al. (2014) along its solution. For the Jacobian J of the vector-field, let $m_2[J]$ and $M_2[J]$ be the smallest and largest eigenvalues of $(J + J^\top)/2$, and set $\sigma[J] = (m_2[J] + M_2[J])/2$. When $\sigma < 0$, an

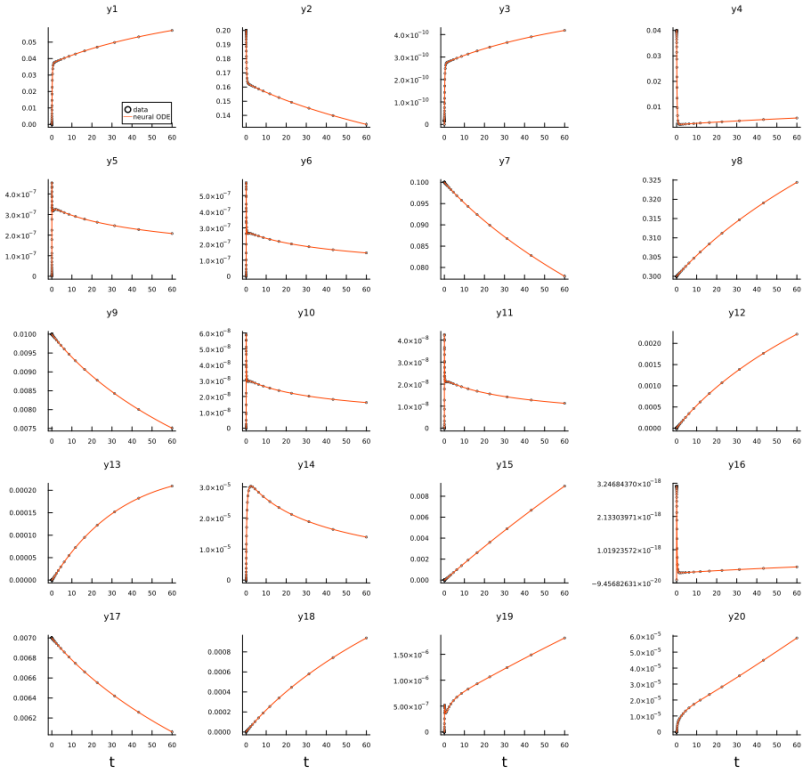


Figure 10: POLLU: observed data and neural ODE fit for all twenty species. (StiffNet + Derivative Matching + Collocation)

Table 4: Reaction rates for the POLLU problem.

$r_1 = k_1 y_1$	$r_8 = k_8 y_9 y_6$	$r_{15} = k_{15} y_3$	$r_{22} = k_{22} y_{19}$
$r_2 = k_2 y_2 y_4$	$r_9 = k_9 y_{11} y_2$	$r_{16} = k_{16} y_4$	$r_{23} = k_{23} y_1 y_4$
$r_3 = k_3 y_5 y_2$	$r_{10} = k_{10} y_{11} y_1$	$r_{17} = k_{17} y_4$	$r_{24} = k_{24} y_{19} y_1$
$r_4 = k_4 y_7$	$r_{11} = k_{11} y_{13}$	$r_{18} = k_{18} y_{16}$	$r_{25} = k_{25} y_{20}$
$r_5 = k_5 y_7$	$r_{12} = k_{12} y_{10} y_2$	$r_{19} = k_{19} y_{16}$	
$r_6 = k_6 y_7 y_6$	$r_{13} = k_{13} y_{14}$	$r_{20} = k_{20} y_{17} y_6$	
$r_7 = k_7 y_9$	$r_{14} = k_{14} y_1 y_6$	$r_{21} = k_{21} y_{19}$	

Table 5: Rate constants for the POLLU problem.

$k_1 = 0.35$	$k_6 = 1.5 \times 10^4$	$k_{11} = 2.2 \times 10^{-2}$	$k_{16} = 3.5 \times 10^{-4}$	$k_{21} = 2.1$
$k_2 = 26.6$	$k_7 = 1.3 \times 10^{-4}$	$k_{12} = 1.2 \times 10^4$	$k_{17} = 1.75 \times 10^{-2}$	$k_{22} = 5.78$
$k_3 = 1.23 \times 10^4$	$k_8 = 2.4 \times 10^4$	$k_{13} = 1.88$	$k_{18} = 1.0 \times 10^8$	$k_{23} = 4.74 \times 10^{-2}$
$k_4 = 8.6 \times 10^{-4}$	$k_9 = 1.65 \times 10^4$	$k_{14} = 1.63 \times 10^4$	$k_{19} = 4.44 \times 10^{11}$	$k_{24} = 1.78 \times 10^3$
$k_5 = 8.2 \times 10^{-4}$	$k_{10} = 9.0 \times 10^3$	$k_{15} = 4.8 \times 10^6$	$k_{20} = 1.24 \times 10^3$	$k_{25} = 3.12$

explicit method generally cannot take steps much longer than $\Delta t = -1/\sigma$ while remaining stable. The stiffness factor $T/\Delta t$ compares this limit with the length T of the integration interval, and $T/\Delta t \gg 1$ implies stiffness. Table 6 reports these quantities at the stiffest point of each solution.

Table 6: Stiffness indicator at the stiffest point of each reference solution (Söderlind et al., 2014), with $\Delta t = -1/\sigma_{\min}$ and unscaled states.

Problem	$\sigma_{\min}$	$T/\Delta t$
Van der Pol ($\mu = 100$)	-150	3.01×10^4
ROBER	-3.38×10^3	3.38×10^6
OREGO	-6.83×10^4	1.23×10^7
HIRES	-106	3.40×10^4
Davis–Skodje ($\varepsilon = 0.01$)	-50.5	758
POLLU	-2.22×10^{11}	1.33×10^{13}

All benchmarks have $T/\Delta t \gg 1$, so all are substantially stiff.

B Architecture Details

B.1 Complete parameterization

We don't pass in raw state values u directly into the neural network. Rather, we pass in normalized and scaled inputs:

$$\hat{u} = \frac{u - y_{mid}}{y_{scale}}$$

Note that we include the following safeguard

$$y_{\text{scale}} = \max\left(y_{\max} - y_{\min}, 10^{-12} \max(|y_{\max}|, |y_{\min}|), \text{floatmin}()\right).$$

We additionally construct a second representation that helps compress large orders of magnitude

$$z = \frac{\text{asinh}(u/\chi)}{Z_{norm}}$$

where

$$u_{\max} = \max\left(\max_i |y_i|, 10^{-30}\right),$$

and

$$\chi = 10^{-3} u_{\max}, \qquad Z_{\text{norm}} = \text{asinh}(10^3).$$

The input to the network is the concatenation

$$\phi(u) = \begin{bmatrix} \hat{u} \\ z \end{bmatrix},$$

We pass the resulting vector into our neural network, which uses $\tanh$ activation functions and has two output heads.

Although our architecture is

$$\underbrace{\kappa * \sinh(\alpha)}_{\text{production}} - \underbrace{\rho * \exp(\beta) * u}_{\text{state-proportional}}.$$

α and β are not the raw outputs of the network, but they are bounded. Define

$$s = \max\left(\max_i \left|\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right|, 10^{-30}\right).$$

Also define

$$\kappa = 10^{-3}s, \qquad a_{\max} = \operatorname{asinh}(3 \times 10^5).$$

The raw output of the production head, p_α is bounded according to

$$\alpha = a_{\max} \tanh\left(\frac{p_\alpha}{a_{\max}}\right),$$

and the resulting production term is

$$P(u) = \kappa \sinh(\alpha).$$

Note that since $\tanh \in (-1, 1)$, $\alpha \in (-a_{max}, a_{max})$, so this term serves as a bound for what can be passed into the $\sinh$ function so that large magnitudes do not explode.

For the state-proportional term, we define the following scaling term:

$$\rho = \frac{s}{\max\left(y_{\text{scale}}, 10^{-12}u_{\max}, \text{floatmin}()\right)}.$$

We define a maximum rate from the observed time points so that we don't output arbitrarily large rates that are not justified in the data

$$r_{\max} = \frac{10}{\min_i(t_{i+1} - t_i)},$$

and we also provide a bound for the state-proportional head

$$g_{\max} = \text{clamp}\left(\operatorname{asinh}\left(\frac{r_{\max}}{\rho}\right), 1, \operatorname{asinh}(10^6)\right).$$

Note that we want to bound p_{rate} so that the state-proportional head can only move by the maximum allowed rate. Therefore, the $g_{\max}$ stems from the fact that we want $\rho \sinh(g_{\max}) = r_{\max}$, so we reach $g_{\max} = \operatorname{asinh}(\frac{r_{\max}}{\rho})$, before the numerical clamp.

The raw output of the state-proportional head, p_{rate} is bounded according to

$$c = g_{\max} \tanh\left(\frac{p_{\text{rate}}}{g_{\max}}\right).$$

For the guaranteed-damping form we set $\beta = c$, and for the signed form we set $\gamma = c$. Thus,

$$f_\theta(u) = P(u) - \rho \exp(\beta(u))u$$

for the guaranteed form, and

$$f_\theta(u) = P(u) + \rho \sinh(\gamma(u))u$$

for the signed form. (see below for description of signed vs guaranteed).

B.2 SIGNED VS. GUARANTEED FORM

The state-proportional arm can take either of the following forms (the neural network outputs either γ or β depending on the form):

$$\text{signed:} \qquad \rho * \sinh(\gamma(u)) * u,$$

$$\text{guaranteed damping:} \qquad -\rho * \exp(\beta(u)) * u.$$

For all problems except VDPOL, we use the guaranteed form

$$-\rho * \exp(\beta(u)) * u,$$

because $\rho > 0$ and $\exp(\beta) > 0$. This term therefore always points toward zero, damping the system and helping prevent divergence during rollout. For VDPOL, both damping and amplification are needed, so we instead use the signed form

$$\rho * \sinh(\gamma(u)) * u.$$

B.3 INITIALIZATION

For the rate head of the neural network, we overwrite the bias to -1 for the signed form and -6 for the guaranteed form, initially biasing the state-proportional term toward damping. We also included a Glorot Glorot & Bengio (2010) uniform initialization, which samples each dense-layer weight from

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right),$$

with biases initialized to zero. When not using Glorot initialization, our model uses Lux's Pal (2023a;b) default initializer. We use the Xoshiro random-number generator to initialize our model, which takes in an experiment seed that we vary across runs.

B.4 NETWORK DIMENSIONS AND PARAMETER COUNTS

The StiffNet's width is set per problem, and the MLP and Kim21-scaled widths are then chosen so their parameter counts match StiffNet's as closely as the integer width allows. Depth is fixed per architecture: the Kim21 follows Kim et al. (2021), using the recommended deeper network.

Table 7: Trainable parameter count P for each architecture. Width is the number of nodes per hidden layer and d is the dimension of each system.

		StiffNet		MLP		Kim21	
Problem	States d	width	P	width	P	width	P
Van der Pol	2	12	268	14	282	7	317
Davis–Skodje	2	12	268	14	282	7	317
ROBER	3	12	318	15	348	7	332
OREGO	3	16	486	19	516	9	516
HIRES	8	16	816	21	827	11	855
POLLU	20	32	4744	40	4940	40	4940
Hidden layers		2 (POLLU: 3)		2 (POLLU: 3)		6 (POLLU: 3)	

B.5 SAMPLING DETAILS

C TRAINING DETAILS

C.1 DERIVATIVE-MATCHING DETAILS

Before the derivative matching stage, we estimate target derivatives directly from the observed data using a three-point finite-difference formula.

Table 8: Number of observations across problems. Adaptive observations are the steps of a stiff solver on the reference solution. Test observations extend the training interval for extrapolation.

		Training		Test	
Problem	Sampling	Interval	Points	Interval	Points
VDPOL ($\mu = 100$)	adaptive (Kvaerno5)	$[0, 200]$	151	$(200, 400]$	302
ROBER	log-spaced	$[10^{-6}, 10^{3}]$	50	$(10^{3}, 10^{12}]$	100
OREGO	adaptive (Rodas5)	$[0, 180]$	112	$(180, 360]$	224
HIRES	log-spaced	$[0, 321.8]$	50	$(321.8, 643.6]$	100
Davis–Skodje ($\varepsilon = 0.01$)	log-spaced	$[0, 15]$	300	$(15, 30]$	600
POLLU	log-spaced	$[0, 60]$	50	$(60, 120]$	100

We also mask extremely sharp derivatives to prevent divergence. For each data point, let

$$a = \left| \frac{Y_j - Y_{j-1}}{t_j - t_{j-1}} \right|, \qquad b = \left| \frac{Y_{j+1} - Y_j}{t_{j+1} - t_j} \right|.$$

We exclude the derivative from the objective when

$$\frac{\max(a, b)}{\max(\min(a, b), 10^{-300})} > 100.$$

This prevents bad finite-difference estimates from ruining the objective.

C.2 Radau IIA discretization and residuals

We include a normalized version of the states during optimization

$$\hat{U} = \frac{U}{y_{scale}}$$

As opposed to the traditional method of including collocation stage derivatives in the optimization problem, we include normalized state increments. Specifically, for collocation intervals of width h,

$$\hat{K} \approx \frac{h}{y_{scale}} f_\theta$$

Intuitively, $\hat{K}$ can be thought of as the fraction of each coordinate's range covered within that interval. Comparing normalized increments is fit for stiff systems because different coordinates can have derivatives on different orders of magnitude within an interval. Therefore, for each coordinate j, the first set of expressions in our collocation residual is

$$\hat{K}_j - \frac{h}{y_{\text{scale},j}} f_{\theta,j}, \forall j$$

The next set of expressions we include in the collocation residual ensure continuity; we require that the change between adjacent collocation nodes must agree with the change predicted by the Radau stages

$$\hat{U}_{i+1} - \hat{U}_i - \sum_{s=1}^{2} b_s \hat{K}_{i,s}$$

where b_s are the Radau IIA weights.

The final set of expressions in the residual enforces our predicted trajectory to equal normalized observations

$$\hat{U}(t_r) - \frac{y_r}{y_{\text{scale}}} = 0, \qquad \forall r.$$

where y_r are the true data.

C.3 COLLOCATION INITIALIZATION

To initialize our mesh, for each observation interval j, we compute the largest normalized change in state

$$J = \max_c \frac{|y_{c,j+1} - y_{c,j}|}{y_{\text{scale},c}}.$$

We then create n_j subdivisions in that interval, where

$$n_j = \text{clamp}\left(10J, 1, 8\right).$$

Therefore, intervals with larger changes in state receive a larger initial mesh.

We arrive at an initial guess for the state values, U_0, through linear interpolation of the data points.

We initialize the Radau stage increments on the intervals using the normalized change between its surrounding nodes state values,

$$K_i^0 = \frac{U_{0,i+1} - U_{0,i}}{y_{\text{scale}}}.$$

Finally, we scale the stage residuals separately for each state coordinate. We define

$$\hat{k}_c = \max\left(10^{-2} \max_i |K_{c,i}^0|, 10^{-12}\right).$$

If the unscaled stage residual for coordinate c is

$$R_c = \hat{K}_c - \frac{h}{y_{\text{scale},c}} f_{\theta,c},$$

we instead use

$$\frac{R_c}{\hat{k}_c}.$$

Therefore, all coordinates contribute appropriately to the loss in the residual.

C.4 ADAPTIVE MESH REFINEMENT AND SELECTION

After each nonlinear least squares solve of the collocation residual, we sum the squared errors of the continuity and the stage residuals within each interval. (Note, we don't include observation residuals in this error). Let this error be E_j, and we refine all intervals such that

$$E_j > 0.4 \max_k E_k.$$

For each selected interval, we double its number of subdivisions, up to a maximum of 48.

After this refinement process, we again repeat the collocation process by interpolating the original observations, constructing stage increments, and reusing network parameters from the previous refinement round.

Note, after all the rounds of refinement, we return the parameters with the lowest rollout error, and this is not always those from the final collocation round.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 METRICS

To quantify the correctness of the learned neural ODEs we have used three different metrics that we present here. For all metrics we consider a differential equation with d species, discretized on a temporal grid with N points in the domain $[T_0, T_1]$. We denote by $y_j(t)$ the true value of the trajectory for species j at time t, and $\hat{y}_j(t)$ the equivalent Neural ODE approximation. Similarly, we let $f : \mathbb{R}^d \to \mathbb{R}^d$ be the right hand side of the differential equation, and $\hat{f}_\theta : \mathbb{R}^d \to \mathbb{R}^d$ the neural network. Lastly, we let $\lambda_j(t)$ and $\hat{\lambda}_j(t)$ refer to the absolute value of the eigenvalues of $J_f(t)$ and $J_{\hat{f}_\theta}(t)$, respectively, where J refers to the Jacobian.

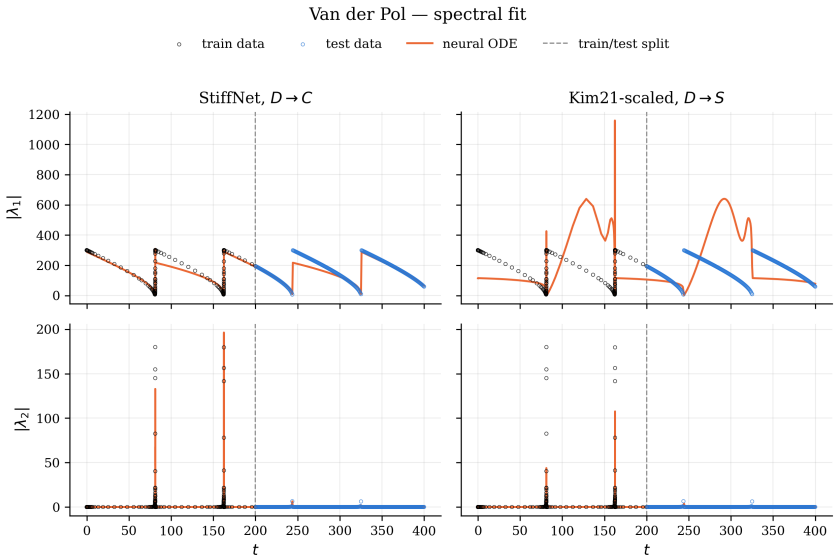


Figure 11: Van der Pol spectral fits for StiffNet trained with derivative matching + collocation along with Kim21-scaled trained with derivative matching + shooting.

All three metrics are normalized in the sense that a neural ODE with the deterministic mapping $x \mapsto 0$, would have an error of 1 for all these metrics. In addition, all metrics first compute an error per species, that is then averaged across all species.

Trajectory error

$$E_{traj} = \frac{1}{d}\sum_{j=1}^{d}\left(\frac{\sum_{i=1}^{N}|\hat{y}_j(t_i) - y_j(t_i)|^2}{\sum_{i=1}^{N}|y_j(t_i)|^2}\right) \tag{13}$$

Vector-Field Error

$$E_{VF} = \frac{1}{d}\sum_{j=1}^{d}\left(\frac{\sum_{i=1}^{N}|\hat{f}_{\theta,j}(y(t_i)) - f_j(y(t_i))|^2}{\sum_{i=1}^{N}|f_j(y(t_i))|^2}\right) \tag{14}$$

Spectral Error

$$E_{spectral} = \frac{1}{d}\sum_{j=1}^{d}\left(\frac{\sum_{i=1}^{N}|\hat{\lambda}_j(t_i) - \lambda_j(t_i)|^2}{\sum_{i=1}^{N}|\lambda_j(t_i)|^2}\right) \tag{15}$$

The trajectory error measures how closely the model reproduces the observed trajectory. The vector-field error measures the discrepancy between the learned and true vector-fields along the trajectory. The spectral error measures whether the learned model successfully recovers the local rates of the dynamics, which are often closely related to stiffness. There is no clear correlation between these three metrics; often, a model is able to produce small trajectory errors with large vector-field errors, or vice versa.

The error computed on the same temporal domain as that of the training data, i.e. $[T_0, T_1]$, is referred to as the *training* error. The error computed on an extrapolated temporal domain, $[T_1, T_2]$, is referred to as the *test* error. We refer to table 8 for an overview over these domains.

D.2 Reliability of Training

Table 9 reports how often each test achieves a normalized error lower than one. Runs < 1 counts all initializations whose normalized trajectory error is below one on the evaluation interval. Standard deviation of trajectory error is computed over the subset of successful runs, and training time is averaged over all runs.

Table 9: Reliability of training with different combinations of architectures and training methods.

Training Method	Problem	Runs < 1	Std. Dev. Traj. Error	Avg. Training Time
StiffNet + Collocation	Van der Pol	16/50	2.749e-01	62s
	POLLU	36/50	4.637e-02	1.57 h
	ROBER	18/50	2.477e-01	49s
	OREGO	11/50	2.947e-01	96s
	HIRES	0/50	–	3.0 min
	Davis–Skodje	4/50	1.602e-01	2.9 min
Kim21 + Shooting	Van der Pol	0/50	–	21.5 min
	POLLU	0/50	–	–
	ROBER	0/50	–	9.5 min
	OREGO	13/50	4.557e-02	1.63 h
	HIRES	0/50	–	2.13 h
	Davis–Skodje	0/50	–	10.4 min
Kim21 + Collocation	Van der Pol	0/50	–	71s
	POLLU	0/50	–	1.85 h
	ROBER	0/50	–	45s
	OREGO	12/50	2.277e-02	94s
	HIRES	0/50	–	3.5 min
	Davis–Skodje	0/50	–	4.0 min

The reported training times also vary substantially. On Van der Pol, StiffNet with collocation takes an average of 62 seconds, compared with 21.5 minutes for Kim21 with shooting. On OREGO, the corresponding times are 96 seconds and 1.63 hours, although the shooting configuration meets the accuracy criterion more often.

All experiments were implemented in Julia Bezanson et al. (2017). Numerical integration for shooting-based training and trajectory evaluation was performed using DifferentialEquations.jl Rackauckas & Nie (2017). Experiments were run on a shared HPC cluster using one CPU core and 8 GB of memory per task. Jobs were scheduled across heterogeneous AMD EPYC nodes, so individual runs may have used different CPU models.

E Analysis of training

The scalar analysis in the main text considers how training corrects an overestimated decay rate. Such a model predicts a transient that has nearly vanished at observation times but the true transient remains appreciable[1]. Reducing the candidate rate improves the fit. The difficulty is that shooting determines its correction using *both* the prediction error and the sensitivity of the predictions to the rate. An appreciable error can therefore produce a small gradient when that sensitivity is small. The main text observes that the derivative matching and simultaneous Radau collocation step each retain a useful correction in this regime.

In this appendix we develop those observations into statements about the two training stages. We start by relating the exact shooting gradient to the numerical sensitivity analysis of Fronk & Petzold (2025b). We then prove the derivative-matching result and quantify its dependence on derivative-target error. Next, we derive the initial Radau correction and show that it decreases the objective for large candidate rates. Finally, a two-dimensional example separates these optimization properties from the distinct problem of *identification*.

E.1 The fixed-data sensitivity problem

Following Fronk & Petzold (2025b), consider the scalar test equation

$$u' = -au, \qquad a = \rho e^{\beta}, \qquad \rho > 0.$$

[1]Clearly operating under the assumption that the observations contain enough information to resolve the fast transient.

Distinguish[2] the candidate rate a from a fixed true rate $a_* > 0$. Assume exact observations $y_i = u_0 e^{-a_* t_i}$ at fixed times $t_i > 0$, with $u_0 \neq 0$. In the following, we will examine the training signal available at different candidate models of the same underlying scalar test system.

For a fixed candidate log rate β, solving $u' = -\rho e^{\beta} u$ with $u(0) = u_0$ gives $u_\beta(t) = u_0 e^{-\rho e^{\beta} t}$. The shooting loss measures how far these predictions are from the observations.

$$L_S(\beta) = \frac{1}{m} \sum_{i=1}^{m} (u_\beta(t_i) - y_i)^2.$$

A large value implies a poor fit. Gradient based training uses the derivative of $L_S(\beta)$ to change the parameter. The chain rule expresses that derivative as

$$L_S'(\beta) = \frac{2}{m} \sum_{i=1}^{m} \underbrace{\big(u_\beta(t_i) - y_i\big)}_{\text{signed prediction error}} \overbrace{\partial_\beta u_\beta(t_i)}^{\text{prediction sensitivity}} . \tag{16}$$

Note that the first factor records the error at the observation time while the second measures how the prediction changes under a small change in the log rate.

For scalar decay, we can compute that sensitivity directly:

$$\partial_\beta u_\beta(t) = u_0 e^{-\rho e^{\beta} t} \partial_\beta(-\rho e^{\beta} t) = -\rho e^{\beta} t u_\beta(t).$$

Now examine the magnitude of that correction for overestimated candidates. For each fixed observation time $t_i > 0$, the quantity $\rho e^{\beta} t_i \to \infty$ as $\beta \to \infty$. Since both e^{-x} and $x e^{-x}$ tend to zero as $x \to \infty$, we see that both $u_\beta(t_i) \to 0$ and $\partial_\beta u_\beta(t_i) \to 0$. Taking limits in the finite sums therefore gives

$$L_S(\beta) \longrightarrow \frac{1}{m} \sum_i (0 - y_i)^2 = \frac{1}{m} \sum_i y_i^2 > 0, \tag{17}$$

$$L_S'(\beta) \longrightarrow \frac{2}{m} \sum_i (-y_i) \cdot 0 = 0. \tag{18}$$

The limiting loss is positive because $u_0 \neq 0$ and at least one fixed observation is non-zero. Thus, the shooting gradient points toward the required reduction in rate, but its magnitude can be arbitrarily small. Moreover, for a fixed learning rate $\alpha > 0$, the gradient descent correction $-\alpha L_S'(\beta)$ becomes correspondingly small. Concretely, a persistent fitting error can coexist with a weak signal for correcting the parameter. We see that this plateau is already present in the *exact* shooting objective.

We ask, then, whether evaluating the predictions with a stable numerical solver changes this conclusion. A numerical solver changes the prediction map being differentiated, so we must examine the sensitivity of that numerical step. The stability function analysis Fronk & Petzold (2025b) provides a direct way to do this.

For the comparison, use a fixed step size $h > 0$ and observations at fixed step indices, $t_i = n_i h$ with $n_i \geq 1$. Set $z_\beta := -h\rho e^{\beta}$. The exact and numerical updates are $u_\beta(t + h) = e^{-h\rho e^{\beta}} u_\beta(t)$ and $u_{n+1} = \mathcal{R}(z_\beta) u_n$ respectively. Thus, we see that the stability function $\mathcal{R}$ is our numerical replacement for the exact exponential multiplier. It therefore determines both the numerical prediction and its response to a parameter change.

Define, as above, the numerical sensitivity $S_n := \partial_\beta u_n$. The numerical shooting loss and its derivative are

$$L_{S,h}(\beta) = \frac{1}{m} \sum_i (u_{n_i} - y_i)^2, \tag{19}$$

$$L_{S,h}'(\beta) = \frac{2}{m} \sum_i \underbrace{\big(u_{n_i} - y_i\big)}_{\text{signed prediction error}} \underbrace{S_{n_i}}_{\text{numerical sensitivity}} \tag{20}$$

[2]Increasing the candidate rate changes the model being evaluated while increasing the true rate changes the underlying estimation problem.

The gradient has the same structure as the exact shooting gradient. The question is therefore whether the numerical sensitivities S_{n_i}, remain substantial for a large candidate rate.

Holding h fixed gives $\partial_\beta z_\beta = z_\beta$. Differentiating the update gives:

$$\begin{aligned} S_{n+1} &= \mathcal{R}(z_\beta) S_n + \mathcal{R}'(z_\beta)\partial_\beta z_\beta u_n \\ &= \mathcal{R}(z_\beta) S_n + z_\beta \mathcal{R}'(z_\beta) u_n \end{aligned}$$

Here $\mathcal{R}'$ is the differentiation with respect to the argument. Observe that the first term carries forward the present sensitivity and the second carries the additional sensitivity due to changing the rate at that step. The sensitivity at an observation time accumulates the contributions over the following steps.

To obtain a concrete bound, consider an A-stable method with a rational stability function: e.g. an A-stable Runge Kutta method such as Radau IIA above. Here rational means $\mathcal{R} = P/Q$ where P and Q are polynomials.

A-stability gives $|\mathcal{R}(z_\beta)| \leq 1$. Boundedness as $z_\beta \to -\infty$ therefore requires $\deg P \leq \deg Q$. The quotient rule then gives

$$\mathcal{R}'(z_\beta) = \frac{(P'Q - PQ')(z_\beta)}{Q(z_\beta)^2} = O(|z_\beta|^{-2}).$$

To see the last bound, if $\deg P \leq \deg Q$, the numerator has degree at least two below that of the denominator. If the degrees are equal, the leading terms in $P'Q$ and PQ' cancel, giving the same bound. Consequently, $z_\beta \mathcal{R}'(z_\beta) = O(|z_\beta|^{-1}) \to 0$ as $\beta \to \infty$.

We can now compute the prediction sensitivity direction. Repeated application of the numerical update gives:

$$u_n = \mathcal{R}(z_\beta)^n u_0.$$

Since u_0 is independent of β and $\partial_\beta z_\beta = z_\beta$, differentiation yields, for $n \geq 1$,

$$S_n = n\mathcal{R}(z_\beta)^{n-1} z_\beta \mathcal{R}'(z_\beta) u_0$$

Using $|\mathcal{R}(z_\beta)| \leq 1$ implies

$$|S_n| \leq n|u_0||z_\beta \mathcal{R}'(z_\beta)| = O(|z_\beta|^{-1}) \to 0$$

for fixed n. A-stability also gives $|u_{n_i} - y_i| \leq |u_0| + |y_i|$, so the prediction error factors in equation 20 remain bounded. Thus

$$L'_{S,h}(\beta) = \frac{2}{m}\sum_i (u_{n_i} - y_i) S_{n_i} \leq \frac{2}{m}\sum_i (|u_0| + |y_i|) n |u_0||z_\beta \mathcal{R}'(z_\beta)| \to 0.$$

Thus stable numerical methods can still supply a vanishing sensitivity to the training gradient. Indeed, for an L-stable method, we additionally have $\mathcal{R}(z_\beta) \to 0$. Since every observation index is positive,

$$u_{n_i} = \mathcal{R}(z_\beta)^{n_i} u_0 \to 0.$$

Thus, similarly,

$$L_{S,h} \to \frac{1}{m}\sum_i y_i^2 > 0, \qquad L'_{S,h}(\beta) \to 0.$$

Thus the numerical shooting objective approaches the same positive plateau as exact shooting, and its gradient again vanishes.

E.2 Derivative matching

Both shooting calculations vanish for the same structural reason. The exact sensitivity $\partial_\beta u_\beta(t_i) = -at_i u_\beta(t_i)$ is proportional to the prediction itself, and the growing factor at_i does not compensate for the exponential decay of $u_\beta(t_i)$. The numerical sensitivity carries the factor $z_\beta \mathcal{R}'(z_\beta)$ which the stability function drives to zero. In each case, the candidate meets the data at a state it generated by propagating from u_0 across the whole interval $[0, t_i]$, and an overestimated rate erases its own imprint during that propagation. We therefore ask what training signal survives if the comparison is made at states that the observations supply.

Derivative matching makes exactly this comparison. It evaluates the candidate vector-field $f_\beta(y) = -\rho e^\beta y$ at each observed state and compares the value with a slope d_i estimated from the data. To examine what this changes, keep the observations fixed and take exact targets $d_i = -a_* y_i$, with at least one $y_i \neq 0$. The local discrepancy is

$$f_\beta(y_i) - d_i = -\rho e^\beta y_i + a_* y_i = -(a - a_*) y_i.$$

The observed state enters as a fixed multiplier, so at any nonzero y_i the discrepancy grows linearly with the excess rate. Observe that nothing here is propagated, so the decay that flattened shooting does not reach this residual.

Training updates the log rate: $a = \rho e^\beta$, so a step of $\Delta\beta$ multiplies the candidate rate by $e^{\Delta\beta}$. In this coordinate the raw discrepancy is the wrong quantity to square. Its squared average has gradient

$$\partial_\beta \Big(\frac{1}{m} \sum_i (a - a_*)^2 y_i^2 \Big) = \frac{2}{m} \sum_i a(a - a_*) y_i^2 = O(e^{2\beta}),$$

so a learning rate that is stable near the true rate produces a divergent step at a large candidate rate. We want a comparison that keeps the discrepancy visible at large rates without growing exponentially in the coordinate being optimized.

The warm start therefore compares the slopes after an asinh transformation:

$$r_i(\beta) = \operatorname{asinh}\left(\frac{-\rho e^\beta y_i}{\kappa} \right) - \operatorname{asinh}\left(\frac{d_i}{\kappa} \right), \qquad L_D(\beta) = \frac{1}{m} \sum_{i=1}^m r_i(\beta)^2. \tag{21}$$

Here $\kappa > 0$ sets the slope magnitude at which the transformation changes from approximately linear to approximately logarithmic. Slopes well below κ are compared as they stand, and slopes well above it are compared through their logarithms. Both κ and the rate scale $\rho > 0$ stay fixed throughout, and the normalization agrees with the derivative loss in the main text.

Write $\beta_* = \log(a_*/\rho)$ for the true log rate. The transformation and the coordinate then fit together. When both $a|y_i|$ and $a_*|y_i|$ are large relative to κ,

$$r_i(\beta) \approx -\operatorname{sign}(y_i) \log(a/a_*) = -\operatorname{sign}(y_i)(\beta - \beta_*)$$

The residual measures an error in the coordinate the optimizer actually updates, and the squared residual behaves like a squared log rate error. The exponential growth above becomes linear growth.

Differentiating the residual shows where the parameterization enters its sensitivity. By the chain rule,

$$r_i'(\beta) = a\, \partial_a r_i = -\frac{a y_i}{\sqrt{\kappa^2 + a^2 y_i^2}}, \tag{22}$$

so $|r_i'(\beta)| \to 1$ as $\beta \to \infty$ for every fixed nonzero state. Two opposing factors balance exactly. The sensitivity with respect to the physical rate, $|\partial_a r_i| = |y_i| / \sqrt{\kappa^2 + a^2 y_i^2}$, decays like $1/a$, and the factor $\partial_\beta a = a$ contributed by the exponential parameterization cancels that decay. A head producing the rate directly would inherit the decaying version.

The gradient of the derivative loss has the same two-factor structure as the shooting gradient equation 16,

$$L_D'(\beta) = \frac{2}{m} \sum_i \underbrace{r_i(\beta)}_{\text{transformed slope error}} \overbrace{r_i'(\beta)}^{\text{log-rate sensitivity}},$$

with the difference that neither factor decays. The error grows like $|\beta - \beta_*|$ and the sensitivity tends to one. For $\beta > \beta_*$ the two have the same sign at every nonzero state, both negative when $y_i > 0$ and both positive when $y_i < 0$. Every observation therefore contributes positively and gradient descent reduces the candidate rate.

How strongly it does so depends on the size of the target slopes relative to κ. Define

$$\eta = \frac{1}{m} \sum_{i=1}^m \frac{d_i^2}{\kappa^2 + d_i^2}.$$

With exact targets and assuming $y_i \neq 0$ for each retained i, $\eta > 0$.

Proposition 3. *Let $d_i = -a_* y_i$. For $\beta \geq \beta_*$,*

$$L_D''(\beta) \geq 2\eta, \qquad L_D'(\beta) \geq 2\eta(\beta - \beta_*). \tag{23}$$

Proof. For each $y_i \neq 0$, use the oddness of asinh to write

$$g_i(\beta) := -\operatorname{sign}(y_i) r_i(\beta) = \operatorname{asinh}(q_i) - \operatorname{asinh}(q_{i,*}),$$

with $q_i = \frac{a|y_i|}{\kappa}$, and $q_{i,*} = \frac{a_*|y_i|}{\kappa}$. This preserves the squared residual. Terms with $y_i = 0$ vanish identically. Since $q_i' = q_i$,

$$g_i' = \frac{q_i}{\sqrt{1+q_i^2}}, \qquad g_i'' = \frac{q_i}{(1+q_i^2)^{3/2}}.$$

For $\beta \geq \beta_*$, we have $q_i \geq q_{i,*}$ and $g_i \geq 0$. Consequently,

$$L_D''(\beta) = \frac{2}{m}\sum_i \big((g_i')^2 + g_i g_i''\big) \geq \frac{2}{m}\sum_i \frac{q_{i,*}^2}{1+q_{i,*}^2} = 2\eta.$$

Since $L_D'(\beta_*) = 0$, integrating this bound over $[\beta_*, \beta]$ gives the gradient inequality. □

The proposition answers our initial question. While the shooting gradient decays to zero as the candidate rate grows, the derivative gradient grows at least linearly in the excess log rate, on a loss whose curvature never falls below 2η. A gradient step of fixed size α therefore moves β down by at least $2\alpha\eta(\beta - \beta_*)$ (a fixed fraction of the excess) provided the step does not carry β past β_*. Shooting, by $equation$ 18, takes steps that shrink to nothing in the same limit.

How large that fraction is depends on the size of the target slopes relative to κ. Indeed, if a fraction p of the observations satisfy $|d_i| \geq c\kappa$ for some $c > 0$, then $\eta \geq pc^2/(1+c^2)$. This becomes concrete because both scales come from the data. Section 3.1 sets $\kappa = 10^{-3}s$ and $\rho = s/y_{\text{scale}}$ from the same observed slope scale s, so $\kappa = 10^{-3}\rho y_{\text{scale}}$ and

$$\frac{|d_i|}{\kappa} = 10^3 e^{\beta_*}\frac{|y_i|}{y_{\text{scale}}}.$$

Observe, then, that an observation contributes at least $1/2$ to η whenever $|y_i| \geq 10^{-3}e^{-\beta_*}y_{\text{scale}}$. For a true rate comparable to the reference rate ρ, where $\beta_* \approx 0$, this covers every state within three orders of magnitude of the observed ranges, and η is close to the fraction of retained observations lying there. The floor weakens only once the retained states have decayed well below that. It is also conservative in the regime of interest. In fact, at large candidate rates every fixed nonzero state has $a|y_i| \gg \kappa$, and $L_D''(\beta) \to 2p_0$ where p_0 is the fraction of nonzero observed states. States whose targets fall below κ still contribute fully once the candidate rate is large enough.

We remark that the correction is not symmetric about β_*. For $\beta < \beta_*$ the residual and its sensitivity have opposite signs at every nonzero state, so $L_D'(\beta) < 0$. With $equation$ 23 this makes β_* the unique stationary point and the global minimizer. But the gradient weakens as the rate falls, because the factor $\partial_\beta a = a$ that rescued the sensitivity at large rates now works against us. Indeed, expanding the residuals about $a = 0$ gives

$$\begin{aligned} L_D(\beta) &\longrightarrow \frac{1}{m}\sum_i \operatorname{asinh}(d_i/\kappa)^2, \\ L_D'(\beta) &= -\frac{2a}{m\kappa}\sum_i |y_i| \operatorname{asinh}(|d_i|/\kappa) + O(a^2), \end{aligned} \tag{24}$$

and L_D'' has the same leading term, so the loss is nearly flat and locally concave there even though its gradient still points towards β_*. This half line is where the implemented warm start begins. The rate head is initialized with bias -6 in the guaranteed damping form (cf. Appendix B.3), giving $\beta_0 \approx -6$ and a strong underestimate. The same scale relation puts $a|y_i|/\kappa \approx 2.5|y_i|/y_{\text{scale}}$ at that initialization, so the starting point sits near the transition rather than deep in the flat region, and the first steps carry β into the range where Proposition 3 applies.

Note that the targets d_i are not observed. The warm start estimates them from the data, and fitting estimated slopes recovers whatever rate those slopes imply rather than a_*. This is important, and the

size of the discrepancy is computable for the scalar problem. For exact observations on a uniform grid of spacing Δ, substituting $y_{i\pm1} = y_i e^{\mp a_* \Delta}$ into the centered difference gives

$$\widehat{d}_i = \frac{y_{i+1} - y_{i-1}}{2\Delta} = -a_{\mathrm{CD}} y_i \tag{25}$$

where $a_{\mathrm{CD}} = \frac{\sinh a_* \Delta}{\Delta}$. This is the uniform grid version of the stencil used in the implementation. Every target is proportional to its own state, so the estimated targets are the exact targets of a different rate! Moreover, fitting them exactly returns a_{CD} in place of a_*. Thus, the relative bias is:

$$\frac{a_{\mathrm{CD}} - a_*}{a_*} = \frac{\sinh(a_*\Delta)}{a_*\Delta} - 1 = \frac{(a_*\Delta)^2}{6} + O((a_*\Delta)^4),$$

which is small on a resolved interval and large otherwise. Its direction, of course, depends on the stencil. For example, a forward difference gives $a_{FD} = (1 - e^{-a_*\Delta})/\Delta < a_*$ instead.

Unfortunately, a general stiff problem has no single $a_*\Delta$, so rather than track a particular stencil, we ask what any target accuracy buys. Let $\widehat{L}_D$ denote equation 21 with fixed estimates $\widehat{d}_i$ in place of d_i, with the states and scales unchanged. Moreover, suppose the retained targets are accurate in relative terms. Namely,

$$|\widehat{d}_i - d_i| \le \varepsilon |d_i| \tag{26}$$

for every retained i with $0 \le \varepsilon < 1$. The centered difference calculation satisfies this with $\varepsilon = \sinh(a_*\Delta)/(a_*\Delta) - 1$ on a uniform grid. The condition asks for no more than a bound on the relative error, and it makes no independence or zero mean assumptions, so it admits the correlated errors that finite differences produce.

Proposition 4. *Under equation 26, the loss $\widehat{L}_D$ attains its global minimum, and every finite stationary point satisfies*

$$(1 - \varepsilon) a_* \le a \le (1 + \varepsilon) a_*. \tag{27}$$

Proof. Fix a nonzero state and let $a_i = -\widehat{d}_i / y_i$ be the rate its estimated target implies. Since $a_* = -d_i/y_i$,

$$a_i - a_* = -\frac{\widehat{d}_i - d_i}{y_i}, \qquad |a_i - a_*| \le \frac{\varepsilon |d_i|}{|y_i|} = \varepsilon a_*,$$

using equation 26 and $|d_i| = a_* |y_i|$. The state cancels, so $a_i \in [(1-\varepsilon)a_*, (1+\varepsilon)a_*]$ and, because $\varepsilon < 1$, $a_i > 0$. Each observation is therefore an exact target for its own positive apparent rate, and the argument of Proposition 3 applies to it individually: after multiplication by $-\operatorname{sign}(y_i)$ its residual is

$$\operatorname{asinh}(a|y_i|/\kappa) - \operatorname{asinh}(a_i|y_i|/\kappa),$$

which has the sign of $a - a_i$ and increases in β, so its contribution to $\widehat{L}'_D$ has the sign of $a - a_i$.

Let $\beta_{\min} \le \beta_{\max}$ be the smallest and largest apparent log rates. For $\beta < \beta_{\min}$ every contribution is negative and for $\beta > \beta_{\max}$ every one is positive, so all stationary points lie in the compact interval $[\beta_{\min}, \beta_{\max}]$, where a minimum is attained; the sign pattern outside makes it global. The bound on the rates gives equation 27. □

Reading the loss term by term also extends Proposition 3 to estimated targets. Above every apparent rate each observation is in its own overestimated-rate regime, so the curvature bound of that proposition applies to each term, and for $\beta \ge \beta_{\max}$,

$$\widehat{L}''_D(\beta) \ge 2\widehat{\eta}, \qquad \widehat{\eta} = \frac{1}{m} \sum_i \frac{\widehat{d}_i^2}{\kappa^2 + \widehat{d}_i^2} \ge (1 - \varepsilon)^2 \eta, \tag{28}$$

the last inequality because $t \mapsto t^2/(\kappa^2 + t^2)$ increases in $|t|$ and $|\widehat{d}_i| \ge (1-\varepsilon)|d_i|$. The estimated targets simply replace the exact ones in the curvature constant, and a relative target error of ε costs at most a factor $(1 - \varepsilon)^2$.

Relative accuracy in the targets thus transfers directly to relative accuracy in the fitted rate, at any rate magnitude, and the strong convexity of Proposition 3 survives with a proportionally weakened

constant. This is part of why derivative matching becomes such a useful warm start. The targets need only to be accurate in proportion to the slopes they estimate.

The relative error assumption turns out to be stronger than the guarantee supplied by the target filter. For noiseless scalar decay on a uniform grid, the ratio of adjacent slope magnitudes is $e^{a_*\Delta}$. The filter threshold of 100 therefore permits centered-difference targets with $a_{\mathrm{CD}}/a_* \approx 10.9$, so a retained target doesn't necessarily satisfy the proposition's accuracy requirement.

However, we can still bound the effect of the target errors without a relative guarantee. Keep the states, scales and target indices fixed, and define

$$e_i = \operatorname{asinh}(\widehat{d}_i/\kappa) - \operatorname{asinh}(d_i/\kappa), \qquad E = \frac{1}{m}\sum_i |e_i|.$$

The perturbed residual is $r_i - e_i$. Since e_i is independent of β and equation 22 gives $|r_i'| \le 1$,

$$\left|\widehat{L}_D'(\beta) - L_D'(\beta)\right| = \left|-\frac{2}{m}\sum_i e_i r_i'(\beta)\right| \le 2E.$$

Combining this estimate with Proposition 3 gives

$$\widehat{L}_D'(\beta) \ge 2\eta(\beta - \beta_*) - 2E, \tag{29}$$

for $\beta \ge \beta_*$. Thus, a restoring gradient is guaranteed whenever $\beta > \beta_* + E/\eta$. The estimate is deterministic and allows correlated target errors. Moreover, the mean value theorem gives

$$E \le \frac{1}{m\kappa}\sum_i |\widehat{d}_i - d_i|.$$

Thus, fixed target errors perturb the log-rate gradient by an amount bounded independently of the candidate rate. The exact-target gradient grows at least linearly with the excess log rate, so sufficiently overestimated rates retain a restoring gradient even when the relative-error condition fails, and moreover the correction mechanism is preserved even where shooting becomes insensitive.

E.3 Joint collocation

Derivative matching holds the states fixed at the observations while collocation makes them variables. Indeed, all of the mesh states, the stage increments, and the network parameters are optimized together, so the trajectory that the residual sees is chosen to sit near the data rather than propagated from u_0 by the candidate. The rate is therefore compared against that trajectory instead and it stays there as the candidate rate changes.

Unfortunately, free states also create a different opportunity for the gradient information to go missing. In particular, an optimizer presented with a large dynamics residual can reduce it by moving the states instead of the rate, and whether it can do so completely depends on whether the change the rate makes to the residual is one the states can make as well. Two things must therefore hold. The residual must be sensitive to the rate at a state that the candidate did not produce, and a step of the joint problem must deliver some of that sensitivity to β rather than spend it on the states.

The first is a question about a derivative, and in the two objectives so far the answer came from a single quantity: the sensitivity. Namely, the prediction sensitivity $\partial_\beta u_\beta(t_i)$ in equation 16, which decayed, and the residual sensitivity $r_i'(\beta)$ in equation 22, which did not. Here, the corresponding quantity is the derivative of the dynamics residual with respect to the parameters θ.

Write $r^{\mathrm{stage}}_{i\ell,c}$ for the stage residuals in Appendix C.2 on interval i, at Radau stage ℓ, in state coordinate c, and write $r^{\mathrm{cont}}, r^{\mathrm{data}}$ for the continuity and data residuals respectively. Let $D = \operatorname{diag}(y_{\mathrm{scale}})$ be the fixed state scale, $\widehat{U}_i$ the normalized mesh state, $\widehat{K}_{i\ell}$ the normalized stage increments, $\widehat{k}_c$ the per coordinate residual scale of Appendix C.3, and

$$V_{i\ell} = D(\widehat{U}_i + \sum_j A_{\ell j}\widehat{K}_{ij})$$

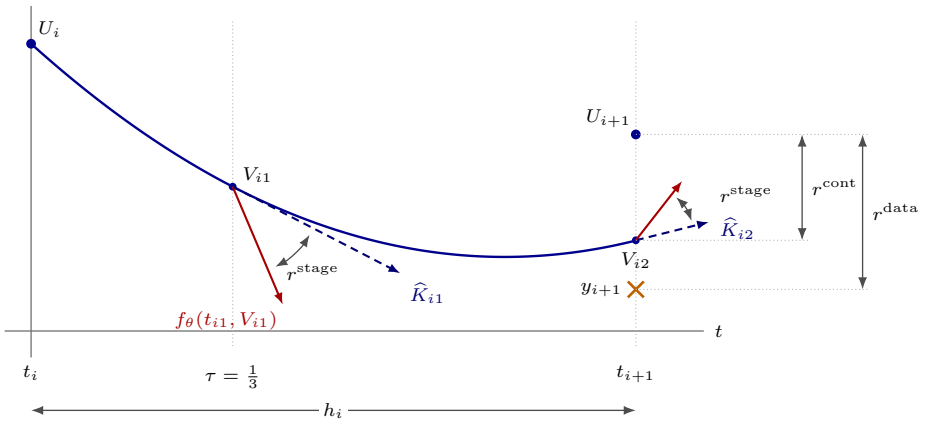


Figure 12: One collocation interval, with its unknowns and three residuals.

the physical stage state. The three residual blocks of Appendix C.2 are

$$r^{\text{stage}}_{i\ell,c} = \frac{1}{\widehat{k}_c}\Big(\widehat{K}_{i\ell,c} - \frac{h_i}{y_{\text{scale}}} f_{\theta,c}(t_{i\ell}, V_{i\ell})\Big), \tag{30}$$

$$r^{\text{cont}}_{i,c} = \widehat{U}_{i+1,c} - \widehat{U}_{i,c} - \sum_{\ell} b_\ell \widehat{K}_{i\ell,c}, \tag{31}$$

$$r^{\text{data}}_{n,c} = \widehat{U}_{n,c} - \frac{y_{n,c}}{y_{\text{scale},c}}, \tag{32}$$

the last at the mesh point carrying observation n. These are the three gaps of the scalar case illustrated by Figure 12.

Observe that only equation 30 contains f_θ, so J_θ is the Jacobian of the stage rows, and differentiating one of them with the mesh variables held fixed gives:

$$\partial_\theta r^{\text{stage}}_{i\ell,c} = -\frac{h_i}{y_{\text{scale}}\widehat{k}_c} \partial_\theta f_{\theta,c}(t_{i\ell}, V_{i\ell}). \tag{33}$$

Compared against the derivative matching Jacobian,

$$\partial_\theta r^{\text{deriv}}_{i,c} = \frac{\partial_\theta f_{\theta,c}(t_i, y_i)}{\sqrt{\kappa_c^2 + f_{\theta,c}(t_i, y_i)^2}}.$$

we see that both are the local parameter derivative of the vector-field at a state but rescaled. Neither contains $\partial_\beta u_\beta(t_i)$ or its numerical counterpart S_{n_i}, so the attenuation established in Appendix E.1 does not enter either one. We therefore meet the first requirement. The quantity in $equation$ 33 is the parameter derivative of the vector-field at the stage state, scaled by fixed constants. It doesn't decay with the candidate rate and doesn't pass through a stability function.

The second requirement is a different question, because equation 33 holds the mesh variables fixed and a step does not. A LM step moves $\widehat{U}$, $\widehat{K}$ and θ simultaneously and the stage residual responds to all three. If the change that $\delta\theta$ makes to that residual is one that some δX also makes, then the two are interchangeable in the linearized problem which doesn't determine which one is used. We thus should check how much of the parameter sensitivity survives once the state variables have absorbed whatever they can.

Absorb the fixed residual weights into r, collect the mesh states and stage increments into X, and write the linearization as $r + J_X\delta X + J_\theta\delta\theta$, where the rows of J_θ are the derivatives equation 33. Minimizing over δX with no damping and substituting back leaves the parameter curvature

$$J_\theta^\top (I - J_X J_X^\dagger) J_\theta, \tag{34}$$

with $J_X^\dagger$ the Moore–Penrose pseudoinverse. Observe that the middle factor projects onto the directions that no state change can reproduce, which is exactly the previous concern. Along any d with

$J_\theta d \in \mathrm{range}(J_X)$ the reduced curvature vanishes and the linearized problem does not determine the step there. Observe that a large $J_\theta d$ lying in that range contributes nothing, so the question is then about the position of J_θ relative to the span of the state columns.

Damping improves this problem but does not resolve it. Indeed, with positive definite damping M_X on the state block, the same elimination gives:

$$P_M = I - J_X(J_X^\top J_X + M_X)^{-1}J_X^\top, \qquad (J_\theta^\top P_M J_\theta + M_\theta)\,\delta\theta = -J_\theta^\top P_M r, \tag{35}$$

and P_M is now positive definite rather than a projection, so no parameter direction is annihilated outright. However, the smallest eigenvalue of P_M may tend to zero as the candidate rate grows, and $J_\theta^\top P_M r$ may vanish with it, so positive definiteness at each rate supplies no uniform bound in the rate. How much reaches θ depends on the damping and the residual weights via quantities that must be evaluated. So, next, we evaluate them on the scalar problem.

One interval suffices. In fact, the sensitivity is local to a stage, so the competition in equation 35 is between one rate column and the state columns, and both live inside a single interval. So take $u' = -au$ on an interval of width h in normalized units. Begin at u_0 and carry an endpoint observation y with $u_0 > y > 0$. The two stage Radau IIA coefficients are

$$A = \begin{pmatrix} 5/12 & -1/12 \\ 3/4 & 1/4 \end{pmatrix}, \qquad b = \begin{pmatrix} 3/4 \\ 1/4 \end{pmatrix},$$

and with $s = ha$, stage increments $k \in \mathbb{R}^2$, endpoint variable x, and stage states $V = u_0\mathbf{1} + Ak$, the residual and objective are

$$r_C(k, x, \beta) = \begin{pmatrix} W(k + sV) \\ \sqrt{w_c}\,(x - u_0 - b^\top k) \\ \sqrt{w_d}\,(x - y) \end{pmatrix}, \qquad F_C = \tfrac{1}{2}\|r_C\|^2,$$

the three blocks of equation 30–equation 32 with W a fixed positive diagonal stage weight and $w_c, w_d > 0$ fixed. Here, k, x, and β play the roles of $\widehat{K}$, $\widehat{U}_{i+1}$ and θ, and are optimized together.

Starting where the implementation starts, Appendix C.3 initializes the mesh states by interpolating the observations and the stage increments by the change between surrounding nodes, which here is $x_0 = y$ and $k_0 = (y - u_0)\mathbf{1}$. The continuity and data residuals vanish there and only the dynamics residual is nonzero: the interpolated trajectory passes through the data, and an overestimated candidate rate disagrees with it. This is the configuration of Figure 12 with its two right-hand gaps closed. Write $V_0 = u_0\mathbf{1} + Ak_0$ for the stage states there and

$$v = A^{-1}V_0 = \begin{pmatrix} u_0 + y \\ y - 3u_0 \end{pmatrix}.$$

Fix $\mu > 0$ and $\rho > 0$. For each $s > 0$, write

$$\beta_s = \log\left(\frac{s}{h\rho}\right) \qquad w_s^0 = (k_0, y, \beta_s)$$

so that $s = h\rho e^{\beta_s}$. We examine the proposed LM step of Section 3.2.2, with fixed identity damping:

$$\delta_s = (\delta k_s, \delta x_s, \delta\beta_s) = \arg\min_p \{\|r_C(w_s^0) + J_C p\|^2 + \mu\|p\|^2\}, \qquad J_C = \partial_{(k,x,\beta)} r_C, \tag{36}$$

taken at the initialization above. The index s identifies the starting rate so that each step is computed from the same observation based initialization of the trajectory variables.

A correction bounded away from zero removes a fixed proportion of the rate however large the overestimate. The question is thus what fraction of that correction the step takes, and whether that fraction survives as $s \to \infty$. It turns out to be

$$c = \frac{q_\mu}{q_\mu + \mu} \in (0, 1), \tag{37}$$

$$q_\mu = \mu\|v\|^2 + \frac{w_c(w_d + \mu)}{w_c + w_d + \mu}\, y^2 > 0 \tag{38}$$

where q_μ is the coefficient left after the state variables are eliminated.

Proposition 5. *Assume $u_0 > y > 0$, $h, \rho, w_c, w_d, \mu > 0$ and that W is positive diagonal. Hold these quantities fixed. At the initialized points w_s^0, the proposed steps satisfy*

$$\delta\beta \longrightarrow -c \tag{39}$$

as $s \to \infty$. For the proposed update, let $a_s = \rho e^{\beta_s} = \frac{s}{h}$ and $a_s^+ = \rho e^{\beta_s + \delta\beta_s}$. Then

$$\frac{a_s^+}{a_s} = e^{\delta\beta_s} \longrightarrow e^{-c} < 1. \tag{40}$$

Shooting with the same Radau formula, initial value, endpoint observation, observation weight, log-rate coordinate, and damping instead gives $\delta\beta_S = O(s^{-1})$.

Proof. Write $p = (p_k, p_x, p_\beta)$, and let δ_s denote the minimizing step at rate s. Since $V = u_0\mathbf{1} + Ak$, the derivatives of $k + sV$ with respect to k and β are $I + sA$ and sV respectively. The continuity and data residual vanish at that initialization. The LM objective therefore takes the form

$$\mathcal{Q}_s(p) = \|W[k_0 + (I + sA)p_k + sV_0(1 + p_\beta)]\|^2 + \Psi(p),$$

where

$$\Psi(p) = w_c(p_x - b^\top p_k)^2 + w_d p_x^2 + \mu\|p\|^2.$$

Here Ψ collects the continuity, data, and damping terms which are independent of s. First bound the minimizing steps uniformly in s. For the trial step $p = (0, 0, -1)$, observe that the term proportional to s disappears from the linearized dynamics residual, leaving Wk_0. Consequently,

$$\mu\|\delta_s\|^2 \le \mathcal{Q}_s(\delta_s) \le \mathcal{Q}_s(0, 0, -1) = \|Wk_0\|^2 + \mu =: C$$

The constant C is independent of s. Moreover, since $\mu > 0$, the steps are bounded so this inequality also bounds their linearized dynamics residuals. Dividing those residuals by s gives

$$\left\| \frac{k_0 + \delta k_s}{s} + A\delta k_s + V_0(1 + \delta\beta_s) \right\| \le \frac{\|W^{-1}\|\sqrt{C}}{s} \to 0.$$

Every limit point $\bar{\delta} = (\bar{\delta}_k, \bar{\delta}_x, \bar{\delta}_\beta)$ therefore satisfies

$$A\bar{\delta}_k + V_0(1 + \bar{\delta}_\beta) = 0$$

where $\bar{\delta}_k = -v(1 + \bar{\delta}_\beta)$. Thus bounded objective values force a relation between stage correction and the rate correction.

To determine which step satisfying this relation is selected, we must also establish that limit points minimize Ψ subject to it. Fix any step p satisfying

$$Ap_k + V_0(1 + p_\beta) = 0.$$

Construct comparison steps by keeping p_x, p_β unchanged and choosing

$$\widetilde{p}_{k,s} = -(I + sA)^{-1}\left[k_0 + sV_0(1 + p_\beta)\right].$$

Their linearized dynamics residual is exactly zero:

$$k_0 + (I + sA)\widetilde{p}_{k,s} + sV_0(1 + p_\beta) = 0.$$

Moreover, since A is invertible,

$$\begin{aligned}\widetilde{p}_{k,s} &= -(A + s^{-1}I)^{-1}\left[s^{-1}k_0 + V_0(1 + p_\beta)\right] \\ &\longrightarrow -A^{-1}V_0(1 + p_\beta) = p_k.\end{aligned}$$

Hence $\widetilde{p}_s \to p$. The minimizing property of δ_s now gives

$$\Psi(\delta_s) \le \mathcal{Q}_s(\delta_s) \le \mathcal{Q}_s(\widetilde{p}_s) = \Psi(\widetilde{p}_s).$$

Along any subsequence converging to $\bar{\delta}$, continuity yields

$$\Psi(\bar{\delta}) \le \Psi(p).$$

Since p was arbitrary, every limit point minimizes Ψ subject to the limiting constraint. The term $\mu\|p\|^2$ makes Ψ strictly convex, so this constrained minimizer is unique. Boundedness and uniqueness of the limit point then imply convergence of the entire family δ_s.

It remains to compute this minimizer. Substituting $p_k = -v(1 + p_\beta)$ and using $b^\top v = y$ gives

$$\Psi(p) = w_c\big(p_x + y(1 + p_\beta)\big)^2 + (w_d + \mu)p_x^2 + \mu\|v\|^2(1 + p_\beta)^2 + \mu p_\beta^2.$$

Completing the square in p_x rewrites this expression as

$$\Psi(p) = (w_c + w_d + \mu)\left(p_x + \frac{w_c y}{w_c + w_d + \mu}(1 + p_\beta)\right)^2 + q_\mu(1 + p_\beta)^2 + \mu p_\beta^2.$$

For each p_β, the first square is minimized by setting it to zero. The remaining scalar quadratic has derivative

$$2q_\mu(1 + p_\beta) + 2\mu p_\beta.$$

Its unique minimizer is

$$p_\beta = -\frac{q_\mu}{q_\mu + \mu} = -c.$$

The convergence established above therefore gives

$$\begin{aligned} \delta\beta_s &\longrightarrow -c, \\ \delta k_s &\longrightarrow -(1 - c)v, \\ \delta x_s &\longrightarrow -\frac{w_c y}{w_c + w_d + \mu}(1 - c). \end{aligned} \tag{41}$$

The latter two limits follow from the limiting constraint and the minimizing value of p_x, respectively. The rate ratio in equation 40 follows by exponentiating the first limit.

For shooting, the stage equations are solved at each candidate rate:

$$(I + sA)k = -su_0\mathbf{1}.$$

The resulting endpoint prediction is $u_0 R_s$, where

$$R_s = 1 - sb^\top(I + sA)^{-1}\mathbf{1} = \frac{1 - s/3}{1 + 2s/3 + s^2/6} = -\frac{2}{s} + O(s^{-2}).$$

With the same observation weight, define

$$r_S = \sqrt{w_d}(u_0 R_s - y), \qquad j_S = \partial_\beta r_S = \sqrt{w_d}\, u_0 s\, \frac{dR_s}{ds}.$$

The rational expression for R_s gives

$$r_S = -\sqrt{w_d}\, y + O(s^{-1}), \qquad j_S = \frac{2\sqrt{w_d}\, u_0}{s} + O(s^{-2}).$$

Minimizing $(r_S + j_S p_\beta)^2 + \mu p_\beta^2$ therefore yields

$$\delta\beta_S = -\frac{j_S r_S}{j_S^2 + \mu} = \frac{2w_d u_0 y}{\mu s} + O(s^{-2}).$$

Thus proving the shooting step comparison. □

We have the following immediate corollary.

Corollary 1. *Under the assumptions of Proposition 5, let $w_s^+ = w_s^0 + \delta_s$ denote the full proposed update. Then*

$$\frac{F_C(w_s^+)}{F_C(w_s^0)} \longrightarrow c^2 e^{-2c} < 1 \tag{42}$$

as $s \to \infty$. Consequently, $F_C(w_s^+) < F_C(w_s^0)$ for all sufficiently large s.

Proof. Write

$$k_s^+ = k_0 + \delta k_s, \qquad V_s^+ = u_0 \mathbf{1} + A k_s^+.$$

By equation 41 and $Av = V_0$,

$$V_s^+ = V_0 + A\delta k_s \to cV_0 \qquad e^{\delta\beta_s} \to e^{-c}.$$

The updated dimensionless rate is thus $se^{\delta\beta_s}$. Hence the stage residual after the full update, divided by the starting rate s satisfies

$$\frac{1}{s} W \left(k_s^+ + se^{\delta\beta_s} V_s^+\right) = W\left(\frac{k_s^+}{s} + e^{\delta\beta_s} V_s^+\right) \to ce^{-c} W V_0.$$

The continuity and data residuals remain bounded because δk_s and δx_s converge. Therefore

$$\frac{r_C(w_s^0)}{s} \to \begin{pmatrix} WV_0 \\ 0 \\ 0 \end{pmatrix}, \qquad \frac{r_C(w_s^+)}{s} \to \begin{pmatrix} ce^{-c} WV_0 \\ 0 \\ 0 \end{pmatrix}.$$

Since $WV_0 \neq 0$ and $F_C = \frac{1}{2}\|r_C\|^2$, taking the ratio of squared norms proves equation 42. The strict inequality follows from $0 < c < 1$. □

The second requirement is therefore satisfied at the observation based initialization. Proposition 5 shows that the joint linearized problem retains a nonvanishing log rate correction even when the trajectory variables are free to move. Corollary 1 further shows that taking the full proposal decreases the nonlinear collocation objective for sufficiently large candidate rates. Thus these results establish a useful correction despite the attenuation of shooting sensitivities.

Further progress depends on the subsequent state configurations and damping. For the scalar fixed mesh problem, one can pursue a continuation argument by first minimizing over the state variables at each fixed rate, then identifying a compact sublevel step at low loss that prevents state collapse, then applying sufficient decrease estimates to the damped joint steps. With suitable bounds on damping and a sufficient decrease line search, such an argument can establish convergence to the discrete collocation fit from within that region. Establishing entry into the region from a large rate initialization or derivative matching warm start requires additional arguments. The iteration level analysis requires too much nuance for the present scope of the paper, and thus we restrict our claims here to the initialized correction and its decrease of the nonlinear objective. However, these results still provide evidence for a potential work-around to the Fronk & Petzold (2025b) vanishing sensitivity problem for shooting based neural ODEs, and we leave the additional analysis to future work.

E.4 Identifying fast modes

Even perfect state and derivative fits can leave a fast eigenvalue undetermined. The issue can arise inside the production rate form itself. Indeed for $a > 1$, consider the following example two dimensional family of systems

$$x' = -x, \qquad y' = (a-1)x - ay,$$

with $x(0) = y(0) = 1$. The production and rate terms are

$$P(x,y) = \begin{pmatrix} 0 \\ (a-1)x \end{pmatrix}, \qquad R(x,y) = \begin{pmatrix} -1 \\ -a \end{pmatrix}$$

Writing $z = y - x$ gives $z' = -az$ with $z(0) = 0$. Hence every member of the family produces the same trajectory $x(t) = y(t) = e^{-t}$ and the same derivatives along it. The Jacobian is

$$J_a = \begin{pmatrix} -1 & 0 \\ a-1 & -a \end{pmatrix},$$

with eigenvalues -1 and $-a$. Along the observed line $x = y$, the a-dependent production and destruction terms cancel exactly. The observations therefore leave the transverse decay rate a entirely undetermined.

To see what information is missing, perturb the initial condition to $y(0) = 1 + q$, keeping $x(0) = 1$. Then

$$y(t) = e^{-t} + qe^{-at}, \qquad \partial_{\log a} y(t) = -atqe^{-at}.$$

For known $q \neq 0$, an exact observation at any $t > 0$ determines a. The sensitivity has maximal magnitude $|q|/e$ at $t = 1/a$ and decays thereafter, so small excitation or observations taken after the fast transient make recovery sensitive to noise. Thus we see that agreement with the vector-field along an observed trajectory does not determine its derivatives in transverse directions. Consequently, even perfect trajectory and derivative fits provide no general guarantee of spectral recovery.

E.5 Training Complexity

Let d be the state dimension, M the number of mesh intervals, N the number of observation times, and p the number of network parameters. With two Radau stages per interval, the collocation problem has $d(M + 1) + 2dM + p = 3dM + d + p$ unknowns and $3dM + dN$ residuals. Since the mesh contains every observation time, $N \leq M + 1$, so the number of residuals is $O(dM)$. Each LM iteration evaluates the residual Jacobian and performs a linear solve; with a dense solver this costs $O((dM + p)^3)$ per iteration. Mesh refinement increases M, by up to a factor of 48 over the observation intervals, so the per-iteration cost grows across refinement rounds.