Same Accuracy, Different Geometry: Solver-Dependent Representation Learning in Third-Order Neural ODEs

Gavin Lee Goodship[1], Stephen O'Sullivan[1], Luis Miralles-Pechuán[1,2]
[1]School of Computer Science, Technological University Dublin, Dublin, Ireland
[2]EUT+ Data Science Institute, European University of Technology, European Union
Email: {D23126880, Luis.Miralles, Stephen.OSullivan}@TUDublin.ie

***Abstract*—Models with similar predictive accuracy are often assumed to learn comparable internal representations. We challenge this assumption in continuous-depth networks by showing that the numerical solver used to discretize Neural Ordinary Differential Equations (Neural ODEs) systematically alters the geometry of learned representations, even under matched predictive performance. Using a third-order Neural ODE framework, we study how Runge–Kutta (RK4) and extended-stability solvers (ESRK-15 and ESRK-21) shape latent trajectories in models trained within typical run-to-run accuracy variability (≈2%) on CIFAR-10 and CIFAR-100. Despite identical integration horizons and comparable validation accuracy, the resulting representations exhibit markedly different geometric structures, as quantified by curvature and jerk statistics derived directly from the latent dynamics. Our results show that solver stability properties and internal amplification act as implicit inductive biases that regulate representational smoothness and higher-order dynamics. These findings demonstrate that numerical integration is not merely an implementation detail, but a structural component of the model that influences how continuous-depth networks encode and regularize information.**

***Index Terms*—Neural ODEs, continuous-depth networks, numerical solvers, Runge–Kutta, extended-stability methods, representation geometry, inductive bias.**

I. Introduction

Neural Ordinary Differential Equations (Neural ODEs) [1] reinterpret deep neural networks as continuous-depth dynamical systems, replacing discrete residual layers with continuous-time flows. This formulation establishes a principled connection between deep learning and dynamical systems theory, enabling adaptive computation, invertibility, and links to physically motivated models. Subsequent work has extended Neural ODEs to stochastic dynamics [2], manifold-valued representations [3], and stability-aware architectures [4], highlighting their expressiveness and flexibility.

Despite these advances, the numerical solver used to discretize the continuous dynamics is often treated as an implementation detail. In practice, however, different solvers produce distinct discrete approximations of the same underlying differential equation [5]. These approximations define different discrete flow operators during both training and inference, suggesting that the choice of a solver may influence not only the optimization behavior but also the internal representations learned by the model. Existing studies have largely examined solver selection in terms of numerical stability, efficiency, or optimization dynamics [6], leaving its representational consequences comparatively underexplored.

A widespread assumption in deep learning is that models achieving similar predictive accuracy learn broadly comparable internal representations. In this work, we show that this assumption does not hold for continuous-depth systems. Neural ODEs trained with different numerical solvers can attain nearly identical classification accuracy while learning latent representations with fundamentally different geometric structure. This divergence persists under matched architectures, optimization settings, and integration horizons, indicating that it cannot be attributed to stochastic training variability alone.

To isolate and quantify these effects, we adopt a third-order Neural ODE formulation. By explicitly modeling velocity, acceleration, and jerk, this framework exposes higher-order representational dynamics directly in the system state, allowing curvature and smoothness to be measured without nested differentiation through the learned vector field. This makes it possible to compare the representation geometry across solvers in a numerically stable and solver-consistent manner. Using this setup, we demonstrate that solver stability properties and internal stage amplification act as implicit inductive biases that systematically shape the geometry of learned latent trajectories.

Together, our results position numerical integration not as a neutral computational choice, but as a structural component of continuous-depth neural networks. The solver defines a discrete dynamical system whose properties influence how information is encoded, regularized, and propagated through the model, challenging the notion of representational equivalence among similarly performing continuous-depth architectures.

II. Contributions

In this paper, the term *representation geometry* refers specifically to the curvature and higher-order smoothness of latent trajectories induced by the continuous-depth flow, rather than to static embedding visualizations (e.g., t-SNE or PCA).

Using a third-order Neural ODE framework, we make the following contributions:

- **Accuracy-matched models learn different latent trajectory geometries:** We show that models trained within

typical run-to-run accuracy variability ($\pm 2\%$) can learn markedly different latent-space geometries depending on the numerical solver, demonstrating that similar performance does not imply similar representations.
- **Solvers as implicit inductive biases:** We identify solver stability properties and internal amplification as mechanisms that systematically regulate curvature and smoothness of latent trajectories, acting as implicit regularizers on representation geometry.
- **Algebraic realization matters:** We show that algebraically equivalent solvers with identical order conditions and linear stability polynomials (e.g., different low-storage ESRK realizations) can induce distinct geometric behavior, identifying the solver realization as an additional and previously overlooked source of inductive bias.
- **Third-order Neural ODEs as a geometric probe:** We introduce a third-order Neural ODE formulation that enables direct and solver-consistent measurement of curvature and jerk without nested differentiation.
- **Bridging numerical analysis and representation learning:** We connect classical solver stability theory with modern representation learning, establishing numerical integration as a structural design axis in continuous-time neural networks.

To support reproducibility, we release the full training and evaluation code, including details of the ESRK implementation and coefficients, at https://doi.org/10.5281/zenodo.19211480.

III. Related Work

Our work lies at the intersection of continuous-depth neural networks, numerical analysis, and representation geometry. We review prior work on Neural ODEs, numerical solvers and stability, and higher-order geometric perspectives on representation learning, and position our contribution within these lines of research.

a) Neural ODEs and Continuous-Depth Models: Neural Ordinary Differential Equations (Neural ODEs) were introduced by Chen *et al.* [1] as continuous-depth analogs of residual networks, connecting deep learning with dynamical systems theory [7], [8]. This formulation enables adaptive computation and memory-efficient training [9], and has since been extended to stochastic dynamics [10], latent-variable models [11], manifold-valued representations [3], [12], and controlled or Hamiltonian systems [13], [14]. Existing work has largely focused on expressivity, stability, and computational efficiency, while the geometric consequences of numerical discretization on learned representations have received comparatively less attention.

b) Numerical Solvers and Stability: The choice of a numerical solver does not alter the intrinsic stability of the underlying continuous ODE, but it defines a discrete dynamical system whose stability properties govern optimization dynamics and gradient flow during training. Early approaches relied on adaptive Runge–Kutta methods [15], [1], while later studies emphasized the importance of solver order, step size, and internal amplification for conditioning gradient flow and training stability [5], [6], [16]. Implicit and stabilized schemes have been explored to mitigate stiffness [17], [18], and adaptive or learned solvers have been proposed to improve computational efficiency [19]. From a numerical analysis perspective, the behavior of the solver is fundamentally determined by the structure of the stability polynomial. Extended-stability explicit Runge–Kutta methods, including Runge–Kutta-Chebyshev and Runge–Kutta-Gegenbauer schemes, construct stability polynomials with controlled spectral properties and enlarged real-axis stability intervals [20], [21]. Although these methods were originally developed for efficient integration of moderately stiff systems, they provide a principled mechanism for modifying the stability region without altering the solver order or computational structure. In contrast to previous work, we leverage this polynomial-based construction to treat stability geometry as a controlled variable and study how it influences the geometry of learned representations under fixed integration horizons and accuracy-matched training conditions.

c) Representation Geometry and Higher-Order Dynamics: Geometric perspectives on representation learning have analyzed curvature, smoothness, and sensitivity in deep networks [22], [23], [24], as well as their connections to generalization through optimization trajectories [25], [26]. From a dynamical systems viewpoint, stability and Jacobian conditioning have been linked to smoother representations [7], [5]. Higher-order neural dynamics have also been proposed to model inertial effects and enhance stability [27], [28]. Building on these insights, we employ a third-order Neural ODE as a geometric probe, demonstrating that solver choice and algebraic realization induce systematically different latent geometries even when predictive performance is matched.

In general, prior work has extensively studied solver stability, optimization behavior, and expressivity in continuous-depth models, yet the geometric impact of numerical integration schemes remains underexplored. We address this gap by showing that numerical solvers act as implicit inductive biases that shape representation geometry in third-order Neural ODEs.

IV. Methodology

We describe the continuous-depth model used in our experiments and motivate the use of a third-order Neural ODE to enable stable, solver-consistent analysis of representation geometry.

a) Continuous-Depth Model: We consider continuous-depth neural networks formulated as Neural Ordinary Differential Equations (Neural ODEs). Given an initial representation $z(0) \in \mathbb{R}^d$, the latent state evolves according to

$$\dot{z}(t) = r_\theta(z(t)), \quad t \in [0, \Delta T], \tag{1}$$

where $r_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is a neural network parameterized by θ. The terminal state $z(\Delta T)$ is passed to a linear classifier to produce class logits.

To explicitly expose higher-order representational dynamics, we extend the first-order formulation to a third-order Neural ODE by augmenting the state:

$$\dot{z}_1(t) = z_2(t), \tag{2}$$

$$\dot{z}_2(t) = z_3(t), \tag{3}$$

$$\dot{z}_3(t) = f_\theta(z_1(t), z_2(t), z_3(t)), \tag{4}$$

where $z_1(t) \in \mathbb{R}^d$ represents the latent features, $z_2(t) = \dot{z}_1(t)$ their velocity, and $z_3(t) = \ddot{z}_1(t)$ their acceleration. The neural network f_θ parameterizes the third derivative (jerk) of the latent trajectory. By embedding velocity and acceleration directly into the system state, this formulation makes higher-order geometric properties of the learned flow explicit and directly accessible during both training and inference, without requiring nested differentiation through the numerical solver.

b) Why Third-Order Dynamics: Standard first-order Neural ODEs (1) encode only instantaneous velocity in latent space. Although second-order formulations make acceleration explicit, recovering higher-order quantities such as curvature or jerk still requires nested automatic differentiation through both the learned vector field and the numerical solver, which is numerically fragile and computationally expensive. In contrast, the third-order system (2)–(4) embeds velocity and acceleration directly into the state, allowing curvature and jerk to be computed directly from the trajectory:

$$\kappa(t) \propto \frac{\|z_3(t)\|}{\|z_2(t)\|}, \qquad j(t) = \|\dot{z}_3(t)\| = \|f_\theta(z_1(t), z_2(t), z_3(t))\|. \tag{5}$$

This quantity reflects both directional change and velocity magnitude along the trajectory, and therefore should be interpreted as a bending proxy rather than an intrinsic curvature in the differential geometric sense. This formulation yields numerically stable and solver-consistent measurements of trajectory geometry, enabling a direct comparison of how different solvers shape latent dynamics.

Because all solvers evolve trajectories over identical integration horizons and fixed time parameterizations, $\kappa(t)$ serves as a consistent proxy for representational bending rather than an intrinsic curvature invariant. Importantly, the goal of this formulation is not to increase model expressivity but to ensure numerical identifiability of geometric quantities. In first- and second-order models, curvature and jerk are only implicitly represented and must be reconstructed via repeated differentiation of the learned vector field, involving Jacobians and higher-order derivatives that are tightly coupled to solver discretization, step size, and internal error amplification.

By embedding higher-order derivatives directly into the state, the third-order formulation eliminates solver-induced differentiation artifacts. As a result, observed geometric differences between solvers reflect genuine differences in learned dynamics rather than numerical error. This makes the third-order Neural ODE a minimal yet robust framework for analyzing solver-dependent representation geometry. Throughout our experiments, we therefore use $\|z_3(t)\|/\|z_2(t)\|$ as a relative measure of latent trajectory bending, capturing how sharply representations change direction during evolution.

V. Experiments

We empirically evaluate how numerical solver choice and realization affect representation geometry in third-order Neural ODEs under accuracy-matched training conditions.

a) Experimental Setup: All experiments use a third-order Neural ODE implemented in PyTorch with identical architectures, initializations, and optimization settings. We fix the integration horizon $\Delta T \in \{10, 30\}$ and evaluate each solver with step sizes and step counts chosen to remain well within its stable operating regime. Extended-stability solvers (ESRK) integrate the full horizon in a single step (e.g., $h = 10$ or $h = 30$), whereas RK4 requires multiple smaller steps (e.g., $h = 2.5$ or $h = 7.5$ with four steps) for stable training. We compare RK4 with ESRK-15 and ESRK-21, and additionally evaluate two algebraically equivalent realizations of ESRK-15 (full and 2S–1 storage).

All models are trained with AdamW (learning rate 3×10^{-4}, weight decay 5×10^{-4}), cosine learning-rate scheduling, label smoothing (0.05), and gradient clipping (norm 20). Each setting is repeated over three random seeds. Following prior work showing $\gtrsim$1% run-to-run accuracy variability on CIFAR-scale benchmarks [29], [30], we treat solver configurations within $\pm 2.0\%$ test accuracy as approximately performance-matched. This threshold covers most solver comparisons (15/18), with a maximum difference of 1.55%; therefore, we focus on differences in learned representation geometry.

b) Metrics: We report validation accuracy together with geometric diagnostics from latent trajectories: curvature $\kappa = \|z_3(t)\|/\|z_2(t)\|$ and jerk magnitude $\|f_\theta(z_1(t), z_2(t), z_3(t))\|$. Both are computed online per epoch from the third-order state variables.

c) CIFAR-10 at Fixed Horizon ($\Delta T = 10$): We first analyze solver-dependent geometry on CIFAR-10 with $\Delta T = 10$. All solvers achieve accuracy-matched validation performance (Table I, Fig. 1, top), enabling isolation of solver-induced effects. Figure 1 directly visualizes the central phenomenon of this work: while the validation accuracy overlaps across solvers, the latent trajectory geometry as measured by jerk and curvature clearly separates.

The jerk dynamics (Fig. 1, middle) are separated during training, with ESRK solvers exhibiting consistently larger magnitudes than RK4. At convergence, mean jerk differs by multiple factors (RK4: 9.6×10^3, ESRK-21: 2.8×10^4, ESRK-15: 4.9×10^4). Curvature (Fig. 1, right) shows a consistent ordering: RK4 yields the lowest curvature, followed by ESRK-15 and ESRK-21. Together, these results indicate that solver choice shapes representation geometry independently of predictive accuracy.

d) CIFAR-100 at Fixed Horizon ($\Delta T = 10$): We repeat the analysis in CIFAR-100 under the same horizon. All settings match CIFAR-10 except for the classifier output dimension. Table II reports the validation accuracy and geometric statistics. Despite matched accuracy, ESRK solvers again exhibit

TABLE I: CIFAR-10 validation accuracy at $\Delta T = 10$ (mean $\pm$ std over 3 seeds).

Solver	Val. Acc. (%)
RK4	84.90 ± 0.13
ESRK-15	84.76 ± 0.16
ESRK-21	84.92 ± 0.52

TABLE II: CIFAR-100 results at $\Delta T = 10$ (mean $\pm$ std over 3 seeds).

Solver	Val. Acc. (%)	Jerk ($\times 10^4$)	Curvature
RK4	54.5 ± 0.3	0.9	0.43
ESRK-15	54.8 ± 0.2	3.2	1.10
ESRK-21	54.9 ± 0.4	2.6	1.34

higher curvature and jerk than RK4, indicating that solver-induced geometry persists with increased task difficulty.

e) CIFAR-10 at Extended Horizon ($\Delta T = 30$): Next, we evaluate CIFAR-10 at $\Delta T = 30$, keeping architecture, initialization, optimizer, and training schedule fixed. The validation accuracy remains comparable between solvers (Table III). Geometric effects persist at the longer horizon (Table IV): RK4 yields the lowest curvature and jerk, while ESRK methods induce stronger higher-order dynamics. The solver ordering is preserved across horizons, suggesting intrinsic solver-dependent behavior rather than transient discretization effects.

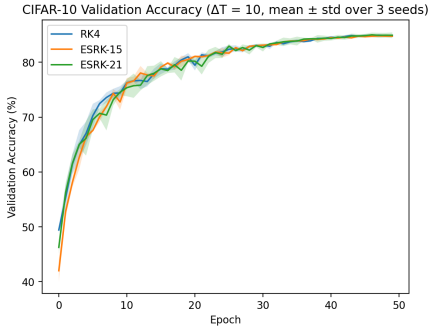
f) Horizon invariance: Across $\Delta T \in \{10, 30\}$ on CIFAR-10, increasing the horizon amplifies higher-order dynamics but does not change the qualitative separation between solvers, supporting the interpretation of numerical solvers as structural inductive biases.

g) CIFAR-100 at Extended Horizon ($\Delta T = 30$): We repeat CIFAR-100 experiments at $\Delta T = 30$ under identical conditions. Validation accuracy remains comparable (Table V), but the geometric ordering reverses relative to CIFAR-10. As shown in Table VI, RK4 exhibits substantially higher jerk ($\sim$240–260$\times 10^3$) and curvature ($\sim$0.75$-$0.80) than ESRK solvers, which converge to lower values. This reversal suggests solver-induced biases interact with task complexity: CIFAR-10 permits higher-curvature regimes under ESRK methods, while CIFAR-100 favors smoother dynamics under extended-stability schemes.

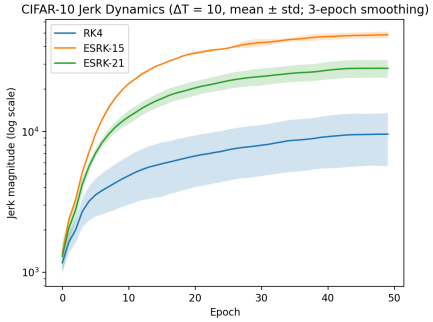
h) Algebraically Equivalent ESRK-15 Realizations: We isolate realization effects by comparing two algebraically equivalent ESRK-15 implementations: a full-storage formulation and a low-storage $2S-1$ variant [31]. Both share the same order, linear stability polynomial, step size, and integration horizon; only the internal stage coupling differs. As illustrated in Table IX, the $2S-1$ form overwrites partial states to reduce memory, altering numerical amplification of intermediate derivatives. Despite matched validation accuracy (Tables VII and VIII), the two realizations exhibit different curvature–jerk trade-offs, showing that algebraic equivalence does not imply geometric equivalence in learned representations.

VI. Discussion

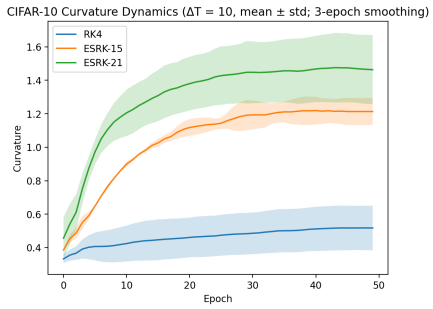
The experiments presented in this work demonstrate that the numerical realization of continuous-depth models exerts a systematic and measurable influence on learned representations. Across datasets and integration horizons, models trained with different Runge–Kutta solvers achieve nearly identical predictive accuracy, yet their latent trajectories exhibit pronounced geometric divergence. This confirms that discretization is not merely a computational convenience, but an active component of model behavior. The magnitude of the observed differences in curvature and jerk varying by factors of three to ten under



Validation accuracy



Jerk magnitude (log scale)



Curvature

Fig. 1: CIFAR-10 at $\Delta T = 10$ (mean $\pm$ std, 3 seeds). Matched accuracy, but distinct jerk and curvature reveal solver-dependent geometry.

TABLE III: CIFAR-10 validation accuracy at $\Delta T = 30$ (mean $\pm$ std over 3 seeds).

Solver	**Val. Acc. (%)**
RK4	83.25 ± 0.07
ESRK-15	84.59 ± 0.07
ESRK-21	84.8 ± 0.1

TABLE IV: CIFAR-10 results at $\Delta T = 30$ (final epoch; mean over 3 seeds).

Solver	Curvature	Jerk ($\times 10^3$)
RK4	0.55–0.60	8–10
ESRK-15	1.15–1.25	40–55
ESRK-21	1.35–1.50	25–35

TABLE V: CIFAR-100 validation accuracy at $\Delta T = 30$ (mean $\pm$ std over 3 seeds).

Solver	Val. Acc. (%)
RK4	53.02 ± 0.52
ESRK-15	$\mathbf{53.37 \pm 0.14}$
ESRK-21	51.90 ± 0.28

TABLE VI: CIFAR-100 geometric statistics at $\Delta T = 30$ (final epoch; mean over 3 seeds).

Solver	Curvature	Jerk ($\times 10^3$)
RK4	0.75–0.80	240–260
ESRK-15	0.40–0.45	6–14
ESRK-21	0.39–0.44	7–13

TABLE VII: CIFAR-10 results for the low-storage ESRK-15 ($2S - 1$) realization at $\Delta T = 30$ (mean $\pm$ std over 3 seeds).

Metric	Value
Val. Acc. (%)	84.26 ± 0.09
Jerk ($\times 10^4$)	7.46 ± 0.90
Curvature	0.62 ± 0.07

matched losses cannot be explained by stochastic optimization noise alone. Instead, these effects arise from structural differences in the way each solver discretizes and propagates the underlying continuous dynamics.

Solvers such as ESRK-15 and ESRK-21, characterized by broader stability regions and stronger internal amplification, tend to produce trajectories with higher curvature and larger jerk, while more conservative schemes such as RK4 yield smoother, lower-energy evolutions. This behavior aligns with classical numerical analysis intuition, where stage coupling and amplification factors govern smoothness and stiffness response in discrete dynamical systems.

The comparison between algebraically equivalent ESRK-15 realizations further reinforces this interpretation. Although the full-storage and $2S-1$ forms satisfy identical order conditions and share the same linear stability polynomial, their internal stage coupling differs substantially. The full form retains all intermediate stages, whereas the $2S - 1$ variant overwrites partial states to reduce memory. This difference alone is sufficient to shift curvature and jerk statistics by 50–80%, demonstrating that algebraic equivalence does not guarantee geometric equivalence in learned dynamics.

The reversal observed on CIFAR-100 at extended integration horizons suggests that solver-induced geometric effects interact with task complexity. Compared to CIFAR-10, CIFAR-100 presents a more challenging optimization landscape, with finer-grained class distinctions and noisier gradient estimates due to fewer samples per class. At longer horizons ($\Delta T = 30$), these factors may interact differently with the stability properties of the solver. Classical RK4, with a narrower stability region, may be more sensitive to stiff or high-frequency gradient components, leading to increased oscillatory behavior in latent trajectories. In contrast, extended-stability solvers are explicitly designed to integrate stiff systems and may damp such dynamics, resulting in smoother representations.

Consistent with this hypothesis, RK4 exhibits a substantial increase in jerk when moving from CIFAR-10 to CIFAR-100 at $\Delta T = 30$, while ESRK methods display the opposite trend. Although we do not directly measure the underlying optimization mechanisms, this divergence suggests that solver stability interacts non-trivially with task difficulty in shaping learned representations. A more detailed analysis of these mechanisms, such as examining gradient variance, Hessian spectra, or latent trajectory path lengths, remains an important direction for future work.

More broadly, these findings point to a phenomenon stronger than a simple solver-induced inductive bias. Each numerical solver defines a distinct discrete flow operator that approximates the same continuous system but possesses different topological and geometric properties. Small changes in the solver coefficients can act as control parameters, shifting the system between qualitatively different geometric regimes. In this sense, solver choice may induce bifurcation-like transitions in representation geometry, analogous to modifying the architecture of the dynamical system itself rather than imposing a mild optimization preference.

These observations support a broader view of continuous-depth models: numerical realization is an intrinsic part of the hypothesis class of the model. Solver selection not only affects efficiency or stability, but also shapes the effective dynamical laws governing learning and inference. Understanding and exploiting this solver-dependent structure opens the possibility of deliberately controlling geometric smoothness, robustness, and expressivity in continuous-time neural architectures.

VII. Conclusion and Future Work

This work demonstrates that the numerical realization of continuous-depth models has a systematic and measurable impact on learned representations. By comparing multiple Runge–Kutta and extended-stability solvers under accuracy-matched training conditions, we show that solver choice—and even solver realization within the same algebraic family—can induce substantially different latent geometries without affecting predictive performance. These differences arise from the internal coupling structures of the solvers, which govern how continuous dynamics are discretized during training and inference. The resulting curvature and jerk statistics reveal that the solver structure directly modulates geometric smoothness and amplification behavior in latent trajectories. These findings establish solver selection as more than a numerical hyperparameter: it represents a structural design axis that shapes the effective hypothesis class of continuous-depth networks. Through its discrete flow operator, each solver determines the geometric and topological properties of learned representa-

TABLE VIII: CIFAR-100 results for the low-storage ESRK-15 ($2S-1$) realization at $\Delta T = 30$ (mean $\pm$ std over 3 seeds).

Metric	Value
Val. Acc. (%)	$\mathbf{55.34 \pm 0.13}$
Jerk ($\times 10^4$)	2.55 ± 0.04
Curvature	0.34 ± 0.003

TABLE IX: Truncated Butcher tableaux for ESRK-15 in (a) full-storage and (b) low-storage ($2S-1$) forms. Both satisfy the same third-order conditions and stability polynomial but differ in internal stage coupling.

(a) Full-storage form

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
c_2 & a_{21} & & & & & & \\
c_3 & a_{31} & a_{32} & & & & & \\
c_4 & a_{41} & a_{42} & a_{43} & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & & \\
c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{s,s-1} & & \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & b_s
\end{array}
$$

(b) Low-storage ($2S-1$) form

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
c_2 & a_1 & & & & & & \\
c_3 & b_1 & a_2 & & & & & \\
c_4 & b_1 & b_2 & a_3 & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & & \\
c_s & b_1 & b_2 & b_3 & \cdots & a_{s-1} & & \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & b_s
\end{array}
$$

tions, potentially inducing regime shifts between smooth and oscillatory dynamics.

Future work will extend these results in several directions. First, a theoretical characterization of the discrete flow operators underlying algebraically equivalent solvers could clarify the mechanisms linking internal coupling to geometric outcomes. Second, extending the analysis to higher-order and adaptive solvers may reveal how step-size adaptation and embedded error control further influence representation geometry. Finally, applying solver-aware design principles to tasks such as robust feature learning, adversarial stability, and generative modeling may demonstrate the practical benefits of tailoring numerical realizations to desired geometric properties. In general, this study positions discretization not as an implementation detail, but as a fundamental component of model design in continuous-depth learning systems. The solver choice defines a controllable bridge between numerical approximation and representation geometry, opening a new avenue for designing differentiable systems that balance expressivity, stability, and geometric regularity.

REFERENCES

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[2] J. O'Leary, J. A. Paulson, and A. Mesbah, "Stochastic physics-informed neural ordinary differential equations," *Journal of Computational Physics*, vol. 468, p. 111466, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999122005289

[3] A. Lou, D. Lim, I. Katsman, L. Huang, Q. Jiang, S. N. Lim, and C. M. De Sa, "Neural manifold ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 548–17 558, 2020.

[4] T. Westny, A. Mohammadi, D. Jung, and E. Frisk, "Stability-informed initialization of neural ordinary differential equations," *arXiv preprint arXiv:2311.15890*, 2023.

[5] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. Oberman, "How to train your neural ODE: the world of Jacobian and kinetic regularization," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 3154–3164. [Online]. Available: https://proceedings.mlr.press/v119/finlay20a.html

[6] J. Gusak, A. Katrutsa, T. Daulbaev, A. Cichocki, and I. Oseledets, "Meta-solver for neural ordinary differential equations," *arXiv preprint arXiv:2103.08561*, 2021.

[7] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, 2017.

[8] L. Ruthotto and E. Haber, "Deep neural networks motivated by partial differential equations," *Journal of Mathematical Imaging and Vision*, vol. 62, no. 3, 2020.

[9] J. Zhuang, N. Dvornek, X. Li, S. Tatikonda, X. Papademetris, and J. Duncan, "Adaptive checkpoint adjoint method for gradient estimation in neural ode," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 639–11 649.

[10] X. Li, T.-K. L. Wong, R. T. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 3870–3882.

[11] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[12] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama, "Dissecting neural odes," *Advances in neural information processing systems*, vol. 33, pp. 3952–3963, 2020.

[13] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Advances in Neural Information Processing Systems*, 2020.

[14] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems*, 2019.

[15] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 1993.

[16] J. M. Worsham and J. K. Kalita, "A guide to neural ordinary differential equations: Machine learning for data-driven digital engineering," *Digital Engineering*, vol. 6, p. 100060, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2950550X25000263

[17] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125

[18] L. Béthune, "Deep learning with lipschitz constraints," Ph.D. dissertation, Université de Toulouse, 2024.

[19] J. Kelly, J. Bettencourt, M. J. Johnson, and D. K. Duvenaud, "Learning differential equations that are easy to solve," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4370–4380, 2020.

[20] S. O'Sullivan, "A class of high-order runge–kutta–chebyshev stability polynomials," *Journal of Computational Physics*, vol. 300, pp. 665–678, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999115005033

[21] ——, "Runge–kutta–gegenbauer explicit methods for advection-diffusion problems," *Journal of Computational Physics*, vol. 388, pp. 209–223, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999119301706

[22] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, 2016.

[23] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: an empirical study," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HJC2SzZCW

[24] L. Wu, Z. Zhu *et al.*, "Towards understanding generalization of deep learning: Perspective of loss landscapes," *arXiv preprint arXiv:1706.10239*, 2017.

[25] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[26] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina, "Entropy-sgd: Biasing gradient descent into wide valleys," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124018, 2019.

[27] Y. Heng, J. Gu, G. Xie, and J. Yi, "Deep learning-based approach for solving forward and inverse partial differential equation problems," in *High-Performance Computing and Artificial Intelligence in Process Engineering*. IOP Publishing Bristol, UK, 2025, pp. 7–1.

[28] K. Haitsiukevich and A. Ilin, "Learning trajectories of hamiltonian systems with neural networks," in *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 562–573.

[29] S. Banerjee, T. Marrinan, R. Cannon, T. Chiang, and A. D. Sarwate, "Measuring training variability from stochastic optimization using robust nonparametric testing," *IEEE Journal of Selected Topics in Signal Processing*, 2025.

[30] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Uncertainty in artificial intelligence*. PMLR, 2020, pp. 367–377.

[31] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis, "Low-storage, explicit runge–kutta schemes for the compressible navier–stokes equations," *Applied numerical mathematics*, vol. 35, no. 3, pp. 177–219, 2000.