Semi-Implicit Neural Ordinary Differential Equations for Learning Chaotic Systems

Hong Zhang
Argonne National Laboratory
hongzhang@anl.gov

Ying Liu
University of Iowa
ying-liu-1@uiowa.edu

Romit Maulik
Pennsylvania State University
rmm7011@psu.edu

Abstract

Classical neural ordinary differential equations (ODEs) trained by using explicit methods are intrinsically constrained by stability, severely affecting their efficiency and robustness in learning complex spatiotemporal dynamics, particularly those displaying chaotic behavior. In this work we propose a semi-implicit neural ODE approach that capitalizes on the partitionable structure of the underlying dynamics. In our method the neural ODE is partitioned into a linear part treated implicitly for enhanced stability and a nonlinear part treated explicitly. We apply this approach to learn chaotic trajectories of the Kuramoto–Sivashinsky equation. Our results demonstrate that our approach significantly outperforms existing approaches for coarse-resolution data and remains efficient for fine-resolution data where existing techniques become intractable.

1 Introduction

Recent advances of neural ordinary differential equations (ODEs) (Chen et al., 2018; 2020; Zhou et al., 2021; Shankar et al., 2020; Linot et al., 2022) have enabled the prediction of spatiotemporal systems from time series data, finding applications in diverse fields such as seismology, epidemiology, urban mobility, and neuroscience.

Predicting chaotic dynamics is inherently challenging, however, because of their sensitivity to initial conditions, where even a small perturbation can lead to substantial errors over time. This stability issue restricts the effectiveness of traditional neural ODE approaches to short-time prediction (Maulik et al., 2020) or prediction in reduced dimensions (Linot & Graham, 2022) for chaotic systems. The primary motivation for this research is to address the prediction of complex, high-dimensional, and chaotic systems.

In existing applications of neural ODEs, explicit time integration methods are frequently used because of their advantages in ease of implementation, minimal storage requirements, and straightforward backpropagation. However, a major drawback of these methods is their *conditional stability*, which imposes limitations on the timestep size. On the other hand, implicit time integration methods offer *unconditional stability* but often come at the cost of computational efficiency compared with explicit methods, since they require solving nonlinear systems at each time step. Typically, nonlinear solvers involve an iterative procedure such as fixed-point iteration or Newton's method. However, these solvers may fail to converge for ill-conditioned systems, and direct backpropogation becomes challenging because of the iterative algorithm's complexity and the memory bottlenecks with automatic differentiation. Some efforts have been made to make nonlinear solvers feasible for training. For instance, the Jacobian-free backpropagation method in (Fung et al., 2022) uses a preconditioning technique, and proximal implicit neural ODEs (Baker et al., 2022) approximate the solution of the nonlinear system with an optimization algorithm. Yet these methods, grounded in fixed-point iteration, compromise convergence speed and stability in favor of easing backpropagation.

Regarding gradient calculation, adjoint methods (Chen et al., 2018; Winston & Kolter, 2020) have been widely utilized in neural ODEs. The vanilla neural ODE (Chen et al., 2018) adopted a continuous adjoint approach, which, however, lacks reverse accuracy. Notably, the gradient is inconsistent with the forward pass, even when employing the same time integrator and step size in both passes. Reverse-accurate variants of neural ODEs have been developed in (Gholaminejad et al., 2019; Zhang et al., 2019; Zhuang et al., 2020; 2021), relying on backpropagation, checkpointing, and recomputation to mitigate memory costs. Zhang et al. (Zhang & Zhao, 2022) proposed a discrete adjoint approach that achieves reverse-accuracy and can leverage optimal checkpointing schedules (Zhang & Constantinescu, 2021; 2023) to balance the trade-off between recomputation and memory usage when memory is constrained.

In this work we capitalize on the structure of underlying dynamical systems and introduce a semi-implicit neural ODE approach to address the computational bottlenecks and stability limitations of traditional neural ODEs. Our approach enables the use of implicit-explicit (IMEX) methods for partitioned systems. To achieve reverse accuracy and memory efficiency, we extend the discrete adjoint approach (Zhang & Zhao, 2022) to IMEX Runge–Kutta methods.

2 Method

We consider an autonomous neural ODE framework with an additively partitioned right-hand side:

$$\frac{du}{dt} = \boldsymbol{\mathcal{G}}(u) + \boldsymbol{\mathcal{H}}(u)\,, \tag{1}$$

where $u(t) \in \mathbb{R}^d$ is the state, t is the time, and $\boldsymbol{\mathcal{G}} : \mathbb{R}^d \to \mathbb{R}^d$ and $\boldsymbol{\mathcal{H}} : \mathbb{R}^d \to \mathbb{R}^d$ are neural networks, for example feed-forward neural networks. Similar to the conventional neural ODE, our framework takes an input $u(t_0)$, learns the representation of the dynamics, and predicts the output $u(t_N)$ or a sequence of outputs $u_{t_i}, t = 1, \ldots, N$ by solving (2) from the starting time t_0 to the ending time t_N with a numerical ODE solver. While this framework is general and applicable for a wide range of applications, we are particularly interested in a model with a nonlinear part and a linear part. Without loss of generality, we assume $\boldsymbol{\mathcal{H}}$ is linear and rewrite (1) to

$$\frac{du}{dt} = \underbrace{\boldsymbol{\mathcal{G}}(u)}_{\text{Nonlinear}} + \underbrace{\boldsymbol{A}u}_{\text{Linear}}\,. \tag{2}$$

To train the partitioned neural network, we adopt a semi-implicit approach to solve (2) and a discrete adjoint approach to compute the gradients, as an alternative to directly backpropogating through an ODE solver.

2.1 Forward pass

Implicit-explicit Runge–Kutta methods are a classical class of semi-implicit numerical schemes for the time discretization of systems in the form (2), where the right-hand side is partitioned based on stiffness; for example, $\boldsymbol{\mathcal{G}}$ is nonstiff and $\boldsymbol{\mathcal{H}}$ is stiff. In an IMEX method, the ODE is integrated explicitly in $\boldsymbol{\mathcal{G}}$ and implicitly in $\boldsymbol{\mathcal{H}}$ by coefficients $(A = \{a_{ij}\}, b, c)$ for the explicit part and $(\tilde{A} = \{\tilde{a}_{ij}\}, \tilde{b}, \tilde{c})$ for the implicit part:

$$U^{(i)} = u_n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{\mathcal{G}}^{(j)} + \Delta t \sum_{j=1}^{i} \tilde{a}_{ij} \boldsymbol{\mathcal{H}}^{(j)}\,,\ i = 1, \ldots, s \tag{3a}$$

$$u_{n+1} = u_n + \Delta t \sum_{j=1}^{s} b_j \boldsymbol{\mathcal{G}}^{(j)} + \Delta t \sum_{j=1}^{s} \tilde{b}_j \boldsymbol{\mathcal{H}}^{(j)}\,, \tag{3b}$$

where Δt is the step size, $A = \{a_{ij}\}$ is strictly lower triangular, $\tilde{A} = \{\tilde{a}_{ij}\}$ is lower triangular and can have zeros on the diagonal (these correspond to explicit stages), and $\circ^{(i)}$ is a stage index i. $\boldsymbol{\mathcal{G}}^{(j)}$ and $\boldsymbol{\mathcal{H}}^{(j)}$ are shorthands for $\boldsymbol{\mathcal{G}}(t + c_i \Delta t, U^{(j)})$ and $\boldsymbol{\mathcal{H}}(t + c_i \Delta t, U^{(j)})$, respectively.

At each stage, one needs to solve a nonlinear system for $U^{(i)}$, typically with an iterative method such as the Newton method. If $\boldsymbol{\mathcal{H}}$ is linear, however, (3b) becomes a linear problem with respect to

$U^{(i)}$:

$$(\boldsymbol{\mathcal{I}} - \Delta t\, \tilde{a}_{ij} A)\, U^{(i)} = u_n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{\mathcal{G}}^{(j)} + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{ij}\, A\, U^j, \tag{4}$$

thus significantly increasing the computational efficiency. In practice, many applications contain a linear stiff term. For example, in diffusion-reaction equations, diffusion can be represented by a linear term and treated implicitly in an IMEX setting. Many neural networks, such as those containing convolution layers or fully connected layers without activation functions, can be viewed as a linear map as well.

2.2 Backward pass

Following the same methodology in (Zhang & Zhao, 2022), we have derived the discrete adjoint formula for the IMEX Runge–Kutta methods in (3b), implemented it in the PETSc library (Balay et al., 2023), and used it through the PNODE framework (Zhang & Zhao, 2022). The discrete adjoint formula is as follows:

$$\begin{aligned}
\left(I - \Delta t\, \tilde{a}_{ii}\, \boldsymbol{\mathcal{H}}_{\boldsymbol{u}}^{T(i)}\right) \boldsymbol{\lambda}_{n+1}^{(i)} &= \Delta t \left(b_i \boldsymbol{\mathcal{G}}_{\boldsymbol{u}}^{T(i)} + \tilde{b}_i \boldsymbol{\mathcal{H}}_{\boldsymbol{u}}^{T(i)}\right) \boldsymbol{\lambda}_{n+1} \\
&\quad + \Delta t\, \boldsymbol{\mathcal{G}}_{\boldsymbol{u}}^{T(i)} \sum_{j=i+1}^{s} a_{ji} \boldsymbol{\lambda}_{n+1}^{(j)} + \Delta t\, \boldsymbol{\mathcal{H}}_{\boldsymbol{u}}^{T(i)} \sum_{j=i+1}^{s} \tilde{a}_{ji} \boldsymbol{\lambda}_{n+1}^{(j)},, \quad i = s, \cdots, 1, \\
\boldsymbol{\lambda}_n &= \boldsymbol{\lambda}_{n+1} + \sum_{j=1}^{s} \boldsymbol{\lambda}_{n+1}^{(j)}, \\
\boldsymbol{\mu}_{n+1}^{(i)} &= \Delta t \left(b_i \boldsymbol{\mathcal{G}}_{\boldsymbol{p}}^{T(i)} + \tilde{b}_i \boldsymbol{\mathcal{H}}_{\boldsymbol{p}}^{T(i)}\right) \boldsymbol{\lambda}_{n+1} \\
&\quad + \Delta t\, \boldsymbol{\mathcal{G}}_{\boldsymbol{p}}^{T(i)} \sum_{j=i+1}^{s} a_{ji} \boldsymbol{\lambda}_{n+1}^{(j)} + \Delta t\, \boldsymbol{\mathcal{H}}_{\boldsymbol{p}}^{T(i)} \sum_{j=i}^{s} \tilde{a}_{ji} \boldsymbol{\lambda}_{n+1}^{(j)}, \\
\boldsymbol{\mu}_n &= \boldsymbol{\mu}_{n+1} + \sum_{j=1}^{s} \boldsymbol{\mu}_{n+1}^{(j)}.
\end{aligned} \tag{5}$$

Here $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ correspond to the partial derivatives of the loss with respect to the initial state and the NN parameters, respectively.

In the beginning of the backward pass, we set

$$\boldsymbol{\lambda}_N = \frac{d\ell}{du}, \quad \boldsymbol{\mu}_N = 0$$

as the terminal condition. $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are propagated backward in time (the index n goes from N to 0) during the backward pass according to (5). At the end of the backward pass, we obtain the gradient

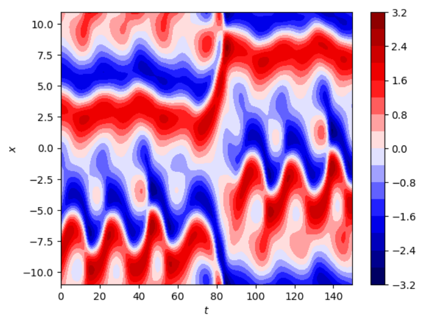
$$\nabla_\theta \ell = \boldsymbol{\mu}_0^T.$$

If $\tilde{a}_{ii} = 0$, the first equation in (5) falls back to an explicit formula. For all $\tilde{a}_{ii}$ that are nonzero, we need to solve a linear system that involves the transposed Jacobian. PETSc provides a large collection of efficient linear solvers, including iterative solvers and direct solvers. For efficiency, we do not build the Jacobian matrix explicitly when using iterative solvers. Instead, we compute the Jacobian-vector product (for the forward pass) and the transposed Jacobian-vector product (for the backward pass) with AD tools such as AutoGrad. Each time, the cost is one forward evaluation of the linear term followed by a backward pass. If $\boldsymbol{A}$ is known explicitly, direct solvers are more favorable since we can factorize the system once and reuse it across time steps and batches. $\boldsymbol{A}$ needs to be refactorized only when changed, for example at a new optimization step.

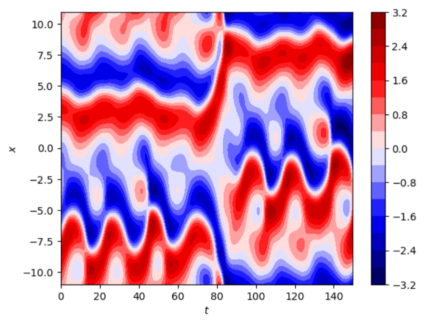
3 Experiments on Kuramoto–Sivashinsky Equation

The Kuramoto–Sivashinsky (KS) equation in one-space dimension is described by

$$\frac{\partial u}{\partial t} = -u \frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, \quad x \in [0, 22]. \tag{6}$$



(a) Ground truth



(b) Prediction

Figure 1: Prediction of the trajectory for the KS equation on a uniform grid of size 512 using SINODE with IMEX-RK2. The neural ODE RHS consists of an MLP block and a CNN block.

It is a well-known nonlinear chaotic partial differential equation (PDE) used for modeling complex spatiotemporal dynamics.

For our experiments, we use the initial condition:

$$u = cos(x/22)sin(1 + x/22).$$

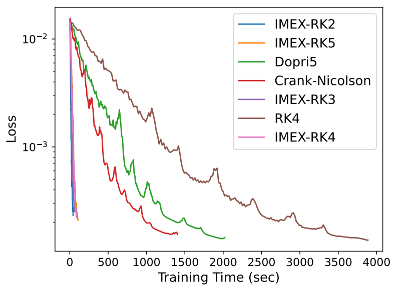
The equation is discretized in space by using a Fourier spectral method and integrated in time with the exponential time-differencing fourth-order Runge–Kutta method (Kassam & Trefethen, 2005). To obtain the training data, we collect a time series $\boldsymbol{u}(t) \in \mathbb{R}^N$ for a time span of 200 after discarding the transient states for the initial time period $[0, 100]$ so that the dynamics is guaranteed to be in the chaotic regime. In this regime, the solution is extremely sensitive to perturbations in the initial data; for example, perturbations can be amplified by 10^8 in a time span of 150 (Kassam & Trefethen, 2005).

We train neural ODEs by approximating the nonlinear term $u\frac{\partial u}{\partial x}$ with a fully connected multilayer perceptron (MLP) and approximating the remaining linear terms on the right-hand side (RHS) of (6) with a fixed convolutional neural network (CNN) layer, following the best settings identified in (Linot et al., 2022). We use circular padding and a CNN filter of size 5 to mimic centered finite-difference operators for the anti-diffusion and hyper-diffusion terms with periodic boundary conditions. The parameters of the filter are fixed to be $[-1.0/\Delta x^4, 4.0/\Delta x^4 - 1.0/\Delta x^2, -6.0/\Delta x^4 + 2.0/\Delta x^2, 4.0/\Delta x^4 - 1.0/\Delta x^2, -1.0/\Delta x^4]$, where the grid spacing $\Delta x = L/N$.

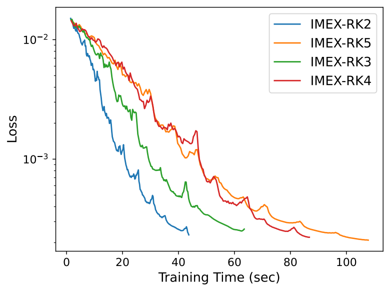
We compare SINODE with explicit methods that are commonly used for neural ODEs and with a fully implicit method proposed in (Zhang & Zhao, 2022). When using the SINODE method, we treat the MLP part implicitly and the CNN part explicitly. A timestep size 0.2 is used for IMEX and the fully implicit method. Because of the stability constraints, the explicit methods do not work well until we decrease the step size to 0.001.

Figure 1 demonstrates that the model prediction made by SINODE is in good agreement with the ground truth for a long time period. In Figure 2 we illustrate the training loss versus the training time for all the methods, providing a clear evaluation of their efficiency. As expected, SINODE using IMEX methods significantly outperform the other methods. For example, SINODE with IMEX-RK2 is approximately 47 times faster than the classical neural ODE with Dopri5 and approximately 33 times faster than the neural ODE with Crank–Nicolson to decrease the training loss to 10^{-3}. The superior performance of SINODE is attributed mainly to superior stability properties and the computational advantages of semi-implicit methods. Note that the improvement in training efficiency becomes more significant as the number of time steps increases because of the increased opportunity to reuse the LU factorization for solving the linear systems. For Figure 2 we use only one time step during the training, reflecting the worst-case performance.

The problem becomes more challenging for training neural ODEs when the grid resolution increases (so the dimension N increases). After we change the grid size from 64 to 512, the performance of neural ODEs with explicit methods or fully implicit methods degrades dramatically, while SINODE maintains high efficiency. Figure 3 shows the results of SINODE for this test case. We can still use the timestep size 0.2 for the training with SINODE, but the explicit methods require a step size as



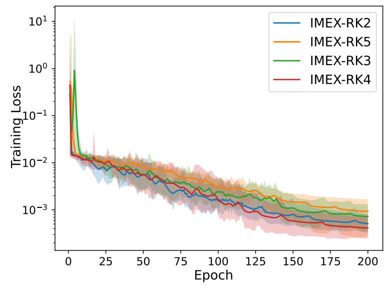
(a) All methods



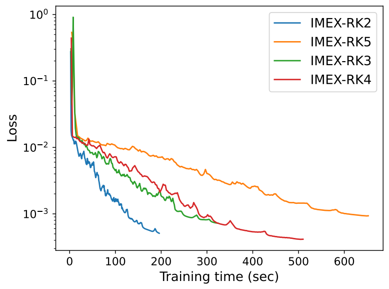
(b) IMEX methods

Figure 2: Training loss versus training time for all the methods (a) and for the IMEX methods (b). The grid size is 64. (b) is a zoom-in plot of (a).

small as 10^{-7} because of the severe stability constraint from the hyper-diffusion term. We are not able to make the fully implicit methods work with the step size 0.2 because the nonlinear solver diverges frequently during the training process. This is not uncommon for classical PDE solvers since the condition number of the system increases as the grid resolution increases. Decreasing the step size by 10 times stabilizes the nonlinear solvers. However, this leads to a per-epoch training time of more than one hour whereas SINODE typically takes less than one second per epoch.



(a) Training loss



(b) Training time vs. loss

Figure 3: SINODE results for KSE on a uniform grid of size 512.

An interesting observation is that the lower-order IMEX-RK methods lead to better training efficiency than do high-order IMEX-RK methods, despite their slightly slower convergence speed due to relatively lower accuracy. This is because higher-order IMEX-RK methods require more stages, leading to more linear solves at each time step. For the coarse-grid case (64), the second-order IMEX-RK performs approximately 2.5 times better than the fifth-order IMEX-RK. For the fine-grid case (512), this gain factor is larger than 5. We expect this gain to increase as the grid resolution increases.

4 Conclusion

In this study we introduce SINODE, a semi-implicit approach for training neural ODEs by harnessing IMEX Runge–Kutta methods and their discrete adjoints. SINODE offers substantial computational advantages over existing approaches, owing to its enhanced stability and efficient linear system solution during time integration. We demonstrated the efficiency of SINODE by applying it to learn the chaotic dynamics of a PDE problem. For coarse-resolution data, SINODE outperforms the classical neural ODE approach by a factor of 47X. With fine -resolution data, we illustrate that SINODE maintains high efficiency, while both explicit methods and fully implicit methods become impractical because of the stability constraints or computational expenses.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under contract DE-AC02-06CH11357 at Argonne National Laboratory. We also acknowledge funding support from ASCR for DOE-FOA-2493, "Data-intensive scientific machine learning."

References

Baker, J., Xia, H., Wang, Y., Cherkaev, E., Narayan, A., Chen, L., Xin, J., Bertozzi, A. L., Osher, S. J., and Wang, B. Proximal implicit ODE solvers for accelerating learning neural ODEs, 2022.

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., Zhang, H., and Zhang, J. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.19, Argonne National Laboratory, 2023.

Chen, R. T., Amos, B., and Nickel, M. Neural spatio-temporal point processes. In *International Conference on Learning Representations*, 2020.

Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D., Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. 2018.

Fung, S. W., Heaton, H., Li, Q., McKenzie, D., Osher, S., and Yin, W. JFB: Jacobian-free backpropagation for implicit networks. volume 36, 2022. doi: 10.1609/aaai.v36i6.20619.

Gholaminejad, A., Keutzer, K., and Biros, G. ANODE: Unconditionally accurate memory-efficient gradients for neural ODEs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 730–736. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/103.

Kassam, A.-K. and Trefethen, L. N. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005. doi: 10.1137/S1064827502410633.

Linot, A. J. and Graham, M. D. Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(7):073110, 07 2022. ISSN 1054-1500. doi: 10.1063/5.0069536.

Linot, A. J., Burby, J. W., Tang, Q., Balaprakash, P., Graham, M. D., and Maulik, R. Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems. *Journal of Computational Physics*, 474, 3 2022. doi: 10.1016/j.jcp.2022.111838.

Maulik, R., Mohan, A., Lusch, B., Madireddy, S., Balaprakash, P., and Livescu, D. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 2020. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2020.132368.

Shankar, V., Portwood, G., Mohan, A., Mitra, P., Rackauckas, C., Wilson, L., Schmidt, D., and Viswanathan, V. Learning non-linear spatio-temporal dynamics with convolutional Neural ODEs. In *Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020)*, 2020.

Winston, E. and Kolter, J. Z. Monotone operator equilibrium networks. volume 2020-December, 2020.

Zhang, H. and Constantinescu, E. M. Revolve-based adjoint checkpointing for multistage time integration. In *International Conference on Computational Science*, (online), in main track, 2021.

Zhang, H. and Constantinescu, E. M. Optimal checkpointing for adjoint multistage time-stepping schemes. *Journal of Computational Science*, 66:101913, 2023. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101913.

Zhang, H. and Zhao, W. A memory-efficient neural ordinary differential equation framework based on high-level adjoint differentiation. *IEEE Transactions on Artificial Intelligence*, pp. 1–11, 2022. doi: 10.1109/TAI.2022.3230632.

Zhang, T., Yao, Z., Gholami, A., Gonzalez, J. E., Keutzer, K., Mahoney, M. W., and Biros, G. ANODEV2: A coupled neural ODE framework. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Zhou, F., Li, L., Zhang, K., and Trajcevski, G. Urban flow prediction with spatial–temporal neural ODEs. *Transportation Research Part C: Emerging Technologies*, 124:102912, 2021. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2020.102912.

Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., and Duncan, J. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 11639–11649, 2020.

Zhuang, J., Dvornek, N. C., Tatikond, S., and Duncan, J. S. MALI: A memory efficient and reverse accurate integrator for neural ODEs. In *International Conference on Learning Representations*, 2021.