Algebraically reversible solvers for neural differential equations

James Foster

University of Bath

Ongoing with Samuel McCallum (Bath)

Outline

1. Neural Ordinary Differential Equations
2. Algebraically reversible ODE solvers
3. Towards more general reversible solvers
4. Conclusion and future work
5. References

What is a neural differential equation?

These are differential equations where the vector field is parametrised as a neural network.

Standard example: Neural ODEs [1], due to Chen et al. (NeurIPS 2018).

$$\frac{dy}{dt} = f_\theta(t, y(t)),$$
$$y(0) = y_0,$$

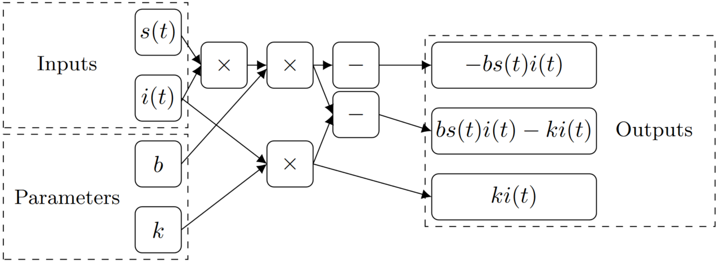
where f_θ can be any neural network (feedforward, convolutional, etc).

Examples of neural ordinary differential equations

A simple example: The SIR model for modelling infectious diseases

$$\frac{d}{dt}\begin{pmatrix} s(t) \\ i(t) \\ r(t) \end{pmatrix} = \begin{pmatrix} -bs(t)i(t) \\ bs(t)i(t) - ki(t) \\ ki(t) \end{pmatrix},$$

where b and k are parameters that are learnt from data.



At the other extreme, Neural ODEs have achieved 70% accuracy for ImageNet classification [2] (outperforming a well-tuned ResNet).

How to train your Neural ODE (backpropagation)

Step 1. Define a differentiable scalar loss function based on the data

$$L\big(y(t)\big) = L\Big(ODESolve\big(y(0), t, f_\theta\big)\Big).$$

Step 2. As "*ODESolve*" is a composition of differentiatiable operations, we can compute $\frac{dL}{d\theta}$ using automatic differentiation / backpropagation.
Step 3. Apply stochastic gradient descent (SGD) with $\frac{dL}{d\theta}$ to minimize L.

However...

When applying backpropagation, we store the full ODE trajectory $\{y_{t_k}\}$.

Thus, the memory cost scales linearly with the number of steps / depth.

How to train your Neural ODE (adjoint method)

Step 1. Define a differentiable scalar loss function based on the data

$$L\big(y(t)\big) = L\Big(ODESolve\big(y(0), t, f_\theta\big)\Big).$$

Step 2. Compute $L\big(y(T)\big)$ via ODE solver. Then $a(t) := \frac{\partial L(y(t))}{\partial y(t)}$ satisfies

$$\frac{da(t)}{dt} = -a(t)^\top \frac{\partial f_\theta\big(t, y(t)\big)}{\partial y}.$$

Step 3. Solve the above adjoint equation via ODE solver, and evaluate

$$\frac{dL}{d\theta} = \int_0^T a(t)^\top \frac{\partial f_\theta\big(t, y(t)\big)}{\partial \theta}\, dt.$$

Step 4. Apply stochastic gradient descent (SGD) with $\frac{dL}{d\theta}$ to minimize L.

Reconstruction and extrapolation of spirals with irregular time points (taken from [1])

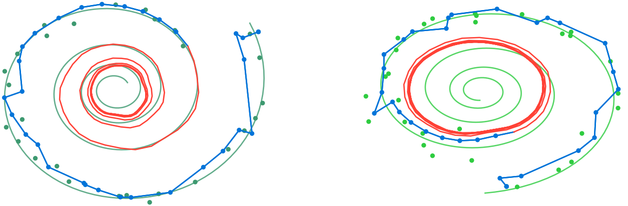
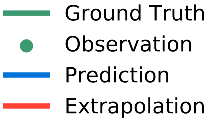


Figure: Recurrent Neural Network



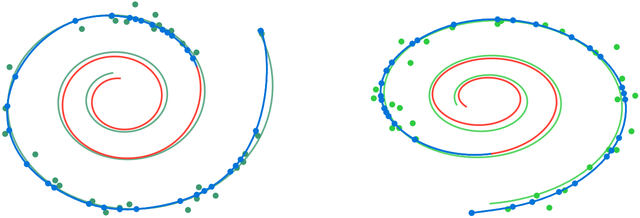


Figure: Neural ODE

Why Neural ODEs and the adjoint method?

- Continuous time, so well suited for handling (irregular) time series
- Flexible, includes “mechanistic” and “deep” models (+ hybrids [3])
- Choice of ODE solver allows trade-offs between accuracy and cost
- Adjoint method is memory efficient! (i.e. doesn’t scale with depth)

However...

Solving the ODE and adjoint equation can give inexact gradients.

Outline

1 Neural Ordinary Differential Equations

2 Algebraically reversible ODE solvers

3 Towards more general reversible solvers

4 Conclusion and future work

5 References

Accurate memory-efficient gradients for Neural ODEs

To get accurate gradients (e.g. by backpropagation or adjoint method), we would need to reconstruct the ODE solution in the backwards pass.

In [2], it was shown this can be achieved using a reversible ODE solver.

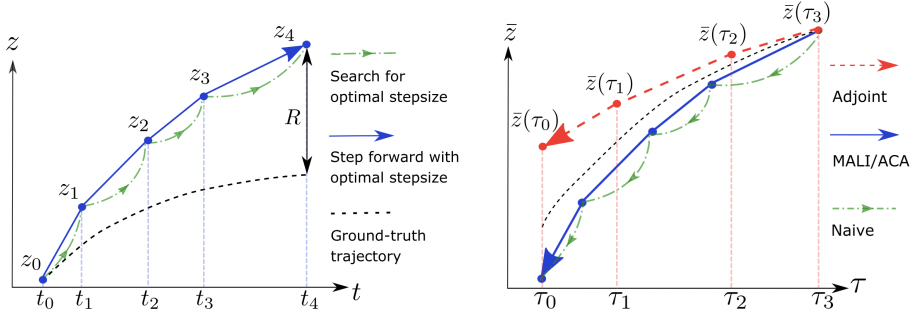


Figure: Illustration of a reversible ODE solver called "ALF" (taken from [2])

Accurate memory-efficient gradients for Neural ODEs

Definition (ODE solver with order of convergence α)

We say an ODE solver $\Phi : \mathbb{R} \times \mathbb{R}^d \mapsto \mathbb{R}$ converges with order $\alpha > 0$ if

$$\|x(h) - \Phi_h(x)\| \leq C|h|^{\alpha+1},$$

where $x(h)$ is the solution at time $|h|$ of an ODE started at $x(0) := x$,

$$x' = f(x) \quad \text{if} \quad h \geq 0, \qquad \text{or} \qquad x' = -f(x) \quad \text{if} \quad h < 0.$$

Definition (Symmetric reversibility)

We say an ODE solver Φ is symmetric reversible if $\Phi_{-h}(\Phi_h(x)) = x$.

Example

For a general $f : \mathbb{R}^d \to \mathbb{R}^d$, Euler's method is not symmetric reversible.

$$(x + f_\theta(x)h) - f_\theta(x + f_\theta(x)h)h \neq x$$

Examples of reversible solvers

Example (Asynchronous Leapfrog Integrator (ICLR 2021))

$$\begin{aligned} X_{n+\frac{1}{2}} &:= X_n + \frac{1}{2}V_n h, \\ V_{n+1} &:= 2f\big(X_{n+\frac{1}{2}}\big) - V_n\,, \\ X_{n+1} &:= X_n + f\big(X_{n+\frac{1}{2}}\big)h, \end{aligned}$$

where $X_0 := x(0)$ and $V_0 := f(X_0)$.

Remark (Algebraic reversibility)

$$\begin{aligned} X_{n+\frac{1}{2}} &= X_{n+1} - \frac{1}{2}V_{n+1} h, \\ V_n &= 2f\big(X_{n+\frac{1}{2}}\big) - V_{n+1}\,, \\ X_n &= X_{n+1} - f\big(X_{n+\frac{1}{2}}\big)h. \end{aligned}$$

Examples of reversible solvers

Example (Reversible Heun's method (NeurIPS 2021))

$$Y_{n+1} := 2X_n - Y_n + f(Y_n)h,$$
$$X_{n+1} := X_n + \frac{1}{2}\big(f(Y_n) + f(Y_{n+1})\big)h,$$

where $X_0 = Y_0 = x(0)$.

Remark (Algebraic reversibility)

$$Y_n = 2X_{n+1} - Y_{n+1} - f(Y_{n+1})h,$$
$$X_n = X_{n+1} - \frac{1}{2}\big(f(Y_{n+1}) + f(Y_n)\big)h.$$

Examples of reversible solvers

Both methods...

- achieve reversibility by introducing extra state.
- have second order convergence with fixed steps.
- have a potentially unstable step of the form $2A - B$.
- have worked in large-scale applications:
  - A Neural ODE with the asynchronous leapfrog integrator achieved better performance than a ResNet-18 (≈ 11.7 million parameters) for classification on the ImageNet dataset [2].
  - A Neural SDE with the reversible Heun scheme was successfully used to model turbulence (≈ 4.6 million parameters) [4].
- can be defined for both ODEs and SDEs. However, in the SDE case, we could only prove convergence for the Reversible Heun scheme.

Examples of reversible solvers

Modelling turbulence is computationally demanding due to the fine mesh and steps used to approximate the PDE. A transformer-based Neural SDE model was recently developed for such simulations [4], and was numerically discretized using the Reversible Heun method.

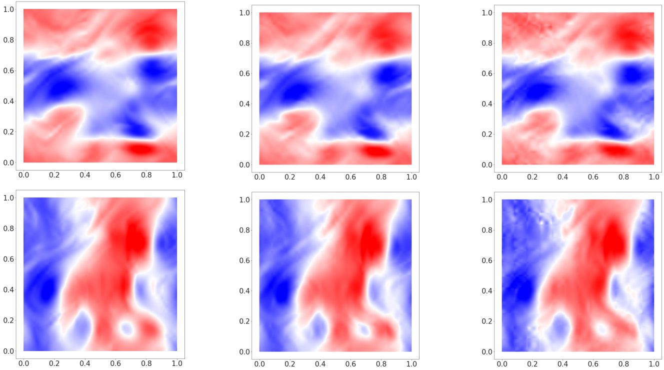


Figure: PDE simulation (left), Neural SDE (middle) and Neural network (right)

Examples of reversible solvers

However [5] and [6] report that the reversible Heun method was too unstable for their applications.

Asynchronous Leapfrog Integrator	Reversible Heun method
$X_{n+\frac{1}{2}} := X_n + \frac{1}{2}V_n h,$ $V_{n+1} := 2f(X_{n+\frac{1}{2}}) - V_n,$ $X_{n+1} := X_n + f(X_{n+\frac{1}{2}})h,$	$Y_{n+1} := 2X_n - Y_n + f(Y_n)h,$ $X_{n+1} := X_n + \frac{1}{2}\big(f(Y_n) + f(Y_{n+1})\big)h$

We believe that any instability is then amplified by these solvers when

- V_n and $f(X_n)$ drift apart (for ALF)
- X_n and Y_n drift apart (for RH)

Outline

1. Neural Ordinary Differential Equations
2. Algebraically reversible ODE solvers
3. Towards more general reversible solvers
4. Conclusion and future work
5. References

Towards more general algebraically reversible solvers

Given an ODE solver Φ_h, we define the map $\Psi_h(x) := \Phi_h(x) - x$ so that

$$\|x(h) - (x + \Psi_h(x))\| \leq C|h|^{\alpha+1}, \tag{1}$$

where $x(h)$ is the solution at time h of the ODE started at $x(0) := x$.

Definition (A "forward-backward" reversible solver for ODEs)

We construct a numerical solution $\{(Y_n, Z_n)\}_{n\geq 0}$ by $Y_0 = Z_0 = x(0)$ and

$$Y_{n+1} := \lambda Y_n + (1-\lambda)Z_n + \Psi_h(Z_n),$$
$$Z_{n+1} := Z_n - \Psi_{-h}(Y_{n+1}),$$

where $h > 0$ is the step size and $\lambda \in (0,1]$ is a "coupling" parameter.

Towards more general algebraically reversible solvers

The new solver is

$$Y_{n+1} := \lambda Y_n + (1-\lambda)Z_n + \Psi_h(Z_n),$$
$$Z_{n+1} := Z_n - \Psi_{-h}(Y_{n+1}),$$

The first property to note is that this is algebraically reversible since

$$Z_n := Z_{n+1} + \Psi_{-h}(Y_{n+1}),$$
$$Y_n := \lambda^{-1}Y_{n+1} + (1-\lambda^{-1})Z_n - \lambda^{-1}\Psi_h(Z_n).$$

Secondly, we introduce $\lambda \in (0,1]$ so that Y_n and Z_n stay close together,

$$Y_{n+1} - Z_{n+1} = \lambda(Y_n - Z_n) + \underbrace{\Psi_h(Z_n) + \Psi_{-h}(Y_{n+1})}_{\text{small if } Z_n \approx x(t_n) \text{ and } Y_{n+1} \approx x(t_{n+1})}.$$

But if λ is too small, it might cause instabilities on the backwards pass.

Overview of error analysis

Currently, our analysis requires the map Ψ_h to be Lipschitz continuous.

More specifically, we assume there exists $\|\Psi\| > 0$ and $h_{\max} > 0$ so that

$$\big\|\Psi_h(x) - \Psi_h(y)\big\| \leq \|\Psi\||h|\|x - y\|, \tag{2}$$

for $x \in \mathbb{R}^d$ and $h \in [-h_{\max}, h_{\max}]$.

From (2) along with our assumption that Ψ is an order α solver, we have

$$\big\|\Psi_h(x + \Psi_{-h}(x)) + \Psi_{-h}(x)\big\| \leq \widetilde{C}|h|^{\alpha+1}. \tag{3}$$

In other words, going forwards and backwards with Ψ gives little error.

Overview of error analysis

Suppose we discretise the ODE over the time horizon $[0, T]$ with N steps (that is, we use a constant step size of $h := \frac{T}{N}$). Then $\|Y - Z\|$ is small as

$$
\begin{aligned}
&\|Y_{n+1} - Z_{n+1}\| \\
&\quad = \|\lambda(Y_n - Z_n) + \Psi_h(Z_n) + \Psi_{-h}(Y_{n+1})\| \\
&\quad \leq |\lambda|\|Y_n - Z_n\| + \|\Psi_h(Z_n) + \Psi_{-h}(Z_{n+1})\| + \|\Psi_{-h}(Y_{n+1}) - \Psi_{-h}(Z_{n+1})\| \\
&\quad \leq |\lambda|\|Y_n - Z_n\| + \|\Psi_h(Z_{n+1} + \Psi_{-h}(Y_{n+1})) + \Psi_{-h}(Z_{n+1})\| \\
&\qquad\qquad + \|\Psi\||h|\|Y_{n+1} - Z_{n+1}\| \\
&\quad \leq |\lambda|\|Y_n - Z_n\| + \underbrace{\|\Psi_h(Z_{n+1} + \Psi_{-h}(Z_{n+1})) + \Psi_{-h}(Z_{n+1})\|}_{\leq \widetilde{C}|h|^{\alpha+1}} \\
&\qquad\qquad + \underbrace{\|\Psi_h(Z_{n+1} + \Psi_{-h}(Y_{n+1})) - \Psi_h(Z_{n+1} + \Psi_{-h}(Z_{n+1}))\|}_{\leq \|\Psi\|^2 h^2 \|Y_{n+1} - Z_{n+1}\|} \\
&\qquad\qquad + \|\Psi\||h|\|Y_{n+1} - Z_{n+1}\|.
\end{aligned}
$$

Overview of error analysis

After showing that Y and Z are close together, we consider the quantity

$$X_n := \lambda^{N-n} Y_n + (1 - \lambda^{N-n}) Z_n \, .$$

This leads to the following error estimate,

$$\begin{aligned}
&\|X_{n+1} - x(t_{n+1})\| \\
&\quad = \left\| \lambda^{N-(n+1)} Y_{n+1} + (1 - \lambda^{N-(n+1)}) Z_{n+1} - x(t_{n+1}) \right\| \\
&\quad = \left\| X_n + \lambda^{N-(n+1)} \Psi_h(Z_n) - (1 - \lambda^{N-(n+1)}) \Psi_{-h}(Y_{n+1}) - x(t_{n+1}) \right\| \\
&\quad \leq \|X_n - x(t_n)\| + \lambda^{N-(n+1)} \underbrace{\left\| \Psi_h(Z_n) + \Psi_{-h}(Y_{n+1}) \right\|}_{=:A} \\
&\qquad\quad + \underbrace{\left\| x(t_n) - \big(x(t_{n+1}) + \Psi_{-h}(Y_{n+1}) \big) \right\|}_{=:B},
\end{aligned}$$

where A and B can be estimated using similar techniques as before.

Overview of error analysis

Theorem (Main result; any ODE solver can made reversible)

Suppose Ψ corresponds to an α-order numerical method for the ODE

$$x' = f(x),$$

where $t \in [0, T]$ for a fixed T. Then under the Lipschitz assumption (2), there exists constants $C, h_{\max} > 0$ such that

$$\big\| Y_k - x(t_k) \big\| \leq C h^{\alpha},$$

for all $k \in \{0, 1, \cdots, N\}$ where $h \in (0, h_{\max}]$, $t_k := kh \in [0, T]$ and

$$Y_{n+1} := \lambda Y_n + (1 - \lambda) Z_n + \Psi_h(Z_n),$$

$$Z_{n+1} := Z_n - \Psi_{-h}(Y_{n+1}),$$

with $\lambda \in (0, 1]$ and $Y_0 = Z_0 = x(0)$.

Stability of reversible ODE solvers

Although we can construct arbitrarily high order ODE reversible solvers, we have not yet addressed the main challenges which concern stability.

Definition (A-stability region)

Consider the following linear ODE,

$$\begin{aligned} y' &= \alpha y, \\ y(0) &= 1, \end{aligned} \tag{4}$$

where $\alpha \in \mathbb{C}$ with $\mathrm{Re}(\alpha) < 0$. A numerical solution $Y = \{Y_k\}_{k\geq 0}$ of (4) is said to be A-stable at α if $Y_k \to 0$ as $k \to \infty$. The stability region is

$$R = \{\alpha \in \mathbb{C} \,:\, \mathrm{Re}(\alpha) < 0 \text{ and } Y = \{Y_k\} \text{ is A-stable at } \alpha\}.$$

The Asynchronous Leapfrog Integrator and Reversible Heun method are not A-stable (for any $\alpha \in \mathbb{C}$).

Stability of reversible ODE solvers

Suppose $\Psi_h(x) = \alpha x h$. Then each step of the reversible ODE solver is

$$Y_{n+1} := \lambda Y_n + (1-\lambda) Z_n + \alpha Z_n h,$$
$$Z_{n+1} := Z_n + \alpha Y_{n+1} h,$$

which can be expressed as

$$\begin{pmatrix} Y_{n+1} \\ Z_{n+1} \end{pmatrix} = A \begin{pmatrix} Y_n \\ Z_n \end{pmatrix},$$

where

$$A := \begin{pmatrix} \lambda & 1 - \lambda + \alpha h \\ \alpha \lambda h & 1 + \alpha(1-\lambda)h + \alpha^2 h^2 \end{pmatrix}.$$

Stability of reversible ODE solvers

Since $\mathrm{tr}A$ and $\det A$ are the sum and product of the eigenvalues $\{\eta_\pm\}$, we compute

$$\begin{aligned}\det A &= \lambda(1+\alpha(1-\lambda)h+\alpha^2h^2) - (1-\lambda+\alpha h)\alpha\lambda h = \lambda,\\ \mathrm{tr}A &= 1+\lambda+\alpha(1-\lambda)h+\alpha^2h^2.\end{aligned}$$

which gives the eigenvalues,

$$\begin{aligned}\eta_\pm = \frac{1}{2}\big(1+\lambda+\alpha(1-\lambda)h+\alpha^2h^2\big)\\ \pm\frac{1}{2}\sqrt{(1-\lambda)^2+(1+\lambda)\big(\alpha(1-\lambda)h+\alpha^2h^2\big)+\big(\alpha(1-\lambda)h+\alpha^2h^2\big)^2},\end{aligned}$$

however we do not yet have an explicit formula for the stability region (i.e. the values of $\alpha\in\mathbb{C}$ such that $|\eta_+|\vee|\eta_-|<1$).

Stability of reversible ODE solvers

Thus, we have stability regions – unlike previous reversible schemes!

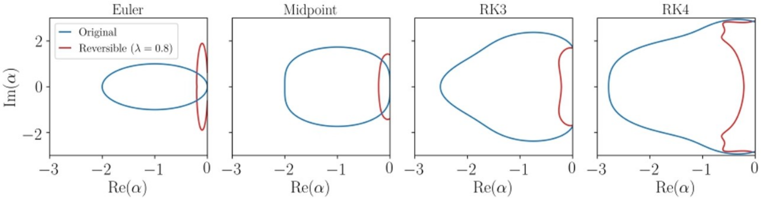


Figure: Stability regions for different reversible schemes ($h = 1$ and $\lambda = 0.8$).

We conjecture that decreasing $\lambda \in (0, 1]$ improves the stability region.

However, if λ is too small, then the backwards solve may be unstable.

A potentially more stable reversible solver

Based on the $\Psi_h(x) = f(x)h$ case, we can instead consider the scheme:

$$\begin{aligned} Y_{n+\frac{1}{2}} &:= \lambda Y_n + (1-\lambda)Z_n + \frac{1}{2}f(Z_n)h, \\ Z_{n+1} &:= Z_n + f(Y_{n+\frac{1}{2}})h, \\ Y_{n+1} &:= Y_{n+\frac{1}{2}} + \frac{1}{2}f(Z_{n+1})h. \end{aligned} \tag{5}$$

which uses two extra function evaluations per step and is reversible as

$$\begin{aligned} Y_{n+\frac{1}{2}} &= Y_{n+1} - \frac{1}{2}f(Z_{n+1})h, \\ Z_n &= Z_{n+1} - f(Y_{n+\frac{1}{2}})h, \\ Y_n &= \lambda^{-1}Y_{n+\frac{1}{2}} + (1-\lambda^{-1})Z_n - \frac{1}{2}\lambda^{-1}f(Z_n)h. \end{aligned}$$

However, the associated eigenvalues will become more complicated...

Outline

1. Neural Ordinary Differential Equations
2. Algebraically reversible ODE solvers
3. Towards more general reversible solvers
4. Conclusion and future work
5. References

Conclusion

- Among the recent advances in neural differential equations, reversible solvers have seen utility due to the accurate and memory-efficient gradients that they provide during training.

- However, the current reversible NDE solvers have stability issues. We believe that this instability is amplified by the "$2A - B$" terms.

- We propose a "forward-backward" approach in which any ODE solver can be converted to a reversible one with the same order (but at the cost of using twice the function evaluations per step).

- This leads to a second order reversible ODE solver (5), which we expect has a non-empty stability region (unlike previous solvers).

Future work

- Error analysis, stability analysis and numerical examples for (5).
- In practice, what are good values for the coupling parameter λ?
- Runge-Kutta methods for ODEs are defined by Butcher Tableaus. For the coefficients in these tableaus, there are order conditions.

  Can we derive such tools to facilitate the use of reversible solvers?
- Similarly, could we derive a Butcher group [7] of reversible solvers?
- Extension to Neural CDEs [8] or RDEs [9] via log-ODE method [10]?

Thank you
for your attention!

Outline

1. Neural Ordinary Differential Equations
2. Algebraically reversible ODE solvers
3. Towards more general reversible solvers
4. Conclusion and future work
5. References

References I

R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud. *Neural Ordinary Differential Equations*, Neural Information Processing Systems, 2018.

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

C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan and A. Edelman. *Universal Differential Equations for Scientific Machine Learning*, arXiv:2001.04385, 2020.

References II

- A. Boral, Z. Yi Wan, L. Zepeda-Núñez, J. Lottes, Q. Wang, Y. Chen, J. R. Anderson and F. Sha. *Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations*, Neural Information Processing Systems, 2023.
- Q. Zhang and Y. Chen. *Path Integral Sampler: A Stochastic Control Approach For Sampling*, International Conference on Learning Representations, 2022.
- A. Howe. *Possible issue with ReversibleHeun solver instability*, https://github.com/patrick-kidger/diffrax/issues/417, 2024.
- J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*, Third Edition, John Wiley & Sons, 2016.

References III

- P. Kidger, J. Morrill, J. Foster and T. Lyons. *Neural Controlled Differential Equations for Irregular Time Series*, Neural Information Processing Systems, 2020.

- J. Morrill, C. Salvi, P. Kidger, J. Foster and T. Lyons. *Neural Rough Differential Equations for Long Time Series*, International Conference on Machine Learning, 2021.

- B. Walker, A. D. McLeod, T. Qin, Y. Cheng, H. Li and T. Lyons. *Log Neural Controlled Differential Equations: The Lie Brackets Make a Difference*, International Conference on Machine Learning, 2024.