Efficient, Accurate and Stable Gradients for Neural Differential Equations

James Foster

University of Bath

Joint with Sam McCallum (Bath)

Outline

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).

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

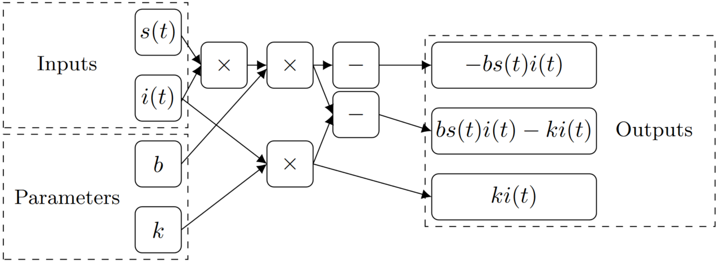
where f_θ can be any neural network (feedforward, convolutional, etc).

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] (competitive with a well-tuned ResNet).

How to train your Neural ODE (backpropagation)

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

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

Step 2. As "*ODESolve*" is a composition of differentiable 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])

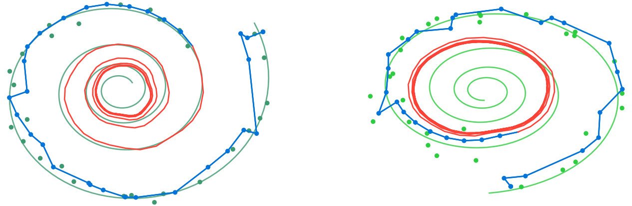
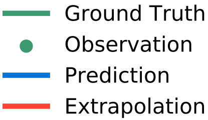


Figure: Recurrent Neural Network



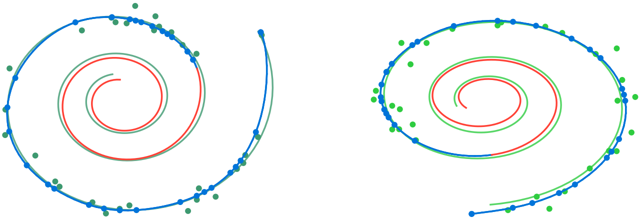


Figure: Neural ODE

Why Neural ODEs and the adjoint method?

- Flexible, includes "mechanistic" and "deep" models (+ hybrids [3])
- Continuous time, so well suited for handling (irregular) time series
- 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 its adjoint equation gives <u>inexact</u> gradients.

Outline

Reversible ODE solvers

We can compute gradients accurately using backpropagation – but that requires us to have the numerical ODE solution for the backwards pass.

In [2], it was shown that the numerical ODE solution can be dynamically recomputed (i.e. constant memory cost) using a reversible ODE solver.

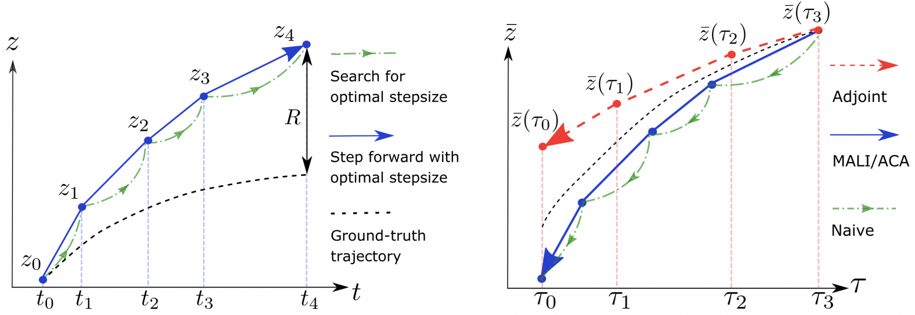


Figure: Illustration of a reversible ODE solver called "MALI" (taken from [2])

Reversible ODE solvers

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 $x(0) := x$,

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

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))

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

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

Remark (Symmetric reversibility)

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

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 (Symmetric 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 step sizes.
- have a potentially unstable term of the form $2A - B$.
- have worked in large-scale applications:
  - A Neural ODE with the asynchronous leapfrog integrator achieved comparable performance to a ResNet-18 (≈ 11 million parameters) for classification on the ImageNet dataset [2].
  - A Neural SDE with the reversible Heun scheme was successfully used for turbulence modelling (≈ 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

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

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

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

Towards more stable reversible solvers

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

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

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

Definition (Proposed reversible ODE solver [7])

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 stable reversible solvers

This approach is based on two ideas:

- Extra state allows for a reversible computation graph.
  (e.g. previous reversible solvers and coupling layers in neural nets)

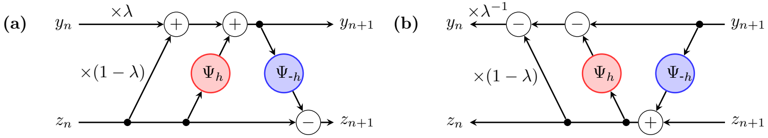


Figure: (a) Forwards ODE solve. (b) Backward ODE solve.

- ODE solvers can be applied with positive and negative step sizes.

$$
\begin{aligned}
x(h) \approx \Phi_h(x(0)) \;\; \text{“}\!\Rightarrow\!\text{”} \;\; & x(0) \approx \Phi_{-h}(x(h)) \\
\text{“}\!\Rightarrow\!\text{”} \;\; & x(0) \approx x(h) + \Psi_{-h}(x(h)) \\
\text{“}\!\Rightarrow\!\text{”} \;\; & x(h) \approx x(0) - \Psi_{-h}(x(0) + \Psi_h(x(0))).
\end{aligned}
$$

Towards more stable reversible solvers

Recall 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 key 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 may cause instabilities on the backwards solve.

Towards more stable reversible solvers

Theorem (Main result; any ODE solver can made reversible [7])

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 a Lipschitz assumption on Ψ, there exists constants $C, h_{\max} > 0$ such that

$$\big\| Y_k - x(t_k) \big\| \leq Ch^{\alpha}, \tag{1}$$

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 challenge which concerns stability.

Definition (A-stability region)

Consider the following linear ODE,

$$\begin{aligned} y' &= \alpha y, \\ y(0) &= 1, \end{aligned} \tag{2}$$

where $\alpha \in \mathbb{C}$ with $\mathrm{Re}(\alpha) < 0$. A numerical solution $Y = \{Y_k\}_{k\geq 0}$ of (2) 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

Numerically computing stability regions gives some promising results:

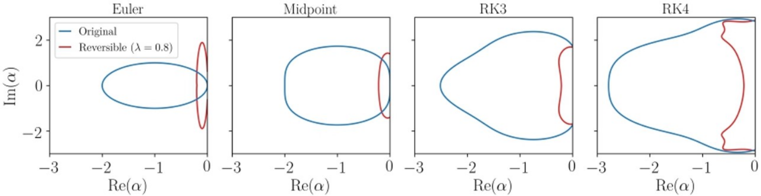


Figure: Stability regions for different reversible schemes ($h = 1$ and $\lambda = 0.8$).

We also see that decreasing $\lambda \in (0, 1]$ increases the stability region. However, if λ is too small, then the backwards solve may be unstable.

Theoretically, we have only been able to find a closed-form expression for the real part of these stability regions [7].

Outline

Experiments

We first generate synthetic time series data $\{x(t_i)\}_{i\geq 0}$ by simulating Chandrasekhar's white dwarf equation,

$$\frac{dx}{dt} = v,$$
$$\frac{dv}{dt} = -\frac{2}{t}v - \left(x^2 - C\right)^{\frac{3}{2}},$$

where $(x(0), v(0)) := (1, 0)$.

We then train a Neural ODE using $\{x(t_i)\}$ to identify the above system.

In particular, we will compare against backpropagation with online recursive checkpointing. In these examples, we will set $\lambda = 0.99$.

Experiments

Method	Loss ($\times 10^{-4}$)	Time (minutes)	Memory (effective checkpoints)
Reversible	0.9	$\mathbf{1.7 \pm 0.4}$	**2**
Checkpointing	0.9	280.0 ± 13.7	**2**
Checkpointing	0.9	30.3 ± 1.6	4
Checkpointing	0.9	10.6 ± 1.1	8
Checkpointing	0.9	9.6 ± 0.6	16
Checkpointing	0.9	8.7 ± 0.7	32
Checkpointing	0.9	5.5 ± 0.8	44

Table: Time and memory cost incurred when training a Neural ODE to identify Chandrasekhar's white dwarf equation (1000 time and training steps). Here, we apply the midpoint method and its reversible counterpart.

Experiments

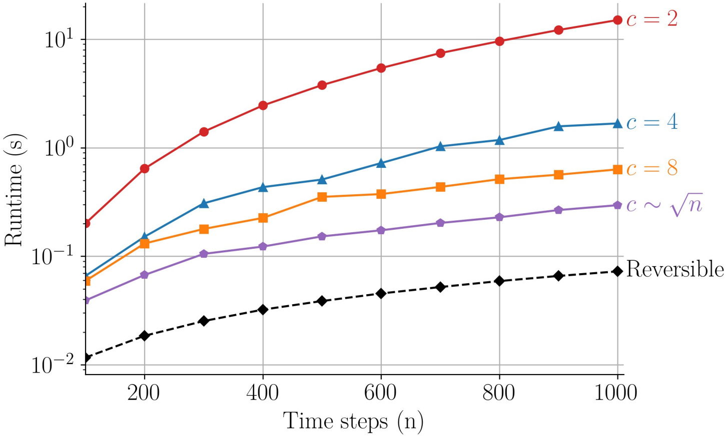


Figure: Combined runtime of a forwards solve and backpropagation through the midpoint ODE solver over n time steps. Here, we compare against backpropagation with online recursive checkpointing at c checkpoints.

Experiments

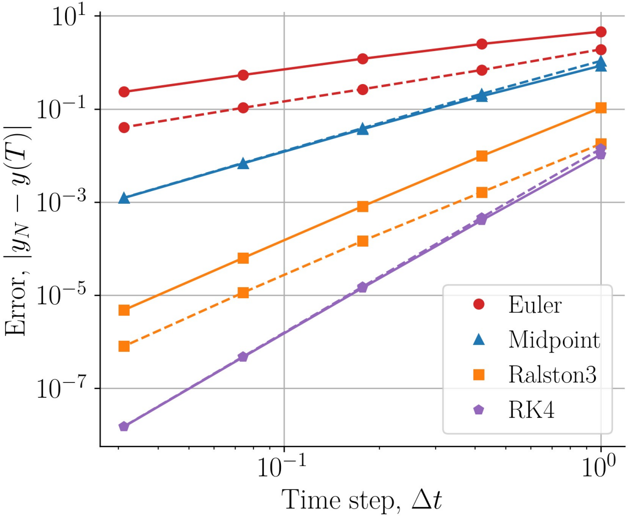


Figure: Convergence of original (solid) and reversible (dashed) ODE solvers.

Experiments

Next, we test the reversible solvers for identifying dynamics from real system data. Specifically, we will use the two datasets from [12], which were obtained from a coupled oscillator and chaotic double pendulum.

Method	Loss ($\times 10^{-3}$)	Time (minutes)	Memory (effective checkpoints)
Reversible	1.0 ± 0.2	$\mathbf{14.3 \pm 3.1}$	**2**
Checkpointing	1.0 ± 0.2	632.2 ± 20.0	**2**
Checkpointing	1.0 ± 0.2	99.0 ± 10.7	4
Checkpointing	1.0 ± 0.2	63.4 ± 9.8	8
Checkpointing	1.0 ± 0.2	53.8 ± 8.8	16
Checkpointing	1.0 ± 0.2	36.6 ± 7.8	31

Table: Time and memory cost incurred when training a Neural ODE to identify the dynamics of a coupled oscillator. Just as for the first experiment, the ODE is solved using the midpoint method or its reversible version.

Experiments

Method	Loss ($\times 10^{-3}$)	Time (minutes)	Memory (effective checkpoints)
Reversible	8.3 ± 3.2	$\mathbf{21.9 \pm 2.1}$	**2**
Checkpointing	9.5 ± 2.0	818.2 ± 21.5	**2**
Checkpointing	8.6 ± 1.9	135.2 ± 7.1	4
Checkpointing	12.8 ± 7.4	82.8 ± 1.2	8
Checkpointing	$\mathbf{7.8 \pm 1.3}$	70.8 ± 3.7	16
Checkpointing	7.9 ± 1.6	62.4 ± 2.7	32

Table: Time and memory cost incurred when training a Neural ODE to identify the dynamics of a chaotic double pendulum (for a short time, $t \in [0, 2]$). Here, the Bogacki-Shampine method is used with <u>adaptive</u> step sizes.

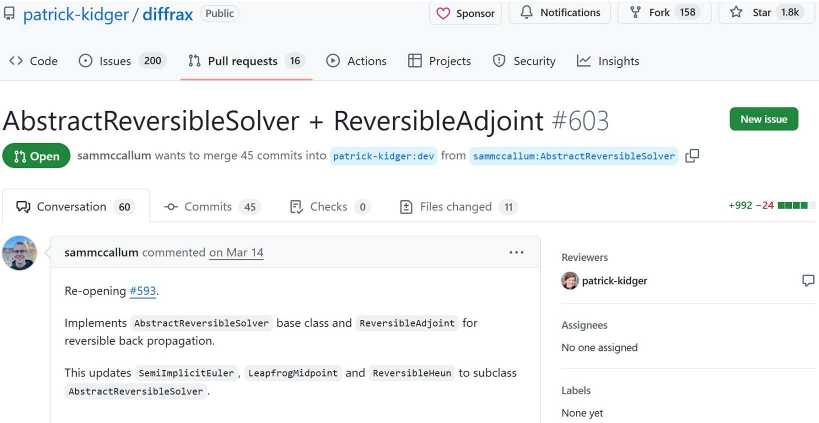
Outline

Conclusion

- Reversible solvers have seen recent interest due to the accurate and memory-efficient gradients that they provide during training.
- However, the current reversible ODE solvers have stability issues. We believe that this instability is amplified by the "$2A - B$" terms.
- We propose an approach in which an explicit ODE solver can be converted to a reversible one with the same order of convergence. Although this requires twice the function evaluations per step, we often observe faster training times due to the memory reduction.
- The reversible solvers produce stability regions and have shown promising empirical results – including against checkpointing.

Ongoing work

- Implementation of our method into the ODE/SDE/CDE library "Diffrax" (github.com/patrick-kidger/diffrax):



- Applications to fixed point systems (e.g. Deep Equilibrium Models).

Deep Equilibrium Models [13]

Given an input $x \in \mathbb{R}^d$, a Deep Equilibrium Model (DEQ) outputs $y \in \mathbb{R}^e$ as the fixed point given by a neural network $f_\theta : \mathbb{R}^e \times \mathbb{R}^d \to \mathbb{R}^e$. That is,

$$y := z^*, \quad \text{where} \quad z^* = f_\theta(z^*, x). \tag{3}$$

Using a standard fixed point solver, the DEQ would output z_N given by

$$z_{n+1} := f_\theta(z_n, x),$$

with $z_0 := 0$. This looks very similar to a feedforward neural network.

However... DEQs also have an "adjoint" equation for their gradients.

(and solving this equation will enable memory-efficient training!)

Deep Equilibrium Models [13]

Theorem (Adjoint fixed point system for DEQs)

For a scalar-valued loss function $L : \mathbb{R}^e \to \mathbb{R}$, we have

$$\frac{\partial}{\partial\theta}\big(L\big(f_\theta(z^*, x)\big)\big) = \left(\frac{\partial f_\theta(z^*, x)}{\partial\theta}\right)^{\top} g,$$

where g solves the fixed point equation

$$g = \left(\frac{\partial f_\theta(z^*, x)}{\partial z^*}\right)^{\top} g + \frac{\partial L}{\partial z}. \tag{4}$$

Just as before, solving (4) leads to gradients that are memory-efficient but approximate.

So, for accurate gradients, we propose a reversible fixed point solver.

Reversible Deep Equilibrium Models

joint with Sam McCallum (Bath) and Kamran Arora (Bath)

Definition (Reversible Deep Equilibrium Model (RevDEQ) [14])

Let $\beta \in (0, 2)$ with $\beta \neq 1$. Then we define the following iterative solver,

$$\begin{aligned} y_{n+1} &:= (1-\beta)y_n + \beta f_\theta(z_n, x), \\ z_{n+1} &:= (1-\beta)z_n + \beta f_\theta(y_{n+1}, x), \end{aligned} \tag{5}$$

where $y_0 = z_0 = 0$.

The fixed point solver (5) is algebraically reversible as

$$z_n = \frac{z_{n+1} - \beta f_\theta(y_{n+1}, x)}{1-\beta},$$

$$y_n = \frac{y_{n+1} - \beta f_\theta(z_n, x)}{1-\beta}.$$

Reversible Deep Equilibrium Models

Theorem (Linear convergence of RevDEQs)

Suppose that $f : \mathbb{R}^e \to \mathbb{R}^e$ *is contractive. That is, there exists* $L \in (0, 1)$*, such that*

$$\|f(y) - f(z)\| \leq L\|y - z\|,$$

for all $y, z \in \mathbb{R}^e$*. We define the sequence* $\{(y_n, z_n)\}_{n\geq 1}$ *by* $y_0 = z_0 = 0$ *and*

$$\begin{aligned} y_{n+1} &:= (1-\beta)y_n + \beta f(z_n), \\ z_{n+1} &:= (1-\beta)z_n + \beta f(y_{n+1}), \end{aligned} \tag{6}$$

where $\beta \in \left(0, \frac{2}{L+1}\right)$*. Then, letting* z^* *denote the unique fixed point of f, we have*

$$\begin{aligned} \|y_n - z^*\| &\leq \alpha^n \|z^*\|, \\ \|z_n - z^*\| &\leq \alpha^n \|z^*\|, \end{aligned}$$

for all $n \geq 1$ *where* $\alpha := |1-\beta| + \beta L$*.*

Reversible Deep Equilibrium Models

Example (decoder-only transformer)

$$f_\theta(z_{1:T}, x_{1:T}) = \mathsf{LN} \circ \phi \circ \mathsf{LN} \circ \mathsf{SA}\big(W_{QKV}(z_{1:T} + x_{1:T})\big),$$

where

- $x_{1:T} = (x_1, \cdots, x_T)$ is an input sequence of embeddings with $x_i \in \mathbb{R}^d$.
- LN is layer normalisation [16]
- ϕ is a two-layer neural network (MLP)
- SA is a self-attention operation with $W_{QKV} \in \mathbb{R}^{3d\times d}$ [17]

We set $\beta = 0.5$ and iterate the reversible fixed point solver until either

$$\|z_n - f(z_n)\|_2 < 10^{-3} \cdot \|f(z_n)\|_2 + 10^{-8} \quad \text{or} \quad n = 4.$$

We test this RevDEQ model on the standard Wikitext-103 dataset [18].

Reversible Deep Equilibrium Models

Model	Parameters	Function Evaluations	Perplexity
Transformer-XL (4 layers) [15]	139M	-	35.8
DEQ [13]	138M	30	32.4
RevDEQ	**110M**	**8**	**23.4**
Transformer-XL (16 layers) [15]	165M	-	24.3
DEQ-TrellisNet [13]	180M	30	29.0
DEQ-Transformer [13]	172M	30	24.2
RevDEQ	**169M**	**8**	**20.7**

Table: Test perplexity on the Wikitext-103 dataset.

Conclusion

- By developing numerical methods that are algebraically reversible, we can make backpropagation memory efficient when solving:
  (a) Differential Equations
  (b) Fixed point problems

- In both cases, our numerical methods involve twice the function evaluations compared to the corresponding standard methods. Though, we often see faster training due to the memory reduction.

- For (a), the challenge for "Reversible ODEs" is numerical stability. This motivated us to develop methods which have stability regions.

- For (b), the challenge for "Reversible DEQs" is computational cost. We hope the improved memory efficiency will compensate for this.

Thank you
for your attention!

and our preprints can be found at:

Sam McCallum and James Foster. *Efficient, Accurate and Stable Gradients for Neural ODEs*, arxiv.org/abs/2410.11648, 2024.

Sam McCallum, Kamran Arora and James Foster. *Reversible Deep Equilibrium Models*, arxiv.org/abs/2509.12917, 2025.

Outline

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*, github.com/patrick-kidger/diffrax/issues/417, 2024.

- S. McCallum and J. Foster. *Efficient, Accurate and Stable Gradients for Neural ODEs*, arXiv:2410.11648, 2024.

References III

W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud. *NFFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*, International Conference on Learning Representations, 2019.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, et al. *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 2011.

A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam and E. Keogh. *The UEA multivariate time series classification archive, 2018*, arXiv:1811.00075, 2018.

References IV

- M. Schmidt and Hod Lipson. *Distilling Free-Form Natural Laws from Experimental Data*, Science, vol. 324, no. 5923, 2009.
- S. Bai, J. Z. Kolter, V. Koltun. *Deep Equilibrium Models*, Neural Information Processing Systems, 2019.
- S. McCallum, K. Arora and J. Foster, *Reversible Deep Equilibrium Models*, arXiv:2509.12917, 2025.
- Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le and R. Salakhutdinov, *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), 2019.

References V

J. L. Ba, J. R. Kiros and G. E. Hinton. *Layer Normalization*, arXiv:1607.06450, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, *Attention Is All You Need*, Neural Information Processing Systems, 2017.

S. Merity, C. Xiong, J. Bradbury and R. Socher. *Pointer sentinel mixture models*. Proceedings of the 4th International Conference on Learning Representations (ICLR), 2016.
(uses the WikiText-103 dataset)

Examples of reversible solvers

Turbulence modelling 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.

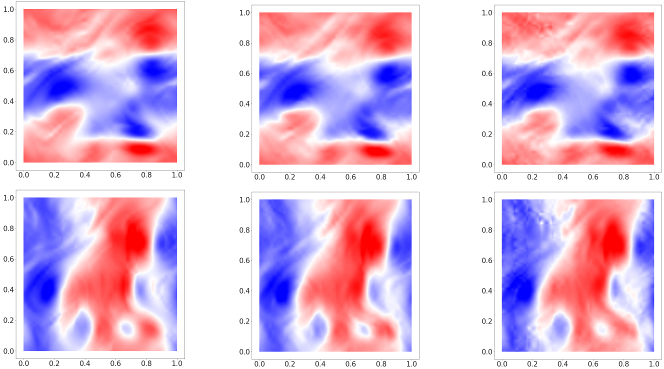


Figure: PDE simulation (left), Neural SDE (middle) and Neural network (right)