Deep learning as optimal control problems and Riemannian discrete gradient descent.

Elena Celledoni

Department of Mathematical Sciences, NTNU, Norway
joint work with
Martin Benning, Matthias Ehrhardt, Christian Etmann, Robert I McLachlan,
Brynjulf Owren, Carola B. Schönlieb, Ferdia Sherry

Les Houches Summer Week,
July 28, 2020

- Mathematics of deep learning.
- Structure preserving deep learning.

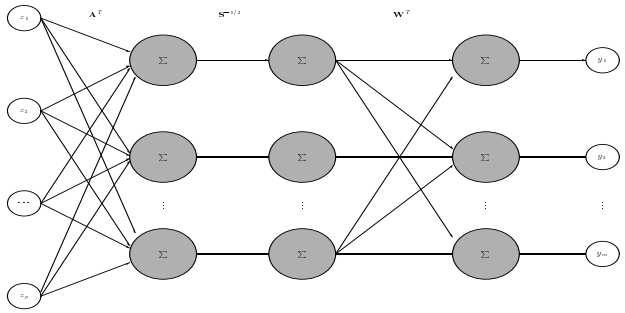
1. M. Benning, E. Celledoni, M. J. Ehrhardt, B. Owren, C. B. Schönlieb, *Deep learning as optimal control problems: models and numerical methods*. arXiv:1904.05657. Journal of Computational Dynamics.
2. E. Celledoni, M. J. Ehrhardt, C. Etmann, R.I. McLachlan, B. Owren, C. B. Schönlieb, F. Sherry, *Structure preserving deep learning*, arXiv:2006.03364.
3. E. Celledoni and S. Fiori, *Neural learning by geometric integration of reduced 'rigid-body' equations*, J CAM, 2004.

Outline:

- Deep learning as optimal control.
- Stability of forward propagation.
- Iterative methods for training (Hamiltonian descent).

ANN (change the picture if you can)

- An **artificial neural network** is a network of processing units (neurons) each taking inputs and producing outputs.

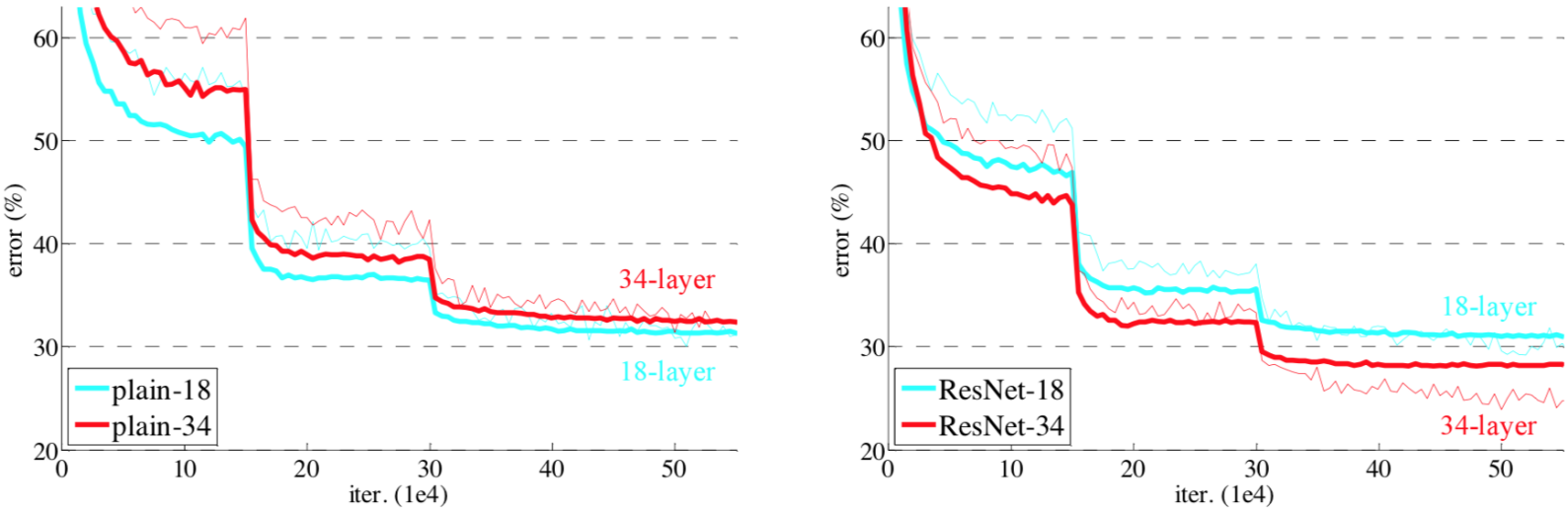


$$\mathbf{y}^{[out]} = \phi_{N-1} \circ \cdots \circ \phi_\ell \circ \cdots \circ \phi_0 \left(\mathbf{y}^{[in]}\right), \qquad \mathbf{y}^{[\ell+1]} = \phi_\ell(\mathbf{y}^{[\ell]}) = \phi(\mathbf{y}^{[\ell]}, U^{[\ell]}), \quad \mathbf{y}^{[0]} = \mathbf{y}^{[in]}$$

- **Learning**: is a process on the network obtained by establishing an appropriate cost function: $\mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$ for all $\mathbf{y}^{[in]} \in \mathcal{D}$. Optimise wrt parameters $U^{[1]}, \ldots, U^{[\ell]}, \ldots, U^{[N]}$.
- $\mathbf{y}^{[in]} \in \mathcal{D}$ belongs to a large database e.g. of images.
- **Stochastic gradient descent** for solving the optimisation problem.
- **Deep learning large** N: important e.g. for visual recognition tasks. Manifold valued for activity recognition. Deeper neural networks are more difficult to train.

Deep learning: classification problems (image recognition)

- He, K., Zhang, X., Ren, S., Sun, J. *Deep residual learning for image recognition.* In: CVPR. (2016)



Iterations of stochastic gradient descent

- Adding layers does not always decrease the error.
- DL can lead to vanishing/exploding gradients.
- ResNet works better than feed forward.

ResNet is made of *stacked "Residual Units"*:

- $$\mathbf{y}^{[\ell+1]} = \phi_\ell(\mathbf{y}^{[\ell]}) = \mathbf{y}^{[\ell]} + f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} = \mathbf{y}^{[in]} \in \mathcal{D}$$

  where

- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ contains the parameters to be determined (weights and biases).

ResNet is made of *stacked "Residual Units"*:

- $$\mathbf{y}^{[\ell+1]} = \phi_\ell(\mathbf{y}^{[\ell]}) = \mathbf{y}^{[\ell]} + f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} = \mathbf{y}^{[in]} \in \mathcal{D}$$

  where

- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ contains the parameters to be determined (weights and biases).
- f is the residual function in the ResNet jargon: the parameters interact linearly with the data, and there is a scalar nonlinearity acting component-wise

ResNet is made of *stacked "Residual Units"*:

- introduce a scalar parameter h_ℓ

$$\mathbf{y}^{[\ell+1]} = \phi_\ell(\mathbf{y}^{[\ell]}) = \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} = \mathbf{y}^{[in]} \in \mathcal{D}$$

- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ contains the parameters to be determined (weights and biases).
- f is the residual function in the ResNet jargon: the parameters interact linearly with the data, and there is a scalar nonlinearity acting component-wise

ResNet is made of *stacked "Residual Units"*:

- introduce a scalar parameter h_ℓ

$$\mathbf{y}^{[\ell+1]} = \phi_\ell(\mathbf{y}^{[\ell]}) = \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} = \mathbf{y}^{[in]} \in \mathcal{D}$$

- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ contains the parameters to be determined (weights and biases).
- f is the residual function in the ResNet jargon: the parameters interact linearly with the data, and there is a scalar nonlinearity acting component-wise
- we consider a cost function

$$\mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right) = \sum_{\mathbf{y}^{[in]} \in \mathcal{D}} \mathcal{J}(\mathbf{y}^{[out]})$$

Summarised

- $\mathbf{y}^{[in]} \in \mathcal{D}$ data
- N the number of layers, N output layer
- $U^{[\ell]}$, $\ell = 0, \ldots, N$, parameters that need to be learned
- $\mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$ for all $\mathbf{y}^{[in]} \in \mathcal{D}$, cost function

Summarised

- $\mathbf{y}^{[in]} \in \mathcal{D}$ data
- N the number of layers, N output layer
- $U^{[\ell]}$, $\ell = 0, \ldots, N$, parameters that need to be learned
- $\mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$ for all $\mathbf{y}^{[in]} \in \mathcal{D}$, cost function

Deep learning problem (ResNet)

$$\min_{U^{[\ell]},\, \ell=0,\ldots,N} \mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$$

subject to

$$\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} := \mathbf{y}^{[in]} \in \mathcal{D}, \qquad \mathbf{y}^{[out]} := \mathbf{y}^{[N]}$$

Summarised

- $\mathbf{y}^{[in]} \in \mathcal{D}$ data
- N the number of layers, N output layer
- $U^{[\ell]}$, $\ell = 0, \ldots, N$, parameters that need to be learned
- $\mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$ for all $\mathbf{y}^{[in]} \in \mathcal{D}$, cost function

Deep learning problem (ResNet)

$$\min_{U^{[\ell]},\, \ell=0,\ldots,N} \mathbb{J}\left(\{\mathbf{y}^{[out]}\}\right)$$

subject to

$$\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} := \mathbf{y}^{[in]} \in \mathcal{D}, \qquad \mathbf{y}^{[out]} := \mathbf{y}^{[N]}$$

Forward Euler discretization of $\dot{\mathbf{y}} = f(\mathbf{y}, U)$, $\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]})$, $h_\ell = 1$.

Interpretation as discrete optimal control problem

1. E. Haber and L. Ruthotto, *Stable Architectures for Deep Neural Networks*, arXiv: 1705.03341v2
2. Qianxiao Li, Long Chen, Cheng Tai, Weinan E, *Maximum Principle Based Algorithms for Deep Learning*, Journal of Machine Learning Research 18 (2018).

The **deep learning problem** can be seen as the discretization of

Optimal control problem

$$\min_{U(t)} \mathbb{J}(\{\mathbf{y}(T)\}), \quad t \in [0, T] \tag{1}$$

subject to

$$\dot{\mathbf{y}} = f(\mathbf{y}, U), \quad \mathbf{y}(0) = \mathbf{y}^{[in]} \in \mathcal{D}. \tag{2}$$

1. E. Haber and L. Ruthotto, *Stable Architectures for Deep Neural Networks*, arXiv: 1705.03341v2
2. Qianxiao Li, Long Chen, Cheng Tai, Weinan E, *Maximum Principle Based Algorithms for Deep Learning*, Journal of Machine Learning Research 18 (2018).

The **deep learning problem** can be seen as the discretization of

Optimal control problem

$$\min_{U(t)} \mathbb{J}(\{\mathbf{y}(T)\}), \quad t \in [0, T] \tag{1}$$

subject to

$$\dot{\mathbf{y}} = f(\mathbf{y}, U), \quad \mathbf{y}(0) = \mathbf{y}^{[in]} \in \mathcal{D}. \tag{2}$$

Why is the optimal control point of view useful:

- it states the deep learning problem in two lines;
- can be used to create new architectures;
- experience shows that continuous models are useful simplifications of the reality and they are easier to study;
- it offers a starting point for analysis.

Deep learning optimal control has attracted attention

- Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, Comm. in Math. and Stat. 2017.
- B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, *Reversible Architectures for Arbitrarily Deep Residual Neural Networks*, arXiv: 1709.03698v1, AAAI (National Conference on Artificial Intelligence).
- E. Haber, L. Ruthotto, *Stable Architectures for Deep Neural Networks*, arXiv: 1705.03341v2
- Qianxiao Li Shuji Hao, *An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks*
- Qianxiao Li, Long Chen, Cheng Tai, Weinan E, *Maximum Principle Based Algorithms for Deep Learning*, J. Machine Learning Research 18 (2018).
- L. Ruthotto and E. Haber, *Deep Neural Networks Motivated by Partial Differential Equations*, arXiv:1804.04272
- Lu, Y., Zhong, A., Li, Q., Bin Dong. (arXiv 2017). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations.
- Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. (2018). *Neural Ordinary Differential Equations*. Presented at NeurIPS.
- Gholami, A., Keutzer, K., Biros, G. (2019, February 26). *ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs.*
- Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., Mahoney, M. (2019, June 9). *ANODEV2: A Coupled Neural ODE Evolution Framework.*
- A Effland, E Kobler, K Kunisch, T Pock (J. Math. Im. 2019). *Variational Networks: An Optimal Control Approach to Early Stopping Variational Methods for Image Restoration*.
- J Aghili, O Mula, arXiv:2007.02428, 2020. *Depth-Adaptive Neural Networks from the Optimal Control viewpoint*.

- We derive **necessary conditions for optimality** of the deep learning optimal control problem leading to a Hamiltonian BVP.
- Investigation of **different Runge-Kutta discretizations** of the underlying continuous deep learning problem.
- The usual numerical approach is *first discretize* $\mathcal{J}$ *then optimise*, and leads to a gradient descent method for determining the control parameters U.
- An alternative approach is *first optimise then discretize*. This method leads to **explicit formulae for the gradients**, these are useful for implementation or for analysis.
- We have **implemented a general RK-discretization** of the deep learning optimal control problem compared it with ResNet and Net.
- The discretizations are adaptive in time. **Learning the step-size**, the number of layers is determined automatically by the algorithms.

First order necessary conditions for optimality of the optimal control problem: Pontryagin maximum principle.

We denote by $\mathbf{y}$ one sample of data propagated through the layers.

Theorem The BVP system

$$\begin{aligned} \dot{\mathbf{y}} &= f(\mathbf{y}, U), \\ \dot{\mathbf{p}} &= -\left(\partial_{\mathbf{y}} f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \\ 0 &= \left(\partial_U f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \end{aligned}$$

with boundary conditions $\mathbf{y}(0) = \mathbf{x}$, $\mathbf{p}(T) = \nabla_{\mathbf{y}} \mathcal{J}|_{\mathbf{y}(T)}$, expresses the first order necessary conditions for optimality of the optimal control problem.

BVP is a **constrained Hamiltonian system** with Hamiltonian

$$H(\mathbf{y}, \mathbf{p}, U) = \mathbf{p}^T f(\mathbf{y}, U).$$

This is an index one differential algebraic equation.

First order necessary conditions for optimality of the optimal control problem: Pontryagin maximum principle.

We denote by $\mathbf{y}$ one sample of data propagated through the layers.

Theorem The BVP system

$$\begin{aligned} \dot{\mathbf{y}} &= f(\mathbf{y}, U), \\ \dot{\mathbf{p}} &= -\left(\partial_{\mathbf{y}} f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \\ 0 &= \left(\partial_U f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \end{aligned}$$

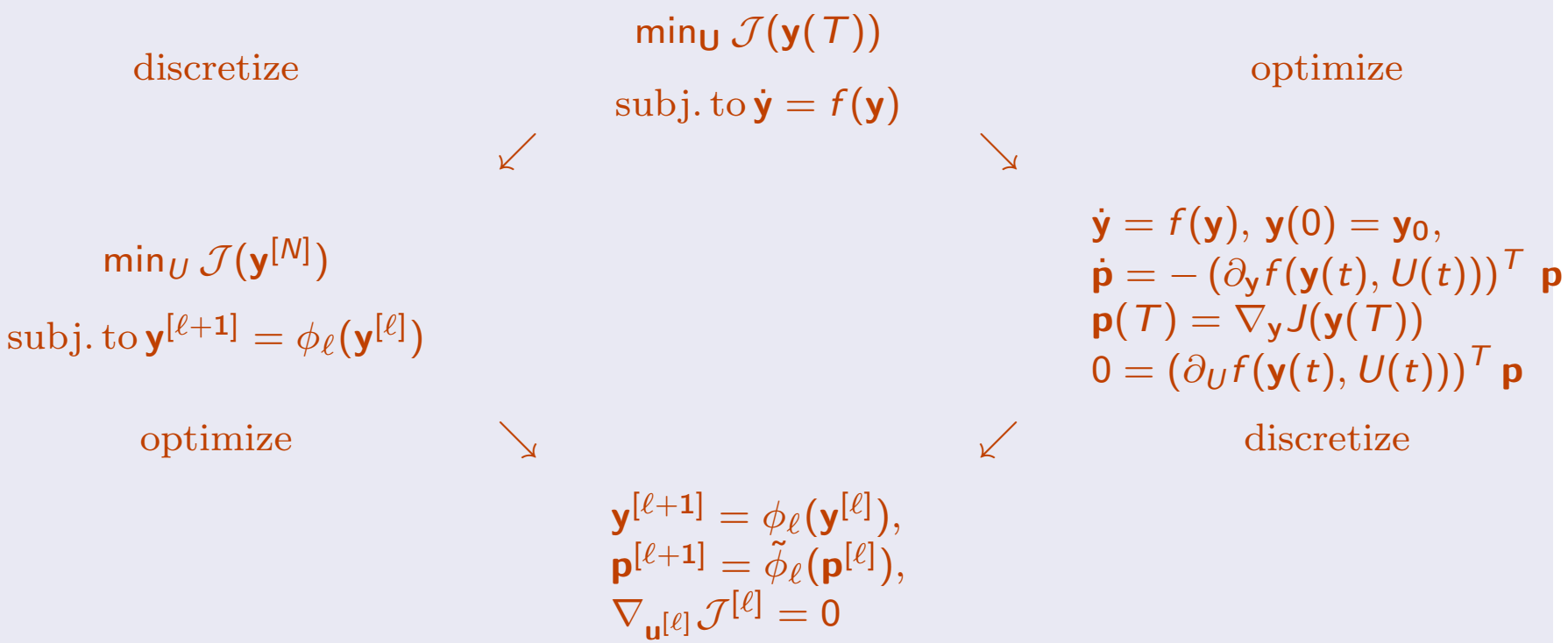
with boundary conditions $\mathbf{y}(0) = \mathbf{x}$, $\mathbf{p}(T) = \nabla_{\mathbf{y}} \mathcal{J}|_{\mathbf{y}(T)}$, expresses the first order necessary conditions for optimality of the optimal control problem.

BVP is a **constrained Hamiltonian system** with Hamiltonian

$$H(\mathbf{y}, \mathbf{p}, U) = \mathbf{p}^T f(\mathbf{y}, U).$$

This is an index one differential algebraic equation. **Symplectic partitioned Runge-Kutta methods** are suited to this problem, because they respect the variational nature of the problem.

With SPRK: Discretize+optimise same as Optimise+discretize.



Where ϕ_ℓ and $\tilde{\phi}_\ell$ denote a pair of partitioned, symplectic Runge-Kutta methods.

Theorem. The discrete BVP system

$$\begin{aligned}
\mathbf{y}^{[\ell+1]} &= \mathbf{y}^{[\ell]} + hf(\mathbf{y}^{[\ell]}U^{[\ell]}), \qquad \ell = 1, \dots, N \\
\mathbf{p}^{[\ell+1]} &= \mathbf{p}^{[\ell]} - h\left(\partial_{\mathbf{y}} f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}_{[\ell+1]}, \qquad \ell = 0, \dots, N-1 \\
0 &= \left(\partial_U f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}^{[\ell+1]}, \qquad \ell = 0, \dots, N-1.
\end{aligned}$$

(here *symplectic Euler method* but true for all symplectic partitioned RK), expresses the first order necessary conditions for optimality of the following **discrete** optimal control problem (**deep learning problem**)

$$\min_{(\mathbf{y}^{[\ell]}, U^{[\ell]}),\, \ell=0,\dots,N-1} \mathcal{J}(\mathbf{y}^{[N]}),$$

subject to

$$\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + hf(\mathbf{y}^{[\ell]}, U^{[\ell]}), \quad \mathbf{y}(0) = \mathbf{y}^{[0]}.$$

Proof: analogue to the continuous case.

Let $\mathbf{y}^{[\ell]}$ and $\mathbf{p}^{[\ell]}$ be given by a partitioned Runge-Kutta method applied to the Hamiltonian BVP. Then the gradient of the cost function $\mathcal{J}$ with respect to the control parameters is given by

$$\lambda_i^{[\ell]} = -\partial_{\mathbf{y}} f(\mathbf{y}_i^{[\ell]}, U^{[\ell]})^T \left(\mathbf{p}^{[\ell+1]} - h \sum_{k=1}^{s} \frac{a_{k,i} b_k}{b_i} \lambda_k^{[\ell]} \right) \quad i = 1, \ldots, s$$

$$\partial_{U^{[\ell]}} \mathcal{J}(y^{[N]}) = h \sum_{i=1}^{s} b_i \partial_{U^{[\ell]}} f(\mathbf{y}_i^{[\ell]}, U^{[\ell]})^T \left(\mathbf{p}^{[\ell+1]} - h \sum_{k=1}^{s} \frac{a_{k,i} b_k}{b_i} \lambda_k^{[\ell]} \right).$$

Remark: We assume there is only one control set per step, same $U^{[\ell]}$ is used for all the stages in layer ℓ.

Algorithm: Training ODE-inspired neural networks with gradient descent.

Input: initial guess for the controls U, step size τ

for $\ell = 1, \ldots, N$ **do**

forward propagation with explicit RK method: compute $\mathbf{y}$ via

$$\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + h_\ell \, f(\mathbf{y}^{[\ell]}, U^{[\ell]})$$

end for

for $\ell = N, \ldots, 1$ **do**

backpropagation with adjoint method: compute $\mathbf{p}$ via

$$\mathbf{p}^{[\ell+1]} = \mathbf{p}^{[\ell]} - h_\ell \left(\partial_{\mathbf{y}} f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}^{[\ell+1]}$$

$$\left(\partial_{\mathbf{y}} f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}^{[\ell+1]} = {K^{[\ell]}}^T \gamma^{[\ell]}$$

and $\gamma^{[\ell]}$ can be computed explicitly.
Compute the gradient $g^{[\ell]} := (\partial_{K^{[\ell]}} \mathcal{J}(\mathbf{y}^{[N]}), \partial_{\mathbf{b}^{[\ell]}} \mathcal{J}(\mathbf{y}^{[N]}))$ using the explicit formulae:

$$\partial_{K^{[\ell]}} \mathcal{J}(\mathbf{y}^{[N]}) = h_\ell \, \gamma^{[\ell]} \, \mathbf{y}^{[\ell],T}$$

$$\partial_{\mathbf{b}^{[\ell]}} \mathcal{J}(\mathbf{y}^{[N]}) = h_\ell \, \gamma^{[\ell]}.$$

end for

update controls: $U = U - \tau g$

τ chosen via backtracking.

- Hager proves convergence of order p for SPRK discretizations to the solution of the optimal control problem under a coercivity assumption.
- In the context of deep learning, Thorpe and van Gennip, study deep limits for ResNet by means of gamma convergence.

References

- W. Hager. *Runge-Kutta methods in optimal control and the transformed adjoint system.* Num. Math., **87**(2):247–282, 2000.
- J. M. Sanz-Serna, *Symplectic Runge-Kutta schemes for adjoint equations automatic differentiation, optimal control and more*, SIAM Rev. **58**, (2015), 3–33.
- M. Thorpe and Y. van Gennip. *Deep limits of residual neural networks.* arXiv preprintarXiv:1810.11741, 2018.

Test sets

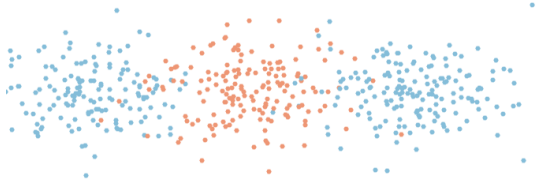


Figure: Dataset: `donut1d`

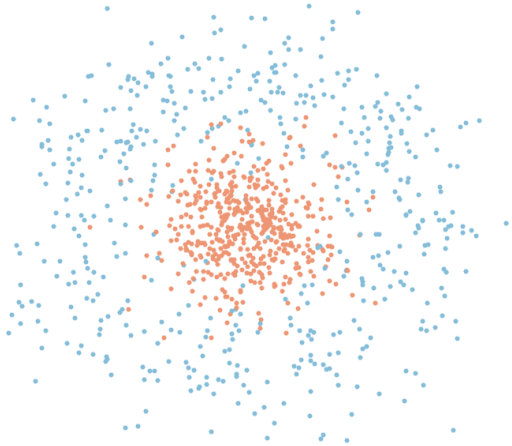


Figure: Dataset: `donut2d`

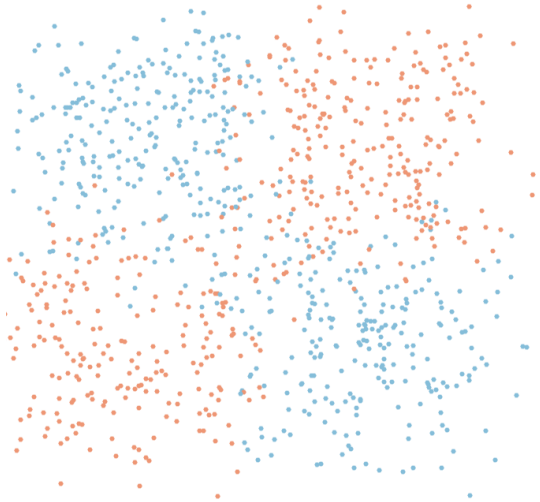


Figure: Dataset: `squares`

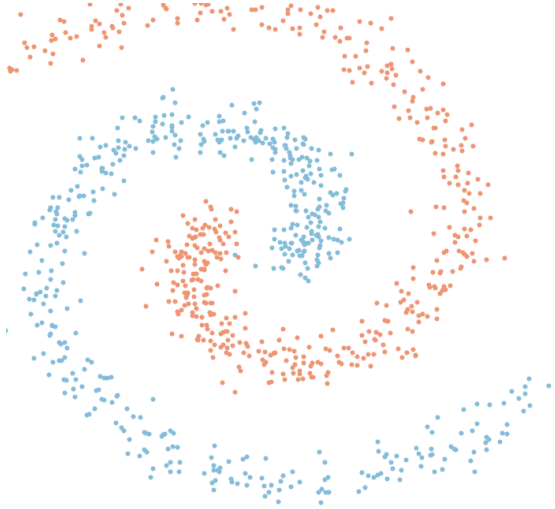


Figure: Dataset: `spiral`

- to training datum $\mathbf{y}^{[in]}$ it corresponds a training label $c^{[in]}$
- $\mathbf{y}^{[\ell+1]} = \phi(\mathbf{y}^{[\ell]}, U^{[\ell]}), \qquad \mathbf{y}^{[0]} = \mathbf{y}^{[in]} \in \mathcal{D}$
- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ contains the parameters to be determined (weights and biases).
- f is the residual function,

  $f(\mathbf{y}^{[\ell]}, U^{[\ell]}) := \sigma(K^{[\ell]}\mathbf{y}^{[\ell]} + \mathbf{b}^{[\ell]}),$

  $\sigma(\xi) := \tanh(\xi),$

  $\sigma(\xi) := \max\{0, \xi\},$

  σ is the activation function
- we consider a cost function

$$\sum_{\mathbf{y}^{[in]} \in \mathcal{D}} \mathcal{J}(\mathbf{y}^{[out]}) := \frac{1}{2} \sum_{\mathbf{y}^{[in]} \in \mathcal{D}} \|\mathcal{C}(K^{[out]}\mathbf{y}^{[out]} + \mathbf{b}^{[out]}) - c^{[in]}\|_2^2,$$

where the hypothesis function (or classifier) $\mathcal{C}(\xi) := \frac{\exp(\xi)}{1+\exp(\xi)}$, $c^{[in]}$ is the label representing the truth for datum $\mathbf{y}^{[in]}$.

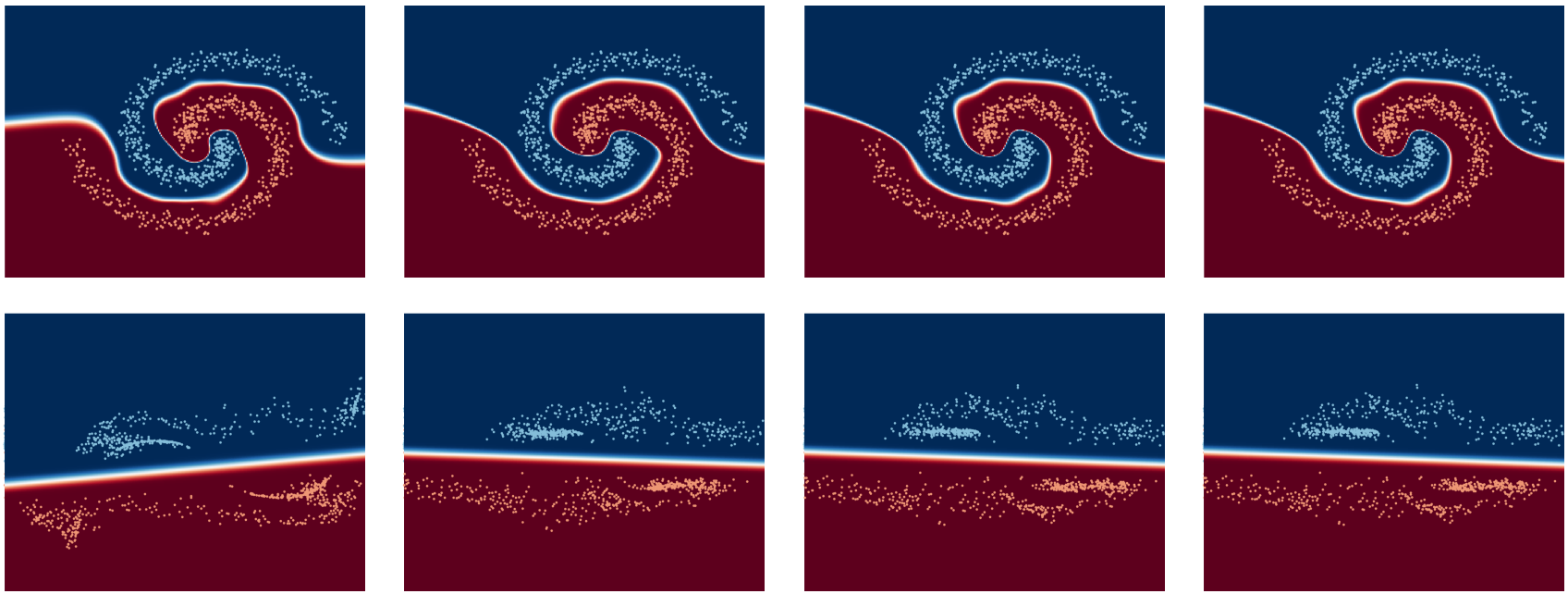
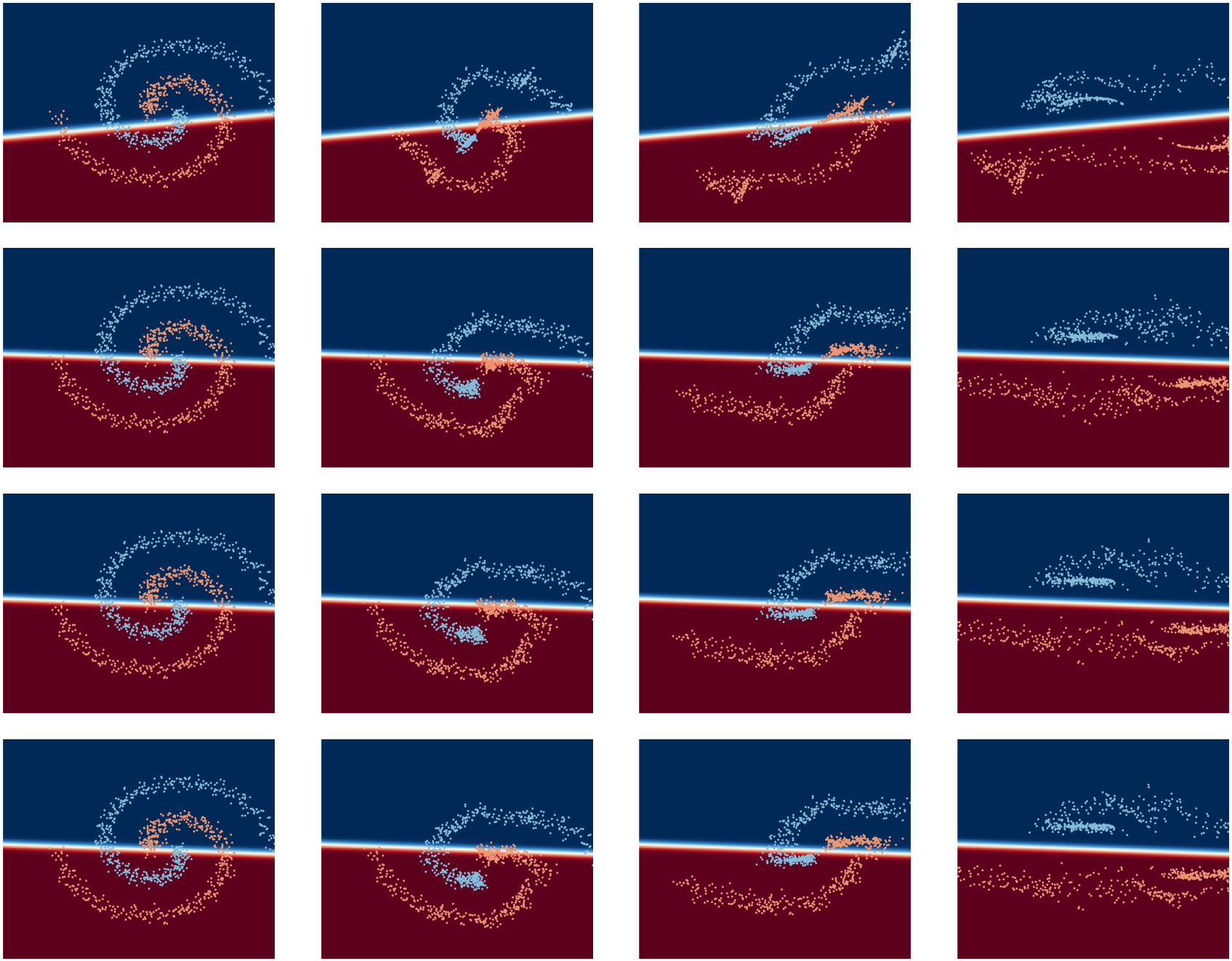


Figure: Prediction (top) and transformed points (bottom) for the Runge–Kutta methods (left to right) ResNet/Euler, Improved Euler, Kutta(3) and Kutta(4). The data set is `spiral` and we use 15 layers.

Transitions in Runge–Kutta methods – spiral



Comparison of Net, ResNet, ODENet and ODENet+simplex

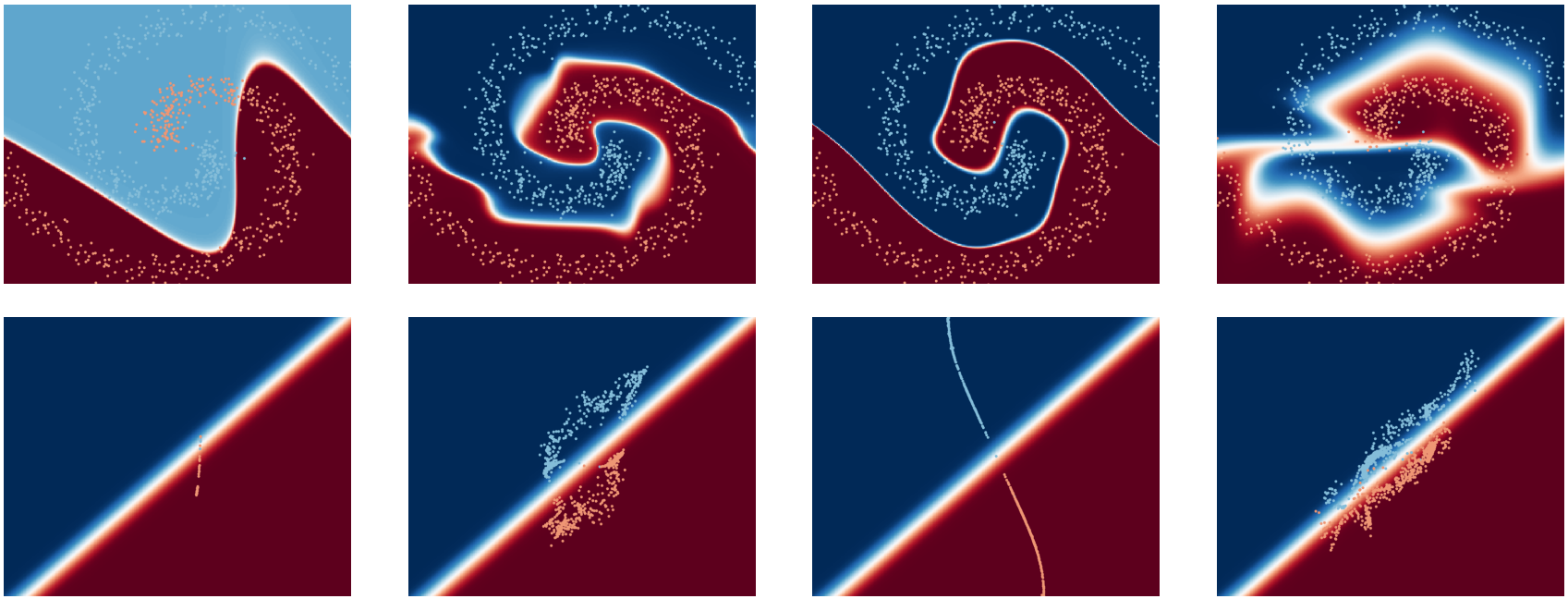


Figure: Learned transformation with fixed classifier. Prediction (top) and transformed data (`spiral`) with linear classifier (bottom) and for Net, ResNet, ODENet and ODENet+simplex.

Summary

- We have formulated the ResNet model as an optimal control problem leading to a Hamiltonian boundary value problem
- We have derived explicit formulas for computing the gradient of the cost function in the model
- We have implemented several methods
  - Arbitrary RK formulas with constant step size
  - The ResNet model where the time step is also a learned parameter of the model
  - We have compared with a standard `FeedForward` algorithm (not related to ODEs)
- We find that different RK methods perform rather similarly. This may indicate that the underlying ODE means something in the deep learning architecture, it seems to be a common denominator for several different methods
- Leaving h as a parameter in the model improves the performance and leads to some interesting effects

M. Benning, E. Celledoni, M. J. Ehrhardt, B. Owren, C. B. Schönlieb, *Deep learning as optimal control problems: models and numerical methods*.

We assume the parameters $U(t) = (A(t), b(t))$ are fixed. Let

$$f(t, \mathbf{y}(t)) := f(\mathbf{y}(t), U(t)) = \sigma(A(t)y(t) + b(t))$$

be the vector field propagating the data through the network. **What structural properties should f have?**

- For example could require that two nearby solutions satisfy

  $$\|\mathbf{y}_1(T) - \mathbf{y}_2(T)\| \leq C\|\mathbf{y}_1(0) - \mathbf{y}_2(0)\|$$

  for a moderatly sized constant C.
  If f is Lipschitz in ite second argument then $C = e^{TL}$, L Lipschitz constant, and $L = L_\sigma \max_t \|A(t)\|$.
- Haber and Ruthotto suggest to use the eigenvalues of the linearized ODE to analyze the stability, this could give indications of stability when the parameters $A(t)$ and $b(t)$ are smooth and vary slowly.

  E. Haber and L. Ruthotto. *Stable architectures for deep neural networks.* Inverse Problems, 34(1):014004, 2017.

Stability of the forward propagation

- Stability can be studied in terms of **Lyapunov** functions $V(y)$ non-increasing (or constant) along solution trajectories (Hopfield networks).

  J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the national academy of sciences, 79(8):2554–2558, 1982
- For nonlinear ODEs **contractivity** in L^2-norm using a one-sided Lipschitz constant $\nu \in \mathbb{R}$ such that for all admissible y_1, y_2 and $t \in [0, T]$ we have

$$\langle f(t, y_2) - f(t, y_1), y_2 - y_1 \rangle \leq \nu \|y_2 - y_1\|^2.$$

  In this case for any two solutions $y_1(t)$ and $y_2(t)$

$$\|\mathbf{y}_1(t) - \mathbf{y}_2(t)\| \leq e^{t\nu} \|\mathbf{y}_1(0) - \mathbf{y}_2(0)\|$$

  so that the problem is contractive if $\nu \leq 0$.
- for $f(t, y(t)) = \sigma(A(t)y(t) + b(t))$, $\nu \leq \sup_{t,D} \lambda_{\max}\left(\frac{DA(t)+(DA(t))^T}{2}\right)$ with D with diagonal entries in $\sigma'(\mathbb{R})$

For σ ReLU function, Zhang and Schaeffer (2020) give growth and stability estimates. We consider a simplified case

$$\dot{y} = -A(t)^T\,\sigma(A(t)y(t) + b(t))$$

which is a gradient system in the sense that

$$\dot{y} = -\nabla_y V, \qquad V(t, y(t)) = \gamma(A(t)y(t) + b(t))\mathbf{1}, \quad \gamma' = \sigma.$$

Theorem

- Let $V(t, y(t))$ be twice differentiable and convex in the second argument. Then $f(t, y(t)) = -\nabla_y V$ satisfies a one-sided Lipschitz condition with $\nu \leq 0$.
- Let σ be absolutely continuous and $0 \leq \sigma'(s) \leq 1$ a.e. in $\mathbb{R}$. Then $f(t, y(t))$ satisfies a one-sided Lipschitz condition for any choice of $A(t)$ and $b(t)$ with

$$-\mu_*^2 \leq \nu_\sigma \leq 0, \qquad \mu_* = \min_t \mu(t)$$

and $\mu(t)$ is the smallest singular value of $A(t)$. In particular $\nu_\sigma = -\mu_*^2$ is attained when $\sigma(s) = s$.

However "too much" contractivity might not be what's needed in deep learning. **Hamiltonian vector fields** $f(t, y(t))$ (non-autonomous) have been proposed by Haber and Ruthotto.

- B-stable Runge-Kutta methods (implicit) preserve **contractivity** independently on the step-size h. Example: implicit Euler method.
- More interesting for deep learning are explicit methods. Consider the monotonicity condition

$$\langle f(t, y_2) - f(t, y_1), y_2 - y_1 \rangle \leq \bar{\nu} \|f(t, y_2) - f(t, y_1)\|^2,$$

with constant $\bar{\nu}$. One can prove that all explicit RK methods with positive weights $b_1, \ldots, b_s$ are contractive for step-sizes

$$h < -\bar{\nu} r$$

where r is a method dependent constant, e.g. $r = 1$ for explicit Euler and RK4. (Dahlquist, 1979).

- Time integration of (non-autonomous) Hamiltonian systems is not well understood.

- methods gradient based
- amenable for implementations *a-la* stochastic gradient

Gradient descent, Hamiltonian descent and Adam

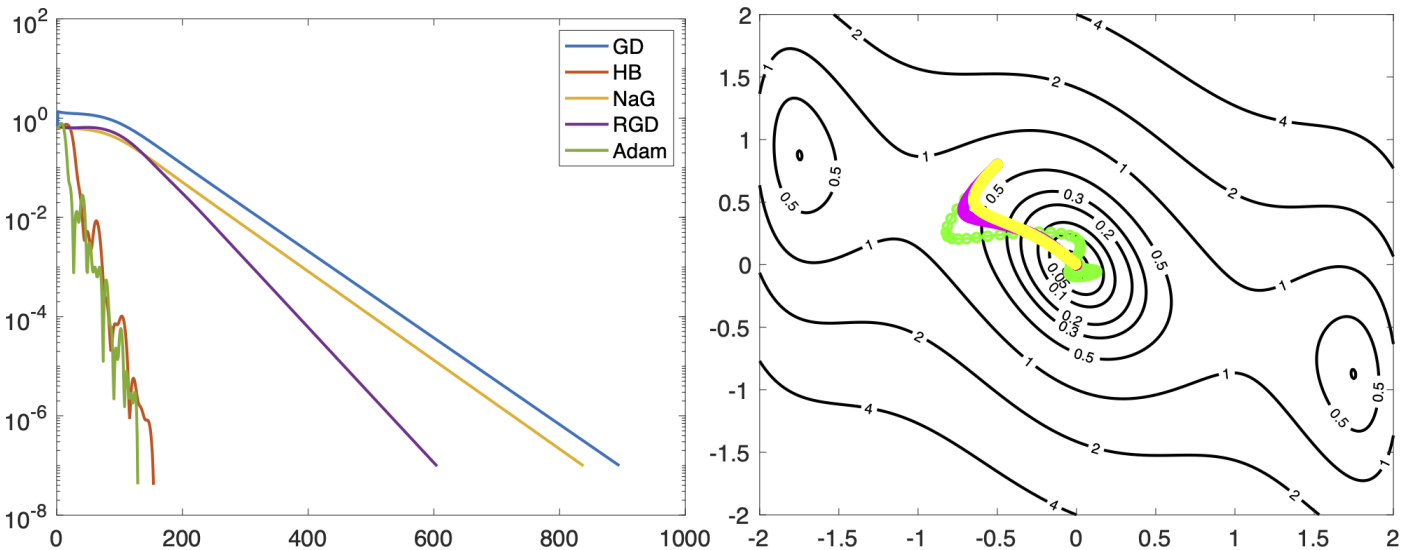


Figure: Camel function: $2x^2 - 1.05\,x^4 + \frac{1}{6}x^6 + xy + y^2$, $\mathbf{u}_0 = (-0.5, 0.8)$. Global minimum at $(0, 0)$.

GD, NaG, HB $h = 0.01$, RGD $h = 0.0001$, Adam $h = 0.1$.

Accelerating gradient descent $\dot{\mathbf{u}} = -\nabla J(\mathbf{u})$: Hamiltonian descent

Consider:

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla J(\mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \frac{\mathbf{p}}{m}. \end{aligned}$$

rewritten as a second order equation for $\mathbf{u}$ is $\ddot{\mathbf{u}} + \frac{\gamma}{m}\dot{\mathbf{u}} = -\frac{1}{m}\nabla J(\mathbf{u})$ which is gradient decent accelerated by momentum. Leading after discretization to **Polyak's heavy ball method** (momentum method):

$$\begin{aligned} \mathbf{p}_{n+1} &= \mathbf{p}_n - h\nabla J(\mathbf{u}_n) - h\gamma \mathbf{p}_n, \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{h}{m}\mathbf{p}_{n+1}, \end{aligned}$$

is a partitioned Runge-Kutta method (forward Euler + backward Euler).
This system is also an example of a *conformal Hamiltonian* system: let $H(\mathbf{p}, \mathbf{u}) = K(\mathbf{p}) + J(\mathbf{u})$ be a (separable) Hamiltonian

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla_{\mathbf{u}} H(\mathbf{p}, \mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \nabla_{\mathbf{p}} H(\mathbf{p}, \mathbf{u}), \end{aligned}$$

with $H(\mathbf{p}, \mathbf{u}) = -\frac{1}{2m}\|\mathbf{p}\|_2^2 + J(\mathbf{u})$, another choice is $K(\mathbf{p}) = \sqrt{\|\mathbf{p}\|^2 + \epsilon}$ leading to *relativistic gradient descent*:

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla_{\mathbf{u}} J(\mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \frac{\mathbf{p}}{\sqrt{\epsilon + \|\mathbf{p}\|^2}}. \end{aligned}$$

Accelerating gradient descent $\dot{\mathbf{u}} = -\nabla J(\mathbf{u})$*: Hamiltonian descent*

Conformal Hamiltonian system, $H(\mathbf{p}, \mathbf{u})$ Hamiltonian function

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla_{\mathbf{u}} H(\mathbf{p}, \mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \nabla_{\mathbf{p}} H(\mathbf{p}, \mathbf{u}). \end{aligned}$$

- Energy dissipative: $\frac{d}{dt} H(\mathbf{p}, \mathbf{u}) = -\frac{\gamma}{2m} \mathbf{p}^T \mathbf{p}$
- Note that if $H(\mathbf{p}, \mathbf{u}) = K(\mathbf{p}) + J(\mathbf{u})$ is separable and $K(\mathbf{p})$ has a global minimiser in 0, limit points of the conformal Hamiltonian system recover stationary points of $J(\mathbf{u})$.

  In fact the equilibria of the conformal Hamiltonian system fulfil

  $$\gamma \mathbf{p} = \nabla_{\mathbf{u}} J(\mathbf{u}), \quad 0 = \nabla_{\mathbf{p}} K(\mathbf{p}).$$

  Let $\mathbf{p}^* = 0$ be the unique global minimum of K then $(\mathbf{u}^*, 0)$

  $$\nabla H(\mathbf{u}^*, 0) = 0 \iff \nabla J(\mathbf{u}^*) = 0$$

  i.e. if and only if $\mathbf{u}^*$ is a stationary point of $J(\mathbf{u})$.
- **Conformal symplectic** numerical methods for the conformal Hamiltonian system, guarantee the correct rate of energy dissipation and can be used to compute the equilibria $(\mathbf{u}^*, 0)$, where $\mathbf{u}^*$ is a stationary point of $J(\mathbf{u})$.

Consider the *conformal Hamiltonian* system:

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla J(\mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \frac{\mathbf{p}}{m}, \end{aligned} \qquad H(\mathbf{p}, \mathbf{u}) = \frac{1}{2m}\|\mathbf{p}\|_2^2 + J(\mathbf{u}),$$

Polyak's heavy ball method (momentum method):

$$\begin{aligned} \mathbf{p}_{n+1} &= \mathbf{p}_n - h\nabla J(\mathbf{u}_n) - h\gamma \mathbf{p}_n, \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{h}{m}\mathbf{p}_{n+1}, \end{aligned}$$

is a partitioned Runge-Kutta method (forward Euler + backward Euler), is conformal symplectic.

Consider the *conformal Hamiltonian* system:

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla J(\mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \frac{\mathbf{p}}{m}, \end{aligned} \qquad H(\mathbf{p}, \mathbf{u}) = \frac{1}{2m}\|\mathbf{p}\|_2^2 + J(\mathbf{u}),$$

Polyak's heavy ball method (momentum method):

$$\begin{aligned} \mathbf{p}_{n+1} &= \mathbf{p}_n - h\nabla J(\mathbf{u}_n) - h\gamma \mathbf{p}_n, \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{h}{m}\mathbf{p}_{n+1}, \end{aligned}$$

is a partitioned Runge-Kutta method (forward Euler + backward Euler), is conformal symplectic.

Nesterov accelerated gradient:

$$\begin{aligned} \mathbf{u}_{n+\frac{1}{2}} &= \mathbf{u}_n + \mu \mathbf{p}_n, \\ \mathbf{p}_{n+1} &= \mu \mathbf{p}_n - h\nabla J(\mathbf{u}_{n+\frac{1}{2}}), \\ \mathbf{u}_{n+1} &= \mathbf{u}_{n+\frac{1}{2}} + \frac{h}{m}\mathbf{p}_{n+1}, \end{aligned}$$

Störmer-Verlet applied to the conformal Hamiltonian sys. after $\tilde{\mathbf{p}} = e^{\gamma t}\mathbf{p}$.

$$\begin{aligned} \dot{\mathbf{p}} &= -\nabla J(\mathbf{u}) - \gamma \mathbf{p}, \quad \gamma > 0, \\ \dot{\mathbf{u}} &= \nabla K(\mathbf{p}), \end{aligned} \qquad H(\mathbf{p}, \mathbf{u}) = K(\mathbf{p}) + J(\mathbf{u}),$$

choose $K(\mathbf{p})$ to get faster convergence e.g.

$$K(\mathbf{p}) = \sqrt{\|\mathbf{p}\|^2 + \epsilon}.$$

RGD related to Adam

$$\mathbf{p}_{n+1} = \frac{1}{1 + h\gamma}(\mathbf{p}_n - h \nabla J(\mathbf{u}_n)), \tag{3}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\mathbf{p}_{n+1}}{\sqrt{\mathbf{p}_{n+1}^T \mathbf{p}_{n+1} + \epsilon}}. \tag{4}$$

G. Franca, J. Sulam, D. Robinson, R. Vidal, *Conformal Symplectic and Relativistic Optimization*, 2019 arXiv

References on conformal Hamiltionian systems and Hamiltonian descent

Relating algorithms to solutions of systems ODEs

1. has been useful to prove their convergence properties;
2. can lead to new and improved iterative methods based on new numerical discretizations;
3. can be useful to gerenalize the methods to the case when the parameters belong to a manifold e.g. a Riemannian manifold.

Conformal Hamiltonian systems and conformal symplectic methods

- R.I. McLachlan and M. Perlmutter, *Conformal Hamiltonian systems*, 2001, J. Geom. Phys.
- R.I. McLachlan, R. Quispel, *Geometric integrators for ODEs*, J. Phys A, 2006.
- A. Bhatt, D. Floyd, and B.E. Moore, *Second Order Conformal Symplectic Schemes for Damped Hamiltonian Systems*, Journal of Scientific Computing, 66(3):1234-1259, 2016.

Hamiltonian descent and Relativistic Gradient Descent

- CJ Maddison, D Paulin, YW Teh, B O'Donoghue, A Doucet, *Hamiltonian descent methods*, arXiv:1809.05042
- B O'Donoghue, CJ Maddison, *Hamiltonian descent for composite objectives*, NurIPS.
- G. Franca, J. Sulam, D. Robinson, R. Vidal, *Conformal Symplectic and Relativistic Optimization*, 2019 arXiv
- W. Su, S. Boyd, E. J. Candes, *A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights*, arXiv:1503.01243

Stiefel $\mathcal{M} = \mathbb{V}_{n,p}$ and Grassmann $\mathcal{M} = \mathbb{G}_{n,p}$ manifolds

	Stiefel manifold	*Grassmann manifold*
$Q \in \mathcal{M}$	$Q^T Q = I_{p\times p}$	$\tilde{Q} = Q\Lambda$, $Q \in \mathbb{V}_{n,p}$, any $\Lambda \in SO(p)$
$V \in T_Q\mathcal{M}$	$V^T Q$ skew-symmetric	$V^T Q = O$

- **Orthogonality constraints** have shown to be a useful tool to regularise the learned parameters and improve the speed and efficiency of the training of neural networks.
- **Applications**: This approach has been used for e.g. *blind source separation of signals* ICA and in CNNs.

- *Neural learning by geometric integration of reduced 'rigid-body' equations* E C, S Fiori, J CAM 172 (2), 247-269
- *A class of intrinsic schemes for orthogonal integration* E C, B Owren, SIAM J. on Num. Anal. 40 (6), 2069-2084
- *On the implementation of Lie group methods on the Stiefel manifold* E C, B Owren, Numerical Algorithms 32 (2-4), 163-183

MEC learning is a second order learning algorithm on the Stiefel manifold. We restrict here to the case of one layer network. We assume we optimise for one element of $\mathbb{V}_{n,p}$.

$$\begin{aligned} \dot{\mathbf{B}} &= (\mathbf{F}+\mathbf{P})\mathbf{U}^T - \mathbf{U}(\mathbf{F}+\mathbf{P})^T, \\ \dot{\mathbf{U}} &= \mathbf{B}\mathbf{U} \end{aligned}$$

with $\mathbf{P} = -\gamma \mathbf{B}\mathbf{U}$. Where $\mathbf{y} = \sigma(\mathbf{U}^T\mathbf{x} + \mathbf{b})$ is the map from input $\mathbf{x}$ to output $\mathbf{y}$, and $\mathbf{F} = -\frac{\partial J}{\partial \mathbf{U}}$.

Gradient descent/Hamiltonian descent on Riemannian manifolds

Consider $(\mathcal{M}, g)$ a Riemannian manifold grad the Riemannian gradient

$$\dot{u} = -\mathrm{grad}\, J(u)$$

descent methods can be designed using **retractions** and discrete (Riemannian) gradients (e.g. coordinate descent Itoh-Abe methods).

Gradient descent/Hamiltonian descent on Riemannian manifolds

Consider $(\mathcal{M}, g)$ a Riemannian manifold grad the Riemannian gradient

$$\dot{u} = -\mathrm{grad}\, J(u)$$

descent methods can be designed using **retractions** and discrete (Riemannian) gradients (e.g. coordinate descent Itoh-Abe methods).

- **GD and Discrete gradient descent**:
  - E. Celledoni, S. Eidnes, B. Owren, T. Ringholm, *Dissipative numerical schemes on Riemannian manifolds with applications to gradient flows.* SISC. vol. 40 (6), 2018.
  - E. Celledoni and S. Fiori, *Neural learning by geometric integration of reduced rigid body equations*, J. CAM, 2004.
  - E. Celledoni, S. Fiori, *Descent methods for optimization on homogeneous manifolds*, MCS, 2008.
  - E. Riis PhD thesis on Discrete gradients and (randomised) Itoh-Abe for optimisation.
- **HD with symplectic methods on Riemannian manifolds**
  - B. Leimkuhler and G. W. Patrick, *A symplectic integrator for Riemannian manifolds*, JNS, 1996.
  - Literature on variational Lie group integrators and variational integrators on manifolds. RATTLE
  - Constrained Hamiltonian systems with dissipation.