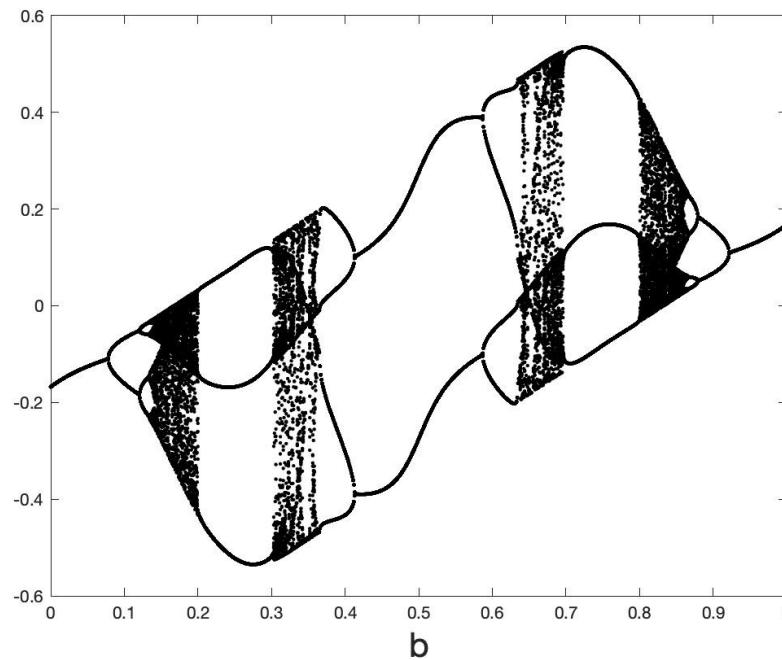


Lecture 7. Neural ODES for, and as, dynamical systems

Chris Budd OBE (Bath)
Seattle, June 2025



Some papers/books to look at

- Chen et. al., *Neural Ordinary Differential Equations*
- Matlab and Simulink, *Dynamical System Modelling Using Neural ODE*
- di Bernardo et. al., *Piecewise smooth dynamical systems*
- Brunton, Kutz et. al, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*
- Ashwin et. al. *RNNS*

Dynamical systems

Flows:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t, \theta)$$

Maps:

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \theta)$$

$x(0), x_0$ Given. Study the evolution from this state

Dynamical systems link to neural networks in a number of ways

- Solving a dynamical system using a PINN .. See lecture 3
- Constructing a dynamical system operator using a Neural Operator .. See lectures 5 and 6
- Approximating the **flow through a NN as a dynamical system** (Neural ODE 1)
[Chen et. al.]
and train that dynamical system
- Using a NN to approximate/learn the RHS of a dynamical system by

$$\frac{du}{dt} = f(t, u, \theta)$$

where f is the output of a NN and learn the parameters θ from data (Neural ODE 2)

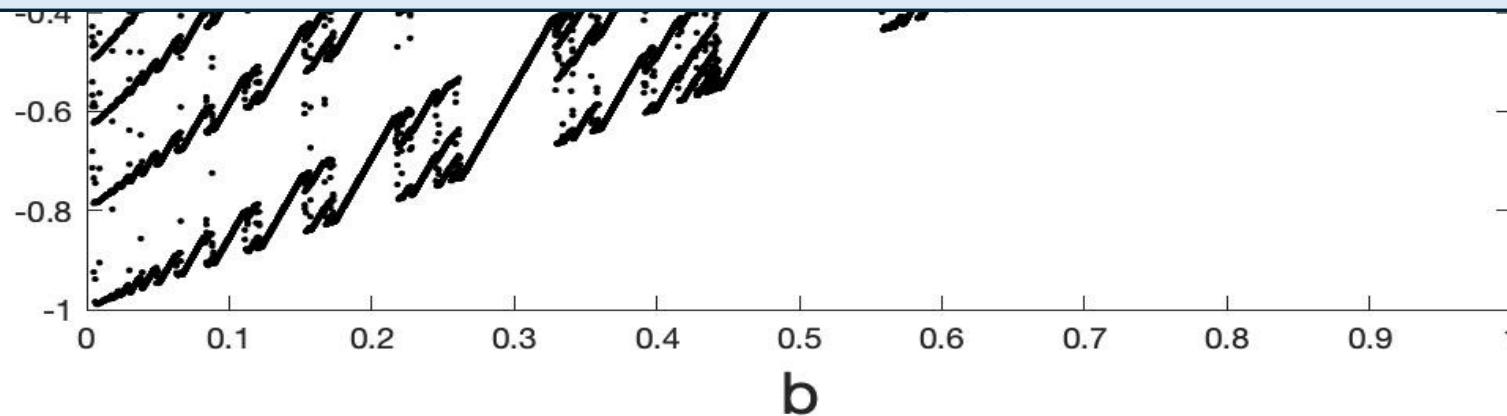
- Take f to be symbolic and learn it from data. SINDy [Brunton et. al.]
- **Predict the dynamics** of a dynamical system using a LSTM or similar



AIM:



Understand, and control, the **dynamical** behaviour and training of (feed forward) Neural Nets and Neural ODEs by using results from the theory of **dynamical systems (smooth and non-smooth)**



Basic properties of smooth dynamical systems e.g. $dx/dt = f(t,x)$

Evolve from initial state x_0

Transient motion followed (usually) by convergence to an omega-limit set ω

Omega limit sets ω depends on the parameters θ can be:

- Attracting Fixed Points. (FPs)
- Periodic orbits
- Heteroclinic connections
- Quasi-periodic orbits
- Chaotic strange attractors

Varying θ leads to smooth changes of the omega-limit set, and changes/transitions between different types of omega-limit set at bifurcations



[See Ashwin et. el apply this methodology to RNNs,ESNs,ENAs]



Examples of smooth bifurcations:

Saddle node and tipping points

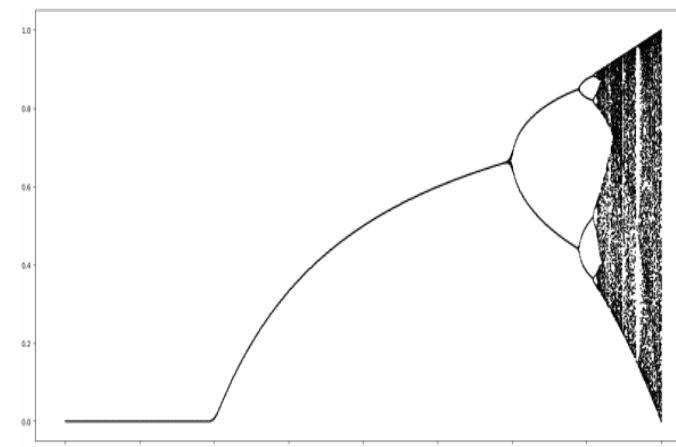
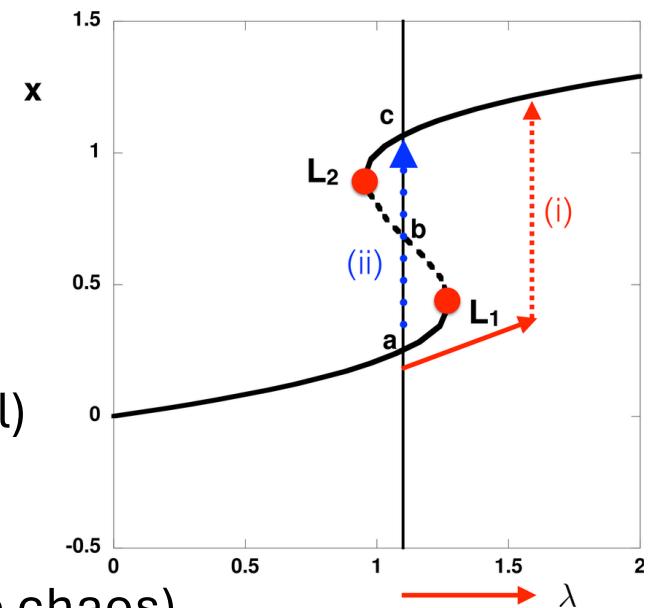
Transcritical and Pitchfork (super and sub critical)

Period-doubling (and period doubling cascade to chaos)

Hopf (super and sub critical)

Cyclic fold

Homoclinic/heteroclinic



NNs as Dynamical Systems

All feed forward NNs and RNNs can be thought of as maps:

Eg.

$$x_{n+1} = \sum_i c_i^n \sigma(A_i^n x^n + b_i^n)$$

Defines a **parametrized map**

NN: Input x_0 output x_N .

Deep if $N \gg 1$. Then we can study the **output of the NN as the omega limit set of the map**. Training then relates to how the omega limit set depends on the parameters eg. bifurcations etc.



Can be

Smooth

Or

Non-smooth

Depending on the activation function



Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) ^[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) ^[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

Discontinuous

Smoothed discontinuous

Discontinuous derivative

Discontinuous derivative

Lack of smoothness can lead to very rich dynamics

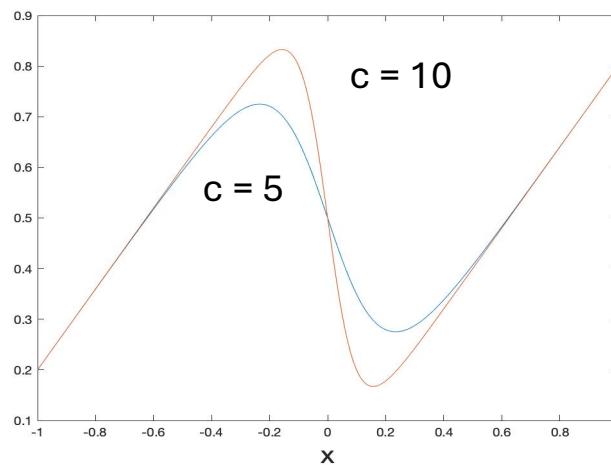
Partly explains the great expressivity of a deep neural network

Case Study : Feed forward NN: One-d smoothed sigmoid map

$$x_{n+1} = a x_n - H(x_n) + b, \quad H(x) = 1, x \geq 0, \quad H(x) = 0, x < 0$$

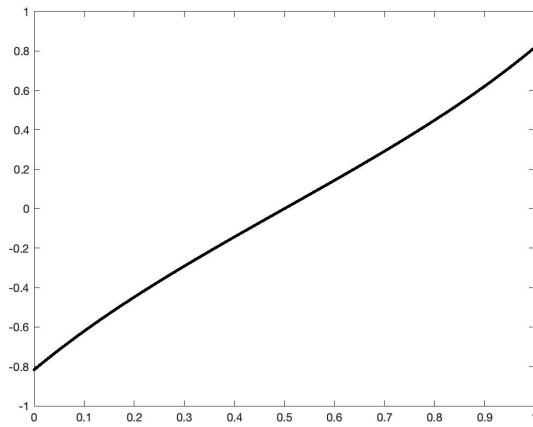
Smoothed to:

$$x_{n+1} = a x_n - (1 + \tanh(c x_n))/2 + b.$$

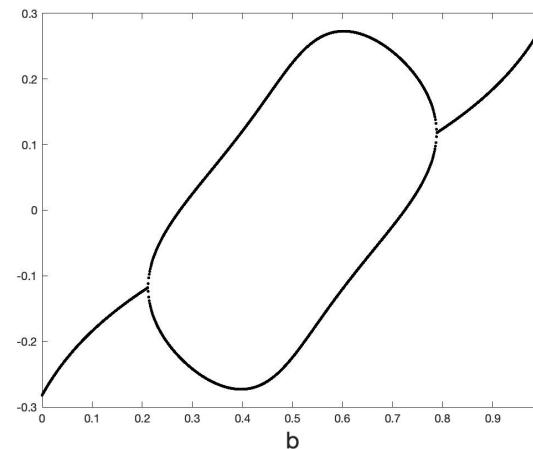


Omega-limit sets as a function of smoothing (when $a = 0.8$)

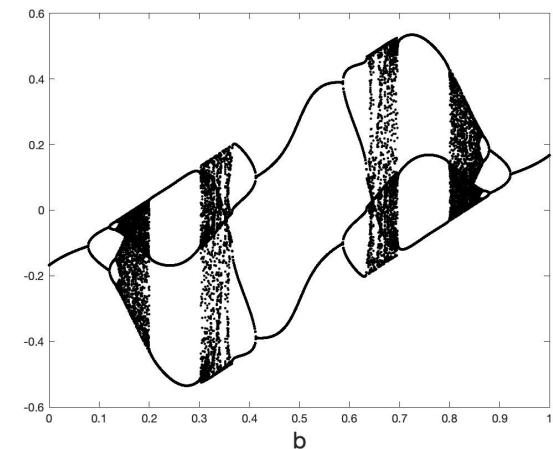
$c = 1$



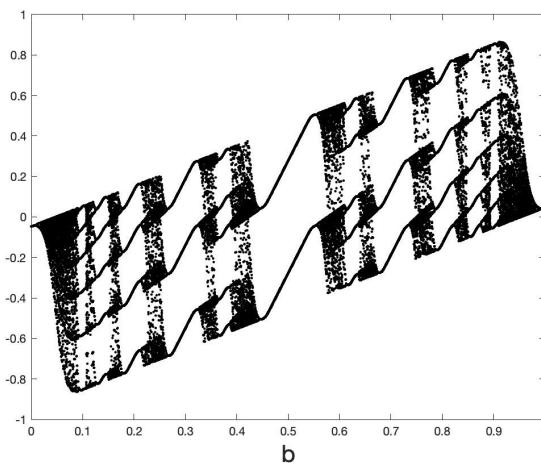
$c = 5$



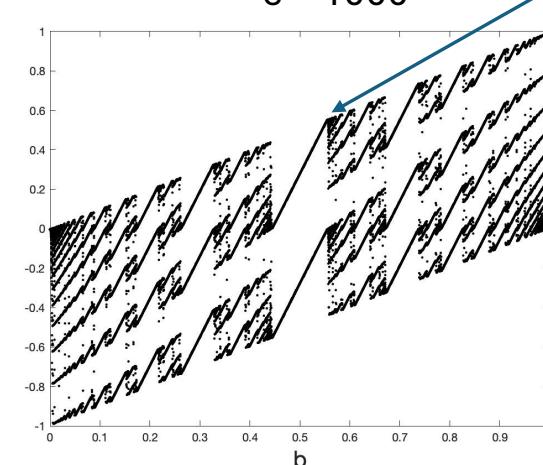
$c = 10$



$c = 50$



$c = 1000$



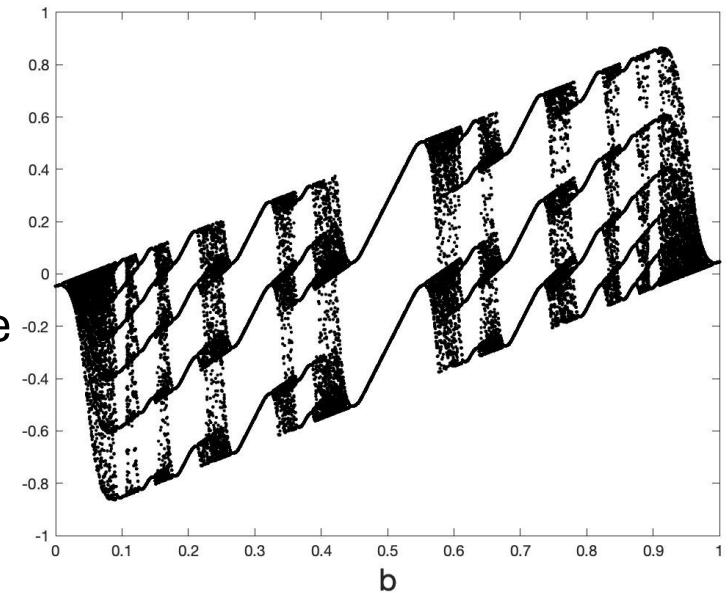
Border collision

Conclusions from the results

- High gradient of the tanh function leads to chaotic behaviour in certain cases. Hard to define a loss function and dependence of the loss function on the input/parameters in this case

Consider $\theta = \{a, b, c\}$ to be trainable parameters

- Only in the case of $c = 1$ is dependence on the training smooth and predictable using back propagation
- Sensitive parameter dependence can make training difficult close to bifurcation points b^* giving a loss function changing as $1/\sqrt{b - b^*}$ in the smoothed case and period adding at the border collision.



Neural ODES [Chen et. al] : The ResNET architecture as a flow

$$x_{n+1} = x_n + \Delta t \sum_i c_i^n \sigma(A_i^n x^n + b_i^n)$$

Let $t = n \Delta t$

$$\frac{x_{n+1} - x_n}{\Delta t} = \sum_i c_i(t) \sigma(A_i(t)x^n + b_i(t))$$

Let $\Delta t \rightarrow 0$

$$\frac{dx}{dt} = \sum_i c_i(t) \sigma(A_i(t)x^n + b_i(t))$$



Neural ODE. Input: $x(0)$. Output: $x(T)$, $T = n \Delta t$



- Idea [Chen et. al.]:
- Replace the NN by solving the Neural ODE using an ODE solver eg. RK4, Adam
- Training now becomes a question of finding the functions $A(t)$, $b(t)$, $c(t)$
- Advantages in memory efficiency and back propagation
- This a problem in optimal control theory

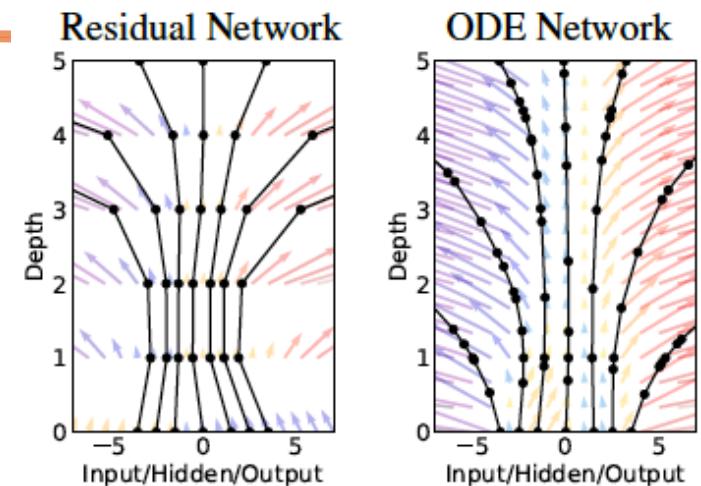


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Training such a ‘Neural ODE’: [Chen et. al 19], [Pontryagin 62]

Consider the parametrized Neural ODE

$$\frac{d\mathbf{h}}{dt} = f(\mathbf{h}(t), t, \theta), \quad \mathbf{h}(0) \rightarrow \mathbf{h}(T).$$

Compute solution using an ODE solver

Loss function $L(\cdot)$

$$L(\mathbf{z}(T)) = L \left(\mathbf{z}(0) + \int_0^T f(\mathbf{z}(t), t, \theta) dt \right)$$



To train the neural net to optimize L we need to find how it changes with θ

Results from classical control theory (and data assimilation) allow us to do this

Step 1

Define **adjoint**

$$\mathbf{a}(t) = \partial L / \partial \mathbf{z}(t) \quad \text{then}$$

$$\frac{d\mathbf{a}}{dt} = -\mathbf{a}^T \partial f / \partial \mathbf{z}$$

Compute $\mathbf{a}(t)$ using an ODE solver ‘backwards’ starting from $\mathbf{z}(T)$



Step 2

Compute the gradient of L by solving a third ODE

$$\frac{dL}{d\theta} = - \int_T^0 \mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt$$

Can then train θ to optimize L using gradient descent

[Chen et. al.] implement this using an adaptive Adam solver.

What can a Neural ODE be trained to do? **Homeomorphism Theorem**

Classical Neural ODEs take the form of

$$dh/dt = f(h, t, \theta) \quad \text{where } f \text{ is a Lipschitz function constant } L$$

Want to train this so that we have the map. $F: h(0) \longrightarrow h(T)$

Theorem *If f is Lipschitz than F is continuous*

Proof Let $F: u(0) \rightarrow u(T)$ and $F: v(0) \rightarrow v(T)$, $z(t) = u(t) - v(t)$

$dz/dt = f(u, t) - f(v, t) < |f(u, t) - f(v, t)| < L |u - v|$. Therefore



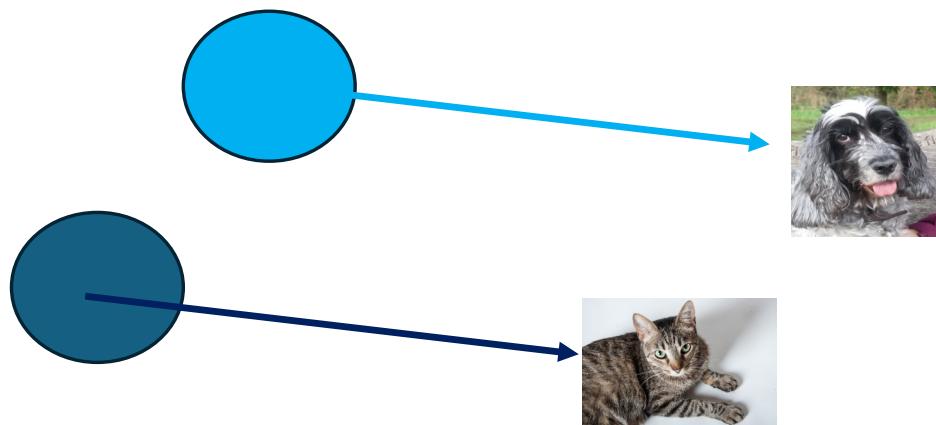
$$|u(T) - v(T)| < e^{Lt} |u(0) - v(0)|$$



Conclusion:

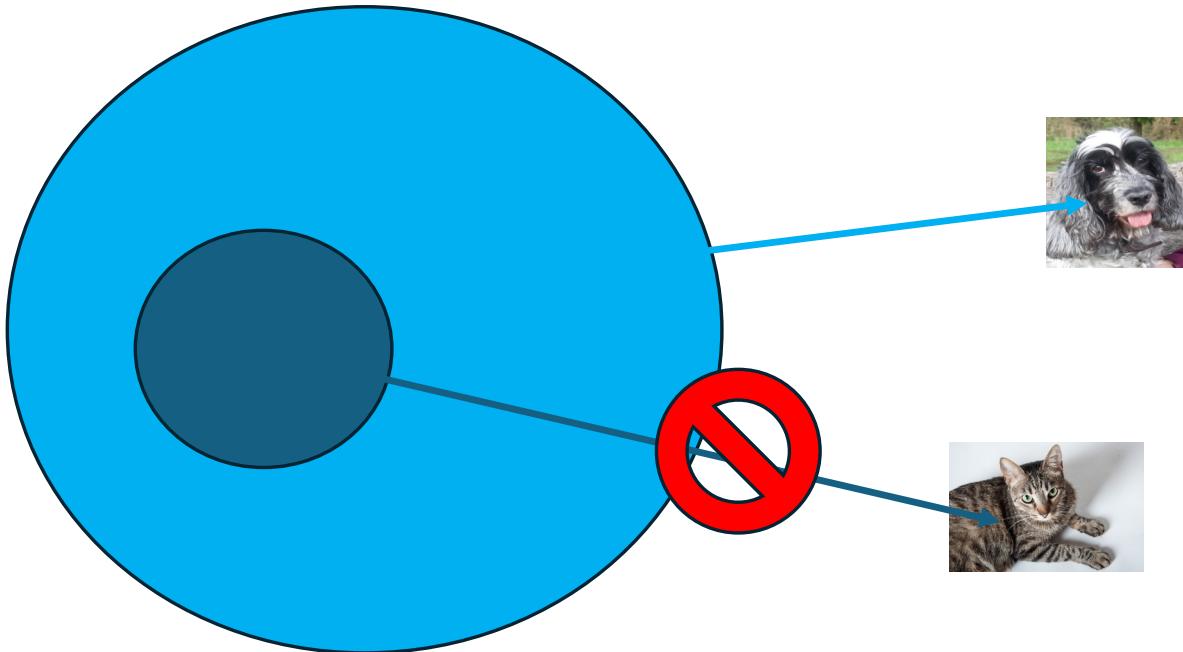
Topology of the input data is the topology of the output data

Example: Want to train a Neural ODE to separate data
eg. dogs and cats. [+Davide Murari]



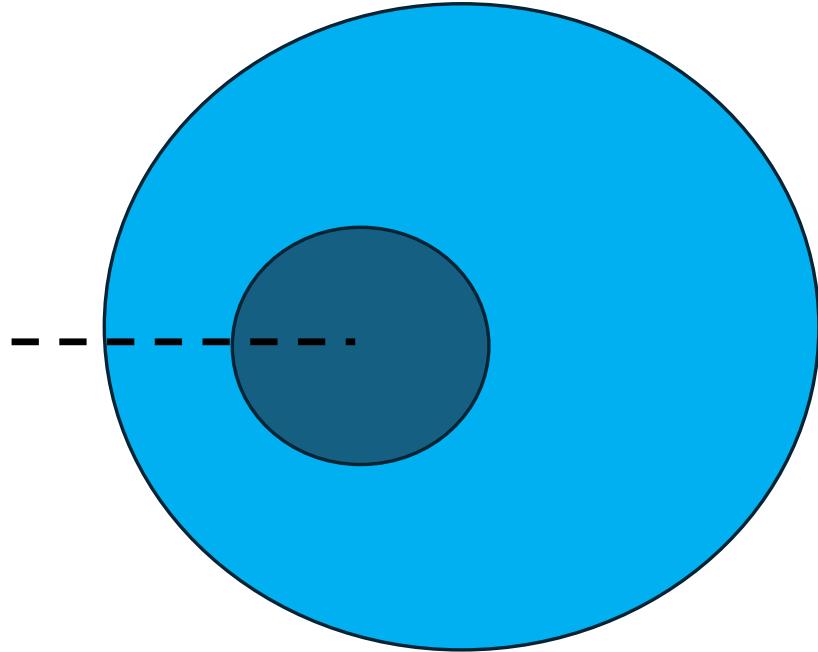
Can resolve this topology
with a Neural ODE

But this data topology cannot be resolved into two points using a smooth Neural ODE



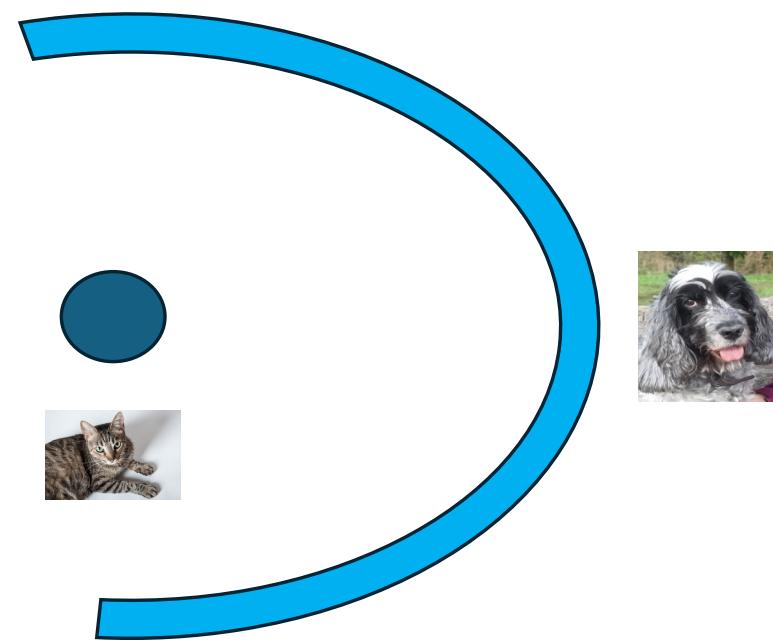
Have to either use a MAP or a non-Lipshitz Neural ODE

This data topology **CAN** be resolved using a non-smooth Neural ODE



$$r < 1 : \frac{dr}{dt} = -r, \quad \frac{d\theta}{dt} = 0, \quad r \geq 1 : \frac{dr}{dt} = r(r^* - r), \quad \frac{d\theta}{dt} = -\theta.$$

Discontinuous flow breaks the topology and allows data separation



Question: How do we train these systems?

Dynamical systems defined by Neural Nets and learning dynamics from data

Original DS:

$$\frac{dx}{dt} = f(x) \quad x \in R^n$$

Data points:

$$y_i = H(x(t_i)) + \epsilon_i, \quad i = 1 \dots N$$

Train: a NN to approximate this dynamics via:

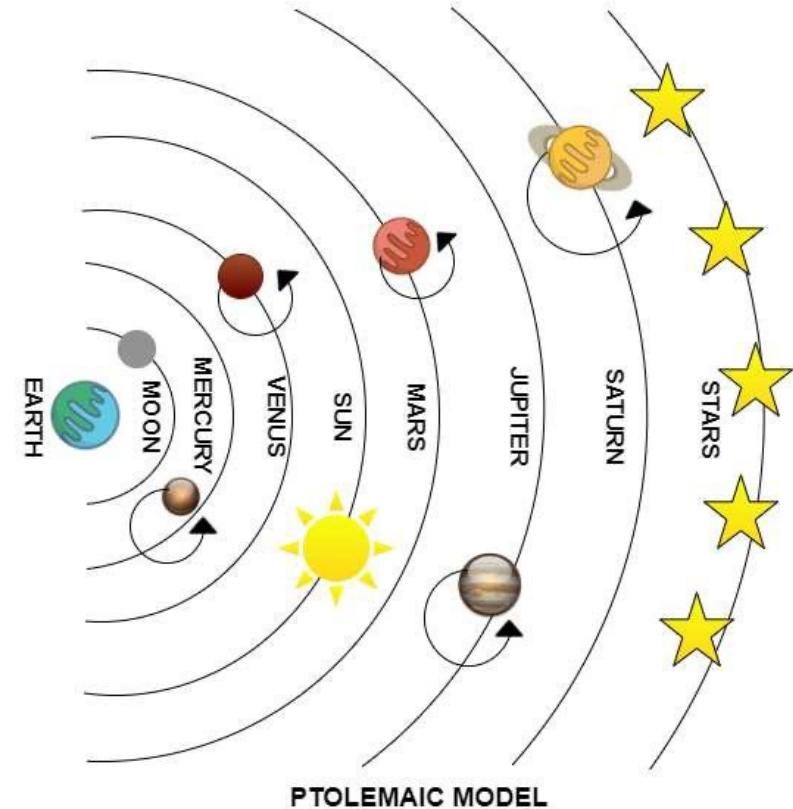
$$\frac{dz}{dt} = f(z, t, \theta) : \quad f: \text{Neural Net}$$

Also (and confusingly) called a Neural ODE eg. by Matlab

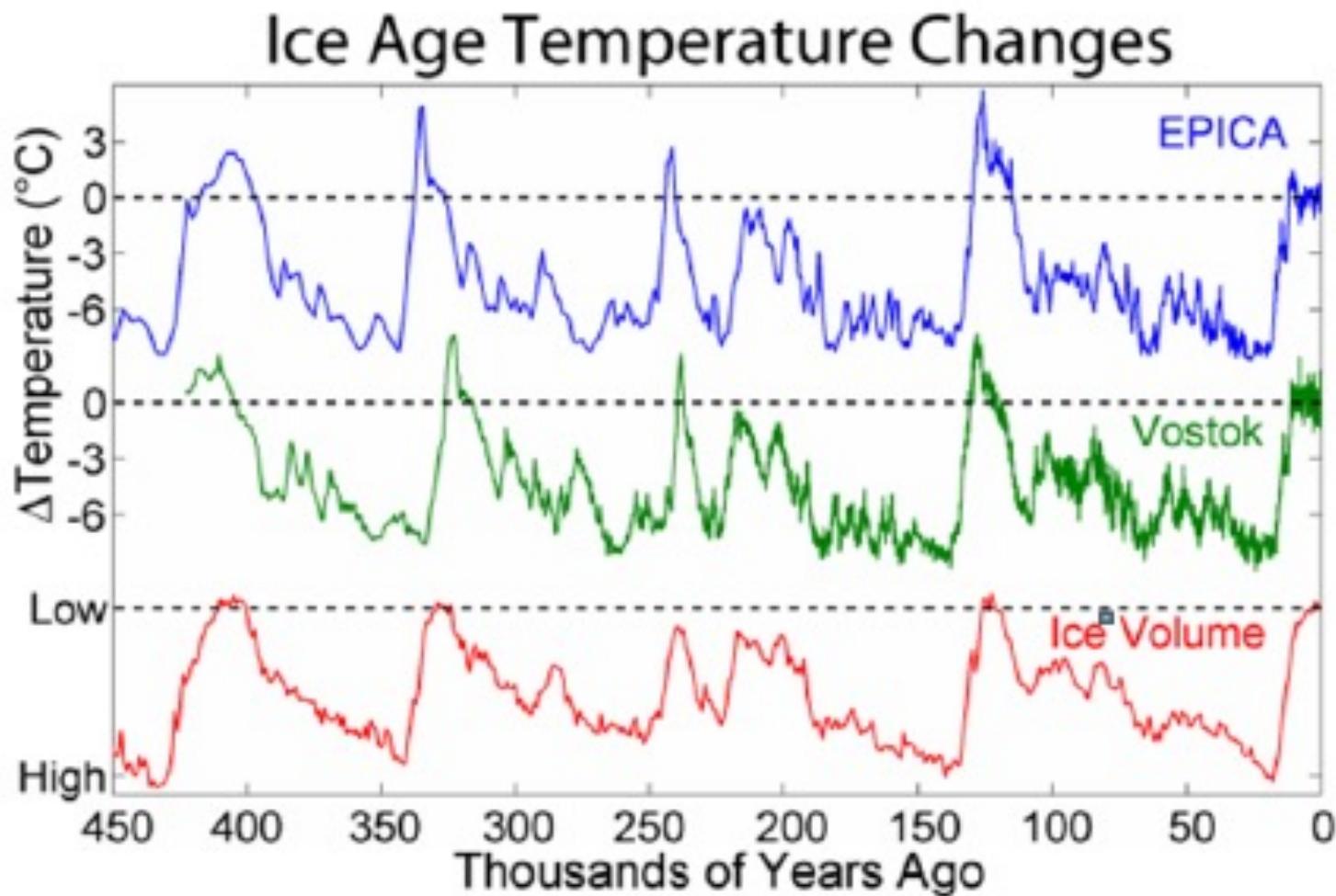


Big literature on these systems

- Ptolemaic model of the solar system
- Standard learning
- Echo State Networks, Reservoir networks
Especially good for chaotic dynamics
- SINDy approach (Kutze) using a very careful choice of RHS (can even work from video data)



Q. What caused the ice ages?



Training such a ‘Neural ODE’. I [From MatLab website]

Example:

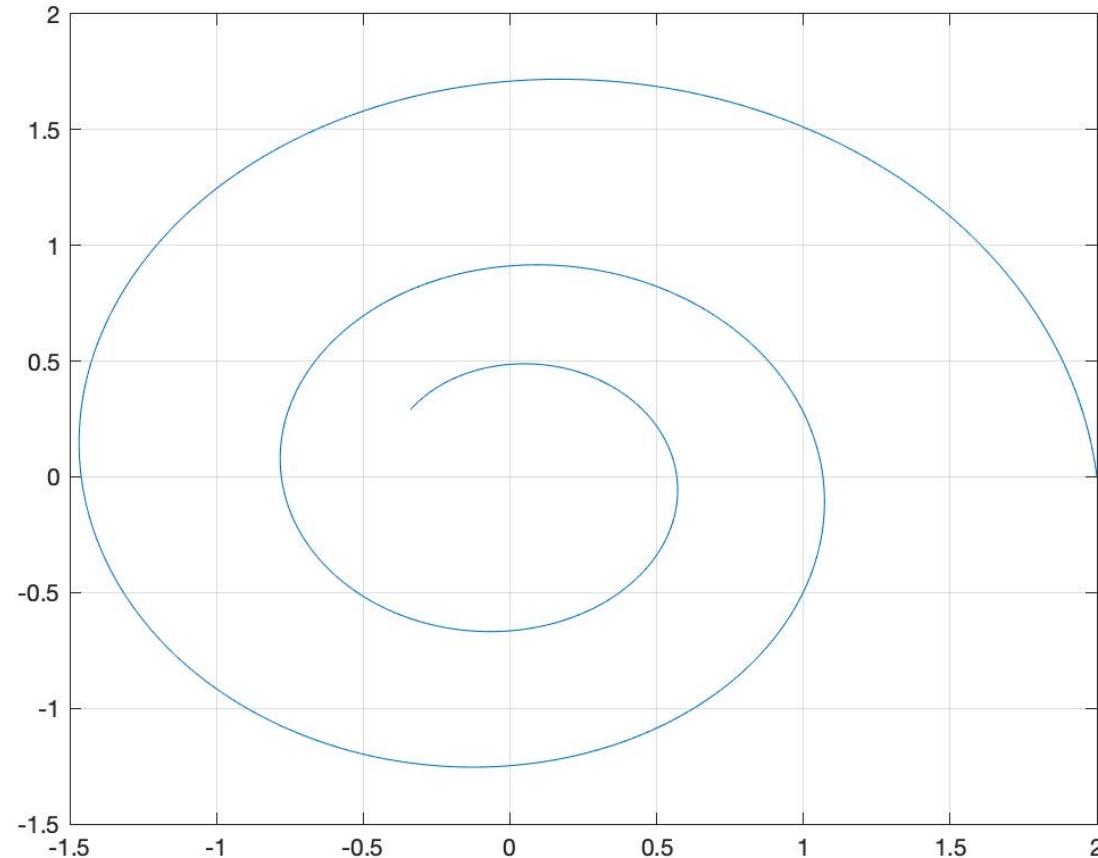
$$\frac{d\mathbf{x}}{dt} = A\mathbf{x}, \quad \mathbf{x} \in R^2$$

$$\mathbf{x}(t) = e^{At} \mathbf{x}(0) \equiv B\mathbf{x}(0).$$

Neural ODE: $\frac{dz}{dt} = f(z, t, \theta), \quad z(0) = x(0), \quad f: \text{Neural Net}$

Solve: using RK45

Training trajectory starting from: $x(0) = [2,0]$, $t = [0,15]$, 2000 points



$$A = \begin{pmatrix} 0.1 & -1 \\ 1 & -0.1 \end{pmatrix}$$

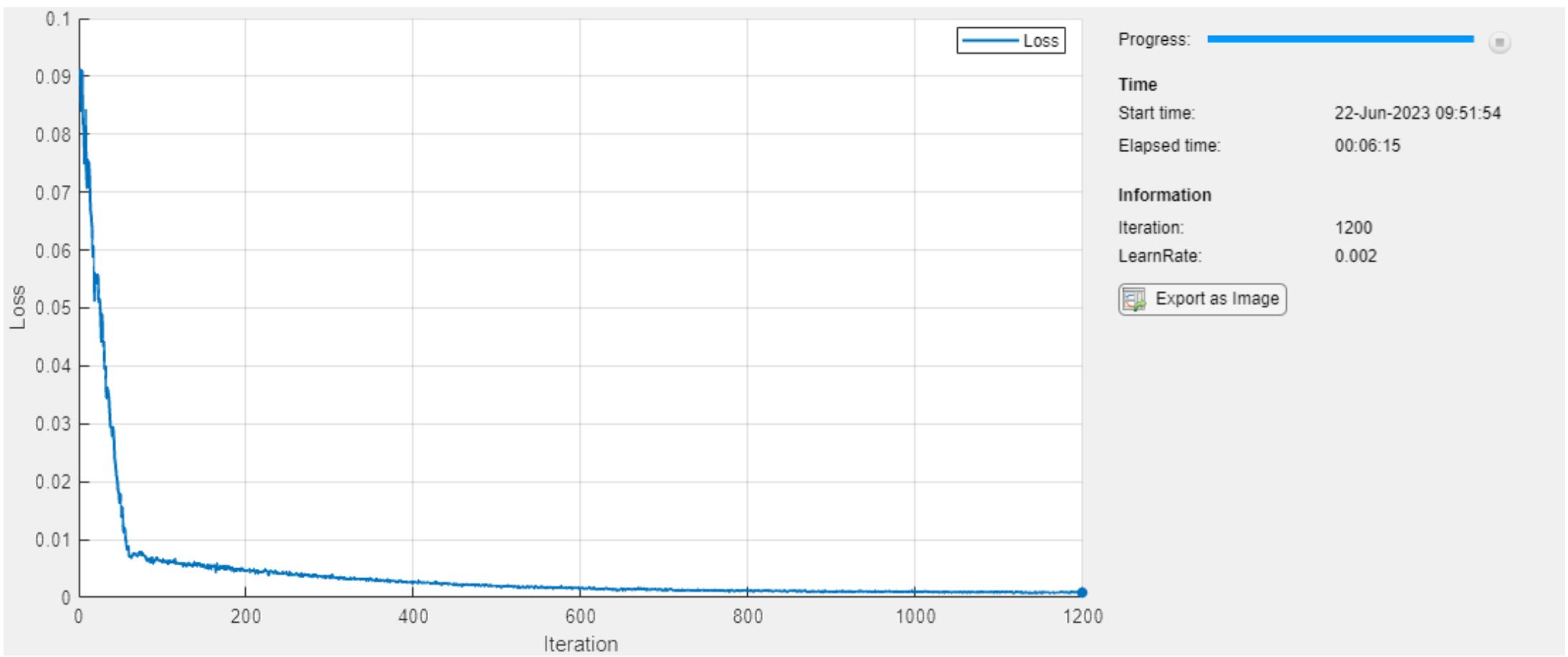
Define loss: $L(\theta)$

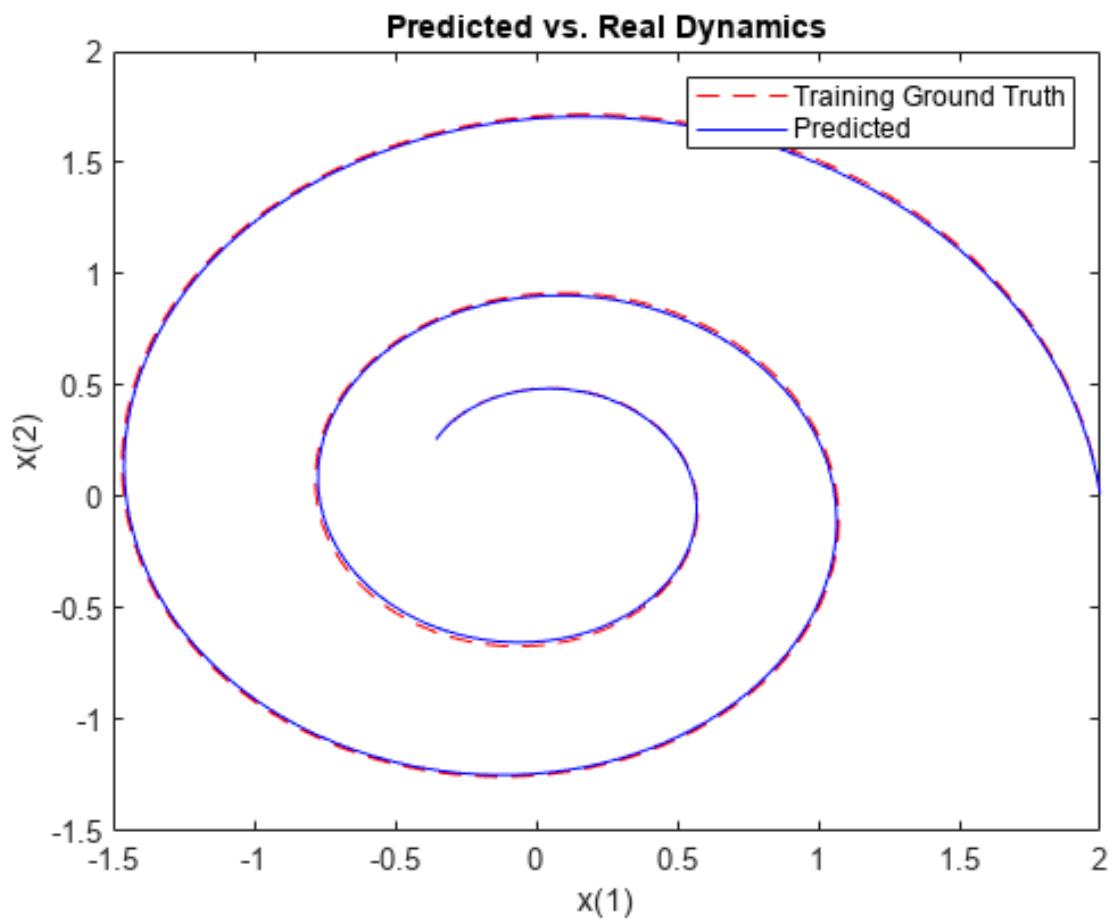
$$L(\theta) = \sum_i |H(z(t_i)) - y_i|^2$$

- Implement NN using 2 layers and tanh activation

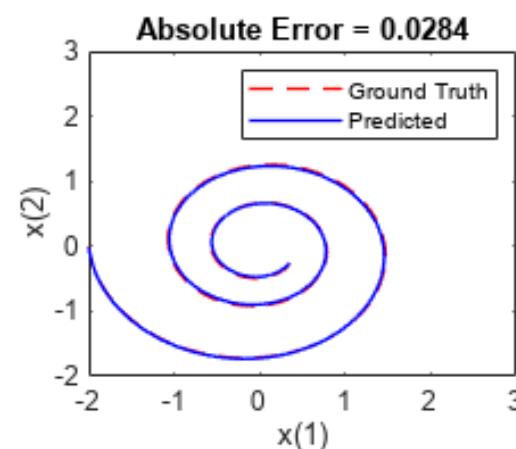
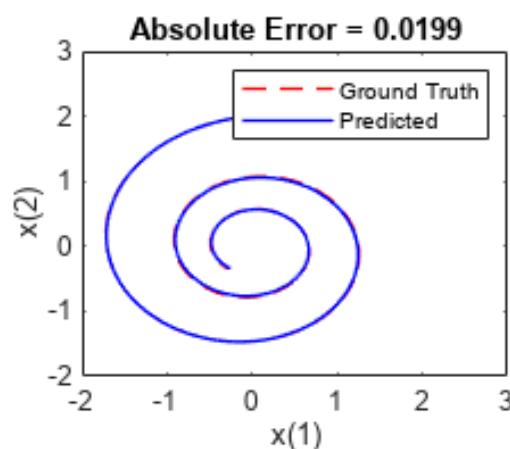
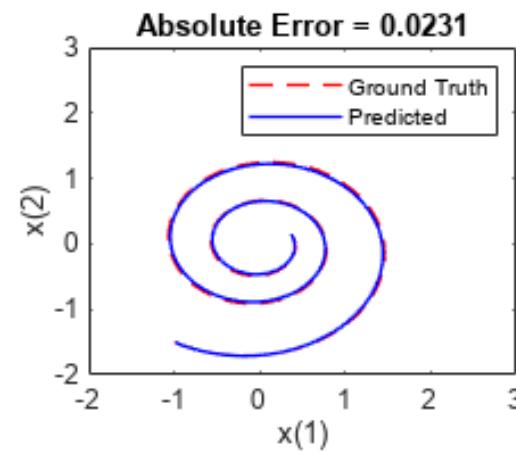
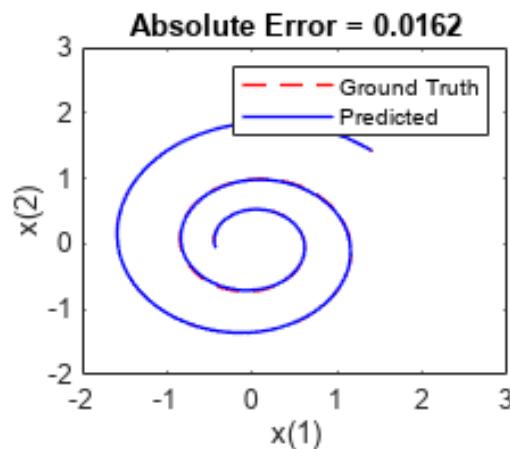
$$f(\mathbf{z}, \theta) = W_2 \tanh(W_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2.$$

- Compute gradients directly from the ODE solver using the adjoint method as before
- Optimize using ADAM using first 400 training points

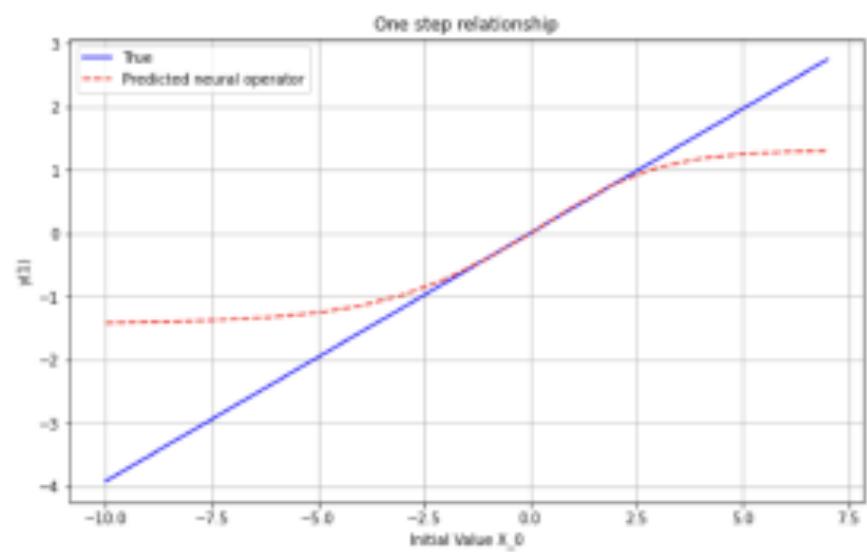
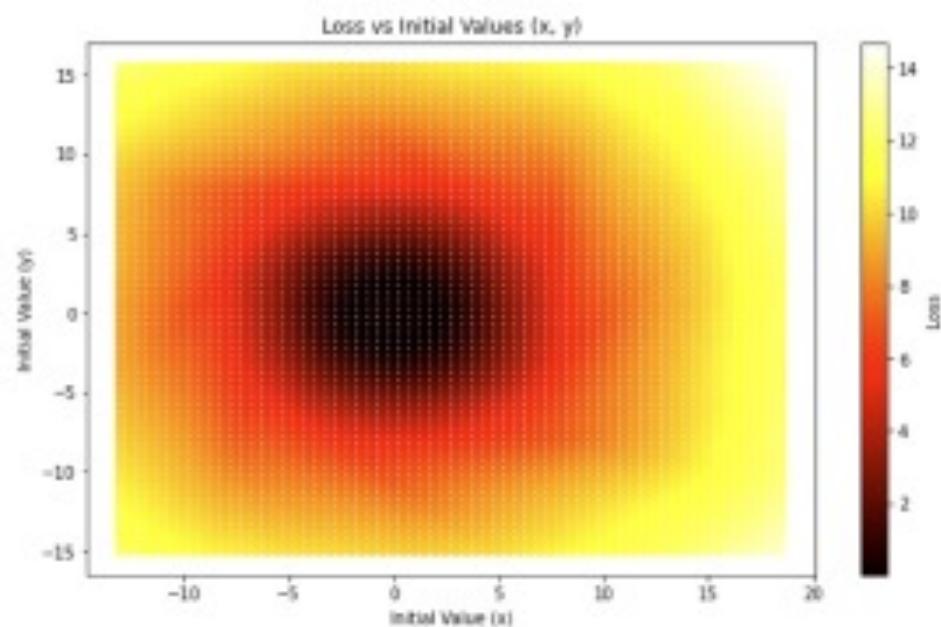




Prediction error is good for initial data close to the training trajectory



Loss increases as we move away from the training trajectory [+Libo Chen]



By way of contrast:

Assume that

$$\frac{d\mathbf{z}}{dt} = B \mathbf{z}$$

Find matrix B

100 points:

$$B = \begin{pmatrix} -0.1001 & -1.0005 \\ 1.0004 & -0.0993 \end{pmatrix}$$

400 points:

$$B = \begin{pmatrix} -0.1000 & -1.0005 \\ 1.0007 & -0.1002 \end{pmatrix}$$

2000 points:

$$B = \begin{pmatrix} -0.1000 & -1.0005 \\ 1.0005 & -0.1001 \end{pmatrix}$$



SINDy [Brunton et. al.]

ODE: $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$

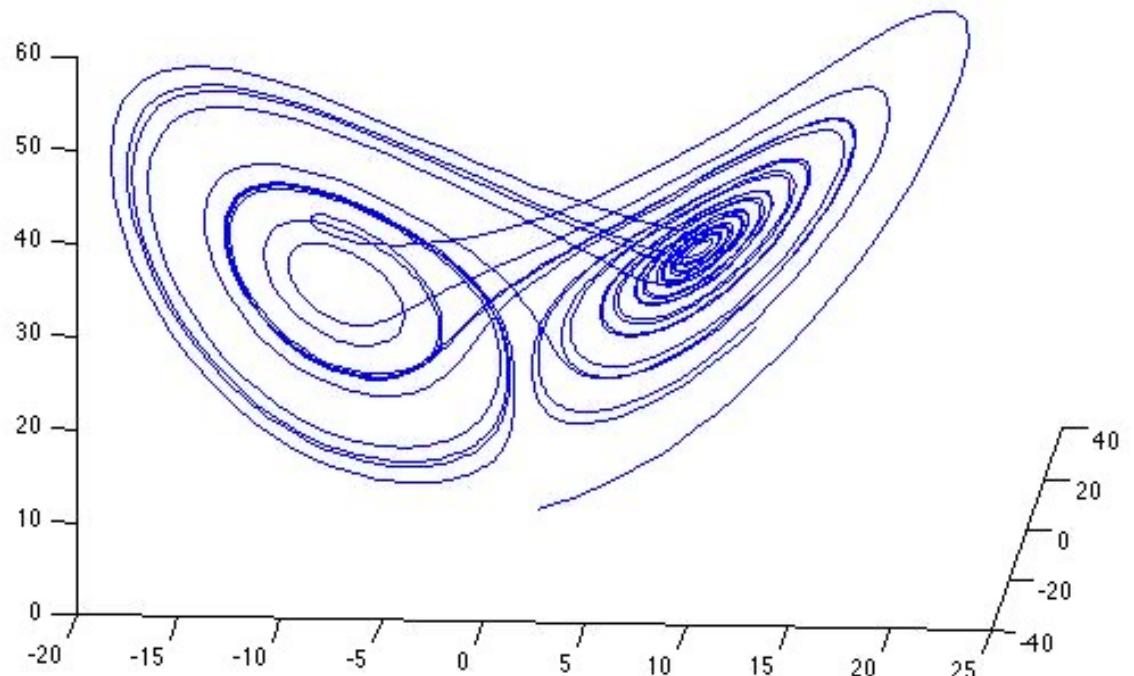
Data: $\mathbf{x}(t_i), \quad \dot{\mathbf{x}}(t_i)$ Plus noise: m measurements

Assume: $f(x)$ can be constructed from a library of known functions

Eg. Lorenz Equations:

r=28, b=8/3

$$\begin{aligned} dx/dt &= \sigma(y - x), \\ dy/dt &= x(\rho - z) - y, \\ dz/dt &= xy - \beta z. \end{aligned}$$



Library: $\Theta(\mathbf{x}) = [\mathbf{1} \mid \mathbf{x} \mid \mathbf{x}^{P_2} \mid \mathbf{x}^{P_3} \mid \sin(\mathbf{x}) \mid \cos(\mathbf{x}) \mid \dots]$

Size. $m \times p$ $m \gg p$

Coefficients (sparse vectors): $\Xi = [\xi_1 \ \xi_2 \ \dots \ \xi_n]$

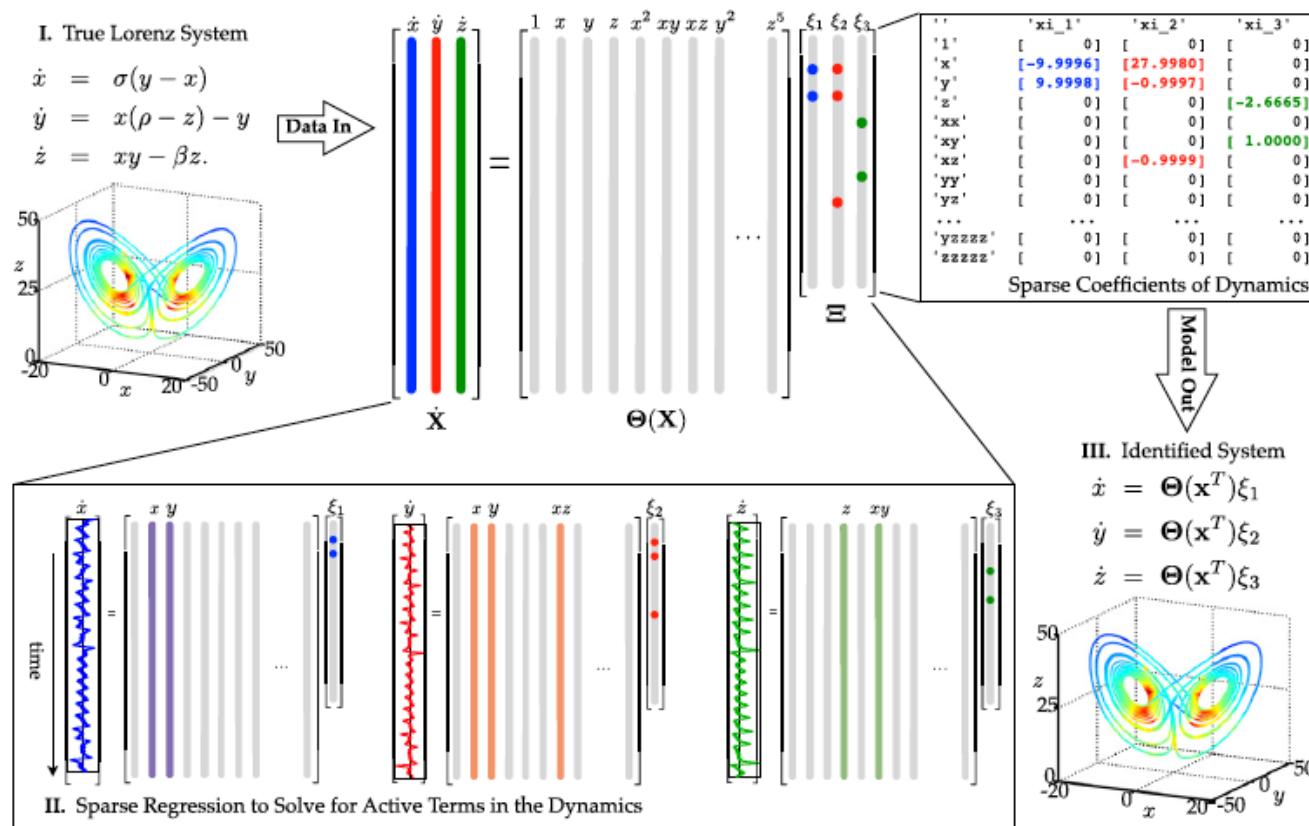
Seek a SPARSE approximation of the form:

$$\dot{\mathbf{x}} = \Theta(\mathbf{x}) \Xi$$



Aim: Find best set Ξ to fit to (filtered, noisy) data

Possibly with numerically estimated derivatives



From Brunton
et. el.

Possible dynamics of Neural ODEs

$$\frac{dx}{dt} = f(x, \theta)$$

- f(.) output of a feed-forward Neural Network
- f(.) Can be piecewise smooth, continuous or even discontinuous

Non-smooth nature of the dynamics places a significant constraint on the possible dynamics of the neural ODE, and helps understand what dynamics might be possible



For example, NN naturally lead to:

Filippov Flows

$$f(x, \theta) = f_i(x, \theta), \quad x \in S_i,$$

$$\bar{S}_i \cap \bar{S}_j \equiv \Sigma_{ij}$$

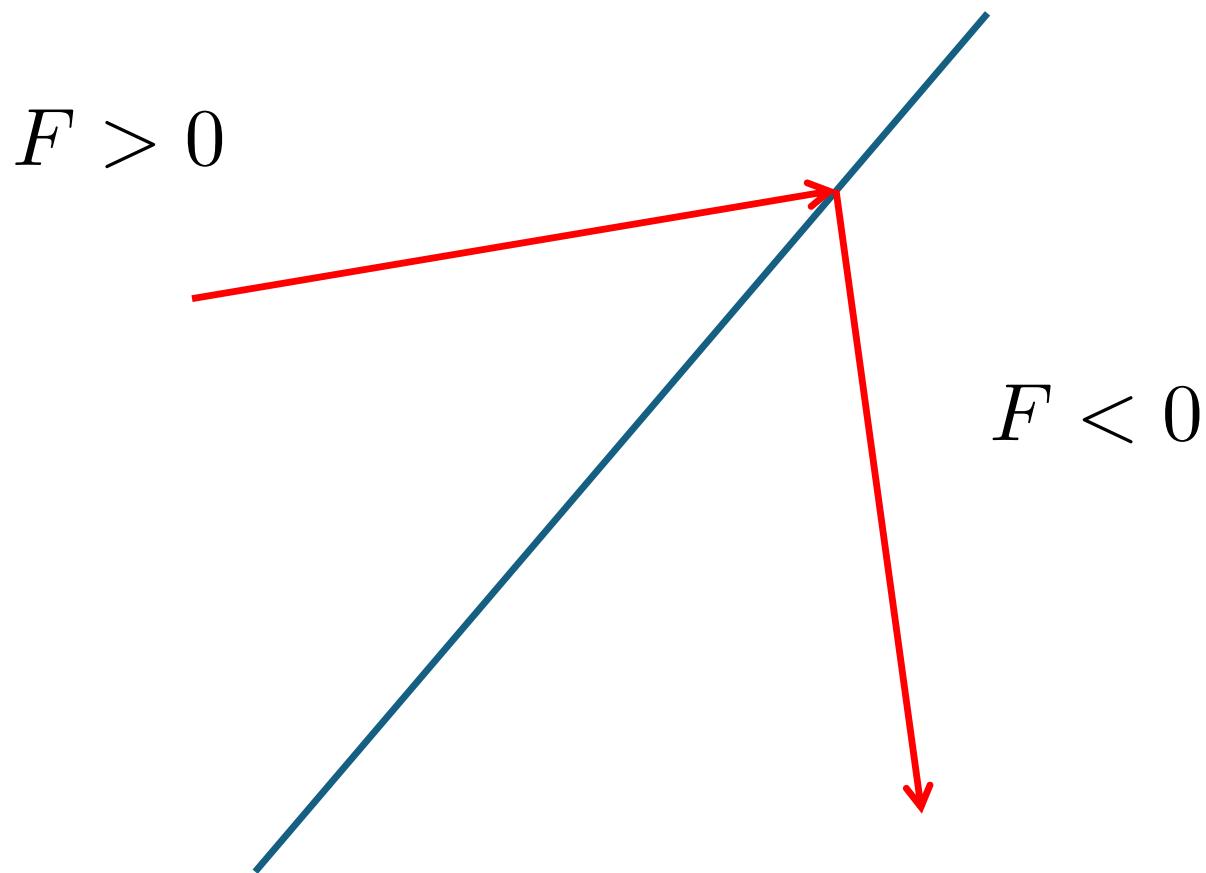
Smooth flow in sets S_i

Transitions across boundaries Σ_{ij}

Possible **sliding** along Σ_{ij} if **f is discontinuous** across the boundary

Filippov dynamical system without sliding

$$\Sigma = \{\mathbf{X} : F(\mathbf{X}) = 0\}$$



Typical ‘transverse’ intersection

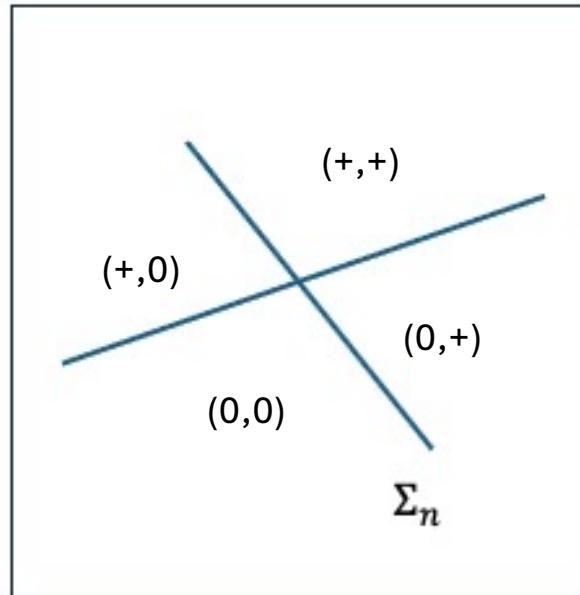
Features of non-smooth flows

- Discontinuity induced bifurcations [di Bernardo et. al.]
 - Border collision bifurcations when a fixed point intersects Σ_{ij}
 - Non-smooth folds
 - Grazing bifurcations when a flow has a non-transverse intersection with Σ_{ij}
- Period-adding and period-incrementing routes to chaos
- Robust Chaotic Behaviour is very common due to the stretching of phase space by the discontinuities/non-smooth effects [Banerjee, Hogan]. Sensitivity to initial conditions and parameters.

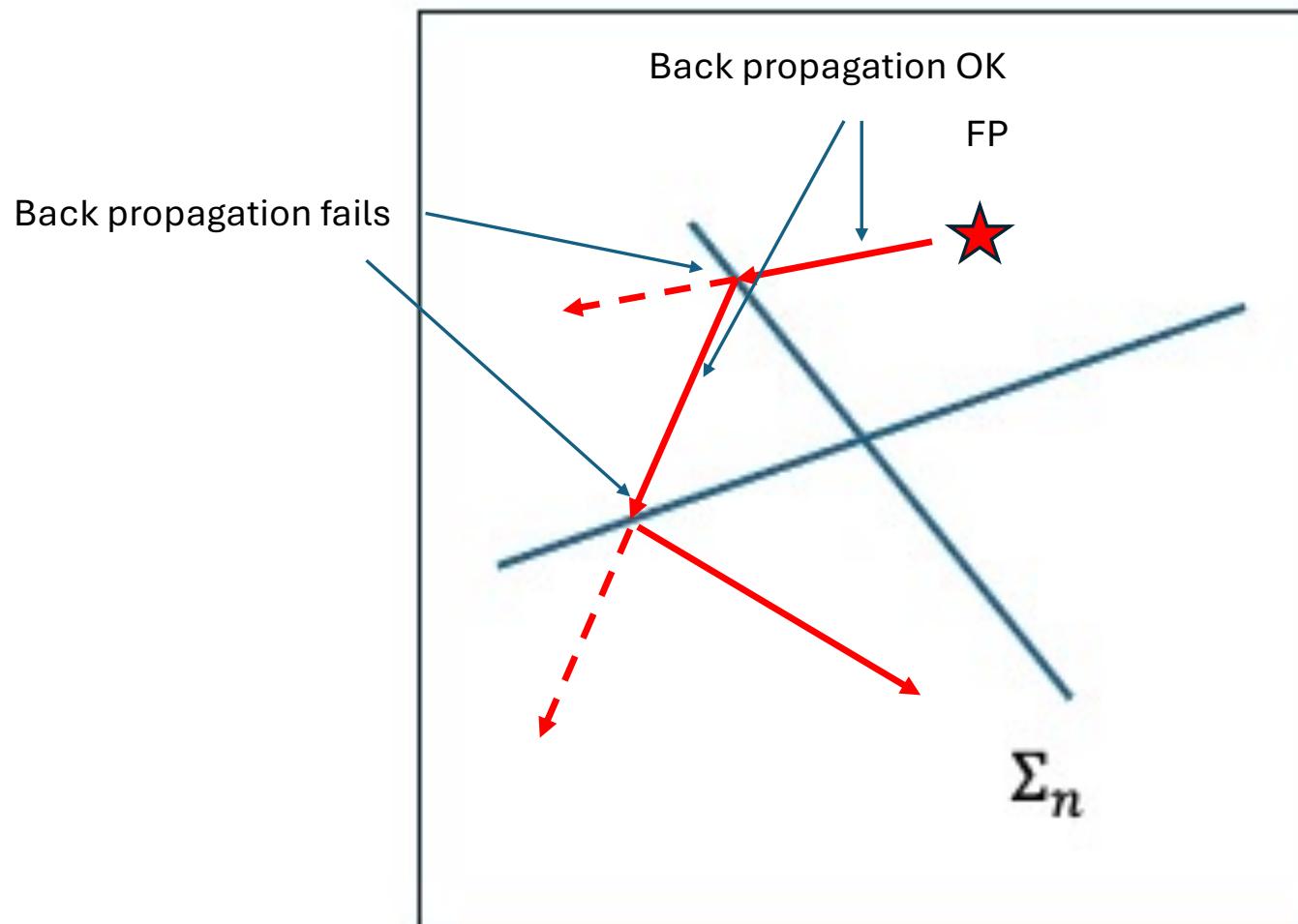


Example: Fixed point, Limit Cycle, and Chaotic Dynamics in ReLU type ODEs

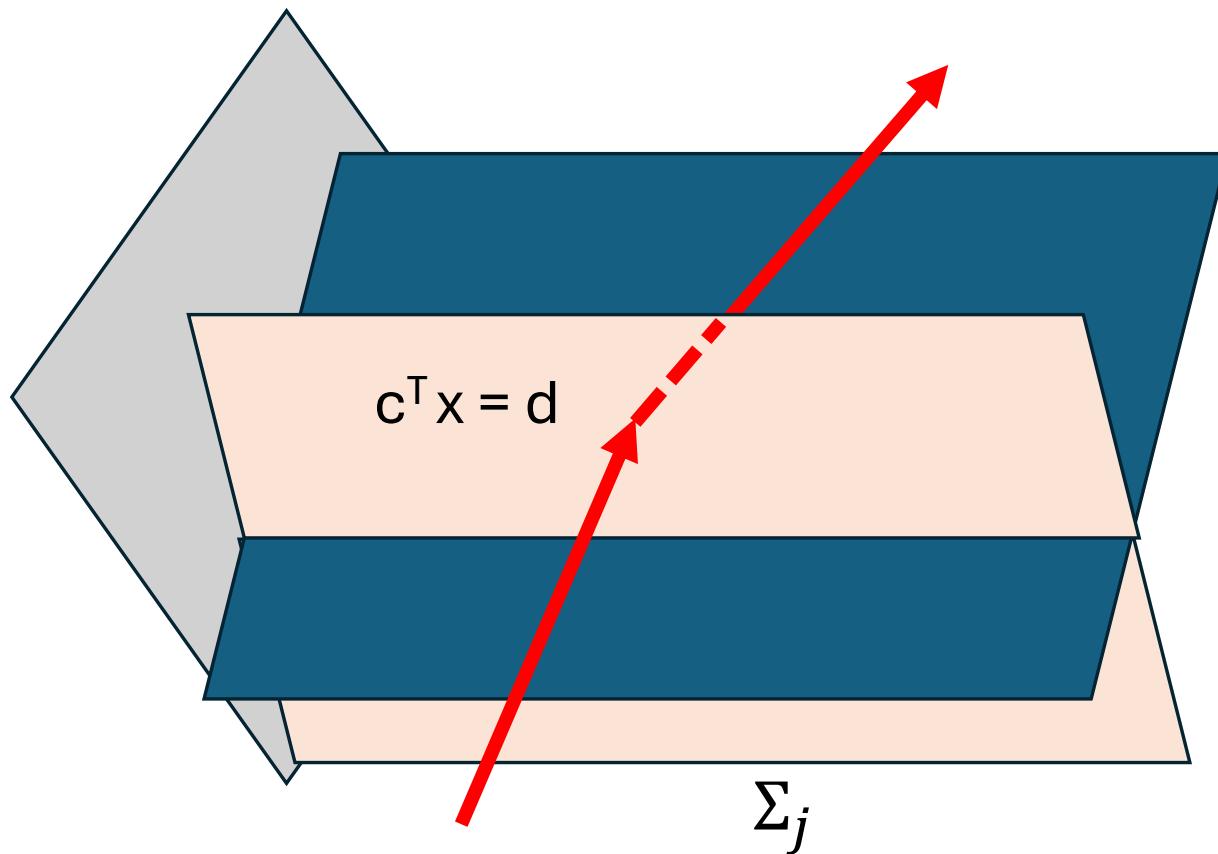
$$\frac{dx}{dt} = A_1 x + b_1 + \text{ReLU}(A_2 x + b_2)$$



Evolution of a fixed point. (2-dimensions)



Local behaviour (n – dimensions) when back propagation fails



General case: Normal form at the transition

All ReLU type dynamics can be expressed locally **close to the switch manifold** as:

$$\begin{aligned}\frac{dx}{dt} &= A_1 x + \mu b & \text{if } c^T x < 0, \\ \frac{dx}{dt} &= A_2 x + \mu b & \text{if } c^T x \geq 0.\end{aligned}$$

Possible scenarios:

- FP \rightarrow FP
- FP + FP \rightarrow \emptyset
- FP \rightarrow Limit cycle
- FP \rightarrow Chaos

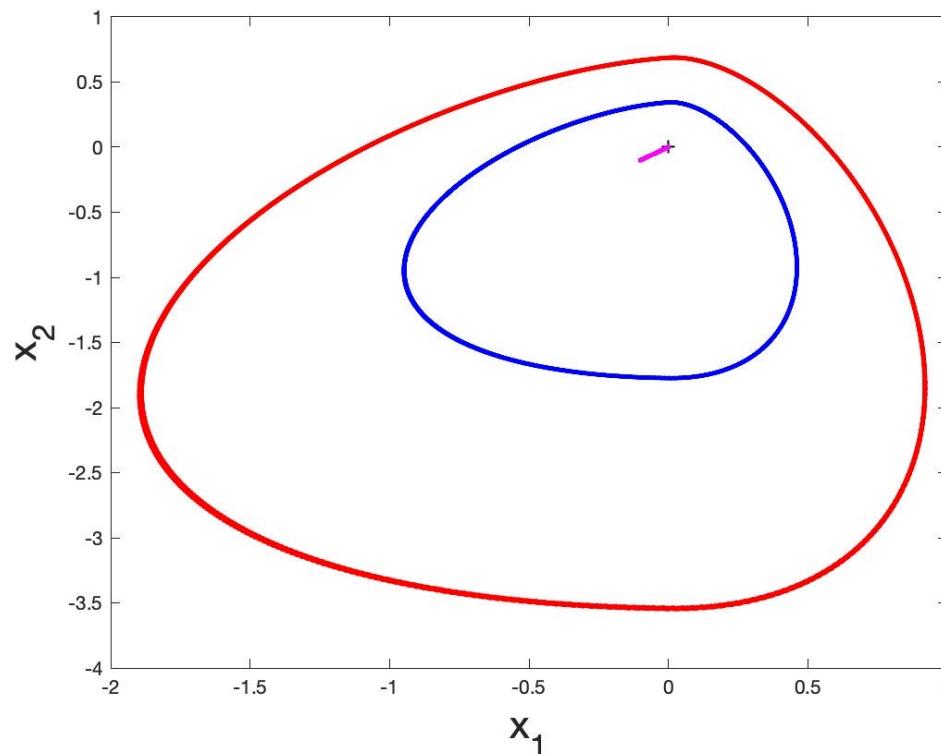
Can classify many of the transitions in terms of the structure of A_i and b .

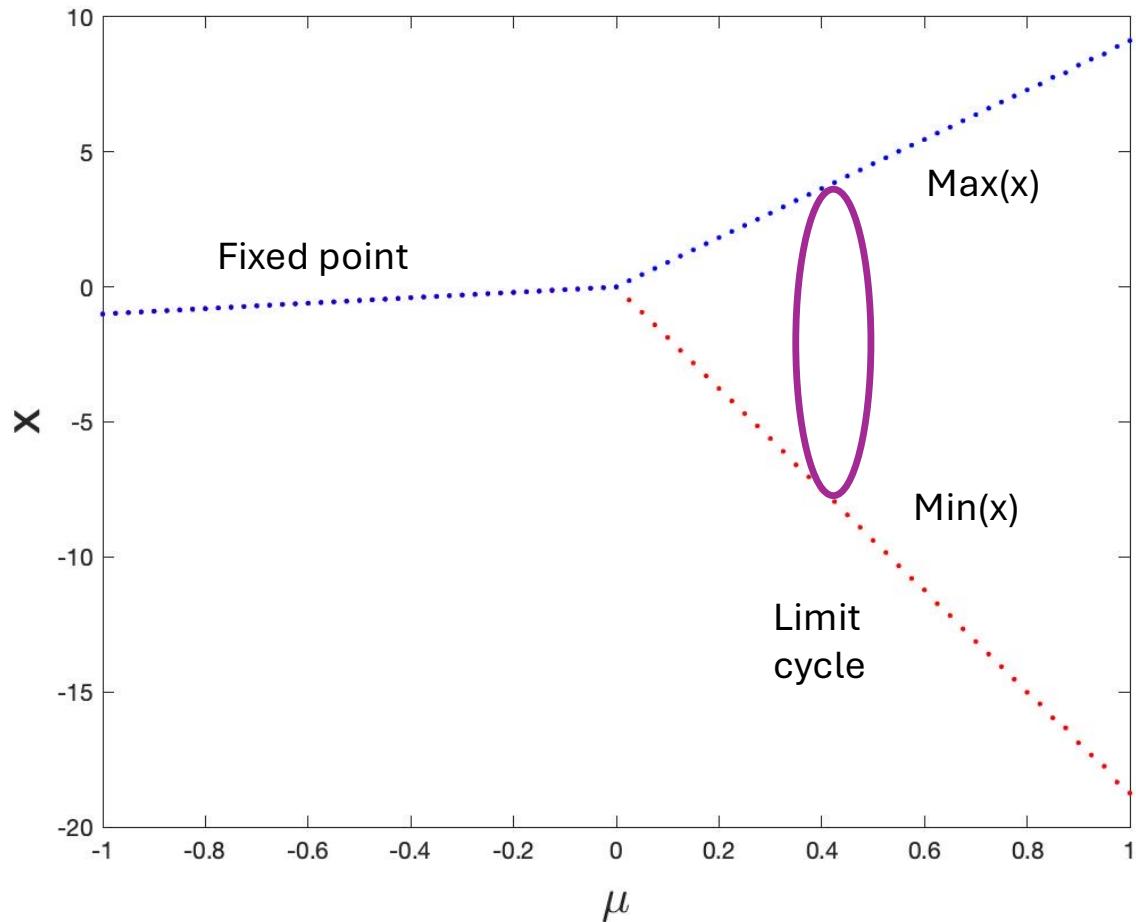
[Freire], [Carmona]



Example: FP → Limit Cycle

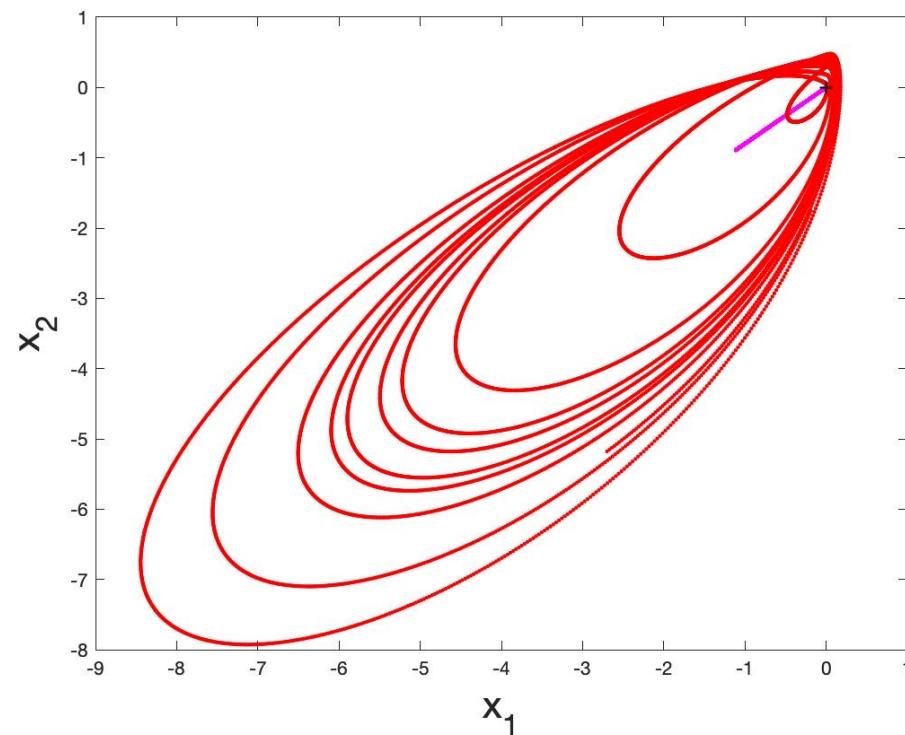
$$A_1 = \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 2 & 1 \\ -5 & 0 \end{pmatrix}, \quad b = (0 \ 1)^T, \quad c = (1 \ 0)^T$$

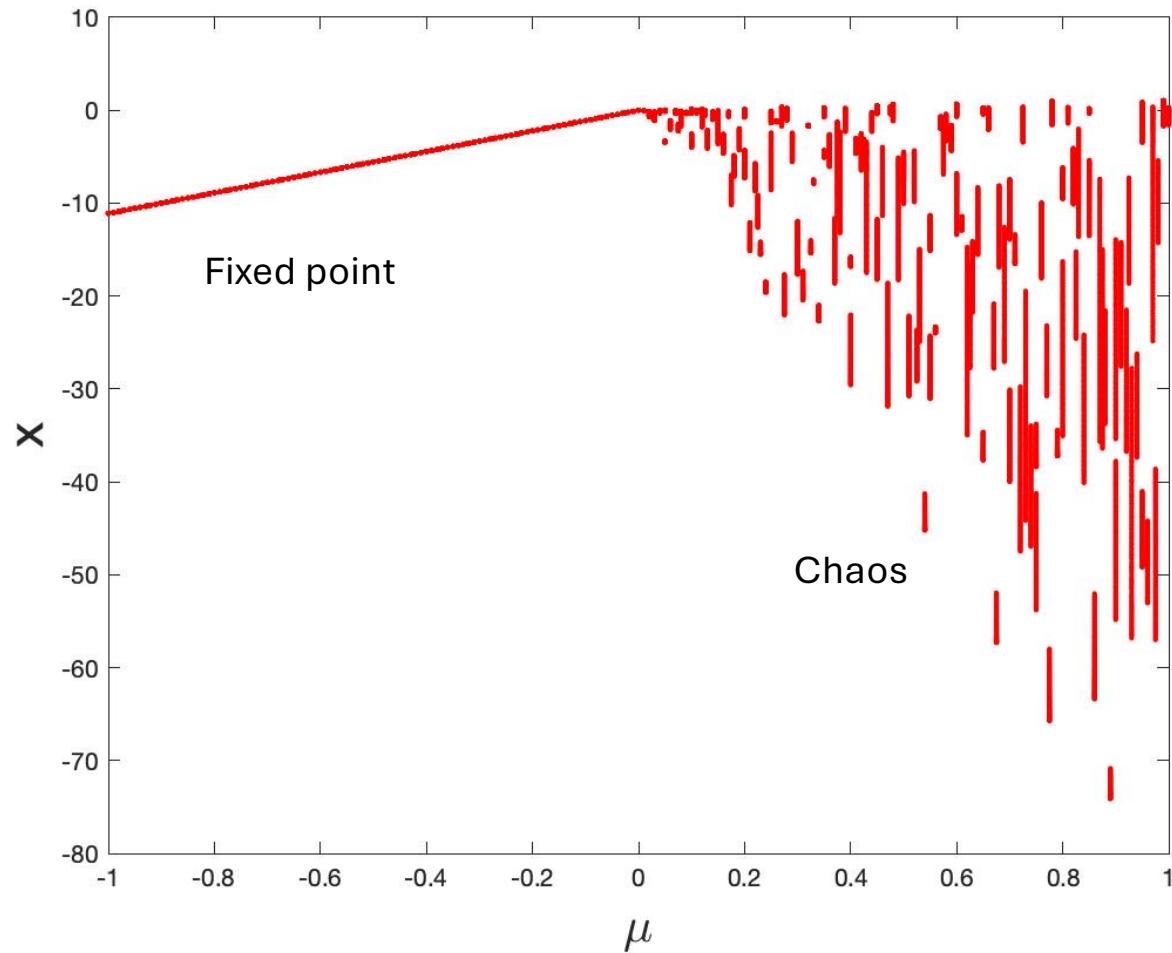




Example 2: FP → Chaos

$$A_1 = \begin{pmatrix} 0.8 & 1 & 0 \\ 0.57 & 0 & 1 \\ -0.09 & 0 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0.1 & 1 & 0 \\ -0.2 & 0 & 1 \\ -60 & 0 & 0 \end{pmatrix}, \quad b = (0 \ 0 \ 1)^T, \quad c = (1 \ 0 \ 0)^T$$





Conclusions

Neural ODEs are effective for

- Describing the flow through a NN
- Learning dynamics from data
- They inherently non smooth (or close to non smooth)
- They work best when informed by the physics

