Advanced Course on Deep Learning and Geophysical Dynamics

Learning and dynamical systems

Said Ouala

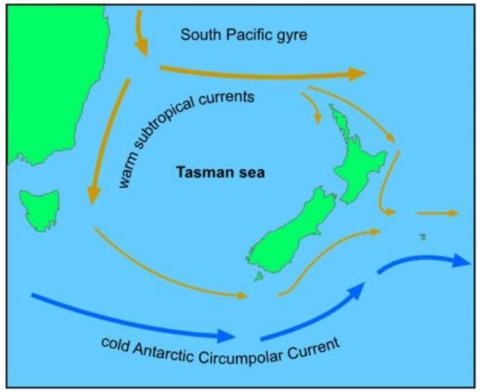
Outline

1. An naive, brief introduction to Dynamical Systems
   - Introduction
   - State space models and Learning formulation
2. Resolution of differential equations : numerical integration
3. Training dynamical systems
   - Continuous time setting
   - Discrete time setting
4. Partial observations of the state space
   - Phase space reconstruction
   - Examples
5. Model evaluation
   - How do we compare data-driven models ?
   - Prediction/forecast vs simulation applications
   - Limit-sets and evaluation metrics
6. Physics informed AI
   - Physics Informed Neural Networks (PINN)
   - Neural networks for closure modeling

An naive, brief introduction to Dynamical Systems, definition

What is a **dynamical system** ?

Dynamical systems are systems that change over time according to a set of relations.



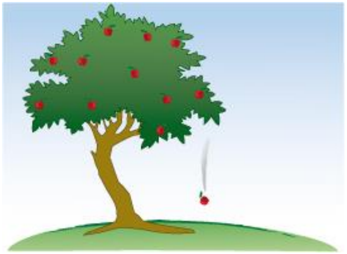
An naive, brief introduction to Dynamical Systems, traditionally

How to derive a **model** for a **dynamical system** ?

An naive, brief introduction to Dynamical Systems

How to derive a **model** for a **dynamical system** ?

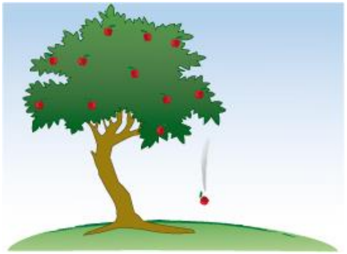
- Step 1 : Which phenomenon to model ? « x_t »



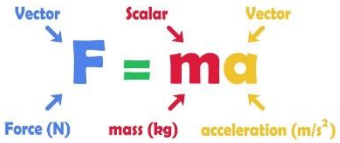
An naive, brief introduction to Dynamical Systems

How to derive a **model** for a **dynamical system** ?

- Step 1 : Which phenomenon to model ? « x_t »



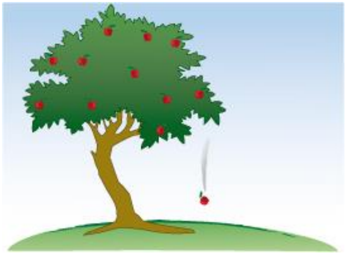
- Step 2 : Domain knowledge



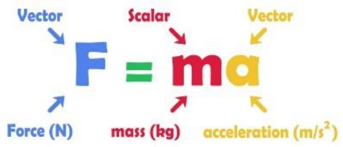
An naive, brief introduction to Dynamical Systems

How to derive a **model** for a **dynamical system** ?

- Step 1 : Which phenomenon to model ? « x_t »



- Step 2 : Domain knowledge



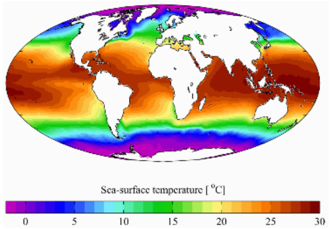
- Step 3 : Write an equation that involves the variable x_t

$$\frac{d^2 x_t}{dt^2} = g$$

An naive, brief introduction to Dynamical Systems

How to derive a **model** for a **dynamical system** ?

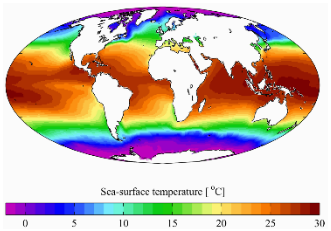
- Step 1 : Which phenomenon to model ? « T_t »



An naive, brief introduction to Dynamical Systems

How to derive a **model** for a **dynamical system** ?

- Step 1 : Which phenomenon to model ? « T_t »

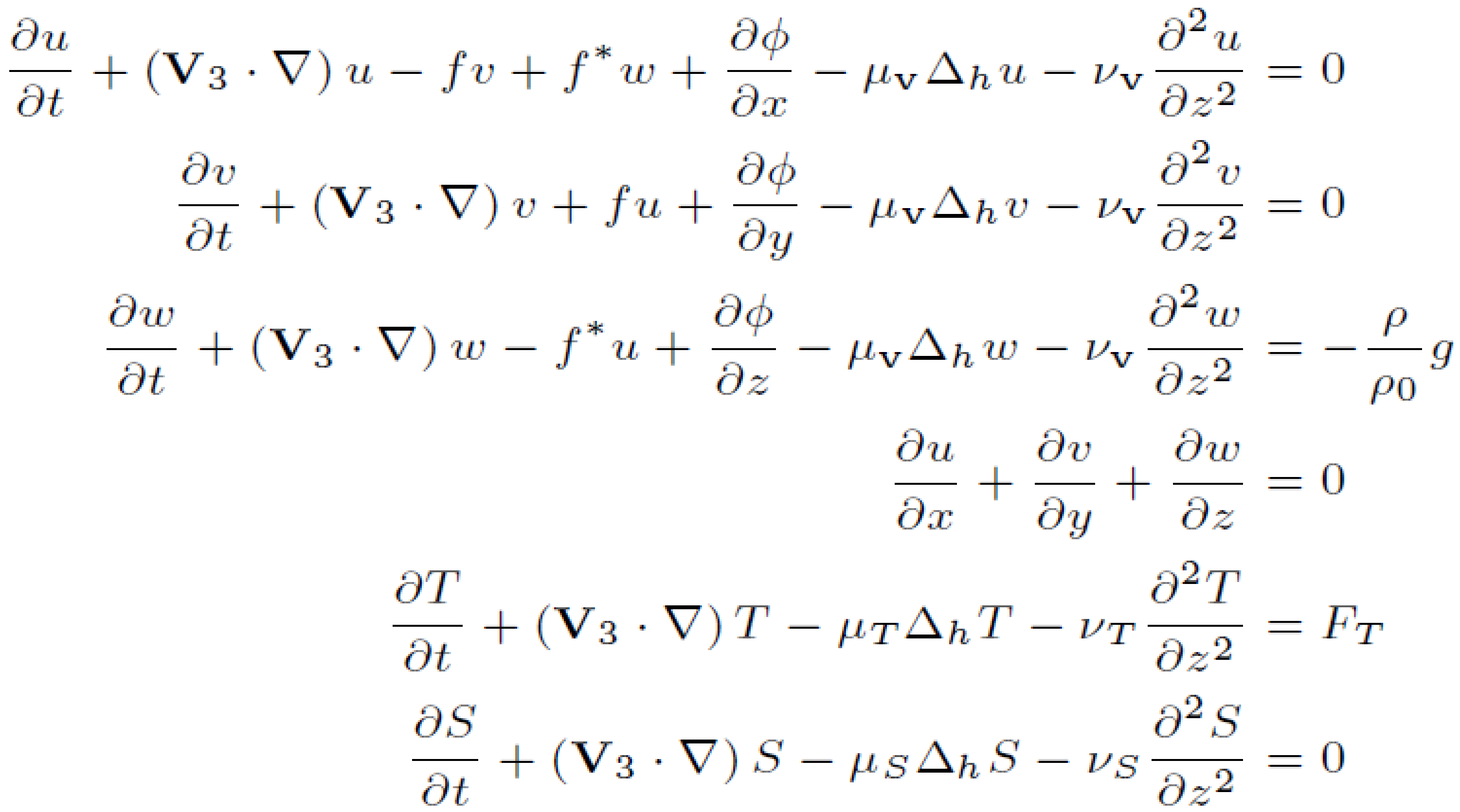


- Step 2 : Domain knowledge : Navier-Stokes

An naive, brief introduction to Dynamical Systems

How to deriv

- Step 1 : Which phenom
- Step 2 : Domain knowle
- Step 3 : Write an equat



$$
\begin{aligned}
\frac{\partial u}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, u - f v + f^* w + \frac{\partial \phi}{\partial x} - \mu_{\mathbf{v}} \Delta_h u - \nu_{\mathbf{v}} \frac{\partial^2 u}{\partial z^2} &= 0 \\
\frac{\partial v}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, v + f u + \frac{\partial \phi}{\partial y} - \mu_{\mathbf{v}} \Delta_h v - \nu_{\mathbf{v}} \frac{\partial^2 v}{\partial z^2} &= 0 \\
\frac{\partial w}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, w - f^* u + \frac{\partial \phi}{\partial z} - \mu_{\mathbf{v}} \Delta_h w - \nu_{\mathbf{v}} \frac{\partial^2 w}{\partial z^2} &= -\frac{\rho}{\rho_0} g \\
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} &= 0 \\
\frac{\partial T}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, T - \mu_T \Delta_h T - \nu_T \frac{\partial^2 T}{\partial z^2} &= F_T \\
\frac{\partial S}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, S - \mu_S \Delta_h S - \nu_S \frac{\partial^2 S}{\partial z^2} &= 0
\end{aligned}
$$

An naive, brief introduction to Dynamical Systems

Overall, a dynamical system can be described by :

An naive, brief introduction to Dynamical Systems

Overall, a dynamical system can be described by :

A state space

The number of variables of interest (is typically set to the minimum number of independent variables) that can be used to model the system : z_t

An naive, brief introduction to Dynamical Systems

Overall, a dynamical system can be described by :

A state space

The number of variables of interest (is typically set to the minimum number of independent variables) that can be used to model the system : z_t

A dynamical function

Describes the temporal evolution of the state space variables: $\frac{dz_t}{dt} = f(.)$

An naive, brief introduction to Dynamical Systems

Example1 : falling object

A state space

A dynamical function

An naive, brief introduction to Dynamical Systems

Example1 : falling object

A state space

x_t

A dynamical function

$$\frac{d^2 x_t}{dt^2} = g$$

An naive, brief introduction to Dynamical Systems

Example1 : falling object

A state space

$$x_t$$

A dynamical function

$$\frac{d^2 x_t}{dt^2} = g$$

A state space

$$z_1 = \frac{dx_t}{dt}$$
$$z_2 = x_t$$

A dynamical function

$$\begin{cases} \dfrac{dz_1}{dt} = g \\ \dfrac{dz_2}{dt} = z_1 \end{cases}$$

An naive, brief introduction to Dynamical Systems

Example2 : non-linear ODE

A state space

$$[z_1, z_2]$$

A dynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{1,t}^2) \end{cases}$$

An naive, brief introduction to Dynamical Systems

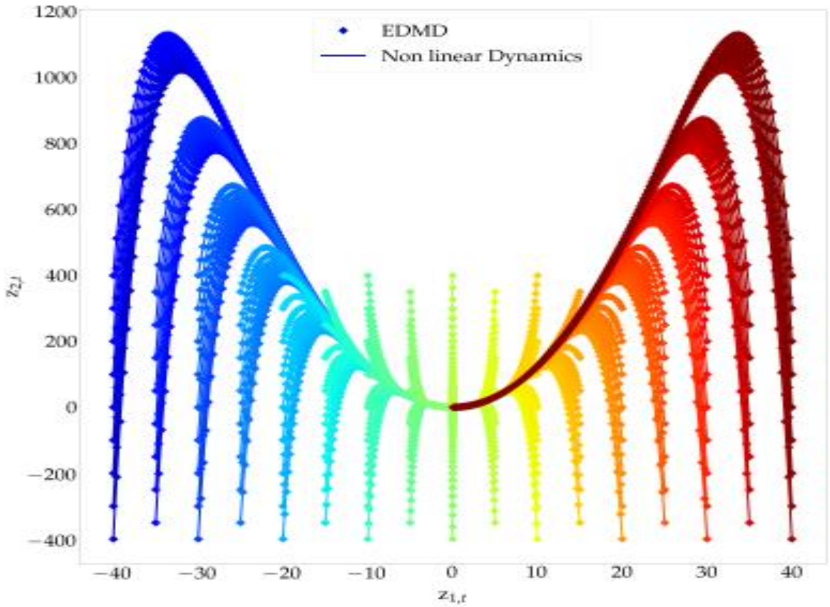
Example2 : non-linear ODE

A state space

$[z_1, z_2]$

A dynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{1,t}^2) \end{cases}$$



An naive, brief introduction to Dynamical Systems

Example2 : non-linear ODE

A state space

$[z_1, z_2]$

A dynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{1,t}^2) \end{cases}$$

A state space

$[z_1, z_2, z_3 = \ z_1^2]$

A dynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{3,t}) \\ \dot{z}_{3,t} = 2\mu z_{3,t} \end{cases}$$

An naive, brief introduction to Dynamical Systems

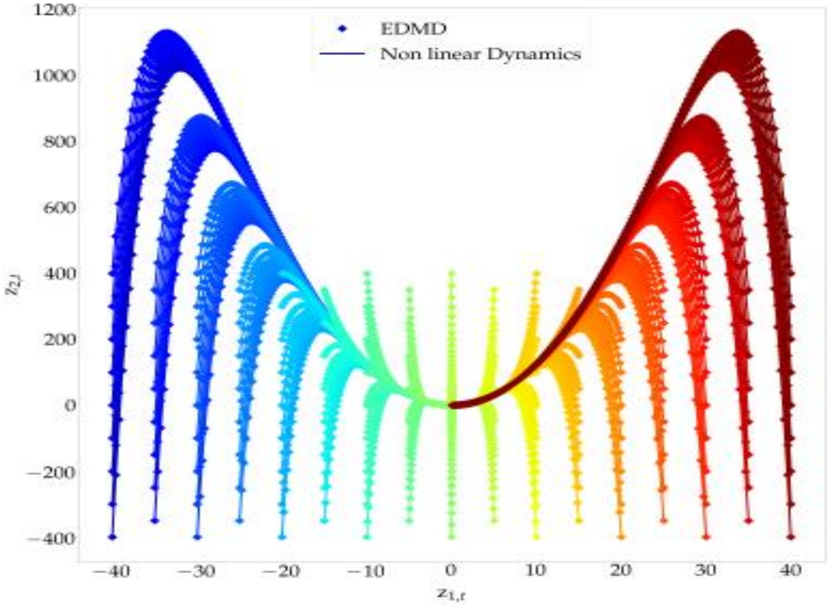
Example2 : non-linear ODE

A state space

$[z_1, z$

A dynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{1,t}^2) \end{cases}$$



A state

$[z_1, z_2, z_3$

ynamical function

$$\begin{cases} \dot{z}_{1,t} = \mu z_{1,t} \\ \dot{z}_{2,t} = \alpha(z_{2,t} - z_{3,t}) \\ \dot{z}_{3,t} = 2\mu z_{3,t} \end{cases}$$

An naive, brief introduction to Dynamical Systems

Example3 : SST data

A state space ?

A dynamical function

An naive, brief introduction to Dynamical Systems

Example3 : SST data

A

$$\frac{\partial u}{\partial t}+(\mathbf{V}_3\cdot\nabla)\,u-fv+f^*w+\frac{\partial\phi}{\partial x}-\mu_{\mathbf{v}}\Delta_h u-\nu_{\mathbf{v}}\frac{\partial^2 u}{\partial z^2}=0$$

$$\frac{\partial v}{\partial t}+(\mathbf{V}_3\cdot\nabla)\,v+fu+\frac{\partial\phi}{\partial y}-\mu_{\mathbf{v}}\Delta_h v-\nu_{\mathbf{v}}\frac{\partial^2 v}{\partial z^2}=0$$

$$\frac{\partial w}{\partial t}+(\mathbf{V}_3\cdot\nabla)\,w-f^*u+\frac{\partial\phi}{\partial z}-\mu_{\mathbf{v}}\Delta_h w-\nu_{\mathbf{v}}\frac{\partial^2 w}{\partial z^2}=-\frac{\rho}{\rho_0}g$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

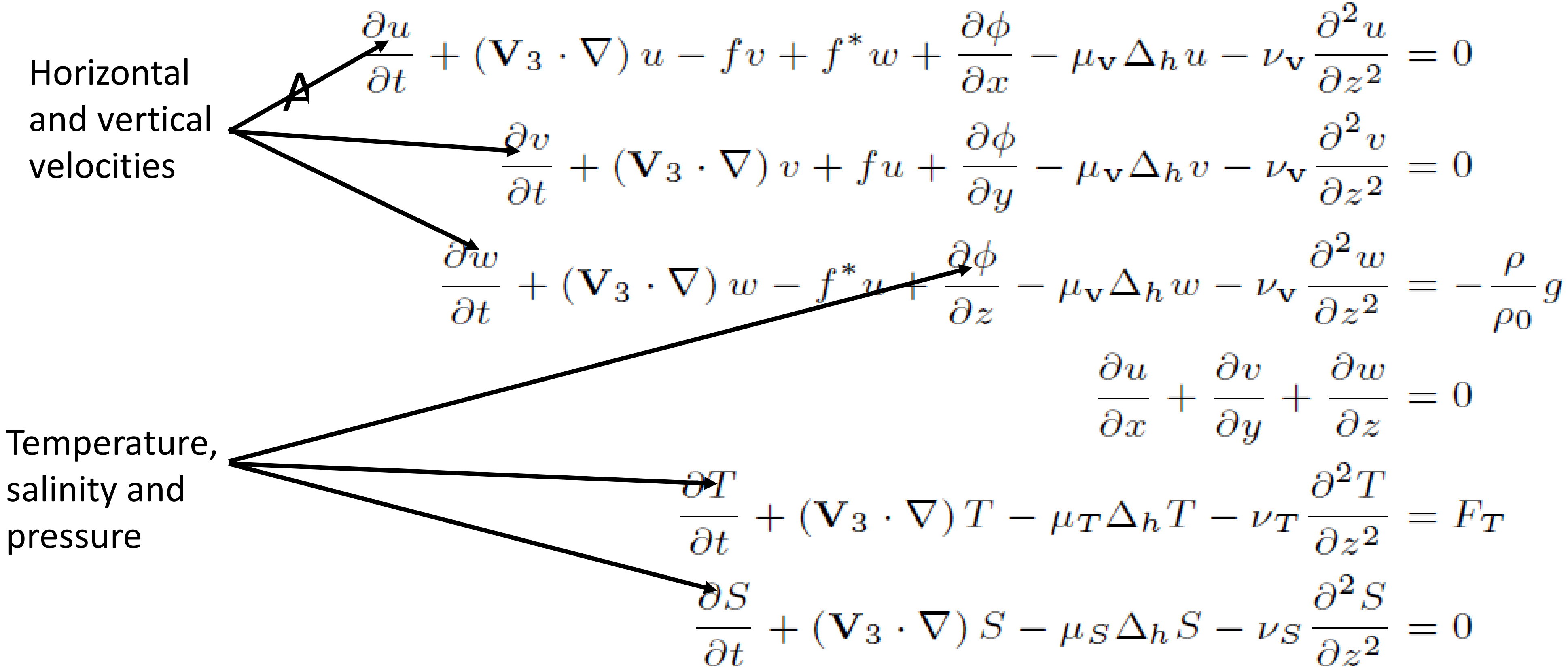
$$\frac{\partial T}{\partial t}+(\mathbf{V}_3\cdot\nabla)\,T-\mu_T\Delta_h T-\nu_T\frac{\partial^2 T}{\partial z^2}=F_T$$

$$\frac{\partial S}{\partial t}+(\mathbf{V}_3\cdot\nabla)\,S-\mu_S\Delta_h S-\nu_S\frac{\partial^2 S}{\partial z^2}=0$$

ion

An naive, brief introduction to Dynamical Systems

Example3 : SST data



Horizontal and vertical velocities

A

Temperature, salinity and pressure

$$\frac{\partial u}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, u - fv + f^* w + \frac{\partial \phi}{\partial x} - \mu_{\mathbf{v}} \Delta_h u - \nu_{\mathbf{v}} \frac{\partial^2 u}{\partial z^2} = 0$$

$$\frac{\partial v}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, v + fu + \frac{\partial \phi}{\partial y} - \mu_{\mathbf{v}} \Delta_h v - \nu_{\mathbf{v}} \frac{\partial^2 v}{\partial z^2} = 0$$

$$\frac{\partial w}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, w - f^* u + \frac{\partial \phi}{\partial z} - \mu_{\mathbf{v}} \Delta_h w - \nu_{\mathbf{v}} \frac{\partial^2 w}{\partial z^2} = -\frac{\rho}{\rho_0} g$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

$$\frac{\partial T}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, T - \mu_T \Delta_h T - \nu_T \frac{\partial^2 T}{\partial z^2} = F_T$$

$$\frac{\partial S}{\partial t} + (\mathbf{V}_3 \cdot \nabla)\, S - \mu_S \Delta_h S - \nu_S \frac{\partial^2 S}{\partial z^2} = 0$$

ion

An naive, brief introduction to Dynamical Systems

Models types :

Depending on the nature of z_t and f, several models can be distinguished :

Ordinary Differential Equations (ODEs) : the functions and derivatives of the differential equation are given with respect to a single independent variable $\frac{dz_t}{dt} = f(z_t)$

Partial Differential Equations (PDEs) : the functions and derivatives of the differential equation are given with respect to a several independent variable $f\left(\mathrm{z}(\mathrm{y}), \frac{\partial \mathrm{z}}{\partial \mathrm{y}_1}(\mathrm{y}), ..., \frac{\partial^2 \mathrm{z}}{\partial \mathrm{y}_1^2}(\mathrm{y}), \frac{\partial^2 \mathrm{z}}{\partial \mathrm{y}_1 \partial \mathrm{y}_2}(\mathrm{y}), ...\right) = 0$

And many others (Delay Differential Equations, Differential-Algebraic Equation, Stochastic Differential Equations …etc.)

Focus of this course : Ordinary Differential Equations

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

Known but expensive f

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

Known but expensive f

Non-linear f

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

Known but expensive f

Non-linear f

High dimensional z

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

Known but expensive f

Non-linear f

High dimensional z

High dimensional z

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

- Unknown f
- Known but expensive f
- Non-linear f
- High dimensional z
- High dimensional z

$$x_t = H(z_t, \Omega_t, \epsilon_t)$$

- Partial observations of z
- spatio-temporal sampling
- Noise and disturbances

An naive, brief introduction to Dynamical Systems, modern research

Why considering learning dynamical systems ?

$$\frac{dz_t}{dt} = f(z_t)$$

Unknown f

Known but expensive f

Non-linear f

High dimensional z

High dimensional z

$$x_t = H(z_t, \Omega_t, \epsilon_t)$$

Partial observations of z

spatio-temporal sampling

Noise and disturbances

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

- Given a collection of measurements $\{x_{t_n}\}_{t_0}^{t_N}$ of a time varying dynamical system.

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

- Given a collection of measurements $\{x_{t_n}\}_{t_0}^{t_N}$ of a time varying dynamical system.
- How to define something like $x_{t_n} = f_\theta(x_{t_{n-1}})$?

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

- Given a collection of measurements $\left\{x_{t_n}\right\}_{t_0}^{t_N}$ of a time varying dynamical system.
- How to define something like $x_{t_n} = f_\theta(x_{t_{n-1}})$?
- Let us assume that $\left\{x_{t_n}\right\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

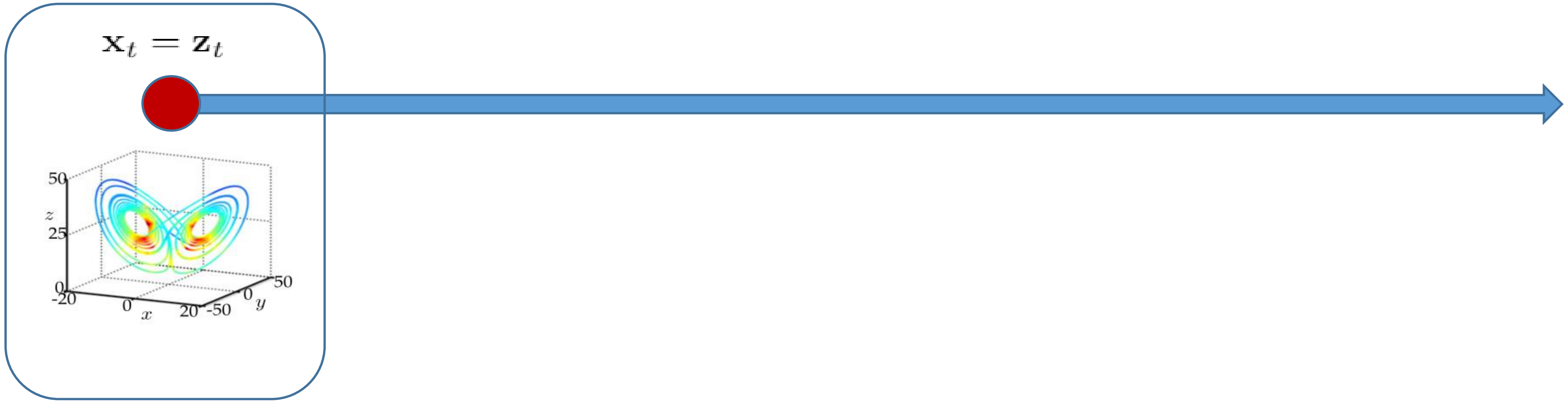
- Given a collection of measurements $\{x_{t_n}\}_{t_0}^{t_N}$ of a time varying dynamical system.
- How to define something like $x_{t_n} = f_\theta(x_{t_{n-1}})$?
- Let us assume that $\{x_{t_n}\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

- Deriving a dynamical model for the observations $\{x_{t_n}\}_{t_0}^{t_N}$ is subject to numerous questions regarding Ω_t, ϵ_t and H

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

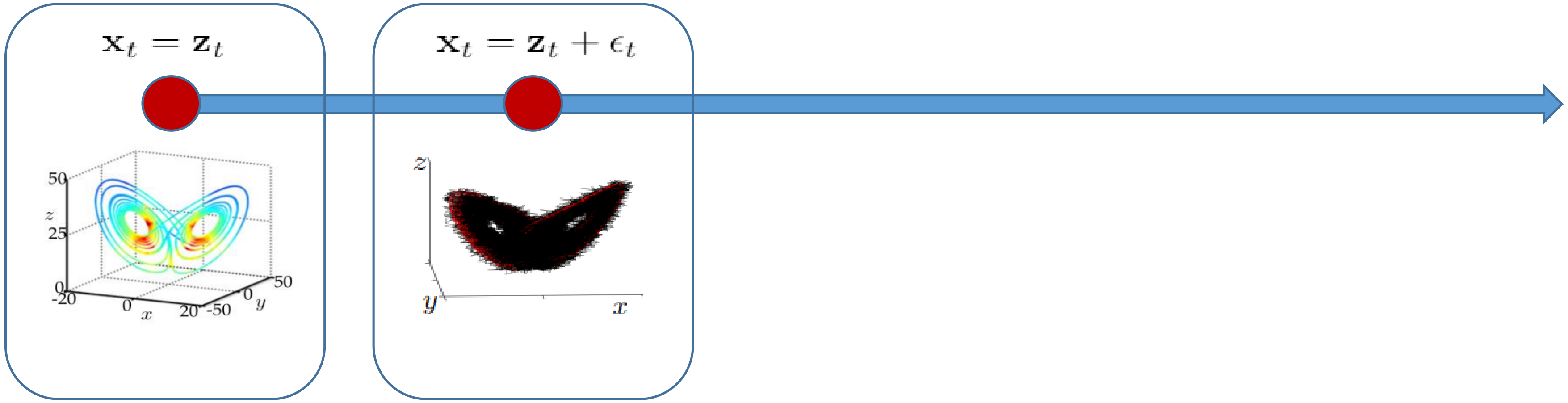


Towards the complexity of real world systems

- Dictionary based approaches (Brunton et al. (2016b))
- Neural networks (Chen et al. (2018))
- Non parametric approaches (Lguensat et al. (2017))

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$



$\mathbf{x}_t = \mathbf{z}_t$

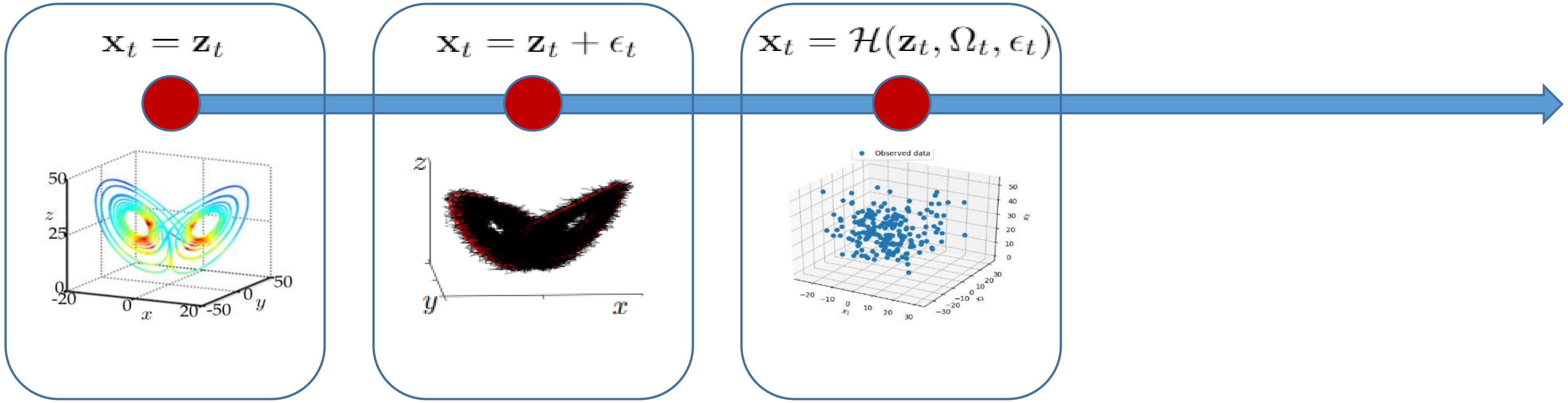
$\mathbf{x}_t = \mathbf{z}_t + \epsilon_t$

Towards the complexity of real world systems

- Data denoising (Lalley and Nobel (2006))

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$



$\mathbf{x}_t = \mathbf{z}_t$

$\mathbf{x}_t = \mathbf{z}_t + \epsilon_t$

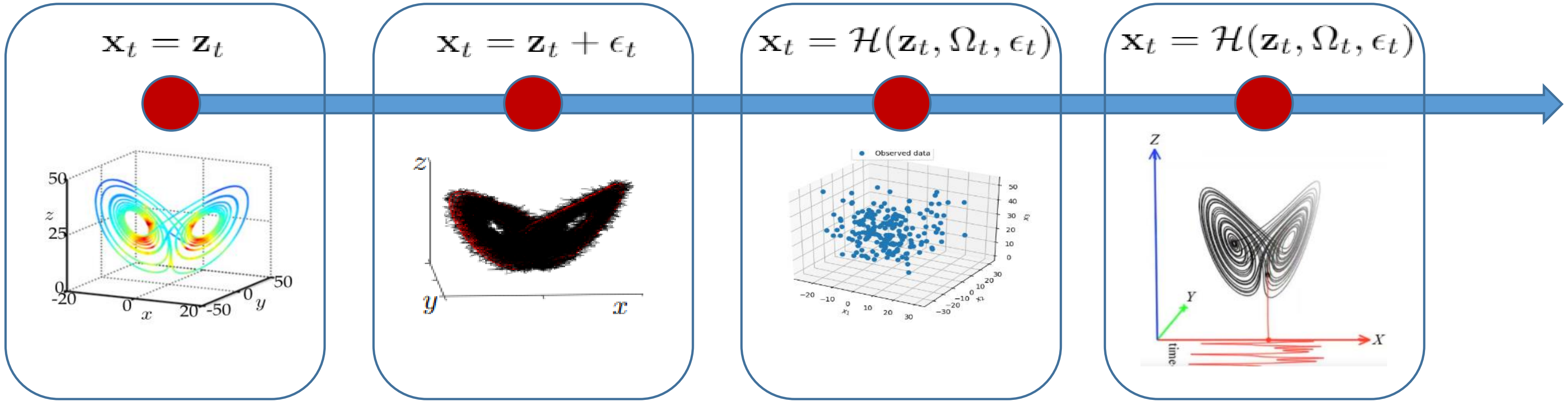
$\mathbf{x}_t = \mathcal{H}(\mathbf{z}_t, \Omega_t, \epsilon_t)$

Towards the complexity of real world systems

- Learning in data assimilation frameworks (Bocquet et al. (2019)), (Brajard et al. (2020)), (Nguyen et al. (2020))
- Deep learning based approaches (Variational autoencoders) (Nguyen et al. (2020))

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$



$\mathbf{x}_t = \mathbf{z}_t$

$\mathbf{x}_t = \mathbf{z}_t + \epsilon_t$

$\mathbf{x}_t = \mathcal{H}(\mathbf{z}_t, \Omega_t, \epsilon_t)$

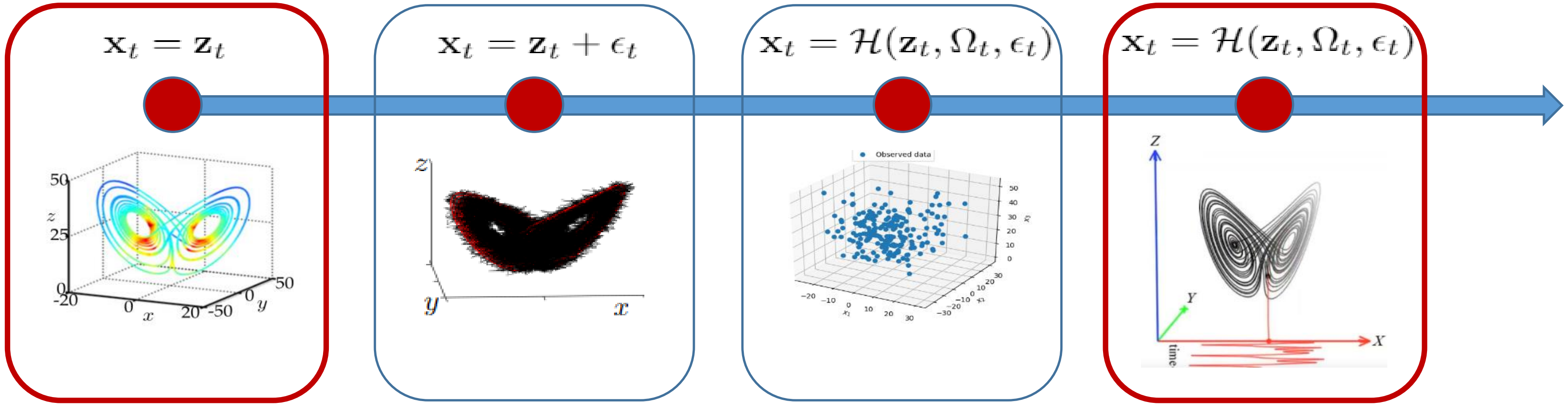
$\mathbf{x}_t = \mathcal{H}(\mathbf{z}_t, \Omega_t, \epsilon_t)$

Towards the complexity of real world systems

- Delay embedding and regression (Kazem et al. (2013))
- Recurrent neural networks

An naive, brief introduction to Dynamical Systems: Learning formulation (unknown f)

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$



$\mathbf{x}_t = \mathbf{z}_t$

$\mathbf{x}_t = \mathbf{z}_t + \epsilon_t$

$\mathbf{x}_t = \mathcal{H}(\mathbf{z}_t, \Omega_t, \epsilon_t)$

$\mathbf{x}_t = \mathcal{H}(\mathbf{z}_t, \Omega_t, \epsilon_t)$

Towards the complexity of real world systems

Resolution of differential equations : numerical integration

Resolution of differential equations : numerical integration

- Problem formulation
- Numerical integration types and single-step explicit techniques
- Performance criteria
- Go further
    - Adaptive step-size techniques
    - Implicit schemes

Resolution of differential equations : numerical integration : Problem formulation

- Let us assume a continuous s-dimensional dynamical system z_t governed by the following non-autonomous time varying ODE

$$\dot{\mathbf{z}}_t = f(t, \mathbf{z}_t)$$

Resolution of differential equations : numerical integration : Problem formulation

- Let us assume a continuous s-dimensional dynamical system z_t governed by the following non-autonomous time varying ODE

$$\dot{\mathbf{z}}_t = f(t, \mathbf{z}_t)$$

- Assuming that, given an initial condition z_{t_0}, we aim to solve this equation for an interval t ∈ [t0, tf]

$$\Phi_t(\mathbf{z}_{t_0}) = \mathbf{z}_{t_0} + \int_{t_0}^{t} f(w, \mathbf{z}_w) dw$$

Resolution of differential equations : numerical integration : Problem formulation

- Let us assume a continuous s-dimensional dynamical system z_t governed by the following non-autonomous time varying ODE

$$\dot{\mathbf{z}}_t = f(t, \mathbf{z}_t)$$

- Assuming that, given an initial condition z_{t_0}, we aim to solve this equation for an interval t ∈ [t0, tf]

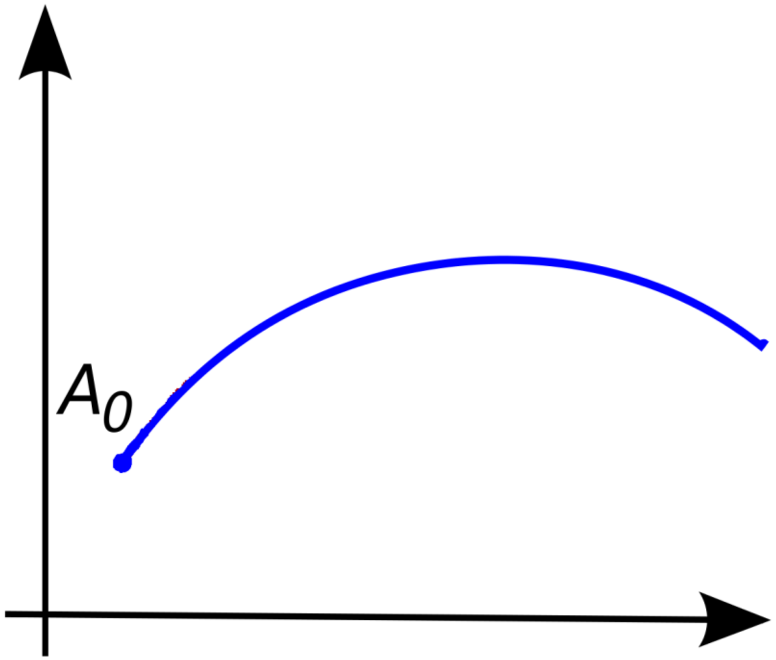
$$\Phi_t(\mathbf{z}_{t_0}) = \mathbf{z}_{t_0} + \int_{t_0}^{t} f(w, \mathbf{z}_w) dw$$

- solving the above equation is only possible for a small subset of non-linear ODEs.

Resolution of differential equations : numerical integration : Problem formulation

- Solution : map the **integral** equation into a **discrete** equation

$$\Phi_t(\mathbf{z}_{t_0}) = \mathbf{z}_{t_0} + \int_{t_0}^{t} f(w, \mathbf{z}_w) dw$$

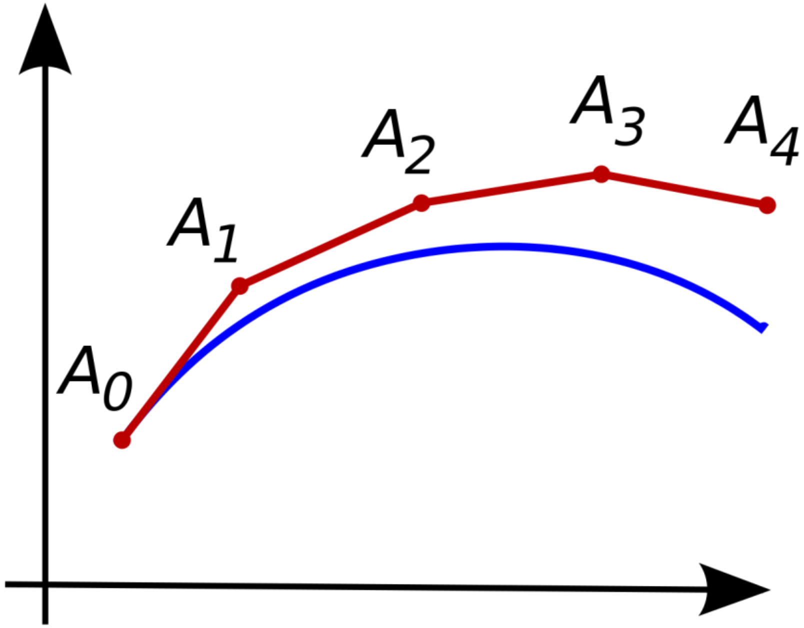


Resolution of differential equations : numerical integration : Problem formulation

- Solution : map the **integral** equation into a **discrete** equation

$$\Phi_t(\mathbf{z}_{t_0}) = \mathbf{z}_{t_0} + \int_{t_0}^{t} f(w, \mathbf{z}_w)dw$$

$$\hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{E},t_n}(\hat{\mathbf{z}}_{t_0}) = \hat{\mathbf{z}}_{t_n} + hf(t_n, \hat{\mathbf{z}}_{t_n}^T)$$



Resolution of differential equations : numerical integration : Problem formulation

- Solution : map the **integral** equation into a **discrete** equation

- Formally, the interval t ∈ [t0, tf] is discretized using a time-step h > 0 as h=(tf −t0)/N and tn = t0 + nh, where 0 ≤ n ≤ N an integer and N is the number of grid points,

$$\begin{cases} \hat{\mathbf{z}}_{t_0} = \mathbf{z}_{t_0} = \mathbf{z}_0 \\ \hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{D},t_{n+1}}(\hat{\mathbf{z}}_{t_n}) \approx \mathbf{z}_{t_{n+1}} = \Phi_{t_{n+1}}(\mathbf{z}_{t_n}) \end{cases}$$

Resolution of differential equations : numerical integration : Problem formulation

- Solution : map the **integral** equation into a **discrete** equation

- Formally, the interval t ∈ [t0, tf] is discretized using a time-step h > 0 as h=(tf −t0)/N and tn = t0 + nh, where 0 ≤ n ≤ N an integer and N is the number of grid points,

$$\begin{cases} \hat{\mathbf{z}}_{t_0} = \mathbf{z}_{t_0} = \mathbf{z}_0 \\ \hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{D},t_{n+1}}(\hat{\mathbf{z}}_{t_n}) \approx \mathbf{z}_{t_{n+1}} = \Phi_{t_{n+1}}(\mathbf{z}_{t_n}) \end{cases}$$

- with $\hat{z}_{t_n}$ a **numerical solution** computed using the **approximation** $\Phi_{D,t_n}$of the analytical solution Φ_{t_n} and D is a given integration scheme.

Resolution of differential equations : numerical integration : Numerical integration types

- Lots of ways to define our time integration scheme Φ_{D,t_n} :

- Single-step techniques vs multistep techniques

- Explicit vs implicit algorithms

- Fixed step-size vs adaptive step-size techniques

Resolution of differential equations : numerical integration : Single-step techniques

- The general form of single-step explicit integration schemes can be derived from the Taylor expansion of the solution of an ODE:

$$\mathbf{z}_{t_{n+1}} = \mathbf{z}_{t_n} + \sum_{k=1}^{p=+\infty} h^k \frac{1}{k!} f^{k-1}(t_n, \mathbf{z}_{t_n})$$

- Examples :

$$\hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{E},t_n}(\hat{\mathbf{z}}_{t_0}) = \hat{\mathbf{z}}_{t_n} + hf(t_n, \hat{\mathbf{z}}_{t_n}^T)$$

Resolution of differential equations : numerical integration : Performance criteria

How to choose an integration scheme ?

Resolution of differential equations : numerical integration : Performance criteria

- General numerical integration problem can be formulated as :

$$\begin{cases} \hat{\mathbf{z}}_{t_0} = \mathbf{z}_{t_0} = \mathbf{z}_0 \\ \hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{D},t_{n+1}}(\hat{\mathbf{z}}_{t_n}) \approx \mathbf{z}_{t_{n+1}} = \Phi_{t_{n+1}}(\mathbf{z}_{t_n}) \end{cases}$$

- How to make sure that $\hat{z}_{t_n}$is close to z_{t_n}?

Resolution of differential equations : numerical integration : Performance criteria

- General numerical integration problem can be formulated as :

$$\begin{cases} \hat{\mathbf{z}}_{t_0} = \mathbf{z}_{t_0} = \mathbf{z}_0 \\ \hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{D},t_{n+1}}(\hat{\mathbf{z}}_{t_n}) \approx \mathbf{z}_{t_{n+1}} = \Phi_{t_{n+1}}(\mathbf{z}_{t_n}) \end{cases}$$

- How to make sure that $\hat{z}_{t_n}$is close to z_{t_n}?

- Errors and stability criteria of integration schemes !

Resolution of differential equations : numerical integration : Performance criteria

- General numerical integration problem can be formulated as :

$$\begin{cases} \hat{\mathbf{z}}_{t_0} = \mathbf{z}_{t_0} = \mathbf{z}_0 \\ \hat{\mathbf{z}}_{t_{n+1}} = \Phi_{\mathcal{D},t_{n+1}}(\hat{\mathbf{z}}_{t_n}) \approx \mathbf{z}_{t_{n+1}} = \Phi_{t_{n+1}}(\mathbf{z}_{t_n}) \end{cases}$$

- How to make sure that $\hat{z}_{t_n}$ is close to z_{t_n}?

- Errors and stability criteria of integration schemes !

Truncation errors

Absolute stability

Resolution of differential equations : numerical integration : Truncation error

- The error committed by using a single integration time step $\epsilon_{t_n} = z_{t_n} - \hat{z}_{t_n}$

Resolution of differential equations : numerical integration : Truncation error

- The error committed by using a single integration time step $\epsilon_{t_n} = z_{t_n} - \hat{z}_{t_n}$

- If we consider the Taylor expansions of both $z_{t_n} and\ \hat{z}_{t_n}$

$$\epsilon_{t_n} = h^{p+1} \frac{1}{(p+1)!} f^{p}\left(t_n, z_{t_n}\right) + \sum_{k=p+2}^{\infty} h^{k} \frac{1}{k!} f^{k-1}\left(t_n, z_{t_n}\right)$$

Resolution of differential equations : numerical integration : Truncation error

- The error committed by using a single integration time step $\epsilon_{t_n} = z_{t_n} - \hat{z}_{t_n}$

- If we consider the Taylor expansions of both $z_{t_n} and\ \hat{z}_{t_n}$

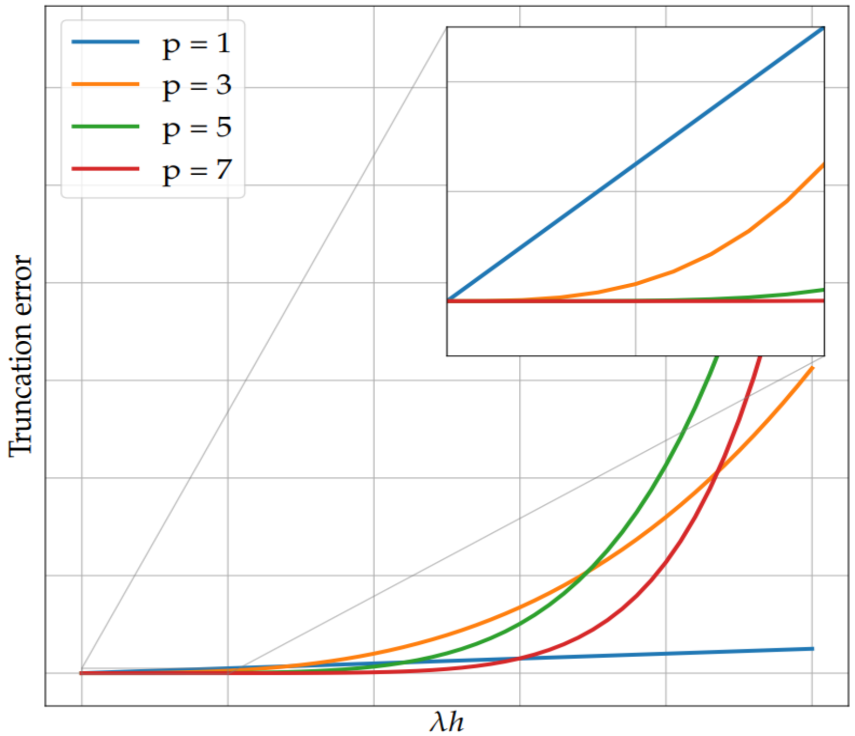
$$\epsilon_{t_n} = h^{p+1} \frac{1}{(p+1)!} f^{p}\left(t_n, z_{t_n}\right) + \sum_{k=p+2}^{\infty} h^{k} \frac{1}{k!} f^{k-1}\left(t_n, z_{t_n}\right)$$

- When considering a linear ODE, and by neglecting terms k>p+1 :

$$\epsilon_{t_n} = h^{p+1} \frac{(\lambda h)^{p+1}}{(p+1)!}$$

Resolution of differential equations : numerical integration : Truncation error

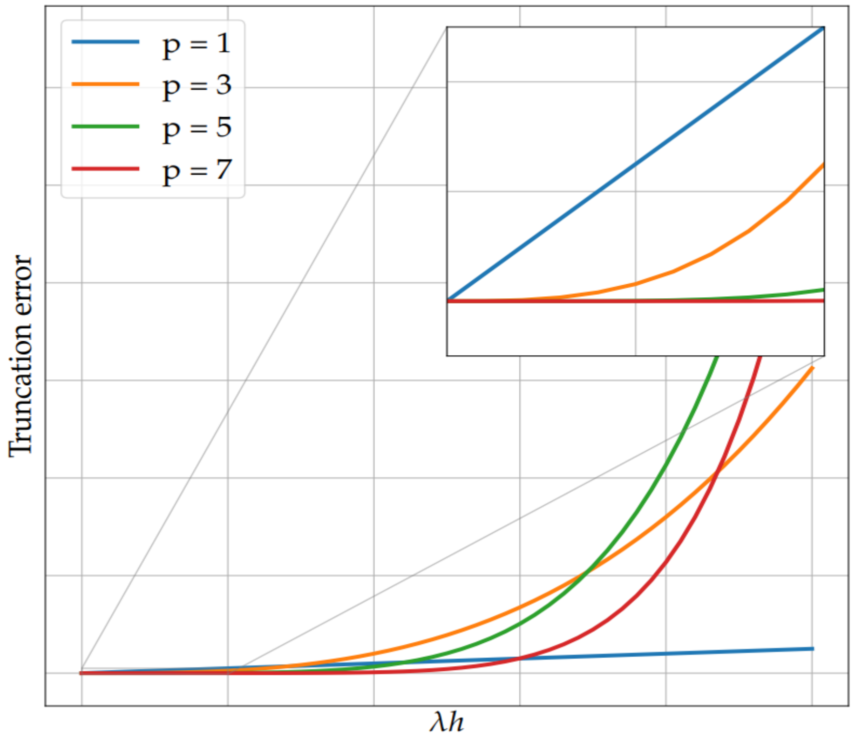
- Truncation error of a order “p” integration scheme on a linear equation :



- High order schemes have a lower truncation error ?
- Above a threshold, low order schemes have a lower truncation error

Resolution of differential equations : numerical integration : Truncation error

- Truncation error of a order “p” integration scheme on a linear equation :



- Let us compare Euler and Runge-Kutta 4 integration techniques
- Find the error ^^ (hint: is this really the truncation error ?)

Resolution of differential equations : numerical integration : Absolute stability

- The truncation error is computed for a **perfect initial condition**
- What happens in practice : errors propagates from a time step to an other and may become unbounded
- Stability analysis : making sure that the integration scheme does not blow up, regardless of its precision properties

Resolution of differential equations : numerical integration : Absolute stability

- Let us consider : $\begin{cases} \dot{\mathbf{z}}_t = \lambda \mathbf{z}_t \\ \mathbf{z}_{t_0} = \mathbf{z}_0 \end{cases}$ With Real(λ) ≤ 0

Resolution of differential equations : numerical integration : Absolute stability

- Let us consider : $\begin{cases} \dot{\mathbf{z}}_t = \lambda \mathbf{z}_t \\ \mathbf{z}_{t_0} = \mathbf{z}_0 \end{cases}$ With Real(λ) ≤ 0

- Analytical solution : $z_{t_n} = \Phi_{t_n}\left(z_{t_{n-1}}\right) = z_{t_{n-1}} e^{\lambda h}$

Resolution of differential equations : numerical integration : Absolute stability

- Let us consider : $\begin{cases} \dot{\mathbf{z}}_t = \lambda \mathbf{z}_t \\ \mathbf{z}_{t_0} = \mathbf{z}_0 \end{cases}$ With Real(λ) ≤ 0

- Analytical solution : $z_{t_n} = \Phi_{t_n}\left(z_{t_{n-1}}\right) = z_{t_{n-1}} e^{\lambda h}$

- The solution of this equation, using an integration scheme can be written as: $\hat{z}_{t_n} = \hat{z}_{t_{n-1}} R(\lambda h)$

Resolution of differential equations : numerical integration : Absolute stability

- Let us consider : $\begin{cases} \dot{\mathbf{z}}_t = \lambda \mathbf{z}_t \\ \mathbf{z}_{t_0} = \mathbf{z}_0 \end{cases}$ With Real(λ) ≤ 0

- Analytical solution : $z_{t_n} = \Phi_{t_n}\left(z_{t_{n-1}}\right) = z_{t_{n-1}} e^{\lambda h}$

- The solution of this equation, using an integration scheme can be written as: $\hat{z}_{t_n} = \hat{z}_{t_{n-1}} R(\lambda h)$

- This numerical solution is stable for a given λh if $|R(\lambda h)| \leq 1$

Resolution of differential equations : numerical integration : Absolute stability

Example : stability analysis of Euler scheme

$$\hat{z}_{t_n} = \Phi_{Euler,t_n}\left(\hat{z}_{t_{n-1}}\right)$$

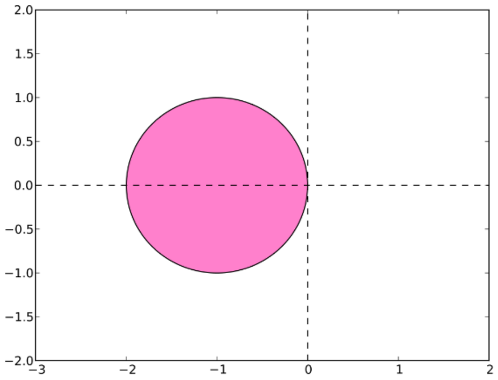
$$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} + \lambda h \hat{z}_{t_{n-1}}$$

Resolution of differential equations : numerical integration : Absolute stability

Example : stability analysis of Euler scheme

$$\hat{z}_{t_n} = \Phi_{Euler,t_n}\left(\hat{z}_{t_{n-1}}\right)$$
$$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} + \lambda h \hat{z}_{t_{n-1}}$$
$$\hat{z}_{t_n} = \hat{z}_{t_{n-1}}(1 + \lambda h)$$

$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} R(\lambda h)$ => $R(\lambda h) = (1 + \lambda h)$



Resolution of differential equations : numerical integration : Absolute stability

Example : stability analysis of the Runge-Kutta 4 scheme

$\hat{z}_{t_n} = \Phi_{RK4,t_n}\left(\hat{z}_{t_{n-1}}\right)$

$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} R(\lambda h)$ with $R = b^T(I - \lambda h\, A^{-1})\mathbf{1}$

Resolution of differential equations : numerical integration : Absolute stability

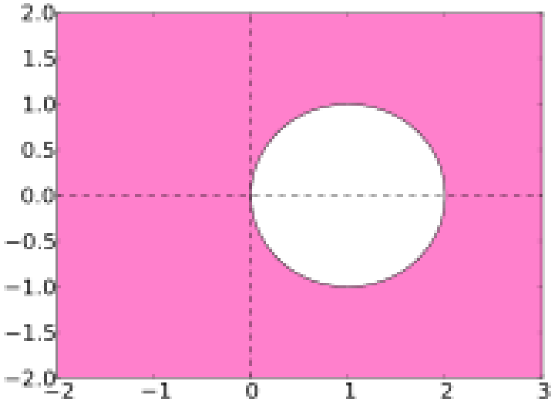
Exercice : stability analysis of the implicit Euler scheme

$$\hat{z}_{t_n} = \Phi_{Euler,t_n}\left(\hat{z}_{t_n}\right)$$

$$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} + \lambda h \hat{z}_{t_n}$$

Resolution of differential equations : numerical integration : Absolute stability

Exercice : stability analysis of the implicit Euler scher



$$\hat{z}_{t_n} = \Phi_{Euler,t_n}\left(\hat{z}_{t_n}\right)$$

$$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} + \lambda h \hat{z}_{t_n}$$

$$\hat{z}_{t_n} = \frac{1}{1-\lambda h}\hat{z}_{t_{n-1}}$$

$\hat{z}_{t_n} = \hat{z}_{t_{n-1}} R(\lambda h)$ => $R(\lambda h) = \frac{1}{1-\lambda h}$

Resolution of differential equations : numerical integration : Recap

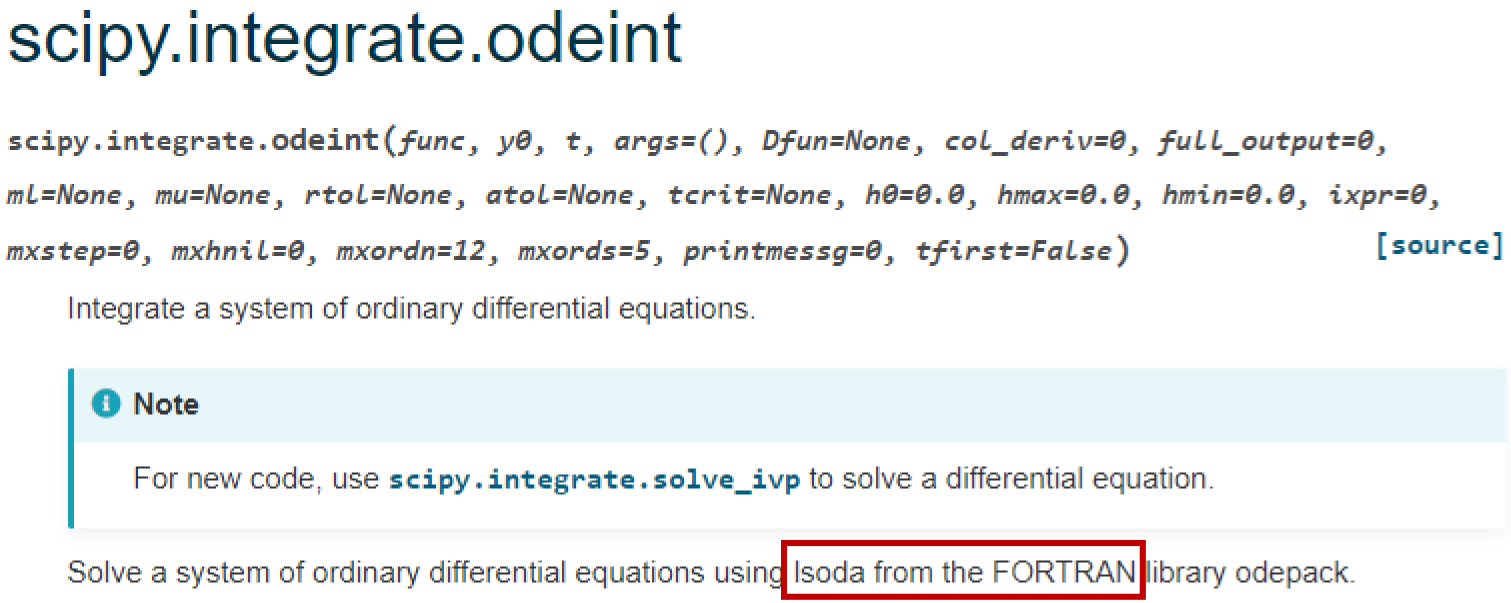
- Differential equations are most of the time solved using numerical schemes
- Numerical schemes should respect some stability and error criteria
- These criteria are most of the time built on linear equations
- What about non-linear differential equations

Resolution of differential equations : numerical integration : Recap

- Differential equations are most of the time solved using numerical schemes
- Numerical schemes should respect some stability and error criterions
- These criteria are most of the time built on linear equations
- What about non-linear differential equations
- In practice, numerical integration techniques are blackboxes ^^’’

Resolution of differential equations : numerical integration : Recap

- Example ODE solver :



scipy.integrate.odeint

```
scipy.integrate.odeint(func, y0, t, args=(), Dfun=None, col_deriv=0, full_output=0,
ml=None, mu=None, rtol=None, atol=None, tcrit=None, h0=0.0, hmax=0.0, hmin=0.0, ixpr=0,
mxstep=0, mxhnil=0, mxordn=12, mxords=5, printmessg=0, tfirst=False)
```

[source]

Integrate a system of ordinary differential equations.

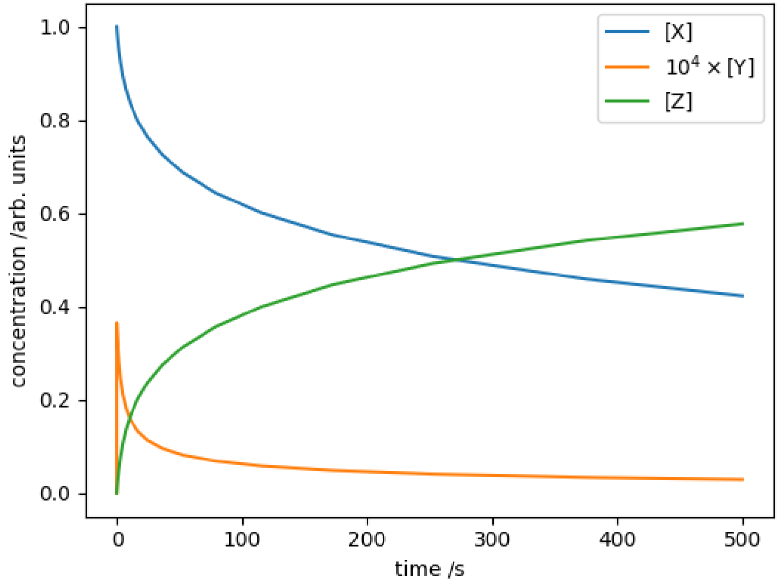
> **Note**
>
> For new code, use `scipy.integrate.solve_ivp` to solve a differential equation.

Solve a system of ordinary differential equations using lsoda from the FORTRAN library odepack.

- Adaptive step size, automatic switching between integration techniques, …etc.

Resolution of differential equations : numerical integration : Recap

- A comment on stiff equations ?



Learning dynamical systems 2 : training formulation

- Continuous time setting :
  - Dictionary based approaches
  - Limitations
- Discrete time setting
  - Neural ODEs
  - Backpropagation through ODE solvers
  - Learning integration schemes

Learning dynamical systems 2 : training formulation

- Let us assume that $\left\{x_{t_n}\right\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

Learning dynamical systems 2 : training formulation

- Let us assume that $\{x_{t_n}\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

- If we assume $x_t = z_t$, we can write a model $\frac{dx_t}{dt} = f_\theta(x_t)$

Learning dynamical systems 2 : training formulation

- Let us assume that $\{x_{t_n}\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

- If we assume $x_t = z_t$, we can write a model $\frac{dx_t}{dt} = f_\theta(x_t)$
- Question : How to optimize the parameters θ ?

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- Naïve solution : Linear regression

$$\frac{dx_t}{dt} = Ax_t$$

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- Naïve solution : Linear regression

$$\frac{dx_t}{dt} = Ax_t$$

- The matrix A can be optimized using least squares : $A = pinv\left(x, \frac{\widehat{dx}}{dt}\right)$

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- Naïve solution : Linear regression

$$\frac{dx_t}{dt} = Ax_t$$

- The matrix A can be optimized using least squares : $A = pinv\left(x, \frac{\widehat{dx}}{dt}\right)$
- Issue : only relevant for linear dynamics :/

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- A more elaborate solution : Dictionary based + Linear regression

$$\frac{dx_t}{dt} = \Theta(x_t)\Xi$$

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- A more elaborate solution : Dictionary based + Linear regression

$$\frac{dx_t}{dt} = \Theta(x_t)\Xi$$

- The matrix $\Theta(x_t)$ is a (matrix) dictionary of non-linear terms and Ξ is the activations of $\Theta(x_t)$

Learning dynamical systems 2 : training formulation

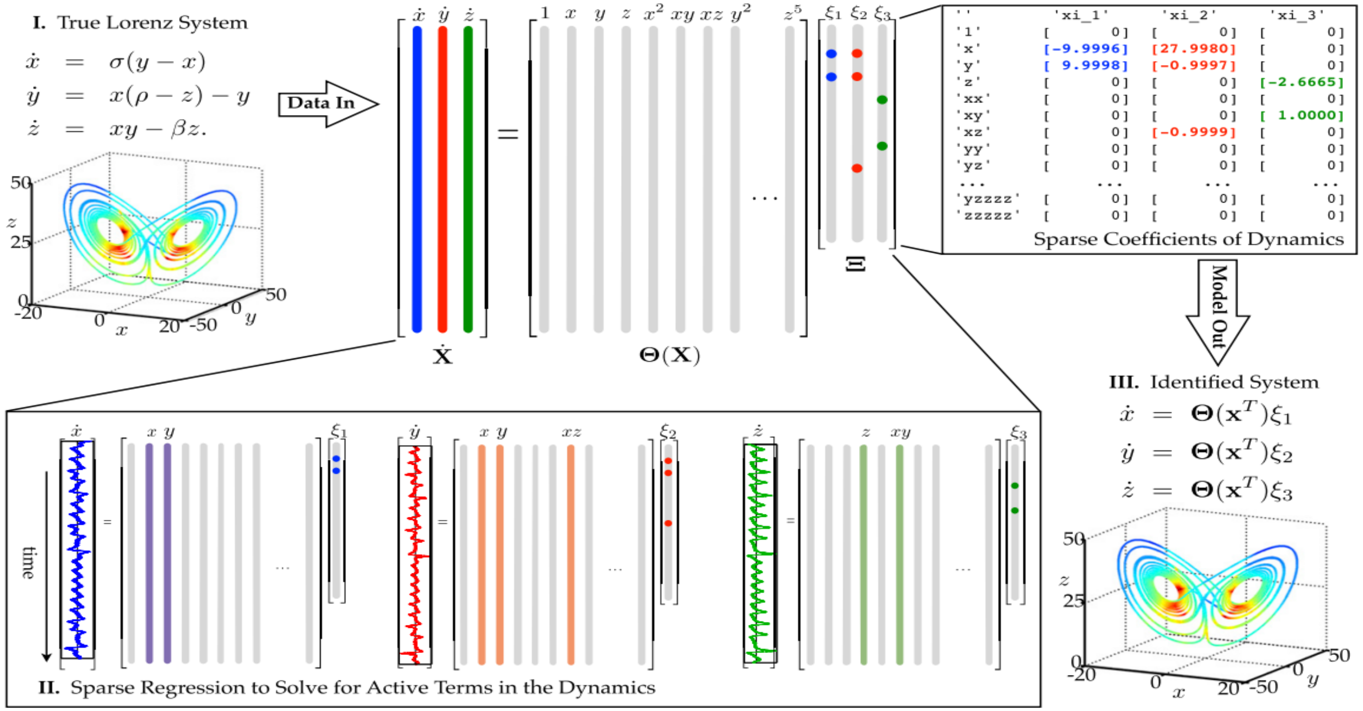
- Question : How to optimize the parameters θ ?
- A more elaborate solution : Dictionary based + Linear regression

$$\frac{dx_t}{dt} = \Theta(x_t)\Xi$$

- The matrix $\Theta(x_t)$ is a (matrix) dictionary of non-linear terms and Ξ is the activations of $\Theta(x_t)$
- Ξ is computed using least squares :$\Xi = pinv\left(\Theta(x_t), \frac{\widehat{dx}}{dt}\right)$

Learning dynamical systems 2 : training formulation

- Question : How to optimize the parameters θ ?
- A more elaborate solution : Dictionary based + Linear regression + thresholded least squares = SINDy


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
Data In
$\dot{x}$ $\dot{y}$ $\dot{z}$
1 x y z x^2 xy xz y^2 z^5
ξ_1 ξ_2 ξ_3
$\dot{\mathbf{X}}$
$\boldsymbol{\Theta}(\mathbf{X})$
$\boldsymbol{\Xi}$
Model Out
III. Identified System
$\dot{x} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_1$
$\dot{y} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_2$
$\dot{z} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_3$
time
II. Sparse Regression to Solve for Active Terms in the Dynamics
z x y
50 25 0 -20 0 20 -50 0 50

```
''          'xi_1'      'xi_2'      'xi_3'
'1'       [      0]  [      0]  [      0]
'x'       [-9.9996]  [27.9980]  [      0]
'y'       [ 9.9998]  [-0.9997]  [      0]
'z'       [      0]  [      0]  [-2.6665]
'xx'      [      0]  [      0]  [      0]
'xy'      [      0]  [      0]  [ 1.0000]
'xz'      [      0]  [-0.9999]  [      0]
'yy'      [      0]  [      0]  [      0]
'yz'      [      0]  [      0]  [      0]
 ...          ...         ...         ...
'yzzzz'   [      0]  [      0]  [      0]
'zzzzz'   [      0]  [      0]  [      0]
```

Sparse Coefficients of Dynamics

Learning dynamical systems 2 : training formulation

- Dictionary based approaches :

Pros : easy optimization, can provide analytical dynamical systems

Cons : What if the non-linearities does not linearize the regression ? Need to estimate the derivatives !!!

Learning dynamical systems 2 : training formulation

- Neural ODEs : given measurements $\{x_{t_n}\}_{t_0}^{t_N}$ with $x_t = z_t$, and assuming the following neural network model

$$\frac{dx_t}{dt} = f_\theta(x_t)$$

Learning dynamical systems 2 : training formulation

- Neural ODEs : given measurements $\left\{x_{t_n}\right\}_{t_0}^{t_N}$ with $x_t = z_t$, and assuming the following neural network model

$$\frac{dx_t}{dt} = f_\theta(x_t)$$

- We start by discretizing the above equation :

$$\hat{x}_{t_n} = \Phi_{D,t_n}(x_{t_{n-1}})$$

Learning dynamical systems 2 : training formulation

- Neural ODEs : given measurements $\{x_{t_n}\}_{t_0}^{t_N}$ with $x_t = z_t$, and assuming the following neural network model

$$\frac{dx_t}{dt} = f_\theta(x_t)$$

- We start by discretizing the above equation :

$$\hat{x}_{t_n} = \Phi_{D,t_n}(x_{t_{n-1}})$$

- The parameters θ are computed by minimization of a forecasting cost:

$$\hat{\theta} = Argmin_\theta \left| x_{t_n} - \Phi_{D,t_n}(x_{t_{n-1}})\ \right|$$

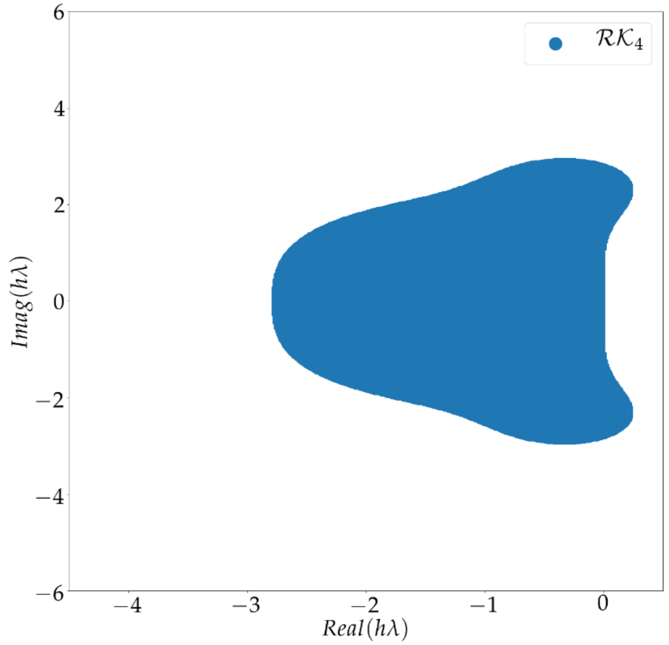
Learning dynamical systems 2 : training formulation

- Neural ODEs : Couple comments
- Comment 1) Which integration scheme to use ?

Learning dynamical systems 2 : training formulation

- Neural ODEs : Couple comments
- Comment 1) Which integration scheme to use ?

- If we use Runge-Kutta 4 with a given time step, we might be in an unstable region of the integration scheme ➔ identifiability is impossible :/



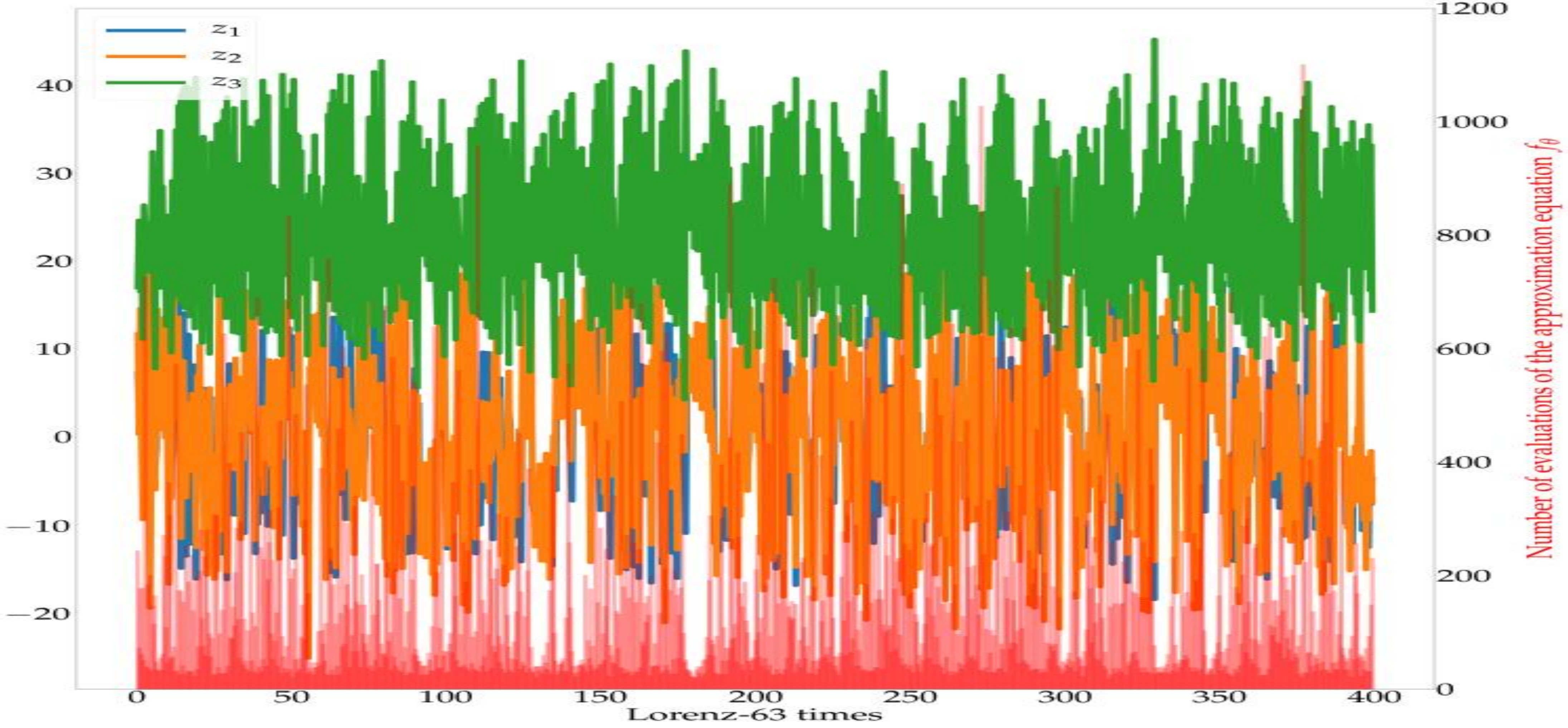
Learning dynamical systems 2 : training formulation

- Neural ODEs : Couple comments
- Comment 1) Which integration scheme to use ?
- Comment 2) Let us use adaptive step-size solvers ^^ (chen et al. 2018)

Learning dynamical systems 2 : training formulation

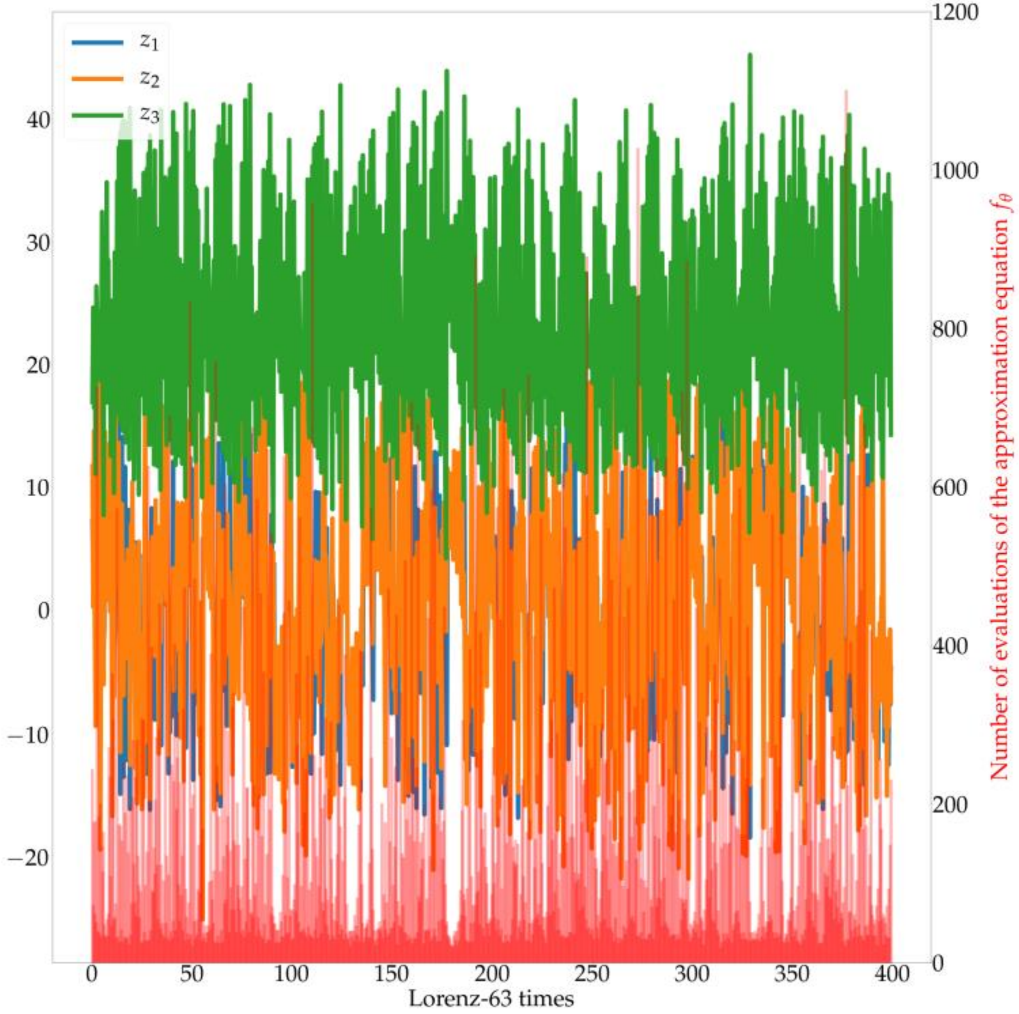
- Neural ODEs : Couple comments
- Comment 1) Which integration scheme to use ?
- Comment 2) Let us use adaptive step-size solvers ^^ (chen et al. 2018) issue : Backpropagation through ODE solvers

Learning dynamical systems 2 : training formulation



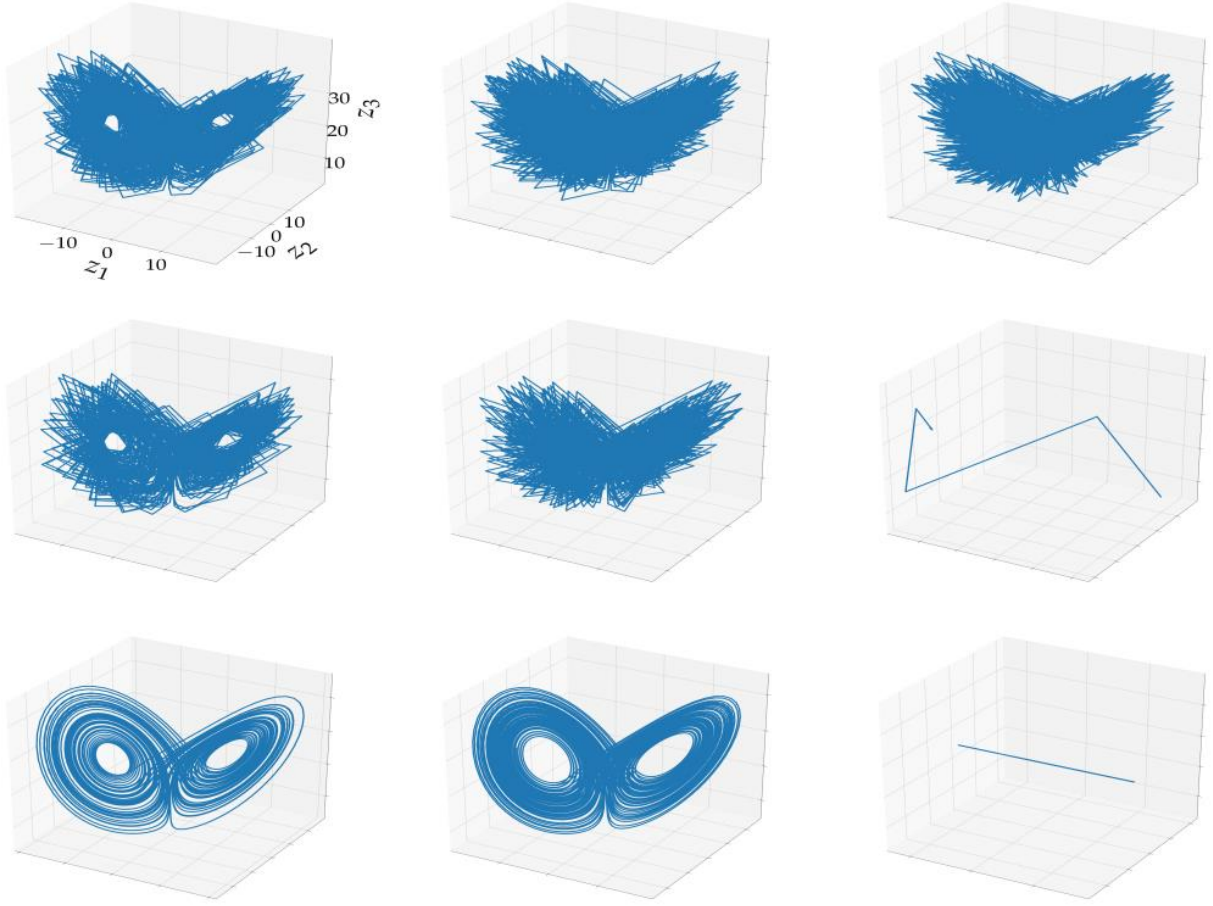
Learning dynamical systems 2 : training formulation

- Neural ODEs : Couple comments
- Comment 1) Which integration sch
- Comment 2) Let us use adaptive ste 2018)
- Storing every single activation of the adaptive step size solver (to do backprop) is expensive



Learning dynamical systems 2 : training formulation

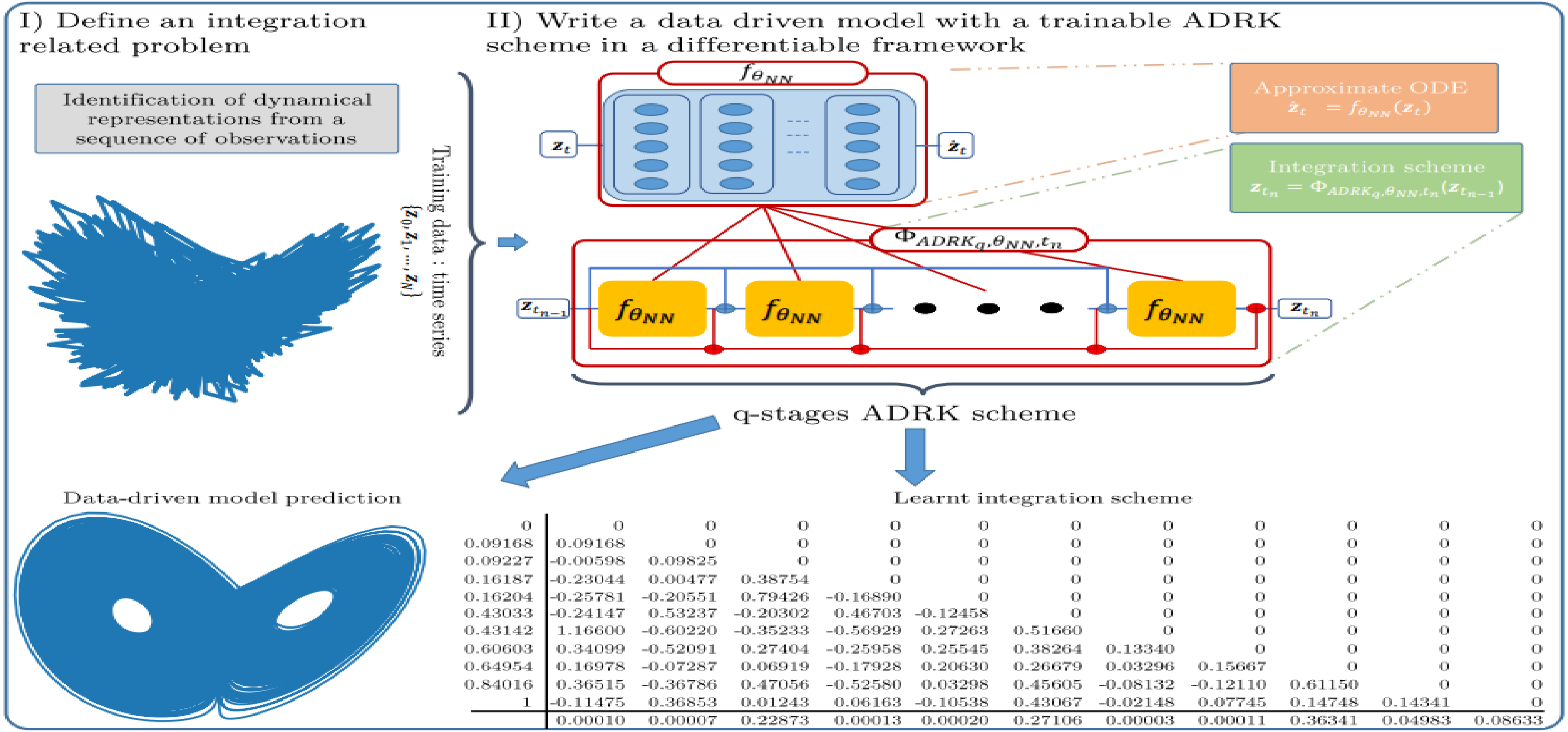
- Neural ODEs : Couple comn
- Comment 1) Which integrati
- Comment 2) Let us use adapti 2018)
- Using as proposed in (Chen et al. 2018) the adjoint sensitivity method to do the backward « freely » can lead to training instability (due to wrong gradients computation)



Learning dynamical systems 2 : training formulation

- Neural ODEs : Couple comments
- Comment 1) Which integration scheme to use ?
- Comment 2) Differentiation through an ODE solver, what does it mean ?
- Comment 3) Can we learn an « optimal » fixed step-size solver ?

Learning dynamical systems 2 : training


I) Define an integration related problem
Identification of dynamical representations from a sequence of observations
Training data : time series
$\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N\}$
II) Write a data driven model with a trainable ADRK scheme in a differentiable framework
$f_{\theta_{NN}}$
$\mathbf{z}_t$
$\dot{\mathbf{z}}_t$
Approximate ODE
$\dot{\mathbf{z}}_t = f_{\theta_{NN}}(\mathbf{z}_t)$
Integration scheme
$\mathbf{z}_{t_n} = \Phi_{ADRK_q, \theta_{NN}, t_n}(\mathbf{z}_{t_{n-1}})$
$\Phi_{ADRK_q, \theta_{NN}, t_n}$
$\mathbf{z}_{t_{n-1}}$
$f_{\theta_{NN}}$
$f_{\theta_{NN}}$
$f_{\theta_{NN}}$
$\mathbf{z}_{t_n}$
q-stages ADRK scheme
Data-driven model prediction
Learnt integration scheme

0	0	0	0	0	0	0	0	0	0	0	0
0.09168	0.09168	0	0	0	0	0	0	0	0	0	0
0.09227	-0.00598	0.09825	0	0	0	0	0	0	0	0	0
0.16187	-0.23044	0.00477	0.38754	0	0	0	0	0	0	0	0
0.16204	-0.25781	-0.20551	0.79426	-0.16890	0	0	0	0	0	0	0
0.43033	-0.24147	0.53237	-0.20302	0.46703	-0.12458	0	0	0	0	0	0
0.43142	1.16600	-0.60220	-0.35233	-0.56929	0.27263	0.51660	0	0	0	0	0
0.60603	0.34099	-0.52091	0.27404	-0.25958	0.25545	0.38264	0.13340	0	0	0	0
0.64954	0.16978	-0.07287	0.06919	-0.17928	0.20630	0.26679	0.03296	0.15667	0	0	0
0.84016	0.36515	-0.36786	0.47056	-0.52580	0.03298	0.45605	-0.08132	-0.12110	0.61150	0	0
1	-0.11475	0.36853	0.01243	0.06163	-0.10538	0.43067	-0.02148	0.07745	0.14748	0.14341	0
	0.00010	0.00007	0.22873	0.00013	0.00020	0.27106	0.00003	0.00011	0.36341	0.04983	0.08633

Learning dynamical systems 2 : training formulation, Recap

- In Neural ODE models, and in addition to the parameterization of the model, the choice of the integration scheme is important
- Fixed step size techniques : simple but can be restrictive
- Adaptive step-size techniques : versatile but can be subject to memory/stability issues
- Learning integration schemes : very fresh, not mature enough

Learning dynamical systems 3 : partial observations of the state space

- Phase space reconstruction
- Examples

Learning dynamical systems 3 : partial observations of the state space

- Let us assume that $\left\{x_{t_n}\right\}_{t_0}^{t_N}$ are measurements of an unknown time varying system :

$$\begin{cases} \dfrac{dz_t}{dt} = f(z_t) \\ x_t = H(z_t, \Omega_t, \epsilon_t) \end{cases}$$

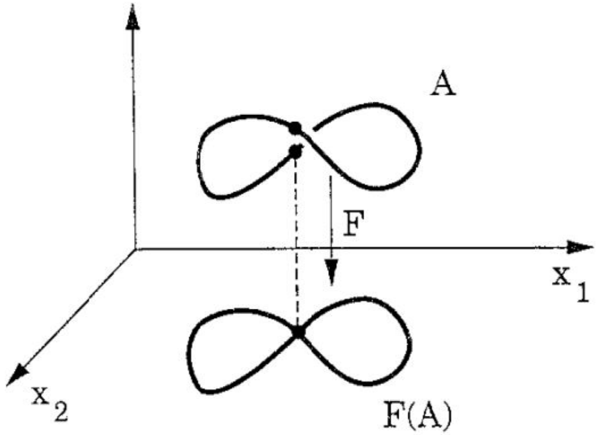
- If we assume $x_t = H(z_t)$, can we find a model $\frac{dx_t}{dt} = f_\theta(x_t)$

Learning dynamical systems 3 : partial observations of the state space

- In order to write $\frac{dx_t}{dt} = f_\theta(x_t)$ we need to make sure that H is an embedding of z_t

Learning dynamical systems 3 : partial observations of the state space

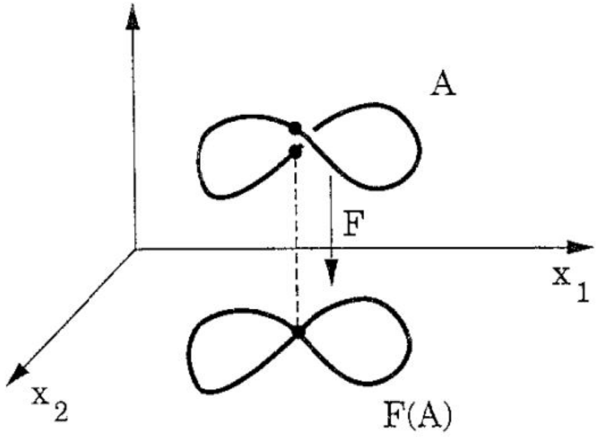
- In order to write $\frac{dx_t}{dt} = f_\theta(x_t)$ we need to make sure that H is an embedding of z_t



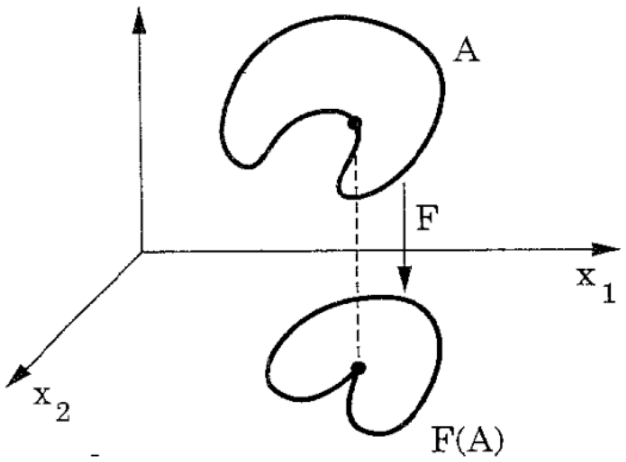
Source: (Sauer et al. (1991))
$\mathcal{H}$ is not one-to-one

Learning dynamical systems 3 : partial observations of the state space

- In order to write $\frac{dx_t}{dt} = f_\theta(x_t)$ we need to make sure that H is an embedding of z_t



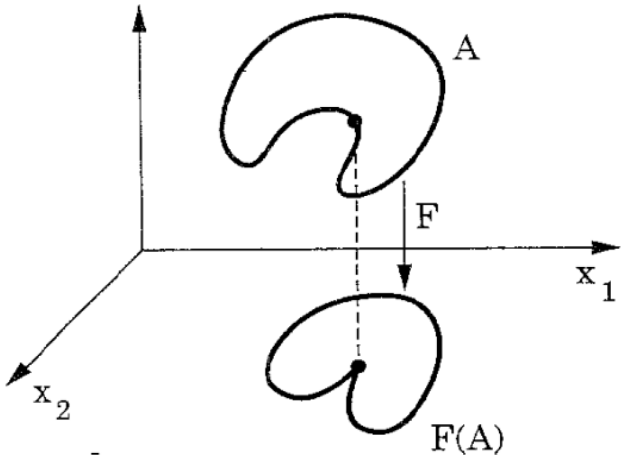
Source: (Sauer et al. (1991))
$\mathcal{H}$ is not one-to-one



Source: (Sauer et al. (1991))
$\mathcal{H}$ is not an immersion of $\mathbf{z}$

Learning dynamical systems 3 : partial observations of the state space

- In order to write $\frac{dx_t}{dt} = f_\theta(x_t)$ we need to make sure that H is an embedding of z_t
- If H is an embedding of z_t than we can do prediction ^^ (using a deterministic model at least)



Source: (Sauer et al. (1991))
$\mathcal{H}$ is not an immersion of $\mathbf{z}$

Learning dynamical systems 3 : partial observations of the state space

- Simple, motivating example :

$$\begin{cases} \dfrac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

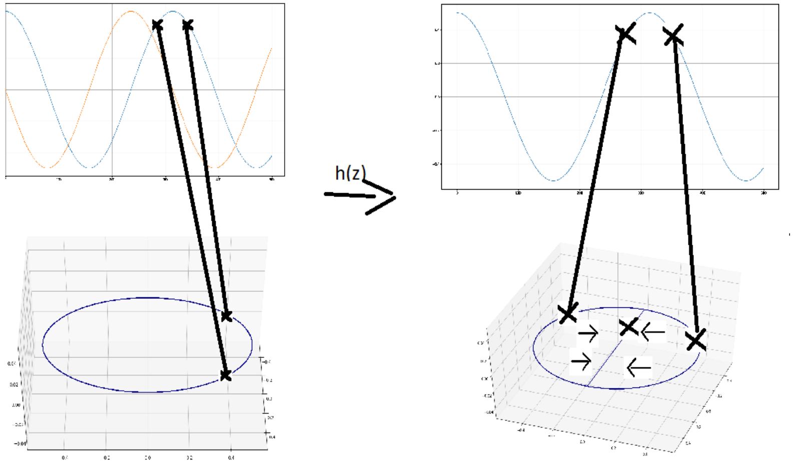
- Is this an embedding ?

Learning dynamical systems 3 : partial observations of the state space

- Simple, motivating example :

$$\begin{cases} \dfrac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

- Is this an embedding ?



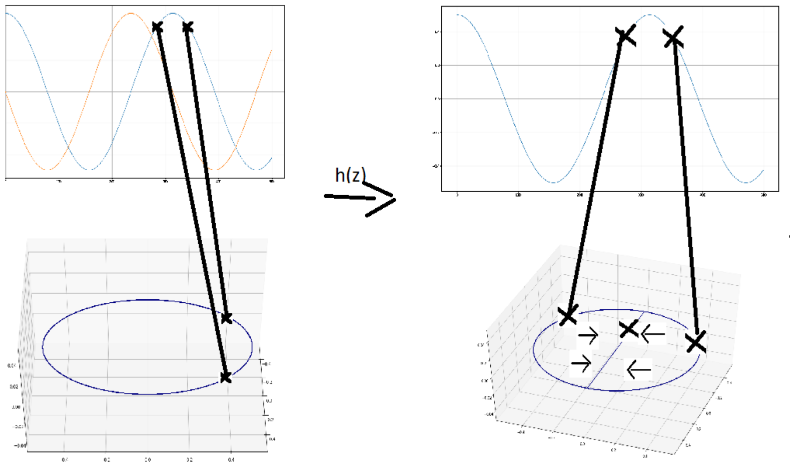
Learning dynamical systems 3 : partial observations of the state space

- Simple, motivating example :

$$\begin{cases} \dfrac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

- Is this an embedding ?

- What happens if we try to fit $\frac{dx_t}{dt} = f_\theta(x_t)$?



Learning dynamical systems 3 : partial observations of the state space

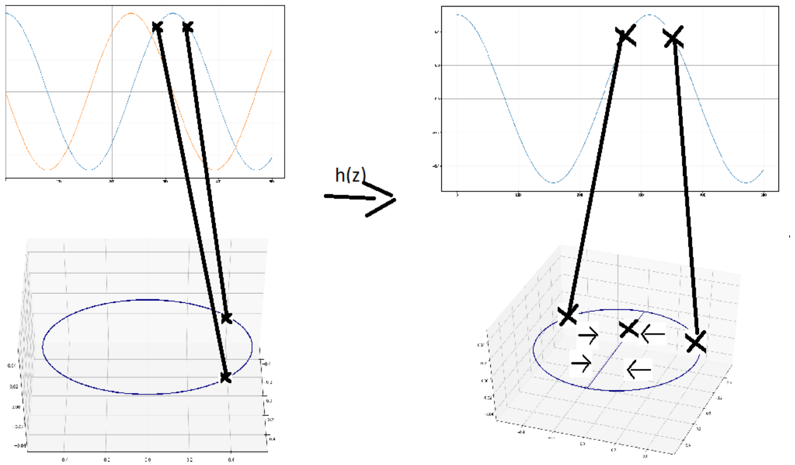
- Simple, motivating example :

$$\begin{cases} \dfrac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

- Is this an embedding ?

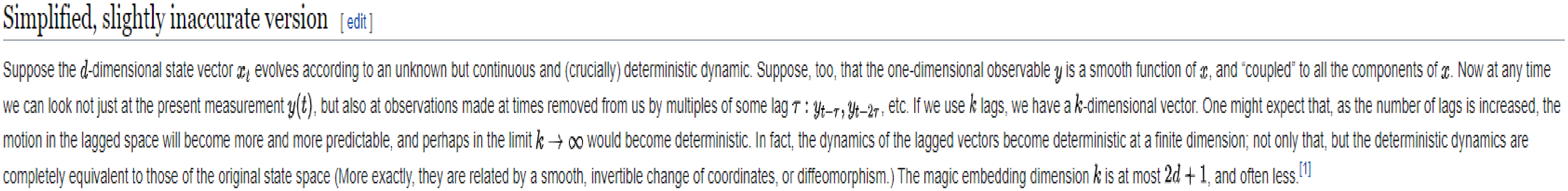
- What happens if we try to fit $\frac{dx_t}{dt} = f_\theta(x_t)$?

- What to do then ?



Learning dynamical systems 3 : partial observations of the state space

- Attractor reconstruction using takens delay embedding :



Simplified, slightly inaccurate version [edit]

Suppose the d-dimensional state vector x_t evolves according to an unknown but continuous and (crucially) deterministic dynamic. Suppose, too, that the one-dimensional observable y is a smooth function of x, and "coupled" to all the components of x. Now at any time we can look not just at the present measurement $y(t)$, but also at observations made at times removed from us by multiples of some lag $\tau : y_{t-\tau}, y_{t-2\tau}$, etc. If we use k lags, we have a k-dimensional vector. One might expect that, as the number of lags is increased, the motion in the lagged space will become more and more predictable, and perhaps in the limit $k \to \infty$ would become deterministic. In fact, the dynamics of the lagged vectors become deterministic at a finite dimension; not only that, but the deterministic dynamics are completely equivalent to those of the original state space (More exactly, they are related by a smooth, invertible change of coordinates, or diffeomorphism.) The magic embedding dimension k is at most $2d + 1$, and often less.[1]

Learning dynamical systems 3 : partial observations of the state space

- Attractor reconstruction using takens delay embedding :

- Let us go back to our simple example :

$$\begin{cases} \dfrac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

- Embedding x_t : $u_t = \left[x_t, x_{t-\tau}, x_{t-2\tau}, \dots, x_{t-(k-1)\tau}\right] \in \mathbb{R}^k$

- Let us compare u_t and z_t

Learning dynamical systems 3 : partial observations of the state space

- Attractor reconstruction using takens delay embedding :

- Let us go back to our simple example :

$$\begin{cases} \frac{dz_t}{dt} = \lambda z_t, \lambda \in \mathbb{C}, \mathrm{imag}(\lambda) \neq 0 \\ x_t = Real(z_t) \end{cases}$$

- Embedding $x_t : u_t = \left[x_t, x_{t-\tau}, x_{t-2\tau}, \dots, x_{t-(k-1)\tau}\right] \in \mathbb{R}^k$

- Let us compare u_t and z_t

- Do prediction on u_t and not on x_t

Learning dynamical systems 3 : partial observations of the state space, Recap

- In realistic applications, you need to, almost systematically do delay embedding (or some sort of an embedding)

- Questions ? How to chose the time delay τ and the dimension of the embedding k?

- What are RNNs in this context ?

Learning dynamical systems 4 : model evaluation

- How do we compare data-driven models ?
- Prediction/forecast vs simulation applications
- Limit-sets and evaluation metrics

Learning dynamical systems 4 : model evaluation, forecast applications

- Just divide your model into training and testing sets

- Compare your forecasted fields (up to a given prediction time step) with respect to the ground truth (RMSE or others)

Learning dynamical systems 4 : model evaluation, simulation applications

- In simulation applications, we need to make sure that the long term predictions of the model converge to the limit-set of the data

- limit-set : the asymptotic behavior of dynamical systems

- The main question is : can we reproduce a limit-set of the observations using the data-driven model

- An other question is : if we can we reproduce a limit-set, what is the range of initial conditions that lead to this limit-set ?

Learning dynamical systems 4 : model evaluation, Limit-sets

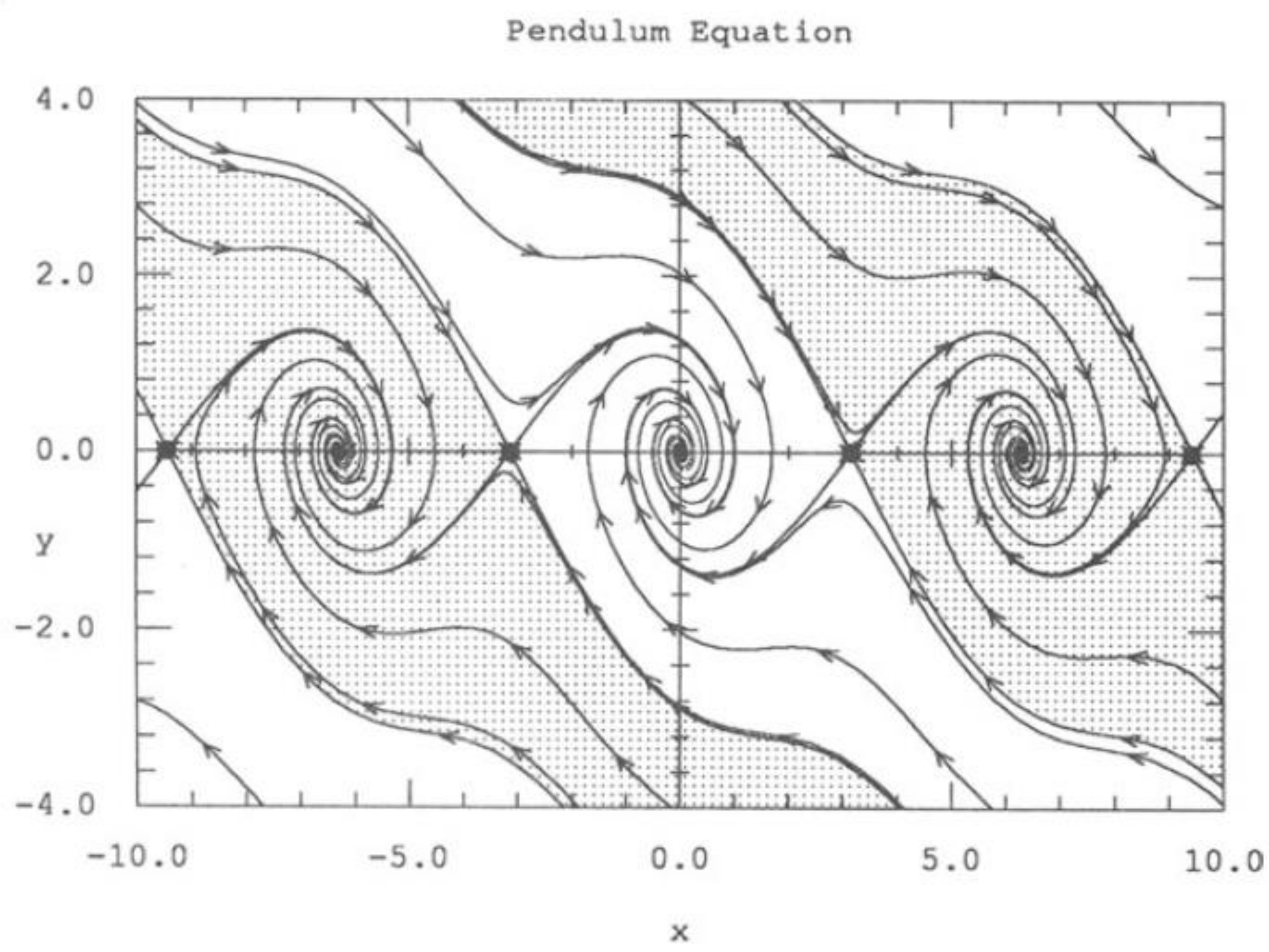
- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Equilibrium points : An equilibrium point z_{eq} is a solution that cancels the vector field i.e. $f\left(z_{eq}\right) = 0$ ➔ $z_{eq} = \Phi_t(z_{eq})$

$$\frac{dz_{1,t}}{dt} = z_2$$
$$\frac{dz_{2,t}}{dt} = -\epsilon z_{2,t} - \sin(z_{1,t})$$

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us co
- Equilibri ıat cancels t



Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Periodic solutions : Definition : A periodic solution of an ODE verifies $z_t = \Phi_{t+T}(z_t)$ with T>0

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Periodic solutions : Definition : A periodic solution of an ODE verifies $z_t = \Phi_{t+T}(z_t)$ with T>0

- Example : VDP equation

$$\frac{dz_{1,t}}{dt} = z_{2,t}$$
$$\frac{dz_{2,t}}{dt} = (1 - z_{1,t}^2)z_{2,t} - z_{1,t}$$

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$
- Periodic solutions : Definition : A periodic solution of an ODE verifies $z_t = \Phi_{t+T}(z_t)$ with T>0
- Periodic solutions : Limit-set : The limit set of a periodic solution is a closed curve in the phase space

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Periodic solutions : Definition : A periodic solution of an ODE verifies $z_t = \Phi_{t+T}(z_t)$ with T>0

- Periodic solutions : Limit-set : The limit set of a periodic solution is a closed curve in the phase space

- Periodic solutions : Spectrum : The Spectrum of a periodic solution contains spikes at integer multiples of the fundamental frequency

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Quasi-Periodic solutions : Definition : A Quasi-periodic solution of an ODE verifies $z_t = \sum_{i=1}^{k} h_{i,t}$ with $h_{i,t}$ are periodic functions with frequencies f_i>0

- The frequencies $f_i = \left|\sum_{n=1}^{p} k_n f_n'\right|$ where $\{f_1', f_2', \dots, f_p'\}$ is a linearly independent basis of frequencies

- Example :

$$z_t = \cos(2\pi f_1 t) + \cos(2\pi f_2 t)$$

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$
- Quasi-Periodic solutions : Definition : A Quasi-periodic solution of an ODE verifies $z_t = \sum_{i=1}^{k} h_{i,t}$ with $h_{i,t}$ are periodic functions with frequencies f_i>0
- The frequencies $f_i = \left|\sum_{n=1}^{p} k_n f_n'\right|$ where $\{f_1', f_2', \dots, f_p'\}$ is a linearly independent basis of frequencies
- Quasi-Periodic solutions : Limit-set : The limit set of quasi-Periodic solution is a torus in the phase space

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Quasi-Periodic solutions : Definition : A Quasi-periodic solution of an ODE verifies $z_t = \sum_{i=1}^{k} h_{i,t}$ with $h_{i,t}$ are periodic functions with frequencies f_i>0

- The frequencies $f_i = \left|\sum_{n=1}^{p} k_n f_n'\right|$ where $\{f_1', f_2', \dots, f_p'\}$ is a linearly independent basis of frequencies

- Quasi-Periodic solutions : Limit-set : The limit set of quasi-Periodic solution is a torus in the phase space

- Quasi-Periodic solutions : Spectrum : The Spectrum of quasi-Periodic solution contains spikes at integer multiples of f_n'

Learning dynamical systems 4 : model evaluation, Limit-sets

- Let us consider the following ODE $\frac{dz_t}{dt} = f(z_t)$

- Chaotic solutions : Everything that is bounded but not an equilibrium, periodic or Quasi-Periodic solutions :

- Chaotic solutions : Limit-set : The limit set of chaotic solutions is a strange attractor

- Chaotic solutions : Spectrum : continuous spectrum, may contain spikes
- Example : Lorenz 63 system

$$\left\{\begin{array}{lcl} \frac{d\mathbf{z}_{t,1}}{dt} & = & \sigma\left(\mathbf{z}_{t,2} - \mathbf{z}_{t,2}\right) \\ \frac{d\mathbf{z}_{t,2}}{dt} & = & \rho\mathbf{z}_{t,1} - \mathbf{z}_{t,2} - \mathbf{z}_{t,1}\mathbf{z}_{t,3} \\ \frac{d\mathbf{z}_{t,3}}{dt} & = & \mathbf{z}_{t,1}\mathbf{z}_{t,2} - \beta\mathbf{z}_{t,3} \end{array}\right.$$

Learning dynamical systems 4 : model evaluation, Recap

- In simulation applications, we need to make sure that a simulated trajectory gives the same asymptotic behaviour as the unknown equation

- Spectrum comparisons ^^’’

- In real applications, getting a correct asymptotic behaviour of data-driven models is extremely difficult

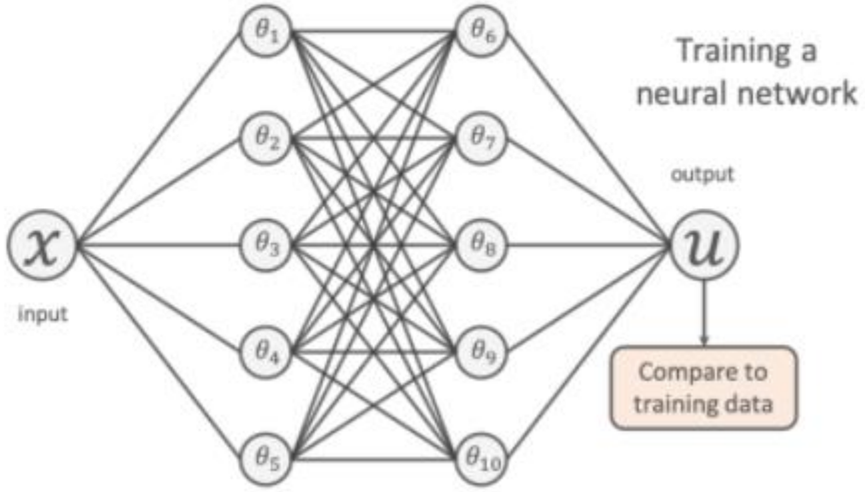
- Solution : Physics informed AI

Learning dynamical systems 5 : Physics informed AI

- Physics Informed Neural Networks (PINN)
- Neural networks for closure modeling

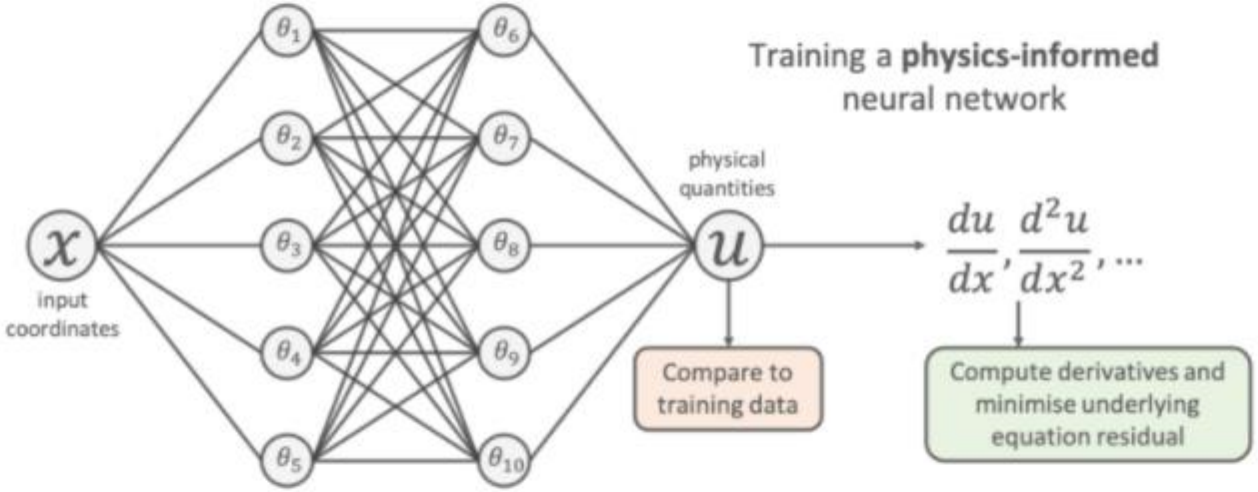
Learning dynamical systems 5 : Physics informed AI, PINN

- Classical neural networks models :



Learning dynamical systems 5 : Physics informed AI, PINN

- PINN :



Training a **physics-informed** neural network

Compare to training data

Compute derivatives and minimise underlying equation residual

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Closure modeling, motivating example : Shallow water equations

$$\begin{cases} \left.\begin{aligned} \frac{\partial u}{\partial t} - F_v = -g\frac{\partial \eta}{\partial x} \\ \frac{\partial v}{\partial t} - F_u = -g\frac{\partial \eta}{\partial y} \end{aligned}\right\} \\ \left.\frac{\partial \eta}{\partial t} + \frac{\partial(\eta + H)u}{\partial x} + \frac{\partial(\eta + H)v}{\partial y} \quad = 0 \right\} \end{cases}$$

Momentum equations are taken to be linear

The continuity equation is solved in its non-linear form

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Let us assume that the state vector can be decomposed as follows:

$$\begin{cases} u = u' + \bar{u} \\ v = v' + \bar{v} \\ \eta = \underbrace{\eta'}_{\text{High resolution, unobserved component}} + \underbrace{\bar{\eta}}_{\text{Observed, large scale component}} \end{cases}$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Let us assume that the state vector can be decomposed as follows:

$$\begin{cases} u = u' + \bar{u} \\ v = v' + \bar{v} \\ \eta = \underbrace{\eta'}_{\text{High resolution, unobserved component}} + \underbrace{\bar{\eta}}_{\text{Observed, large scale component}} \end{cases}$$

How to model the large scale component ?

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Plugging this decomposition into the SWE yields

$$
\begin{aligned}
&\frac{\partial \bar{u}}{\partial t} + \frac{\partial u'}{\partial t} - F_v = -g(\frac{\partial \bar{\eta}}{\partial x} + \frac{\partial \eta'}{\partial x}) \\
&\frac{\partial \bar{v}}{\partial t} + \frac{\partial v'}{\partial t} - F_u = -g(\frac{\partial \bar{\eta}}{\partial y} + \frac{\partial \eta'}{\partial y}) \\
&\frac{\partial \bar{\eta}}{\partial t} + \frac{\partial \eta'}{\partial t} + \frac{\partial (\bar{\eta} + H)\bar{u}}{\partial x} + \frac{\partial (\bar{\eta} + H)\bar{v}}{\partial y} \\
&+ \frac{\partial (\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial (\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} = 0
\end{aligned}
$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Plugging this decomposition into the SWE yields:

$$\frac{\partial \bar{u}}{\partial t} + \boxed{\frac{\partial u'}{\partial t}} - F_v = -g(\frac{\partial \bar{\eta}}{\partial x} + \boxed{\frac{\partial \eta'}{\partial x}})$$

$$\frac{\partial \bar{v}}{\partial t} + \boxed{\frac{\partial v'}{\partial t}} - F_u = -g(\frac{\partial \bar{\eta}}{\partial y} + \boxed{\frac{\partial \eta'}{\partial y}})$$

$$\frac{\partial \bar{\eta}}{\partial t} + \boxed{\frac{\partial \eta'}{\partial t}} + \frac{\partial (\bar{\eta} + H)\bar{u}}{\partial x} + \frac{\partial (\bar{\eta} + H)\bar{v}}{\partial y}$$

$$\boxed{+ \frac{\partial (\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial (\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x}} = 0$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Applying an averaging operator and assuming that f' is zero :

$$\frac{\partial \bar{u}}{\partial t} - \bar{F}_v = -g\frac{\partial \bar{\eta}}{\partial x}$$

$$\frac{\partial \bar{v}}{\partial t} - \bar{F}_u = -g\frac{\partial \bar{\eta}}{\partial y}$$

$$\frac{\partial \bar{\eta}}{\partial t} + \frac{\partial(\bar{\eta}+H)\bar{u}}{\partial x} + \frac{\partial(\bar{\eta}+H)\bar{v}}{\partial y}$$

$$\boxed{+\frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial \eta'\bar{u}}}{\partial x} + \frac{\overline{\partial \eta' u'}}{\partial x} + \frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial \eta'\bar{u}}}{\partial x} + \frac{\overline{\partial \eta' u'}}{\partial x}} = 0$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Applying an averaging operator and assuming tl f' is zero :

$$\left.\begin{aligned}\frac{\partial \bar{u}}{\partial t} - \bar{F}_v = -g\frac{\partial \bar{\eta}}{\partial x}\\ \frac{\partial \bar{v}}{\partial t} - \bar{F}_u = -g\frac{\partial \bar{\eta}}{\partial y}\end{aligned}\right\}$$

Momentum equations are linear, the HR terms are averaged to zero

$$\frac{\partial \bar{\eta}}{\partial t} + \frac{\partial(\bar{\eta}+H)\bar{u}}{\partial x} + \frac{\partial(\bar{\eta}+H)\bar{v}}{\partial y}$$

$$+ \frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial\eta'\bar{u}}}{\partial x} + \frac{\overline{\partial\eta'u'}}{\partial x} + \frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial\eta'\bar{u}}}{\partial x} + \frac{\overline{\partial\eta'u'}}{\partial x} = 0$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Applying an averaging operator and assuming tl f' is zero :

$$\left.\begin{aligned}\frac{\partial \bar{u}}{\partial t} - \bar{F}_v &= -g\frac{\partial \bar{\eta}}{\partial x}\\ \frac{\partial \bar{v}}{\partial t} - \bar{F}_u &= -g\frac{\partial \bar{\eta}}{\partial y}\end{aligned}\right\}$$

Mom

line

$$\frac{\partial \bar{\eta}}{\partial t} + \frac{\partial(\bar{\eta}+H)\bar{u}}{\partial x} + \frac{\partial(\bar{\eta}+H)\bar{v}}{\partial y}$$

$$\overbrace{\boxed{+\overline{\frac{\partial(\bar{\eta}+H)u'}{\partial x}} + \overline{\frac{\partial \eta'\bar{u}}{\partial x}} + \overline{\frac{\partial \eta' u'}{\partial x}} + \overline{\frac{\partial(\bar{\eta}+H)u'}{\partial x}} + \overline{\frac{\partial \eta'\bar{u}}{\partial x}} + \overline{\frac{\partial \eta' u'}{\partial x}}}} = 0$$

Due to the non linearity in the continuity equation, the HR terms can not be averaged and need to be parametrized

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

Applying an averaging operator and assuming tl f' is zero :

$$\left.\begin{aligned}\frac{\partial \bar{u}}{\partial t} - \bar{F}_v &= -g\frac{\partial \bar{\eta}}{\partial x}\\ \frac{\partial \bar{v}}{\partial t} - \bar{F}_u &= -g\frac{\partial \bar{\eta}}{\partial y}\end{aligned}\right\}$$

Mom

line

$$\frac{\partial \bar{\eta}}{\partial t} + \frac{\partial(\bar{\eta}+H)\bar{u}}{\partial x} + \frac{\partial(\bar{\eta}+H)\bar{v}}{\partial y}$$
$$\overbrace{\boxed{+\frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial\eta'\bar{u}}}{\partial x} + \frac{\overline{\partial\eta' u'}}{\partial x} + \frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x} + \frac{\overline{\partial\eta'\bar{u}}}{\partial x} + \frac{\overline{\partial\eta' u'}}{\partial x}}}= 0$$

Due to the non linearity in the continuity equation, the HR terms can not be averaged and need to be parametrized

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

What do this decomposition means ?

$$\begin{cases} u = u' + \bar{u} \\ v = v' + \bar{v} \\ \eta = \underbrace{\eta'}_{\text{High resolution, unobserved component}} + \underbrace{\bar{\eta}}_{\text{Observed, large scale component}} \end{cases}$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

What do this decomposition means ?

$$\begin{cases} u = u' + \bar{u} \\ v = v' + \bar{v} \\ \eta = \underbrace{\eta'}_{\text{High resolution, unobserved component}} + \underbrace{\bar{\eta}}_{\text{Observed, large scale component}} \end{cases}$$

$$\begin{cases} \bar{u} = u - u' \\ \bar{v} = v - v' \\ \bar{\eta} = \eta - \eta' \end{cases}$$

It is a projection from the HR space to the LR one

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

In order to parametrize

$$+\frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x}+\frac{\overline{\partial\eta'\bar{u}}}{\partial x}+\frac{\overline{\partial\eta'u'}}{\partial x}+\frac{\overline{\partial(\bar{\eta}+H)u'}}{\partial x}+\frac{\overline{\partial\eta'\bar{u}}}{\partial x}+\frac{\overline{\partial\eta'u'}}{\partial x}=0$$

We need to know/study the properties of this projection and especially:

$$\begin{cases}\bar{u}=u-u'\\ \bar{v}=v-v'\\ \bar{\eta}=\eta-\eta'\end{cases}$$

- Is this projection one-to-one ?

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

If $$\begin{cases} \bar{u} = u - u' \\ \bar{v} = v - v' \\ \bar{\eta} = \eta - \eta' \end{cases}$$ is one-to-one, we can find a map from the LR space to the HR space

In this situation, we can parameterize our closure as a function of the LR states

$$\frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} = f_\theta(\bar{u}, \bar{v}, \bar{\eta})$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

If $\begin{cases} \bar{u} = u - u' \\ \bar{v} = v - v' \\ \bar{\eta} = \eta - \eta' \end{cases}$ is one-to-one, we can find a map from the LR space to the HR space

In this situation, we can parameterize our closure as a function of the LR states

$$\frac{\partial(\bar{\eta}+H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial(\bar{\eta}+H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} = f_\theta(\bar{u}, \bar{v}, \bar{\eta})$$

Good for short term forecast and simulation of the LR equation

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

However $\begin{cases} \bar{u} = u - u' \\ \bar{v} = v - v' \\ \bar{\eta} = \eta - \eta' \end{cases}$ is most of the time not one-to-one

In this situation such parameterization can lead to poor results, especially in simulation

$$\frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} = f_\theta(\bar{u}, \bar{v}, \bar{\eta})$$

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

However $\begin{cases} \bar{u} = u - u' \\ \bar{v} = v - v' \\ \bar{\eta} = \eta - \eta' \end{cases}$ is most of the time not one-to-one

In this situation such parameterization can lead to poor results, especially in simulation

$$\frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} + \frac{\partial(\bar{\eta} + H)u'}{\partial x} + \frac{\partial \eta' \bar{u}}{\partial x} + \frac{\partial \eta' u'}{\partial x} = f_\theta(\bar{u}, \bar{v}, \bar{\eta})$$

Since the closure term can not be inferred (deterministically) from the LR states

Learning dynamical systems 5 : Physics informed AI, NN for closure modeling

- If our time series are long enough, we can reconstruct a diffeomorphic copy of the HR states from a collection of LR time series
- A straightforward embedding can be a delay embedding (parametrized by a RNN)
- In practice, the closure terms can be parametrized by a RNN (Charalampopoulos et al 2019)
- Examples ^^

Outline

1. An naive, brief introduction to Dynamical Systems
   - Introduction
   - State space models and Learning formulation
2. Resolution of differential equations : numerical integration
3. Training dynamical systems
   - Continuous time setting
   - Discrete time setting
4. Partial observations of the state space
   - Phase space reconstruction
   - Examples
5. Model evaluation
   - How do we compare data-driven models ?
   - Prediction/forecast vs simulation applications
   - Limit-sets and evaluation metrics
6. Physics informed AI
   - Physics Informed Neural Networks (PINN)
   - Neural networks for closure modeling

Refs

- Parker, T. S., & Chua, L. (2012). *Practical numerical algorithms for chaotic systems*. Springer Science & Business Media.
- Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, *113*(15), 3932-3937.
- Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

Refs

- Ouala, S., Debreu, L., Pascual, A., Chapron, B., Collard, F., Gaultier, L., & Fablet, R. (2021). Learning Runge-Kutta Integration Schemes for ODE Simulation and Identification. *arXiv preprint arXiv:2105.04999*.
- Ouala, S., Nguyen, D., Drumetz, L., Chapron, B., Pascual, A., Collard, F., ... & Fablet, R. (2020). Learning latent dynamics for partially observed chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *30*(10), 103121.
- Sauer, Tim, James A. Yorke, and Martin Casdagli. "Embedology." *Journal of statistical Physics* 65.3 (1991): 579-616.

Refs

- Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational Physics* 378 (2019): 686-707.
- Charalampopoulos, Alexis-Tzianni. *Machine learning non-local closures for turbulent anisotropic multiphase flows*. Diss. Massachusetts Institute of Technology, 2019.