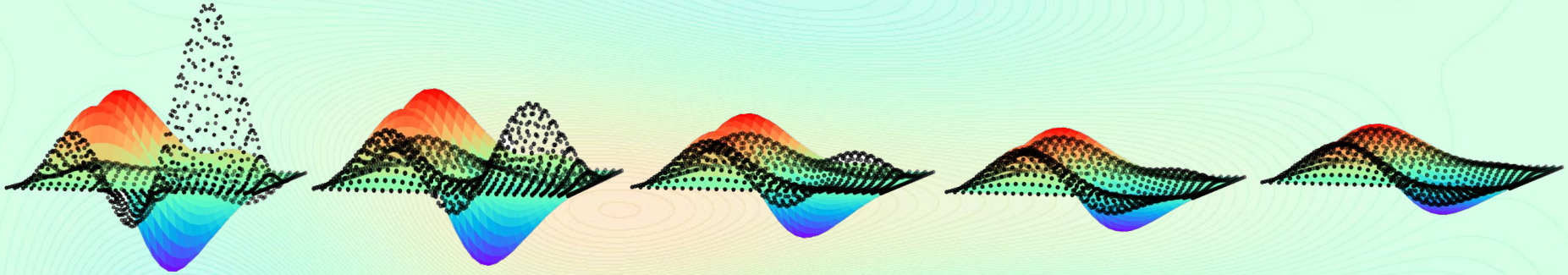
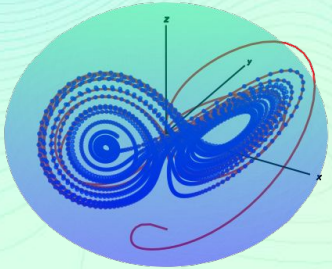
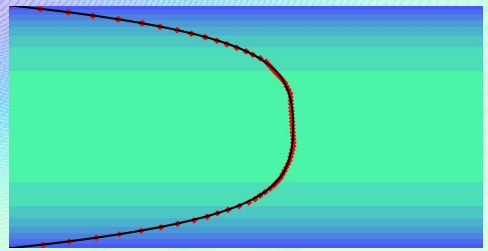
Data Assimilation with DeepONet



Learning Solutions to Differential Equations With Missing Initial Data

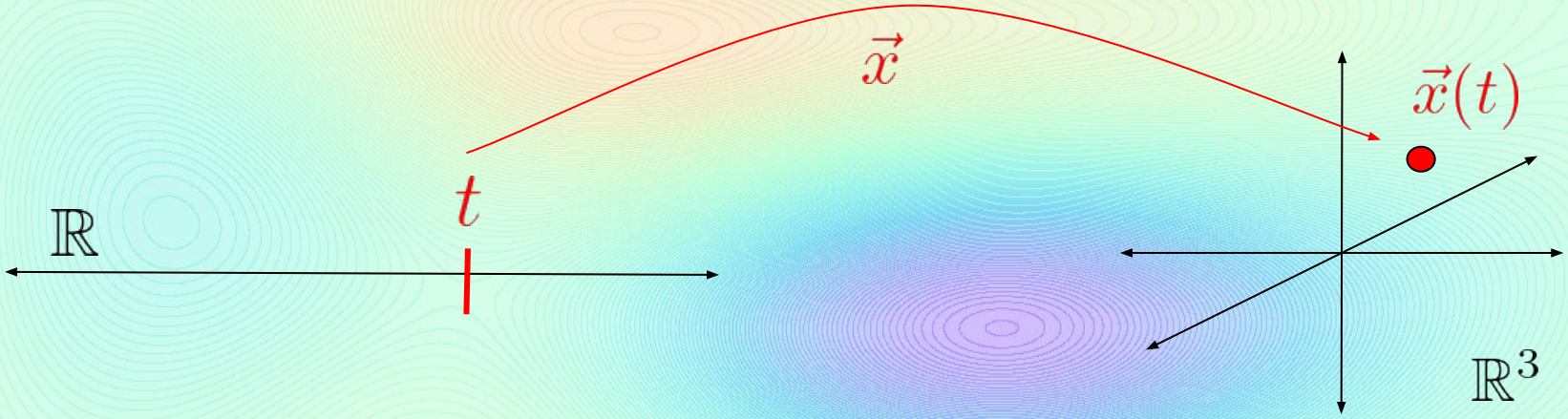


Gary Guzzo
Math 881
Dr. Tian



Differential Equations (1/5)

Define $\vec{x} : \mathbb{R} \to \mathbb{R}^3$ by $\vec{x}(t) = \big(x(t), y(t), z(t)\big)$



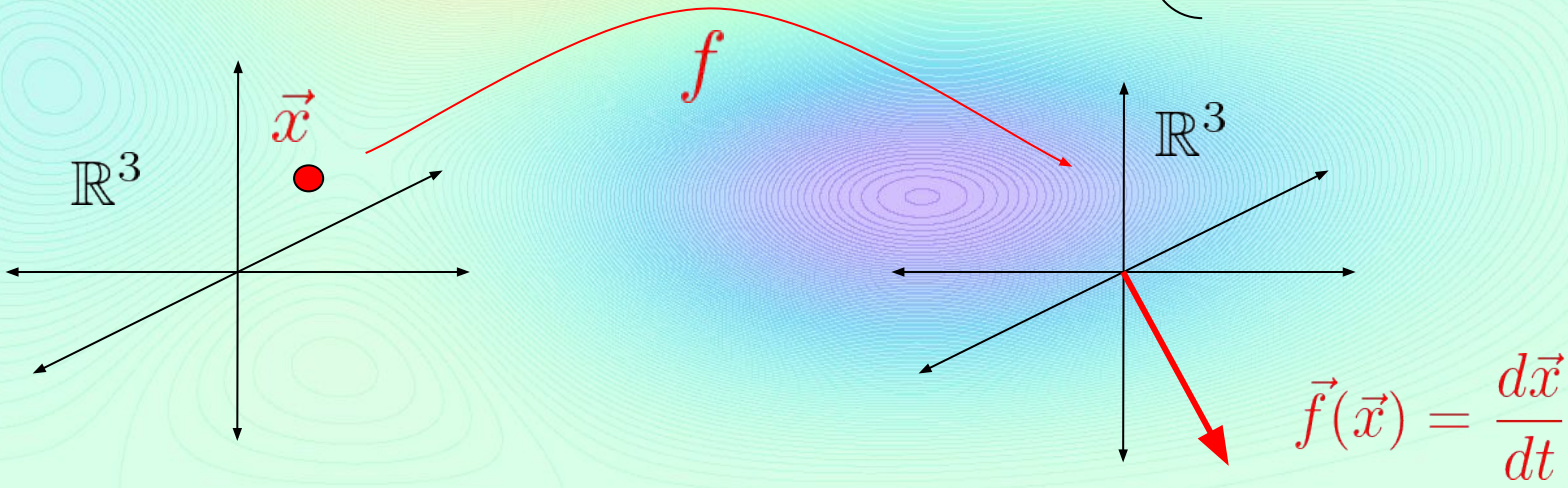
Solutions to ODEs will be of the form of $\vec{x}(t)$

Differential Equations (2/5)

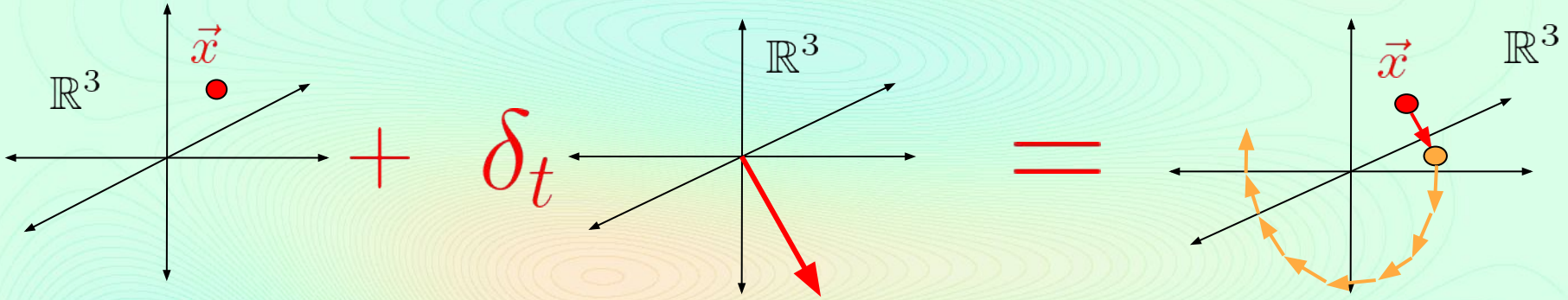
Let $\vec{x} = (x, y, z)$

Define $\vec{f} : \mathbb{R}^3 \to \mathbb{R}^3$ by $\vec{f}(\vec{x}) = \dfrac{d\vec{x}}{dt}$

$$\begin{cases} \dfrac{dx}{dt} = f_1(x, y, z) \\ \dfrac{dy}{dt} = f_2(x, y, z) \\ \dfrac{dz}{dt} = f_3(x, y, z) \end{cases}$$



Differential Equations (3/5)



$$\vec{x}_n \longrightarrow \begin{cases} \dfrac{dx}{dt} = f_1(x,y,z) \\ \dfrac{dy}{dt} = f_2(x,y,z) \\ \dfrac{dz}{dt} = f_3(x,y,z) \end{cases}$$

$$\vec{x}_{n+1} = \vec{x}_n + \delta_t \vec{f}(\vec{x})$$

Derivative is *known*: Plug in, take linear combo

Differential Equations (4/5)

$$\text{Solve } \frac{d\vec{x}}{dt} = \vec{f}(\vec{x}) \; ; \; \vec{x}(0) = \vec{x}_0 \in \mathbb{R}^3$$

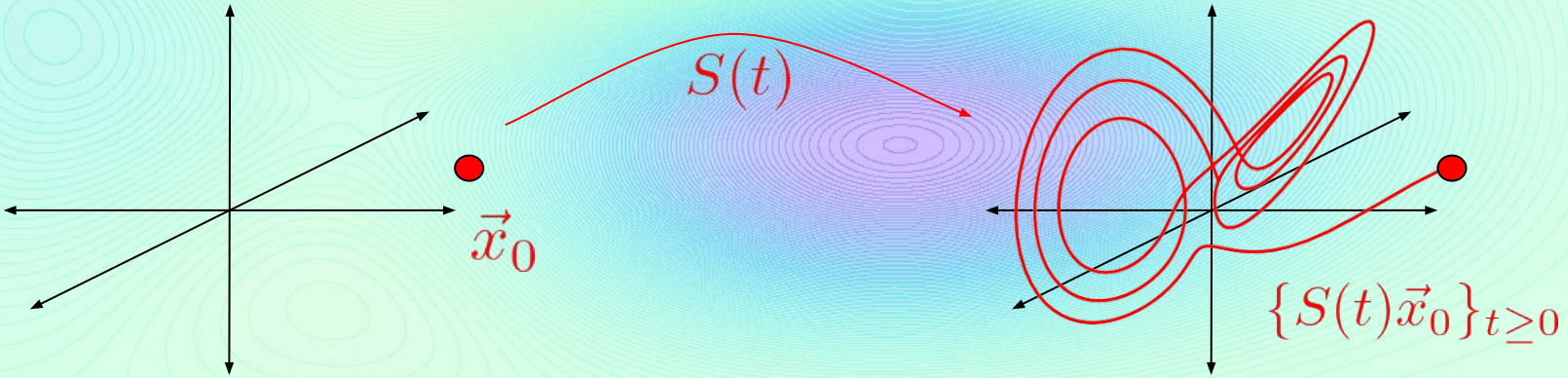
$$\vec{x}(t; \vec{x}_0) = \vec{x}_0 + \int_0^t \vec{f}(\vec{x}(s))\vec{ds}$$

Differential Equations (5/5)

Define $\forall t \in \mathbb{R}$ the **solution operator** $S(t) : \mathbb{R}^3 \to \mathbb{R}^3$ by

$$S(t)\vec{x}_0 := \vec{x}(t; \vec{x}_0)$$

Then $\{S(t)\}_{t\in\mathbb{R}}$ gives a **trajectory**



Lorenz System (1/9)

Initial value problem: $\dfrac{d\vec{x}}{dt} = L(\vec{x}) \;\; ; \;\; \vec{x}(0) = \vec{x}_0$

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

$\sigma = 10, \;\; \rho = 28, \;\; \beta = 8/3$

Lorenz System (2/9)

We numerically solve the recursion

$$\vec{x}(t_{n+1}) = \vec{x}(t_n) + \delta_t \frac{d}{dt}\vec{x}(t_n), \quad n = 0, 1, 2, \ldots$$

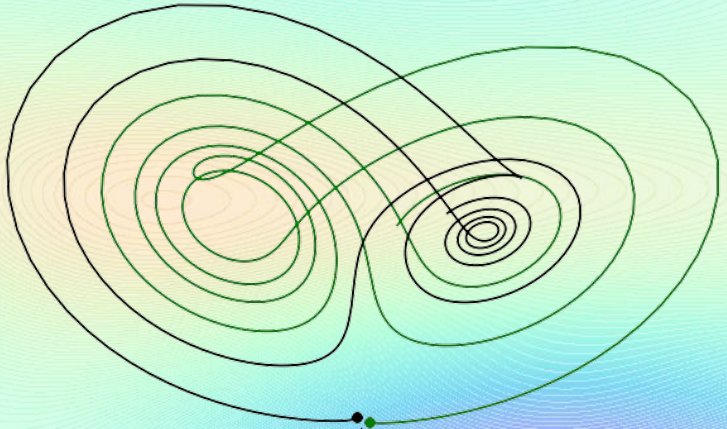
Solution operator $S(t_n) : \mathbb{R}^3 \to \mathbb{R}^3$

$$S(t_n)\vec{x}_0 := \vec{x}_0 + \sum_{i=0}^{n} L(\vec{x}_i)\delta_t$$

Lorenz System (3/9)

Chaos; although $\vec{x}_0$ and $\vec{y}_0$ are close, . . .

$\rho = 28$



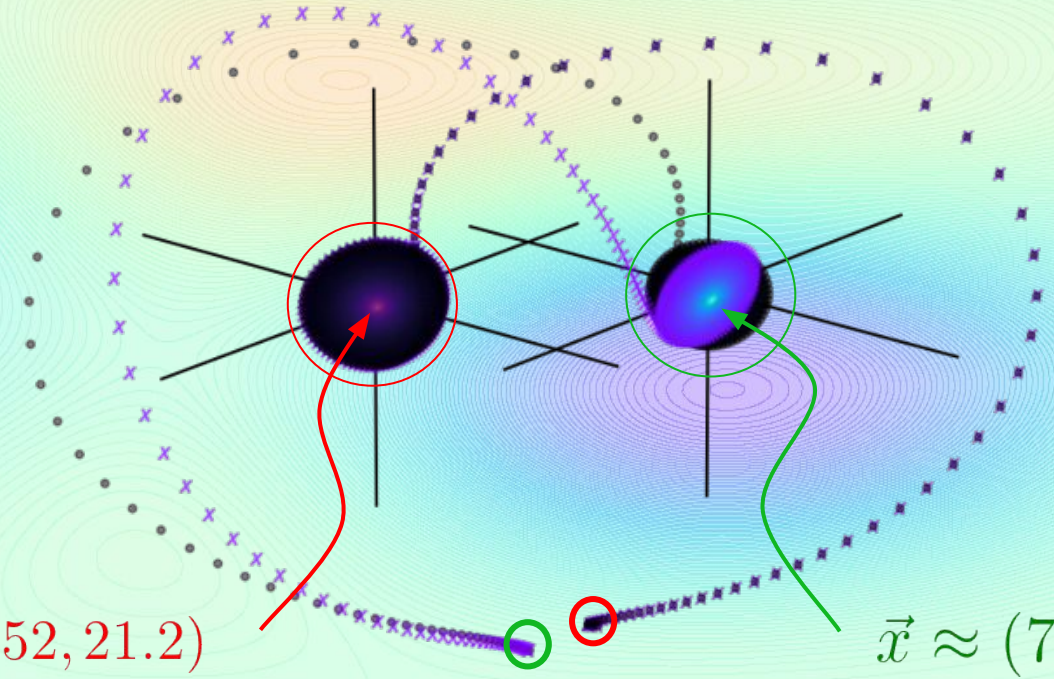
$$||\vec{x}_0 - \vec{y}_0|| < 1$$

. . . the trajectories $S(t)\vec{x}_0$ and $S(t)\vec{y}_0$ are not

Lorenz System (4/9)

Stability; if $\vec{x}_0$ and $\vec{y}_0$ are close, . . .

. . . then we expect the trajectories to be close



$\rho = 22.2$

[Clack]

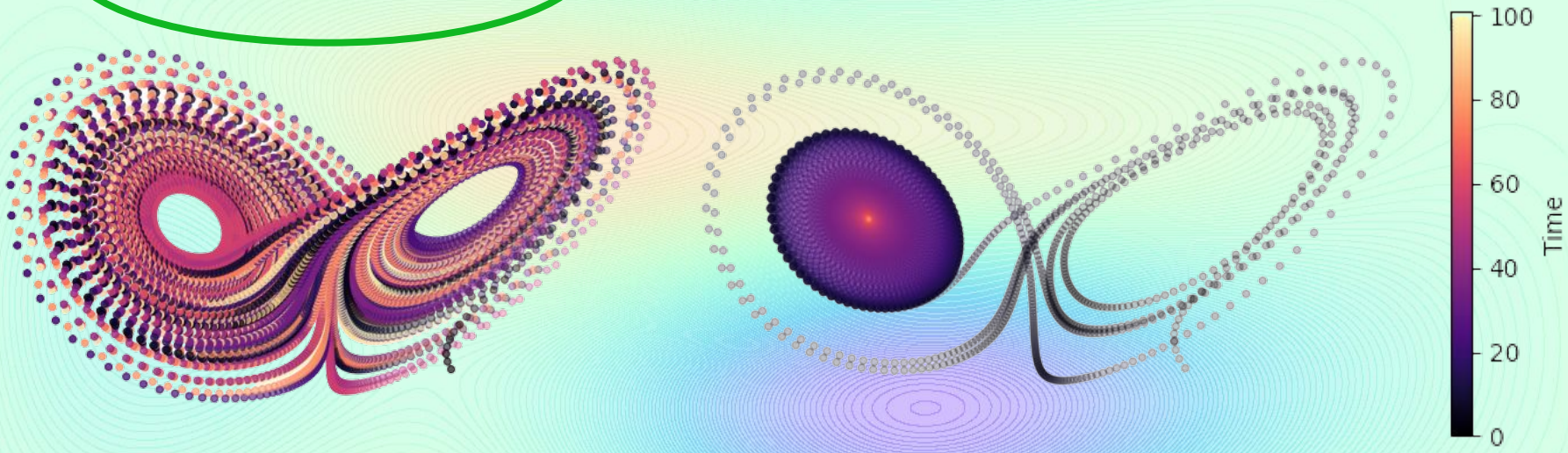
$\vec{x} \approx (-7.52, -7.52, 21.2)$

$\vec{x} \approx (7.52, 7.52, 21.2)$

Lorenz System (5/9)

Chaotic $\rho = 28$

Stable $\rho = 22.2$



Lorenz System (6/9)

The *semi-group* $\{S(t)\}_{t\geq 0}$ for the Lorenz system is **dissipative**

$$\left(\exists \mathcal{B} \subset \mathbb{R}^3,\ \mathcal{B} \text{ compact in } \mathbb{R}^3\ :\ X \subset \mathbb{R}^3,\ X \text{ bounded in } \mathbb{R}^3\right)$$

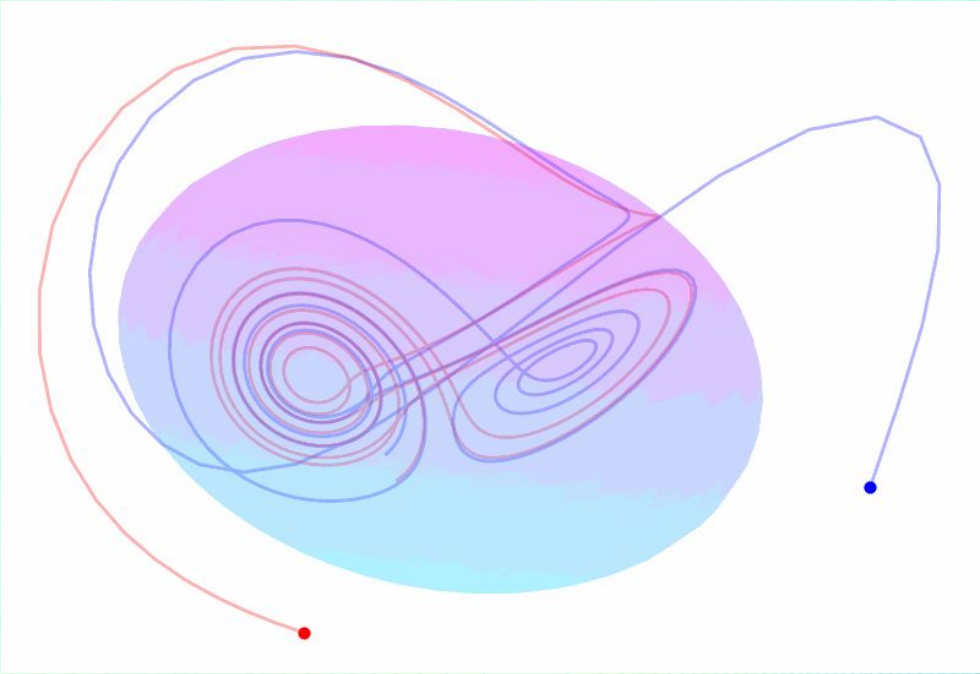
$$\Rightarrow$$

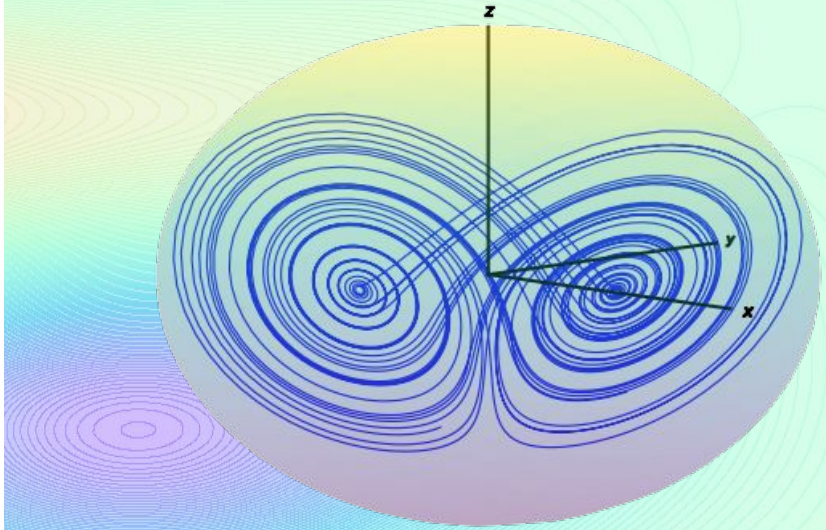
$$\left(\exists t = t_0(X)\ :\ S(t)X \subset \mathcal{B}\ \text{ for all }\ t \geq t_0(X)\right)$$

We call $\mathcal{B}$ an **absorbing set**

Lorenz System (7/9)

$$\text{Abosrbed by the ball } \mathcal{B} = B_{30}[(0,0,18)] \subset \mathbb{R}^3$$



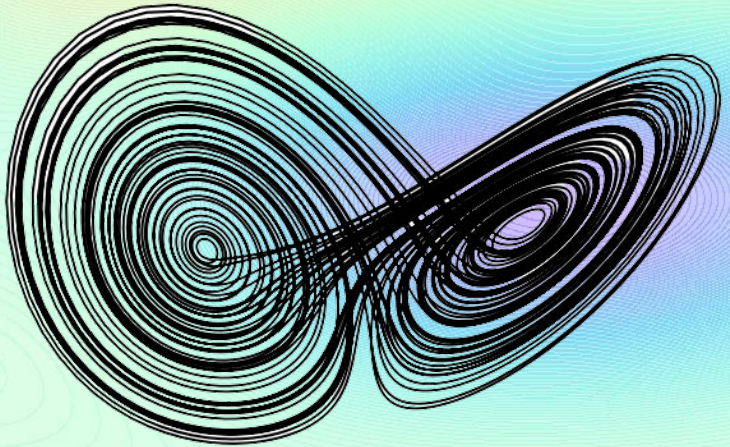


[Robinson] proves $\mathcal{B}$ is absorbing

Lorenz System (8/9)

The smallest absorbing set is called the **global attractor**

$$\mathcal{A}_S := \bigcap_{t \geq 0} S(t)\mathcal{B}$$



Lorenz System (9/9)

Proposition [Robinson]: Given a trajectory $\vec{x}(t) = S(t)\vec{x}_0$, and given $\epsilon > 0$, $T > 0$, there exists a time $t_* = t_*(\epsilon, T)$ and a point $\vec{y}_0 \in \mathcal{A}_S$ such that

$$||\vec{x}(t_* + t) - S(t)\vec{y}_0|| \leq \epsilon \qquad \forall t \in [0, T]$$

Data-Assimilation (***Nudging***) (1/4)

Let $\vec{u}(t) = \Big(u(t), v(t), w(t)\Big) \in \mathbb{R}^3 \quad \forall t \geq 0$

Solve $\dfrac{d\vec{u}}{dt} = L(\vec{u}) - \mu(u - x) \; ; \; \vec{u}(0) = \vec{u}_0$

$$\begin{pmatrix} \dfrac{du}{dt} \\ \dfrac{dv}{dt} \\ \dfrac{dw}{dt} \end{pmatrix} = \begin{pmatrix} \sigma(v-u) - \mu(u-x) \\ u(\rho - w) - v \\ uv - \beta w \end{pmatrix}$$

from true Lorenz

Data-Assimilation (***Nudging***) (2/4)

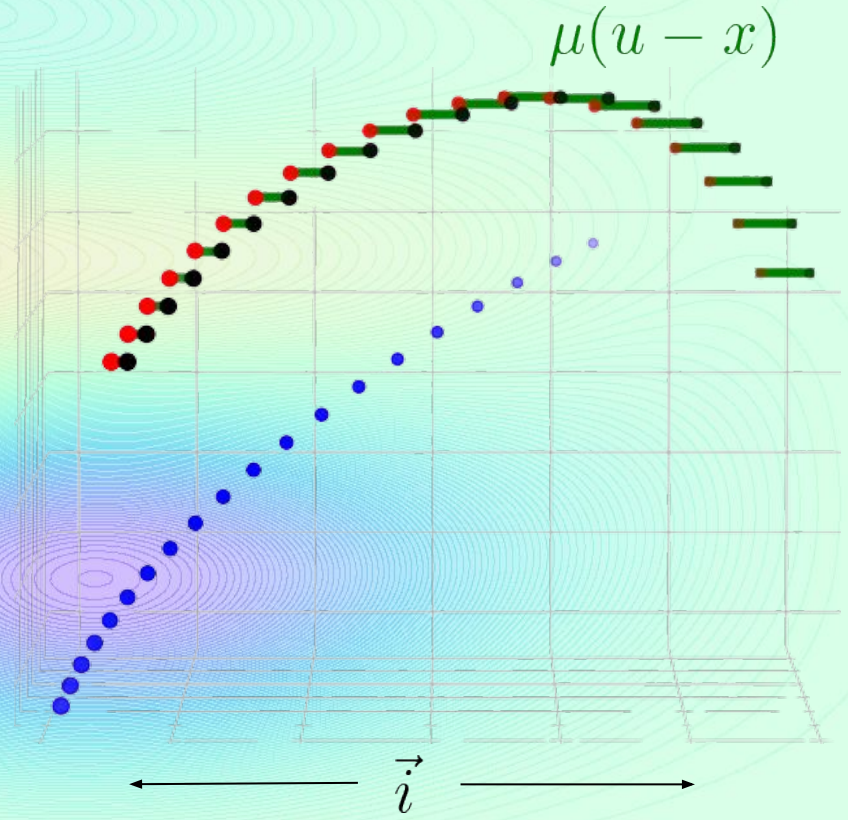
"True" or "Reference" Equation :

$$\frac{d\vec{x}}{dt} = L(\vec{x}) \; ; \; \vec{x}(0) = \textbf{unknown}$$

"Nudged" Equation:

$$\frac{d\vec{u}}{dt} = L(\vec{u}) - \mu(u - x)$$

$$\vec{u}(0) = \text{arbitrary fixed}$$



Data-Assimilation (***Nudging***) (3/4)

$$\frac{d\vec{u}}{dt} = L(\vec{u}) - \mu(u - x) \; ; \; \vec{u}(0) = \vec{u}_0$$

$$\vec{u}(t; \vec{u}_0) = \vec{u}_0 + \int_0^t \Big(L\vec{u}(s) - \mu(u - x) \Big) \vec{ds}$$

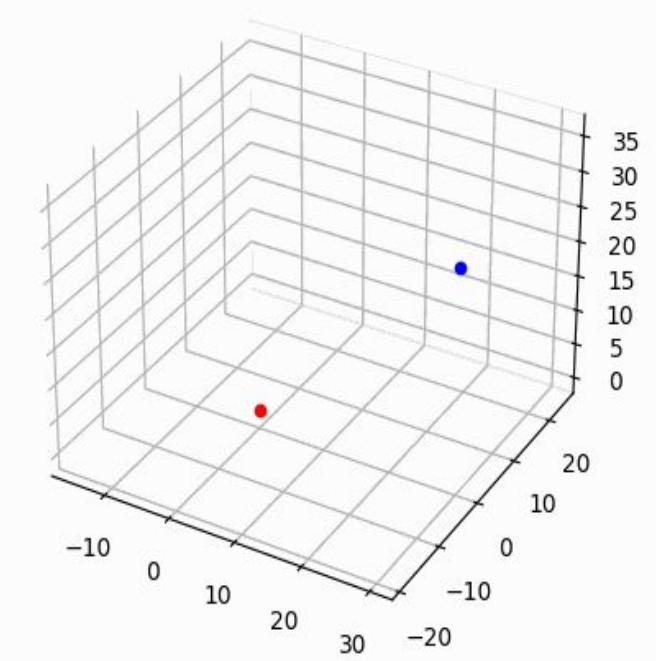
$$= \vec{u}_0 + \int_0^t L\vec{u}(s)\vec{ds} - \mu \int_0^t \Big(u(s) - x(s) \Big) ds$$

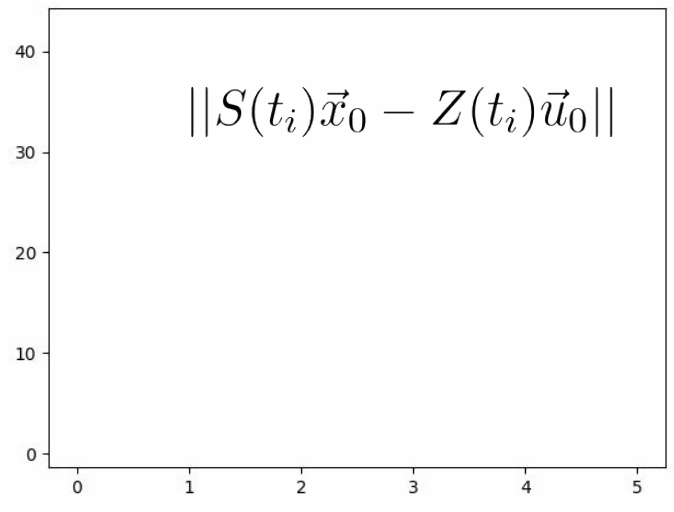
$$Z(t_n)\vec{u}_0 := S(t_n)\vec{u}_0 - \mu \sum_{i=0}^{n} \Big(u(t_i) - x(t_i) \Big) \delta_t$$

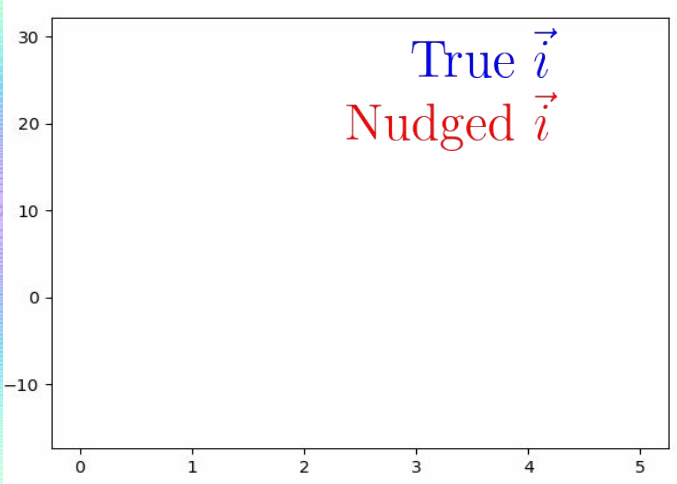
Nudging (4/4)

True Lorenz: $\vec{x}(t;\vec{x}_0) = S(t)\vec{x}_0$

Nudged Lorenz: $\vec{u}(t;\vec{u}_0) = Z(t)\vec{u}_0$







Deep Operator Networks (***DeepONet***) [Lu] (1/6)

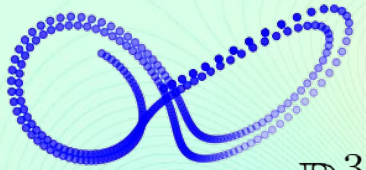
Goal: Approximate the solution to $\frac{d\vec{x}}{dt} = L(\vec{x})\ ;\ \vec{x}(0) = \vec{x}_0$

Method: Learn the **solution operator** $S(t)\vec{x}_0 = \vec{x}_0 + \int_0^t L\vec{x}(s)\vec{d}s$

$$S(t_n)\vec{x}_0 = \vec{x}_0 + \sum_{i=0}^{n} L\vec{x}(t_i)\delta_t$$

Deep Operator Networks (***DeepONet***) (2/6)

Branch input: collection of "**states**" $\vec{x}(t_i)$



$\mathbb{R}^3$

$\rightarrow$ $\vec{x}(t_1)$ $\vec{x}(t_2)$ $\vdots$ $\vec{x}(t_n)$ $\rightarrow$ Branch Net (FNN)

$\mathbb{R}$

$\rightarrow$ τ_1 τ_2 $\vdots$ τ_n $\rightarrow$ Trunk Net (FNN)

$= G(\tau)\vec{x}_i$

$\approx S(\tau)\vec{x}_i$

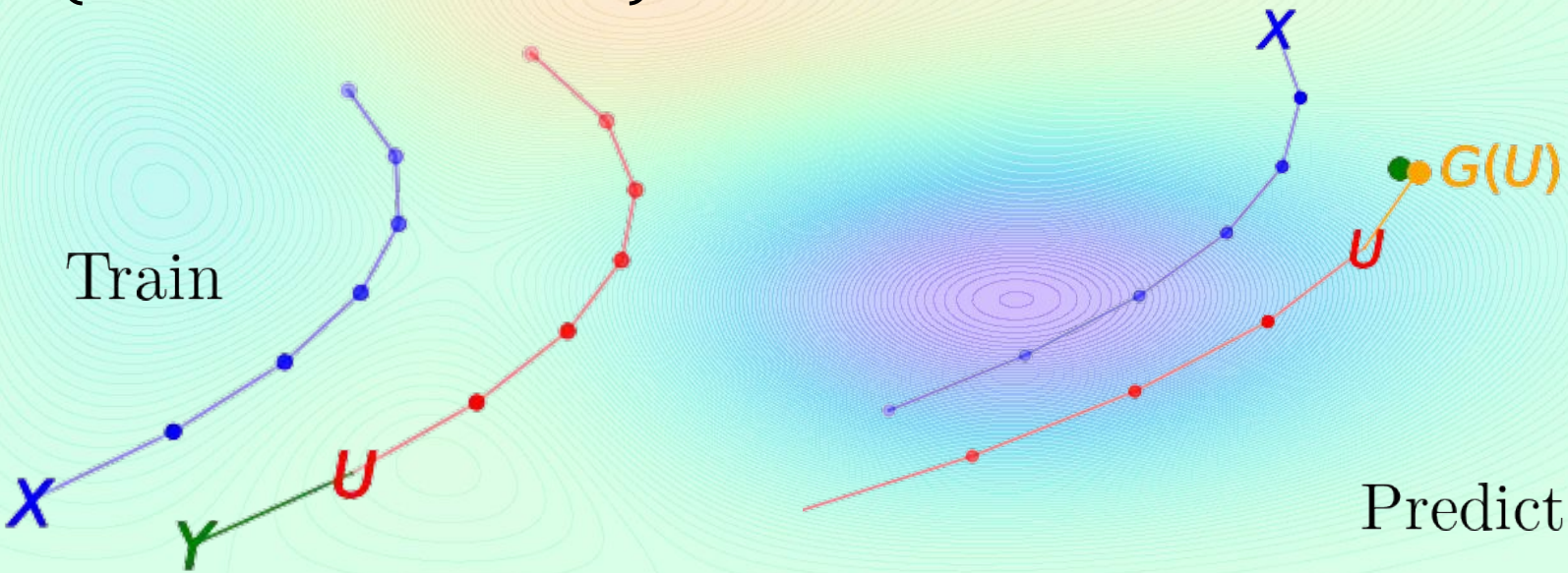
Trunk input: "**locations**" τ_i at which to evaluate $L\vec{x}(\tau)$

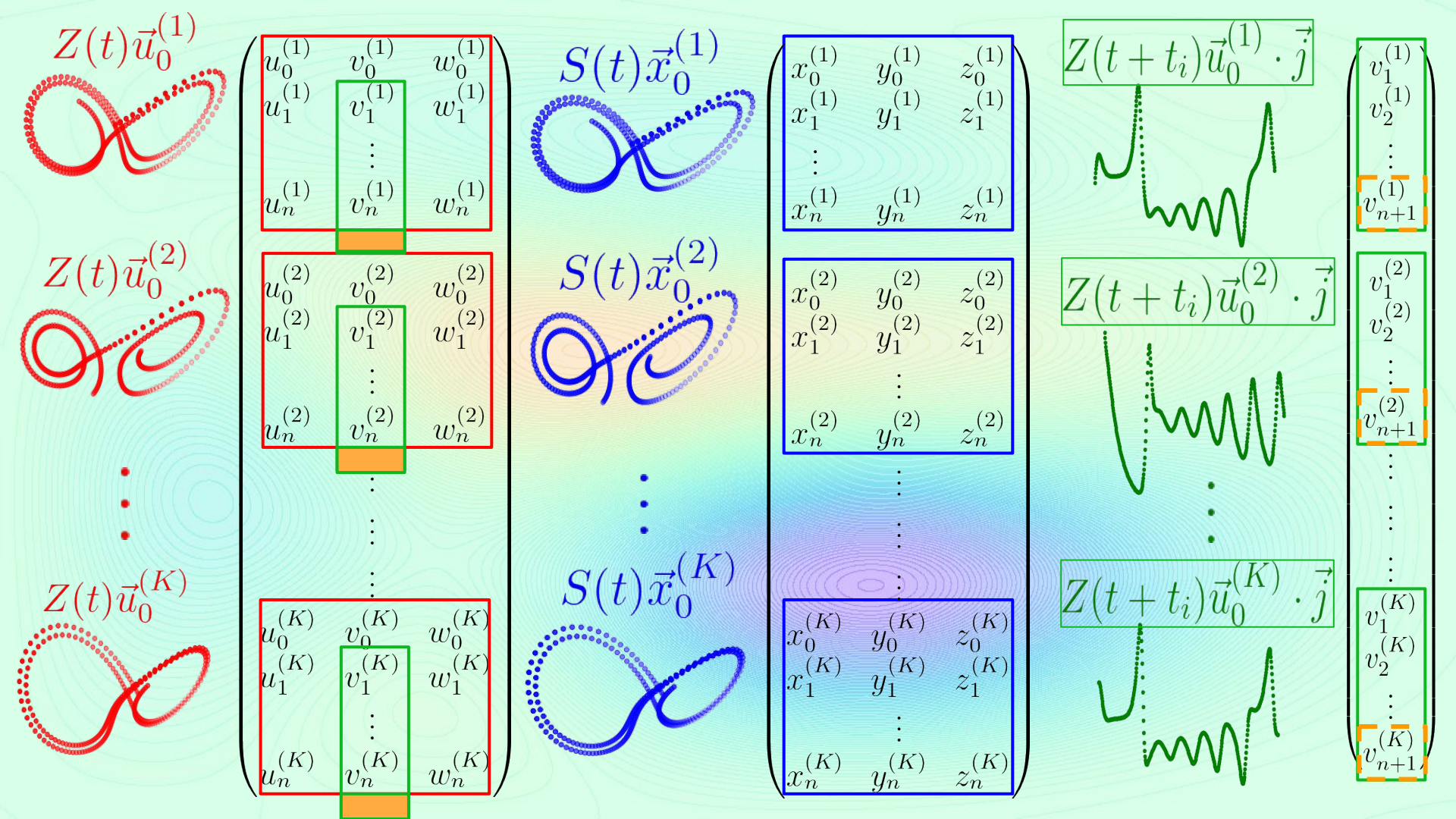
Deep Operator Networks (***DeepONet***) (3/6)

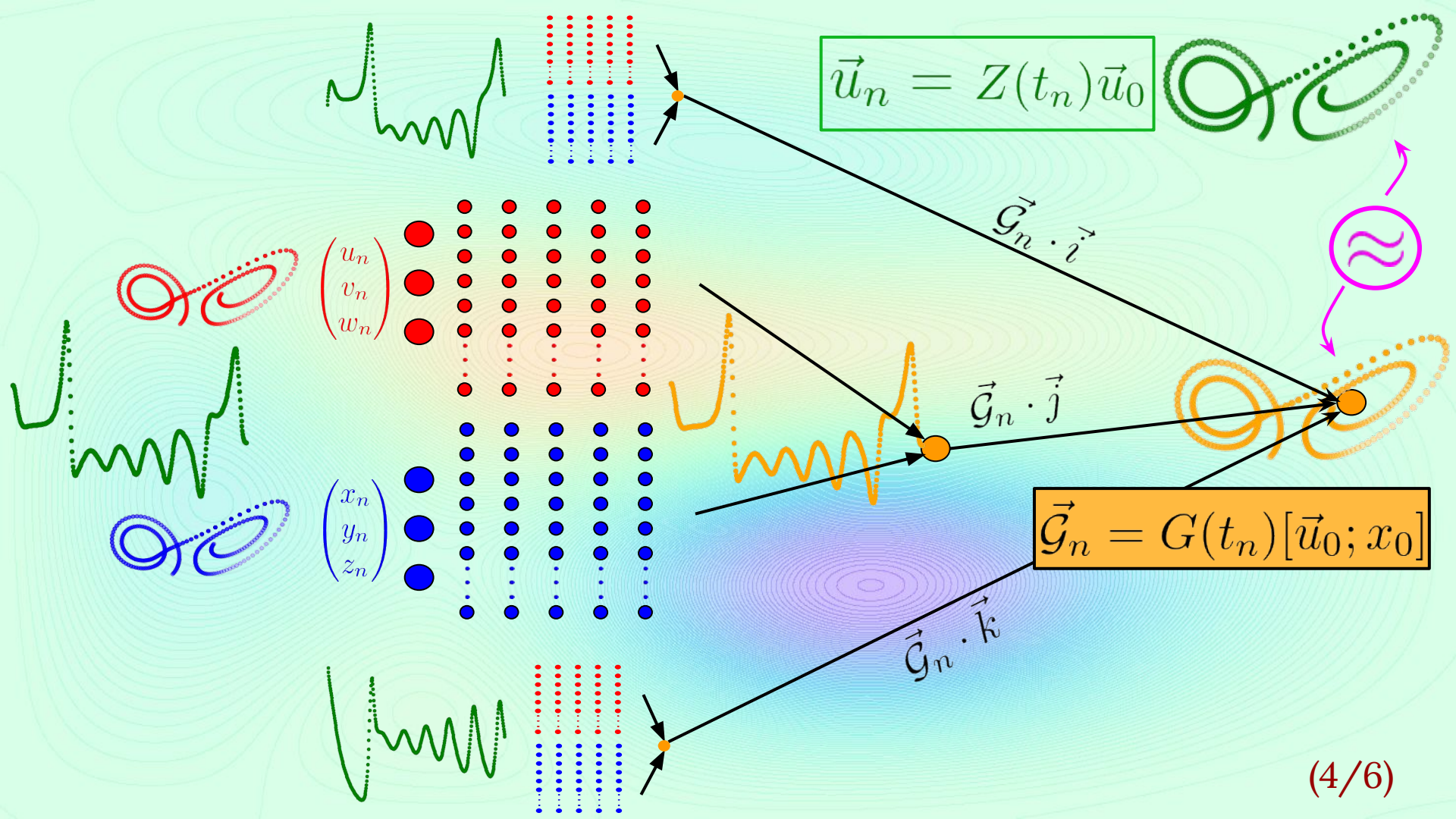
Goal: from state $\vec{u}_n$ (and state $\vec{x}_n$), we want to predict $\vec{u}_{n+1}$

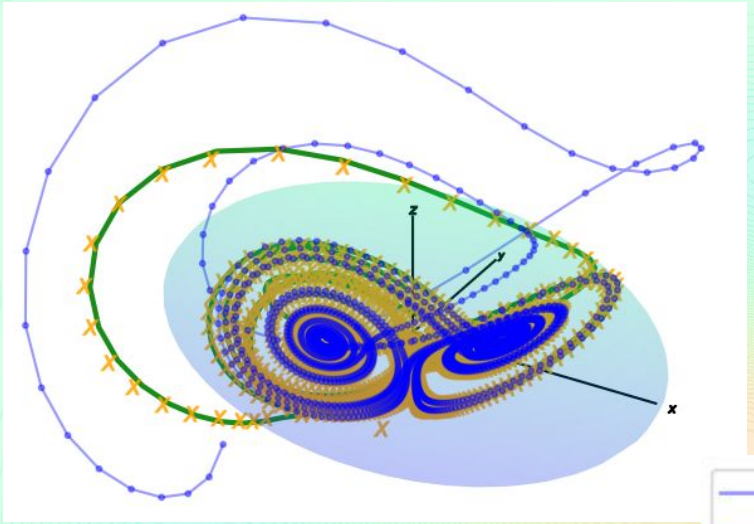
$$\left\{ \begin{array}{l} \text{Data: } \mathcal{X}_n = (\vec{u}_n, \vec{x}_n) \\ \text{Label: } \mathcal{Y}_n = \vec{u}_{n+1} \end{array} \right\} \longrightarrow \left\{ \text{Prediction: } \vec{\mathcal{G}}_n \approx \vec{u}_{n+1} \right\}$$

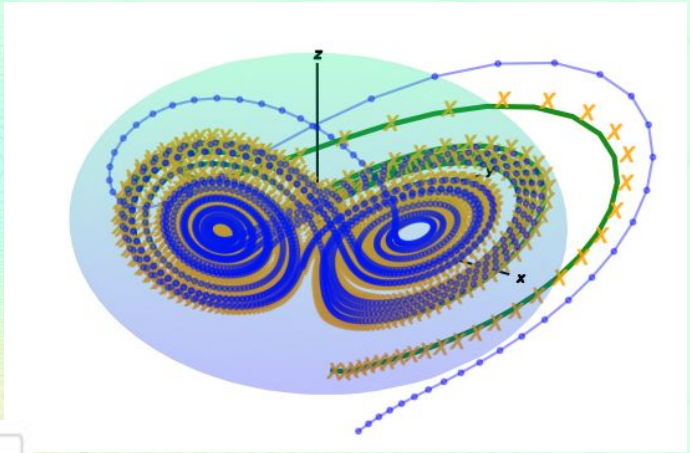
[Antil]



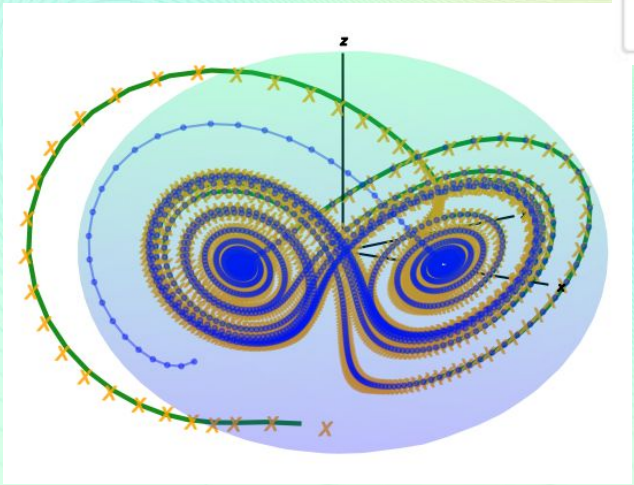

$Z(t)\vec{u}_0^{(1)}$
$Z(t)\vec{u}_0^{(2)}$
$Z(t)\vec{u}_0^{(K)}$
$u_0^{(1)}$ $v_0^{(1)}$ $w_0^{(1)}$
$u_1^{(1)}$ $v_1^{(1)}$ $w_1^{(1)}$
$u_n^{(1)}$ $v_n^{(1)}$ $w_n^{(1)}$
$u_0^{(2)}$ $v_0^{(2)}$ $w_0^{(2)}$
$u_1^{(2)}$ $v_1^{(2)}$ $w_1^{(2)}$
$u_n^{(2)}$ $v_n^{(2)}$ $w_n^{(2)}$
$u_0^{(K)}$ $v_0^{(K)}$ $w_0^{(K)}$
$u_1^{(K)}$ $v_1^{(K)}$ $w_1^{(K)}$
$u_n^{(K)}$ $v_n^{(K)}$ $w_n^{(K)}$
$S(t)\vec{x}_0^{(1)}$
$S(t)\vec{x}_0^{(2)}$
$S(t)\vec{x}_0^{(K)}$
$x_0^{(1)}$ $y_0^{(1)}$ $z_0^{(1)}$
$x_1^{(1)}$ $y_1^{(1)}$ $z_1^{(1)}$
$x_n^{(1)}$ $y_n^{(1)}$ $z_n^{(1)}$
$x_0^{(2)}$ $y_0^{(2)}$ $z_0^{(2)}$
$x_1^{(2)}$ $y_1^{(2)}$ $z_1^{(2)}$
$x_n^{(2)}$ $y_n^{(2)}$ $z_n^{(2)}$
$x_0^{(K)}$ $y_0^{(K)}$ $z_0^{(K)}$
$x_1^{(K)}$ $y_1^{(K)}$ $z_1^{(K)}$
$x_n^{(K)}$ $y_n^{(K)}$ $z_n^{(K)}$
$Z(t+t_i)\vec{u}_0^{(1)} \cdot \vec{j}$
$Z(t+t_i)\vec{u}_0^{(2)} \cdot \vec{j}$
$Z(t+t_i)\vec{u}_0^{(K)} \cdot \vec{j}$
$v_1^{(1)}$ $v_2^{(1)}$ $v_{n+1}^{(1)}$
$v_1^{(2)}$ $v_2^{(2)}$ $v_{n+1}^{(2)}$
$v_1^{(K)}$ $v_2^{(K)}$ $v_{n+1}^{(K)}$

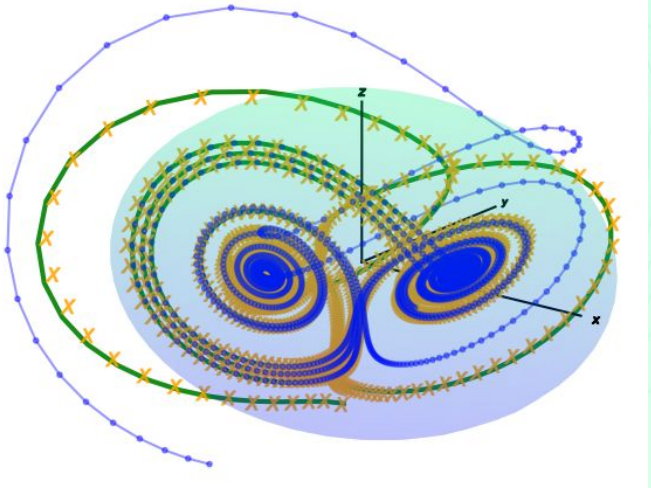

$\vec{u}_n = Z(t_n)\vec{u}_0$
$\vec{\mathcal{G}}_n = G(t_n)[\vec{u}_0; x_0]$
$\begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}$
$\begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}$
$\vec{\mathcal{G}}_n \cdot \vec{i}$
$\vec{\mathcal{G}}_n \cdot \vec{j}$
$\vec{\mathcal{G}}_n \cdot \vec{k}$
$\approx$

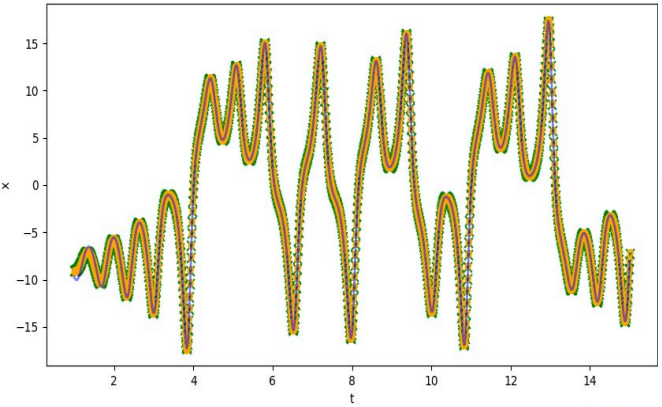

z
y
x

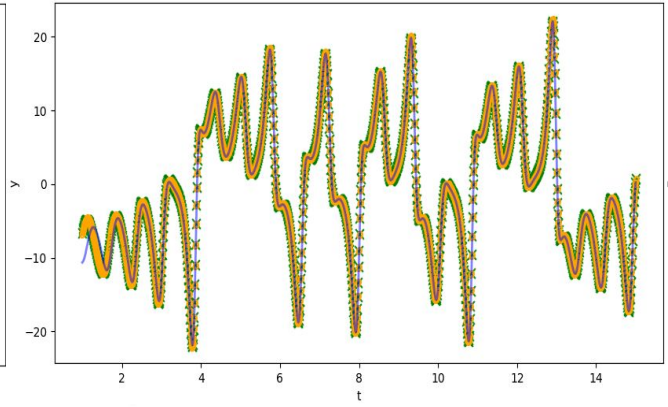

z
y
x

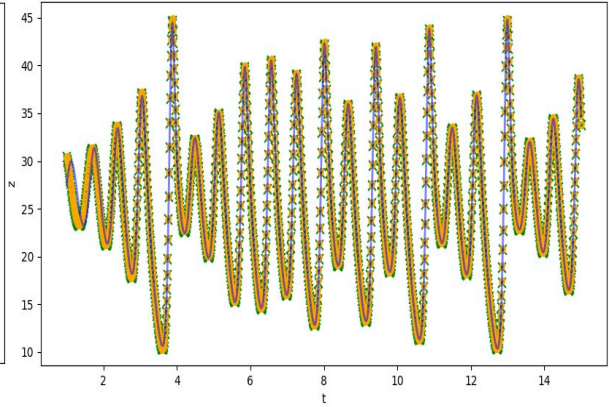

True Lorenz
DA Lorenz
DeepONet DA Lorenz

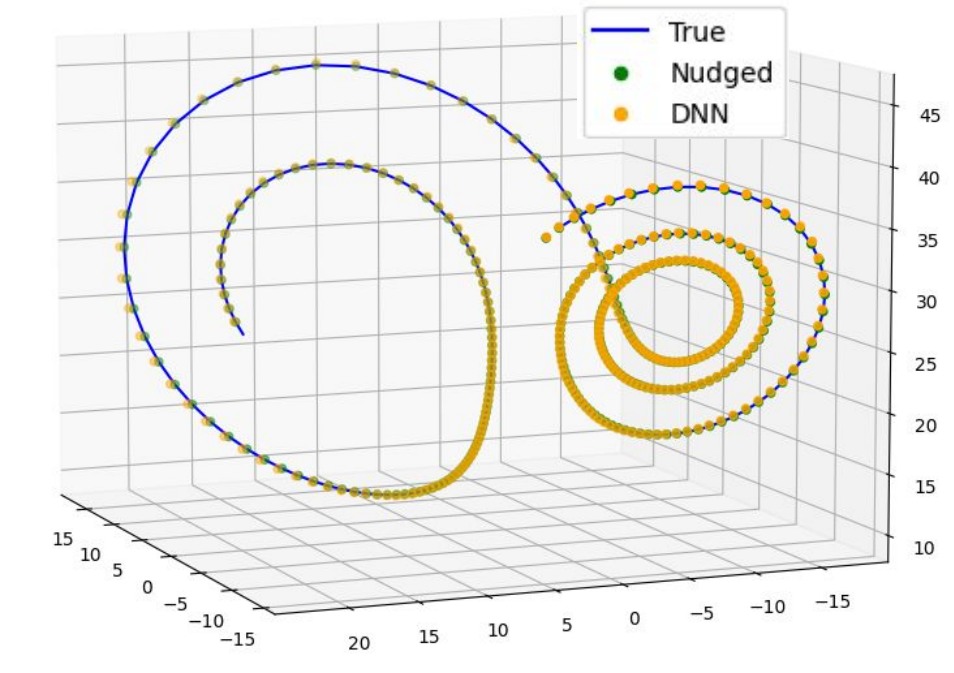

z
y
x

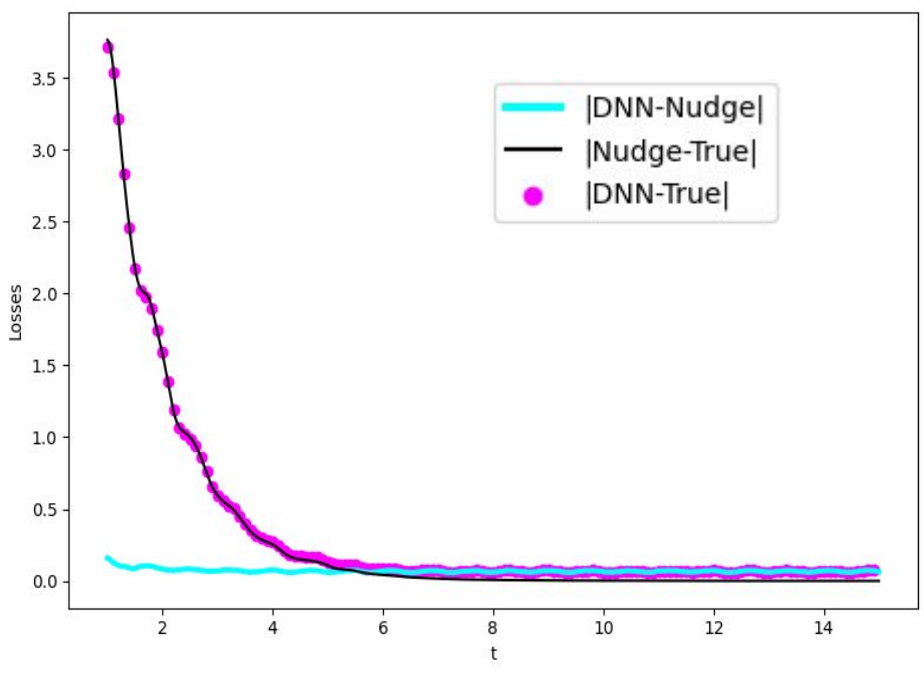

z
y
x


x
t


y
t

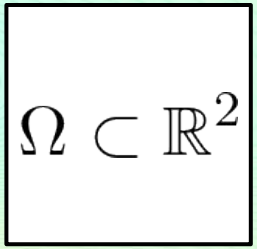

z
t


True
Nudged
DNN


|DNN-Nudge|
|Nudge-True|
|DNN-True|
Losses
t

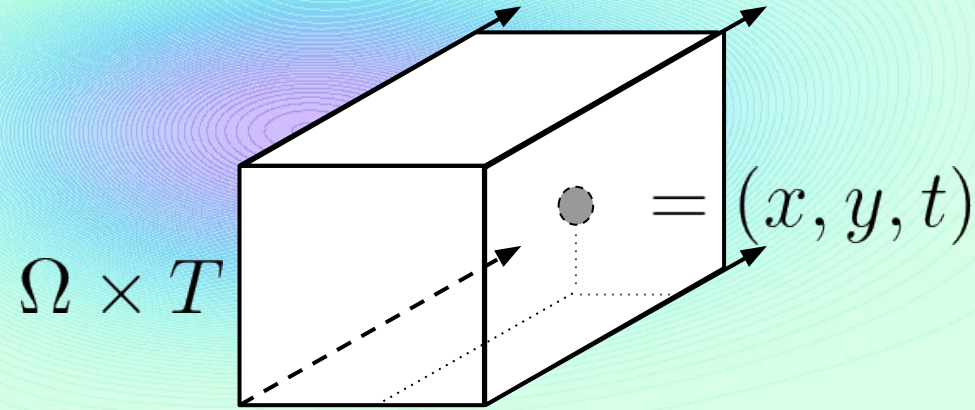
Learning the Heat Equation (1/11)

Let $\Omega = [0,1] \times [0,1] \subset \mathbb{R}^2$



Let $T = \mathbb{R}^+ \cup \{0\}$ $\quad (i.e.\ T \geq 0)$

Then $\Omega \times T \subset \mathbb{R}^3$

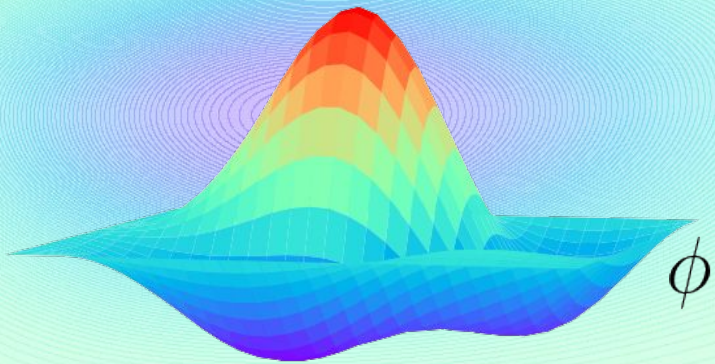


Learning the Heat Equation (2/11)

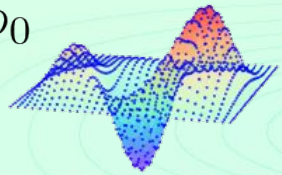
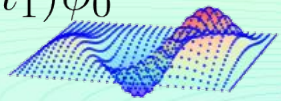
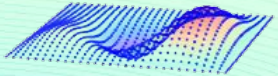
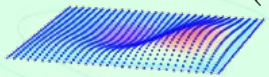
$$u = u(x, y, t) \text{ for } (x, y, t) \in \Omega \times T$$

$$\begin{cases} \boxed{\dfrac{\partial u}{\partial t} = \alpha\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) \text{ in interior of } \Omega} \\ u(x, y, t) = 0 \text{ on } \partial\Omega \\ u(x, y, 0) = \phi(x, y) \end{cases}$$

Heat Equation



$\phi(x, y)$

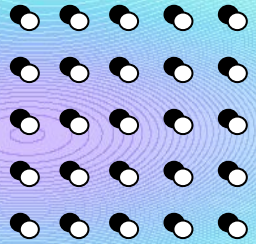





$\phi_0 \qquad S(t_1)\phi_0 \qquad S(t_2)\phi_0 \qquad \cdots \qquad S(t_{N_t})\phi_0$

$$\mathcal{U} = \left\{ \begin{pmatrix} u^0_{0,0} & u^0_{1,0} & \cdots & u^0_{M,0} \\ u^0_{0,1} & u^0_{1,1} & \cdots & u^0_{M,1} \\ \vdots & \vdots & \ddots & \vdots \\ u^0_{0,N} & u^0_{1,N} & \cdots & u^0_{M,N} \end{pmatrix}, \begin{pmatrix} u^1_{0,0} & u^1_{1,0} & \cdots & u^1_{M,0} \\ u^1_{0,1} & u^1_{1,1} & \cdots & u^1_{M,1} \\ \vdots & \vdots & \ddots & \vdots \\ u^1_{0,N} & u^1_{1,N} & \cdots & u^1_{M,N} \end{pmatrix}, \cdots, \begin{pmatrix} u^{N_t}_{0,0} & u^{N_t}_{1,0} & \cdots & u^{N_t}_{M,0} \\ u^{N_t}_{0,1} & u^{N_t}_{1,1} & \cdots & u^{N_t}_{M,1} \\ \vdots & \vdots & \ddots & \vdots \\ u^{N_t}_{0,N} & u^{N_t}_{1,N} & \cdots & u^{N_t}_{M,N} \end{pmatrix} \right\}$$

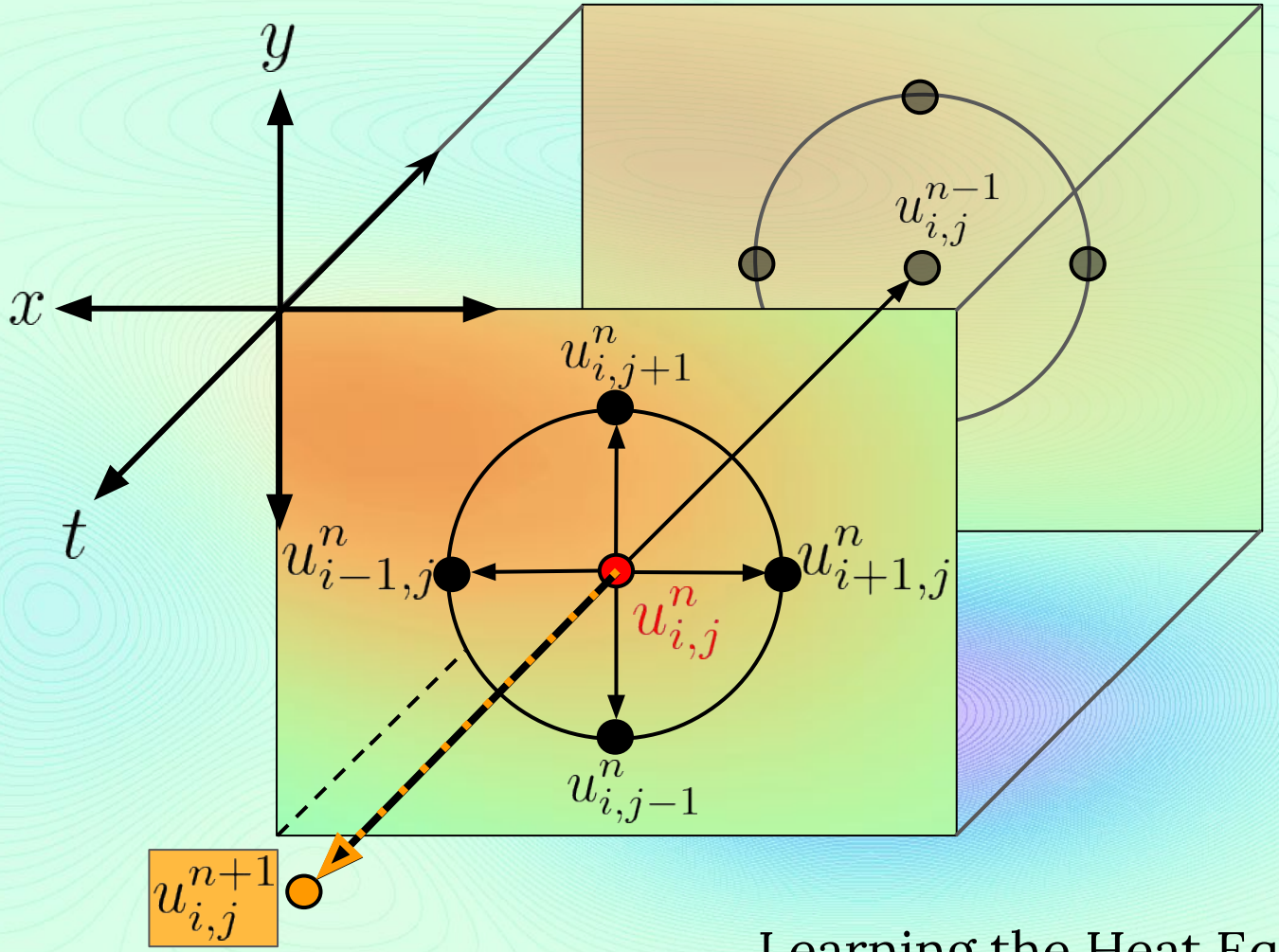
$$= \left\{ u^0 \ , \ u^1 \ , \ \ldots \ , \ u^{N_t} \right\} = \left\{ S(t_i)\phi_0 \right\}_{t_0 \le t_i \le t_{N_t}}$$

$$\mathcal{D} := \begin{pmatrix} (x_0, y_0) & (x_1, y_0) & \cdots & (x_M, y_0)) \\ (x_0, y_1) & (x_1, y_1) & \cdots & (x_M, y_1) \\ \vdots & \vdots & \ddots & \vdots \\ (x_0, y_N) & (x_0, y_N) & \cdots & (x_M, y_N) \end{pmatrix} =$$



$$\subset \boxed{\Omega} \subset \mathbb{R}^2$$

$$\underline{\mathcal{D}} := \left\{ \mathcal{D} \ , \ \mathcal{D} \ , \ \ldots \ , \ \mathcal{D} \right\}$$


y
x
t
$u^{n-1}_{i,j}$
$u^{n}_{i,j+1}$
$u^{n}_{i-1,j}$
$u^{n}_{i+1,j}$
$u^{n}_{i,j}$
$u^{n}_{i,j-1}$
$u^{n+1}_{i,j}$

Learning the Heat Equation (5/11)

Convert the **partial differential equation**

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

to a **partial difference equation**

$$\frac{u_{i,j}^{n+1} - u_{i,n}^{n}}{\delta_t} = \alpha\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\delta_x)^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\delta_y)^2}\right)$$

$$u_{i,j}^{n+1} = \delta_t \alpha\left(\cdots\right) + u_{i,j}^{n}$$

$$\delta_t \leq \frac{\delta_x \delta_y}{2} \Rightarrow \text{ stable [Pierce]}$$

Learning the Heat Equation (6/11)

Nudged heat **differential** equation

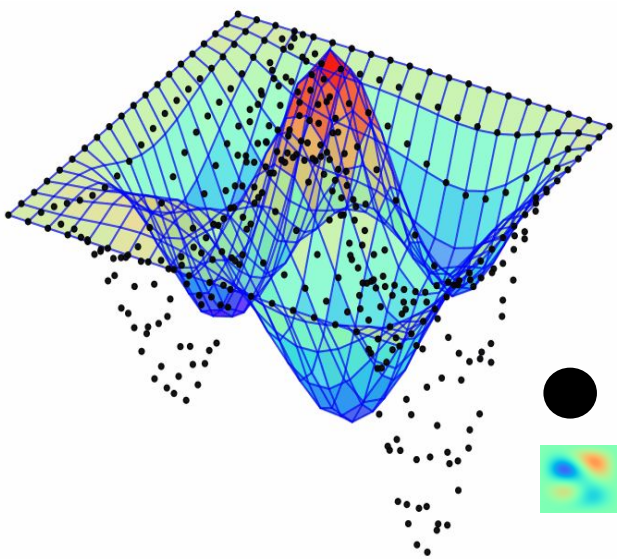
$$\frac{\partial w}{\partial t} = \alpha\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) - \mu(w - u)$$

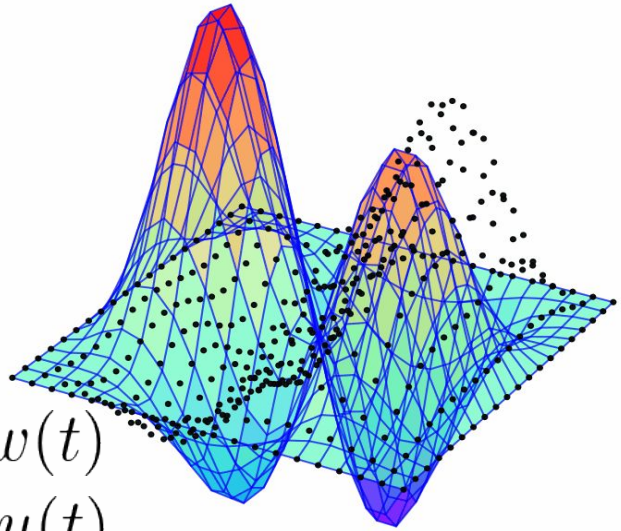
Nudged heat **difference** equation

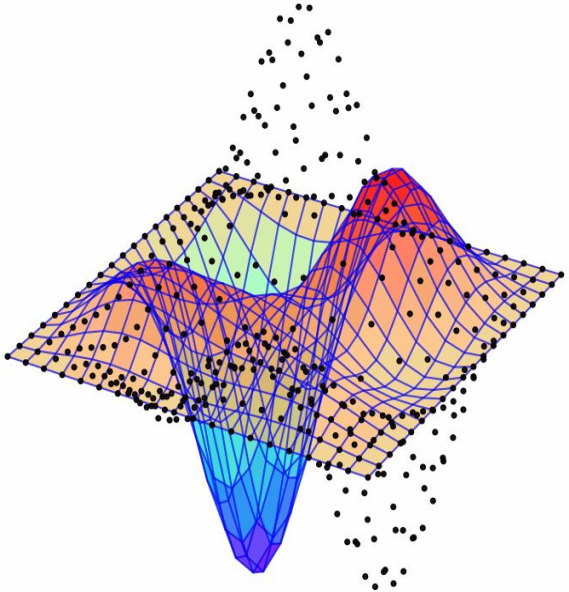
$$\frac{w_{i,j}^{n+1} - w_{i,n}^{n}}{\delta_t} = \alpha\left(\frac{w_{i+1,j}^{n} - 2w_{i,j}^{n} + w_{i-1,j}^{n}}{(\delta_x)^2} + \frac{w_{i,j+1}^{n} - 2w_{i,j}^{n} + w_{i,j-1}^{n}}{(\delta_y)^2}\right) - \mu(w_{i,j}^{n} - u_{i,j}^{n})$$

$$\Rightarrow$$

$$w_{i,j}^{n+1} = \delta_t\left[\alpha\left(\frac{w_{i+1,j}^{n} - 2w_{i,j}^{n} + w_{i-1,j}^{n}}{(\delta_x)^2} + \frac{w_{i,j+1}^{n} - 2w_{i,j}^{n} + w_{i,j-1}^{n}}{(\delta_y)^2}\right) - \mu(w_{i,j}^{n} - u_{i,j}^{n})\right] + w_{i,j}^{n}$$

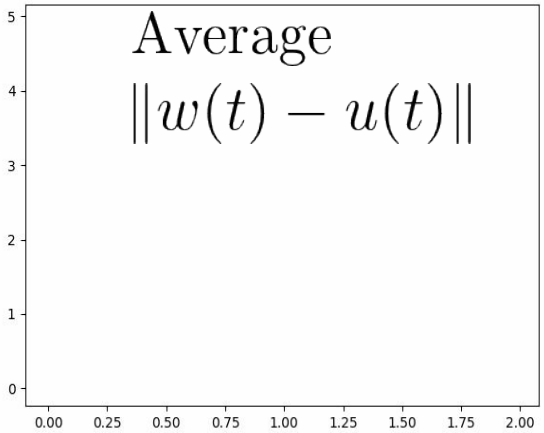


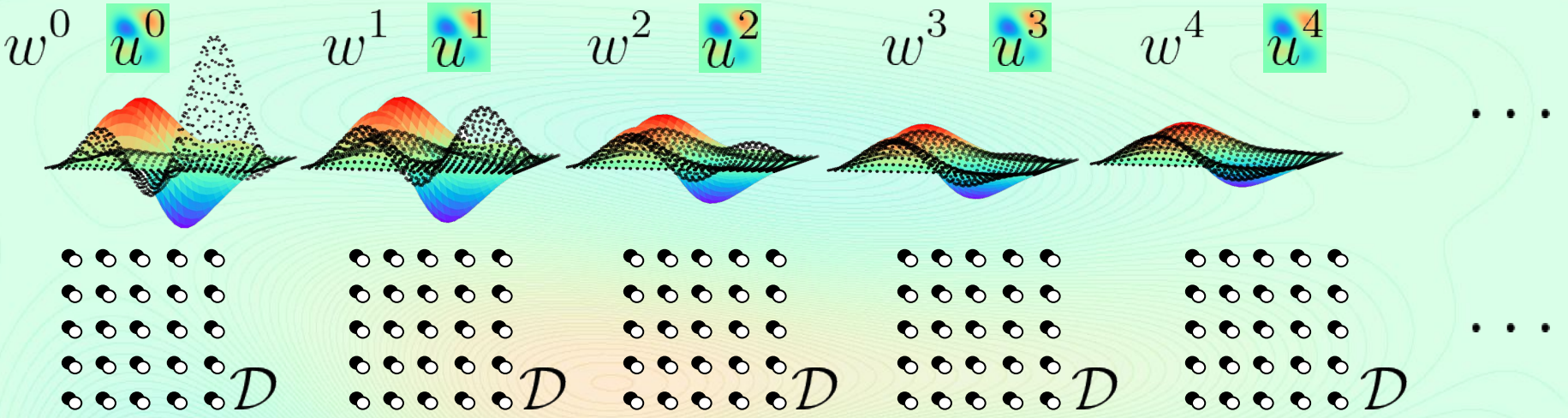




$w(t)$

$u(t)$



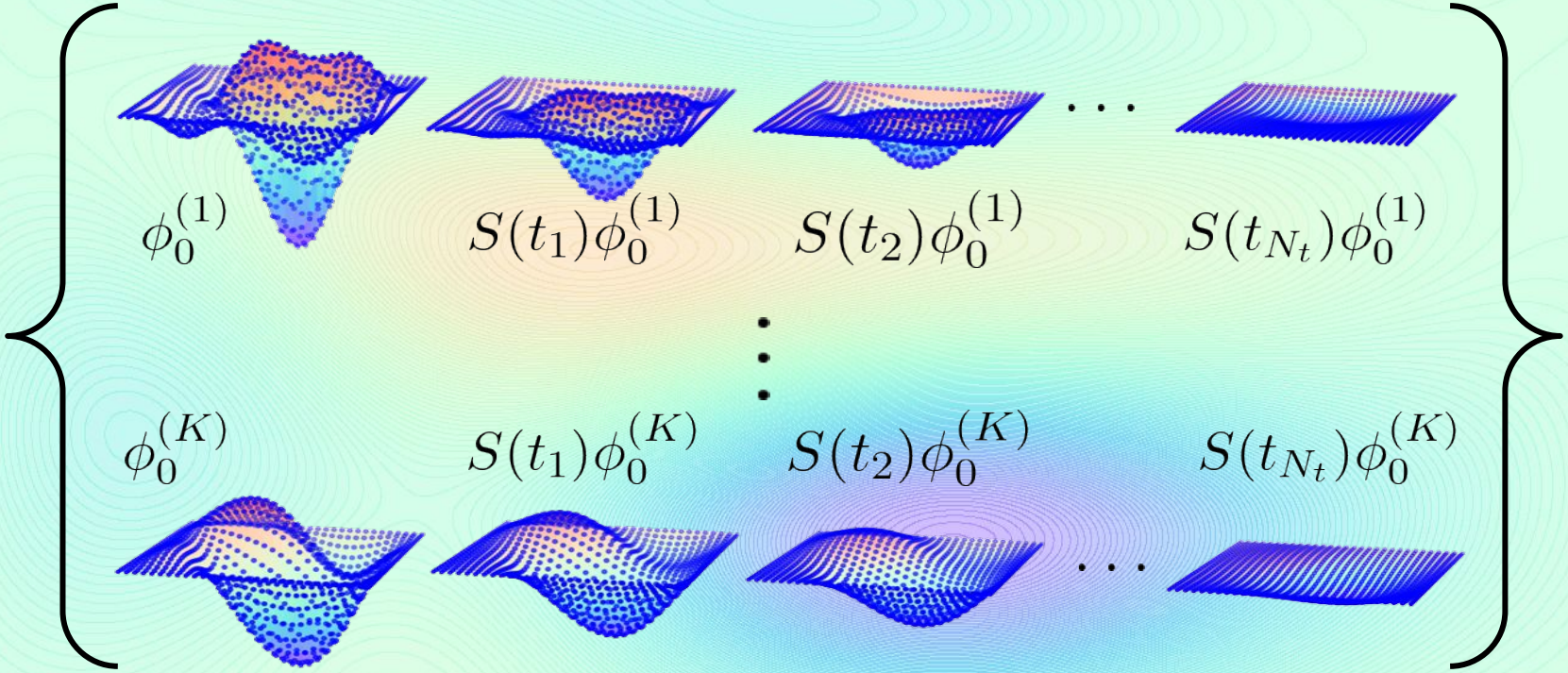


$$\mathcal{W} = \left\{ w^0 \, , \; w^1 \, , \; \ldots \, , \; w^{N_t} \right\} = \left\{ Z(t_i)\psi_0 \right\}_{t_0 \le t_i \le t_{N_t}}$$

$$\mathcal{U} = \left\{ u^0 \, , \; u^1 \, , \; \ldots \, , \; u^{N_t} \right\} = \left\{ S(t_i)\phi_0 \right\}_{t_0 \le t_i \le t_{N_t}}$$

$$\underline{\mathcal{D}} = \left\{ \mathcal{D} \, , \; \mathcal{D} \, , \; \ldots \, , \; \mathcal{D} \right\}$$

$$\underline{\mathcal{U}} = \left\{ \mathcal{U}^{(k)} \right\}_{k=1,\dots,K} = \left\{ \left\{ S(t_i)\phi_0^{(k)} \right\}_{t_0 \le t_i \le t_{N_t}} \right\}_{k=1,\dots,K}$$



$$\underline{\mathcal{Y}}^{(k)}_{i,j}(t_n) := \underline{\mathcal{W}}^{(k)}_{i,j}(t_{n+1})$$

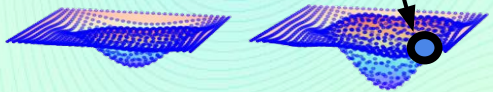
$$\underline{\mathcal{G}}^{(k)}_{i,j}(t_n) \approx \underline{\mathcal{W}}^{(k)}_{i,j}(t_{n+1})$$

$$\underline{\mathcal{W}}^{(k)}_{i,j}(t_n) = Z(t_n)\psi_0(x_i, y_j)$$

$$\cdots$$

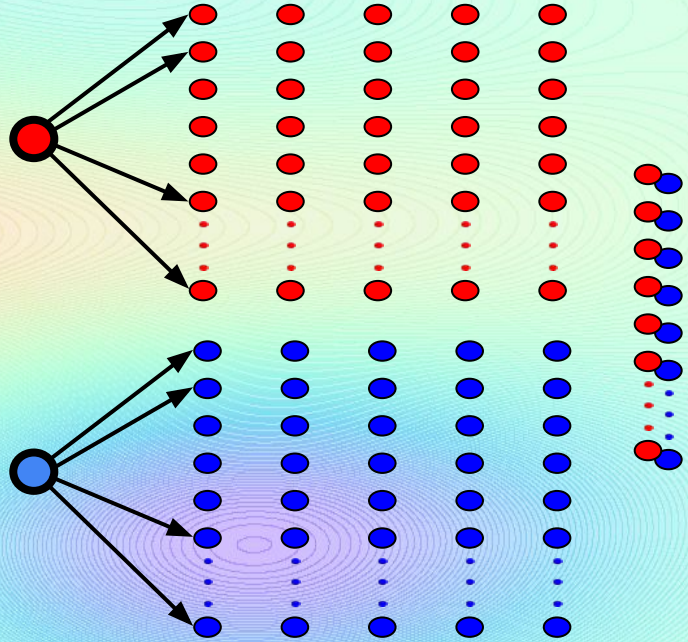
t_{n+1} t_n

$$\underline{\mathcal{U}}^{(k)}_{i,j}(t_n) = S(t_n)\phi_0(x_i, y_j)$$



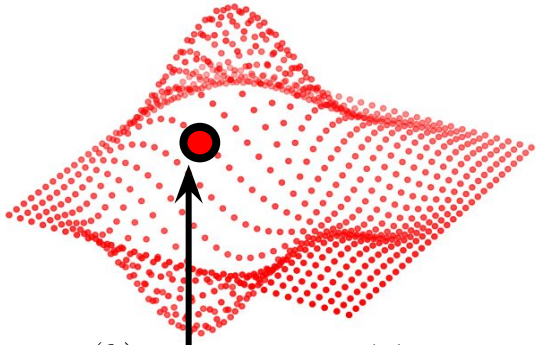
$$\cdots$$

t_{n+1} t_n

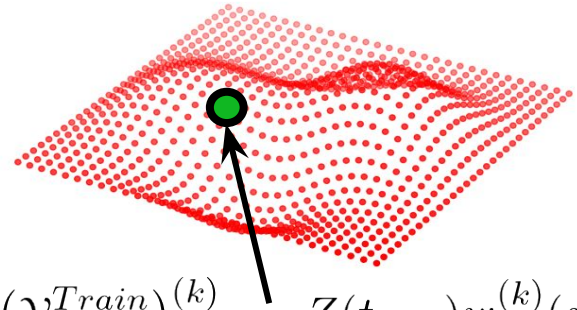


$$= \underline{\mathcal{G}}^{(k)}_{i,j}(t_n)$$

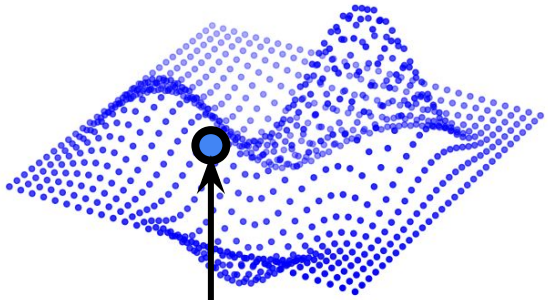
Learning the Heat Equation (10/11)



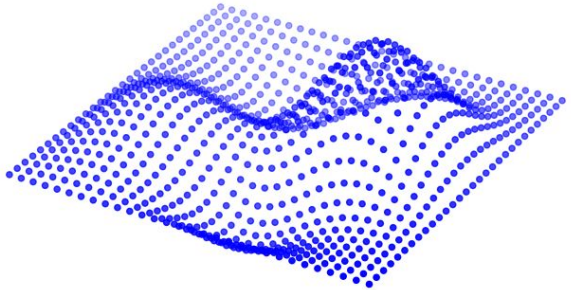
$$\left(\mathcal{X}_0^{Train}\right)_{i,j,n}^{(k)} = Z(t_n) w_0^{(k)}(x_i, y_j)$$

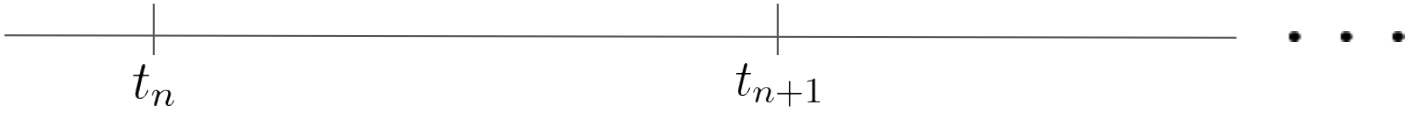


$$\left(\mathcal{Y}^{Train}\right)_{i,j,n}^{(k)} = Z(t_{n+1}) w_0^{(k)}(x_i, y_j)$$

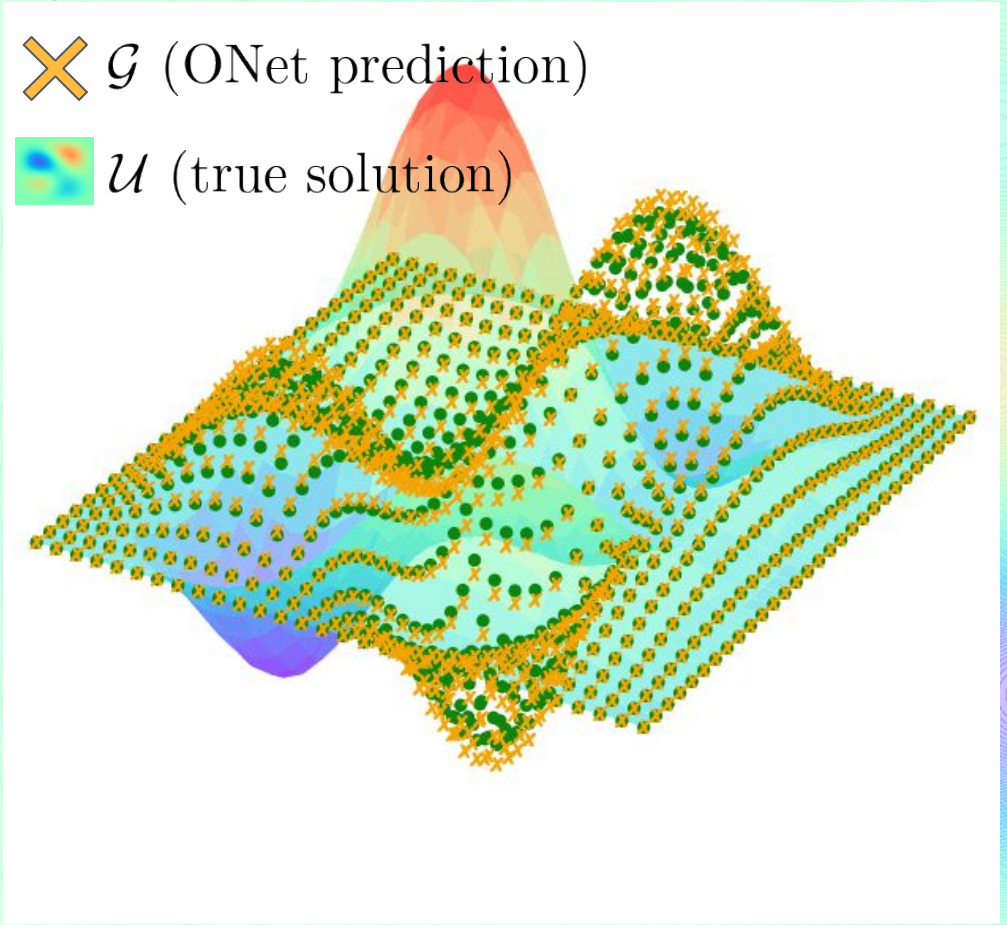


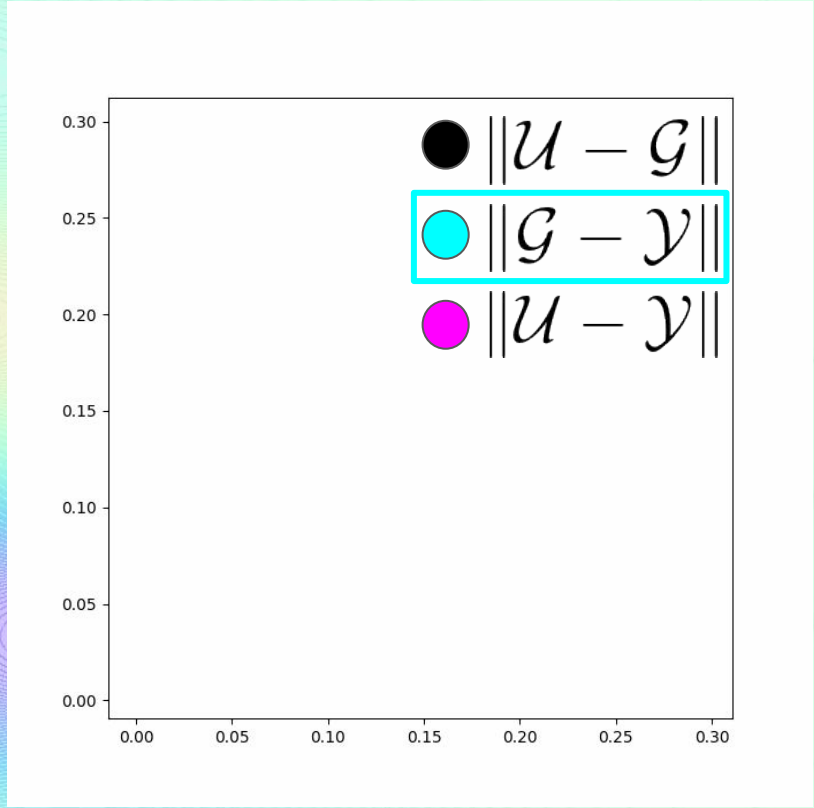
$$\left(\mathcal{X}_1^{Train}\right)_{i,j,n}^{(k)} = S(t_n) u_0^{(k)}(x_i, y_j)$$





Learning the Heat Equation (11/11)

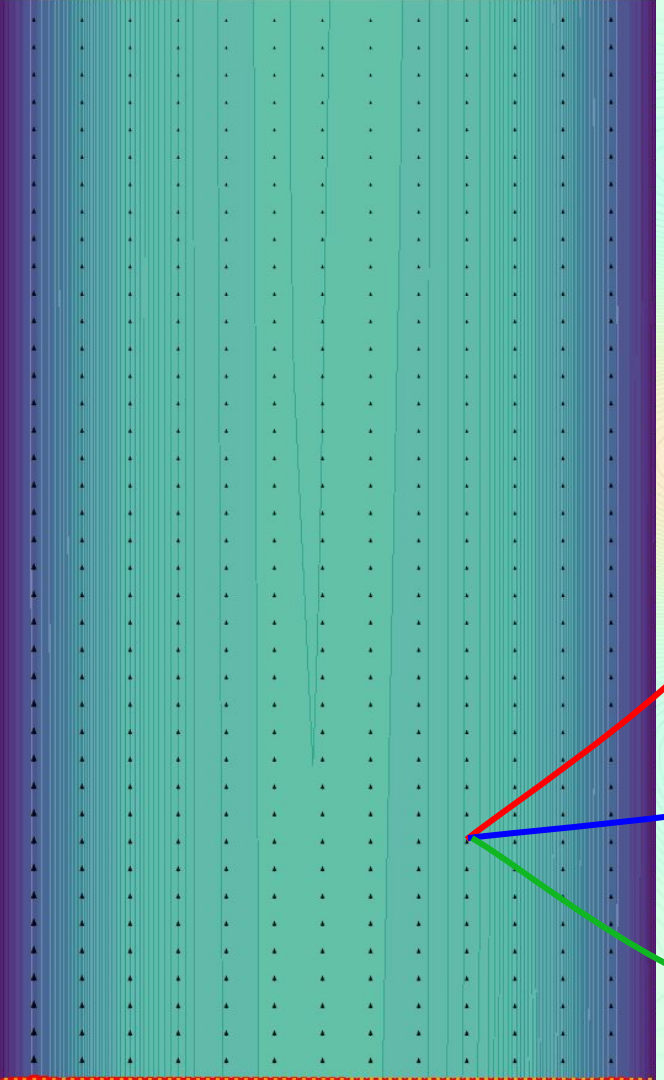


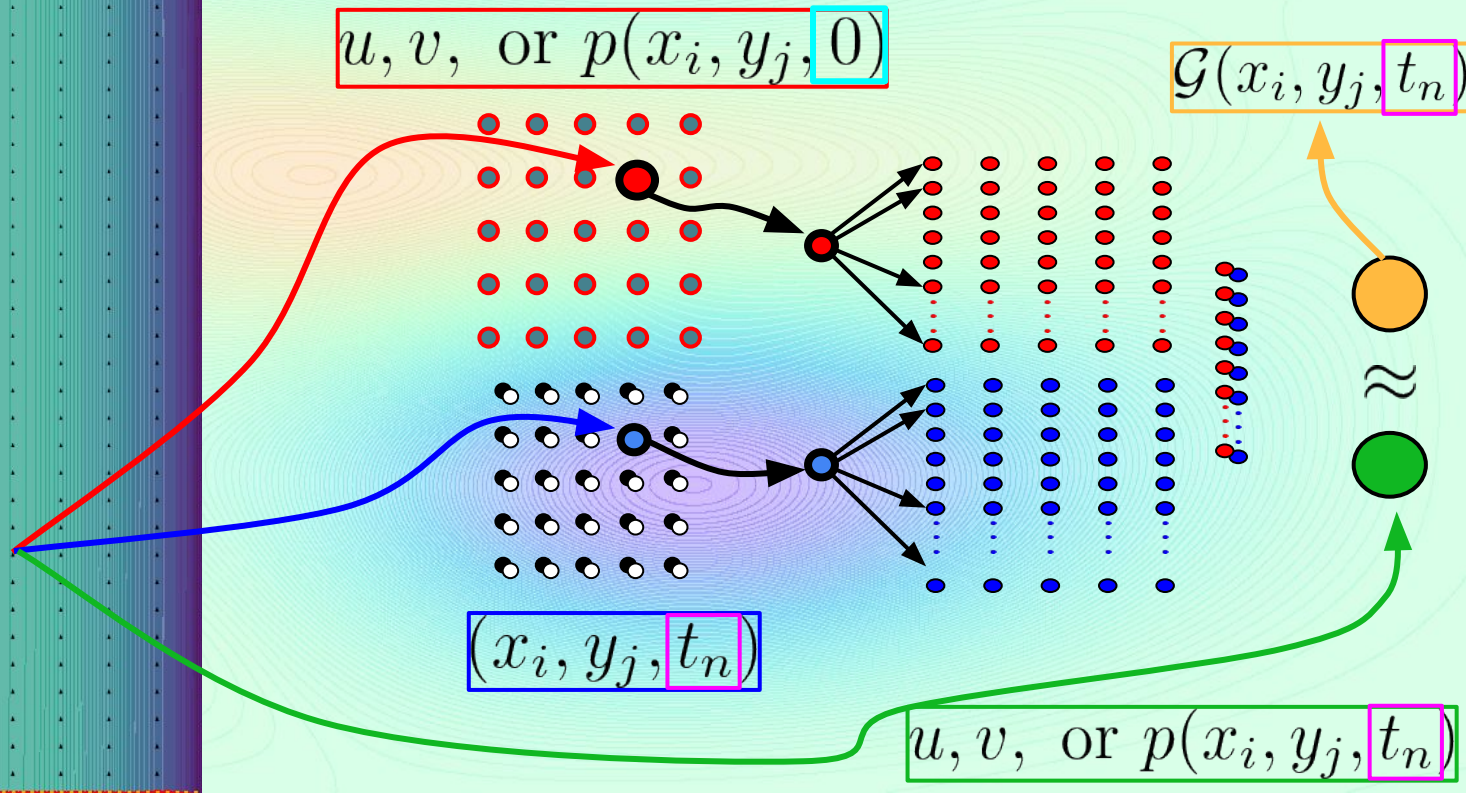


Recall $\mathcal{Y}_n = \mathcal{W}_{n+1} \approx \mathcal{G}_n$

$\vec{u} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$, but depends also on $p : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$

$\therefore$ Each of u, v, and $p : \mathbb{R}^3 \to \mathbb{R}$





Thank you all for your time!

Special thanks to Dr. Tian for the help and experience.

Works Cited:

Robinson, James. *Infinite-Dimensional Dynamical Systems*. Cambridge University. 2001.

Pierce, Anthony. *Lecture 8: Solving the Heat equations using finite difference methods*. University of British Columbia. 26 January 2018.

Lu, Lu. *DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators*. CoRR. 2019.

Antil, Lohner, Price. *Data Assimilation with Deep Neural Nets Informed by Nudging*. arXiv. 2021.

Clack, Christopher. *Dynamics of the Lorenz Equations*. University of Manchester. 2006.