MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Application of Physics-Informed Neural Networks in Nonlinear Systems Identification and Parameter Estimation

Ali Forootani

February 19, 2024

CSC group

SINDy Algorithm

SINDy algorithm identifies nonlinear dynamical systems from data, based on the assumption that many systems have relatively few active terms in the dynamics $\dot{x}(t) = f\big(x(t)\big)$.

- SINDy uses sparse regression to identify **active terms** out of a **library** of candidate linear and nonlinear model terms.

1. **Measuring** m snapshots of the state x in time and arrange them into a data matrix

$$X = [x_1 \ x_2 \ \cdots \ x_m]^T$$

2. **Computing the library** of D candidate nonlinear functions $\Theta(X) \in \mathbb{R}^{m \times D}$

$$\Theta(X) = [1 \ X \ X^2 \ \cdots X^d \ \cdots \ sin(X) \ \cdots]$$

3. **Computing time derivatives** of the state $X_t = [\dot{x}_1 \ \dot{x}_2 \ \ldots \ \dot{x}_m]$ and solving $X_t = \Theta(X)\Xi$

Extension to PDEs

Least Squares minimization:

$$\Xi = \arg\min_{\hat{\Xi}} \frac{1}{2} \|U_t - \Theta(U)\hat{\Xi}\|_2^2 + R(\hat{\Xi})$$

where, $R(\Xi)$ is a **regularizer to promote sparsity**

Examples:

- Sequentially Thresholded Least-Squares **(STLS)**, $R(\Xi) = \lambda \|\Xi\|_0$
- Sequentially thresholded ridge regression **(STRidge)**, $R(\Xi) = \lambda_1 \|\Xi\|_0 + \lambda_2 \|\Xi\|_2$

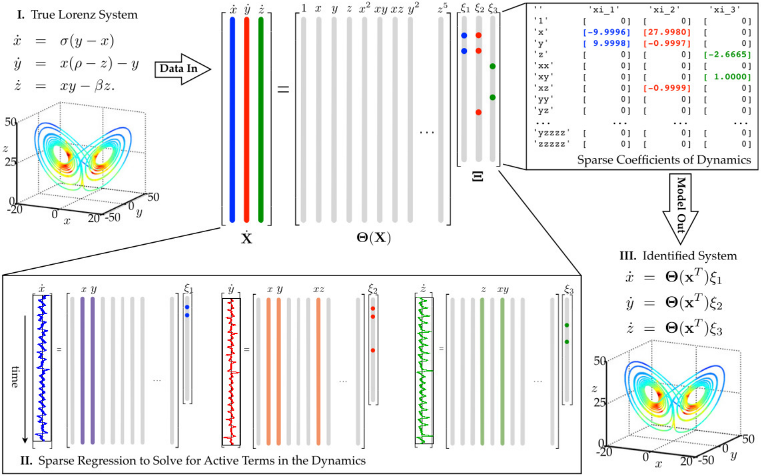
$\| \cdot \|_0$ stands for total number of non-zero elements in a vector

- For **PDEs** Similar to original SINDy with the **library including partial derivatives** if we have $u_t = \mathcal{G}(u), \; x \in \Omega, t \in [0, T]$

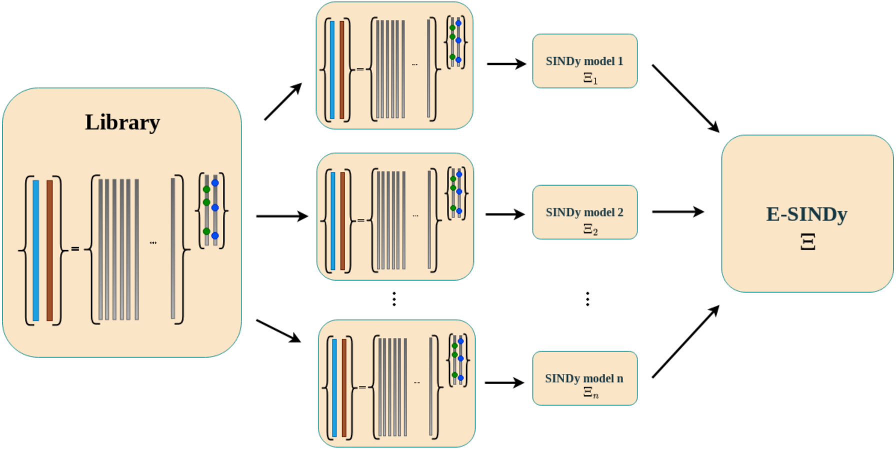
$$\Theta(U) = [1 \; U \; U^2 \; \cdots, \; U_x \; \cdots \; UU_x \; \cdots]$$

SINDy Algorithm

∗ Picture taken from the original paper



Ensemble-SINDy



iNeural-SINDy: **Integrating schemes and neural networks assisted SINDy approach**

SINDy **drawbacks:**

- Sensitivity to accurate derivative information
- Sensitivity to noisy data

Our solution:

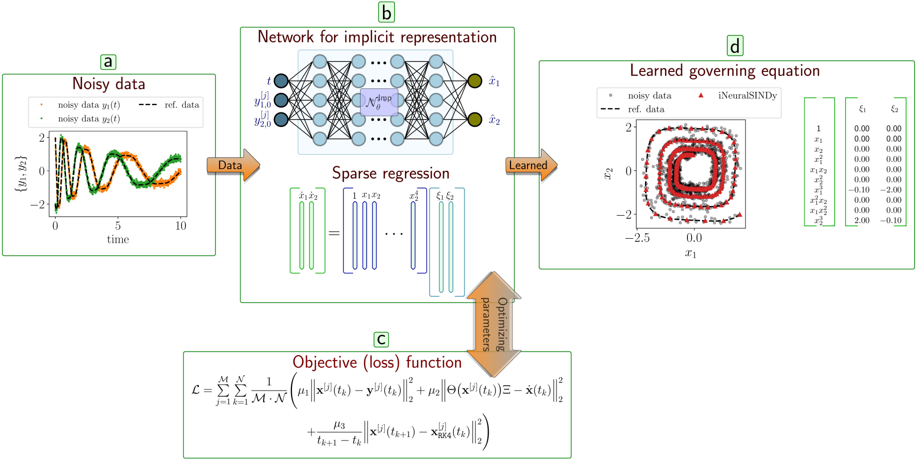
- Using *Deep Neural Network* (DNN) and *integration scheme*

$$\mathcal{L} = \mu_1 \mathcal{L}_{\texttt{MSE}} + \mu_2 \mathcal{L}_{\texttt{deri}} + \mu_3 \mathcal{L}_{\texttt{RK4}}, \quad \mu_1, \mu_2, \mu_3 \in [0,1], \tag{1}$$

$$\mathcal{L}_{\texttt{MSE}} = \frac{1}{\mathcal{N}} \sum_{k=1}^{\mathcal{N}} \left\| x(t_k) - y(t_k) \right\|_2^2, \quad \mathcal{L}_{\texttt{deri}} = \frac{1}{\mathcal{N}} \sum_{k=1}^{\mathcal{N}} \left\| \dot{x}(t_k) - \Theta\big(x(t_k)\big)\Xi \right\|_2^2, \tag{2}$$

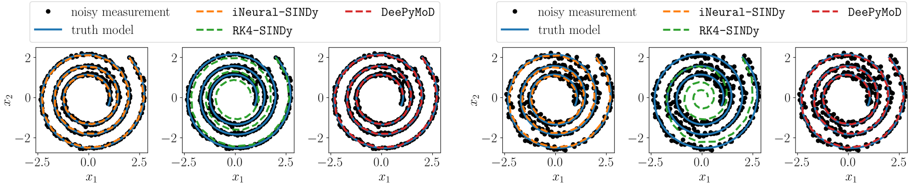
$$\mathcal{L}_{\texttt{RK4}} = \frac{1}{h} \frac{1}{\mathcal{N}} \sum_{k=1}^{\mathcal{N}} \left\| x(t_k) - \mathcal{F}_{\texttt{RK4}}\Big(\Theta\big(x(t_k)\big)\Xi, x(t_k), h_k\Big) \right\|_2^2. \tag{3}$$

iNeural-SINDy



Two-dimensional damped oscillators:

$$\begin{aligned}\dot{x}_1(t) &= -0.1x_1(t) + 2.0x_2(t)\\ \dot{x}_2(t) &= -2.0x_1(t) - 0.1x_2(t).\end{aligned} \tag{4}$$



(a) noise 0.04 (b) noise 0.08

PINN

In PINN we focus on computing data-driven solutions to partial differential equations of the general form

$$u_t + \mathcal{G}(u) = 0,\ x \in \Omega, t \in [0, T], \tag{5}$$

$\mathcal{G}(\cdot)$ is a nonlinear differential operator, and Ω is a subset of $\mathbb{R}^D$.
We define $g(t, x)$ to be given by the left-hand-side of equation (5); i.e.

$$g := u_t + \mathcal{G}(u) \tag{6}$$

as usual we would like to approximate $u(t, x)$ by a DNN. This assumption along with (6) result in a *physics informed neural network* $g(t, x)$.

The shared parameters between the neural networks $u(t,x)$ and $g(t,x)$ can be learned by minimizing the mean squared error loss

$$\mathtt{MSE} = \mathtt{MSE}_u + \mathtt{MSE}_g, \tag{7}$$

where

$$\mathtt{MSE}_u = \frac{1}{\mathcal{N}_u}\sum_{i=1}^{\mathcal{N}_u}\left|u(t_u^i, x_u^i) - u^i\right|^2, \mathtt{MSE}_g = \frac{1}{\mathcal{N}_g}\sum_{i=1}^{\mathcal{N}_g}\left|g(t_g^i, x_g^i)\right|. \tag{8}$$

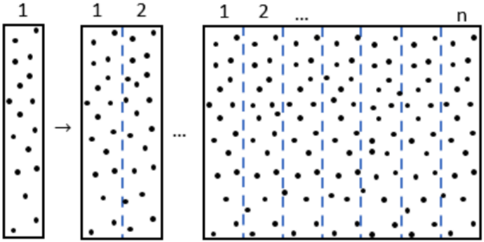
and if we want o solve the PDE then we need also boundary condition, and accordingly its corresponding loss $\mathtt{MSE}_b$.

$$\mathtt{MSE}_b = \frac{1}{\mathcal{N}_b}\sum_{i=1}^{\mathcal{N}_b}\left|u(t_u^b, x_u^b) - u^b\right|^2 \tag{9}$$

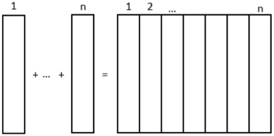
Improved PINN

Some strategies to improve the accuracy of the PINN:

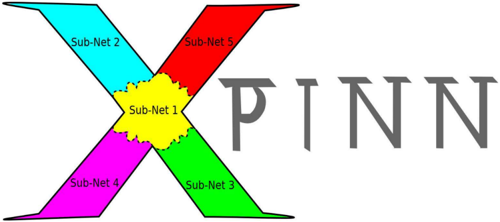
- Adding weights in the loss function
- Mini-batching strategy to improve convergence
- Adaptive time-sampling



- Time-marching



- Extended PINN(XPINN)

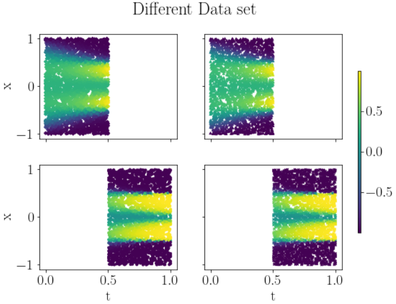


Allen-Cahn equation:

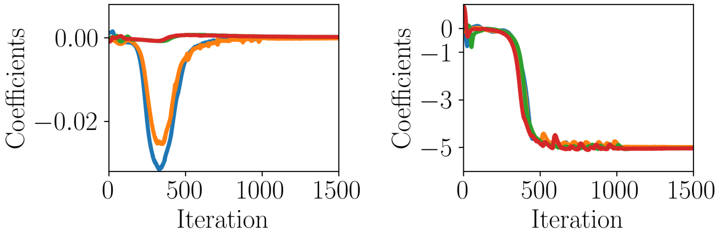
$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \quad x \in [-1, 1], \ t \in [0, 1]$$

Simulation setup for **ensemble** `XPINN`:

- 2 subdomains each having 2 ensemble, using 5% of data set
- DNN structure of each ensemble: $2 * 128 * 128 * 128 * 1$
- Fixing the library terms



Ensembled XPINN



Estimated Coefficients:

$$\text{first: } \begin{bmatrix} 2.0689e-04 \\ -5.0332e+00 \end{bmatrix}, \text{ second: } \begin{bmatrix} -1.3232e-04 \\ -4.9878e+00 \end{bmatrix}$$

$$\text{third: } \begin{bmatrix} 2.0832e-04 \\ -5.0352e+00 \end{bmatrix}, \text{ forth: } \begin{bmatrix} 1.2654e-05 \\ -4.9930e+00 \end{bmatrix}$$

- Greedy Samples computed via `Q-DEIM` algorithm

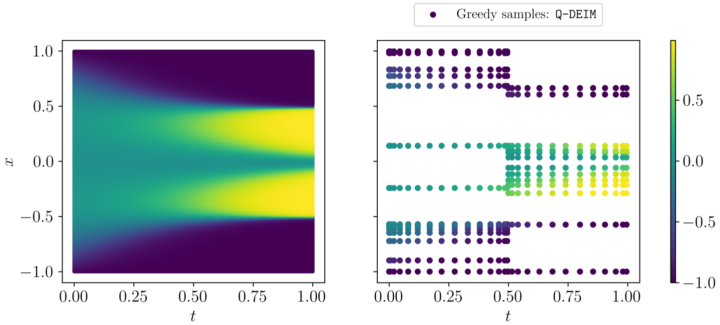
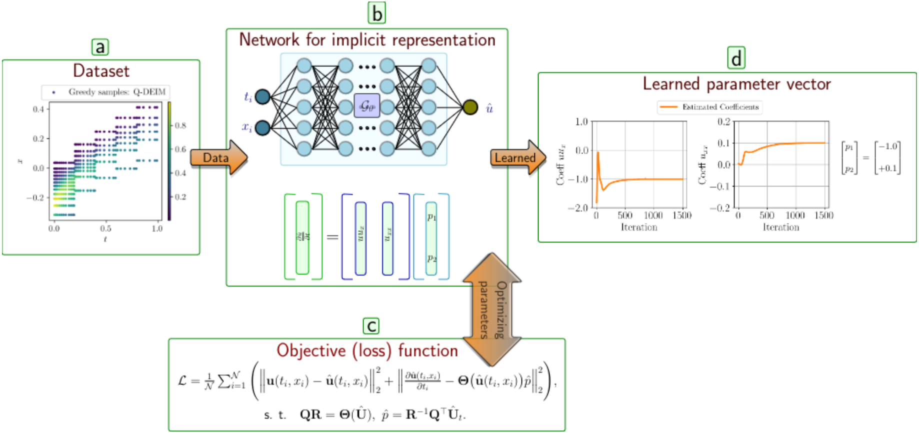


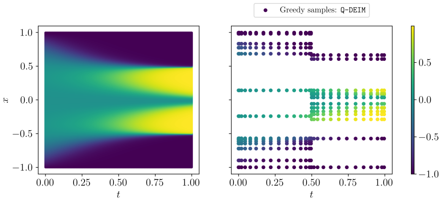
Figure: (left) Entire dataset ; (right) Greedy samples by for Allen-Cahn equation

GS-PINN

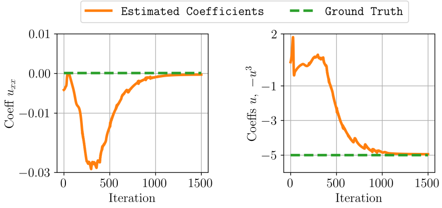

a
Dataset
Greedy samples: Q-DEIM
b
Network for implicit representation
Data
Learned
d
Learned parameter vector
Estimated Coefficients
Iteration
Iteration
Optimizing parameters
c
Objective (loss) function
$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\left\|\mathbf{u}(t_i,x_i) - \hat{\mathbf{u}}(t_i,x_i)\right\|_2^2 + \left\|\frac{\partial \hat{\mathbf{u}}(t_i,x_i)}{\partial t_i} - \mathbf{\Theta}(\hat{\mathbf{u}}(t_i,x_i))\hat{p}\right\|_2^2\right),$
s. t. $\mathbf{QR} = \mathbf{\Theta}(\hat{\mathbf{U}}),\ \hat{p} = \mathbf{R}^{-1}\mathbf{Q}^\top\hat{\mathbf{U}}_t.$

COMPUTATIONAL METHODS IN SYSTEMS AND CONTROL THEORY

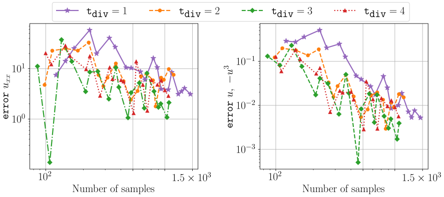
Allen-Cahn



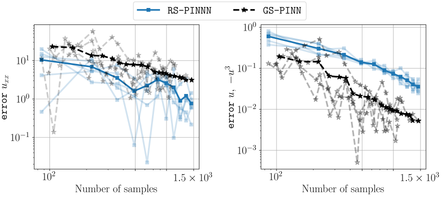
(a) greedy samples



(b) coefficients

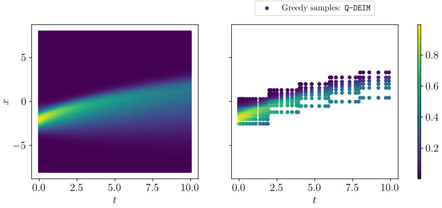


(c) different configuration

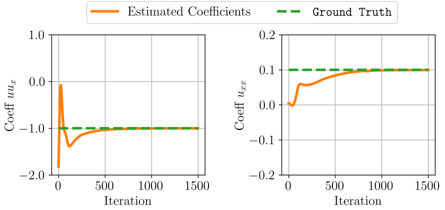


(d) comparison

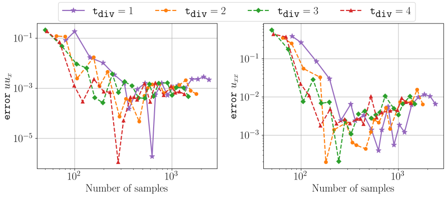
Burgers'



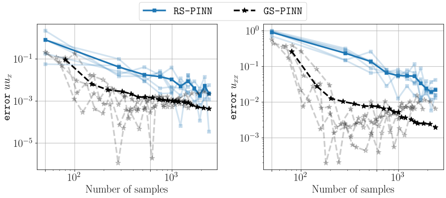
(a) greedy samples



(b) coefficients



(c) different configuration



(d) comparison