MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Identification of Nonlinear Dynamical Systems from Noisy Measurements

Peter Benner and Pawan Goyal

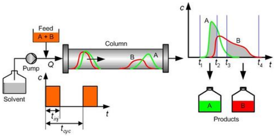
Data-Driven Physics Simulation (DDPS) Seminar
Lawrence Livermore National Laboratory, Livermore
September 30, 2021

Supported by:

DFG-Graduiertenkolleg
MATHEMATISCHE
KOMPLEXITÄTSREDUKTION

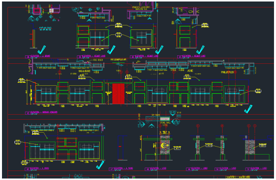
BiG max
MAX PLANCK RESEARCH NETWORK
on big-data-driven materials science

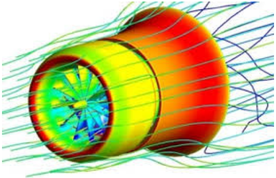
Contents



Dynamical models are important

- to analyze transient behavior under operating conditions,
- for controller design and synthesis,
- parameter optimization,
- prediction of future behavior, e.g., for digital twins.





Problem set-up

- Construct a mathematical model

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

describing dynamics of the process from data.

Problem set-up

- Construct a mathematical model

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

  describing dynamics of the process from data.

- Neural network-based approaches: Recurrent neural networks and long short time memory networks
  - ✓ require no prior knowledge;
  - ✗ by nature, they are not parsimonious;
  - ✗ interpretability (no explicit governing equations);
  - ✗ generalizability.

Problem set-up

- Construct a mathematical model

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

describing dynamics of the process from data.

- Neural network-based approaches: Recurrent neural networks and long short time memory networks
  - ✓ require no prior knowledge;
  - ✗ by nature, they are not parsimonious;
  - ✗ interpretability (no explicit governing equations);
  - ✗ generalizability.

- So, can we pose a reasonable hypothesis to obtain interpretable and generalizable dynamical process models?

Hypothesis

- In a dynamical model,

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

  the function $\mathbf{f}(\mathbf{x}(t))$, defining the vector field, can be given by sparse selection of features of a dictionary of "observables" of $\mathbf{x}(t)$.

Hypothesis

- In a dynamical model,

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

  the function $\mathbf{f}(\mathbf{x}(t))$, defining the vector field, can be given by sparse selection of features of a dictionary of "observables" of $\mathbf{x}(t)$.

- Precisely, we assume $\mathbf{f}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t)) \cdot \xi$, where
  - $\boldsymbol{\Phi}(\mathbf{x})$ is a feature dictionary, i.e.,

$$\boldsymbol{\Phi}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^{\mathscr{P}_2}, \mathbf{x}^{\mathscr{P}_3}, \ldots, \mathbf{e}^{-\mathbf{x}}, \mathbf{e}^{-2\mathbf{x}}, \ldots, \mathbf{sin}(\mathbf{x}), \mathbf{cos}(\mathbf{x}), \ldots\right],$$

    in which the $\mathbf{x}^{\mathscr{P}_i}, i \in \{2, 3, \ldots\}$, denote polynomials, e.g., $\mathbf{x}^{\mathscr{P}_2}$ contains all possible degree-2 polynomials of elements of $\mathbf{x}$:

$$\mathbf{x}^{\mathscr{P}_2} = \left[\mathbf{x}_1^2, \mathbf{x}_1\mathbf{x}_2, \ldots, \mathbf{x}_2^2, \mathbf{x}_2\mathbf{x}_3, \ldots, \mathbf{x}_n^2\right].$$

  - ξ is a sparse vector selecting the right features from the dictionary.

Discovering Governing Equations

Hypothesis

- In a dynamical model,

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

  the function $\mathbf{f}(\mathbf{x}(t))$, defining the vector field, can be given by sparse selection of features of a dictionary of "observables" of $\mathbf{x}(t)$.

- Precisely, we assume $\mathbf{f}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t)) \cdot \xi$, where
  - $\mathbf{\Phi}(\mathbf{x})$ is a feature dictionary, i.e.,

$$\mathbf{\Phi}(\mathbf{x}) = \left[1, \mathbf{x}, \mathbf{x}^{\mathscr{P}_2}, \mathbf{x}^{\mathscr{P}_3}, \ldots, \mathbf{e}^{-\mathbf{x}}, \mathbf{e}^{-2\mathbf{x}}, \ldots, \mathbf{sin}(\mathbf{x}), \mathbf{cos}(\mathbf{x}), \ldots\right],$$

    in which the $\mathbf{x}^{\mathscr{P}_i}, i \in \{2, 3, \ldots\}$, denote polynomials, e.g., $\mathbf{x}^{\mathscr{P}_2}$ contains all possible degree-2 polynomials of elements of $\mathbf{x}$:

$$\mathbf{x}^{\mathscr{P}_2} = \left[\mathbf{x}_1^2, \mathbf{x}_1\mathbf{x}_2, \ldots, \mathbf{x}_2^2, \mathbf{x}_2\mathbf{x}_3, \ldots, \mathbf{x}_n^2\right].$$

  - ξ is a sparse vector selecting the right features from the dictionary.

- Under this hypothesis, there is a large body of available literature, e.g., [..., Bongard/Lipson 07, Schmidt/Lipson '09, Wang et al '11, Daniels/Nemenman '15, Mangan at al '16, Yang et al '16, Schaeffer '17, Raissi et al '19, ...], in particular **SINDy** [Brunton/Proctor/Kutz '16].

Main challenges of the approach:

- Requires derivative information to identify ξ via

$$\min_{\xi} \|\dot{\mathbf{x}}(t) - \Phi(\mathbf{x}(t)) \cdot \xi\|.$$

- If data are sparsely sampled, or are noisy, it is quite challenging to obtain good derivative data.

Main challenges of the approach:

- Requires derivative information to identify ξ via

$$\min_{\xi} \|\dot{\mathbf{x}}(t) - \Phi(\mathbf{x}(t)) \cdot \xi\|.$$

- If data are sparsely sampled, or are noisy, it is quite challenging to obtain good derivative data.

- Construction of a rich enough feature dictionary that at the same time allows efficient computation.

Discovering Governing Equations

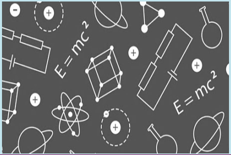
Main challenges of the approach:

- Requires derivative information to identify ξ via

$$\min_{\xi} \|\dot{\mathbf{x}}(t) - \Phi(\mathbf{x}(t)) \cdot \xi\|.$$

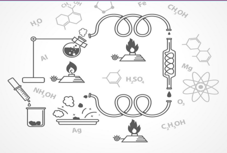
- If data are sparsely sampled, or are noisy, it is quite challenging to obtain good derivative data.

- Construction of a rich enough feature dictionary that at the same time allows efficient computation.
  - Potential remedy: employ additional information to construct the feature dictionary:



Underlying physical laws

Domain knowledge

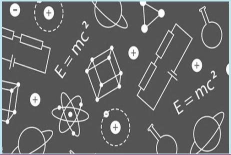


Main challenges of the approach:

- Requires derivative information to identify ξ via

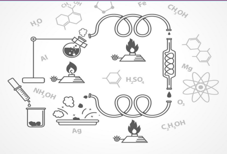
$$\min_{\xi} \|\dot{\mathbf{x}}(t) - \Phi(\mathbf{x}(t)) \cdot \xi\|.$$

- If data are sparsely sampled, or are noisy, it is quite challenging to obtain good derivative data.
  - Remedy: incorporate Runge-Kutta scheme to avoid the need for derivative data.
- Construction of a rich enough feature dictionary that at the same time allows efficient computation.
  - Potential remedy: employ additional information to construct the feature dictionary:



Underlying physical laws

Domain knowledge



Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

- From the large variety of methods, we focus on the 4^{th} order Runge-Kutta scheme.

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

- From the large variety of methods, we focus on the 4^{th} order Runge-Kutta scheme.
  - Predicts $\mathbf{x}(t_{i+1})$ using a weighted sum of the vector field $\mathbf{f}$ at specific locations.

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

- From the large variety of methods, we focus on the 4^{th} order Runge-Kutta scheme.
  - Predicts $\mathbf{x}(t_{i+1})$ using a weighted sum of the vector field $\mathbf{f}$ at specific locations.
  - Precisely, we have

$$\mathbf{x}(t_{i+1}) \approx \mathbf{x}_{i+1} := \mathbf{x}(t_i) + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad \text{with} \quad h := t_{i+1} - t_i,$$

where

$$k_1 = \mathbf{f}\left(\mathbf{x}(t_i)\right), \quad k_2 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_1}{2}\right), \quad k_3 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_2}{2}\right), \quad k_4 = \mathbf{f}\left(\mathbf{x}(t_i) + hk_3\right).$$

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

- From the large variety of methods, we focus on the 4^{th} order Runge-Kutta scheme.
  - Predicts $\mathbf{x}(t_{i+1})$ using a weighted sum of the vector field $\mathbf{f}$ at specific locations.
  - Precisely, we have

$$\mathbf{x}(t_{i+1}) \approx \mathbf{x}_{i+1} := \mathbf{x}(t_i) + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad \text{with} \quad h := t_{i+1} - t_i,$$

where

$$k_1 = \mathbf{f}\left(\mathbf{x}(t_i)\right), \quad k_2 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_1}{2}\right), \quad k_3 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_2}{2}\right), \quad k_4 = \mathbf{f}\left(\mathbf{x}(t_i) + hk_3\right).$$

  - The local truncation error is in $\mathcal{O}(h^5)$, while the total accumulated error is in $\mathcal{O}(h^4)$.

Recall: 4^{th}-order Runge-Kutta Scheme

- Let us consider a differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)).$$

- Assume our goal is to predict $\mathbf{x}(t_{i+1})$, given $\mathbf{x}(t_i)$ and the function $\mathbf{f}(\cdot)$.

- From the large variety of methods, we focus on the 4^{th} order Runge-Kutta scheme.
  - Predicts $\mathbf{x}(t_{i+1})$ using a weighted sum of the vector field $\mathbf{f}$ at specific locations.
  - Precisely, we have

$$\mathbf{x}(t_{i+1}) \approx \mathbf{x}_{i+1} := \mathbf{x}(t_i) + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \quad \text{with} \quad h := t_{i+1} - t_i,$$

  where

$$k_1 = \mathbf{f}\left(\mathbf{x}(t_i)\right), \quad k_2 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_1}{2}\right), \quad k_3 = \mathbf{f}\left(\mathbf{x}(t_i) + h\frac{k_2}{2}\right), \quad k_4 = \mathbf{f}\left(\mathbf{x}(t_i) + hk_3\right).$$

  - The local truncation error is in $\mathcal{O}(h^5)$, while the total accumulated error is in $\mathcal{O}(h^4)$.
  - We use the notation $\mathbf{x}_{k+1} \approx \mathcal{F}_{\text{RK4}}\left(\mathbf{f}, \mathbf{x}(t_k), h\right)$.

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.
- But the vector field $\mathbf{f}(\cdot)$ is not known — this is what we want to find!

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.
- But the vector field $\mathbf{f}(\cdot)$ is not known — this is what we want to find!
- Based on the sparsity hypothesis:

$$\mathbf{f}(\mathbf{x}) := \mathbf{\Phi}(\mathbf{x})\xi,$$

where $\mathbf{\Phi}(\mathbf{x})$ is a feature dictionary, and ξ selects right features from the dictionary.

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.
- But the vector field $\mathbf{f}(\cdot)$ is not known — this is what we want to find!
- Based on the sparsity hypothesis:

$$\mathbf{f}(\mathbf{x}) := \boldsymbol{\Phi}(\mathbf{x})\xi,$$

  where $\boldsymbol{\Phi}(\mathbf{x})$ is a feature dictionary, and ξ selects right features from the dictionary.
- Thus, we have

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t))\xi,$$

  or for each component of $\mathbf{x}(t)$:

$$[\dot{\mathbf{x}}_1(t), \ldots, \dot{\mathbf{x}}_n(t)] = [\Phi(\mathbf{x}(t))\xi_1, \cdots, \Phi(\mathbf{x}(t))\xi_n]$$

Blending Runge-Kutta with Sparse Regression (RK4-SINDy)

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.
- But the vector field $\mathbf{f}(\cdot)$ is not known — this is what we want to find!
- Based on the sparsity hypothesis:

$$\mathbf{f}(\mathbf{x}) := \mathbf{\Phi}(\mathbf{x})\xi,$$

  where $\mathbf{\Phi}(\mathbf{x})$ is a feature dictionary, and ξ selects right features from the dictionary.
- Thus, we have

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t))\xi,$$

  or for each component of $\mathbf{x}(t)$:

$$[\dot{\mathbf{x}}_1(t), \ldots, \dot{\mathbf{x}}_n(t)] = [\Phi(\mathbf{x}(t))\xi_1, \cdots, \Phi(\mathbf{x}(t))\xi_n]$$

- If derivative information is known, one can apply the standard Sparse Identification of Nonlinear Dynamics (Std-SINDy) approach [Brunton at al. '16].

Blending Runge-Kutta with Sparse Regression (RK4-SINDy)

- We have seen how to fuse the RK4 scheme to predict variables at the next time step given the vector field $\mathbf{f}$.
- But the vector field $\mathbf{f}(\cdot)$ is not known — this is what we want to find!
- Based on the sparsity hypothesis:

$$\mathbf{f}(\mathbf{x}) := \mathbf{\Phi}(\mathbf{x})\xi,$$

  where $\mathbf{\Phi}(\mathbf{x})$ is a feature dictionary, and ξ selects right features from the dictionary.
- Thus, we have

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t))\xi,$$

  or for each component of $\mathbf{x}(t)$:

$$[\dot{\mathbf{x}}_1(t), \ldots, \dot{\mathbf{x}}_n(t)] = [\Phi(\mathbf{x}(t))\xi_1, \cdots, \Phi(\mathbf{x}(t))\xi_n]$$

- If derivative information is known, one can apply the standard Sparse Identification of Nonlinear Dynamics (Std-SINDy) approach [Brunton at al. '16].
- **Here: leverage RK4 scheme to avoid derivative information!**

- For simplicity, let us consider two samples of $\mathbf{x}$ at time t_i and t_{i+1}.

- For simplicity, let us consider two samples of $\mathbf{x}$ at time t_i and t_{i+1}.
- With the help of RK4, we have:

$$[\mathbf{x}_1(t_{i+1}), \ldots, \mathbf{x}_n(t_{i+1})] \approx [\mathbf{x}_1(t_i), \ldots, \mathbf{x}_n(t_i)] + \mathcal{F}_{\mathsf{RK4}}\Big(\underbrace{(\Phi, \xi_i)}_{\equiv \mathbf{f}}, \mathbf{x}(t_i), h\Big),$$

where

$$\mathcal{F}_{\mathsf{RK4}}\left((\Phi, \xi_i), \mathbf{x}(t_i), h\right) = \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

with

$$\begin{aligned} k_1 &= [\mathbf{\Phi}(\mathbf{x}(t))\xi_1, \ldots, \mathbf{\Phi}(\mathbf{x}(t))\xi_n], \\ k_2 &= [\mathbf{\Phi}(\tilde{\mathbf{x}}_1(t))\xi_1 \ldots, \mathbf{\Phi}(\tilde{\mathbf{x}}_1(t))\xi_n] \quad \text{with} \quad \tilde{\mathbf{x}}_1 = \mathbf{x} + \tfrac{h}{2} \cdot k_1 \\ k_3 &= [\mathbf{\Phi}(\tilde{\mathbf{x}}_2(t))\xi_2, \ldots, \mathbf{\Phi}(\tilde{\mathbf{x}}_2(t))\xi_n] \quad \text{with} \quad \tilde{\mathbf{x}}_2 = \mathbf{x} + \tfrac{h}{2} \cdot k_2, \\ k_4 &= [\mathbf{\Phi}(\tilde{\mathbf{x}}_3(t))\xi_3, \ldots, \mathbf{\Phi}(\tilde{\mathbf{x}}_3(t))\xi_n] \quad \text{with} \quad \tilde{\mathbf{x}}_1 = \mathbf{x} + h \cdot k_3. \end{aligned}$$

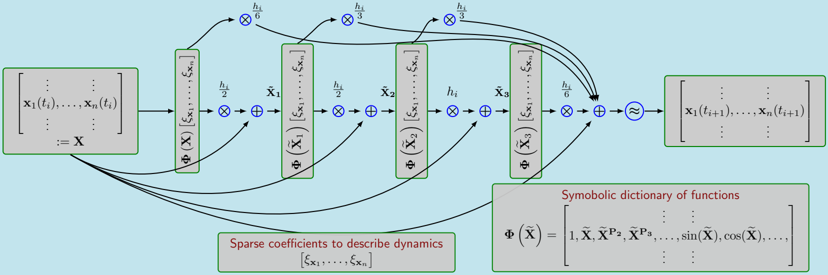
Blending Runge-Kutta with Sparse Regression (RK4-SINDy)

- For simplicity, let us consider two samples of $\mathbf{x}$ at time t_i and t_{i+1}.
- With the help of RK4, we have:

$$[\mathbf{x}_1(t_{i+1}), \ldots, \mathbf{x}_n(t_{i+1})] \approx [\mathbf{x}_1(t_i), \ldots, \mathbf{x}_n(t_i)] + \mathcal{F}_{\mathsf{RK4}}\Big(\underbrace{(\Phi, \xi_i)}_{\equiv \mathbf{f}}, \mathbf{x}(t_i), h\Big),$$

where

$$\mathcal{F}_{\mathsf{RK4}}\left((\Phi, \xi_i), \mathbf{x}(t_i), h\right) = \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$



Optimization Problem Statement

- Determine the sparsest ξ_i's, minimizing

$$\Big\| \underbrace{[\mathbf{x}_1(t_{i+1}),\ldots,\mathbf{x}_n(t_{i+1})]}_{\mathbf{x}(t_{i+1})} - \underbrace{[\mathbf{x}_1(t_i),\ldots,\mathbf{x}_n(t_i)]}_{\mathbf{x}(t_i)} - \mathcal{F}_{\mathsf{RK4}}\left((\boldsymbol{\Phi},\xi_i),\mathbf{x}(t_i),h\right)\Big\|.$$

Optimization Problem Statement

- Determine the sparsest ξ_i's, minimizing

$$\Big\| \underbrace{[\mathbf{x}_1(t_{i+1}), \ldots, \mathbf{x}_n(t_{i+1})]}_{\mathbf{x}(t_{i+1})} - \underbrace{[\mathbf{x}_1(t_i), \ldots, \mathbf{x}_n(t_i)]}_{\mathbf{x}(t_i)} - \mathcal{F}_{\mathsf{RK4}}\left((\mathbf{\Phi}, \xi_i), \mathbf{x}(t_i), h\right) \Big\|.$$

- The above problem is NP hard $\rightsquigarrow$ not possible to solve in polynomial time.

Optimization Problem Statement

- Determine the sparsest ξ_i's, minimizing

$$\left\| \underbrace{[\mathbf{x}_1(t_{i+1}), \ldots, \mathbf{x}_n(t_{i+1})]}_{\mathbf{x}(t_{i+1})} - \underbrace{[\mathbf{x}_1(t_i), \ldots, \mathbf{x}_n(t_i)]}_{\mathbf{x}(t_i)} - \mathcal{F}_{\mathsf{RK4}}\left((\boldsymbol{\Phi}, \xi_i), \mathbf{x}(t_i), h\right) \right\|.$$

- The above problem is NP hard $\rightsquigarrow$ not possible to solve in polynomial time.
- Remedy: ℓ_1-norm relaxation ("LASSO"), i.e.,

$$\left\| \mathbf{x}(t_{i+1}) - \mathbf{x}(t_i) - \mathcal{F}_{\mathsf{RK4}}\left(\boldsymbol{\Phi}, \xi_i, \mathbf{x}(t_i), h\right) \right\| + \gamma \cdot \sum_i \|\xi_i\|_1.$$

Optimization Problem Statement

- Determine the sparsest ξ_i's, minimizing

$$\Big\| \underbrace{[\mathbf{x}_1(t_{i+1}), \ldots, \mathbf{x}_n(t_{i+1})]}_{\mathbf{x}(t_{i+1})} - \underbrace{[\mathbf{x}_1(t_i), \ldots, \mathbf{x}_n(t_i)]}_{\mathbf{x}(t_i)} - \mathcal{F}_{\mathsf{RK4}}\left((\mathbf{\Phi}, \xi_i), \mathbf{x}(t_i), h\right) \Big\|.$$

- The above problem is NP hard $\rightsquigarrow$ not possible to solve in polynomial time.
- Remedy: ℓ_1-norm relaxation ("LASSO"), i.e.,

$$\Big\| \mathbf{x}(t_{i+1}) - \mathbf{x}(t_i) - \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{\Phi}, \xi_i, \mathbf{x}(t_i), h\right) \Big\| + \gamma \cdot \sum_i \|\xi_i\|_1.$$

- Under a certain condition (related to the restricted isometry property), the relaxed optimization problem may yield the sparsest solution.

Optimization Problem Statement

- Determine the sparsest ξ_i's, minimizing

$$\Big\| \underbrace{[\mathbf{x}_1(t_{i+1}), \dots, \mathbf{x}_n(t_{i+1})]}_{\mathbf{x}(t_{i+1})} - \underbrace{[\mathbf{x}_1(t_i), \dots, \mathbf{x}_n(t_i)]}_{\mathbf{x}(t_i)} - \mathcal{F}_{\mathsf{RK4}}\left((\mathbf{\Phi}, \xi_i), \mathbf{x}(t_i), h\right) \Big\|.$$

- The above problem is NP hard $\rightsquigarrow$ not possible to solve in polynomial time.
- Remedy: ℓ_1-norm relaxation ("LASSO"), i.e.,

$$\Big\| \mathbf{x}(t_{i+1}) - \mathbf{x}(t_i) - \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{\Phi}, \xi_i, \mathbf{x}(t_i), h\right) \Big\| + \gamma \cdot \sum_i \|\xi_i\|_1.$$

- Under a certain condition (related to the restricted isometry property), the relaxed optimization problem may yield the sparsest solution.
- But often, in practice, this condition is not full-filled. Therefore, we look at a sequential thresholding type algorithm (similar to [Brunton et al. '16]).

- In essence, the idea is to set small coefficients to zero and solve the optimization for the rest of the non-zero coefficients.

Blending Runge-Kutta with Sparse Regression (RK4-SINDy)

Sequential Thresholding Approach

[GOYAL/B. '21]

- In essence, the idea is to set small coefficients to zero and solve the optimization for the rest of the non-zero coefficients.

Algorithm 2 Sequential Thresholding Procedure (Fix Thresholding)

Input: Measurement data $\{\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_\mathcal{N})\}$ and the cutoff parameter λ.

Solve the following optimization problem to get $\Theta := \{\xi_1, \ldots, \xi_n\}$:

$$\sum_i \|\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i) - \mathcal{F}_{\mathsf{RK4}}\left(\mathbf{\Phi}, \xi_i, \mathbf{x}(t_i), h\right)\| + \gamma \cdot \|\Theta\|_1. \quad (1)$$

```
small_idx = (|Θ| < λ)                                   ▷ Determine indices at which coefficients are < λ
Err = ‖Θ (small_idx)‖
while Err ≠ 0 do
    Update Θ by solving (1) with the constraint Θ (small_idx) = 0
    small_idx = (|Θ| < λ)                               ▷ Determine indices at which coefficients are < λ
    Err = ‖Θ (small_idx)‖
end while
```

Output: The sparse $\mathbf{\Theta}$ that picks right features from the dictionary.

Additional Remarks

- The optimization problem

$$\sum_i \|\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i) - \mathcal{F}_{\mathsf{RK4}}\left(\boldsymbol{\Phi}, \xi_i, \mathbf{x}(t_i), h\right)\| + \gamma \cdot \|\Theta\|_{l_1}.$$

  is nonlinear and non-convex, and there is in general no analytical and no unique solution
- Here, we use gradient based optimization, e.g., ADAM [Kingma/Ba '15].
- For gradient computation, we utilize the computational graph based library PyTorch.
- Furthermore, the optimization problem involves the thresholding parameter λ, which can be found by cross-validation.
- Alternatively, we propose an iterative thresholding algorithm in which we truncate the smallest non-zero element in each iteration to find the sparsest solution. [Goyal/B. '21]

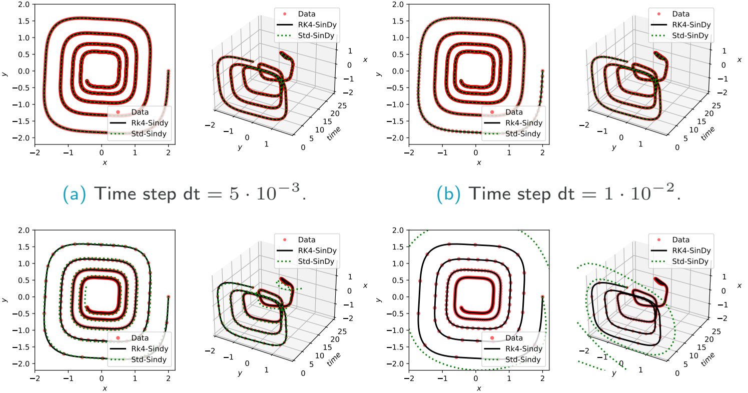
Cubic Oscillator

- Consider a cubic damped oscillator, governed by

$$\dot{\mathbf{x}}(t) = -0.1\mathbf{x}(t)^3 + 2.0\mathbf{y}(t)^3,$$
$$\dot{\mathbf{y}}(t) = -2.0\mathbf{x}(t)^3 - 0.1\mathbf{y}(t)^3.$$

- We construct a feature dictionary, containing polynomials features up to degree 5.
- We compare the proposed method RK4-SINDy with Std-SINDy.

([Brunton et al. '16])



(a) Time step $\mathrm{dt} = 5 \cdot 10^{-3}$.

(b) Time step $\mathrm{dt} = 1 \cdot 10^{-2}$.

(c) Time step $\mathrm{dt} = 5 \cdot 10^{-2}$.

(d) Time step $\mathrm{dt} = 1 \cdot 10^{-1}$.

Figure: Cubic 2D model: A comparison of the transient responses of discovered models using data at different regular time-steps.

Time step	RK4-SINDy	Std-SINDy
$5 \cdot 10^{-3}$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.996\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.996\mathbf{x}(t)^3 - 0.099\mathbf{y}(t)^3$
$1 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.099\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t)^3 + 1.994\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.996\mathbf{x}(t)^3 - 0.099\mathbf{y}(t)^3$
$5 \cdot 10^{-2}$	$\dot{\mathbf{x}}(t) = -0.100\mathbf{x}(t)^3 + 1.995\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -1.997\mathbf{x}(t)^3 - 0.100\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = -0.092\mathbf{x}(t)^3 + 2.002\mathbf{y}(t)^3$ $+ 0.076\mathbf{x}^4\mathbf{y} - 0.107\mathbf{x}^2\mathbf{y}^3$ $\dot{\mathbf{y}}(t) = -1.981\mathbf{x}(t)^3 - 0.092\mathbf{y}(t)^3$ $+ 0.078\mathbf{x}^3\mathbf{y}^2 - 0.068\mathbf{x}\mathbf{y}^4$
$1 \cdot 10^{-1}$	$\dot{\mathbf{x}}(t) = -0.103\mathbf{x}(t)^3 + 2.000\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = -2.001\mathbf{x}(t)^3 - 0.098\mathbf{y}(t)^3$	$\dot{\mathbf{x}}(t) = 0.090\mathbf{x}(t) - 0.097\mathbf{x}(t)^2 - 0.463\mathbf{x}(t)^3$ $+ \cdots + 0.381\mathbf{x}(t)^3\mathbf{y}(t)^2 - 0.258\mathbf{x}(t)\mathbf{y}(t)^4$ $\dot{\mathbf{y}}(t) = 0.100\mathbf{x}(t) + 0.104\mathbf{x}(t)^2 + 0.051\mathbf{x}(t)\mathbf{y}(t)$ $+ \cdots + 0.381\mathbf{x}(t)^3\mathbf{y}(t)^2 - 0.258\mathbf{x}(t)\mathbf{y}(t)^4$

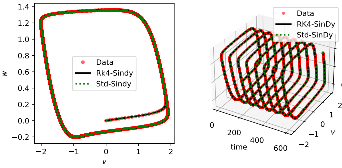
Table: Cubic 2D model: The table reports the discovered governing equations by employing RK4-SINDy and Std-SINDy.

Fitz-Hugh Nagumo Model

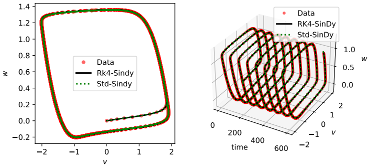
- Next, we consider the Fitz-Hugh Nagumo system, a basic model for neuron spiking:

$$\begin{aligned} \mathbf{v}(t) &= \mathbf{v}(t) - \mathbf{w}(t) - \frac{1}{3}\mathbf{v}(t)^3 + 0.5, \\ \mathbf{w}(t) &= 0.040\mathbf{v}(t) - 0.028\mathbf{w}(t) + 0.032. \end{aligned} \tag{2}$$

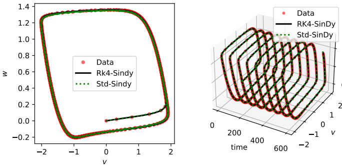
- We construct a feature dictionary, containing polynomials up to degree 5.
- We compare the proposed method RK4-SINDy with Std-SINDy.



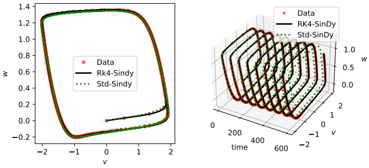
(a) Time step dt $= 1.0 \cdot 10^{-1}$.



(b) Time step dt $= 2.5 \cdot 10^{-1}$.



(c) Time step dt $= 5.0 \cdot 10^{-1}$.



(d) Time step dt $= 7.5 \cdot 10^{-1}$.

Figure: FHN model: A comparison of the transient responses of the discovered differential equations using data collected at different regular time-steps.

dt	RK4-SINDy	Std-SINDy
$1.0 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.499 + 0.998\mathbf{v} - 0.998\mathbf{w} - 0.333\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.498 + 0.996\mathbf{v} - 0.996\mathbf{w} - 0.332\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$2.5 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.499 + 0.998\mathbf{v} - 0.998\mathbf{w} - 0.333\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.494 + 0.985\mathbf{v} - 0.989\mathbf{w} - 0.328\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$5.0 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.501 + 1.001\mathbf{v} - 1.001\mathbf{w} - 0.334\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.482 + 0.943\mathbf{v} - 0.959\mathbf{w}$ $- 0.034\mathbf{vw} - 0.311\mathbf{v}^3 + 0.024\mathbf{vw}^2$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$
$7.5 \cdot 10^{-1}$	$\dot{\mathbf{v}}(t) = 0.502 + 1.001\mathbf{v} - 1.003\mathbf{w} - 0.334\mathbf{v}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.027\mathbf{w}$	$\dot{\mathbf{v}}(t) = 0.459 + 0.816\mathbf{v} - 0.982\mathbf{w}$ $- 0.013\mathbf{v}^2 + \cdots + 0.131\mathbf{vw}^2 - 0.021\mathbf{w}^3$ $\dot{\mathbf{w}}(t) = 0.032 + 0.040\mathbf{v} - 0.028\mathbf{w}$

Table: FHN model: Discovered models using data at various time-step using RK4-SINDy and Std-SINDy.

- Observe that for data collected large steps, the standard SINDy fails, potentially due to large error in derivative estimates.
- On the other hand, RK4-SINDy accurately discovers dynamical models as it does not require derivative information explicitly.

Extensions – Parametric Systems

- The approach readily applies to parametric systems.
- Consider a parametric system *(where parameters do not vary with time!)*

$$\dot{\mathbf{x}}(t;\mu) = \mathbf{f}(\mathbf{x}(t;\mu)).$$

- Reformulation with state vector augmented by parameters as $\mathbf{x}_\mu(t) = \left[\mathbf{x}(t), \mu\right]$.
- Consequently, we have

$$\dot{\mathbf{x}}_\mu(t) = \left[\mathbf{f}(\mathbf{x}_\mu(t)), 0\right].$$

- Hence, we can readily apply RK4-SINDy by creating a dictionary involving the parameters μ.

Hopf normal form

- Dynamics of parametric Hopf normal form is given by

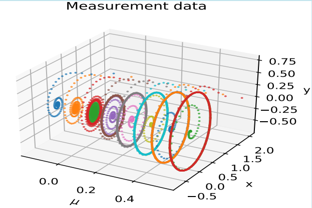
$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mu\mathbf{x}(t) - \mathbf{y}(t) - \mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right),\\ \dot{\mathbf{y}}(t) &= \mathbf{x}(t) + \mu\mathbf{y}(t) - \mathbf{y}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right).\end{aligned}$$

Hopf normal form

- Dynamics of parametric Hopf normal form is given by

$$\dot{\mathbf{x}}(t) = \mu\mathbf{x}(t) - \mathbf{y}(t) - \mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right),$$
$$\dot{\mathbf{y}}(t) = \mathbf{x}(t) + \mu\mathbf{y}(t) - \mathbf{y}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right).$$

- We collect measurements for various initial conditions and parameters with time step 0.2 which are corrupted by adding 1% Gaussian noise.

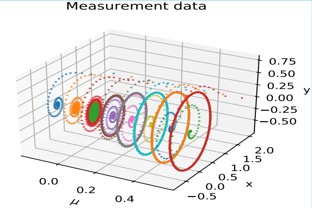


Hopf normal form

- Dynamics of parametric Hopf normal form is given by

$$\dot{\mathbf{x}}(t) = \mu\mathbf{x}(t) - \mathbf{y}(t) - \mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right),$$
$$\dot{\mathbf{y}}(t) = \mathbf{x}(t) + \mu\mathbf{y}(t) - \mathbf{y}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right).$$

- We collect measurements for various initial conditions and parameters with time step 0.2 which are corrupted by adding 1% Gaussian noise.
- We construct a dictionary, containing polynomials up to degree 3, including the parameter.



Numerical Examples
Hopf normal form

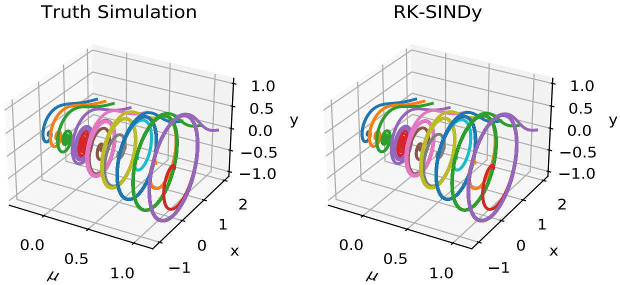


Figure: Simulations for parameters from a test set different from the training parameters.

Method	Discovered model
RK4-SINDy	$\dot{\mathbf{x}}(t) = 1.001\mu\mathbf{x}(t) - 1.001\mathbf{y}(t) - 0.996\mathbf{x}(t)\left(\mathbf{x}(t)^2 + \mathbf{y}(t)^2\right)$ $\dot{\mathbf{y}}(t) = 1.001\mathbf{x}(t) + 1.010\mu\mathbf{y}(t) - 1.006\mathbf{x}(t)^2\mathbf{y}(t) - 1.004\mathbf{y}(t)^3$
Std-SINDy	$\dot{\mathbf{x}}(t) = -0.961\mathbf{y}(t) + 0.719\mu\mathbf{x}(t) + 0.822\mu\mathbf{y}(t) - 0.735\mathbf{x}(t)^3$ $- 1.044\mathbf{x}(t)^2\mathbf{y} - 0.686\mathbf{x}(t)\mathbf{y}(t)^2 - 0.846\mathbf{y}(t)^3$ $\dot{\mathbf{y}}(t) = 0.986\mathbf{x}(t) + 0.899\mu\mathbf{y}(t) - 0.882\mathbf{x}(t)^2\mathbf{y}(t) - 0.904\mathbf{y}(t)^3.$

Challenge in rational functions:

- Several dynamical models are given by rational functions, specially in chemical and biological modeling.

Extension of RK4-SINDy to Rational Non-linearity

Challenge in rational functions:

- Several dynamical models are given by rational functions, specially in chemical and biological modeling.
- E.g., if we were to discover the model: $\dot{\mathbf{x}}(t) = -\dfrac{\mathbf{x}(t)}{1 + 0.3\mathbf{x}(t)}$, then, in a classical dictionary based learning, we precisely need to have a feature containing $\dfrac{1}{1 + 0.3\mathbf{x}(t)}$.

Extension of RK4-SINDy to Rational Non-linearity

Challenge in rational functions:

- Several dynamical models are given by rational functions, specially in chemical and biological modeling.
- E.g., if we were to discover the model: $\dot{\mathbf{x}}(t) = -\dfrac{\mathbf{x}(t)}{1+0.3\mathbf{x}(t)}$, then, in a classical dictionary based learning, we precisely need to have a feature containing $\dfrac{1}{1+0.3\mathbf{x}(t)}$.
- This puts severe restrictions; many times not even feasible to guess such features.

Extension of RK4-SINDy to Rational Non-linearity

Challenge in rational functions:

- Several dynamical models are given by rational functions, specially in chemical and biological modeling.
- E.g., if we were to discover the model: $\dot{\mathbf{x}}(t) = -\dfrac{\mathbf{x}(t)}{1 + 0.3\mathbf{x}(t)}$, then, in a classical dictionary based learning, we precisely need to have a feature containing $\dfrac{1}{1 + 0.3\mathbf{x}(t)}$.
- This puts severe restrictions; many times not even feasible to guess such features.

Remedy

- We hypothesize that the right-hand side function $\mathbf{f}(\mathbf{x}(t))$ defining the dynamical systems can be given by a ratio of two functions in which each function is defined by selecting features from an appropriate dictionary, i.e.,

Extension of RK4-SINDy to Rational Non-linearity

Challenge in rational functions:

- Several dynamical models are given by rational functions, specially in chemical and biological modeling.
- E.g., if we were to discover the model: $\dot{\mathbf{x}}(t) = -\dfrac{\mathbf{x}(t)}{1 + 0.3\mathbf{x}(t)}$, then, in a classical dictionary based learning, we precisely need to have a feature containing $\dfrac{1}{1 + 0.3\mathbf{x}(t)}$.
- This puts severe restrictions; many times not even feasible to guess such features.

Remedy

- We hypothesize that the right-hand side function $\mathbf{f}(\mathbf{x}(t))$ defining the dynamical systems can be given by a ratio of two functions in which each function is defined by selecting features from an appropriate dictionary, i.e.,
- $\mathbf{f}(\mathbf{x}(t)) = \dfrac{\mathbf{g}_{\mathsf{N}}(\mathbf{x}(t))}{1 + \mathbf{g}_{\mathsf{D}}(\mathbf{x}(t))} = \dfrac{\mathbf{\Phi}(\mathbf{x})\xi_{\mathsf{N}}}{1 + \mathbf{\Phi}(\mathbf{x})\xi_{\mathsf{D}}}$, where $\mathbf{\Phi}(\mathbf{x})$ is a dictionary, and $\xi_{\mathsf{N},\mathsf{D}}$ are sparse vectors.

Michaelis-Menten kinetics

- Michaelis-Menten kinetics describes an Enzyme dynamics and is governed by

$$\dot{\mathbf{s}}(t) = 0.6 - \frac{1.5\mathbf{s}(t)}{0.3 + \mathbf{s}(t)}.$$

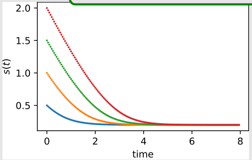
Michaelis-Menten kinetics

- Michaelis-Menten kinetics describes an Enzyme dynamics and is governed by

$$\dot{\mathbf{s}}(t) = 0.6 - \frac{1.5\mathbf{s}(t)}{0.3 + \mathbf{s}(t)}.$$

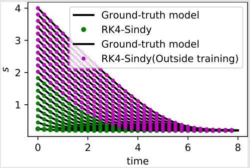
- We collect data using 4 trajectories.
- We construct a dictionary of polynomial features of degree 3.
- Learn a parsimonious model using RK4-SINDy for rational nonlinear systems.

Training trajectories



Learned model (normalized)

$$\dot{\tilde{\mathbf{s}}}(t) = \frac{-0.666 - 1.335\tilde{\mathbf{s}}(t)}{1.000 + 0.512\tilde{\mathbf{s}}(t)}$$



So far

- We have presented the discovery of dynamical models using sparse regression combined with an RK4 scheme

  $\rightsquigarrow$ no derivative estimate required!

So far

- We have presented the discovery of dynamical models using sparse regression combined with an RK4 scheme

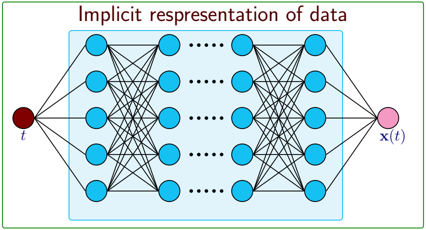
  $\rightsquigarrow$ no derivative estimate required!

- Bottleneck:
  - Success depends on quality of dictionary.
  - Although RK4-SINDy appears to be robust for noise up to 5%, for higher level noise, it may fail.

Remedy

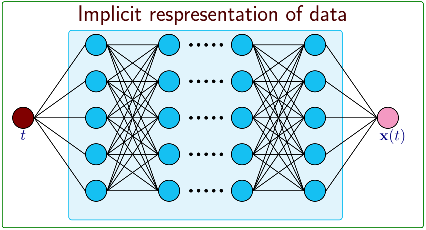
- We investigate a black-box modeling approach based on neural networks.
- The goal is twofold:
  - Denoising the measurement data (for noise $> 10\%$).
  - Also, a black-box model, describing dynamics
    $\rightsquigarrow$ no prior knowledge is needed (e.g., of dictionary).

- Learn implicit representation of measurement, i.e., for given time t as input to the network, the output is $\mathbf{x}(t)$.



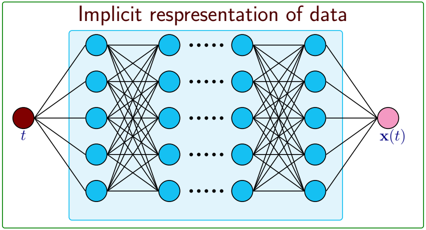
Implicit Neural Networks

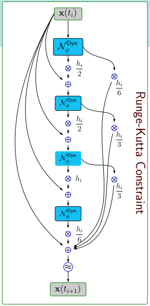
- Learn implicit representation of measurement, i.e., for given time t as input to the network, the output is $\mathbf{x}(t)$.
- Since measurements are noisy, we need to regularize the network which otherwise would overfit-



- Learn implicit representation of measurement, i.e., for given time t as input to the network, the output is $\mathbf{x}(t)$.
- Since measurements are noisy, we need to regularize the network which otherwise would overfit-
- We regularize using a Runge-Kutta scheme:
  - The output of the implicit network should be such that it follows a RK4 scheme.
  - To leverage RK4, we require a function, defining the vector field $\mathbf{f}(\mathbf{x}(t))$.
  - So, let us assume, the vector field is defined by a neural network $\mathcal{N}_{\mathbf{\Phi}}^{\mathsf{Dyn}}(\mathbf{x})$, i.e.,

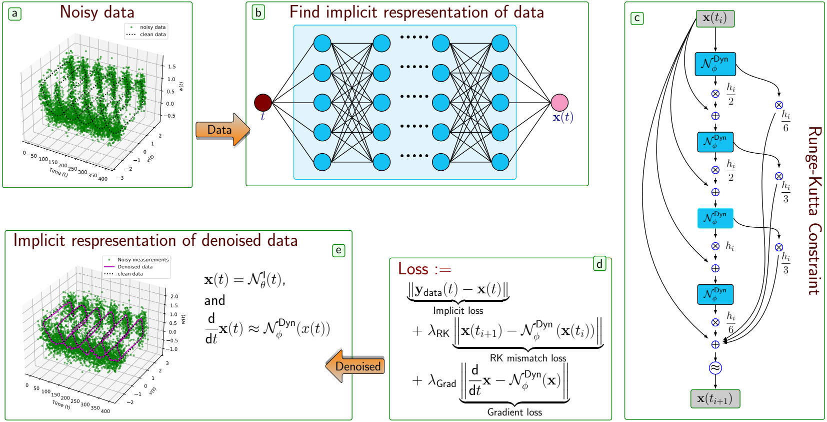
$$\dot{\mathbf{x}}(t) = \mathcal{N}_{\mathbf{\Phi}}^{\mathsf{Dyn}}(\mathbf{x}).$$





Implicit Neural Networks with Runge-Kutta Constraints

- Combination all these components:

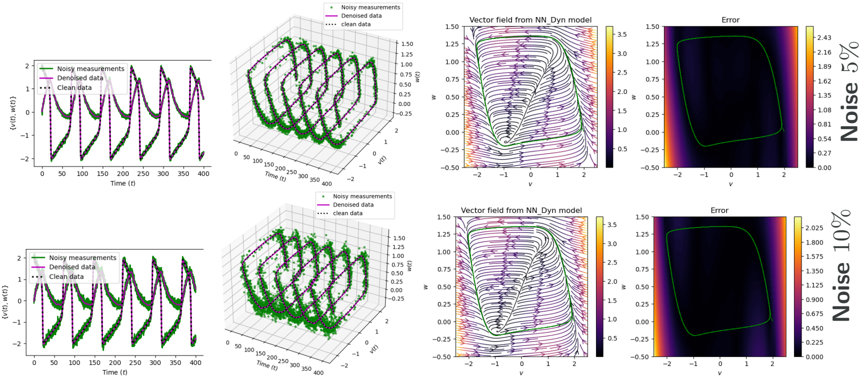


- Note that this provides an implicit network $\mathcal{N}_\theta^{\mathsf{I}}$ generating denoised data, and a network $\mathcal{N}_{\Phi}^{\mathsf{Dyn}}$ defining the dynamics.

- Consider again the Fitz-Hugh Nagumo model, describing neuron spiking:

$$\begin{aligned}\mathbf{v}(t) &= \mathbf{v}(t) - \mathbf{w}(t) - \frac{1}{3}\mathbf{v}(t)^3 + 0.5, \\ \mathbf{w}(t) &= 0.040\mathbf{v}(t) - 0.028\mathbf{w}(t) + 0.032.\end{aligned} \tag{2}$$

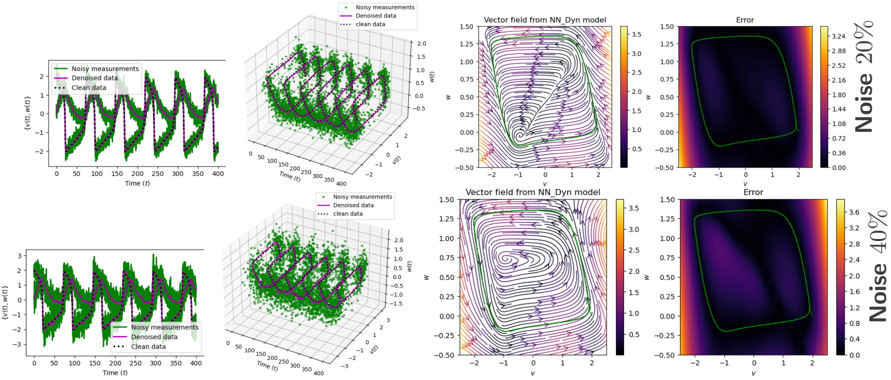
- We collect data for the initial condition $[2, 0]$ and corrupt it by adding Gaussian white noise of different levels.



Noisy measurements
Denoised data
Clean data
clean data
Time (t)
Vector field from NN_Dyn model
Error
Noise 5%
Noise 10%

Numerical Experiments

Fitz-Hugh Nagumo Models



An Extension to PDE Data

Keys points in extending the methodolgy to PDEs

- The black-box methodology to learn dynamical models can be extended to PDE data.

Keys points in extending the methodolgy to PDEs

- The black-box methodology to learn dynamical models can be extended to PDE data.
- In this case, an implicit network takes spatial coordinates as inputs, too.

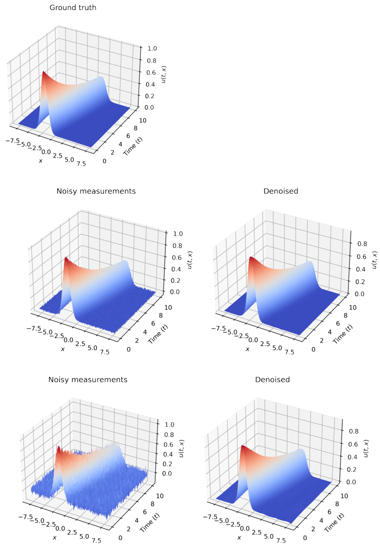
An Extension to PDE Data

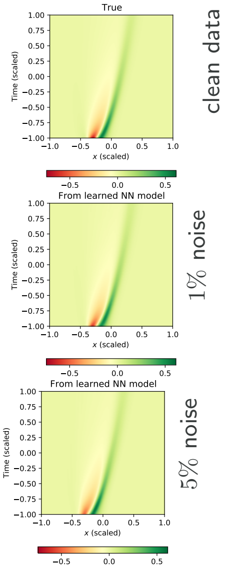
Keys points in extending the methodolgy to PDEs

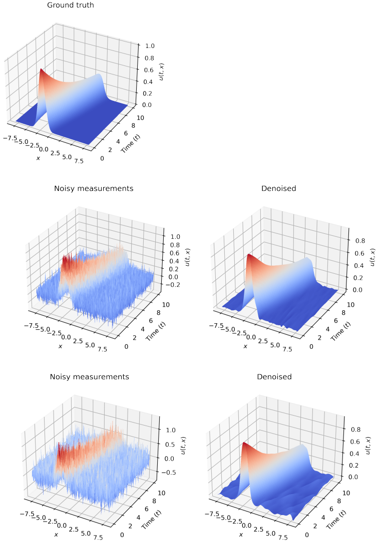
- The black-box methodology to learn dynamical models can be extended to PDE data.
- In this case, an implicit network takes spatial coordinates as inputs, too.
- The neural network defining the vector field consists of convolutional neural networks to make use of spatial information.

An Extension to PDE data

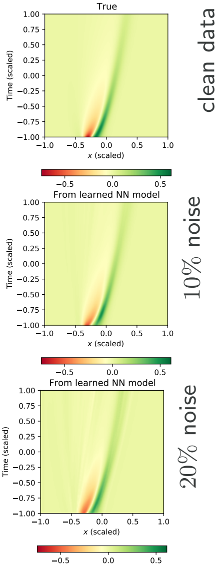
Burgers' equation







Ground truth
Noisy measurements
Denoised
Noisy measurements
Denoised
x
Time (t)
$u(t,x)$



True
From learned NN model
From learned NN model
Time (scaled)
x (scaled)
clean data
10% noise
20% noise

Outlook

Summary

- We have blended a Runge-Kutta scheme with sparse regression to discover governing equations $\rightsquigarrow$ no derivative estimate required.
  - Models are interpretable, parsimonious, and generalizable outside training regime.
- Discussed extensions to discover parametric and rational nonlinear models.
- Proposed neural networks-based approach to denoise measurements, and simultaneously learn dynamical models:
  - We utilized implicit networks blended with a Runge-Kutta scheme.
  - One can use the obtained de-noised measurements in other applications, e.g., in RK4-SINDy for dictionary based discover of analytic equations.

Outlook

Summary

- We have blended a Runge-Kutta scheme with sparse regression to discover governing equations $\rightsquigarrow$ no derivative estimate required.
  - Models are interpretable, parsimonious, and generalizable outside training regime.
- Discussed extensions to discover parametric and rational nonlinear models.
- Proposed neural networks-based approach to denoise measurements, and simultaneously learn dynamical models:
  - We utilized implicit networks blended with a Runge-Kutta scheme.
  - One can use the obtained de-noised measurements in other applications, e.g., in RK4-SINDy for dictionary based discover of analytic equations.

Next steps

- Neural networks-based approach is purely black-box $\rightsquigarrow$ hard to interpret and generalize.
  - Investigating how to fuse physics or prior to improve the performance as well as to obtain interpretable and generalizable models
- It is known that high-dimensional dynamical models (PDE solutions) often evolve in a low-dimensional manifold.
  - How to make use of this information in learning low-dimensional models from noisy PDEs data?

Outlook

Summary

- We have blended a Runge-Kutta scheme with sparse regression to discover governing equations $\rightsquigarrow$ no derivative estimate required.
  - Models are interpretable, parsimonious, and generalizable outside training regime.
- Discussed extensions to discover parametric and rational nonlinear models.
- Proposed neural networks-based approach to denoise measurements, and simultaneously learn dynamical models:
  - We utilized implicit networks blended with a Runge-Kutta scheme.
  - One can use the obtained de-noised measurements in other applications, e.g., in RK4-SINDy for dictionary based discover of analytic equations.

Next steps

- Neural networks-based approach is purely black-box $\rightsquigarrow$ hard to interpret and generalize.
  - Investigating how to fuse physics or prior to improve the performance as well as to obtain interpretable and generalizable models
- It is known that high-dimensional dynamical models (PDE solutions) often evolve in a low-dimensional manifold.
  - How to make use of this information in learning low-dimensional models from noisy PDEs data?

Thank you for your attention!!

Selected References

Bongard, J. and Lipson, H. (2007).
Automated reverse engineering of nonlinear dynamical systems.
Proc. Nat. Acad. Sci. U.S.A., 104(24):9943–9948.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016).
Discovering governing equations from data by sparse identification of nonlinear dynamical systems.
Proc. Nat. Acad. Sci. U.S.A., 113(15):3932–3937.

Daniels, B. C. and Nemenman, I. (2015).
Automated adaptive inference of phenomenological dynamical models.
Nature Comm., 6(1):1–8.

Goyal, P. and Benner, P. (2021).
Discovery of nonlinear dynamical systems using a Runge-Kutta inspired dictionary-based sparse regression approach.
e-print 2105.04869, arXiv.
cs.LG.

Mangan, N. M., Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016).
Inferring biological networks by sparse identification of nonlinear dynamics.
IEEE Trans. Molecular, Biological and Multi-Scale Comm., 2(1):52–63.

Rico-Martinez, R. and Kevrekidis, I. G. (1993).
Continuous time modeling of nonlinear systems: A neural network-based approach.
In *IEEE Int. Conf. on Neural Networks*, pages 1522–1525.

Rudy, S. H., Kutz, J. N., and Brunton, S. L. (2019).
Deep learning of dynamics and signal-noise decomposition with time-stepping constraints.
J. Comput. Phys., 396:483–506.