Numerical multiscale analysis of complex dynamical systems with machine learning

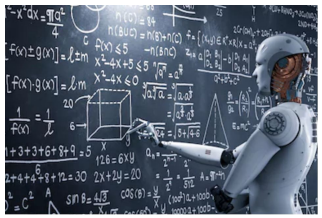
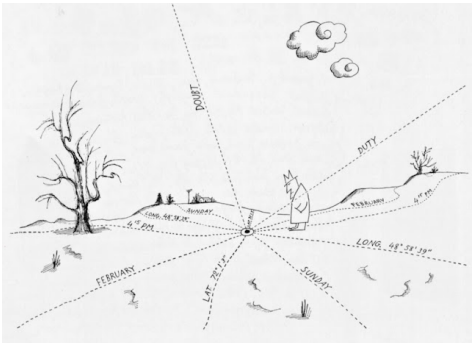
Use case per il WP1 del CN-HPC del PNRR, 6 March 2023

Constantinos Siettos

Dept. of Mathematics and Applications &
School of Advanced Studies (Scuola Superiore Meridionale),
University of Naples Federico II

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Open challenging Tasks

- (A) Can ML beat traditional numerical analysis methods for the solution of stiff ODEs and PDEs?, 
- (B) Deal with the so-called "curse of dimensionality" when trying to efficiently learn ML models with good generalization properties, and 

- (C) Discover from data the appropriate macroscopic quantities/physics for the emergent dynamics, 

- (D) Bridge Machine Learning with Physics-based Modelling, Discover Physical Laws from Data

Outline

- Can ML beat traditional numerical analysis methods for the solution of stiff large-scale ODEs and PDEs?
  - Numerical Solution of the Forward problem: the classical way
  - Solving the Forward (and INVERSE) Problem with Machine Learning: Curse of Dimensionality
- Dealing with the Curse of Dimensionality
  - Proposed Physics Informed Random Projection Neural Network framework
  - Benchmark Problems: The Forward Problem
- Discovery of Physics: The Inverse Problem: Constructing and Analysing PDEs from BIG Data/ MANIFOLD LEARNING

The Forward and Inverse Problem in Complex Systems Modelling

IT's ALL ABOUT DISCOVERING AND SOLVING DIFFERENTIAL EQUATIONS IN A CLOSED FORM

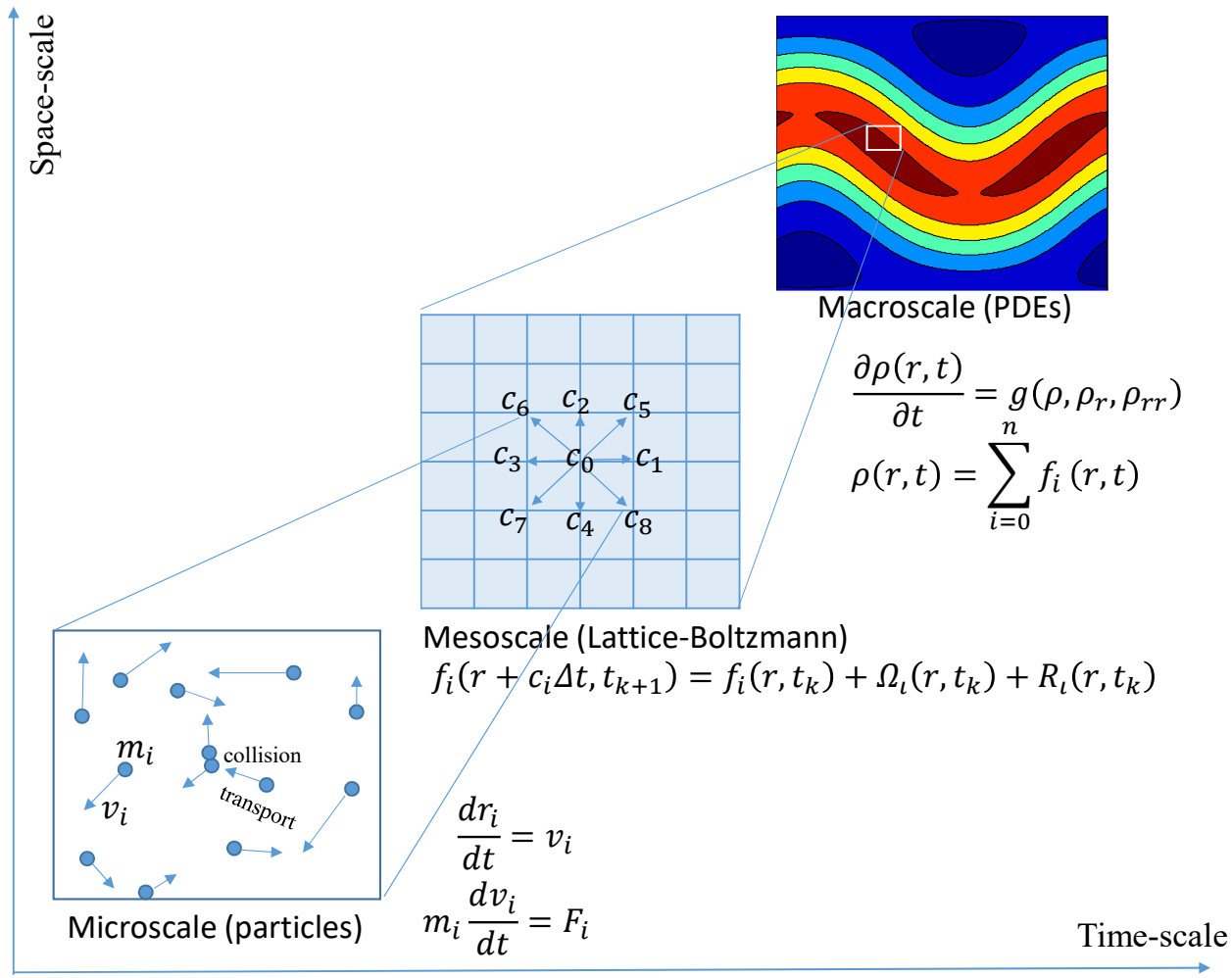
- Forward Problem: Numerical Solution of Large-Scale Differential Equations

-

Inverse Problem: Modelling and forecasting the emergent dynamics of multiphysics and multiscale systems from DATA

- Both remain open problems!

The Forward and Inverse Problem in Complex Systems Modelling



Numerical Solution of the Forward and Inverse Problem for Differential Equations

with Machine Learning: The "classical Machine Learning way"

Let's assume a set of m points $\boldsymbol{x}_i \in \Omega \subset \mathbb{R}^d$ of the independent (spatial) variables, defining the grid in the domain Ω, n_Ω points along the boundary of the domain, $\partial\Omega$ and n_t points in the time interval. Then the "classical way" to solve (time-dependent) differential equations in the general form

$$\frac{\partial u}{\partial t} = L(\boldsymbol{x}, u, \nabla u, \nabla^2 u, \dots), \tag{1}$$

where u satisfies the boundary conditions $Bu = g,$ in $\partial\Omega$ (B is the boundary differential operator) involves the solution of a minimization problem of the form:

Numerical Solution of Differential Equations

with Neural Networks: The "classical Machine Learning way"

$$\min_{\boldsymbol{P},\boldsymbol{Q}} E(\boldsymbol{P},\boldsymbol{Q}) := \sum_{i=1}^{m}\sum_{j=1}^{n_t}\left\|\frac{d\Psi}{dt}(.) - L(\boldsymbol{x}_i,\Psi(.),\nabla\Psi(.),\nabla^2\Psi(.),\dots)\right\|^2 + \tag{2}$$

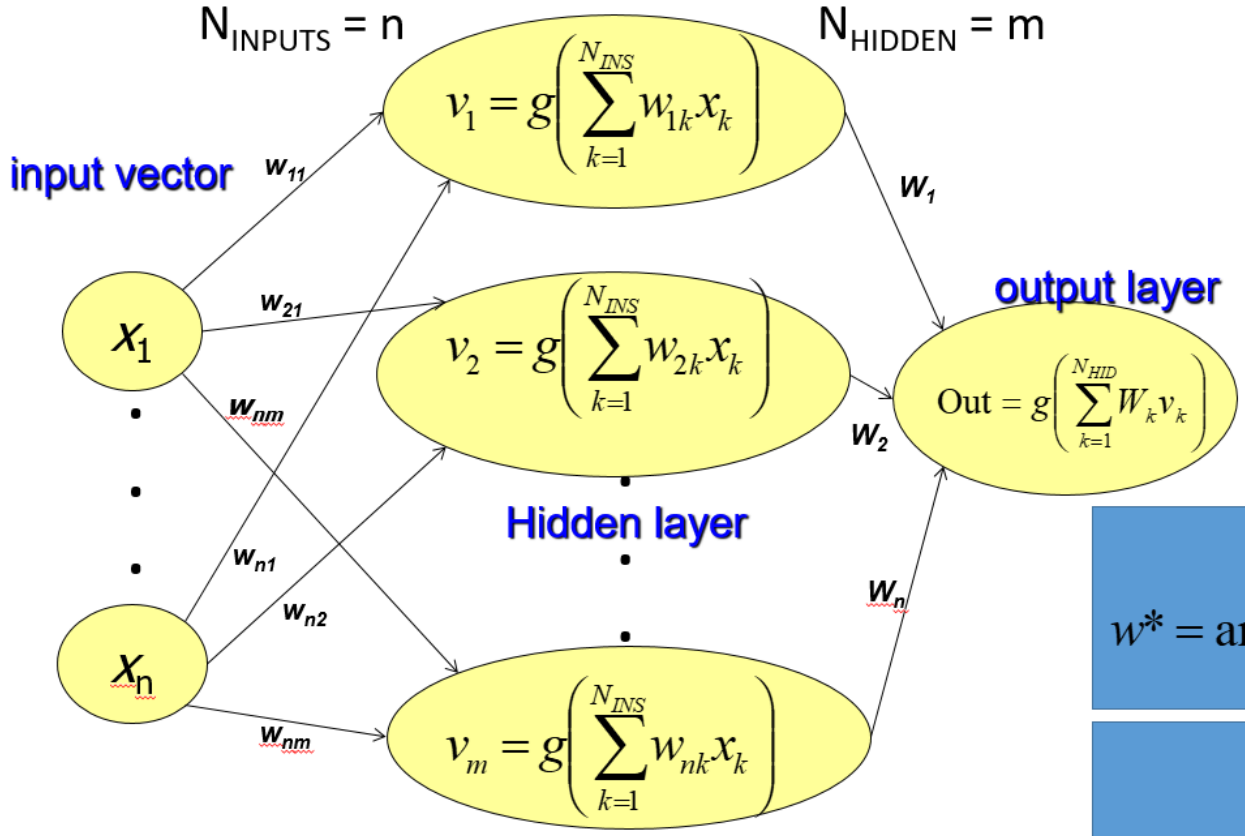
$$\sum_{j=1}^{n_\Omega}\|B\Psi(.) - g\|^2,$$

where $\Psi(.) := \Psi(t_j,\boldsymbol{x}_i,N(t_j,\boldsymbol{x}_i,\boldsymbol{P},\boldsymbol{Q}))$ represents a "trial" function approximating the solution u at $\boldsymbol{x}_i$ and $N(t_j,\boldsymbol{x}_i,\boldsymbol{P},\boldsymbol{Q})$ is a machine learning algorithm; $\boldsymbol{P}$ contains the parameters of the machine learning scheme (e.g. for a FNN the internal weights $\boldsymbol{W}$, the biases $\boldsymbol{B}$, the weights between the last hidden and the output layer $\boldsymbol{W}^o$), $\boldsymbol{Q}$ contains the hyperparameters.

Machine Learning

Training is computationally demanding even for the simplest structures!

Iteratively: e.g. with quasi-Newton BFGS, Back-Propagation, Adams etc.



1. Given training data:

$$\{x_i, y_i\}_{i=1}^{N}$$

2. Choose loss function

$$l = (\hat{y}, y_i) \in \Re$$

3. Define goal

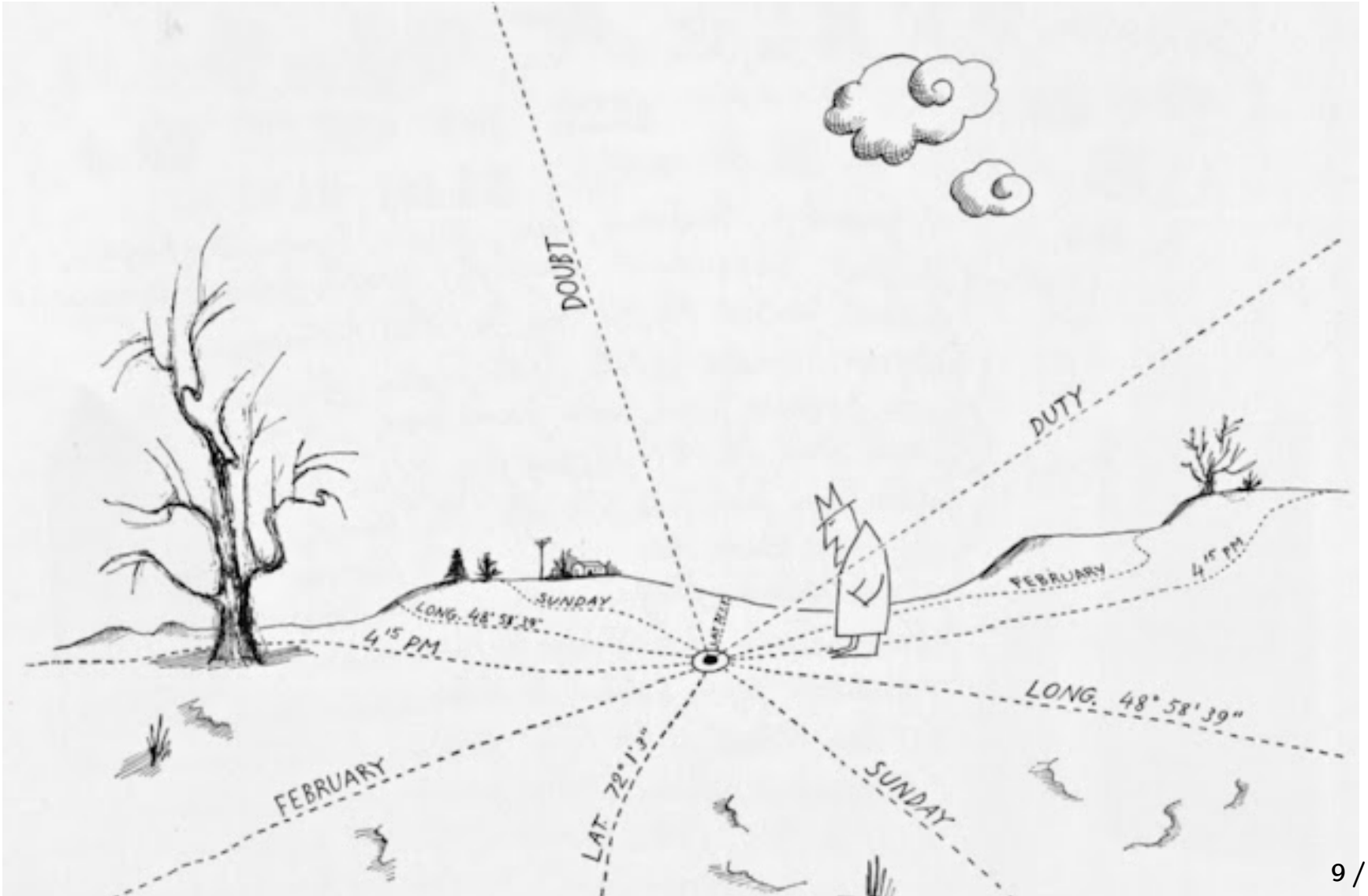
$$w^* = \arg\min_w \sum_{i=1}^{N} l\left(\hat{y}(x_i, \theta), y_i\right)$$

4. Train

$$w^{k+1} = w^k - \eta_k \nabla l$$

Curse of Dimensionality

In Life: The Curse of Dimensionality Saul Steinberg 1968



Numerical Solution of the Forward Problem

The Proposed Method: RANDOM PROJECTIONS

Taking into account that the trial solution must satisfy the initial value conditions $y_i(x_0) = \alpha_i$, $i = 1, 2, \ldots, m$, we set:

$$\Psi_i(t, \boldsymbol{w}_i^o, \boldsymbol{p}_i) = \alpha_i + (t - t_0) N_i(t, \boldsymbol{w}_i^o, \boldsymbol{p}_i), \tag{3}$$

where $N_i(t, \boldsymbol{w}_i^o, \boldsymbol{p}_i)$ is a single-output FNN with parameters the output weights $\boldsymbol{w}_i^o = [w_{1i}^o \; w_{2i}^o \; \ldots \; w_{hi}^o]^T \in \mathbb{R}^h$, and $\boldsymbol{p}_i$ contains the remaining parameters associated with that network.

- we fix all the internal weights to 1
- centers of RBFs to be equidistant in the domain
- set randomly from appropriately chosen uniform distributions the biases b_{ji} and the width parameters σ_{ji}

THE FORWARD PROBLEM: Numerical Solution of Differential Equations

The Proposed Method

We seek a numerical solution based on n collocation points $t_1, t_2, \ldots, t_n$. The objective function we seek to minimize is given by:

$$\mathcal{L}(\boldsymbol{W}^o) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(M_{ij}\frac{d\Psi_{Ni}}{dt}(t_j, \boldsymbol{w}_i, \boldsymbol{w}_i^o, \boldsymbol{p}_i) - \right. \tag{4}$$

$$\left. - f_i(t_j, \boldsymbol{\Psi}(t_j, \boldsymbol{W}, \boldsymbol{W}^o, \boldsymbol{P})\right)^2.$$

See that the proposed method results to a solution that can be computed at any point of the domain. This is fundamentally different from the traditional numerical schemes

THE FORWARD PROBLEM: Numerical solution of Differential Equations

The Proposed Method

The only parameters that have to be determined by training the network are the output weights w_{ji}^o.
For n collocation points, the outputs of each network N_i, $i = 1, 2, \ldots m$, are given by:

$$\boldsymbol{N}_i(t_1, t_2, \ldots t_n, \boldsymbol{w}_i^o, \boldsymbol{p}_i) = R_i \boldsymbol{w}_i^o, \tag{5}$$

where $\boldsymbol{N}_i(t_1, t_2, \ldots t_n, \boldsymbol{w}_i^o, \boldsymbol{p}_i) \in \mathbb{R}^n$, and
$R_i = R_i(t_1, \ldots, t_n, \boldsymbol{p}_i) \in \mathbb{R}^{n \times h}$ is defined as

$$R_i(t_1, \ldots, t_n, \boldsymbol{p}_i) = \begin{bmatrix} G_{1i}(t_1) & \cdots & G_{hi}(t_1) \\ \vdots & \vdots & \vdots \\ G_{1i}(t_n) & \cdots & G_{hi}(t_n) \end{bmatrix}. \tag{6}$$

Thus, by setting $\boldsymbol{F}(\boldsymbol{W}^o) = [F_1(\boldsymbol{W}^o) \cdots F_q(\boldsymbol{W}^o) \cdots F_{(nm)}(\boldsymbol{W}^o)]^T$, the new update $d\boldsymbol{W}^{o(\nu)}$ at the (ν)-th iteration is computed by the solution of the system

$$\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F}\, d\boldsymbol{W}^{o(\nu)} = -\boldsymbol{F}(\boldsymbol{W}^{o(\nu)}), \tag{7}$$

where $\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F} \in \mathbb{R}^{nm \times mh}$ is the Jacobian matrix of $\boldsymbol{F}$.
In general $h > n$, i.e. the minimization problem in (7) is an under-determined linear system where the Jacobian is not full row-rank.

THE FORWARD PROBLEM BENCHMARK ODEs/PDEs: Numerical Results

- Four benchmark stiff ODE/PDE problems (with steep gradients),
  - van der Pol model,
  - Robertson index-1 DAEs
  - 1D Nonlinear Viscous Burgers PDE,
  - 1D, 2D, Liouville–Bratu–Gelfand PDE
- We test the performance of the scheme, in terms of
  - approximation accuracy
  - computational times
  - compare with the matlab ode suite `ode23t` and `ode15s`, implementing a variable-step variable-order multistep method, suitable for low-dimensional stiff and index-1 DAEs problems

$$\nu \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x} = 0 \tag{8}$$

in the unit interval $[0, 1]$. For our analysis, we considered two different sets of boundary conditions:

- Dirichlet boundary conditions

$$u(0) = \gamma\,, \;\; u(1) = 0\,, \;\; \gamma > 0\;; \tag{9}$$

- Mixed boundary conditions: Neumann condition on the left boundary and zero Dirichlet on the right boundary:

$$\frac{\partial u}{\partial x}(0) = -\vartheta\,, \;\; u(1) = 0\,, \;\; \vartheta > 0\;. \tag{10}$$

The Nonlinear Viscous Burgers

Dirichlet BC

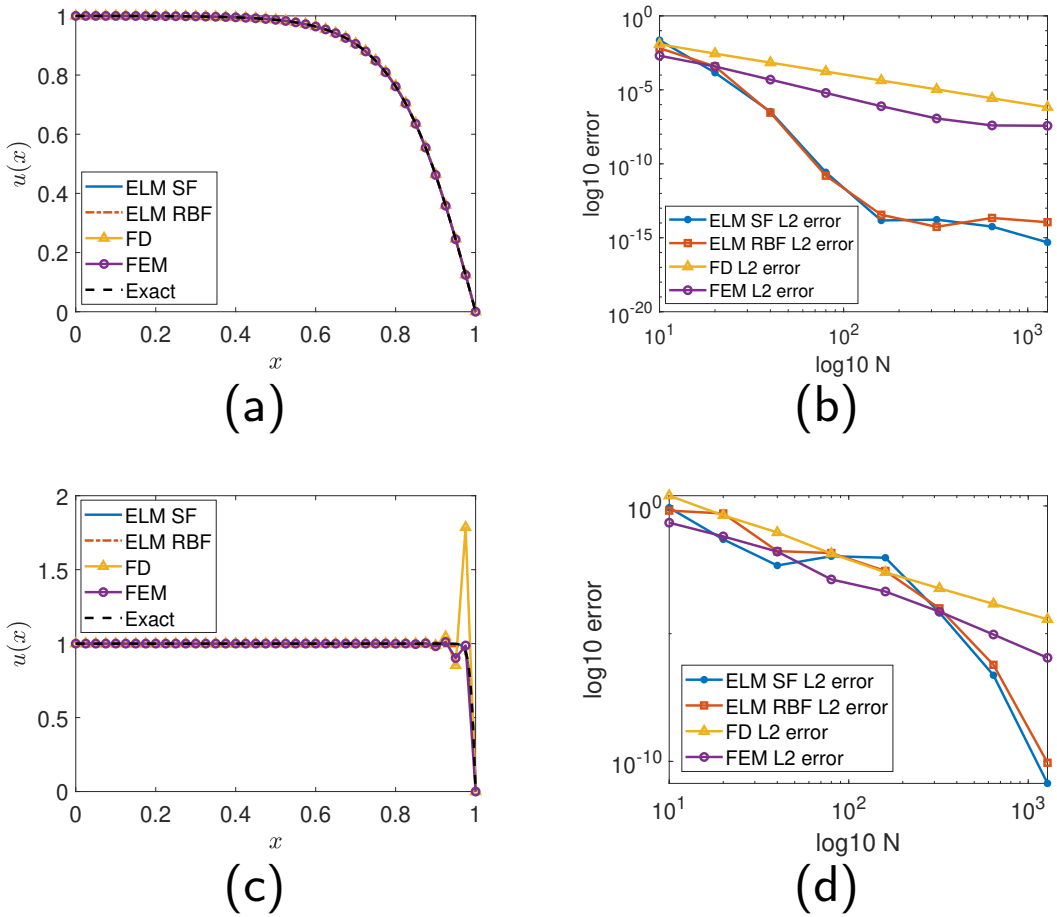


Figure: (a,b) viscosity $\nu = 0.1$: (a) Solutions for $N = 40$; (b) L_2–norm vs exact solution (c,d) viscosity $\nu = 0.007$: (c) Solutions for $N = 40$; (d) L_2–norm vs. exact solution.

The Nonlinear Viscous Burgers

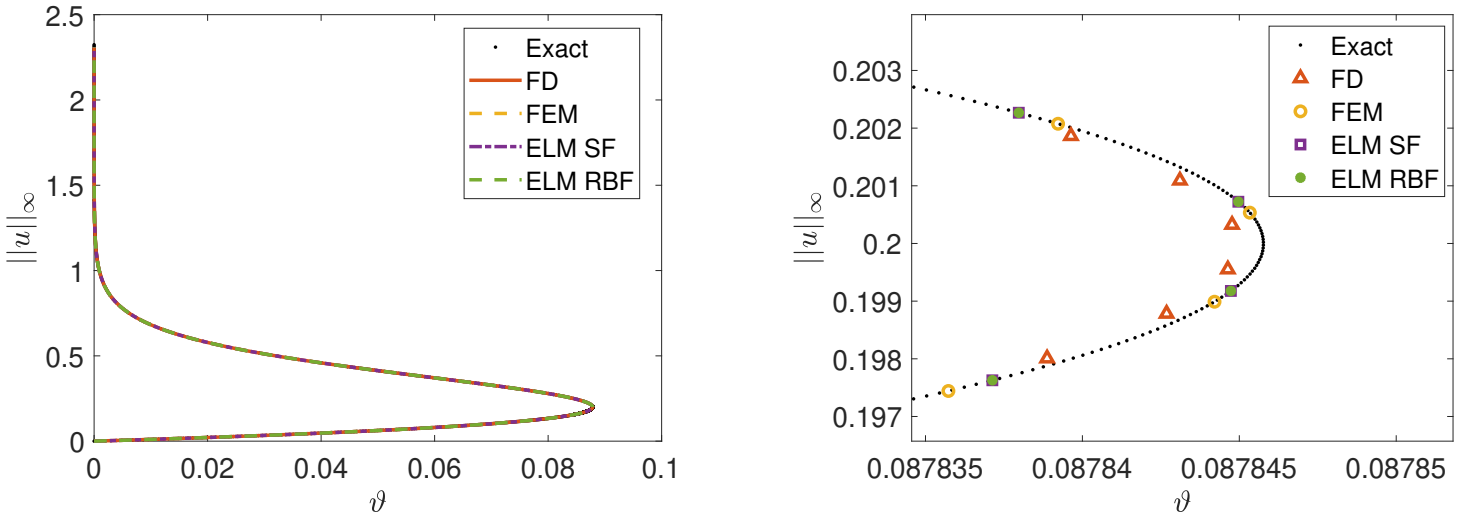
Execution Times

	ELM SF			ELM RBF		
N	5%	mean	95%	5%	mean	95%
80	2.73e-03	4.70e-03	4.38e-03	2.31e-03	2.67e-03	3.16e-03
160	8.46e-03	9.97e-03	1.12e-02	7.16e-03	8.20e-03	9.08e-03
320	3.72e-02	4.23e-02	4.60e-02	3.52e-02	3.89e-02	4.28e-02
640	1.60e-01	1.67e-01	1.75e-01	1.56e-01	1.69e-01	1.97e-01
	FD			FEM		
N	5%	mean	95%	5%	mean	95%
80	1.72e-04	3.37e-04	3.48e-04	2.33e-02	2.49e-02	2.76e-02
160	4.31e-04	4.55e-04	5.26e-04	5.68e-02	6.39e-02	6.95e-02
320	1.29e-03	1.33e-03	1.44e-03	1.22e-01	1.24e-01	1.31e-01
640	1.05e-02	1.10e-02	1.16e-02	3.34e-01	3.40e-01	3.55e-01

Table: Execution times (s) for the Burgers equation (8) with Dirichlet boundary conditions (9) and $\nu = 0.1$.

The Nonlinear Viscous Burgers

The Bifurcation Diagram with Mixed BC



(a) (b)

Figure: (a) One-dimensional Burgers equation (8) with mixed boundary conditions. Bifurcation diagram with respect to the Neumann boundary value θ as obtained for $\nu = 1/10$, with FD, FEM and ELM schemes with a fixed problem size $N = 400$; (b) Zoom near the turning point.

THE INVERSE PROBLEM: CONSTRUCTING PDES FROM BIG DATA

The assumption here is that the emergent dynamics of the complex system under study on a domain $\Omega \times [t_{\mathbf{0}}, t_{end}] \subseteq \mathbb{R}^d \times \mathbb{R}$ can be modelled by a system, of say m PDEs in the form of:

$$\frac{\partial u^{(i)}(x,t)}{\partial t} \equiv u_t^{(i)} = F^{(i)}(t, x, u(x,t), \mathcal{D}u(x,t), \mathcal{D}^{\mathbf{2}}u(x,t), \ldots, \mathcal{D}^{\nu}u(x,t), \varepsilon), \qquad (x,t) \in \Omega \times [t_{\mathbf{0}}, t_{end}], \qquad i = 1, 2, \ldots, m \tag{11}$$

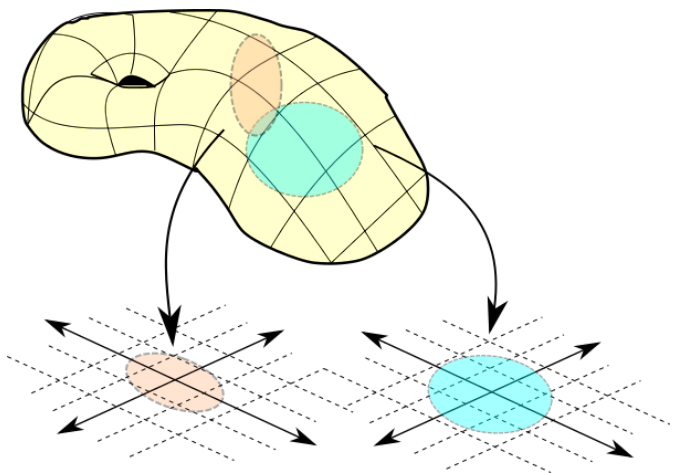
where $u(x,t) = [u^{(\mathbf{1})}(x,t), \ldots, u^{(m)}(x,t)]$, $F^{(i)}$, $i = 1, 2, \ldots m$ is a non-linear operator, $\mathcal{D}^{\nu}u(x,t)$ is the generic multi-index ν-th order spatial derivative at time t i.e.:

$$\mathcal{D}^{\nu}u(x,t) := \left\{ \frac{\partial^{|\nu|}u(x,t)}{\partial x_{\mathbf{1}}^{\nu_{\mathbf{1}}} \ldots \partial x_d^{\nu_d}}, |\nu| = \nu_{\mathbf{1}} + \nu_{\mathbf{2}} + \cdots + \nu_d, \nu_{\mathbf{1}}, \ldots, \nu_d \geq 0 \right\},$$

and ε denotes the (bifurcation) parameters of the system.

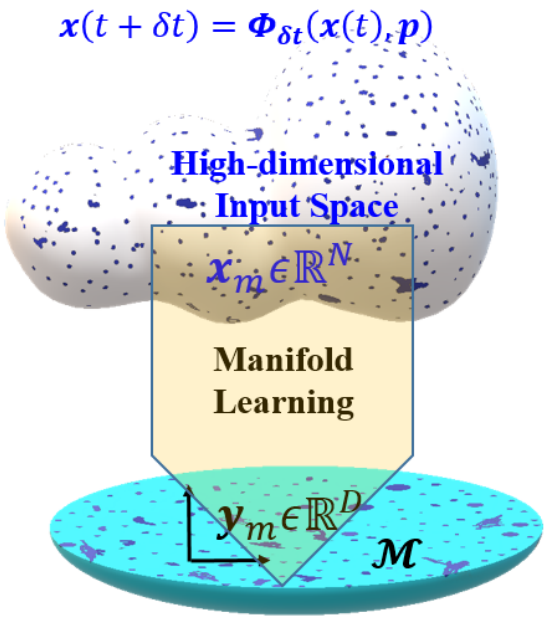
THE INVERSE PROBLEM: DISCOVERING THE LAWS OF PHYSICS FROM DATA -> MANIFOLD LEARNING

A manifold is a topological space that locally resembles Euclidean space near each point. More precisely, an n-dimensional manifold is a topological space with the property that each point has a neighborhood that is homeomorphic to an open subset of n-dimensional Euclidean space.



MANIFOLD LEARNING FIND THE INTRINSIC DIMENSION AND TOPOLOGY OF THE EMERGENT DYNAMICS

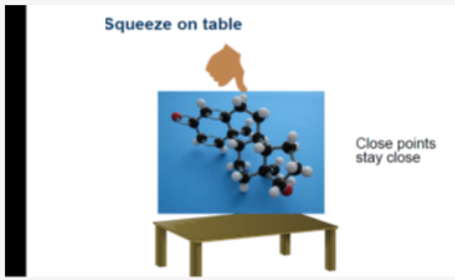
... AND A SET OF MACROSCOPIC VARIABLES THAT PARAMETRIZE IT.



MANIFOLD LEARNING: METHODS

Linear

SVD, PCA,

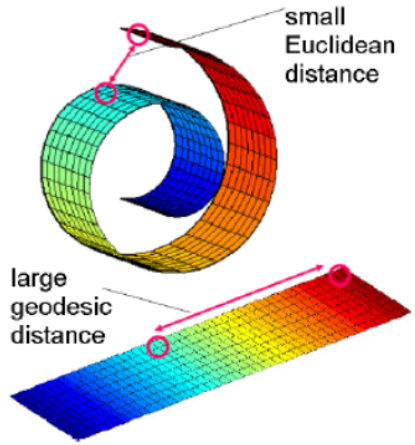


Multidimensional Scaling (MDS)

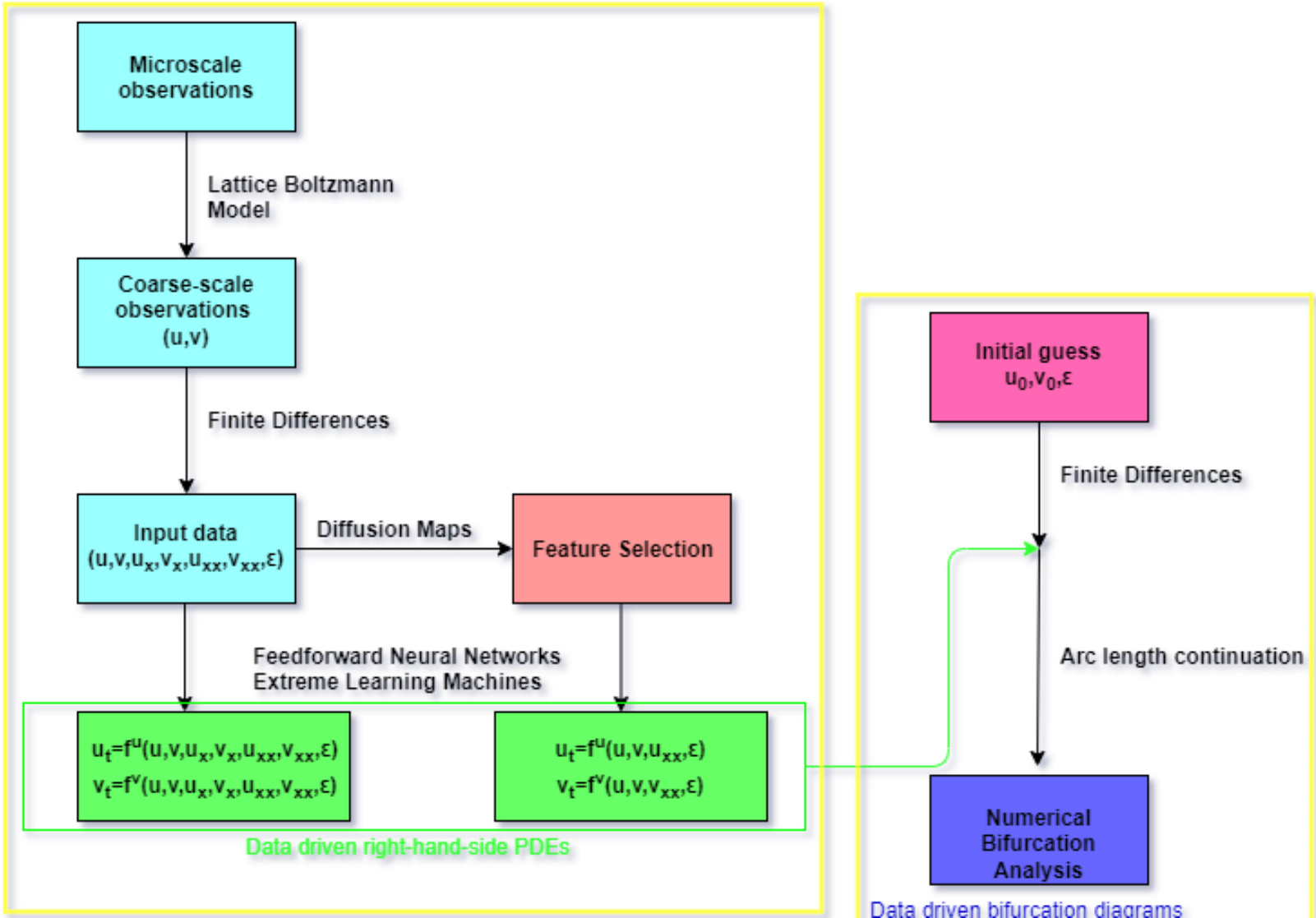
Nonlinear

Kernel PCA, Local Linear Embedding,

Isomap (Isoparametric mapping)*, Diffusion Maps



THE INVERSE PROBLEM: CONSTRUCTING PDEs FROM DATA: THE SCHEMATIC



THE INVERSE PROBLEM: IDENTIFYING THE FITZHUGH-NAGUMO PDES FROM DATA PRODUCED BY LATTICE BOLTZMANN SIMULATIONS

The evolution of activation $u : [x_0, x_{end}] \times [t_0, t_{end}] \to \mathbb{R}$ and inhibition $v : [x_0, x_{end}] \times [t_0, t_{end}] \to \mathbb{R}$ dynamics are described by the following two coupled nonlinear parabolic PDEs:

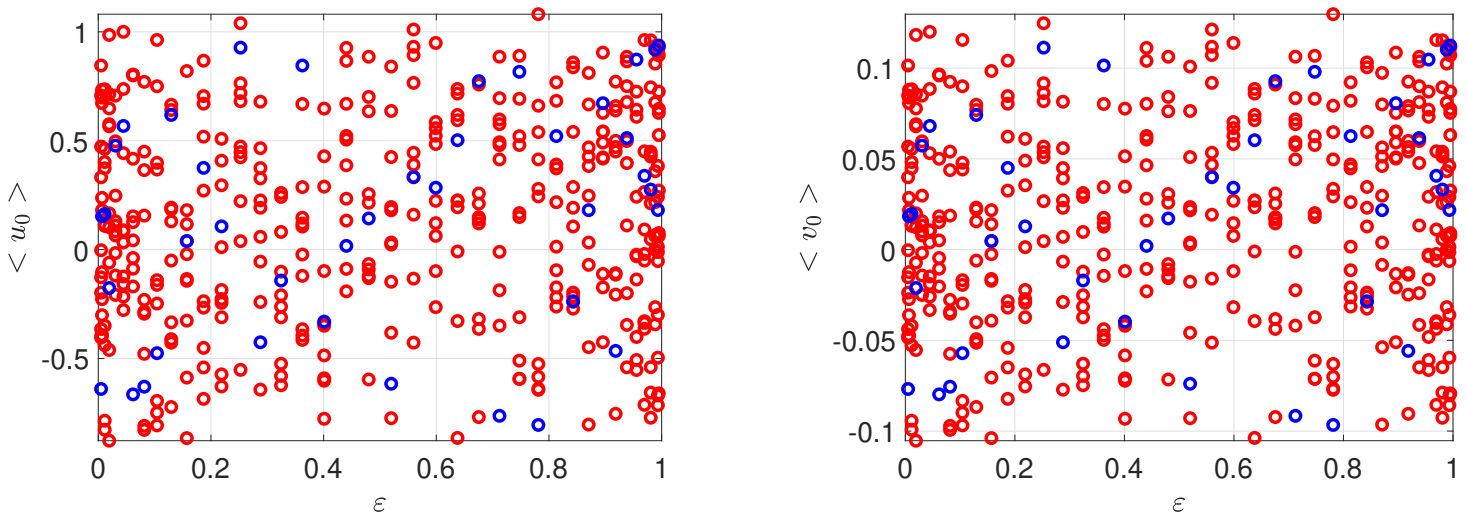
$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= D^u \frac{\partial^2 u(x,t)}{\partial x^2} + u(x,t) - u(x,t)^3 - v(x,t), \\ \frac{\partial v(x,t)}{\partial t} &= D^v \frac{\partial^2 v(x,t)}{\partial x^2} + \varepsilon(u(x,t) - \alpha_1 v(x,t) - \alpha_0), \end{aligned} \tag{12}$$

with homogeneous von Neumann Boundary conditions:

$$\begin{aligned} \frac{du(x_{end},t)}{dx} &= 0, \quad \frac{dv(x_0,t)}{dx} = 0. \\ \frac{du(x_{end},t)}{dx} &= 0, \quad \frac{dv(x_0,t)}{dx} = 0. \end{aligned} \tag{13}$$

THE INVERSE PROBLEM: THE DATA GRID

We end up with a dataset consisting of 40 (values of ε)$\times$10 (initial conditions)$\times$448 (time points ignoring the first 2s of the transient)$\times$40 (space points) $\backsimeq$ 7.168.000 data points.



(a) (b)

Figure: Coarse initial conditions for (a) u and (b) v for the training. The grid is spanned with Chebychev-Gauss-Lobatto points for epsilons in the interval $[0.005, 0.995]$

THE INVERSE PROBLEM: IDENTIFYING THE SET OF VARIABLES WITH DIFFUSION MAPS

	$u_t = (\phi_1^u, \phi_2^u, \phi_3^u)$		$v_t = (\phi_1^v, \phi_2^v, \phi_3^v)$	
	Features	**Total Loss**	**Features**	**Total Loss**
1d	(u)	4.3E-03	(u)	7.6E-03
2d	(u, v)	6.37E-06	(u, v)	1.91E-05
3d	(u, v, u_{xx})	2.77E-07	(u, v, v_{xx})	6.29E-07
4d	(u, v, u_x, u_{xx})	1.03E-07	(u, v, v_x, v_{xx})	1.34E-07

Table: The "best" set of variables that can effectively parametrize the intrinsic coordinates ($(\phi_1^u, \phi_2^2, \phi_3^u)$ and $(\phi_1^v, \phi_2^v, \phi_3^v)$) and the corresponding sums of total loses across all the values of the bifurcation parameter ε.

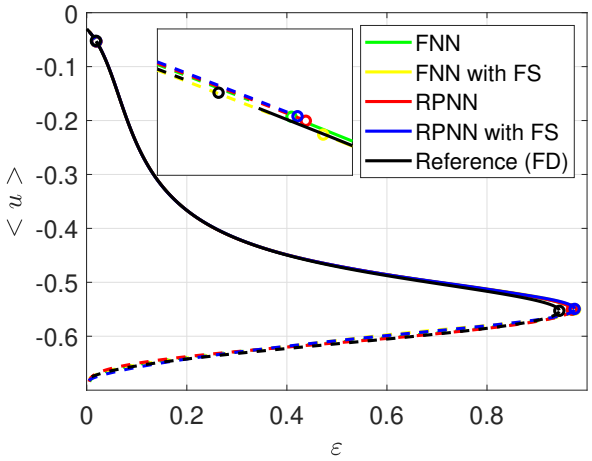
THE INVERSE PROBLEM: The reconstructed PDEs

Hence, the proposed feature selection approach based on parsimonious Diffusion Maps, revealed correctly the structure of the embedded PDEs in the form of:

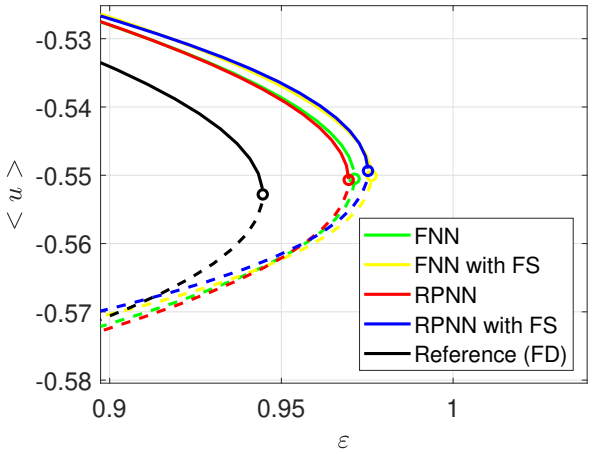
$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \hat{F}^u(u(x,t), v(x,t), u_{xx}(x,t), \varepsilon), \\ \frac{\partial v(x,t)}{\partial t} &= \hat{F}^v(u(x,t), v(x,t), v_{xx}(x,t), \varepsilon) \end{aligned} \tag{14}$$

where $\hat{F}^u$ and $\hat{F}^v$ are the outputs of the FNNs (or the RPNNs).

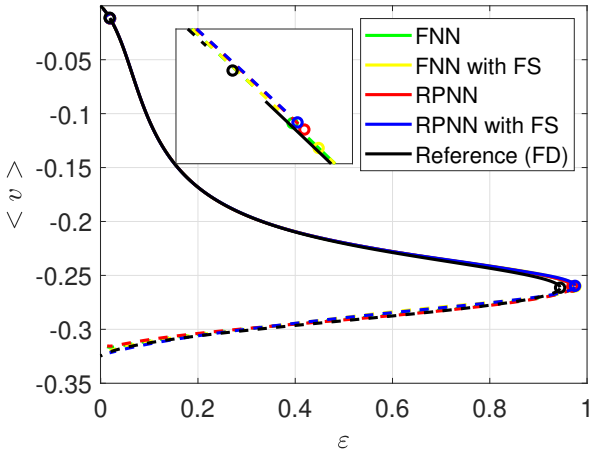
THE INVERSE PROBLEM: The reconstructed Bifurcation Diagram of the FHN dynamics



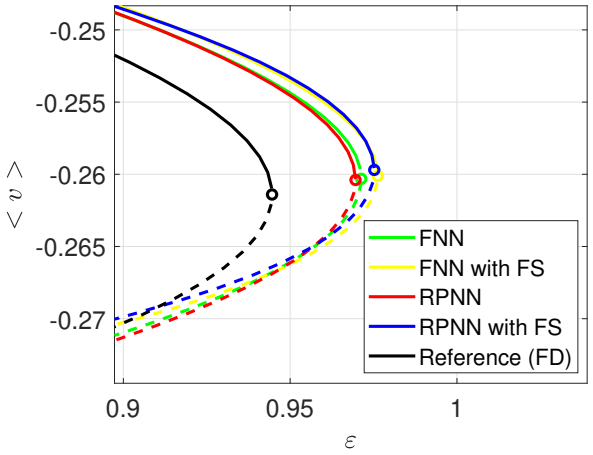
(a)



(b)



(c)



(d)

THE INVERSE PROBLEM: Computational time for training

The training phase of the FNN required 800 epochs and around 4 hours with minimum tolerance set to $1e-07$.

-

Instead, the training phase of the RPNNs for $\hat{u}_t$ and $\hat{v}_t$ required around 8 minutes each in matlab R2020b using a single core of an Intel i7-10750H CPU @ 2.60GHz and 16GB RAM, thus resulting to a training phase at least 20 times faster.

Conclusions

- The high-dimensionality that intrinsically characterizes the state-space of relevant multiscale/complex problems, compounded by the inherent modelling uncertainties across scales, severely challenge our ability to efficiently understand, learn, analyze and control their collective behavior.
- We proposed a new numerical framework based on Machine Learning for the numerical solution of both the forward and inverse problems in complex systems modelling based on Random Projection Networks that deals with the Curse of Dimensionality and by coupling the Equation-Free framework with Manifold Learning and Control

Requests: Access to 2 PHD students and 1 Post-doc researcher to a HPC cluster for running large-scale agent-based simulations and for learning physical-laws from BIG DATA with machine-learning

References

Coarse-scale PDEs from Fine-scale Observations via Machine Learning, 2020
Lee, S., Kooshkbaghi, M., Spiliotis, K., Siettos, C.I., Kevrekidis, I.G.,
Chaos, 30, 013141 https://doi.org/10.1063/1.5126869.

Numerical Solution and Bifurcation Analysis of Nonlinear Partial Differential Equations with Extreme Learning Machines, 2021
Fabiani, G., Calabrò, F. and Russo, L. and Siettos, C.
Journal Scientific Computing 89, 44 https://doi.org/10.1007/s10915-021-01650-5

Numerical Bifurcation Analysis of PDEs From Lattice Boltzmann Model Simulations: a Parsimonious Machine Learning Approach, 2022
Galaris, E., Fabiani, G., Gallos, I., Kevrekidis, I. and Siettos C.
Journal Scientific Computing 92, 34 https://doi.org/10.1007/s10915-022-01883-y

Parsimonious Physics-Informed Random Projection Neural Networks for Initial-Value Problems of ODEs and index-1 DAEs, 2022
Fabiani, G., Galaris, E., Russo, L. and Siettos C.
https://arxiv.org/abs/2203.05337

Time-series Forecasting using Manifold learning, Radial Basis Function Interpolation and Geometric Harmonics, 2022
Papaioannou P., Talmon, R., Kevrekidis, I., and Siettos C.
Chaos 32, 083113, https://doi.org/10.1063/5.0094887

Data-driven control of agent-based models: An Equation/Variable-free machine learning approach, 2023
Patsatzis, D., Russo, L., Kevrekidis, I., and Siettos C.
Journal of Computational Physics 4788, 111953,
https://www.sciencedirect.com/science/article/pii/S0021999123000487