Inferring Force Fields with Different Methods

Stochastic Differential Equations

Meza Stefano, Pasian Francisco, Altamirano David, Zhang Qiqi

June 29, 2023

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Outline

1 Motivation

2 Introduction

3 Method 1: Physics and Information Theory

4 Method 2: Bayesian Statistics

5 Method 3: Deep Neural Network

6 Discussion

7 Conclusion

Motivation

- Nature, industry, and human society are teeming with phenomena, processes, and systems that evolve with the effects of random noise. Examples of such systems include the motion of Brownian particles in a fluid, the evolution of tumors driven by tumor-immune interactions, the price of stocks, and the movement of winds.

- Stochastic differential equations (SDEs) are a powerful mathematical approach for modeling and analyzing systems affected by random noise.

- Measuring microscopic force fields (e.g. motion of Brownian particles) is of fundamental importance to understanding microscale systems.

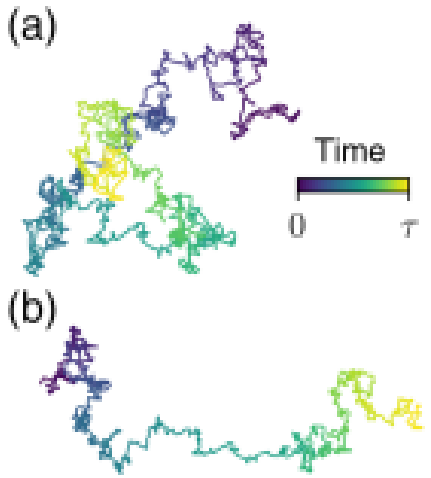
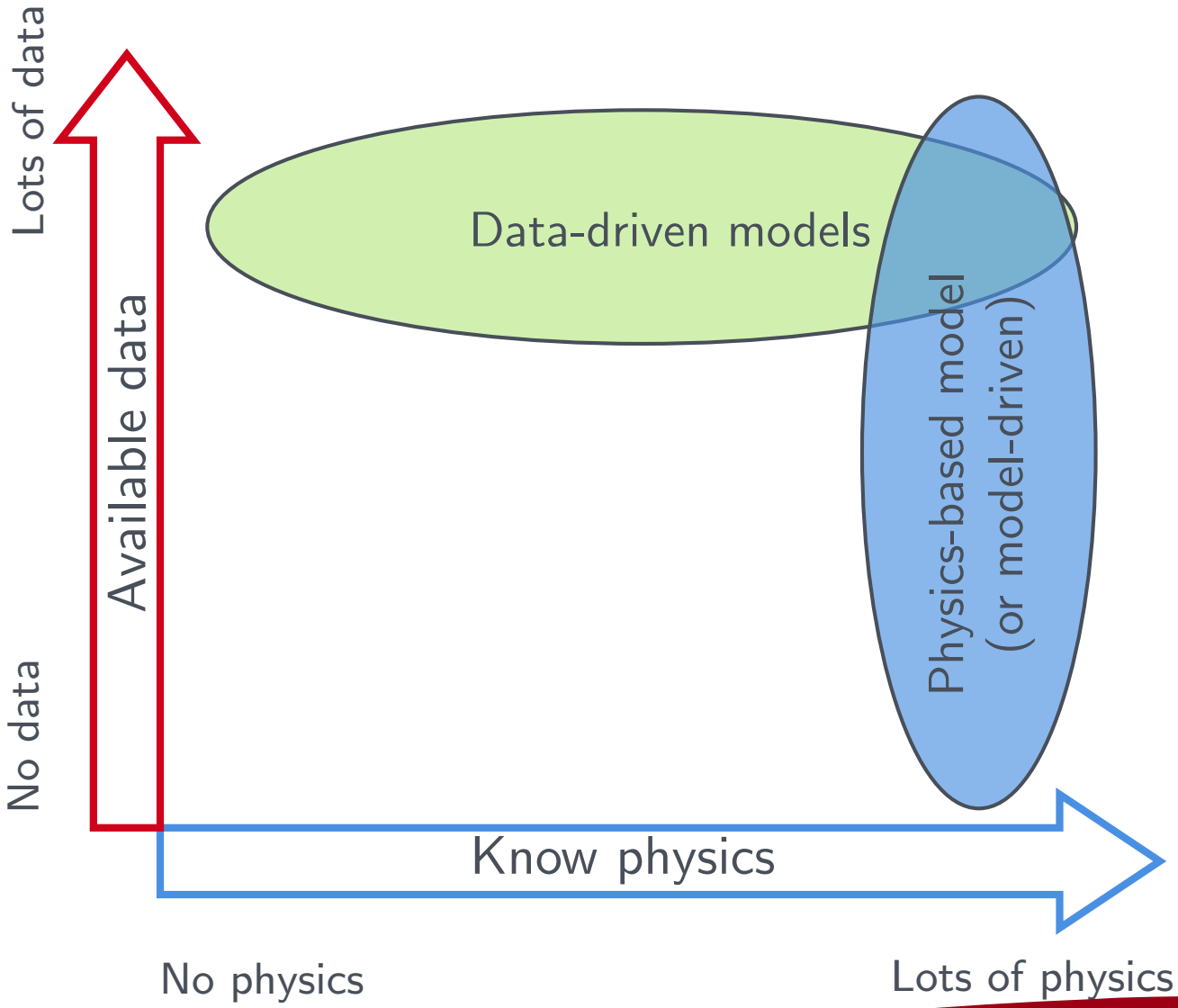


Figure: a) Absence of forces. b) Presence of a force field.

Different Approaches Comparison



Different Approaches Comparison

Our goal

Determine the force field parameters from stochastic trajectories.

We dedicate three different approaches for the force field inference:

1) Information theory to approximate force fields and spatially variable diffusion coefficients
   - Frishman, Anna and Ronceray, Pierre - *Learning Force Fields from Stochastic Trajectories*. Phys. Rev. X. 10, 2. (2020)
2) A sparse Bayesian learning (SBL) technique to develop a linear regression problem and determine SDEs.
   - Yasen Wang et al. - *Data-Driven Discovery of Stochastic Differential Equations*. Engineering. 17, 244-252. (2022)
3) A toolbox to calibrate the force fields using Machine Learning techniques, namely, Recurrent Neural Networks
   - Aykut Argun et al. - *Enhanced force-field calibration via machine learning*. Applied Physics Reviews 7, 041404 (2020)

Stochastic Differential Equations

Stochastic Differential Equations (SDEs) can be considered as ordinary differential equations (ODEs) with a stochastic term coupled with a diffusion coefficient and the Wiener process.

The n-dimensional time-independent SDEs

$$d\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t))dt + \boldsymbol{G}(\boldsymbol{x}(t))^{\frac{1}{2}} d\boldsymbol{W}(t) \tag{1}$$

$\boldsymbol{x}(\mathrm{t}) \in R^n$ is the state vector at time t
$\boldsymbol{f}(\boldsymbol{x}(\mathrm{t})) \in R^n$ is the state-dependent drift vector
$\boldsymbol{G}(\boldsymbol{x}(\mathrm{t})) \in R^{n \times n}$ is the positive-definite diffusion matrix
$\boldsymbol{W}(\mathrm{t})$ is an n-dimensional standard Brownian motion

Euler–Maruyama Discretization

Because the Wiener process is nowhere differentiable, a more formal description of the SDE is

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \boldsymbol{f}(\boldsymbol{x}(t))dt + \int_0^t \boldsymbol{G}(\boldsymbol{x}(t))^{\frac{1}{2}} d\boldsymbol{W}(t) \quad (2)$$

Generally, analytic solutions of SDEs are not available. Hence, the Euler–Maruyama method was employed to discretize the SDE to get numerical solutions

$$\hat{\boldsymbol{x}}((k+1)\Delta t) - \hat{\boldsymbol{x}}(k\Delta t) = \boldsymbol{f}(\hat{\boldsymbol{x}}(k\Delta t))\Delta t + \boldsymbol{G}(\hat{\boldsymbol{x}}(k\Delta t))^{\frac{1}{2}} \sqrt{\Delta t}\epsilon_k \quad (3)$$

Considering continuous-time approximations, for any t $\in [k\Delta t, (k+1)\Delta t)$we set

$$\hat{\boldsymbol{x}}(t) = \hat{\boldsymbol{x}}(k\Delta t) \quad (4)$$

Ito Formula

Let X_t where t $\in [0, T]$ be an Ito process with differentials

$$dX_t = H_t dB_t + K_t dt \tag{5}$$

f:R$\rightarrow R$ be a twice differentiable function.

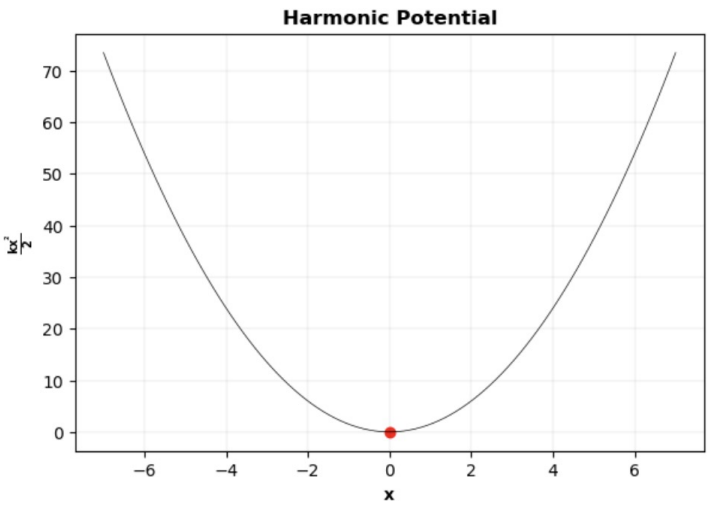
Then f(X_t) is an Ito process with

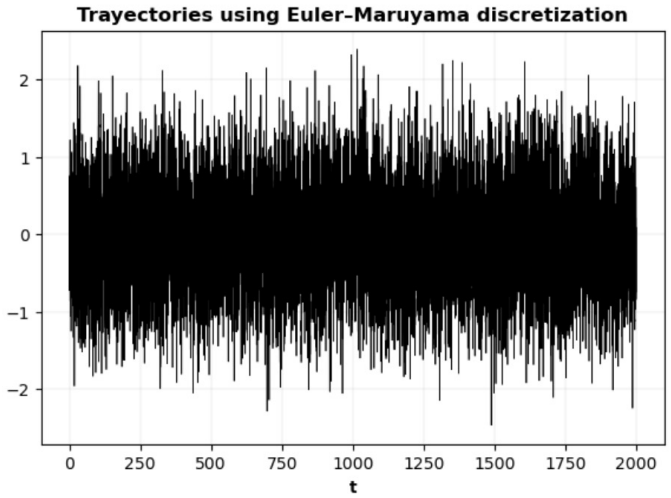
$$d(f(X_t)) = f'(X_t)dX_t + \frac{1}{2}f''(X_t)d[X]_t \tag{6}$$

i.e.

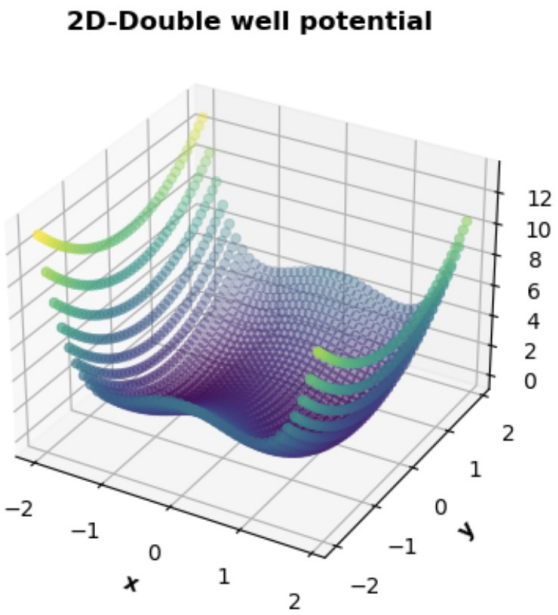
$$\begin{aligned} f(X_t) &= f(X_0) + \int_0^t f'(X_s)dX_s + \frac{1}{2}\int_0^t f''(X_s)d[X]_t \\ &= f(X_0) + \int_0^t f'(X_s)H_s dB_s + \int_0^t f'(K_s)ds + \frac{1}{2}\int_0^t f''(X_s)H_s^2 ds \end{aligned}$$

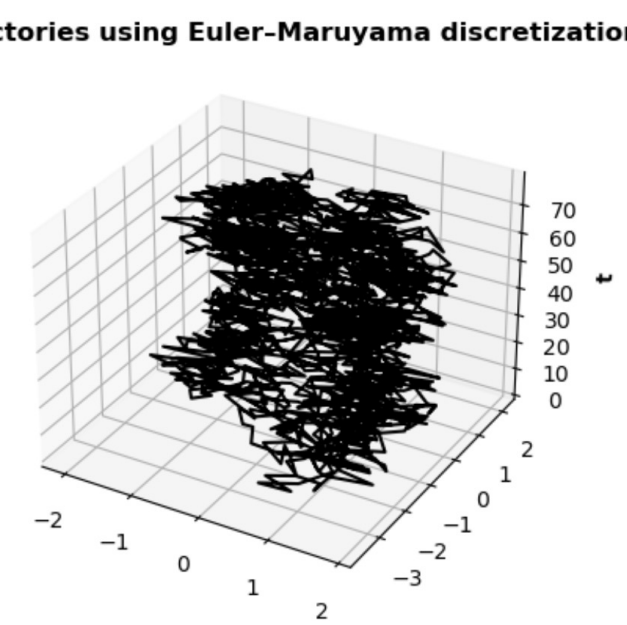
Example 1 - 1D Harmonic Potential



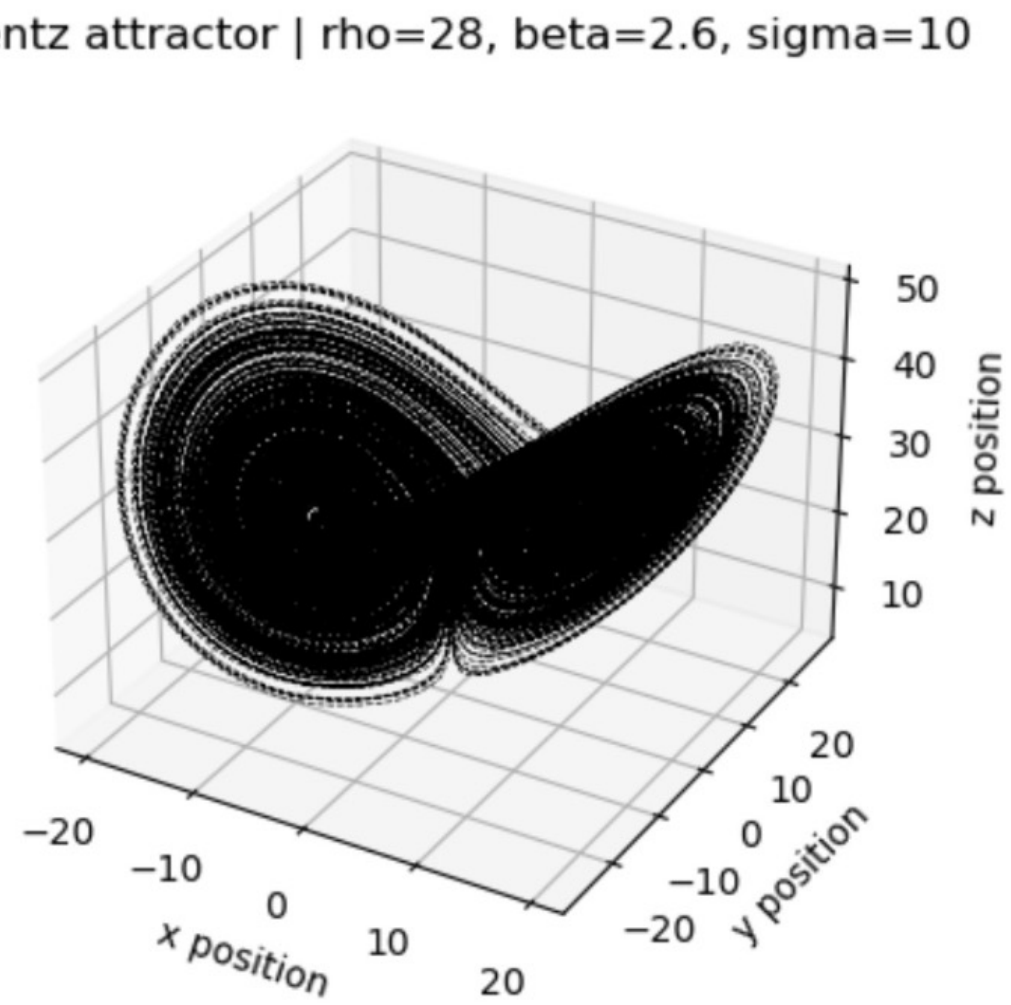


Example 2 - 2D Double-Well Potential

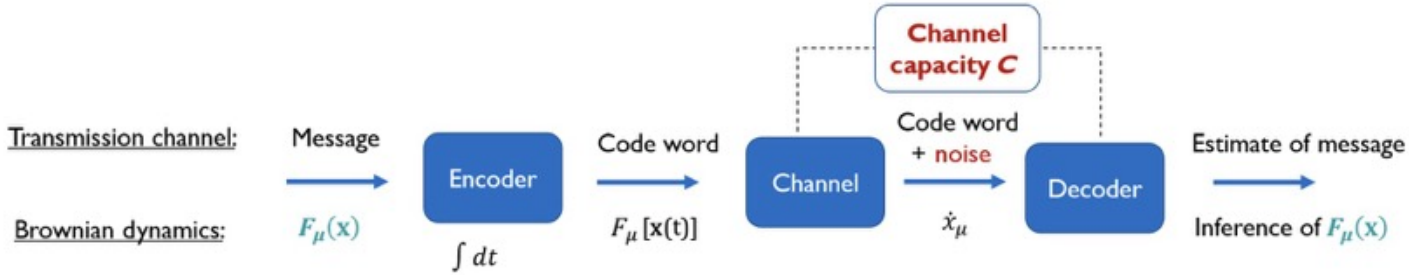




Example 3 - Lorenz Attractors



The Information Content of Brownian Trajectories



Interpret Brownian dynamics as a noisy transmission channel, the dynamics of a Brownian system correspond to an infinite bandwidth Gaussian channel.
Information can be read out from such a channel at a maximal rate C called the channel capacity. The corresponding channel's capacity is

$$C = \frac{1}{4}\int F_\mu(x) D^{-1}_{\mu\nu} F_\nu(x) P(x) dx \qquad (7)$$

SFI-Principle of the method

Input. A time series $\{\boldsymbol{x}^i\}_{i=1\ldots N_{steps}}$ at times $t_i = 0\ldots\tau$ of d-dimensional data. We define $\Delta\boldsymbol{x}^i = \boldsymbol{x}^{i+1} - \boldsymbol{x}^i, \Delta t_i = t_{i+1} - t_i,$ and the mid-interval positions $\boldsymbol{x}^{i+\frac{1}{2}} = (\boldsymbol{x}^{i+1} + \boldsymbol{x}^i)/2$. We write x^i_μ the μ-th component of $\boldsymbol{x}^i$

Model. The input data is assumed to obey Brownian dynamics,

$$\dot{x}_\mu = F_\mu(\boldsymbol{x}) + \sqrt{2D(\boldsymbol{x})}_{\mu\nu}\xi_\nu \tag{8}$$

Output. Stochastic Force Inference returns the inferred diffusion tensor field $\mathrm{D}_{\mu\nu}(\boldsymbol{x})$, force field $F_\mu(\boldsymbol{x})$ and out-of-equilibrium velocity field $\mathrm{v}_\mu(\boldsymbol{x})$. These are obtained by linear regression of local estimators with a set of n_b fitting functions $b_\alpha(\boldsymbol{x})_{\alpha=1\ldots n_b}$ (the "projection basis").

Choice of the basis. A good projection basis should be smooth, adapted to the symmetries of the problem, and with a number of functions in agreement with the information available in the trajectory. If an educated guess on the functional form of the field to infer is available, use it, otherwise use a generic basis:

- Polynomial basis, up to order n
- Fourier basis

Normalization. The inference methods use normalized fitting functions $c_\alpha(\boldsymbol{x})$, obtained from the unnormalized functions $b_\alpha(\boldsymbol{x})$ through:

$$\hat{c}_\alpha(\boldsymbol{x}) = \hat{B}^{-\frac{1}{2}}_{\alpha\beta} b_\beta(\boldsymbol{x}) \tag{9}$$

$$\hat{B}_{\alpha\beta} = \sum_i \frac{\Delta t_i}{\tau} b_\alpha(\boldsymbol{x}^i) \tag{10}$$

SFI–Diffusion inference

- Pick and normalize a basis to infer a space-independent diffusion tensor choose b=$\{1\}$.
- Compute the diffusion projection coefficients:

$$\hat{D}_{\mu\nu\alpha} = \frac{1}{4\tau} \sum_i \hat{c}_\alpha(\boldsymbol{x}^i)[\Delta x^i_\mu \Delta x^i_\nu + \Delta x^{i-1}_\mu \Delta x^{i-1}_\nu] \tag{11}$$

- The reconstructed diffusion tensor field is:

$$\hat{D}_{\mu\nu}(\boldsymbol{x}) = \hat{D}_{\mu\nu\alpha} c_\alpha(\boldsymbol{x}) \tag{12}$$

Error estimate. The relative error on the projection coefficients is

$$\frac{\delta \hat{D}^2}{D^2} \sim \frac{dn_b}{N_{steps}} \tag{13}$$

- Pick and normalize a basis.
- The force projection coefficients are:

$$\hat{F_{\mu\alpha}} = \hat{v}_{\mu\alpha} - \sum_i \frac{\Delta t_i}{\tau} \partial_\nu [\hat{D}_{\mu\nu} \hat{c}_\alpha](\boldsymbol{x}^{i+\frac{1}{2}}) \tag{14}$$

- The reconstructed force is:

$$\hat{F}_\mu(\boldsymbol{x}) = \hat{F}_{\mu\alpha} c_\alpha(\boldsymbol{x}) \tag{15}$$

Error estimate. The error on the projection coefficients is

$$\frac{\delta \hat{F}^2}{F^2} \sim \frac{dn_b}{2\hat{I}_b} \tag{16}$$

SFI–Entropy production

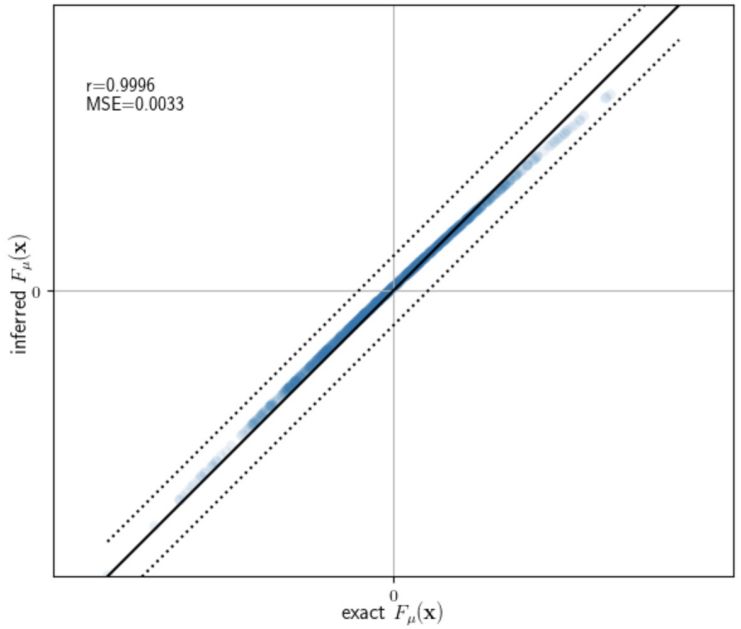
The estimator for the entropy produced along the trajectory is:

$$\Delta \hat{S}_b = \sum_i \hat{v}_\mu(\boldsymbol{x}^i) \hat{D}^{-1}_{\mu\nu}(\boldsymbol{x}^i) \hat{v}_\nu(\boldsymbol{x}^i) \tag{17}$$

with a absolute error of order $(2\Delta \hat{S}_b)^{\frac{1}{2}} + 2dn_b$.

Example 1 - 1D Harmonic Potential

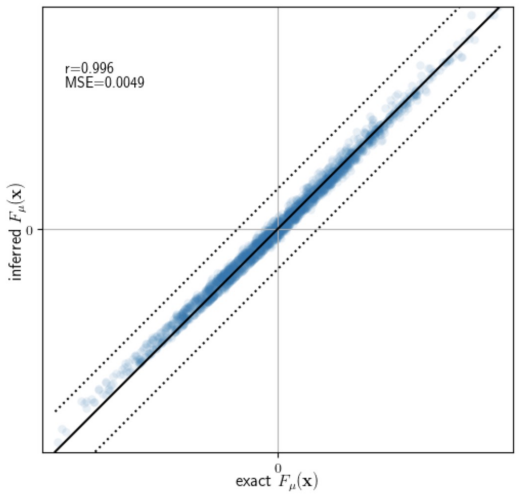
$$dx(t) = -3x(t)dt + \sqrt{2}dW(t)$$




```
Comparing to exact data...
Mean squared error on force: 0.007056916987085469
Mean squared error on diffusion: 0.00048820658693336826
```

Example 2 - 2D Harmonic Potential

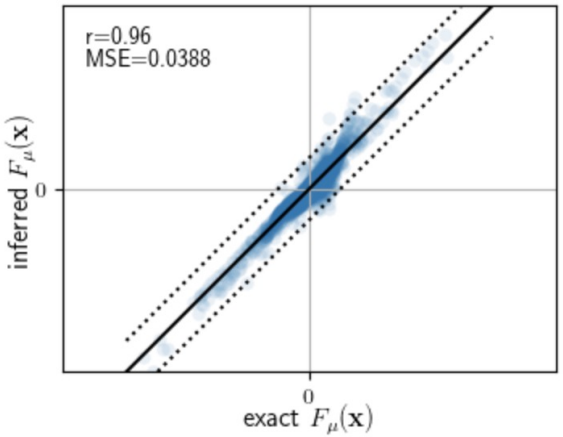
$$dx(t) = -3x(t)dt + 3y(t)dt + \sqrt{2}dW(t)$$
$$dy(t) = -3y(t)dt + 3x(t)dt + \sqrt{2}dW(t)$$




```
Comparing to exact data...
Mean squared error on force: 0.010062514525953309
Mean squared error on diffusion: 0.00014344862096400454
```

Example 3 - 2D Double-Well Potential

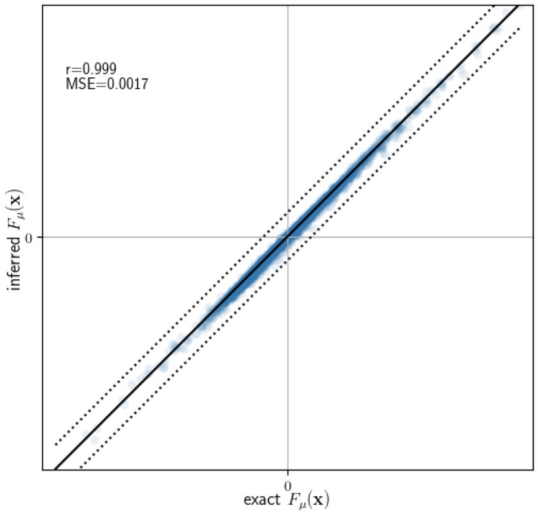
$$dx(t) = -4(x(t)^3 - x(t))dt + 3y(t)dt + dW(t)$$
$$dy(t) = -2y(t)dt + dW(t)$$

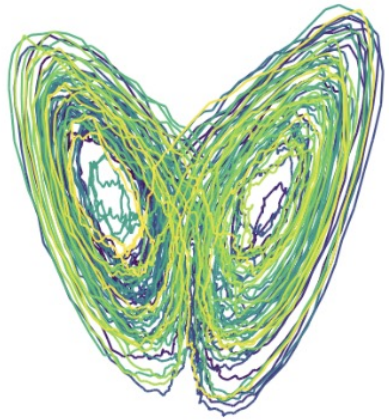


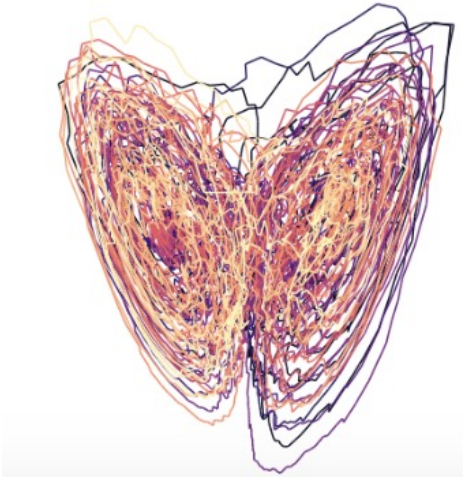

```
Comparing to exact data...
Mean squared error on force: 0.07305560182240116
Mean squared error on diffusion: 0.0022379411451160535
```

Example 4 - Lorenz Attractor

rho=27, beta=2, sigma=10

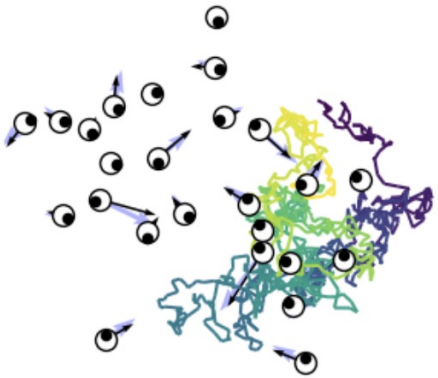


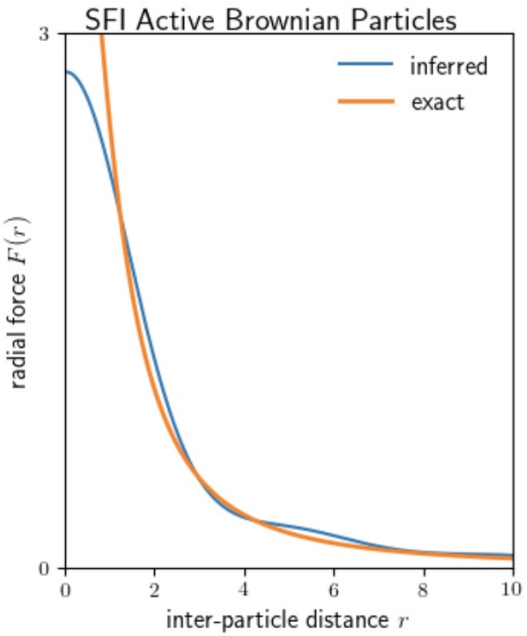


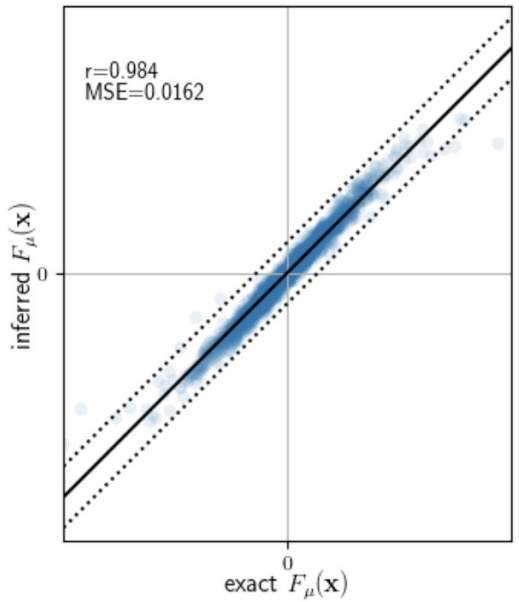


Example 5 - Active Brownian Particles

Stochastic force inference for harmonically trapped active Brownian particles with soft repulsive interactions $F(r) = 1/(1 + r^2)$ between particles at distance r.







Bayesian Identification of Stochastic Differential Equations (BISDE): Definition

Novel method, developed by Yasen Wang (2022), which uses SBL (Sparse Bayesian Learning) for identifying SDEs formulating it as a regression problem. It identifies all entries of the drift and diffusion from a relatively limited quantities of time-series data.

Advantages

Input-Output regression

One trajectory

SBL

Disadvantages

- Prior knowledge
- Thereshold
- Physical Units

How does it work?

SBL can identify the underlying model structure given input vectors with the corresponding output vectors.

Input:

$$X = \begin{bmatrix} | & | & | & | \\ x(1) & x(1) & \cdots & x(N) \\ | & | & | & | \end{bmatrix}^T, \; x(i) \in \mathbb{R}^n$$

Output:

$$Y = [\; y(1) \quad y(2) \quad \cdots \quad y(N) \;]^T, \; y(i) \in \mathbb{R}$$

First we need to construct a library made of basis functions.

$$\Phi = [\; 1 \; X \; X^2 \; \cdots \; sin(X) \; cos(X) \;], \; \Phi \in \mathbb{R}^{NXM}$$

BISDE: Backbone II

To estimate the weight vector, we solve the following regression equation:

$$Y = \Phi\Theta + \varepsilon \ ,$$

where Φ is the library, Θ is the weight vector and ε is the noise vector which follows a Gaussian Distribution.
The likelihood is:

$$p(Y|\Theta;\Psi) = (2\pi)^{-N/2}\,|\Psi|^{-1/2}\exp\left(-\frac{1}{2}(Y-\Psi\Theta)^T\Psi^{-1}(Y-\Psi\Theta)\right),$$

and imposing the sparsity-promoting Gaussian we get the following prior:

$$p(\Theta;\gamma) = \prod_{M}^{i=1}(2\pi\gamma_i)^{-1/2}\exp\left(\frac{\theta_i^2}{\gamma_i}\right) \ ,$$

where γ is the hyperparameter vector.

Given γ we can infer:

$$p(\theta_i = 0|\gamma_i = 0) = 1$$

Finally the posterior distribution of Θ is:

$$p(\Theta|Y; \Psi, \gamma) \propto p(Y|\Theta; \Psi)p(\Theta, \gamma) = \mathcal{N}(\mu, \Sigma)$$

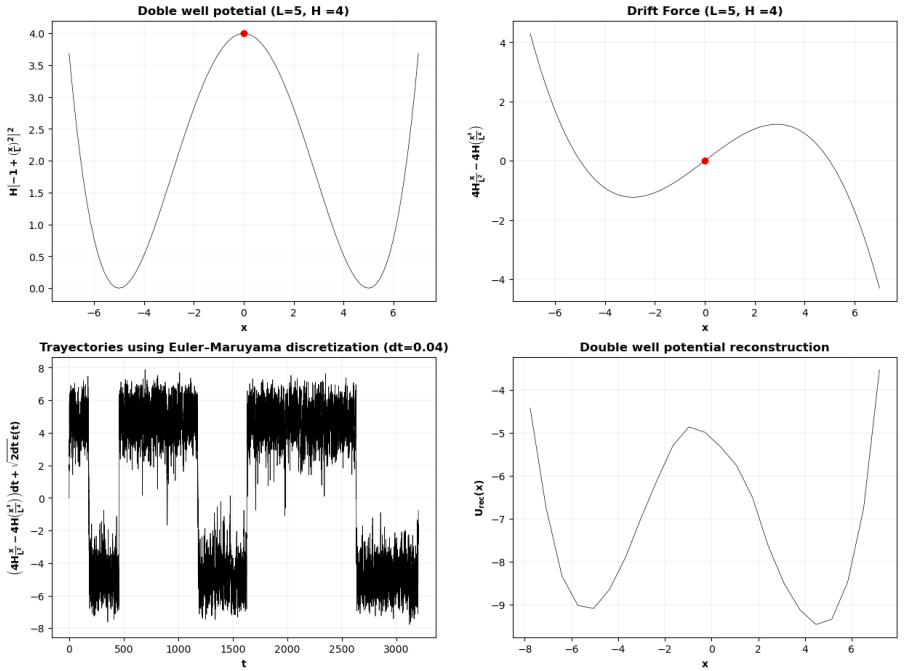
$$\mu = \Sigma\Phi\Psi^{-1}Y \ , \ \Sigma = (\Phi\Psi^{-1}\Phi + \Gamma^{-1})^{-1\prime}, \ \Gamma = diag(\gamma)$$

Based on the MAP principle, μ is selected as the estimator of the weight vector Θ. Next we must maximize the type-II likelihood function. After running several iterations we get the optimal hyperparameter γ and:

$$\Theta = \mu = \Phi^T\Psi^{-1}\Phi + \Gamma^{-1})^{-1}\Phi^T\Psi^{-1}Y$$

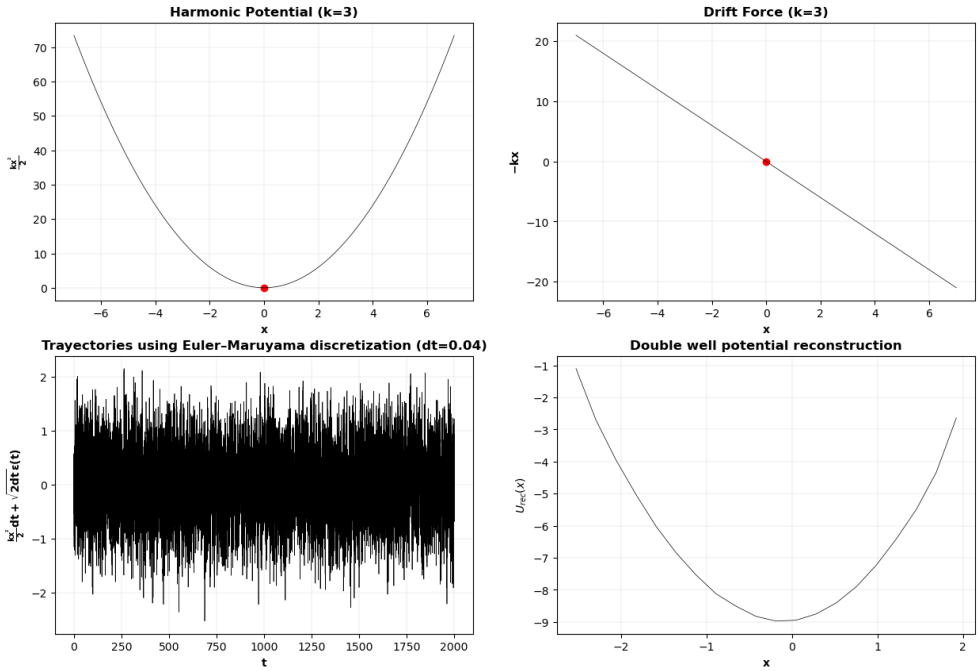
Double Well Potential

param	*DATA*	**drift**		**difussion**
iter=100	**library**	x	x**3	1
tsh=0.02	**real**	0.64	-0.0256	2
N=75000	**inferred**	0.6586	-0.0260	2.012
Δt=0.04	**% error**	2.83%	1.82%	0.59%
init=0	**MSE**	0.0014		0.0001



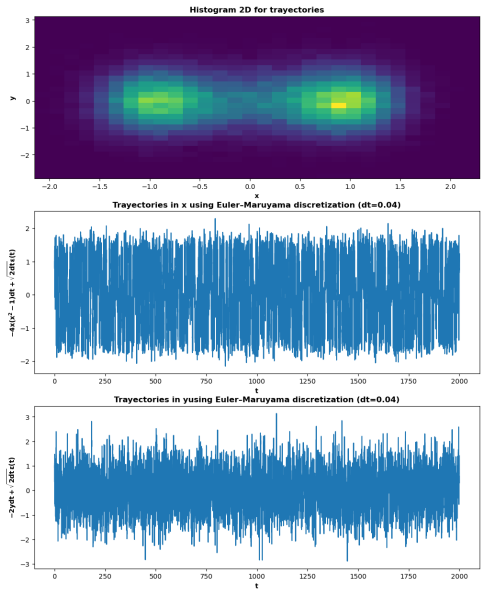
Harmonic Oscillator

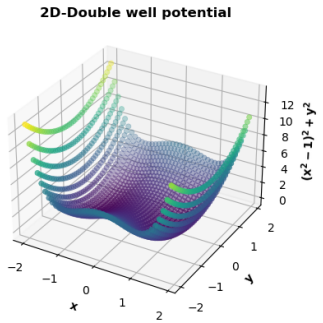
param	*DATA*	**drift**	**difussion**
iter=100	**library**	x	1
tsh=0.02	**real**	-3	2
N=50000	**inferred**	-3.09	2.007
Δt=0.04	**% error**	2.99%	0.38%
init=0	**MSE**	0.0011	0.0001

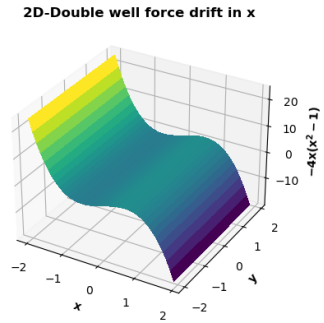


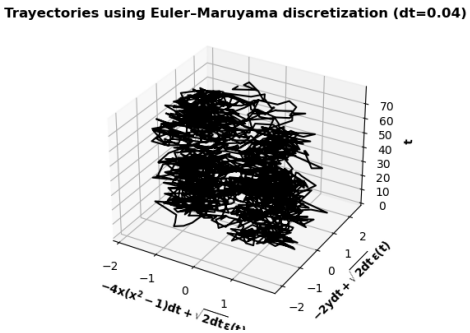
2D Double Well Potential

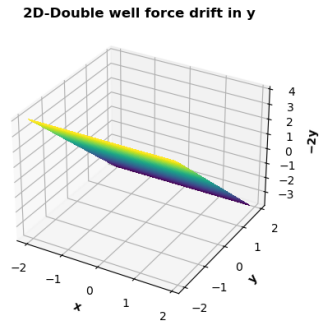
2*param	*DATA*	**drift**			**difussion**	
	dim	x		y	x	y
iter=1000	**library**	x	x**3	y	1	1
tsh=0.05	**real**	4	-4	-2	2	2
N=50000	**inferred**	4.083	-4.04	-1.96	1.98	2.00
Δt=0.04	**% error**	2.03%	1.023	1.57%	0.81%	0.06%
init=0	**MSE**	2.57e-6		-9.6e-7	1e-5	7.5e-7





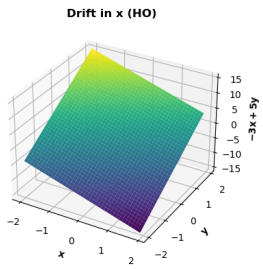


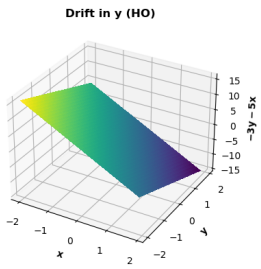


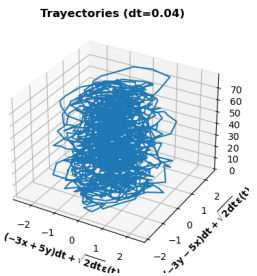


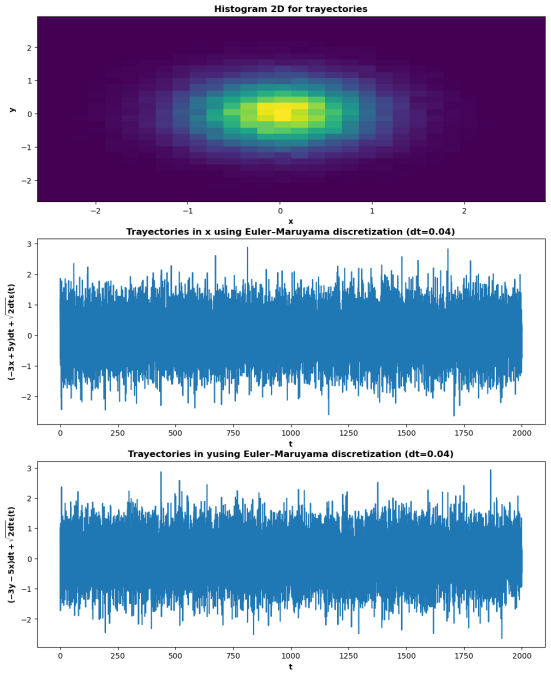
2D Harmonic Potential

2*param	*DATA*	drift				difussion	
	dim	x		y		x	y
iter=1000	**library**	x	y	x	y	1	1
tsh=0.05	**real**	-3	5	-5	-3	2	2
N=50000	**inferred**	-2.98	4.99	-5.01	-2.98	1.97	2.005
Δt=0.04	**% error**	0.62%	0.14%	0.20%	0.61%	1.08%	0.26%









Lorentz:

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

Four-Wing:

$$\frac{dx}{dt} = ax + yz \ , \ \frac{dy}{dt} = bx + cy - xz \ , \ \frac{dz}{dt} = -z - xy$$

Halvorsen:

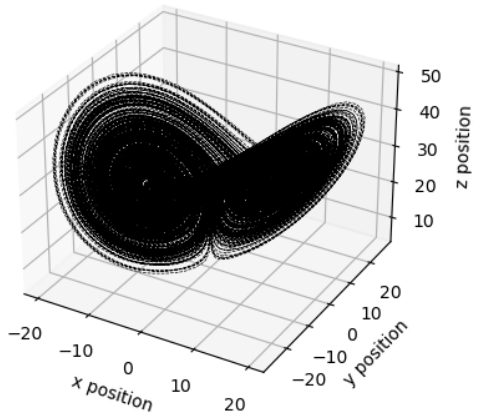
$$\frac{dx}{dt} = -qx - 4y - 4z - y^2 \ , \ \frac{dy}{dt} = -qy - 4z - 4x - z^2$$

$$\frac{dz}{dt} = -qz - 4x - 4y - x^2$$

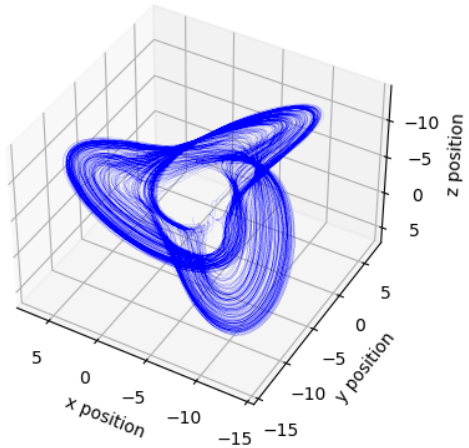
3D Attractors

2*param	*DATA*	**drift — diff=0.1**						
	Attractor	**Lorentz**			**Four-Wings**			**Halvorsen**
iter=50000	**val**	rho	sigma	beta	a	b	c	q
tsh=0.05	**real**	28	10	2.6	0.2	0.1	-0.4	1.89
N=50000	**inferred**	28.007	10.005	2.59	0.16	0.12	-0.411	1.8944
Δt=0.01	**% error**	0.02%	0.05%	0.002%	20%	20%	2.87%	0.23%

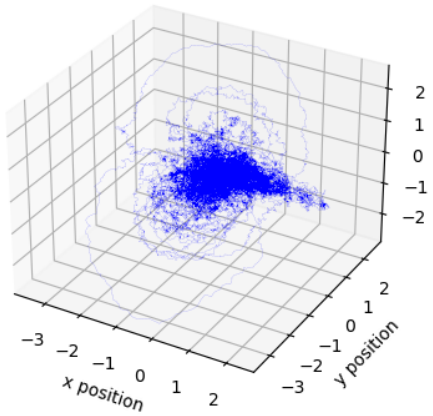
Lorentz attractor | rho=28, beta=2.6, sigma=10



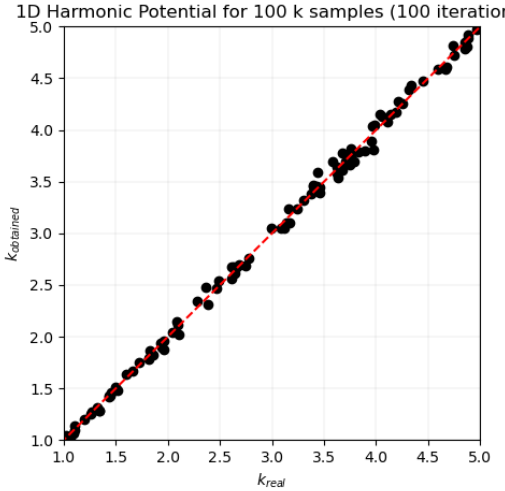
Halvorsen attractor | q=1.89

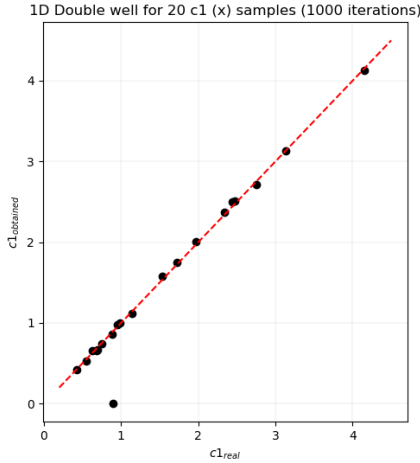


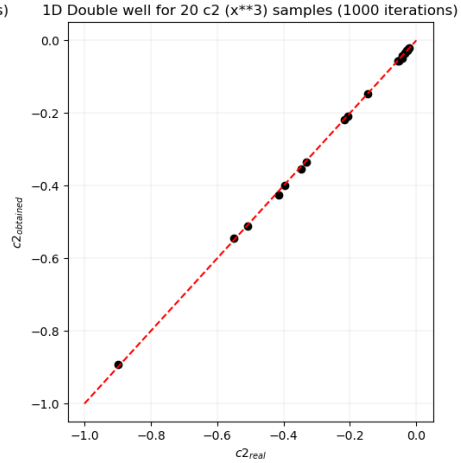
Four-Wing attractor | a=0.2, b=0.1, c=-0.4

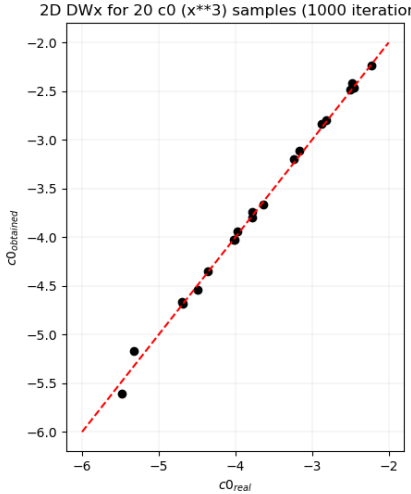


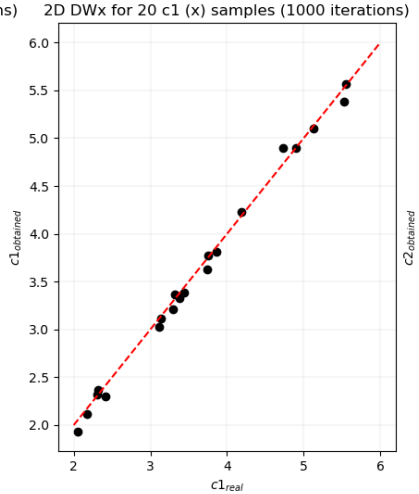
BISDE: N examples

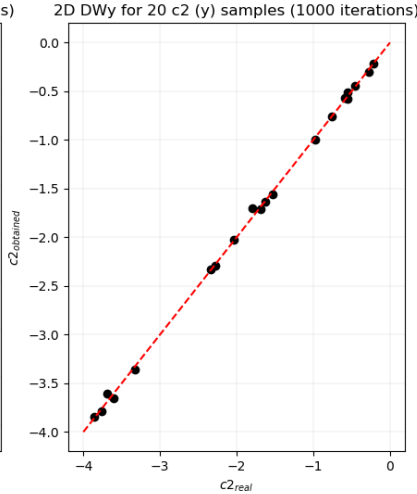




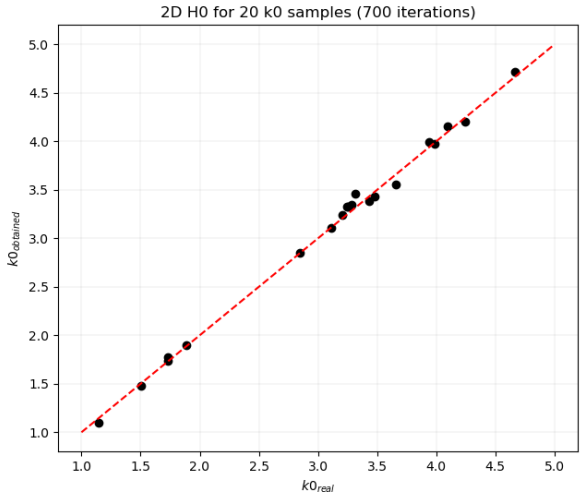


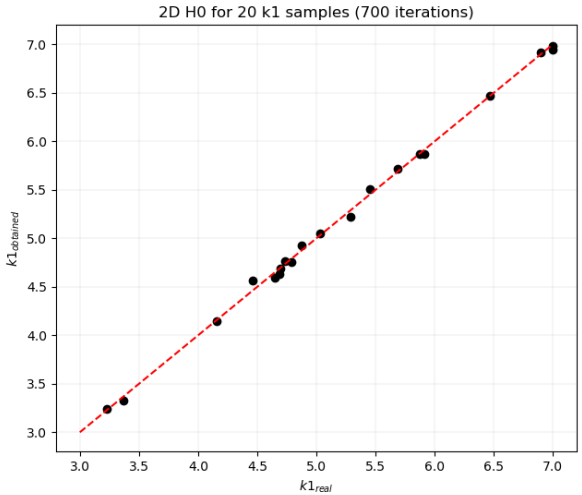






BISDE: N examples





Example	R^2
1D Harmonic	0.99(k)
1D Double Well	0.997(c1) , 0.999 (c2)
2D Harmonic	0.992(k0) , 0.993(k1)
2D Double well	0.997(c0) , 0.995(c1) , 0.999(c2)

Recurrent Neural Network

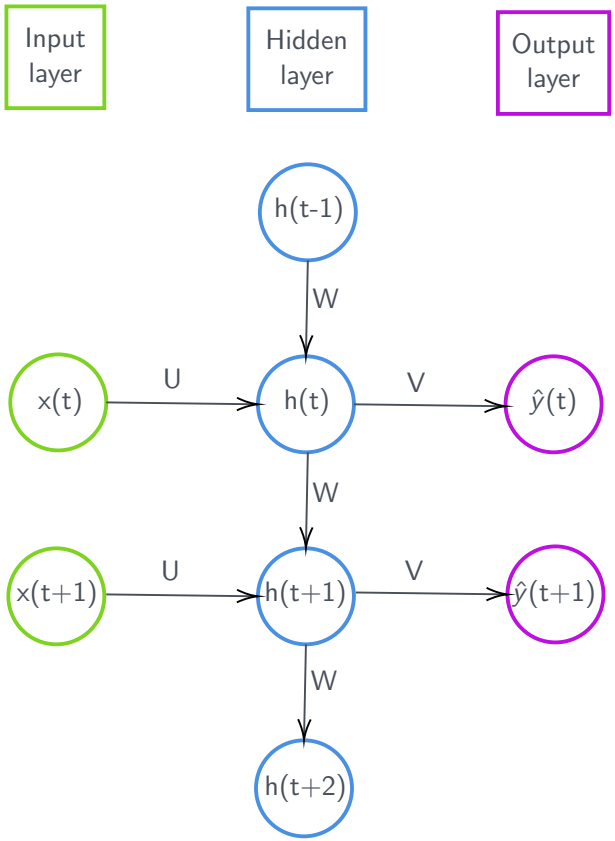
Helps to process sequential data by incorporating the concept of memory.

$$h(t) = tanh(Wh(t-1) + Ux(t))$$
$$\hat{y}(t) = softmax(Vh(t))$$

Loss function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{N_i} L(\hat{y}(t), y(t))$$

- L: cost function between the real and predicted values on a single time step.
- m: size of the training set.
- θ: vector of the model parameters



To train the RNN is required backpropagation through time (BPTT).

LSTM Network

It is capable of learning long term dependencies in data.

- Forget gate

$$f(t) = \sigma(x(t)U_f + h(t-1)W_f)$$

- Input gate

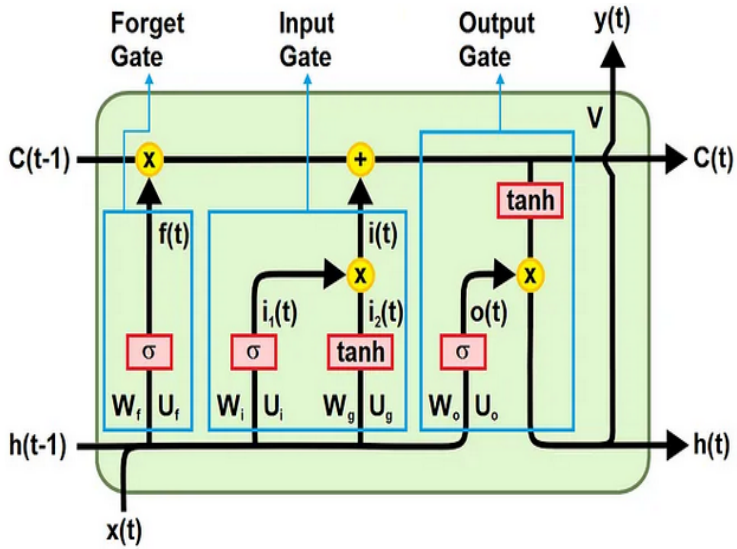
$$i_1(t) = \sigma(x(t)U_i + h(t-1)W_i)$$
$$i_2(t) = tanh(x(t)U_g + h(t-1)W_g)$$
$$i(t) = i_1(t) * i_2(t)$$

- Cell State

$$C(t) = \sigma(f(t) * C(t-1) + i(t))$$

- Output gate

$$o(t) = \sigma(x(t)U_o + h(t-1)W_o$$
$$h(t) = tanh(C_t) * o(t)$$



https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

GRU Network

It is a new generation of RNN that uses the update gate and reset gate.

- Reset gate

$$r(t) = \sigma(x(t)U_r + h(t-1)W_r)$$

- Update gate

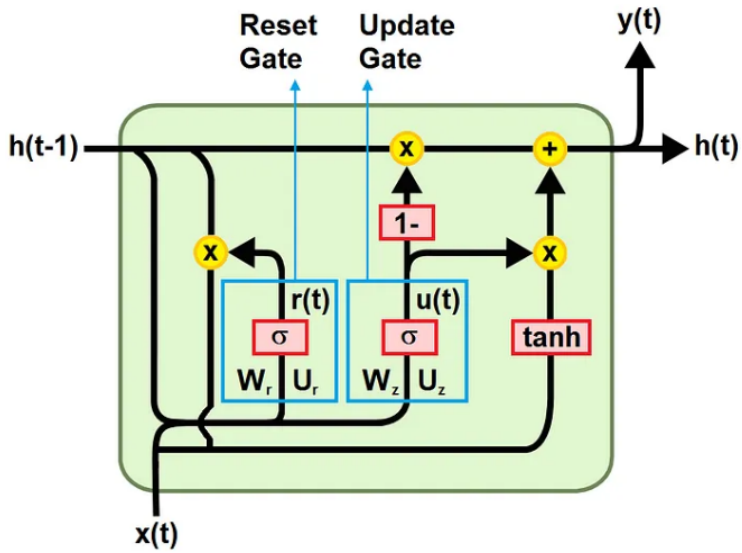
$$z(t) = \sigma(x(t)U_z + h(t-1)W_z)$$

- Current memory

$$\tilde{h}(t) = tanh(x(t)U_h + (r(t) * h(t-1))W_h)$$

- Final memory

$$h(t) = (1 - z(t)) * h(t-1) + z(t) * \tilde{h}(t)$$



https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

Adding attention layer

It refers to a model's ability to focus on specific elements.
Developed to improve the performance on long input sequences.

- Encoder

$$h(t) = f(Wh(t-1) + Ux(t))$$

- Context vector

$$e(j,t) = V_a tanh(U_a s(t-1) + W_a h(j))$$

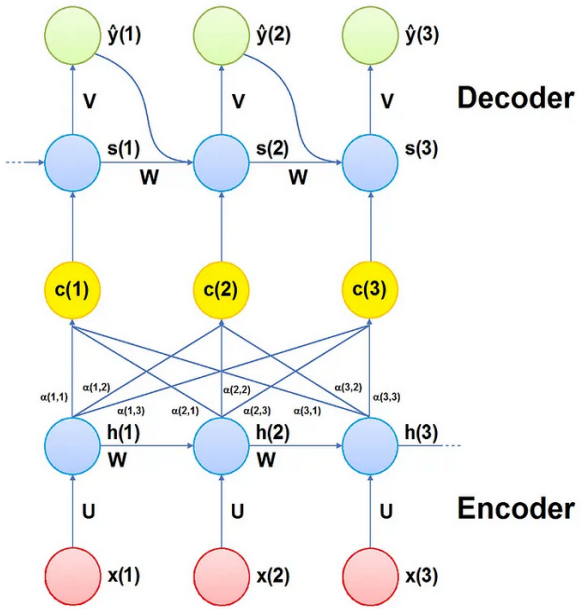
$$\alpha(j,t) = \frac{exp(e(j,t))}{\sum_{j=1}^{M} exp(e(j,t))}$$

$$c(t) = \sum_{j=1}^{T} \alpha(j,t) h(j)$$

- Decoder

$$s(t) = f(s(t-1), \hat{y}(t-1), c(t))$$

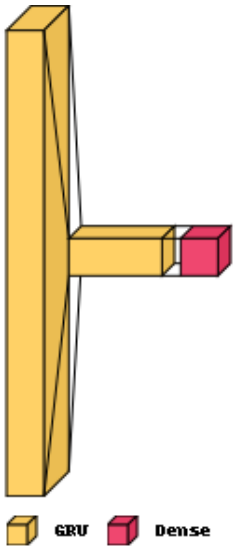
$$\hat{y}(t) = softmax(Vs(t))$$



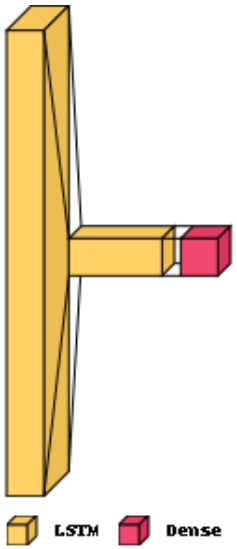
https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

Neural Networks Arquitecture

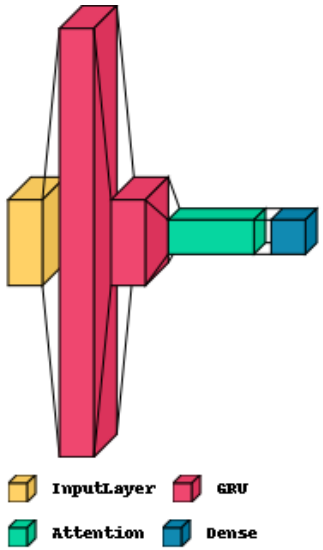
GRU



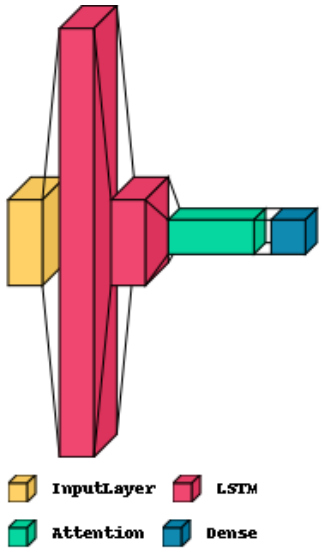
LSTM



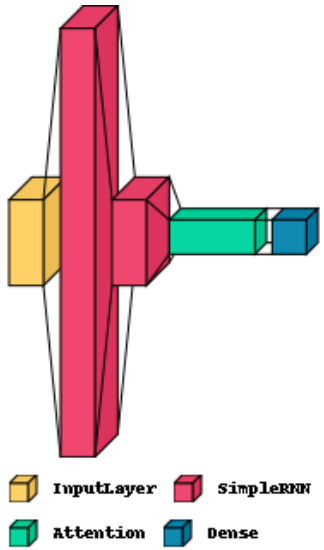
GRU with attention



LSTM with attention



SimpleRNN with attention



Using: input shape = (None, 50), layers dimensions = (250, 50) and number of outputs = 1 for harmonic potential in 1D.

Number of Parameters:

271,851	361,251	271,921	361,321	90,421

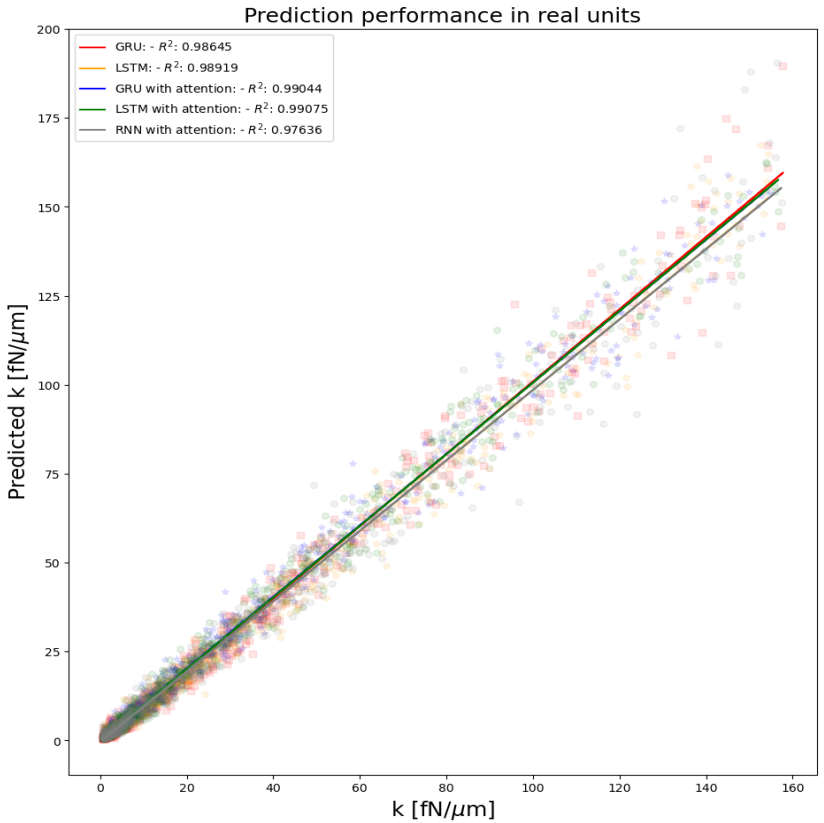
Performance of the models (MSE)

Sample sizes = (32, 128, 512)
Iteration numbers = (3001, 2001, 1001)

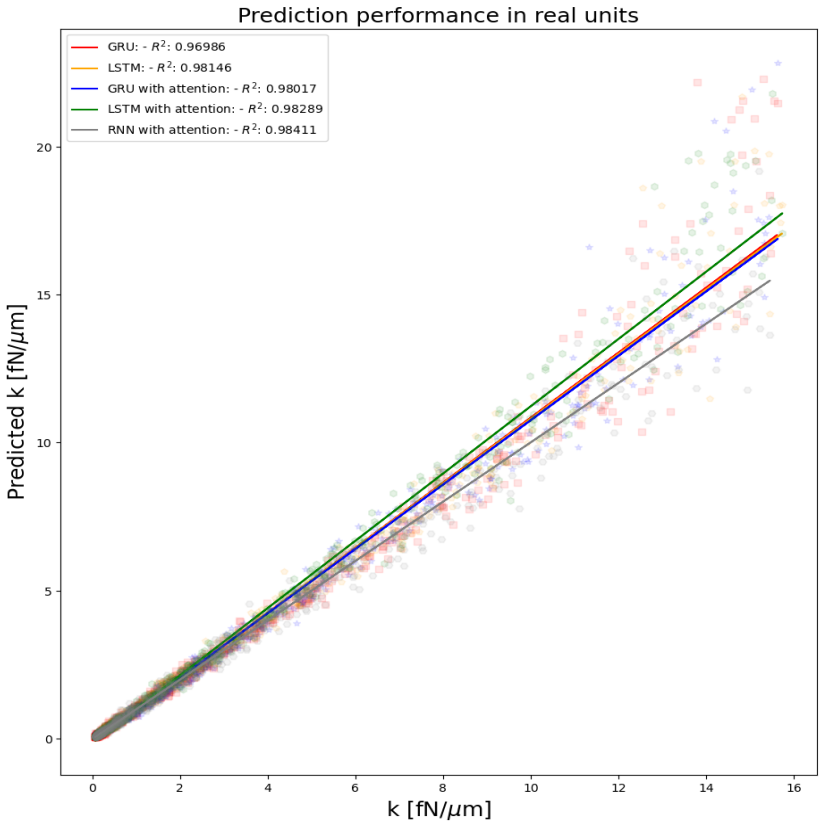
Models	Physical Units				Natural Units			
	HO-1D	DW-1D	DW-2D	L-3D	HO-1D	DW-1D	DW-2D	L-3D
GRU	0.10832	0.10439	0.06931	0.15359	0.04104	0.03126	0.02117	x
LSTM	0.10767	0.10310	0.06891	0.21383	0.04090	0.03602	0.02513	x
GRU attention	0.10881	0.10394	0.06792	0.20517	0.04148	0.02957	0.02009	x
LSTM attention	0.10760	0.10179	0.06801	0.30495	0.04133	0.03392	0.02354	x
SimpleRNN attention	0.11996	0.11667	0.07921	1.20126	0.05866	0.03883	0.02425	x

Prediction using harmonic potential in 1D

Physical Units

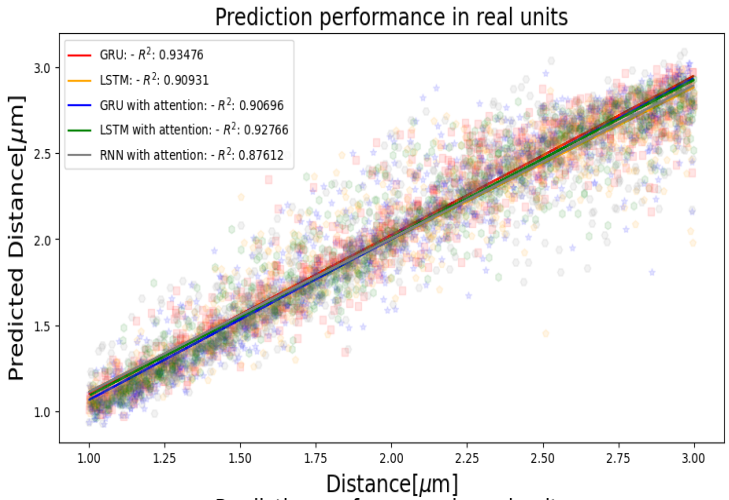


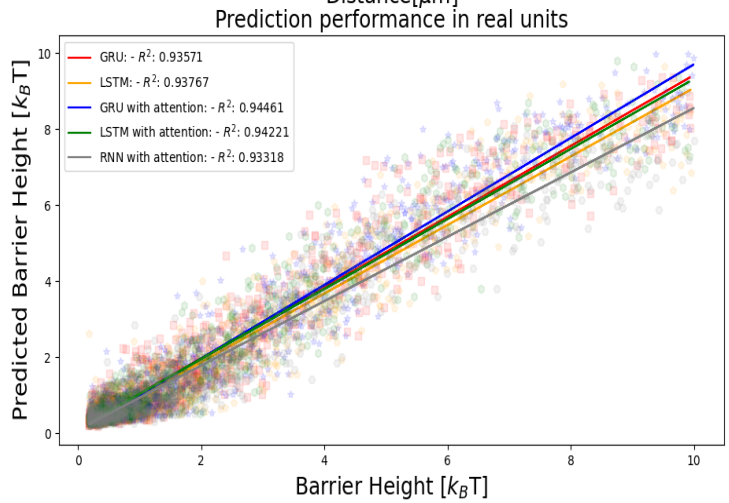
Natural Units



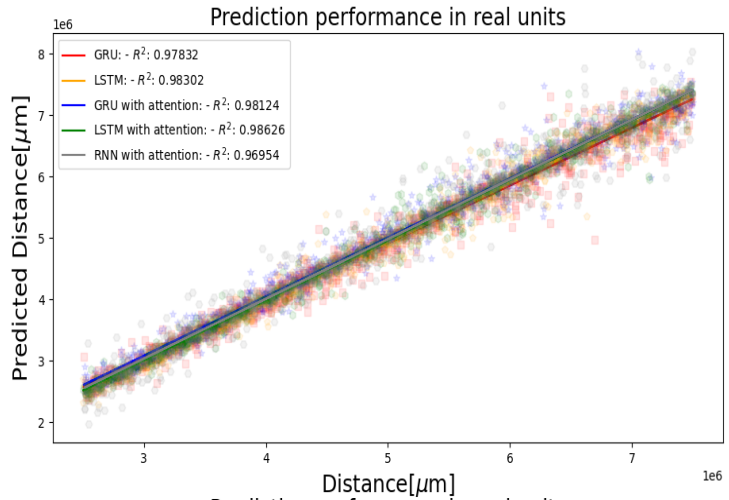
Prediction using DW potential in 1D

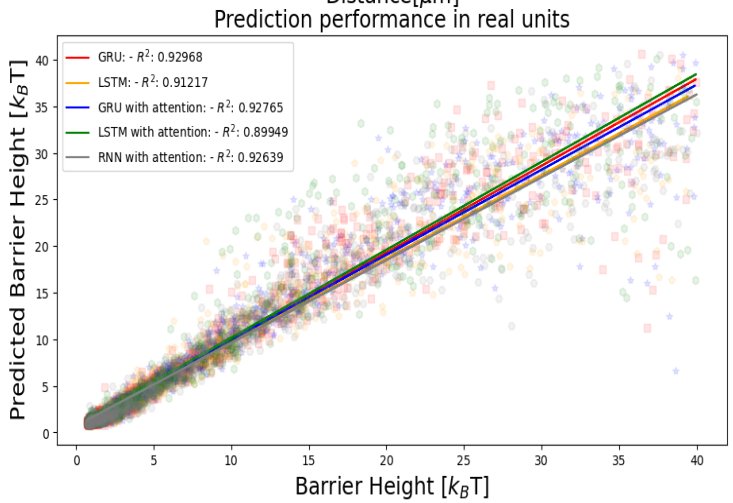
Physical Units





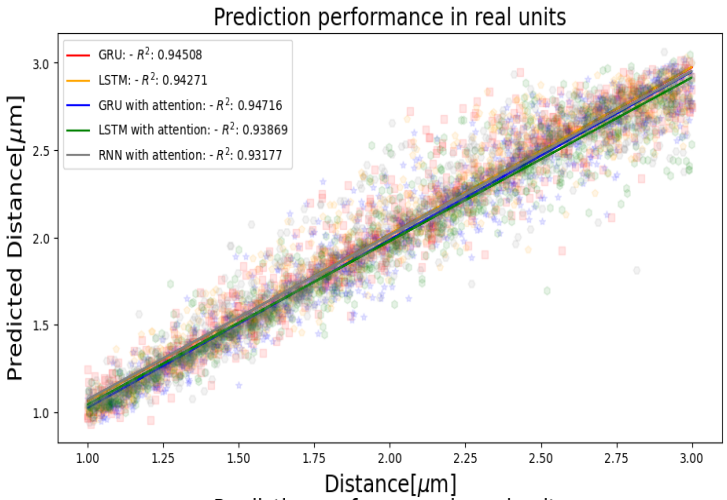
Natural Units

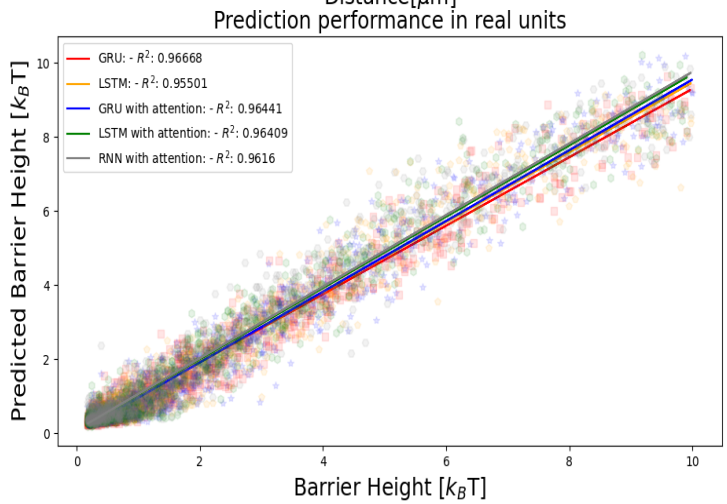




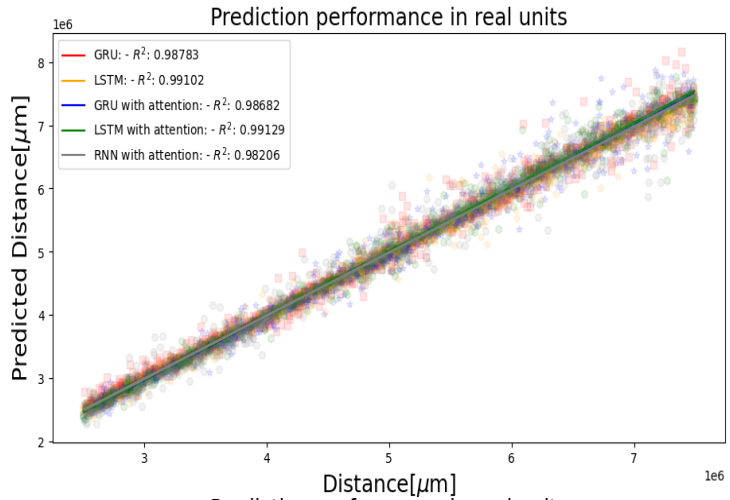
Prediction using DW potential in 2D

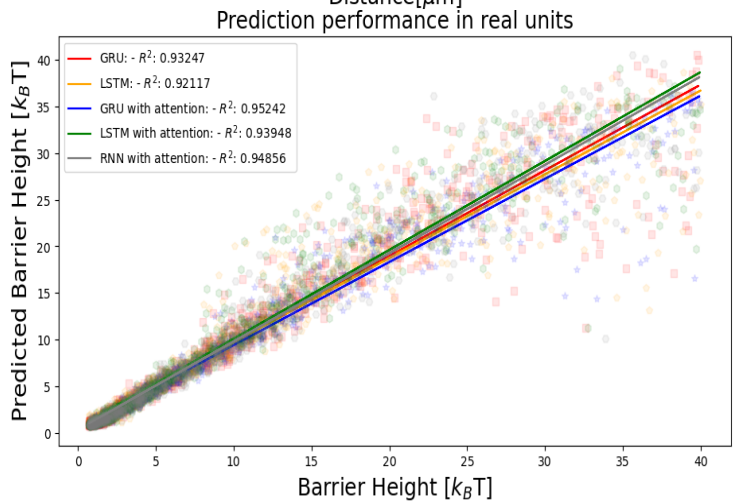
Physical Units





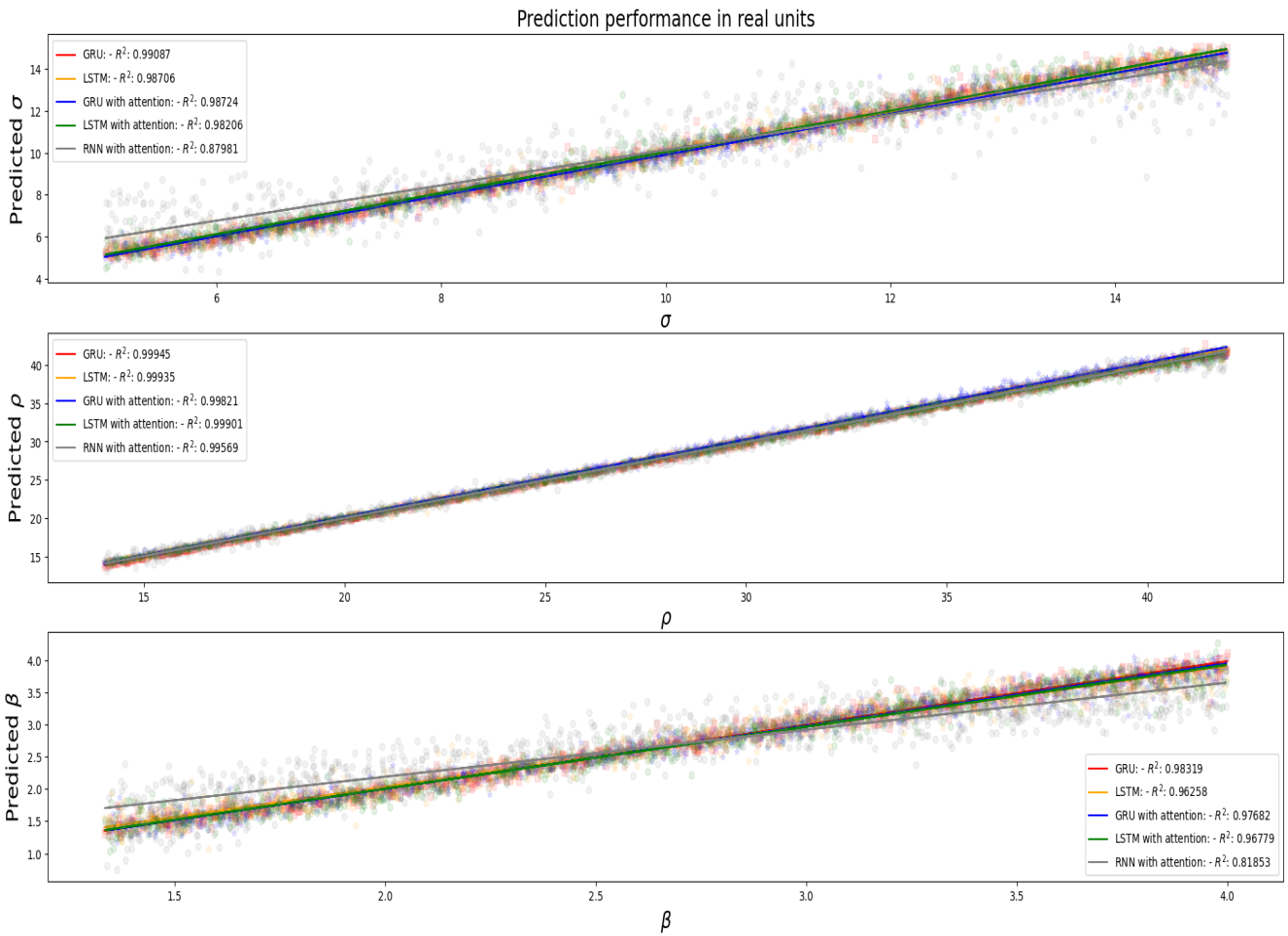
Natural Units





Prediction using Lorenz potential in 3D

Physical Units



Summary of the predictions

Physical Units

Models	HO-1D	DW-1D		DW-2D		L-3D		
	k	L	H	L	H	σ	ρ	β
GRU	0.98645	0.93476	0.93571	0.94508	0.96668	0.99087	0.99945	0.98319
LSTM	0.98919	0.90931	0.93767	0.94271	0.95501	0.98706	0.99935	0.96258
GRU attention	0.99044	0.90696	0.94461	0.94716	0.96441	0.98724	0.99821	0.97682
LSTM attention	0.99075	0.92766	0.94221	0.93869	0.96409	0.98206	0.99901	0.96779
SimpleRNN attention	0.97636	0.87612	0.93318	0.93177	0.96160	0.87981	0.99569	0.81853

Natural Units

Models	HO-1D	DW-1D		DW-2D	
	k	L	H	L	H
GRU	0.96986	0.97832	0.92968	0.98783	0.93247
LSTM	0.98146	0.98302	0.91217	0.99102	0.92117
GRU attention	0.98017	0.98124	0.92765	0.98682	0.95242
LSTM attention	0.98289	0.98626	0.89949	0.99129	0.93948
SimpleRNN attention	0.98411	0.96954	0.92639	0.98206	0.94856

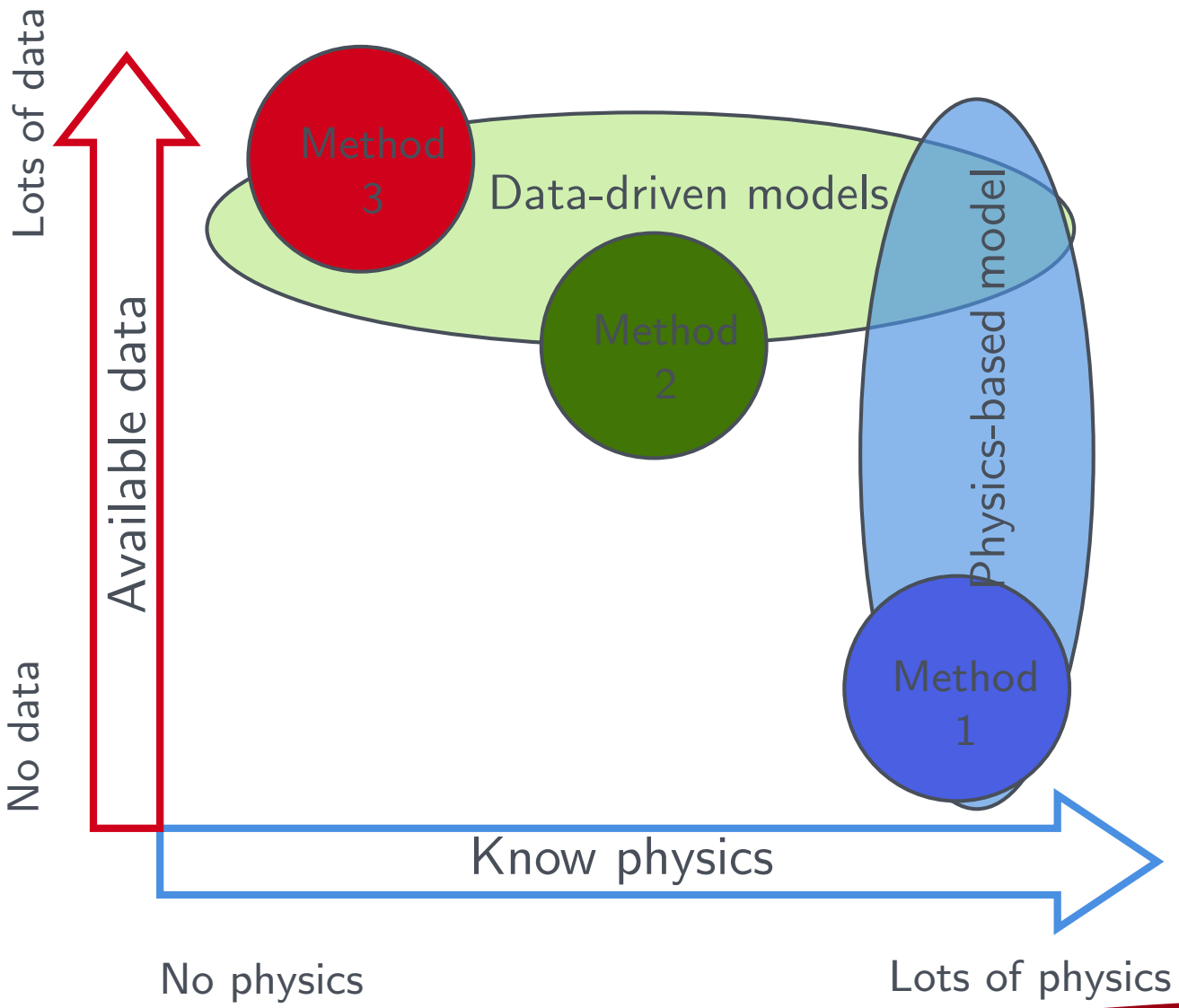
Overall performance

Methods	scale	dimesionality	time duration	trajectories generated	adaptability
1	small (10^{-6}) - large (10^{0})	low (up to 3D)	~ 1 [min]	1	good
2	large (10^{0})	low (up to 3D)	~ 3 [min]	1	medium
3	small (10^{-6}) - large (10^{0})	low (up to 3D)	$60 - 240$ [min]	$\{32, 128, 512\}$	medium

Overall precision

	MSE				R^2			
Methods	H0-1D	DW-1D	DW-2D	L-3D	H0-1D	DW-1D	DW-2D	L-3D α, β, γ
1	0.0033	-	0.0388	0.0017	0.9996	-	0.96 (all)	0.999 (all)
2	-	-	-	-	0.990774	0.9657; 0.99973	0.99711; 0.99576; 0.99922	-
3	0.10760 (LSTM+att.)	0.10179 (LSTM+att.)	0.06792 (GRU+att.)	0.15359 (GRU)	0.99075 (LSTM+att.)	0.93476 (GRU); 0.94461 (GRU+att.)	0.93476 (GRU); 0.94461 (GRU+att.)	0.99087; 0.99945; 0.98319 (GRU)

Discussion


Lots of data
Available data
No data
Method 3
Data-driven models
Method 2
Physics-based model
Method 1
Know physics
No physics
Lots of physics

Final remarks

- We have analyzed the different approaches inferring the parameters of the force fields on the motion of Brownian particles up to 3-dimensions.
- Our approach ranges from a more physics-based model to a pure data-driven model and compared each method's performance.
- The most data efficient method is the method based on information theory
- From the data-driven models, the more time efficient turn out to be the Bayesian Learning method.
- When lots of data is available and poor knowledge of physics, a Neural Network approach would be the best fit.
- So, which method we pick? Well...it depends mostly on the problem and the data available.