CHAOS Lab

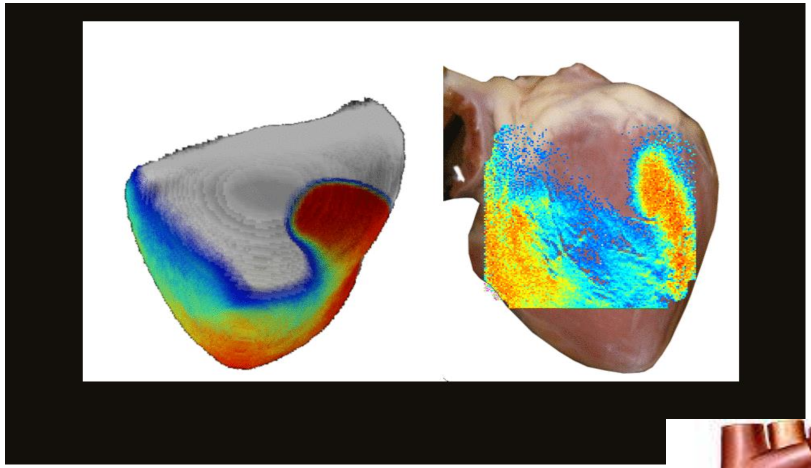
Complex Heart Arrhythmias and other Oscillating Systems

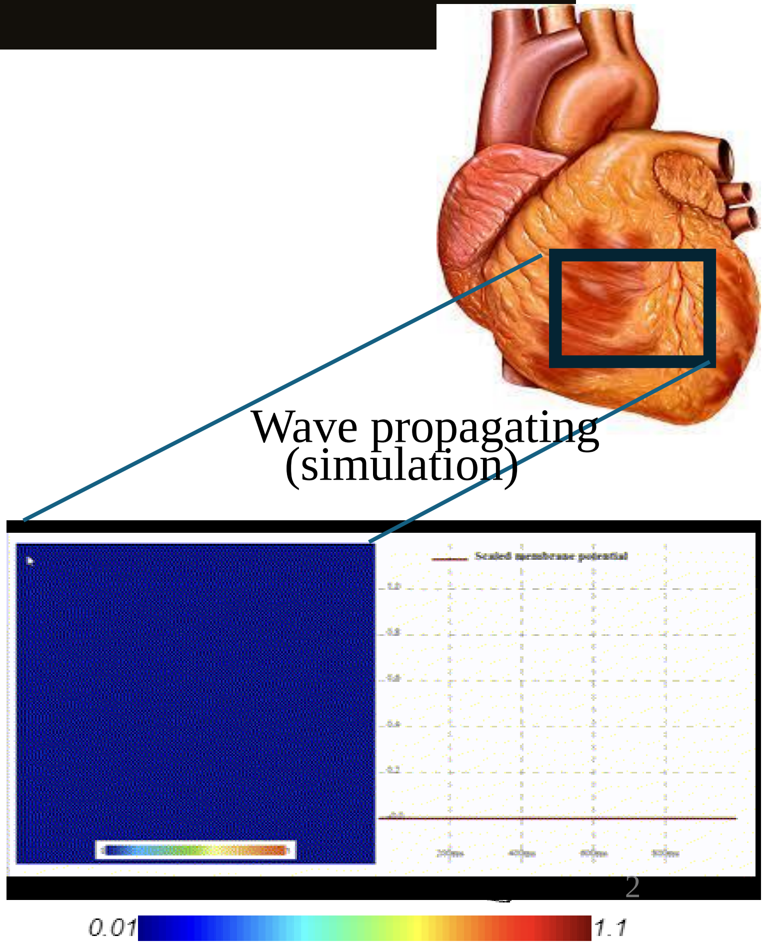
Quantifying the Complexity of Cardiac System Simulation by analyzing APD sequence

Xiaodong (Will) AN

Contents

- Introduction
- Method (Literature Review)
- Current Result and Future Work
- Conclusion





Introduction

- This research tries to do these things:

- 1. Quantify the complexity of cardiac system simulations with parameter changing
- 2. Quantify simulations with meandering cases.
- 3. Quantify experimental results that have noise.

- With determination of the chaotic and non-chaotic regions for different parameters, it can guide the cardiac medication to avoid deadly chaos and help us understand the cardiac model more.

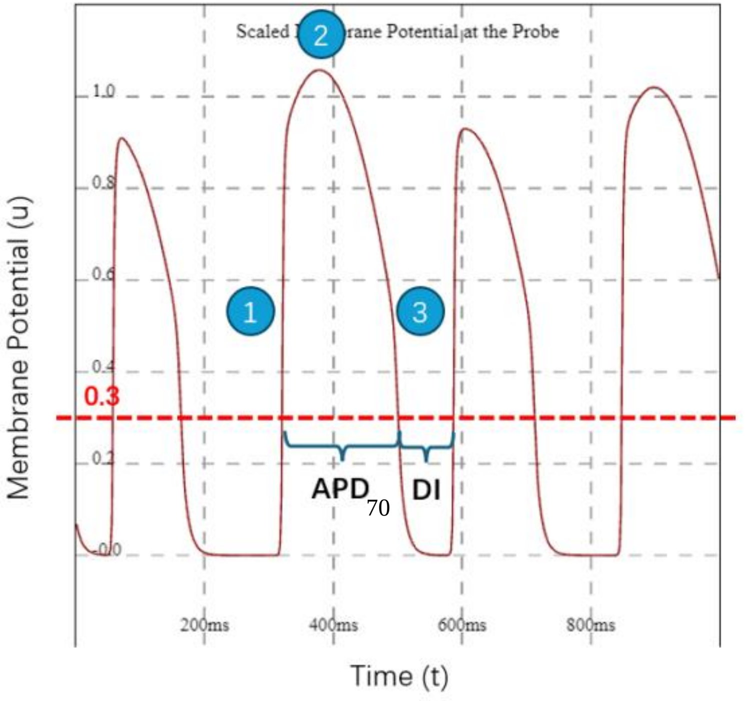
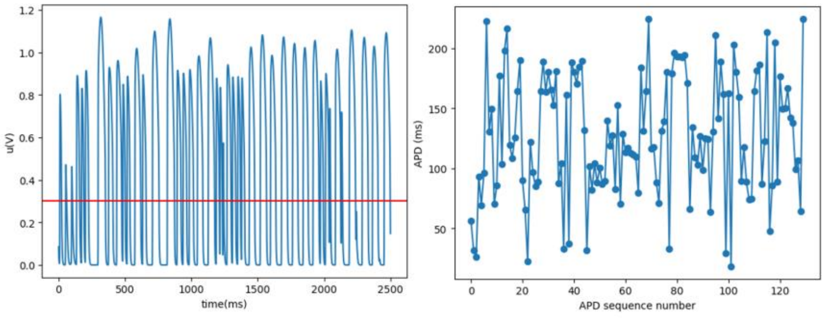
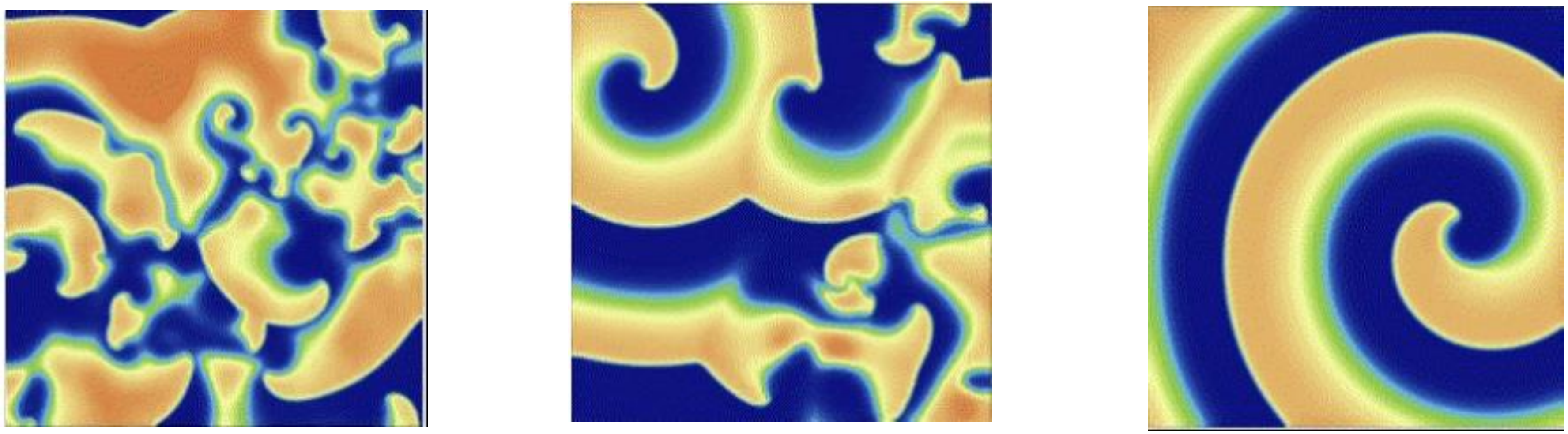


Figure 1: Membrane potential by 3V SIM Model



Introduction – Chaos Quantification



They have different complexities, but how do we quantify them?

Introduction – Chaos Quantification

- There are **several approaches** for chaos quantification, including leading Lyapunov exponent (Characteristic exponent), Correlation dimension, and return map.

- Lyapunov exponent: Quantification of the exponential growth rate in phase space.
- Correlation dimension: The dimension of the strange attractors in phase space.
- Return map: Visualization of complexity.

Introduction – Lyapunov Exponent (LE)

2.4 Lyapunov Exponent

Lyapunov exponent is a quantitative measure of the divergence rate for nearby trajectories, implying the stability of a nonlinear system, spatially or temporally. Typically, following that basic definition, in a 1D system, the Lyapunov exponent can be naively calculated as [7]:

$$\lambda(x(0)) = \lim_{t\to\infty} \lim_{\delta x(0)\to 0} \frac{1}{t} \ln \frac{\delta x(t)}{\delta x(0)}. \tag{9}$$

where $\delta x(0)$ is the initial separation of two trajectories.

Now, heading to the higher dimensional system, it becomes [13]:

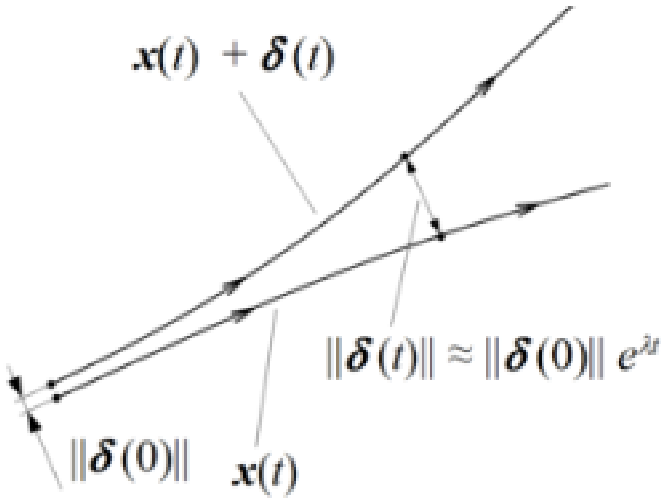
$$\lambda(\boldsymbol{X}(0)) = \lim_{t\to\infty} \frac{1}{t} \ln \frac{\|J^t(\boldsymbol{X}(0))\delta\boldsymbol{X}(0)\|}{\|\delta\boldsymbol{X}(0)\|} = \lim_{t\to\infty} \frac{1}{2t} \ln\left(\hat{n}^\top J^{t\top} J^t \hat{n}\right). \tag{10}$$

where:

- $\hat{n} = \frac{\delta\boldsymbol{X}(0)}{\|\delta\boldsymbol{X}(0)\|}$ is the direction vector.
- $J^t(\boldsymbol{X}(0)) = \Pi_{t'=1}^{t'=t} J^{t'}(\boldsymbol{X}(0))$ is the Jacobian matrix.
- $J^t_{ij}(\boldsymbol{X}(0)) = \frac{\partial \boldsymbol{X}_i(t)}{\partial \boldsymbol{X}_j(0)}$ is the element of the Jacobian matrix.

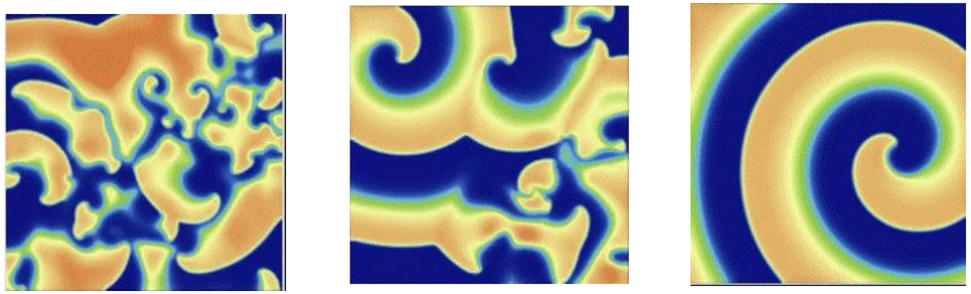
However, in actual calculation, we cannot have infinite time series, so the only one we can get is finite-time Lyapunov exponent, and it is defined as [13]:

$$\lambda(\boldsymbol{X}(0), t) = \frac{1}{2t} \ln\left(\hat{n}^\top J^t J^{t\top} \hat{n}\right). \tag{11}$$



Introduction

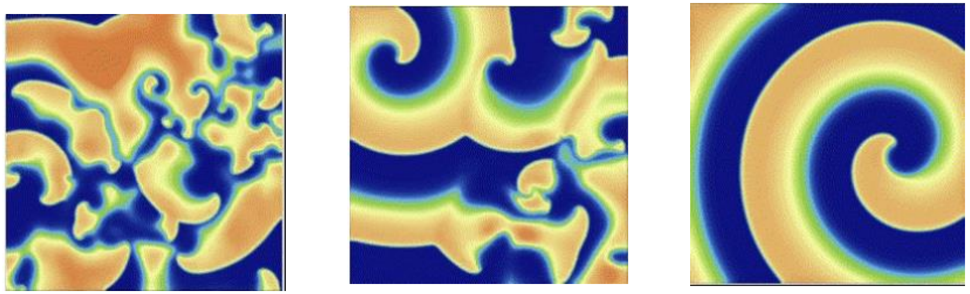
Simulation: 3V SIM Model



$$\Gamma = \left\{ \boldsymbol{X}_i = \left(u_i, v_i, w_i \right) \right\}, i = 1, \ldots, N.$$

Quantify Methods (LE)

Complexity (LE)



0.3 **0.15** **0.01**

Guide us to avoid deadly chaos and understand the cardiac model more.

Method

- 3V SIM Model

- Lyapunov Exponent:
  - Phase Space Reconstruction
  - Wolf's Algorithm
  - Spatial-Temporal Algorithm
  - Noise and Chaos Distinguishment

Method – 3V SIM Model

- 3V SIM model or Fenton-Karma Model was developed in 1990s and it quantitatively reproduced APD vs DI curve (restitution curve) which determines the APD and relevant propagation velocity after repolarization.

- Three variables: u,v,w
- Three currents: I_fi (Na+), I_si (Ca2+), I_so (K+)

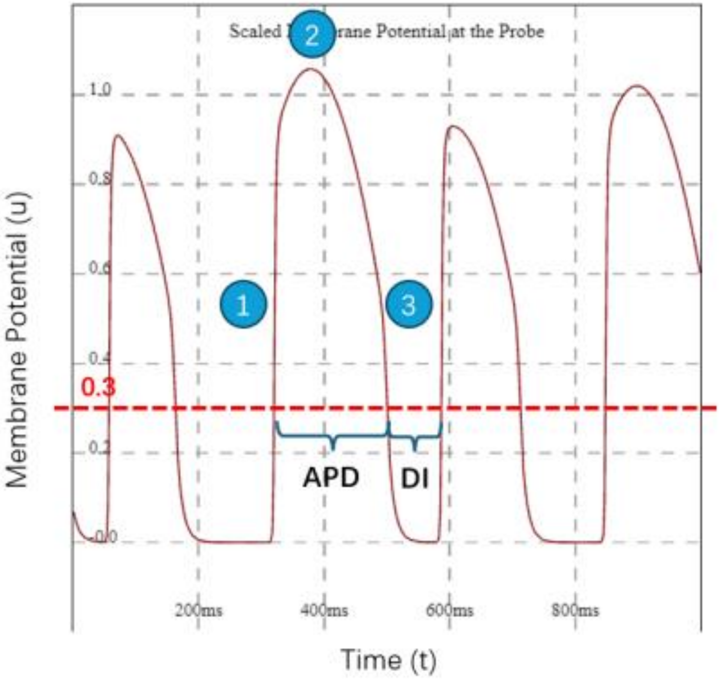


Figure 1: Membrane potential by 3V SIM Model

Method – 3V SIM Model

Finally, the equations are defined as below [6]:

$$\partial_t u(\boldsymbol{x}, t) = D\nabla^2 u - (I_{\mathrm{fi}}(u, v) + I_{\mathrm{so}}(u) + I_{\mathrm{si}}(V, w))/C_m \quad (1)$$

$$\partial_t v(\boldsymbol{x}, t) = (1-p)(1-v)/\tau_v^-(u) - pv/\tau_v^+(u) \quad (2)$$

$$\partial_t w(\boldsymbol{x}, t) = (1-p)(1-w)/\tau_w^-(u) - pw/\tau_w^+(u) \quad (3)$$

$$I_{\mathrm{fi}}(u, v) = -vp(u-u_c)(1-u)/\tau_d \quad (4)$$

$$I_{\mathrm{so}}(u) = u(1-p)/\tau_0 + p/\tau_r \quad (5)$$

$$I_{\mathrm{si}}(u, w) = -w(1 + tanh(k(u - u_c^{\mathrm{si}})))/(2\tau_{\mathrm{si}}) \quad (6)$$

where:

$$p = \mathcal{H}(u - u_c) \quad (7)$$

$$q = \mathcal{H}(u - u_v) \quad (8)$$

and $\mathcal{H}()$ is Heaviside step function

$$\tau_v^-(u) = \Theta(u - u_v)\tau_{v1}^- + \Theta(u_v - u)\tau_{v2}^- .$$

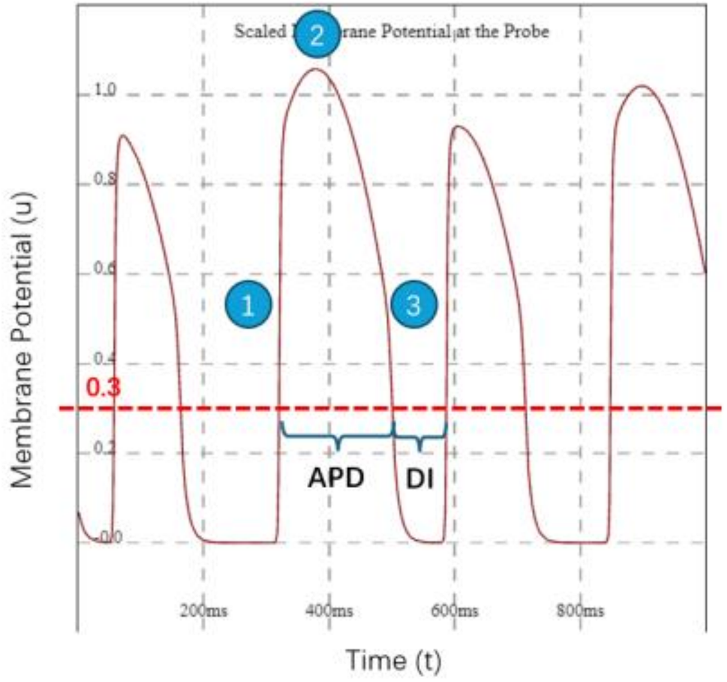


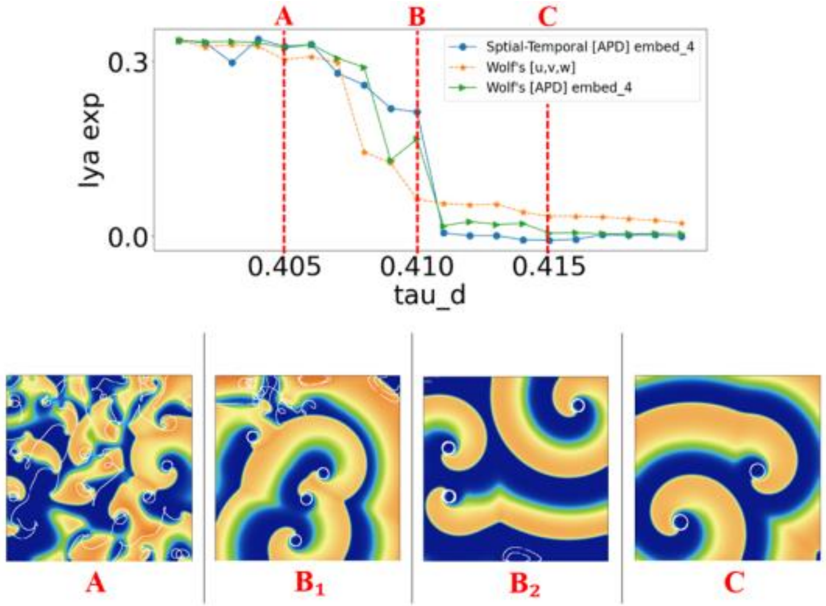
Figure 1: Membrane potential by 3V SIM Model

Method – 3V SIM Model

- Qualitatively, I (Na+) = v / tau_d.

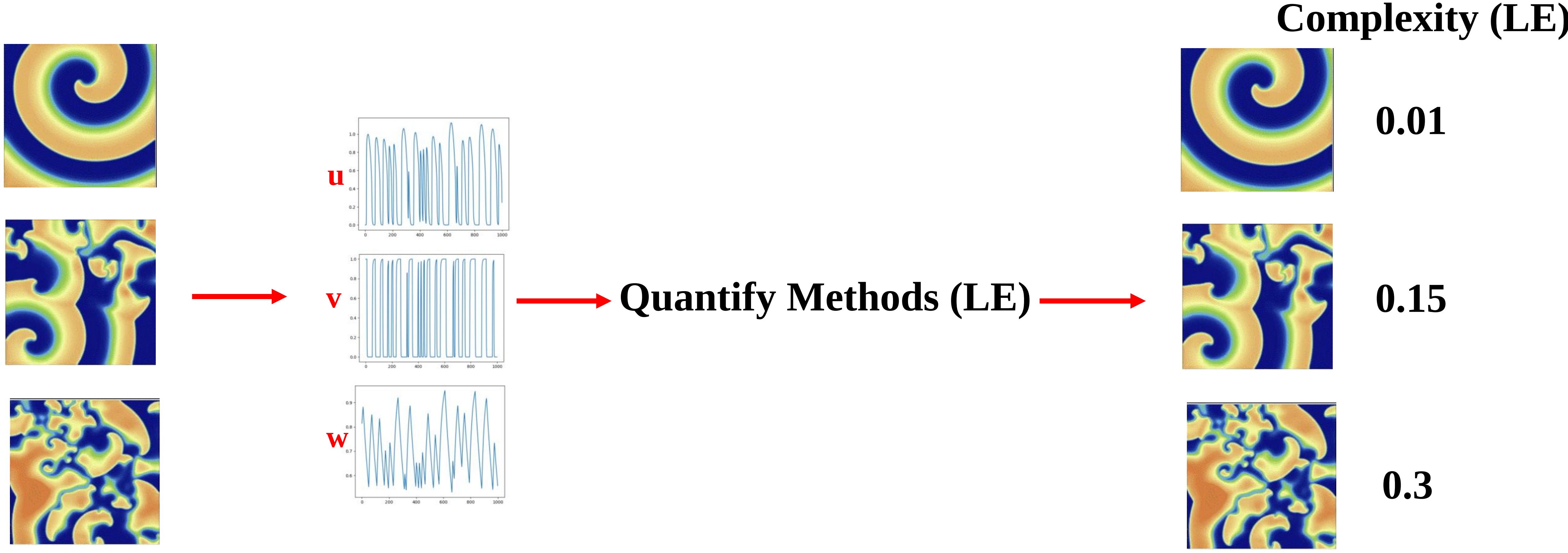
$$I_{\text{fi}}(u, v) = -vp(u - u_c)(1 - u)/\tau_d \quad (4)$$

- So, we can regard tau_d as the resistance of the sodium current
- In later section, I discussed the different complexity by changing tau_d



Method – LE (full phase space)

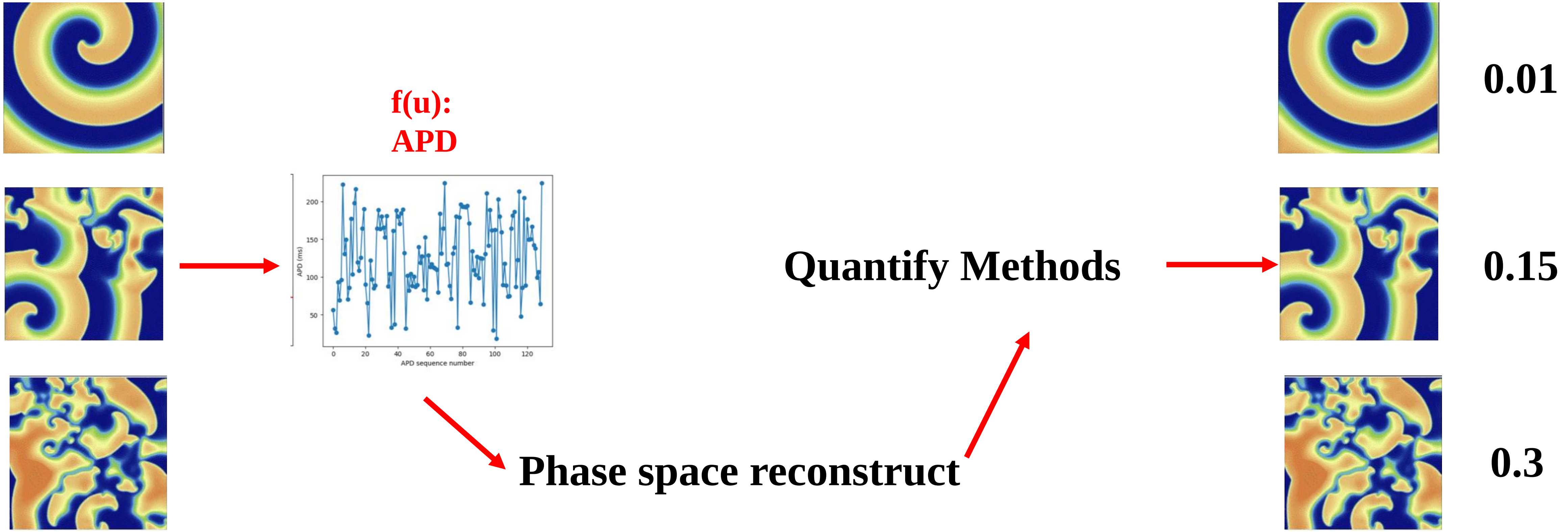
- We can input the full phase space (all variables) into Algo, and get LE.



Method – LE (APD)

- We can also input the APD into Algo, and get LE.
- With less data needed, but nonlinear property retained.

Complexity (LE)



Method – Taken's Theorem

- Limited observations of state variables can retain the Lyapunov exponent by proper lag-embedding.

Given a time series γ:

$$\gamma = \{x_i\}, i = 1, \ldots, N, \tag{12}$$

we could recontruct the phase space $\boldsymbol{\Gamma}$ as :

$$\boldsymbol{\Gamma}^{m,\tau} = \left\{ \boldsymbol{X}_i^{m,\tau} = \Big(x_i(\boldsymbol{k}), x_{i+\tau}(\boldsymbol{k}), \ldots, x_{i+(m-1)\tau}, (\boldsymbol{k}) \Big) \right\}, i = 1, \ldots, N - m\tau + \tau. \tag{13}$$

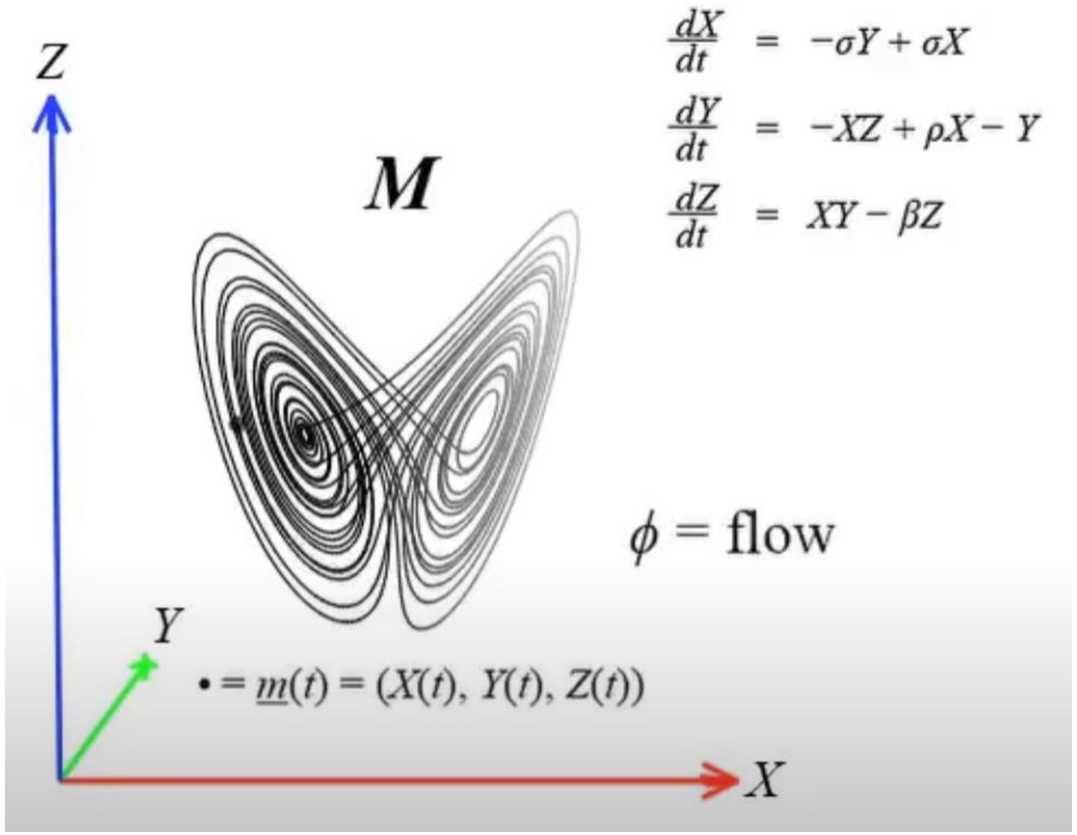
where

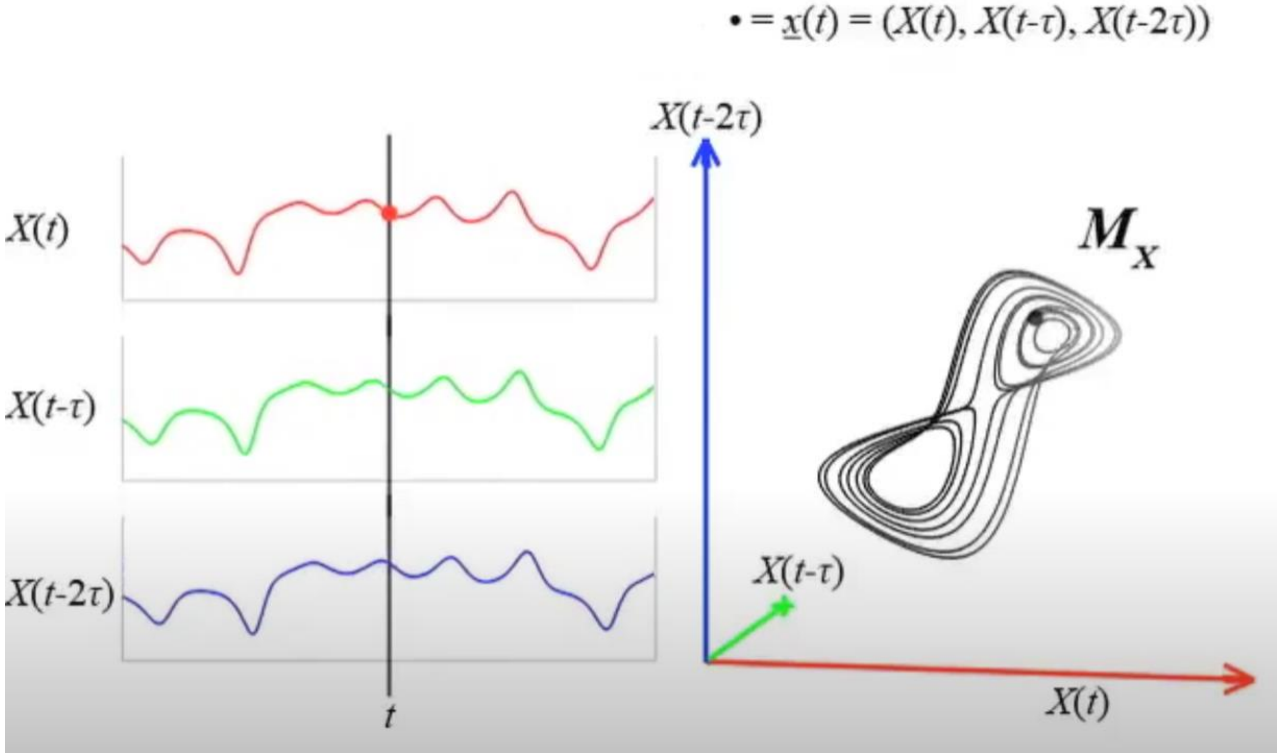
- i is the time step.
- m is the embedded dimension of the time series.
- τ is the lagging of the time series.
- N is the total time steps of the time series.

```
m = 2 # embedded dimension
tau = 2 # lagging

time_series = [x0, x1, x2, x3, x4, x5, x6]
time_series_embedded = [[x0, x2], [x1, x3], [x2, x4], [x3, x5], [x4, x6]]
```

Method – Taken's Theorem





https://www.youtube.com/watch?v=6i57udsPKms&t=40s
Sugihara, George, et al. "Detecting causality in complex ecosystems." *science* 338.6106 (2012): 496-500.

Method – Taken's Theorem (lagging τ)

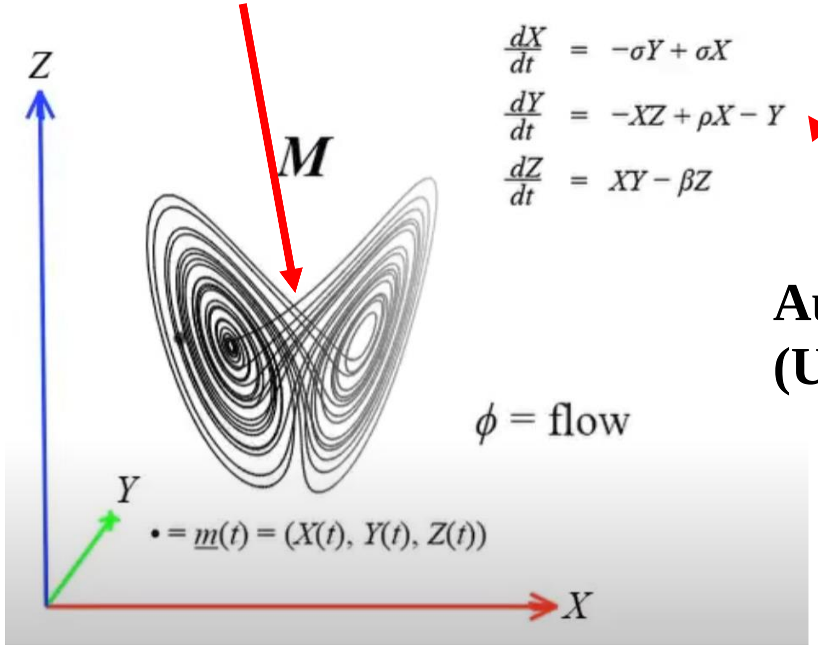
- To get the proper embedding, we decide lagging τ first and then dimension m.

- Embeddings with the same m but different τ are equivalent in the mathematical sense for noise-free data [15]. Therefore, for the purposes of this research, where simulations are conducted without the influence of noise, a simple choice of $\tau = 1$ is sufficient.

Method – FNN

- For embedded dimension m, there are methods including False Nearest Neighbor (FNN) method [21].

Unwanted "crossing" in 2d projection, or a pair of false neighbors



Autonomous system (Uniqueness)

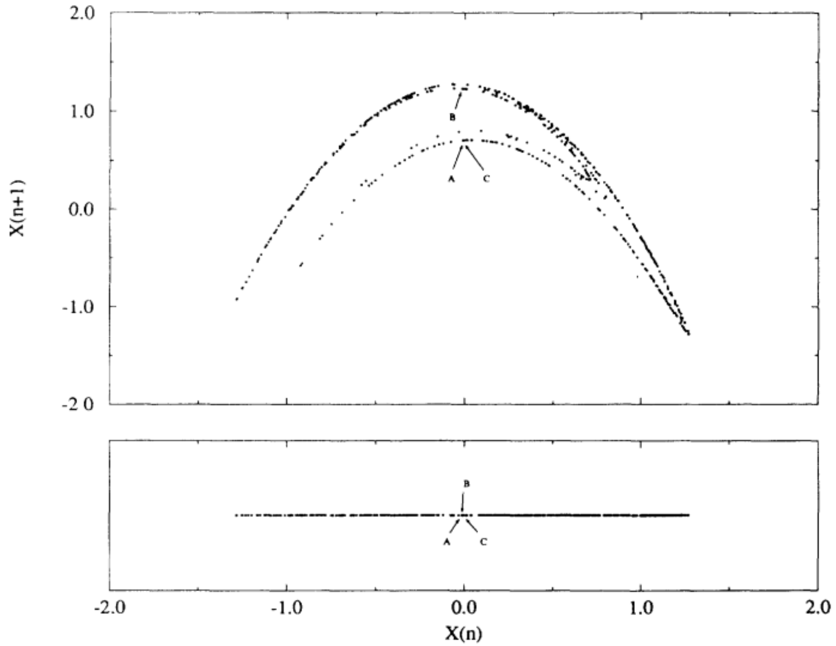


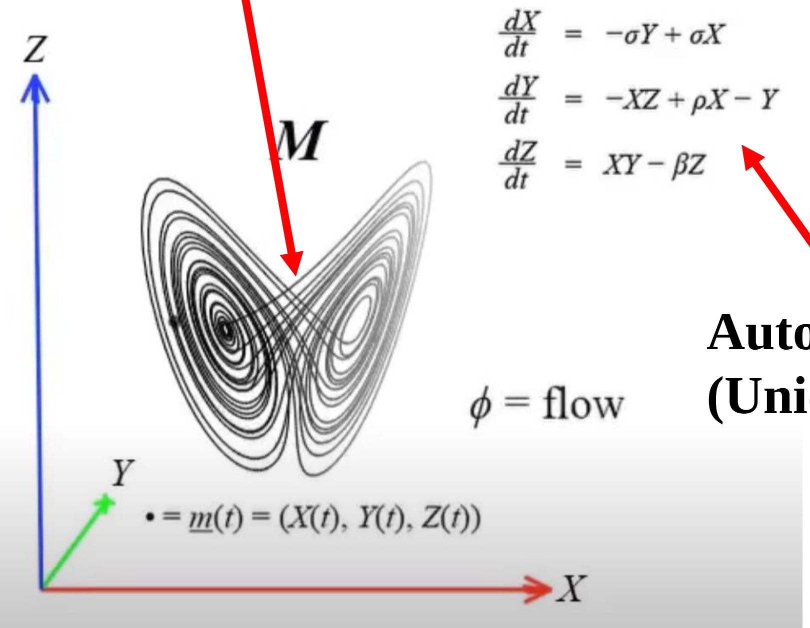
FIG. 1. The R^1 and R^2 embeddings of the x coordinate of the Hénon map of the plane. It is known that for this map $d_E = 2$. The points **A** and **B** are false neighbors while the points **A** and **C** are true neighbors.

Kennel, Matthew B., Reggie Brown, and Henry DI Abarbanel. "Determining embedding dimension for phase-space reconstruction using a geometrical construction." *Physical review A* 45.6 (1992): 3403.

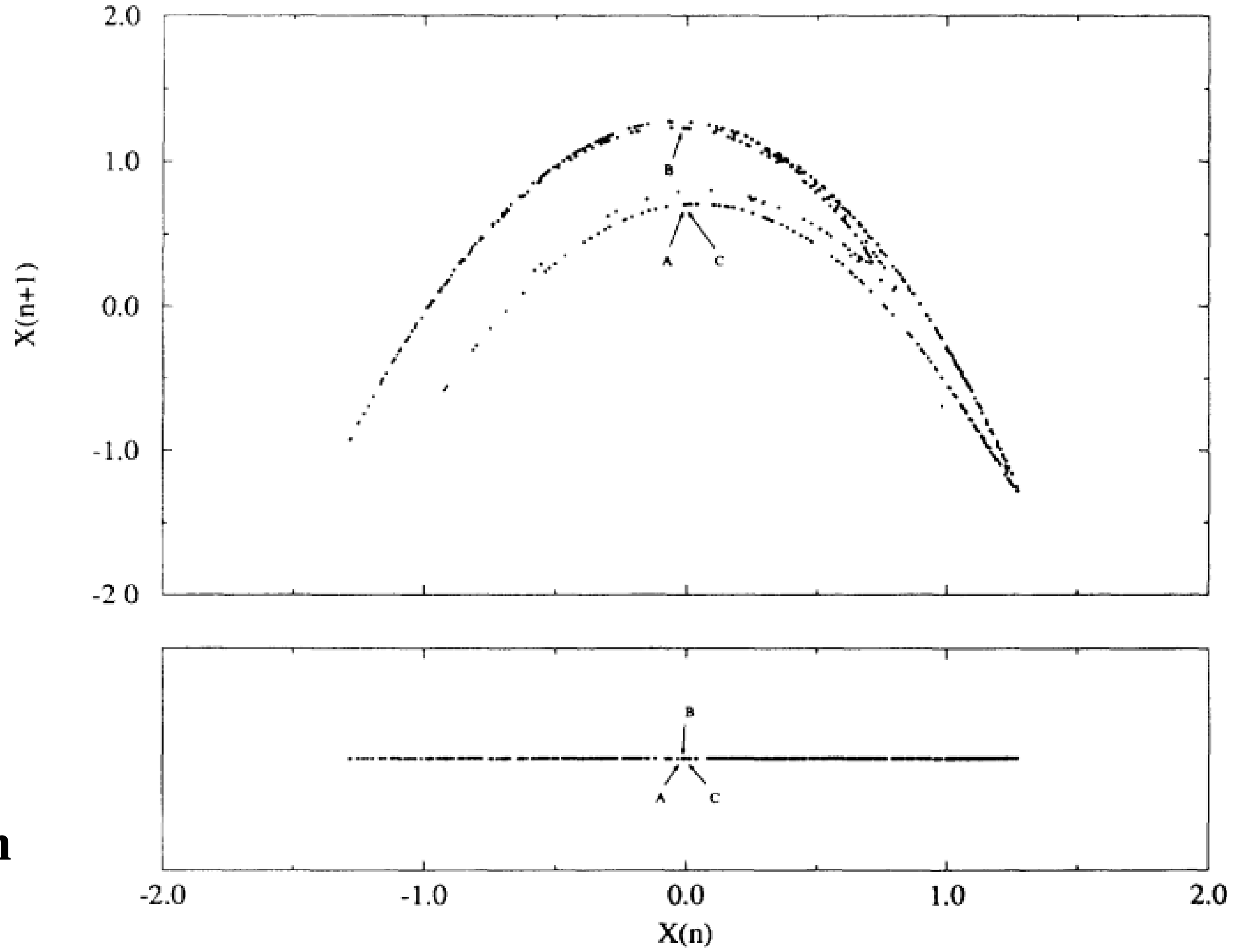
Method – FNN

The optimal dimension is found when percentage of FNN is dropped to a very low value

Unwanted "crossing" in 2d projection, or a pair of false neighbors

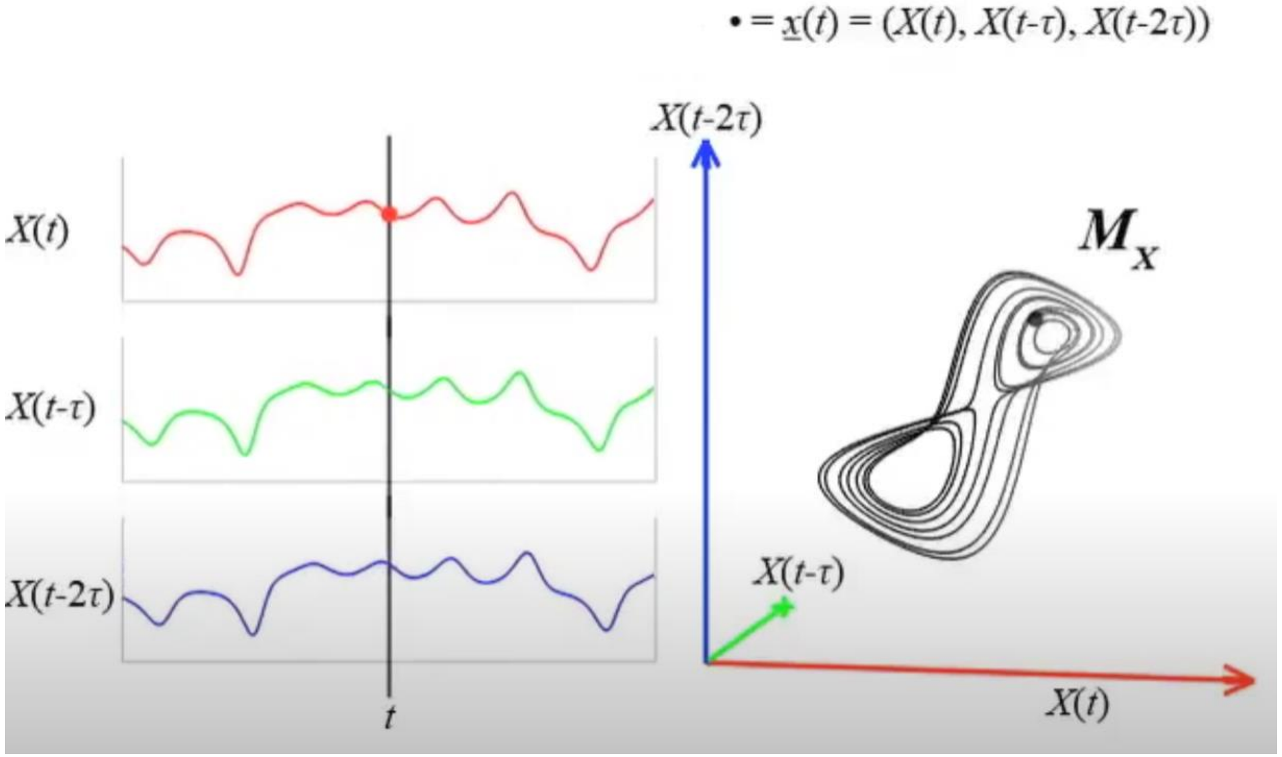


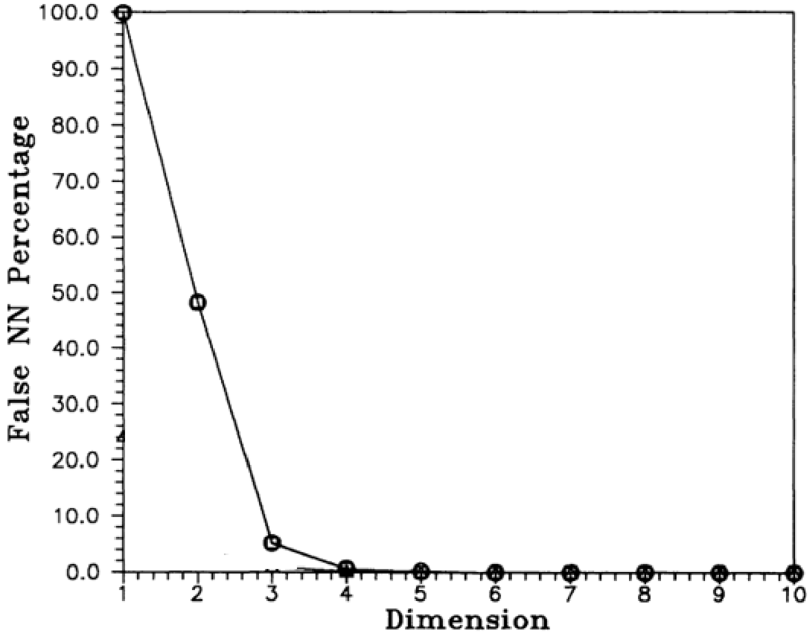
Autonomous system (Uniqueness)



Kennel, Matthew B., Reggie Brown, and Henry DI Abarbanel. "Determining embedding dimension for phase-space reconstruction using a geometrical construction." *Physical review A* 45.6 (1992): 3403

Method – FNN





FNN for Lorentz Attractors

Method – FNN

Uniqueness except for this

Finally, the equations are defined as below [6]:

$$\partial_t u(\boldsymbol{x}, t) = D\nabla^2 u - (I_{\text{fi}}(u, v) + I_{\text{so}}(u) + I_{\text{si}}(V, w))/C_m \quad (1)$$

$$\partial_t v(\boldsymbol{x}, t) = (1 - p)(1 - v)/\tau_v^-(u) - pv/\tau_v^+(u) \quad (2)$$

$$\partial_t w(\boldsymbol{x}, t) = (1 - p)(1 - w)/\tau_w^-(u) - pw/\tau_w^+(u) \quad (3)$$

$$I_{\text{fi}}(u, v) = -vp(u - u_c)(1 - u)/\tau_d \quad (4)$$

$$I_{\text{so}}(u) = u(1 - p)/\tau_0 + p/\tau_r \quad (5)$$

$$I_{\text{si}}(u, w) = -w(1 + tanh(k(u - u_c^{\text{si}})))/(2\tau_{\text{si}}) \quad (6)$$

where:

$$p = \mathcal{H}(u - u_c) \quad (7)$$

$$q = \mathcal{H}(u - u_v) \quad (8)$$

and $\mathcal{H}()$ is Heaviside step function

$$\tau_v^-(u) = \Theta(u - u_v)\tau_{v1}^- + \Theta(u_v - u)\tau_{v2}^- .$$

Method – FNN

- Even with the Laplacian term, there is only 0.01% crossing in true phase space. So, FNN can work for 3V SIM Model with error of O(1e-4)

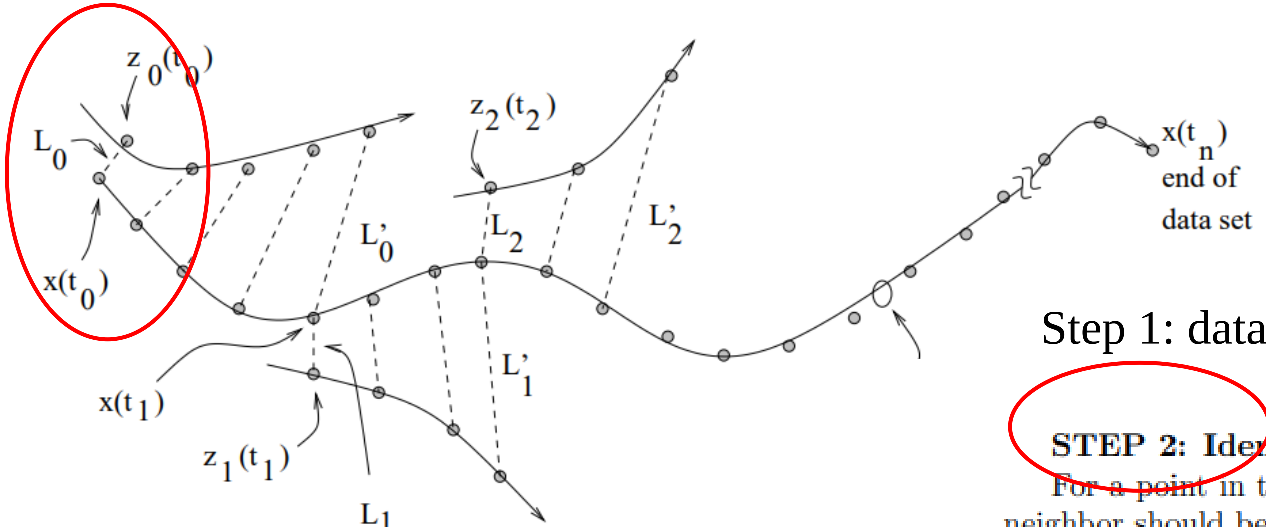
```python
data_length = 10000
crossing_happen_times = 0
for i in range(data_length):
    for j in range(i+1, data_length):
        dis = 0
        dis += (V[0][0][i] - V[0][0][j]) **2
        dis += (V[1][0][i] - V[1][0][j]) **2
        dis += (V[2][0][i] - V[2][0][j]) **2

        if dis < 1e-10:
            dis_plus_1 = (V[0][0][i+1] - V[0][0][j+1]) **2 + (V[1][0][i+1] - V[1][0][j+1]) **2 + (V[2][0][i+1] - V[2][0][j+1]) **2
            if dis_plus_1 > 1e-10:
                print(dis,dis_plus_1,i,j)
                crossing_happen_times += 1

print('data length is: ',data_length)
print('crossing happen times: ',crossing_happen_times)
```

```
9.169554004984093e-11 5.823367832391568e-06 1596 7434
data length is:  10000
crossing happen times:  1
```

Method – Wolf's Algo



https://home.cs.colorado.edu/~lizb/chaos/wolf-notes.pdf

Step 1: data prepare (original phase space or phase space reconstruction)

STEP 2: Identify Nearby Trajectories

For a point in the phase space, find a nearby point (a neighbor) that lies on a different trajectory. This neighbor should be close in space (both their magnitude and direction) but not necessarily in time to avoid correlations between temporally adjacent trajectories.

Specifically, a point $\boldsymbol{X}_i$ where $i \approx \frac{N}{2}$ in first iteration and $i = i'$ otherwise, is chosen, then by iterating through $\boldsymbol{\Gamma}$, we find its nearest neighbor $\boldsymbol{X}_j$ by calculating the Euclidean distance:

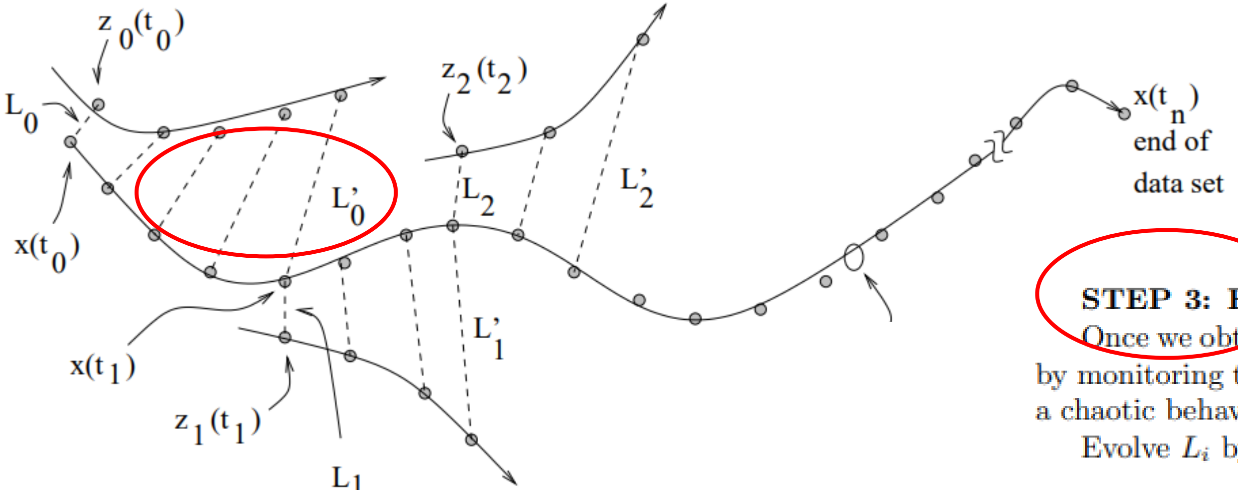
$$j = \arg\min_j \|\boldsymbol{X}_i^{m,\tau} - \boldsymbol{X}_j^{m,\tau}\| = \arg\min_j L_i < \epsilon, \quad (15)$$

where

- $j \in [1, N - m\tau + \tau]$.
- $j \neq i$.
- $\frac{\boldsymbol{X}_i^{m,\tau} \boldsymbol{X}_j^{m,\tau}}{|\boldsymbol{X}_i^{m,\tau}||\boldsymbol{X}_j^{m,\tau}|} < \theta$.
- $\theta = \frac{\pi}{9}$ is the maximum initial angular distance.
- ϵ is the maximum initial separation.

If the algorithm cannot find a close enough pair whose Euclidean distance is smaller than ϵ, it should report to the user and change the ϵ accordingly.

Method – Wolf's Algo



https://home.cs.colorado.edu/~lizb/chaos/wolf-notes.pdf

STEP 3: Evolve and Measure Divergence

Once we obtain a pair of neighbors, following Eq. 10, the finite-time Lyapunov exponent is then computed by monitoring the exponential divergence of the trajectory difference over a certain time interval, indicating a chaotic behavior.

Evolve L_i by one time step each until:

$$\|\boldsymbol{X}_{i'}^{m,\tau} - \boldsymbol{X}_{j'}^{m,\tau}\| = L_{i'} > \epsilon, \tag{16}$$

or

$$i' = N - m\tau + \tau \text{ or } j' = N - m\tau + \tau, \tag{17}$$

where ϵ should be chosen sufficiently large to ensure the two neighbors exhibit chaotic behavior.

If the evolved distance exceeds ϵ, repeat **STEP 2 & 3**. If not, break the loop and continue to **STEP 4**.

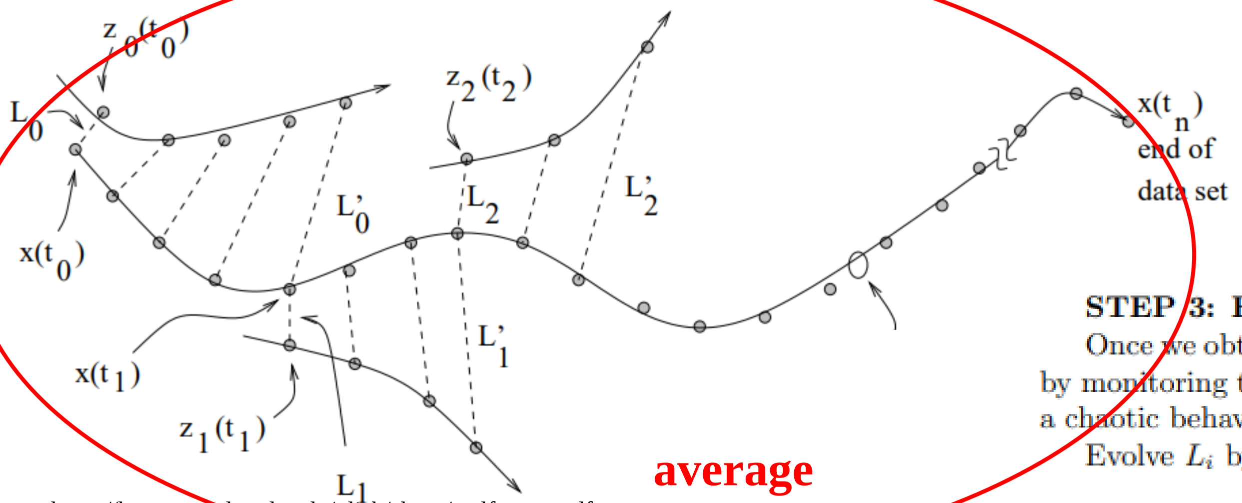
STEP 4: Measure the Lyapunov Exponent [14]

Having obtained multiple finite-time Lyapunov exponents throughout the iterations, the next step is to compute an averaged value, yielding a more robust estimate of the system's Lyapunov exponent.

During the loop, record all the $\frac{L_{i'}}{L_i}$, the i in first loop as i_0 and the i' in last loop as i_f. Finally, the Lyapunov exponent is:

$$\lambda = \frac{1}{i_f - i_0} \sum_{\text{All Loops}} \log_2 \frac{L_{i'}}{L_i}. \tag{18}$$

Method – Wolf's Algo



average

https://home.cs.colorado.edu/~lizb/chaos/wolf-notes.pdf

STEP 3: Evolve and Measure Divergence

Once we obtain a pair of neighbors, following Eq. 10, the finite-time Lyapunov exponent is then computed by monitoring the exponential divergence of the trajectory difference over a certain time interval, indicating a chaotic behavior.

Evolve L_i by one time step each until:

$$\|\boldsymbol{X}_{i'}^{m,\tau} - \boldsymbol{X}_{j'}^{m,\tau}\| = L_{i'} > \epsilon, \tag{16}$$

or

$$i' = N - m\tau + \tau \text{ or } j' = N - m\tau + \tau, \tag{17}$$

where ϵ should be chosen sufficiently large to ensure the two neighbors exhibit chaotic behavior.

If the evolved distance exceeds ϵ, repeat **STEP 2 & 3**. If not, break the loop and continue to **STEP 4**.

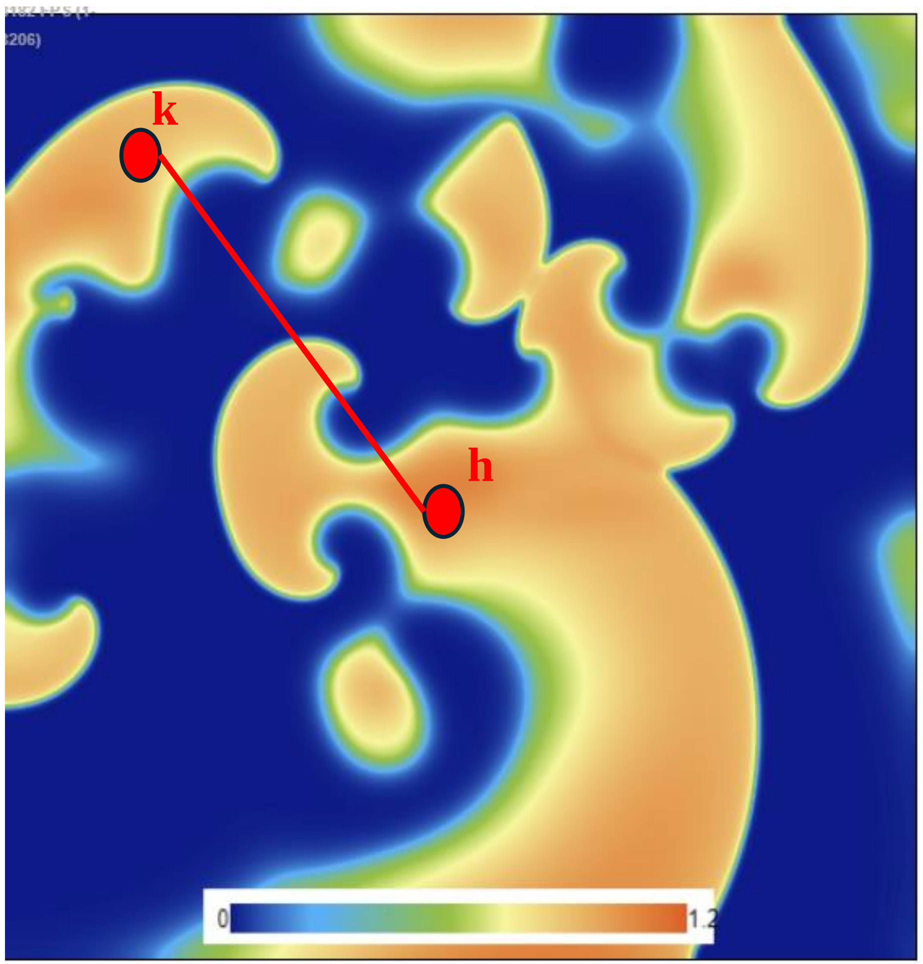
STEP 4: Measure the Lyapunov Exponent [14]

Having obtained multiple finite-time Lyapunov exponents throughout the iterations, the next step is to compute an averaged value, yielding a more robust estimate of the system's Lyapunov exponent.

During the loop, record all the $\frac{L_{i'}}{L_i}$, the i in first loop as i_0 and the i' in last loop as i_f. Finally, the Lyapunov exponent is:

$$\lambda = \frac{1}{i_f - i_0} \sum_{\text{All Loops}} \log_2 \frac{L_{i'}}{L_i}. \tag{18}$$

Method – Spatial Temporal LE (SLE)



Step 1: data prepare (original phase space or phase space reconstruction)

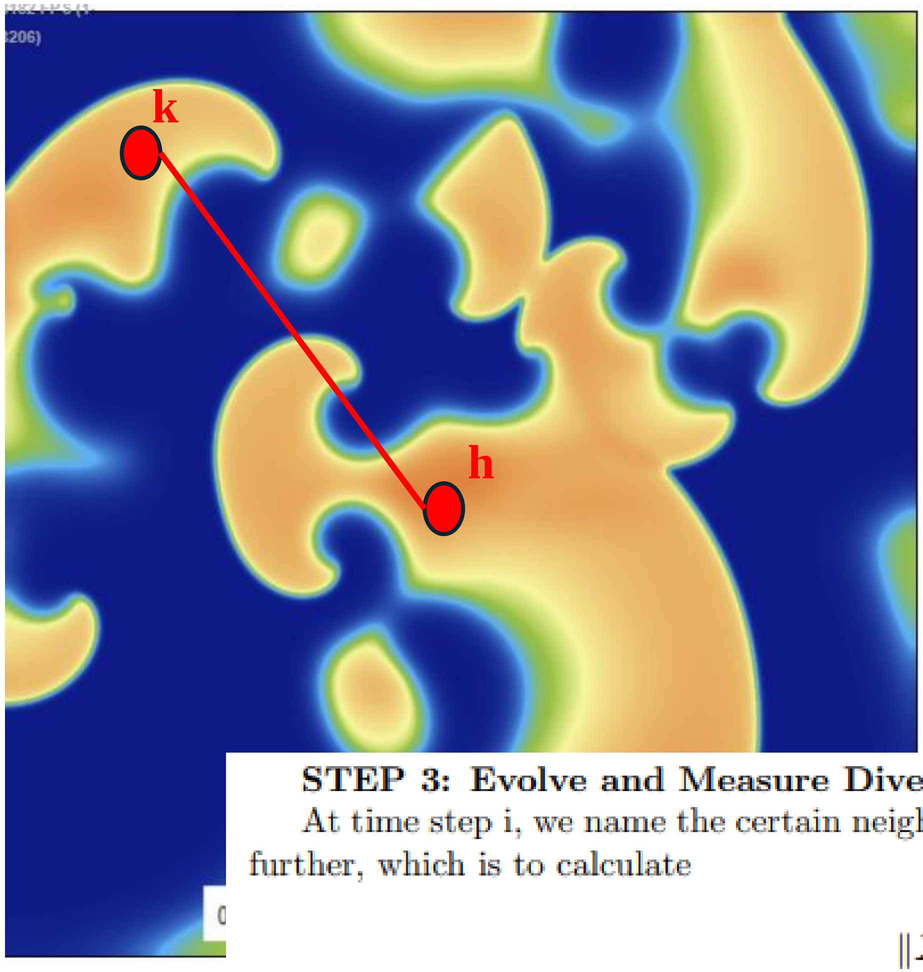
STEP 2: Identify Nearby Trajectories

For each vector $\boldsymbol{X}_i^{m,\tau}(\boldsymbol{k}) \in \boldsymbol{\Gamma}^{m,\tau}(\boldsymbol{k}), \forall \boldsymbol{k} \in \boldsymbol{\Lambda}^2$, we search its neighbor $\boldsymbol{h} \in \boldsymbol{\Lambda}^2, \boldsymbol{h} \neq \boldsymbol{k}$ such that th following condition holds:

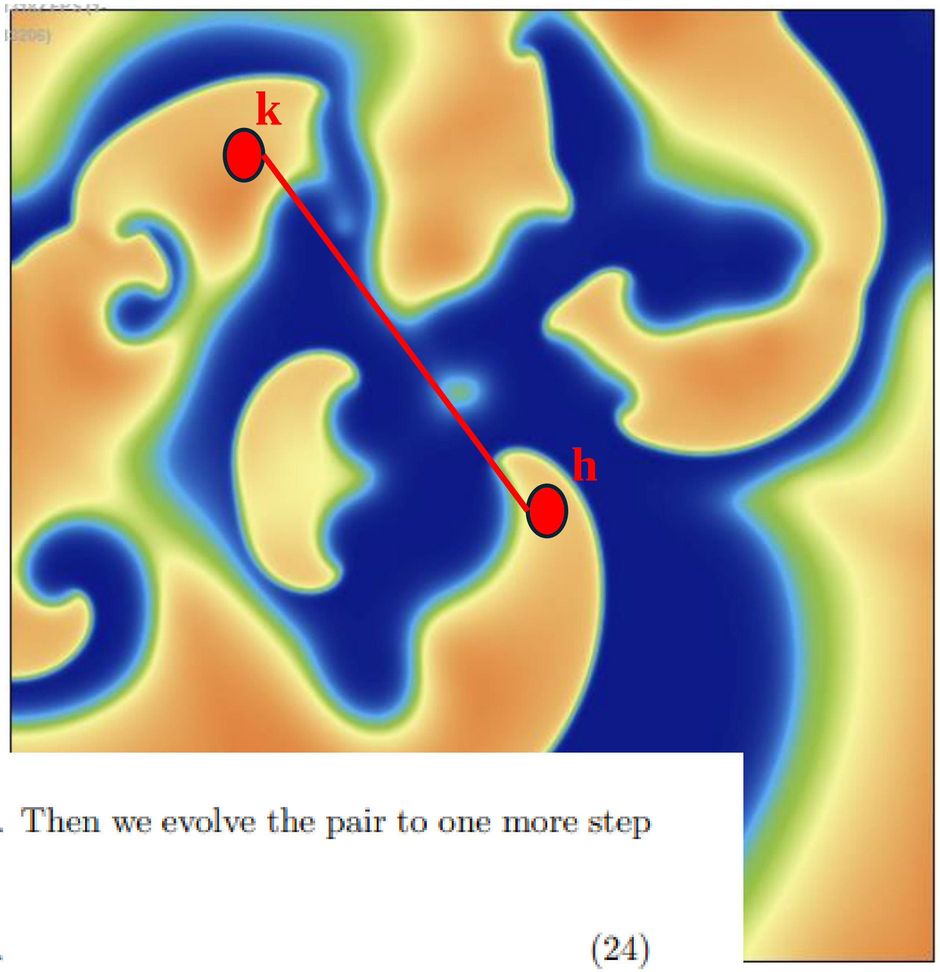
$$\|\boldsymbol{X}_i^{m,\tau}(\boldsymbol{k}) - \boldsymbol{X}_i^{m,\tau}(\boldsymbol{h})\| = \sqrt{\sum_{i'=i}^{i+m\tau-\tau} \left(\boldsymbol{X}_{i'}^{m,\tau}(\boldsymbol{k}) - \boldsymbol{X}_{i'}^{m,\tau}(\boldsymbol{h})\right)^2} < \epsilon, \quad (23$$

where ϵ is the maximum initial separation.

Method – Spatial Temporal LE (SLE)



After duration 1
$\longrightarrow \longrightarrow \longrightarrow \longrightarrow$

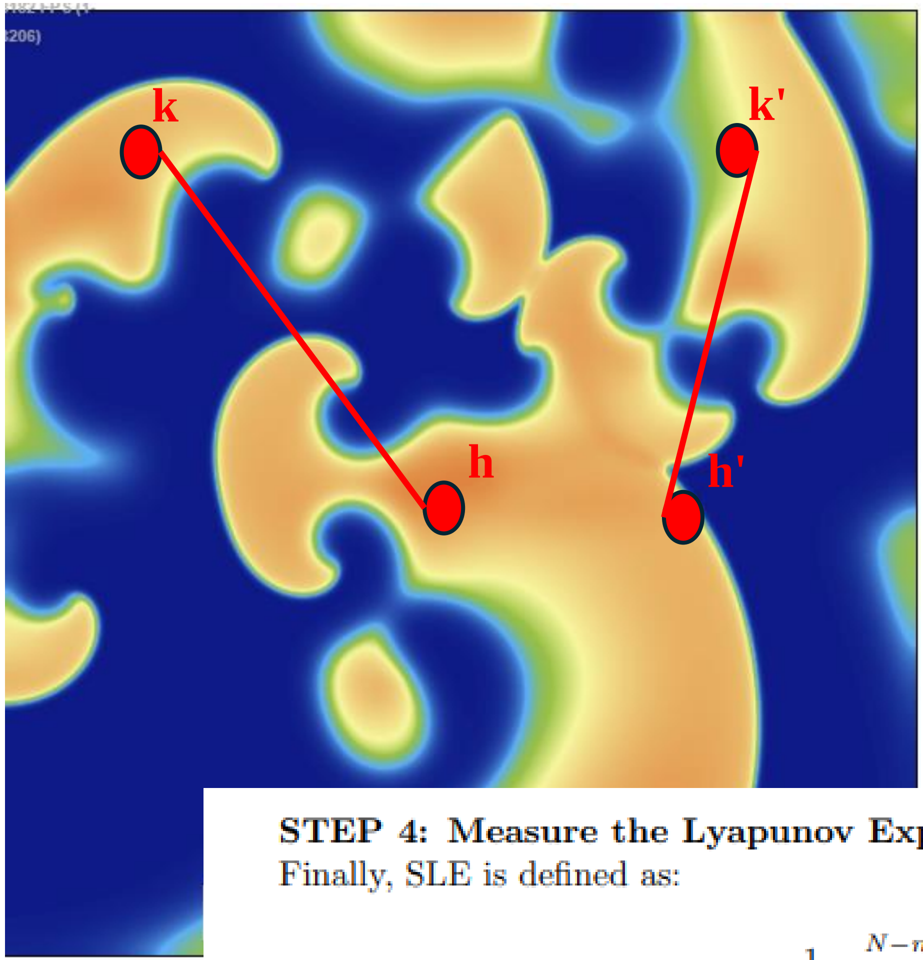


STEP 3: Evolve and Measure Divergence

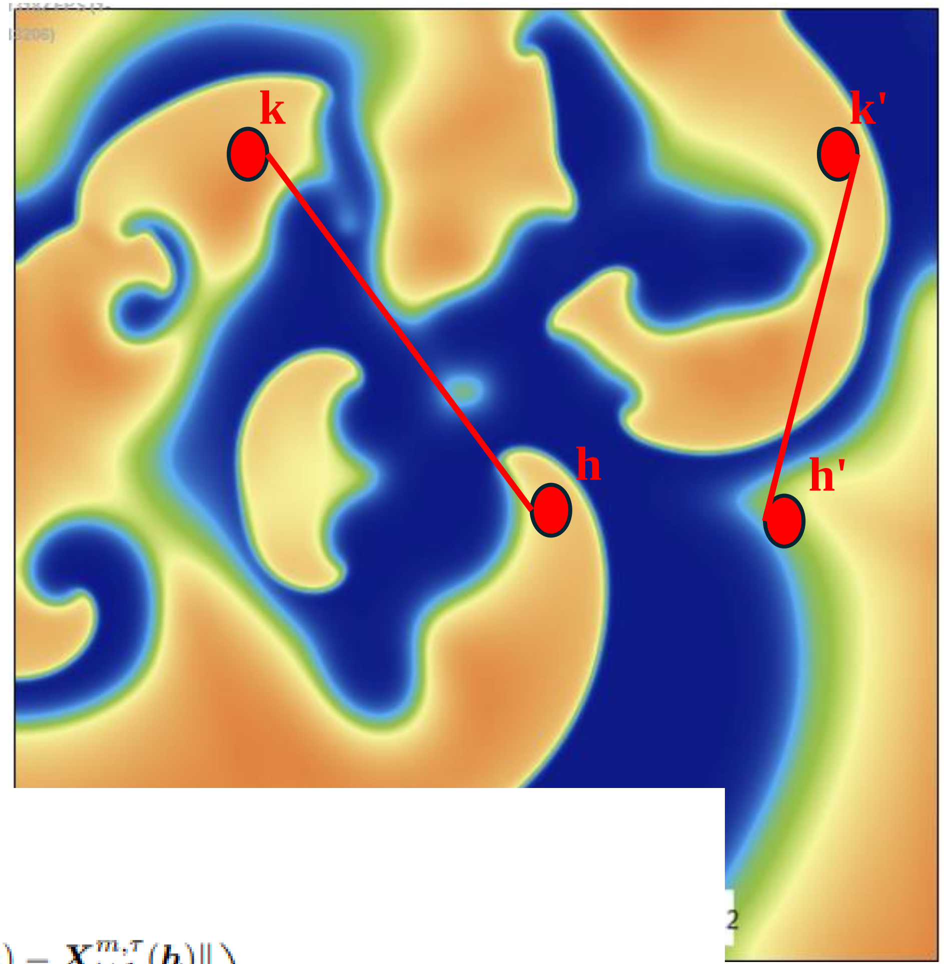
At time step i, we name the certain neighboring pair by $\langle \boldsymbol{k}, \boldsymbol{h} \rangle$. Then we evolve the pair to one more step further, which is to calculate

$$\| \boldsymbol{X}_{i+1}^{m;\tau}(\boldsymbol{k}) - \boldsymbol{X}_{i+1}^{m;\tau}(\boldsymbol{h}) \|. \tag{24}$$

Method – Spatial Temporal LE (SLE)



After duration 1
→ → → →



STEP 4: Measure the Lyapunov Exponent

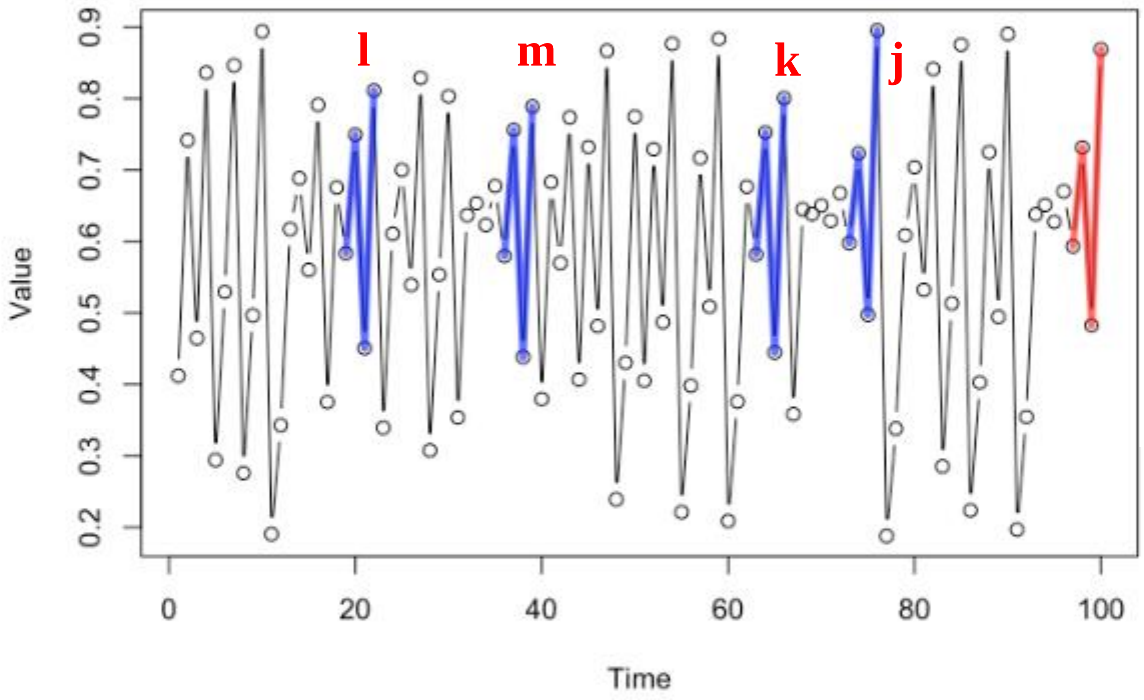
Finally, SLE is defined as:

$$\lambda(m,\tau) = \frac{1}{N_{pair}} \sum_{i=1}^{N-m\tau} \sum_{\langle \boldsymbol{k},\boldsymbol{h} \rangle} \log\left(\frac{\|\boldsymbol{X}_{i+1}^{m,\tau}(\boldsymbol{k}) - \boldsymbol{X}_{i+1}^{m,\tau}(\boldsymbol{h})\|}{\|\boldsymbol{X}_{i}^{m,\tau}(\boldsymbol{k}) - \boldsymbol{X}_{i}^{m,\tau}(\boldsymbol{h})\|} \right). \tag{25}$$

Method – Noise Chaos distinguishment

- Simplex Projection

Embed_dim = 4, tau = 1



Step 1: data prepare (original phase space or phase space reconstruction)

STEP 2: Identify Nearby Trajectories

Then we find 3 nearest neighbors with index $\{j, k, l\}$ to embedded data at step i:

$$j = \arg\min_{j \neq i} \|\boldsymbol{X}_{i-m\tau}^{m,\tau} - \boldsymbol{X}_{j-m\tau}^{m,\tau}\|, j \in [1 + m\tau, N - s],$$

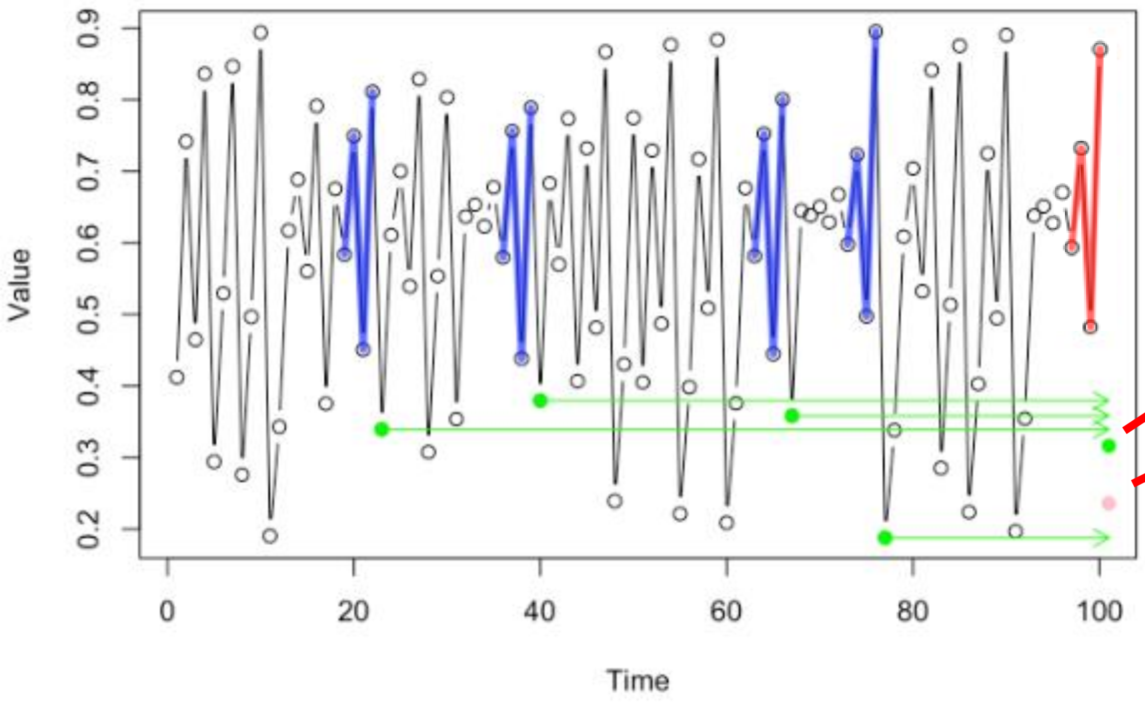
$$k = \arg\min_{k \neq i,j} \|\boldsymbol{X}_{i-m\tau}^{m,\tau} - \boldsymbol{X}_{k-m\tau}^{m,\tau}\|, k \in [1 + m\tau, N - s],$$

$$l = \arg\min_{l \neq i,j,k} \|\boldsymbol{X}_{i-m\tau}^{m,\tau} - \boldsymbol{X}_{l-m\tau}^{m,\tau}\|, l \in [1 + m\tau, N - s].$$

m = ~

Method – Noise Chaos distinguishment

- Simplex Projection



STEP 3: Predict

After that, we predict $\hat{x}_{i+s}$ as:

$$\hat{x}_{i+s} = f(\boldsymbol{X}^{m,\tau}_{i-m\tau}, s) = \frac{\sum_{i'=\{j,k,l\}} \frac{x_{i'+s}}{\|\boldsymbol{X}^{m,\tau}_{i-m\tau} - \boldsymbol{X}^{m,\tau}_{i'-m\tau}\|}}{\sum_{i'=\{j,k,l\}} \frac{1}{\|\boldsymbol{X}^{m,\tau}_{i-m\tau} - \boldsymbol{X}^{m,\tau}_{i'-m\tau}\|}}, \tag{34}$$

where we weigh the neighbors by their reciprocal of the Euclidean distance to $\boldsymbol{X}^{m,\tau}_{i-m\tau}$.

STEP 4: Plot Correlation Coefficient (Prediction Score)

To proceed, we plot $\{X_{i+s}\}$ vs. $\{\hat{X}_{i+s}\}$, as shown in Fig. 5 and get the correlation coefficient (prediction score) as:

$$\rho(s) = \rho_{\{X_{i+s}\},\{\hat{X}_{i+s}\}} = \frac{\mathrm{Cov}\left[\{X_{i+s}\},\{\hat{X}_{i+s}\}\right]}{\sigma_{\{X_{i+s}\}}\sigma_{\{\hat{X}_{i+s}\}}} \tag{35}$$

Finally, by plotting the correlation coefficient $\rho(s)$ vs. s, we could distinguish between chaos and noise, as shown in Fig. 5.

Method – Noise Chaos distinguishment

- Unautocorrelated noise would not be predicted by similar patterns since its next data point is not correlated with current pattern.
- We can expect the chaos has decreasing prediction score with increasing prediction steps. But noise just keep a flat line.

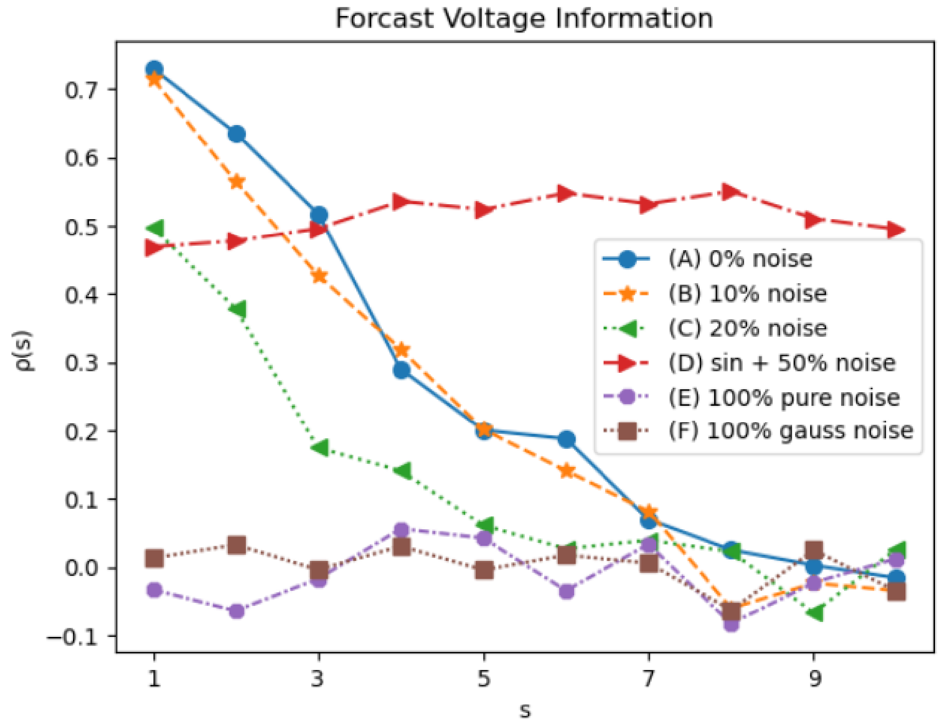
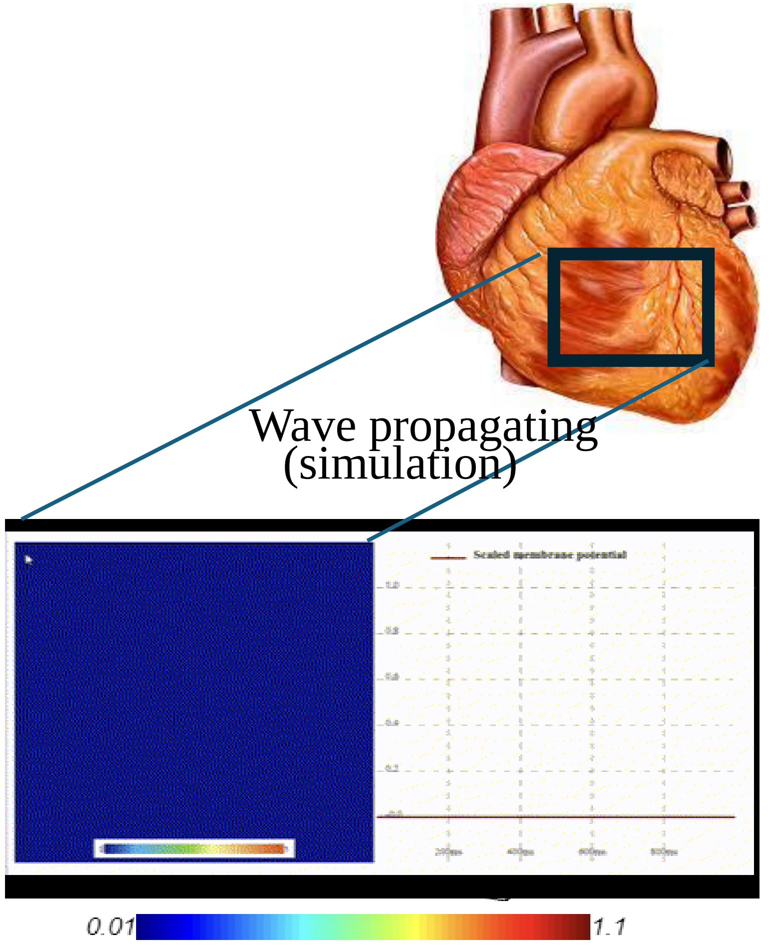


Figure 5: Forcast Plot. $\rho(s)$ is the correlation coefficient (prediction score) and s is the prediction step. More details can be seen in Eq. 35.

Method - Simulation

- We take advantage of fast GPU simulations using webGL.
- https://abubujs.org/ (Dr. Abouzar Kaboudian)
- It can run real-time simulation on PC, tablet and even cell phone.



Parameter	Symbol	Value
Simulation Time Step	$d\tau$	0.1 ms
Measurement Time Step	dt	4 ms
Space Step	dx, dy	$\frac{18}{512}$ cm
Texture Size	\	$512 * 512$ pixels
Measurement Pixels	$\{\boldsymbol{k}\}$	$25 * 25$ pixels
Transient Time	T_t	20000
Measurement Time	ΔT	[20000, 70000], [20000,340000]
# of Voltage per Pixel	$\{u(\boldsymbol{k})\}$	12500, 1250000
# of APD per Pixel	$\{\text{APD}(\boldsymbol{k})\}$	~ 200, ~ 20000

Table 1: Information of the input data. The red color is the input data for Spatial-Temporal Algo, and the blue color is for Wolf's Algo

Current Result and Future Work – FNN

- Now, by plotting the FNN percentage with respect to m, we get optimal m when it no longer decreases

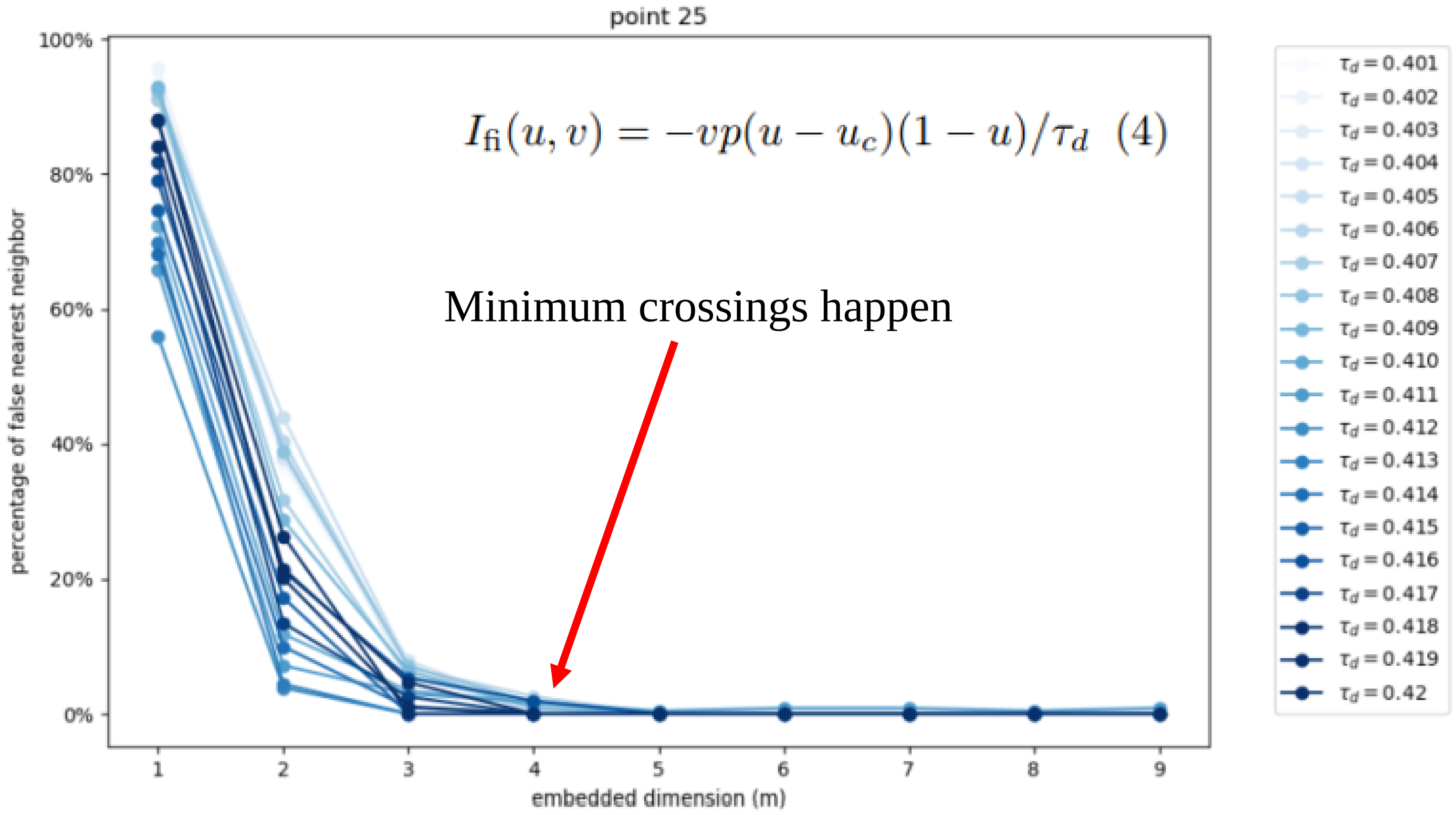


Figure 3: The percentage of false nearest neighbours with input data $x \equiv$ APD, with lag $\tau = 1$.

Current Result and Future Work

- For tau_d, which is the resistance of Na+, I quantified the chaos, which qualitatively match with the simulation map.

- Demo: https://chaos.gatech.edu/eaav6019/files/2D-3V-Model/index.html

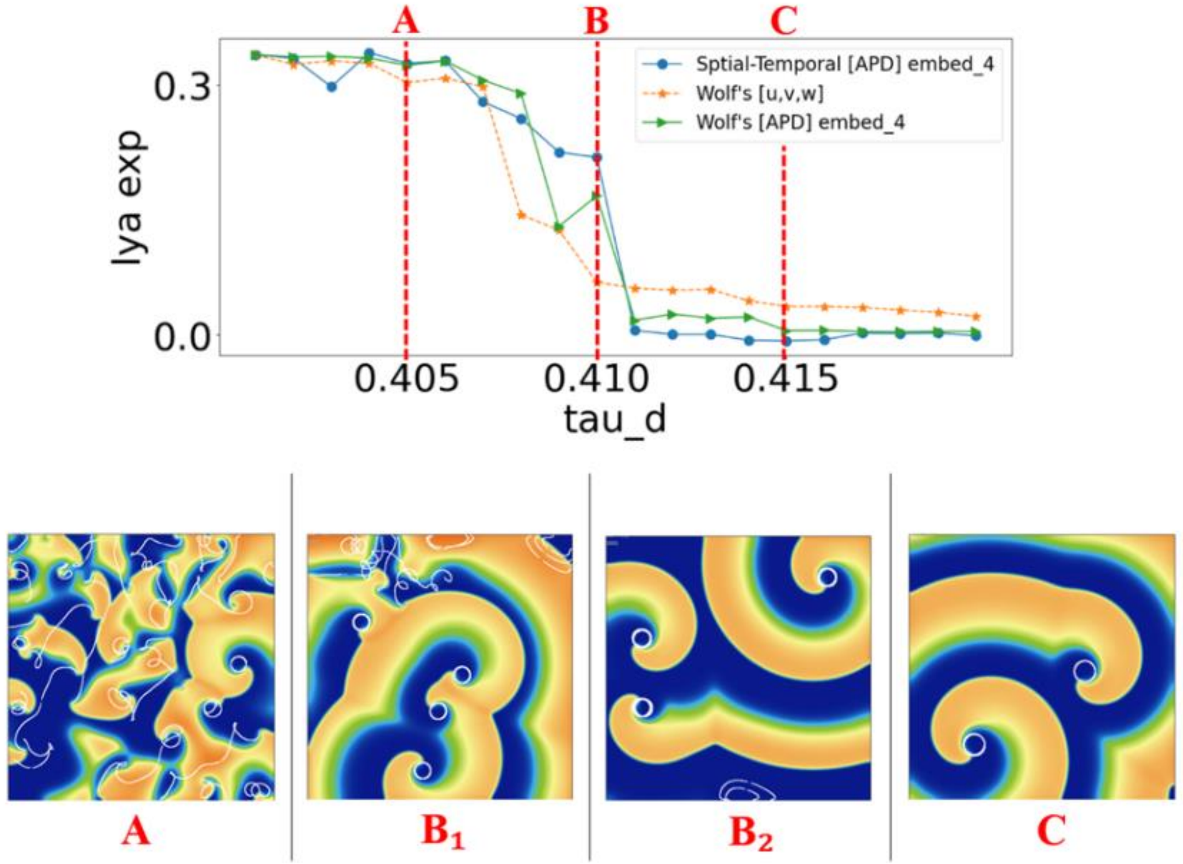


Figure 7: Top: Lyapunov Exponent (scaled) for τ_d where A, B, C represents different tau_d state. Bottom: Actual simulations of different τ_d states. The white line represents the tip trajectory of the spiral wave. B_1 and B_2 represent two possibilities that the B state could become. State A stays chaotic, state B stays either less chaotic or quasiperiodic, and state C stays only periodic.

Current Result and Future Work

- Quantification of more complex models/patterns
- For example, here is the LE for different tip meandering cases.
- Notice here “temporal” used TISEAN (Nonlinear Time Series Analysis)

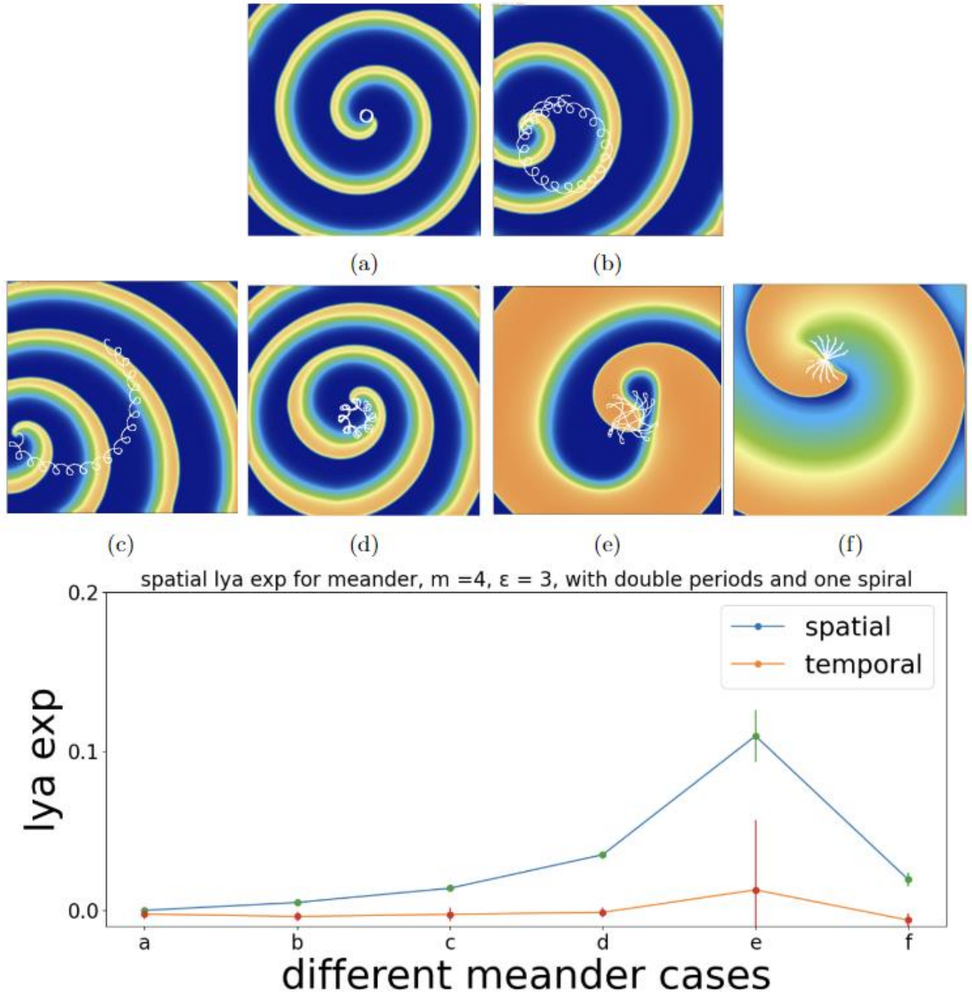
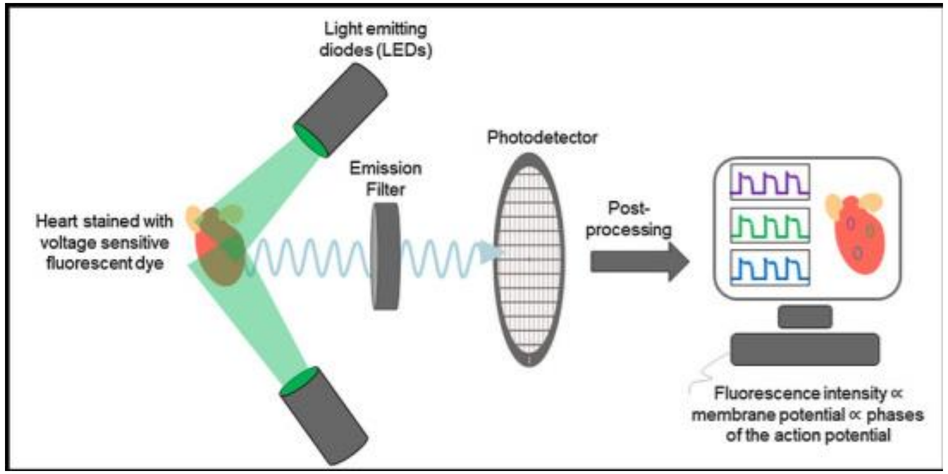
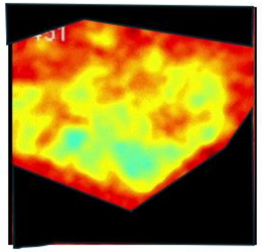


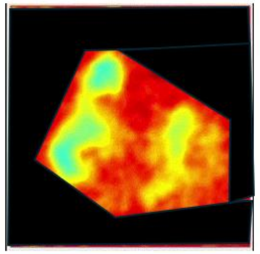
Figure 9: (a) to (e): Different meander cases in 3V SIM Model. [(a): $\tau_d = 0.41$; (b): $\tau_d = 0.392$; (c): $\tau_d = 0.381$; (d): $\tau_d = 0.36$; (e): $\tau_d = 0.25$; (f): parameter set 2[1]].

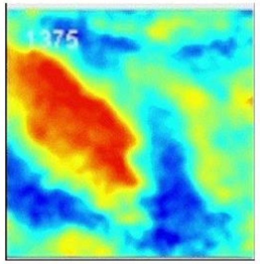
Current Result and Future Work

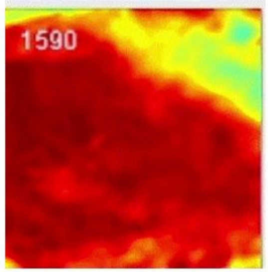
- Quantification of experimental results.
- For example, here is the LE for pig hearts.



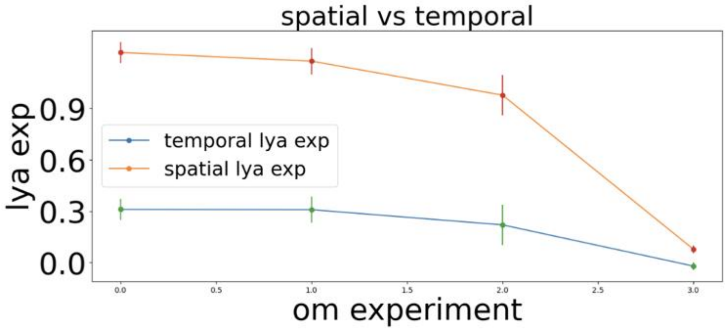

Chaotic, endocardium


Chaotic, epicardium


Less chaotic



Periodic



Conclusion

- The 3V SIM Model is a simplified cardiac model, retaining essential activation and inactivation characteristics while having fewer variables.
- I showed that APD data could be alternative choice for determining the Lyapunov exponent.
- What’s more, integrating spatial information with the Spatial-Temporal Algorithm could significantly reduce the amount of APD data needed, enabling quicker and even real-time determination of the Lyapunov exponent.
- It can help drug development by showing which particular region of parameters are sensitive and likely to induce chaotic behavior.

Thanks to CHAOS Lab!

- Current Members:
  - Casey
  - Evan
  - Henry
  - Jimena
  - Lynn
  - Mikael
  - Mikhail
  - Will
  - Flavio (Adviser)

Any Question?