Identifying spiral wave tips with reservoir computing

Will An

Journal Club

Introduction

- This paper used reservoir computing method (RC) to determine spiral wave tips from simulation results, compared with Jacobian determinant method (JM), which is considered as “ground truth” result.
- RC is a simplified Recurrent Neural Network (RNN), typically used in predicting chaos system by modeling its nonlinear property.
- For example, RC can predict Y,Z given X in Lorentz system.
- Or to predict {**X**(t)} based on {**X₀**}

$$
\begin{aligned}
\frac{dX}{dt} &= -\sigma Y + \sigma X \\
\frac{dY}{dt} &= -XZ + \rho X - Y \\
\frac{dZ}{dt} &= XY - \beta Z
\end{aligned}
$$

Introduction – Why RC?

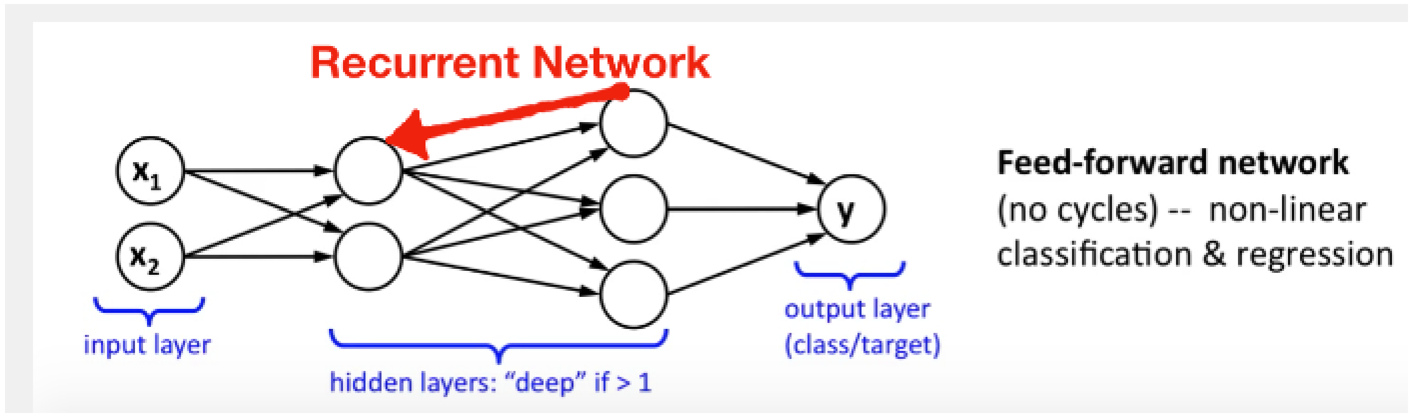
- Usual method requires data from a subsequent time interval for calculating the tip position at a given moment.

- RC could identify the spatial tips only with one time slice (one frame).

Introduction – Recurrent Network

- Usually, neural network uses the hidden layer (layers in the middle) in forward direction, meaning there will not be any loop back.

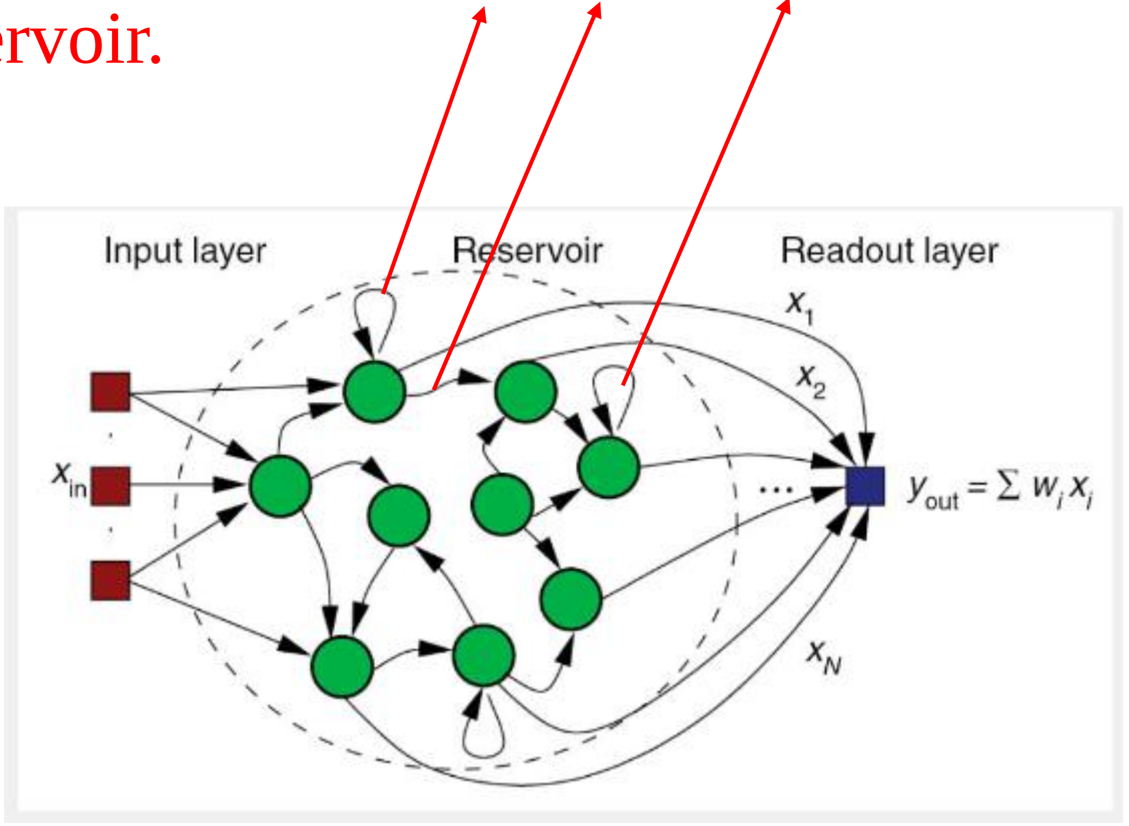
- But in RNN or RC, it goes back and forth to reuse the neuron in the hidden layers.


Recurrent Network
x_1
x_2
y
input layer
hidden layers: "deep" if > 1
output layer (class/target)
Feed-forward network
(no cycles) -- non-linear classification & regression

http://idle.systems/posts/reservoir_computing.html

Introduction – RC

- Now, for RC, its neurons are sparsely interconnected in the hidden layer or the reservoir.

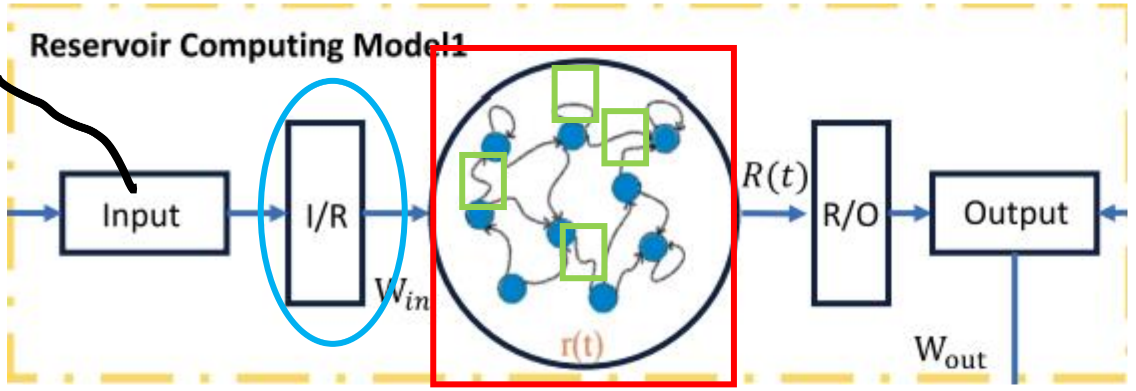


Method – RC

Reservoir update function:

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha\tanh\left(\mathbf{A}(t) + \mathbf{W}_{\mathrm{in}}(\mathbf{V}_{\mathrm{S}}(t+\Delta t)+\xi)\right)$$

$$\boxed{\underset{\mathbf{r}_{N,1}(\Delta t)}{\begin{pmatrix} n_1\\ \vdots \\ n_N\end{pmatrix}}} = (1-\alpha)\boxed{\underset{\mathbf{r}_{N,1}(0)}{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}} + \alpha\tanh\left\{ \boxed{\underset{\mathbf{A}_{N,N}}{\begin{pmatrix} a_1 & \cdots & a_2\\ \vdots & \ddots & \vdots \\ a_3 & \cdots & a_4\end{pmatrix}}}\boxed{\underset{\mathbf{r}_{N,1}(0)}{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}} + \underset{W_{\mathrm{in}\,N,M}}{\begin{pmatrix}w_1\\ \vdots\\ w_N\end{pmatrix}} \underset{\mathbf{V}_{\mathrm{S}\,M,1}(\Delta t)}{\begin{pmatrix}u_1\\ \vdots\\ u_M\end{pmatrix}} + bias \right\}$$



where:

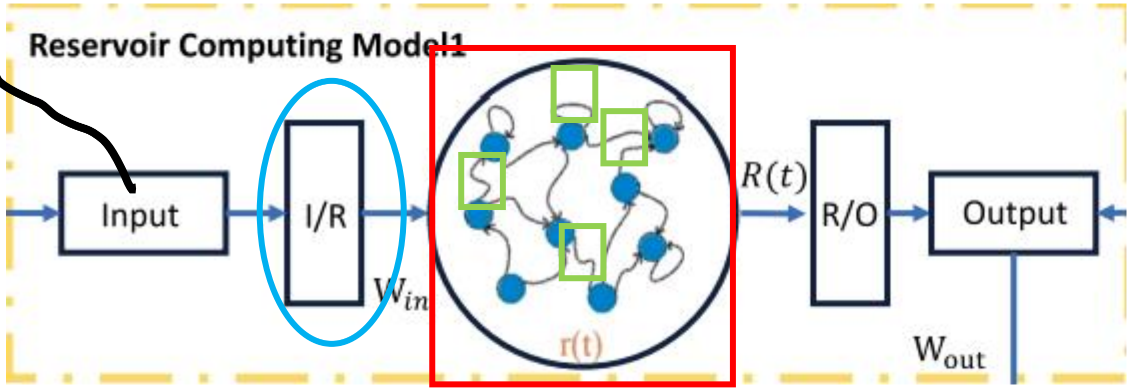
- $\mathbf{r}$ is the reservoir state vector.
- α is the updating rate for reservoir $\mathbf{r}$.
- $\mathbf{A}$ is the reservoir layer's weighted adjacency matrix.
- $\mathbf{W}_{\mathrm{in}}$ is the weight matrix between the input layer and the reservoir layer.
- $\mathbf{V}_{\mathrm{S}}(t)$ is the input singal at time t.
- ξ is the bias.
- N is the number of neurons/nodes in reservoir.
- M is the length (how many pixels) of the input data.

Method – RC

Reservoir update function:

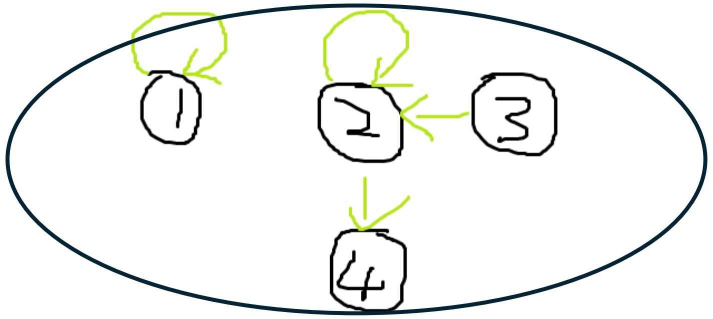
$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha\tanh\left(\mathbf{A}(t) + \mathbf{W}_{\text{in}}(\mathbf{V}_{\text{S}}(t+\Delta t)+\xi)\right)$$

$$\boxed{\underbrace{\begin{pmatrix} n_1\\ \vdots \\ n_N\end{pmatrix}}_{\mathbf{r}_{N,1}(\Delta t)}} = (1-\alpha)\boxed{\underbrace{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}_{\mathbf{r}_{N,1}(0)}} + \alpha\tanh\left\{ \boxed{\underbrace{\begin{pmatrix} a_1 & \cdots & a_2\\ \vdots & \ddots & \vdots \\ a_3 & \cdots & a_4\end{pmatrix}}_{\mathbf{A}_{N,N}}} \boxed{\underbrace{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}_{\mathbf{r}_{N,1}(0)}} + \underbrace{\begin{pmatrix}w_1\\ \vdots\\ w_N\end{pmatrix}}_{W_{\text{in}\,N,M}} \underbrace{\begin{pmatrix}u_1\\ \vdots\\ u_M\end{pmatrix}}_{V_{\text{S}\,M,1}(\Delta t)} + bias\right\}$$



W_in is randomly generated between [-1,1]: $\left\{\begin{matrix} -0.5 & 0.3 & 0.7\\ 0.2 & 0.4 & -0.6\\ 0.1 & -0.9 & -1.0\\ 0.6 & -0.4 & -0.7\end{matrix}\right\}$

A is sparse matrix generated by sparse random Erdős–Rényi method: $\left\{\begin{matrix} -0.5 & 0 & 0 & 0\\ 0 & 0.4 & -0.5 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0.2 & 0 & 0\end{matrix}\right\}$



So, before Reservoir to Output (R/O) we get everything random.

Essentially, it only learns W_out.

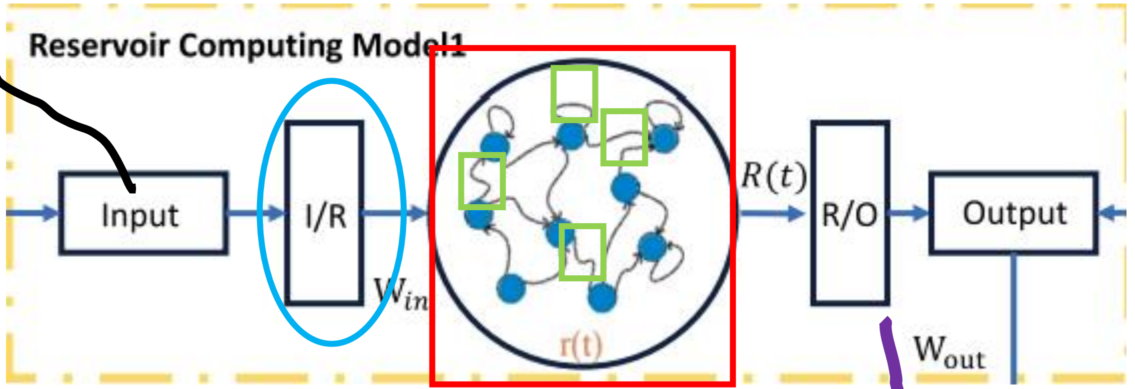
That makes it simplified and computationally cheap.

Method – RC

Reservoir update function:

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha\tanh\left(\mathbf{A}(t) + \mathbf{W}_{\text{in}}(\mathbf{V}_{\text{S}}(t+\Delta t)+\xi)\right)$$

$$\boxed{\underset{\mathbf{r}_{N,1}(\Delta t)}{\begin{pmatrix} n_1\\ \vdots \\ n_N\end{pmatrix}}} = (1-\alpha)\boxed{\underset{\mathbf{r}_{N,1}(0)}{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}} + \alpha\tanh\left\{ \boxed{\underset{\mathbf{A}_{N,N}}{\begin{pmatrix} a_1 & \cdots & a_2\\ \vdots & \ddots & \vdots \\ a_3 & \cdots & a_4\end{pmatrix}}}\boxed{\underset{\mathbf{r}_{N,1}(0)}{\begin{pmatrix}0\\ \vdots\\ 0\end{pmatrix}}} + \underset{W_{\text{in}\,N,M}}{\begin{pmatrix}w_1\\ \vdots\\ w_N\end{pmatrix}}\underset{\mathbf{V}_{\text{S}\,M,1}(\Delta t)}{\begin{pmatrix}u_1\\ \vdots\\ u_M\end{pmatrix}} + bias\right\}$$



It iterates to get r(0)…r(t)

Then it defines the output as: $\mathbf{R}(t) = [\mathbf{r}(t); \mathbf{V}_S(t); \xi]$ $\quad \hat{\mathbf{V}}_L(t) = \mathbf{W}_{out}\mathbf{R}(t).$

$\hat{V}$_L(t) is the prediction result (tip location) from RC, V_L(t) is the 'ground truth' tip location from JM.

It learns W_out based on minimize loss function between prediction and truth using ridge regression.

Method – RC - Ridge regression

$$\hat{\mathbf{V}}_L(t) = \mathbf{W}_{out}\mathbf{R}(t).$$

- The loss function is: $\mathcal{L}(W_{\text{out}}) = \|V_L - RW_{\text{out}}^T\|^2 + \eta\|W_{\text{out}}\|^2$
  - L2 term |W|^2 is added to prevent overfitting by punishing large W element.

- After some simplification, it gets W_out.

$$\mathcal{L}(W_{\text{out}}) = \|V_L - RW_{\text{out}}^T\|^2 + \eta\|W_{\text{out}}\|^2$$

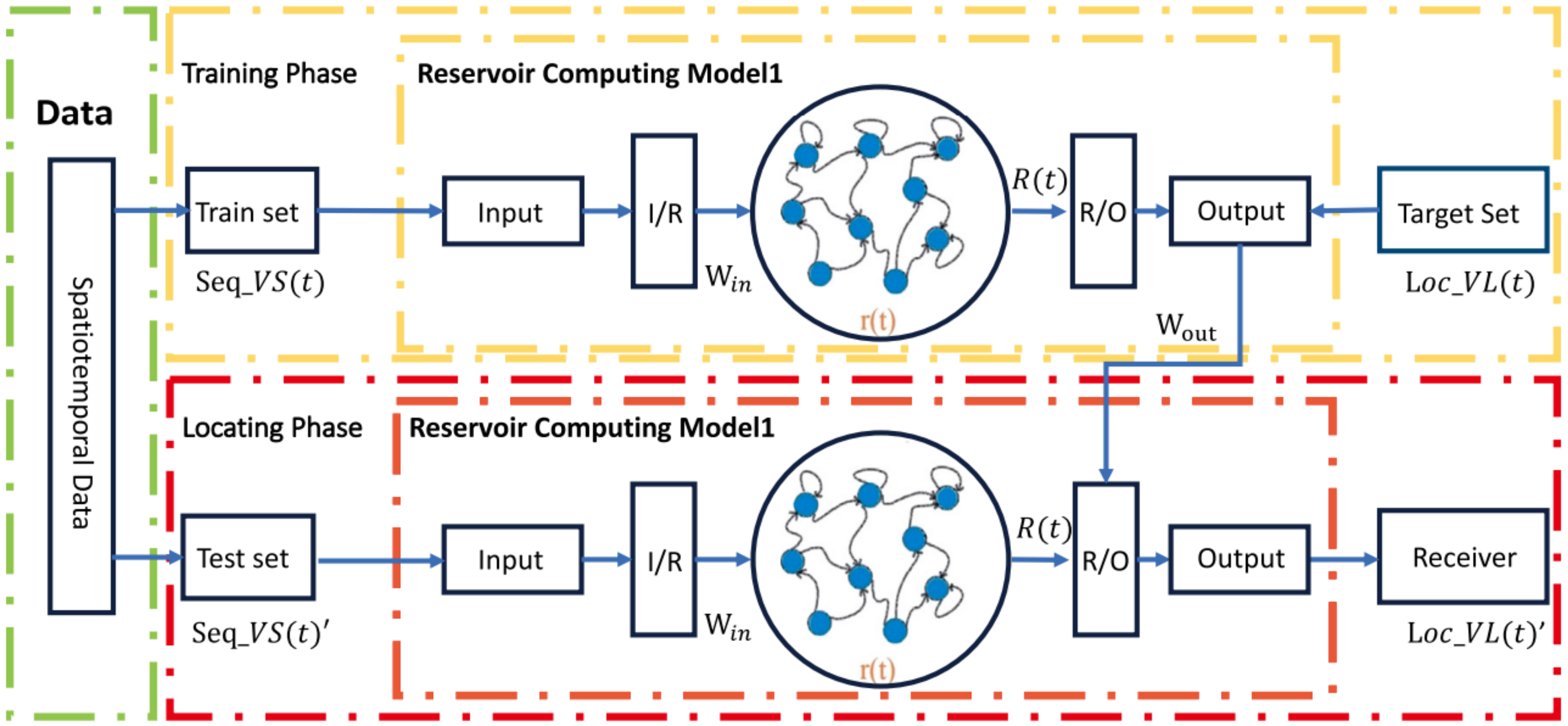
$$\frac{\partial \mathcal{L}}{\partial W_{\text{out}}} = -2R^T V_L + 2(R^T R + \eta I)W_{\text{out}}^T = 0$$

$$R^T V_L = (R^T R + \eta I)W_{\text{out}}^T$$

$$\mathbf{W}_{out} = \mathbf{V}_L(t)\mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \eta\mathbf{I}),$$

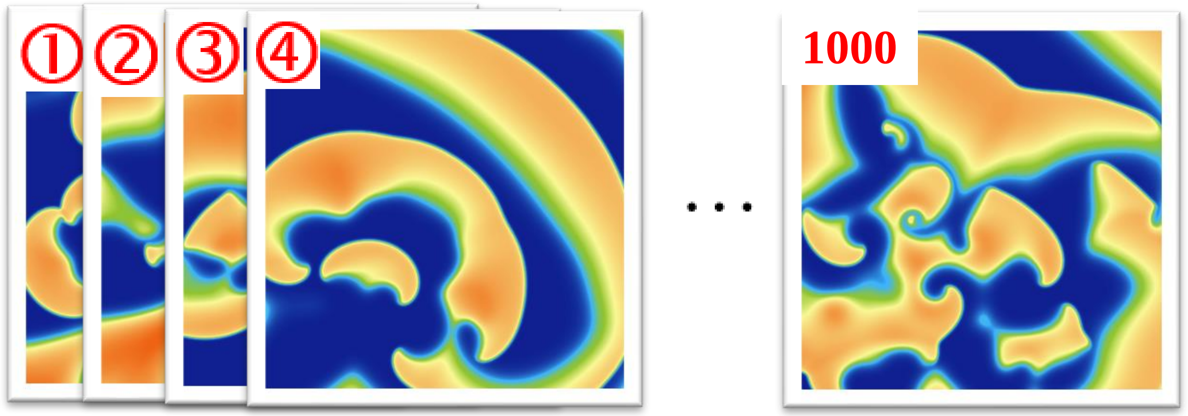
Method – RC

- Now, with same W_in, reservoir setup, and learned W_out, it test the data V_S’.



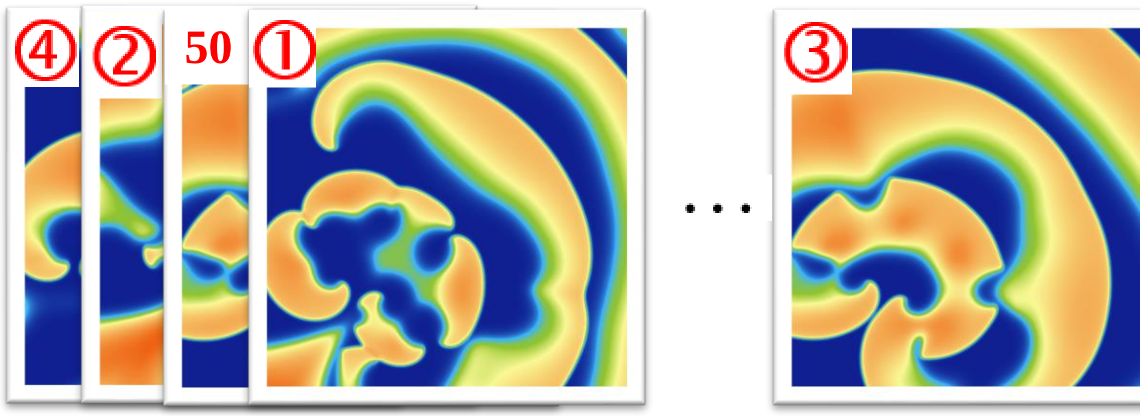
Method – Training (1000 frames)

Robust under case of fast change of tips number



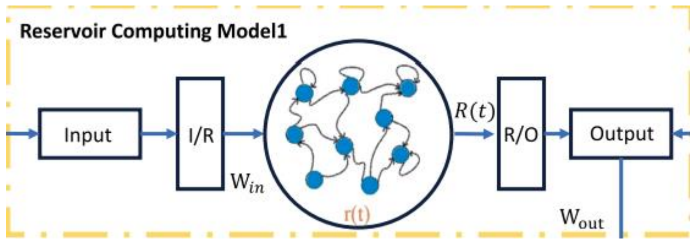
. . .

Shuffle



. . .

Jacobian Determinant Method



Ridge Regression

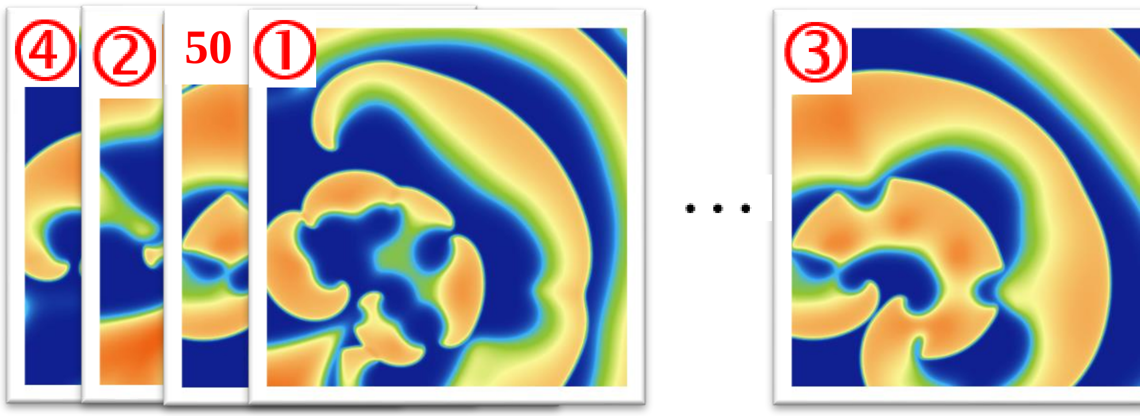
Train

W_out

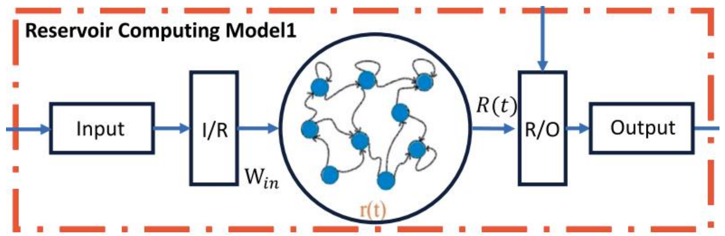
Method – Testing (100 frames)

W_out

Shuffle



W_out



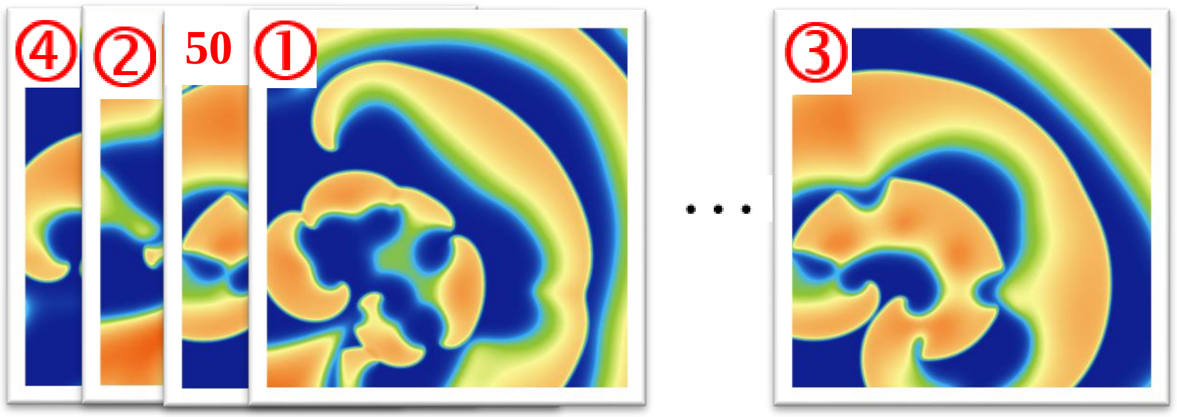
Test

Jacobian Determinant Method

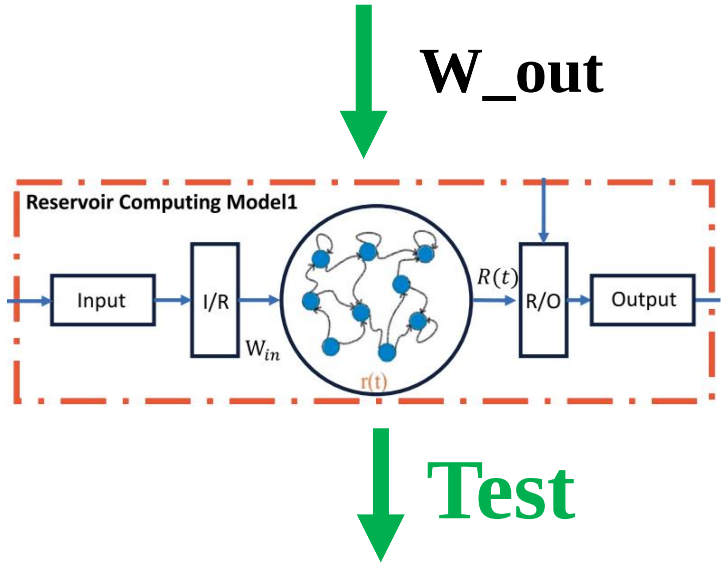
Get Accuracy & Error

Output Data

Method – Output Data



- Output Data: $(0, 0.8, 0.1, \cdots, 1.0)^T$
  - Value from 0 to 1 means the probability to be the tip.
- Select the tip when the non-zero (and >0.5) element is a local max.



Output Data $\longrightarrow (0, 0.8, 0.1, \cdots, 1.0)^T \longrightarrow (0, 1, 0, \cdots, 1)^T$

Method – Performance Algorithm

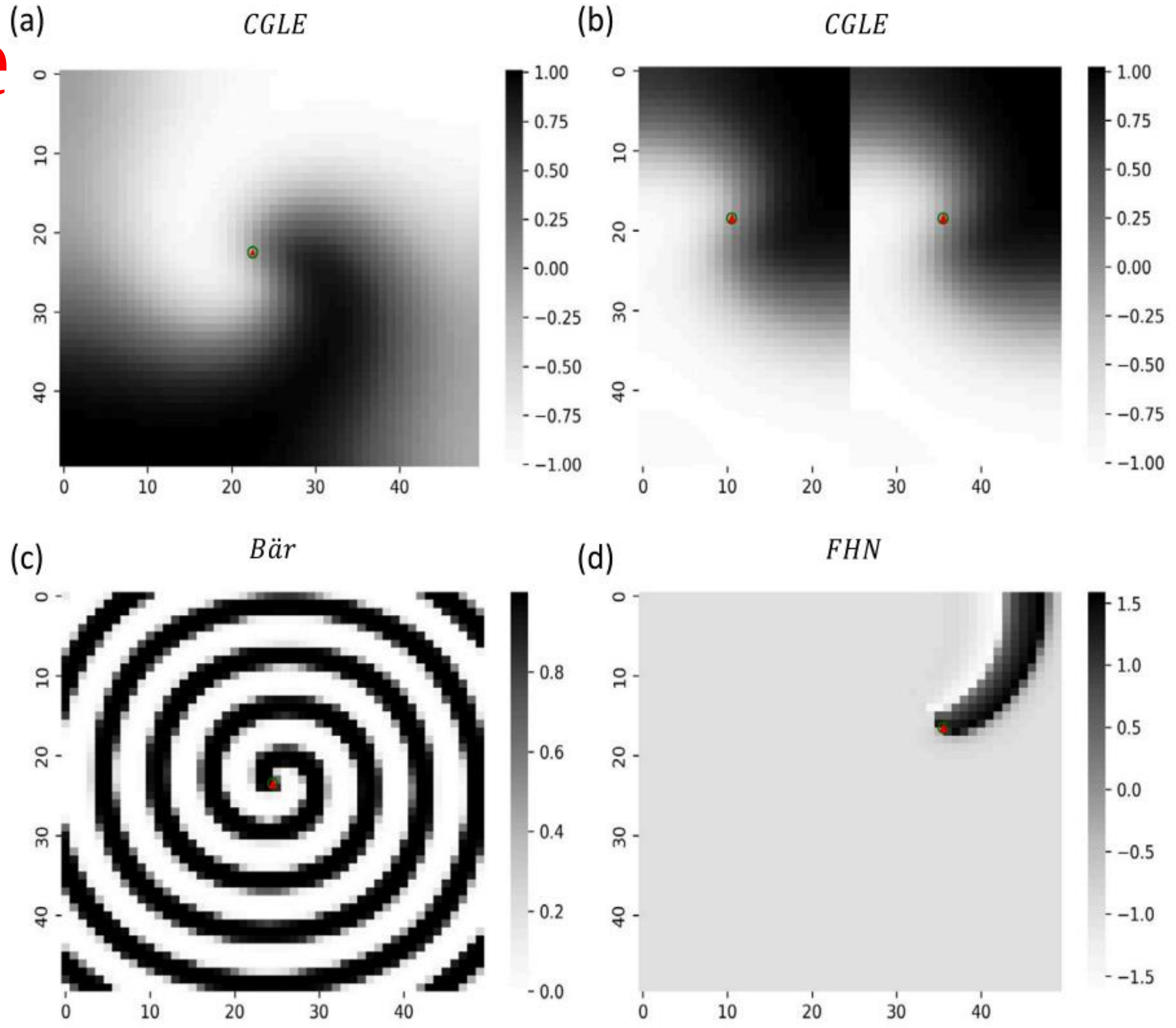
- Accuracy, AC(qualification):
- It is the percentage of correct tips by RC.
- Tips in a spatial range of ±1 pixel would be considered as correct.

- Distance Error, DE (quantification):
- It measures the numerical distance error between tip pixels from RC and from JM.

Result – One Tip

- It testify the RC on different Reaction-Diffusion Model (CGLE, Bär, FHN)
  - the complex Ginzburg–Landau equation, BAR model, FitzHugh–Nagumo model.

Result – One Tip



RC ▲

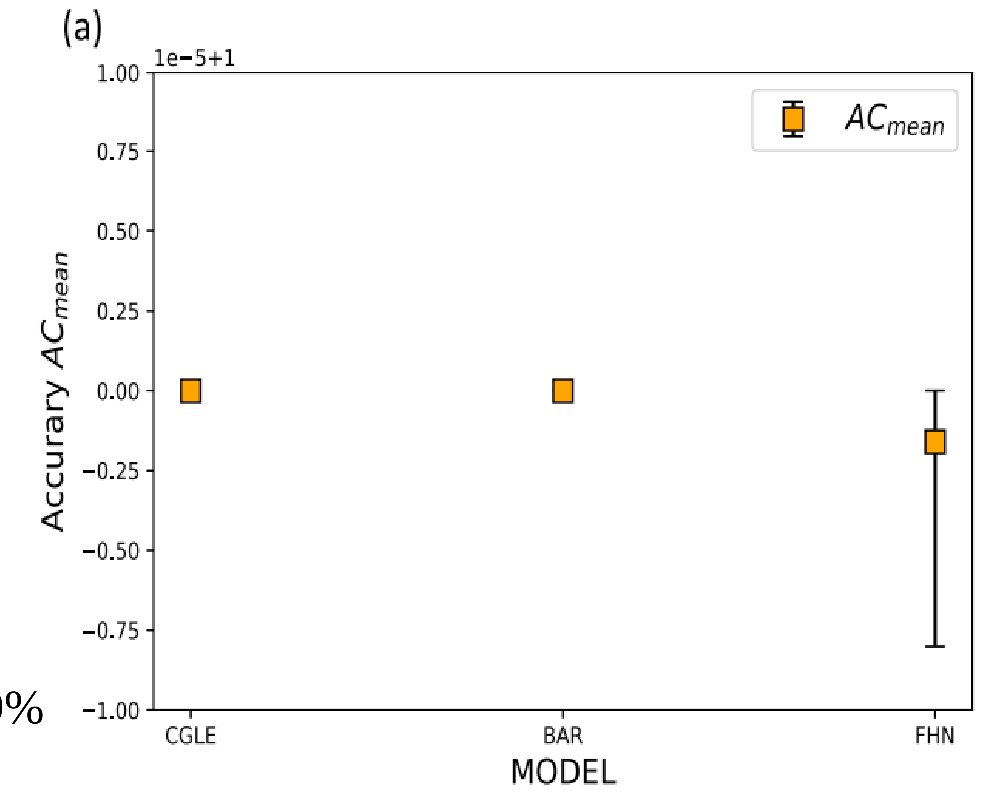
JM ○

Fig. 2. A comparison of spiral wave tips identified by Jacobian determinant method (JM) and RC for CGLE, Bär and FHN are displayed ((a–d), respectively). Here, the corresponding tips calculated by JM are shown by green hollow circles, and tips calculated by RC are shown by red solid triangles. Subplot (a) shows the result of the one-tip mode, where the tip is chosen by the non-zero element which is the local maximum value. The results of two-tip case within the CGLE system are shown in subplot (b). Subplot (c) displays the results of the tip rotating around the center under the Bär system. The computed results of spiral chimeras under the FHN are shown in subplot (d). Overall, the identification of spiral wave tips coincides well with the target values. (Without special instructions, the green hollow circles and red solid triangles remain the same definition.) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Result – One Tip

100%

99.999%



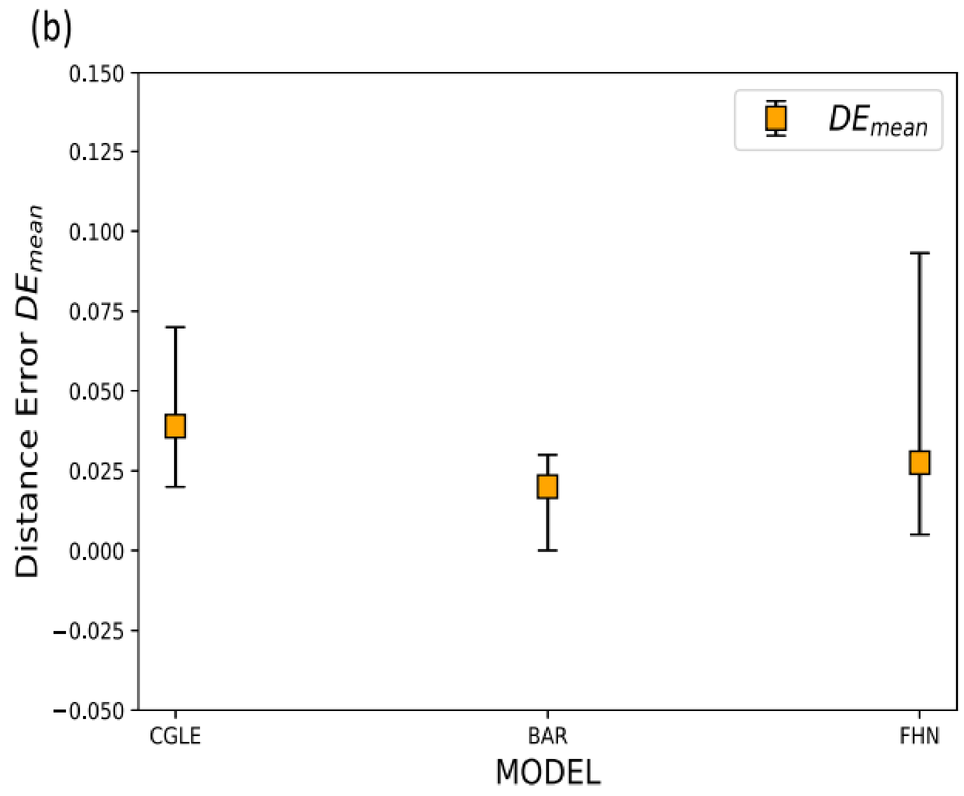


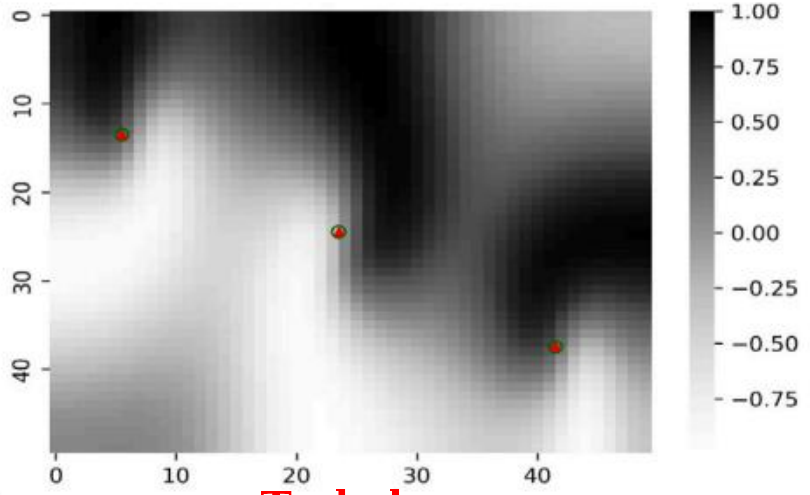
Fig. 3. The results of accuracy (AC) and distance error (DE) are calculated by 10 experiments again in three different one-tip systems, each time using 100 different time slices and the same trained model ((a, b), respectively). Even the poorest FHN system has a minimum accuracy AC_{mean} of 99.9992% and a maximum distance error DE_{mean} of 0.0932. Overall, the average accuracy AC is greater than 99.99%, and the average distance error DE_{mean} stays within 0.05 for all systems. It is demonstrated that the RC model makes these three systems feasible.

Result – Multiple Tips

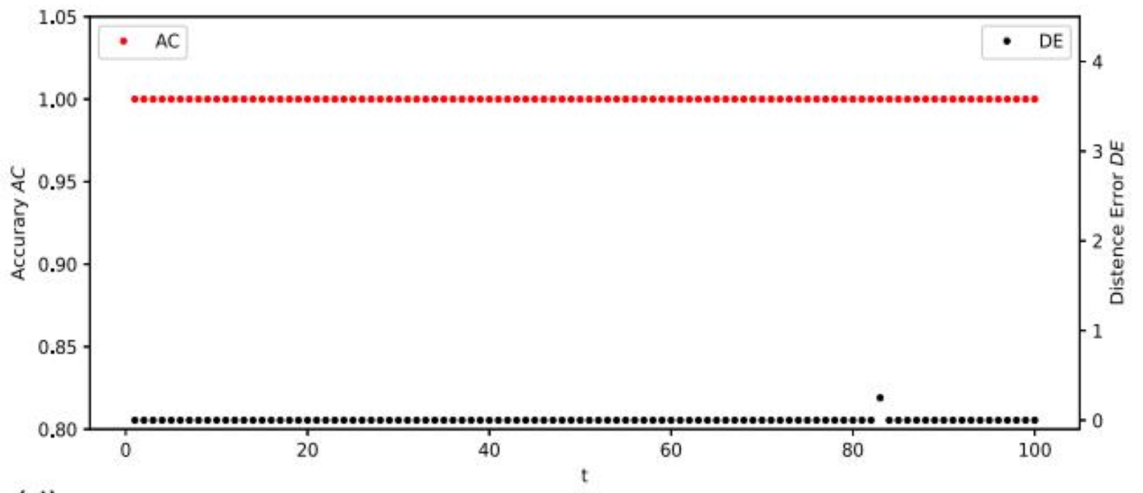
RC ▲

JM ○

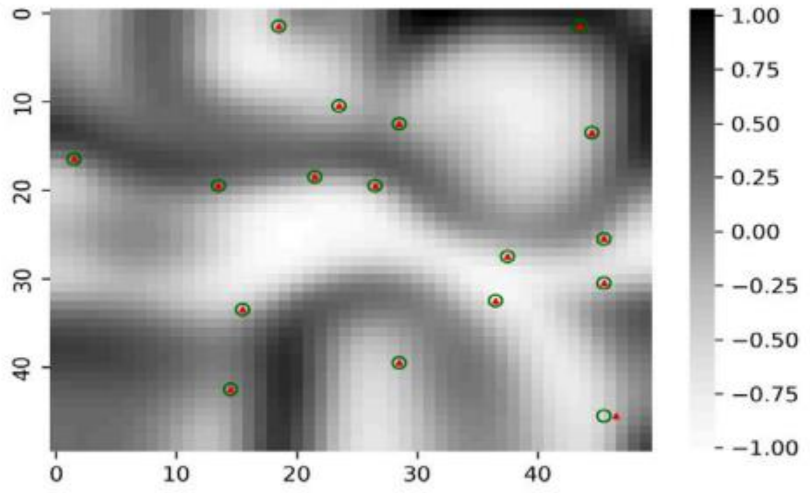
(a) Evolving spiral waves



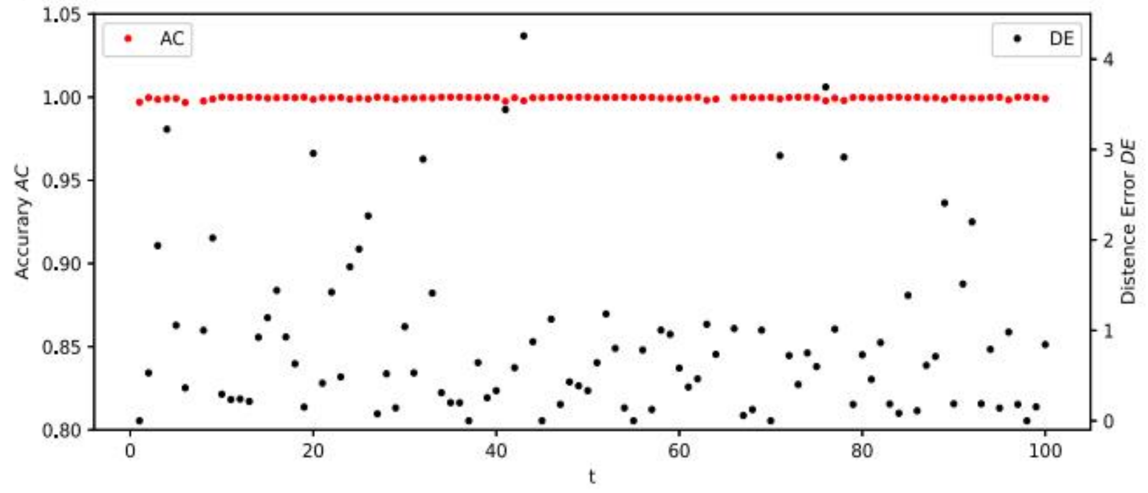
(b)



(c) Turbulence



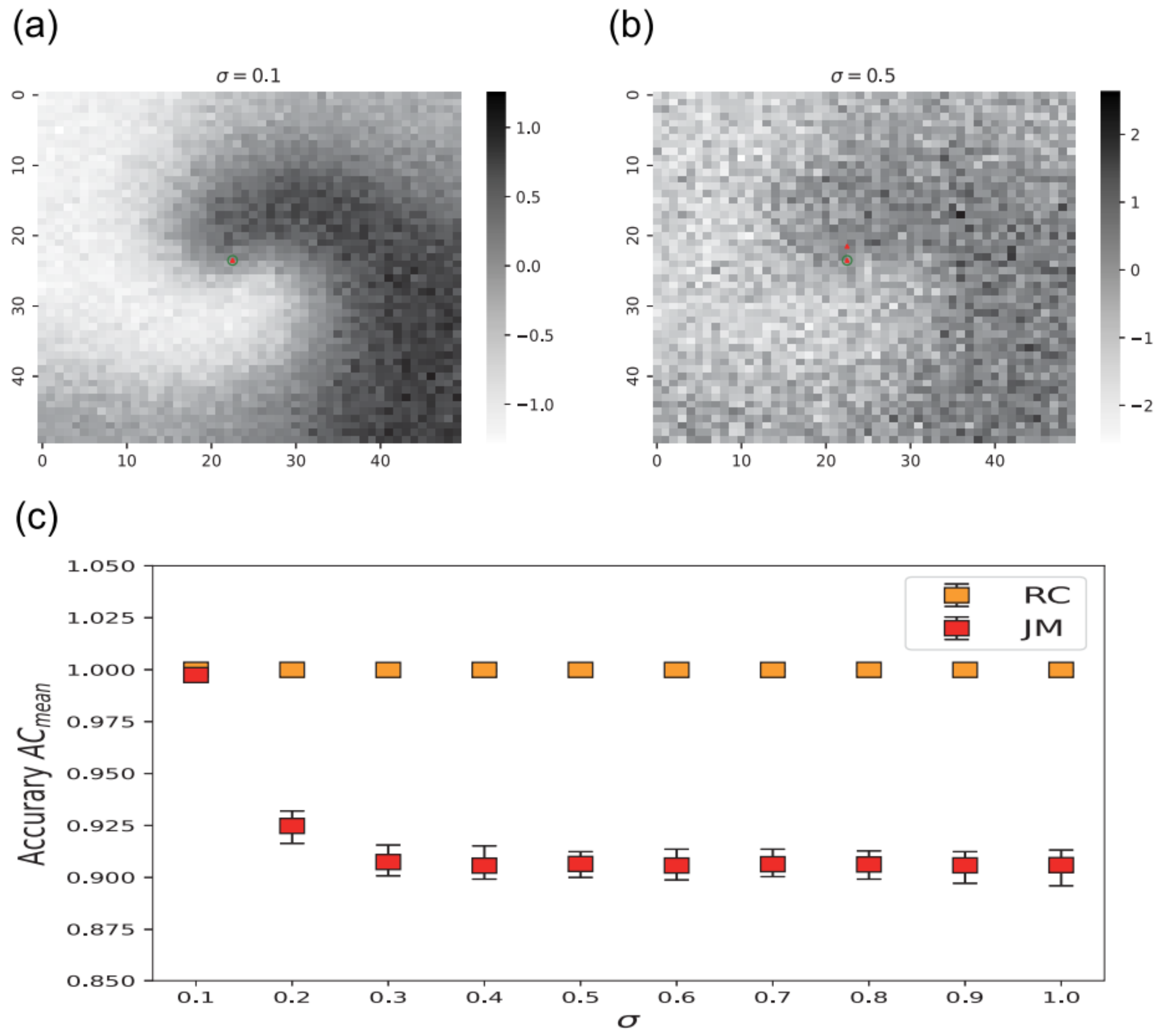
(d)



Result – Noise Test

RC ▲

JM ○

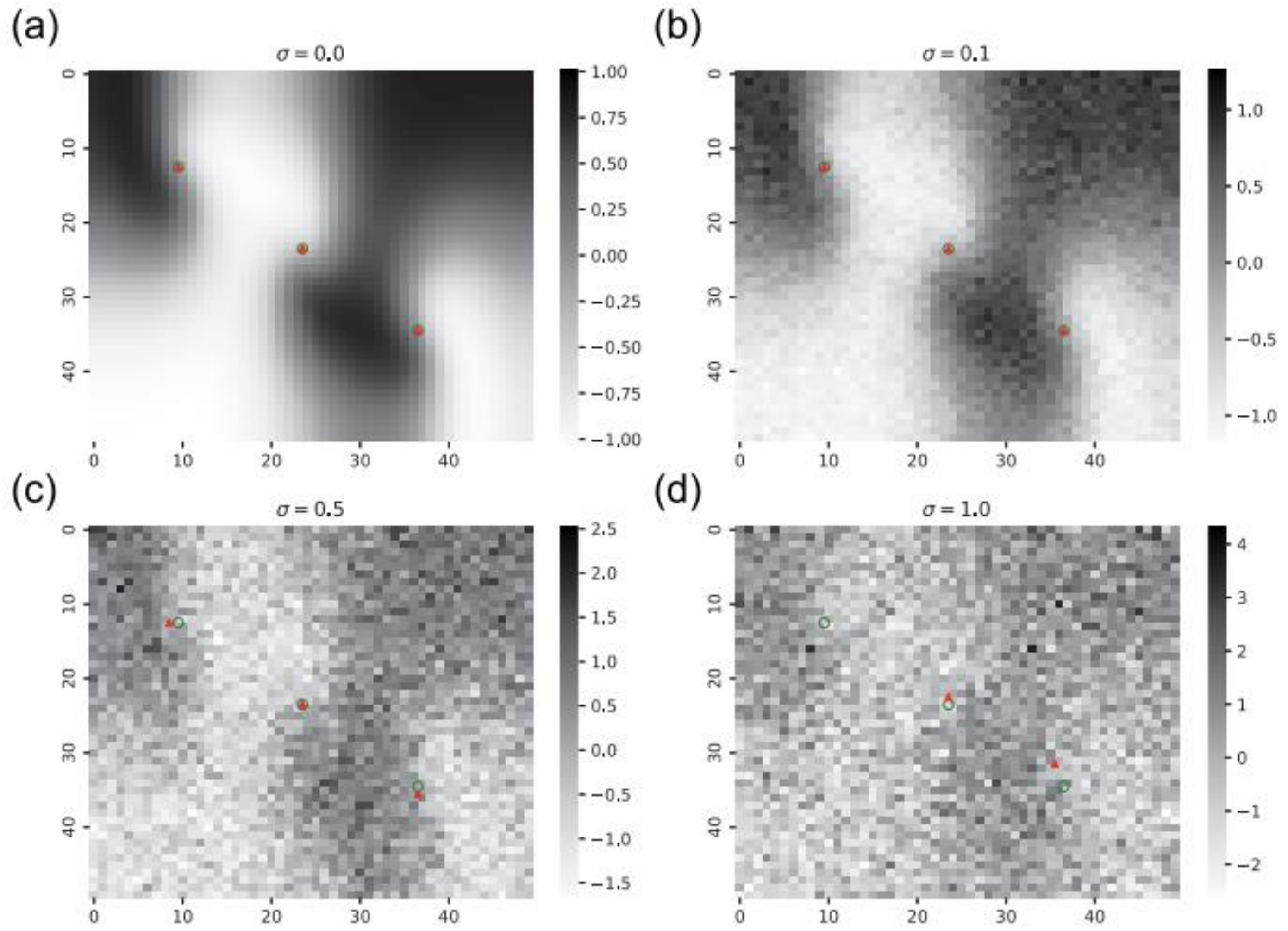


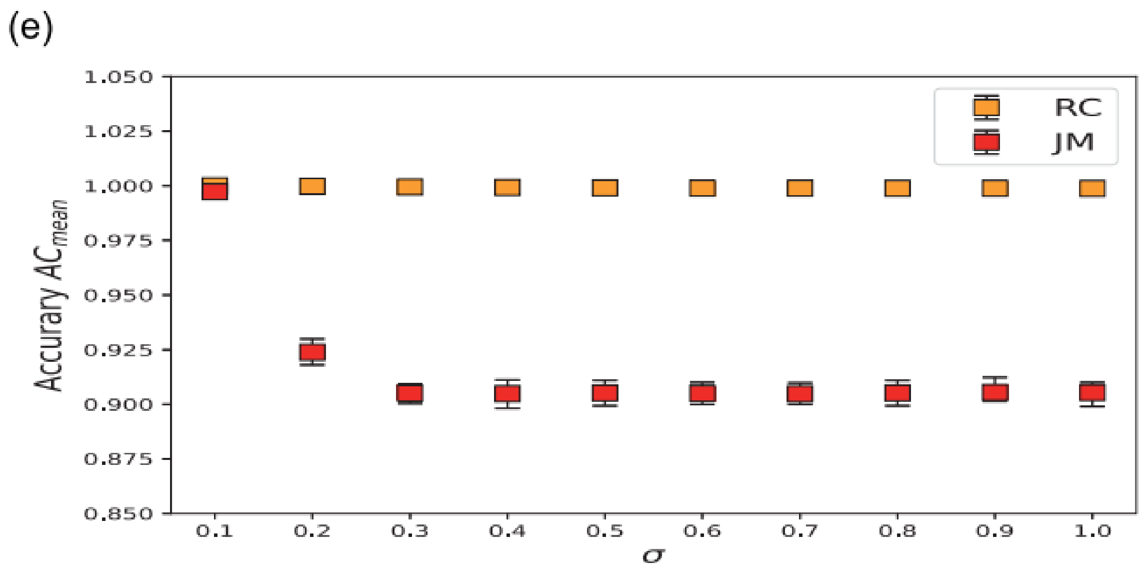
σ: Noise Level

Result – Noise Test

RC ▲

JM ○





σ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

Conclusion

- Reservoir Computing (RC) can be used to identify spiral tips in both simple (one tip) and complex (multiple tips) cases.
- It exhibits extreme tip identification accuracy even though there is high noise intensity.
- It greatly reduces the computational cost by using only one time slice instead of subsequent time series.