Reservoir Computing:
Liquid State Machine and Echo State Property

Belghomari Abdelmalek
Master Student at ENSICAEN, France
Research Intern in Neural Information Systems laboratory
at Shibaura Institute of Technology, Japan
E-mail : abdelmalek.belghomari@ecole.ensicaen.fr

August 12, 2024

Contents

1 Abstract **3**

2 Introduction **3**

3 Echo State Property in Echo State Networks **4**
3.1 What is an ESN ? 4
3.2 My first footsteps into Reservoir Computing 4
3.3 A closer look at the hyperparameters 6
3.4 First Experiments 6

4 Echo State Property in Liquid State Machine **7**
4.1 What is a Liquid State Machine ? 7
4.2 Building an LSM 8
4.3 Results Interpretation 12

5 Conclusion **15**

A Annex 1: Weekly Report 1 **17**

B Annex 2: Weekly Report 2 **20**

C Annex 3: Weekly Report 3 **24**

D Annex 4: Weekly Report 4 **26**

E Annex 5: Weekly Report 5 **28**

F Annex 6: Weekly Report 6 **34**

G Annex 7: Weekly Report 7 **39**

H Annex 8: Weekly Report 8 **43**

I Annex 9: Weekly Report 9 **46**

J Annex 10: Weekly Report 10 **53**

1 Abstract

Reservoir computing (RC) is an innovative computational framework designed to tackle complex temporal processing tasks by using dynamic systems. It is particularly well-suited for handling time series predictions, pattern recognition, and real-time signal processing. This approach employs a fixed, randomly connected neural network called a "reservoir" that transforms input data into a high-dimensional space, where a simple linear read-out layer can perform effective predictions or classifications.

There are two models in reservoir computing that first caught my attention; Spiking Neural Networks (SNN) and Liquid State Machines (LSM). They are inspired by the biological principles underlying neural dynamics; SNNs use spikes to convey information, mimicking the natural processes of neuronal firing, while LSMs incorporate both spiking neurons and synaptic dynamics to emulate the transient and adaptable nature of neural circuits. These models provide a powerful toolkit for exploring the computational capabilities of the brain and investigating how the brain processes information. By capturing the temporal and spatial aspects of neural activity, SNNs and LSMs provide information on the mechanisms of cognition, learning, and brain plasticity. Consequently, reservoir computing not only improves our understanding of artificial intelligence but also contributes to demystifying the complexities of the human brain.

2 Introduction

In this report, we will focus on the Echo State Property (ESP), a fundamental concept in reservoir computing, particularly in the context of Echo State Networks (ESN). The ESP ensures that the influence of initial conditions on the network's state fades over time, allowing the network to effectively process input sequences of varying lengths.

While the ESP is well studied and understood for ESNs, its existence and function in the context of Liquid State Machines (LSM) are less clear. LSMs, which incorporate more complex neuronal dynamics through spiking neural networks, offer potential for applications that closely resemble natural brain processes. This report aims to explore whether a similar criterion to ESP can be defined for LSMs, which would allow for the formulation of a necessary and sufficient condition for their optimal operation.

By examining the ESP in the context of LSMs, we would like to establish a theoretical framework that not only helps to better understand the computational capabilities of LSMs, but also provides valuable insights into modeling biological neural networks. Thus, studying ESP for LSMs could enrich our understanding of the fundamental principles underlying learning and information processing in computational and biological neural networks.

3 Echo State Property in Echo State Networks

3.1 What is an ESN ?

The Echo State Property (ESP) is a key characteristic of Echo State Networks (ESNs), which are a fundamental model within the broader field of reservoir computing. An ESN consists of a large, recurrent neural network (RNN) with fixed weights, known as the reservoir, and a trainable output layer. The ESP ensures that the network's state is primarily determined by the input sequence rather than its initial conditions. This property is crucial because it allows the network to "echo" the influence of the input data, enabling effective learning and prediction without the instability associated with RNNs.

In ESNs, the reservoir is partially (due to the sparsity) connected, and the only part that is trained is the output layer. The input data passes through the reservoir, where it is transformed into a high-dimensional representation. This representation is then used by the output layer to make predictions or decisions. A key advantage of ESNs is their ability to handle time-series data effectively, making them particularly suitable for tasks such as sequence prediction and temporal pattern recognition.

3.2 My first footsteps into Reservoir Computing

As part of my initial work in reservoir computing, I began by studying Echo State Networks, one of the simplest form of reservoir computing models. This approach allowed me to gradually enter the field by understanding the fundamental principles and mechanisms that drive these networks.

After reading the Reservoir Computing[1] book by Kohei Nakajima, I deepened my understanding through various educational videos, including one made by Professor Nakajima. These resources introduced me to the concept of RC. Training the readout only allows efficient and effective learning from time series data, with less data rather than other machine learning methods, highlighting the potential of ESNs in this area. One aspect of ESNs that I found particularly interesting is the potential to incorporate feedback loops, although they are not always necessary. It can enhance the network's ability to capture dynamic temporal dependencies within the data, making the model even more powerful for specific tasks.

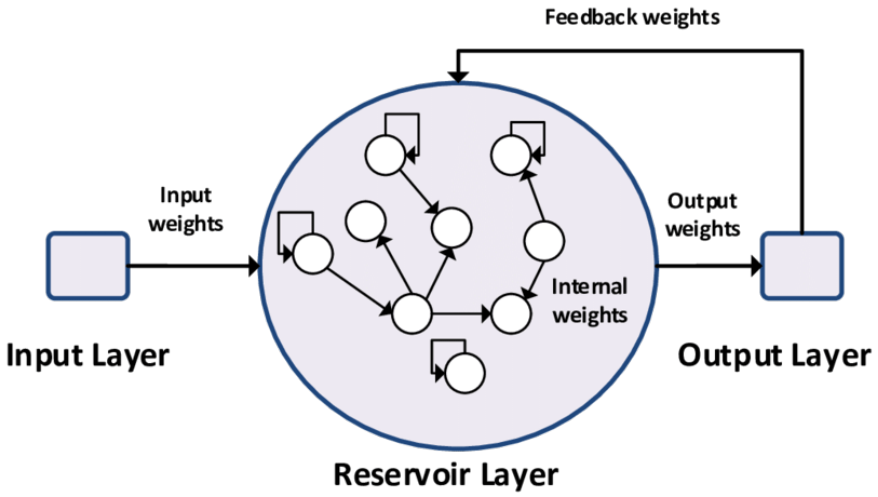


Figure 1: Scheme of an Echo State Network

Echo State Network System Equations

Here is the equation system that make the ESN principle working :

The basic discrete-time sigmoid unit echo state network with N reservoir units, K inputs, and L outputs is governed by the state update equation:

$$x(n+1) = f(Wx(n) + W_{\text{in}}u(n+1) + W_{\text{fb}}y(n)), \tag{1}$$

where $x(n)$ is the N-dimensional reservoir state, f is a sigmoid function (usually the logistic sigmoid or the tanh function), W is the $N \times N$ reservoir weight matrix, W_{in} is the $N \times K$ input weight matrix, $u(n)$ is the K-dimensional input signal, W_{fb} is the $N \times L$ output feedback matrix, and $y(n)$ is the L-dimensional output signal. In tasks where no output feedback is required, W_{fb} is nulled. The extended system state $z(n) = [x(n); u(n)]$ at time n is the concatenation of the reservoir and input states. The output is obtained from the extended system state by

$$y(n) = g(W_{\text{out}}z(n)), \tag{2}$$

where g is the output activation function.

These equations are adapted from the Echo State Network page on Scholarpedia[2].

During my research, I focused on the understanding of the various hyperparameters that can influence the performance of the reservoir in predicting time-series data. These hyperparameters include the spectral radius, input scaling, and reservoir sparsity, each playing a critical role in ensuring that the ESP is maintained, and the network functions optimally.

3.3 A closer look at the hyperparameters

In my exploration of Reservoir Computing, I discovered that specifics hyperparameters significantly influence the model's performance, particularly the sparsity of connections within the reservoir and the spectral radius of the reservoir weight matrix. Sparsity, which refers to the proportion of zero weights in the reservoir, affects the network's connectivity and can lead to more efficient information processing by reducing redundancy and enhancing signal propagation. On the other hand, the spectral radius of the reservoir's weight matrix, is critical for maintaining a balance between memory and stability within the network. A spectral radius close to one ensures that the reservoir's state dynamics are neither too stable (leading to information decay) nor too chaotic (resulting in instability), thus enabling the network to retain and process temporal information effectively[2]. By carefully tuning these hyperparameters, the reservoir can be optimized to capture and predict complex time series data with greater accuracy.

3.4 First Experiments

After doing some research on the web on how to implement this kind of model, I found reservoirPy[3] a GitHub Repository made and maintained by French researchers. So, my first goal was to implement the ESN using my own code and the one provided by other researchers to test the ESP with time series data and chaotic time series data.

Here are the results that I got :

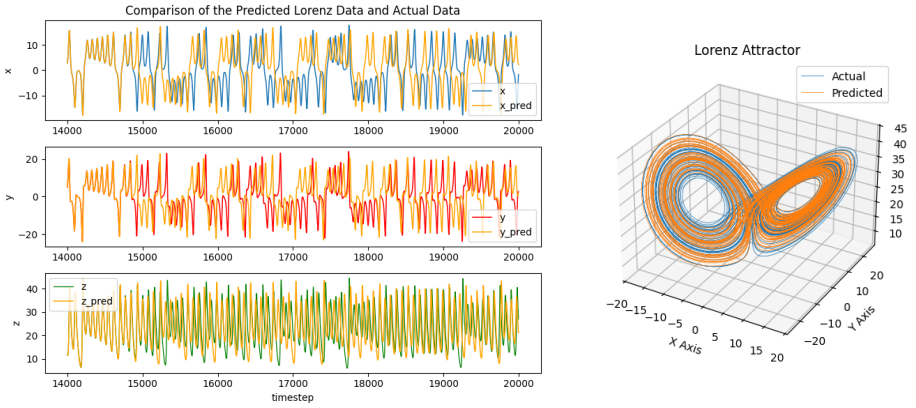


Figure 2: Results of the prediction of the Lorenz time series function by an ESN

The ESN's prediction is initially quite accurate, effectively capturing the values and behaviour of the chaotic function. However, as the prediction progresses, the ESN struggles to maintain accuracy and fails to consistently track the chaotic dynamics. The Echo State Property (ESP) demonstrates its strength in interpreting the values of the predicted function.

The ESP is a fundamental requirement for the reservoir computing principle to work effectively. It states that the reservoir should asymptotically wash out any information from its initial conditions, allowing the input signal to dominate the reservoir dynamics. This property is crucial for ensuring that the network's state is primarily driven by the input rather than lingering initial states.

In practice, the spectral radius of the reservoir weight matrix is a critical factor in achieving the ESP. It is generally observed that when the spectral radius is smaller than one, the ESP is maintained, allowing the network to process inputs effectively without instability. However, contrary to a common misconception, the ESP can sometimes be maintained even with a spectral radius larger than one, particularly when input amplitudes are large[2]. This suggests that the relationship between the spectral radius and the ESP is more nuanced than simply keeping the spectral radius below one.

As I explored different hyperparameters in Reservoir Computing, I found that both the spectral radius and the sparsity of connections within the reservoir significantly impact the model's performance. Adjusting these parameters allows for better control over the reservoir's dynamics, enhancing the network's ability to predict time-series data effectively.

4 Echo State Property in Liquid State Machine

4.1 What is a Liquid State Machine ?

A Liquid State Machine (LSM) is a computational model used within the field of reservoir computing, particularly suited for processing time-varying signals. An LSM, described by W.Maass in his lecture[4] and research paper[5] consists of a large, recurrent network of spiking neurons that operate in continuous time, effectively capturing the dynamic behavior of input signals. As the ESN, the core idea of LSMs is to project the input into a high-dimensional dynamic space using the recurrent network, which creates a rich temporal representation of the input signal.

LSMs are characterized by the following components:

- **Input Layer**: The input layer injects the time-varying signal into the recurrent network. This signal can be any continuous or discrete temporal signal. It works as an ESN input layer.
- **Recurrent Neural Network**: This network consists of spiking neurons that are randomly and sparsely connected. The neurons can be either excitatory or inhibitory, leading to complex and dynamic interactions within the network. The network's state evolves in response to the input signal, creating a high-dimensional representation of the input's temporal features.
- **Readout Layer**: The readout layer is a trainable component that maps the high-dimensional dynamic states of the recurrent network to the desired output. It also works as the ESN's output layer, but there are not

interconnected spiking neurons. Here is what we map from the readout neuron :

Current received by the readout neuron:

$$I_y(t) = \sum w_{oi} \cdot f(t) = \sum w_{oi} \cdot f[u(t)]$$

The integrated net current over $[0, T]$:

$$\int_0^T I_y(t)\, dt = \sum w_{oi} \cdot \int_0^T f(t)\, dt = \sum w_{oi} \cdot \int_0^T f[u(t)]\, dt$$

The LSM operates in two phases:

1. **Initialization Phase**: The network is initialized with random weights and connections. The weights are typically adjusted to ensure a balance between excitatory and inhibitory connections, allowing for stable and rich dynamics.
2. **Operation Phase**: During operation, the input signal is fed into the network, and the recurrent interactions among neurons produce a dynamic response. The readout layer processes this response to generate the final output.

The advantages of LSMs include:

- **Temporal Processing**: LSMs are highly effective at processing and recognizing temporal patterns due to their dynamic nature and ability to capture temporal dependencies.
- **Robustness**: The use of spiking neurons and sparse connections contributes to the robustness of LSMs, making them less sensitive to noise and variations in the input signal.
- **Biological Inspiration**: The concept of LSMs is inspired by the neocortex, a vital brain region for functions like sensory perception and conscious thought, characterized by interconnected layers of neurons developed over time[6].

LSMs have been applied to various tasks, including speech recognition[6], time series prediction, and robotic control[7], demonstrating their versatility and effectiveness in handling complex temporal data.

4.2 Building an LSM

To be able to test the ESP for the LSM, I needed to first take a look at existing python libraries and tools to help me build one. I first tried to see if I could create a model using my own ESN code or the reservoirPy library. I documented myself about the Maass model[5] that I wanted to reproduce (because it is the "original" one) to get interpretable results. I had to use some synaptic and neurons properties. For example, I used **Tsodyks synapses** and **Leaky Integrate and Fire** (LIF) neurons[5][8].

What are LIF Neurons?

- LIF neurons are a type of Spiking Neurons.
- The "leaks" involve a decay of the neuron's electrical potential:

$$\tau_m \frac{dV_m}{dt} = -(V_m - V_{\text{resting}}) + R_m \cdot (I_{\text{syn}}(t) + I_{\text{background}} + I_{\text{inject}}(t))$$

where $\tau_m = C_m \cdot R_m$ is the membrane time constant, R_m is the membrane resistance, $I_{\text{syn}}(t)$ is the current supplied by the synapses, $I_{\text{background}}$ is a constant background current, and $I_{\text{inject}}(t)$ represents currents induced by a "teacher." If V_m exceeds the threshold voltage V_{thresh}, it is reset to V_{reset} and held there for the length T_{refract} of the absolute refractory period.

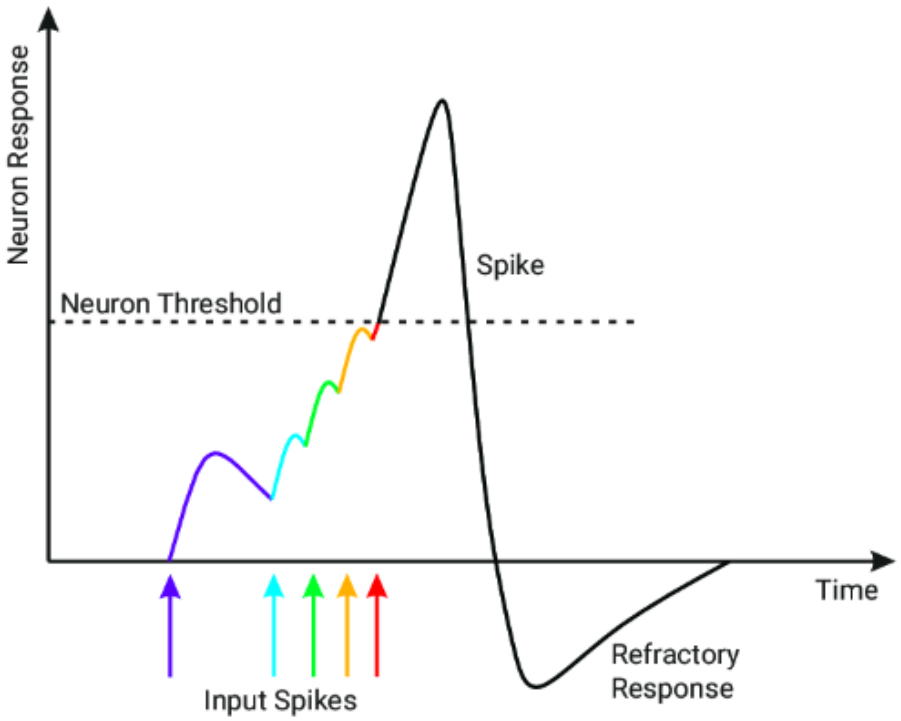


Figure 3: Depiction of the dynamical activity of a spiking neurone. The neurone receives input coming either from the data or lower layers (shown here as coloured arrows), which generate bumps in the membrane voltage; we refer to this voltage in the paper as u(t). If the voltage u(t) exceeds a threshold V_{thresh}, shown here as the dotted line, the neurone outputs a spike, and then enters a refractory phase where it is less likely to fire another spike for a short time

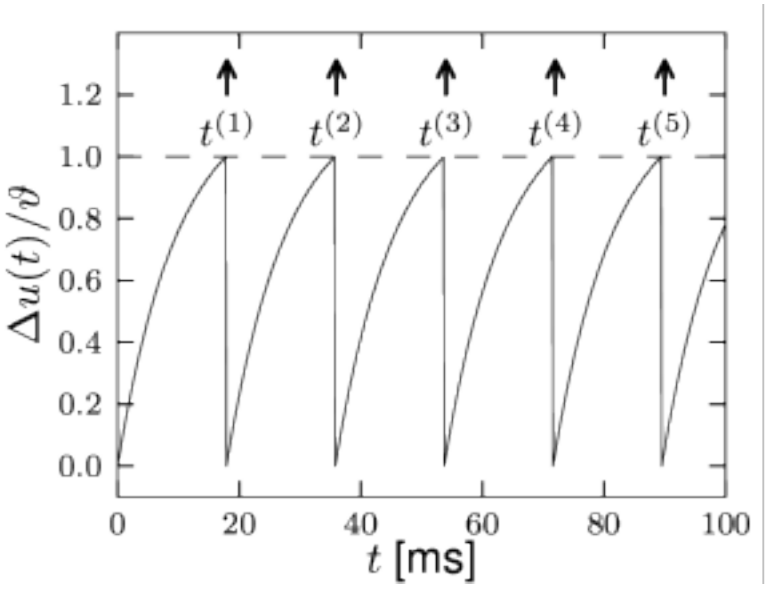


Figure 4: Leaky Integrate-and-fire model: Time course of the membrane potential of an integrate-and-fire neuron driven by constant input current $I_0 = 1.5$. The voltage $\Delta u(t) = u - u_{\text{rest}}$ is normalized by the value of the threshold ϑ. Units of input current are chosen so that $I_0 = 1$ corresponds to a trajectory that reaches the threshold for $t \to \infty$. After a spike, the potential is reset to $u_r = u_{\text{rest}}$.

I contacted the researchers from INRIA (one of the biggest French laboratories), that have implemented reservoirPy, and they told me that their library wasn't ready to create an LSM yet. So, I looked up to create one by myself. But that would have been really intricate and complex due to the brain mimetic of biological neurons. It would have taken a long time to create a library to do so. So I looked on the Internet if there was any library to code bio neurons. I found two of them : **Brian Simulator** and **NEST Simulator**. Brian seemed easier to use and grasp, while **the NEST Simulator** was harder to get hands-on with but more comprehensive. As a result, I preferred using **NEST Simulator**. By doing some research on NEST-simulator, I found the LSM GitHub Repository[9] done by the PhD researchers working with W.Maass. As they followed all the settings described in the Maass' experiment, I used the main code to get the settings from the original experiment. I adapted the code to grasp the nest library and get and inject the spikes into the LSM reservoir to get the output and train it.

I used two types of input, one provided by the git repository trying to predict the results of an XOR function and the other one being predicting the chaotic Lorenz function.

In the meantime, as it took time to understand all the complexities of **NEST**, I tried to adapt Professor R. Hosaka's code to obtain and interpret

the first results. Professor Hosaka's model was simple yet complete. This model uses the same processes as the code that I developed for the ESN, but it added the LIF neuronal spiking model in the reservoir. The weights in the reservoir are either positive or negative, with an inhibitory/stimulating ratio, to replicate the behaviour of a real LSM reservoir. During the experiments, the ratio was maintained at 80%, as it shouldn't be a hyperparameter that influences the ESP. For the **NEST** model, I used the *iaf_psc_alpha*[1] neuron model.

Here are the results of the experiments :

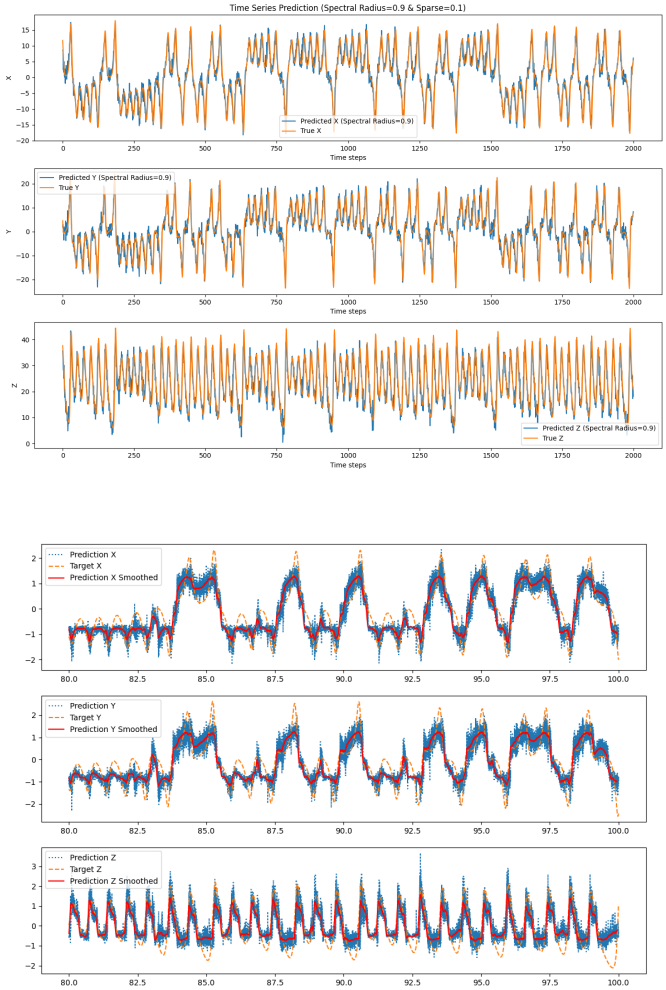


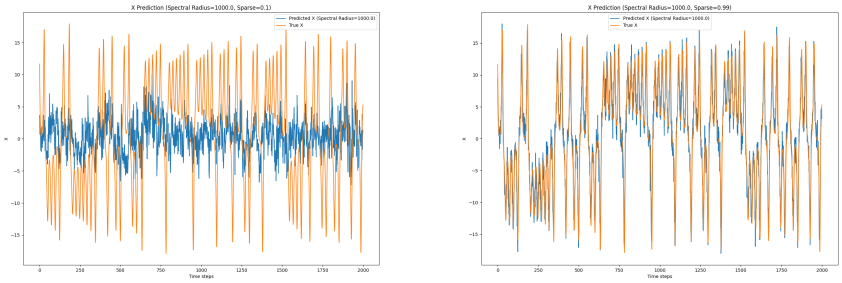
Figure 5: Results of the prediction of the Lorenz time series function by an LSM respectively without and with NEST

[1]For more information about the *iaf_psc_alpha* neuron model, click here.

4.3 Results Interpretation

First of all, the results that we got with both LSM models shows that the prediction we obtained is better than the one with an ESN. This improvement is likely due to the reservoir's enhanced ability to capture information and the biomimetic "memory" being more effective for chaotic functions. Second of all, we can see that the predictions are jittery. This comes from the fact that reservoir's neurons use the LIF neuron model. As we saw in [Figure 4], the IAF activation function looks like a step function, contributing to this jitteriness. Finally, the model using NEST is significantly more complex than the simpler model. This increased complexity likely accounts for its more jittery predictions, but it also enables the model to better capture the behaviour of the function.

From these results, I experimented to tune the reservoir sparsity or the spectral radius to see if I can find any ESP. As expected, the larger the spectral radius was above one, the worse the prediction was. Additionally, I also noticed something else. If I increase the spectral radius abusively high ; $\rho = 1000.0$, increasing the reservoir's sparsity to a very high level, such as $sparsity = 0.99$ (where $sparsity$ is a value between 0 and 1), helped cushion this and resulted in reasonably accurate predictions."



(a) High Spectral Radius and Low Sparsity(b) High Spectral Radius and High Sparsity

Figure 6: Comparison of A. High Spectral Radius and Low Sparsity vs B. High Spectral Radius and High Sparsity

At first, I wanted to find a theoretical way to determine the ESP, but I didn't find any documentation on the internet or in the documents we had in the laboratory. After discussion with some other researchers, they advised me to find experimental methods to determine the ESP empirically. Not wanting to lose more time searching for a mathematical proof, I decided to proceed with empirical methods, believing that this approach would eventually help me recognize when a theoretical result might be acceptable or not.

I searched for methods to test the ESP, but I haven't found anything conclusive. I then came up with two simple methods.

- **Test the same input with different Reservoir initial Conditions**:

We give a random initialization to the internal states within the reservoir. By feeding the model with the same input, we can determine the ESP by seeing if the output differs. If the output still remains the exact same no matter of the initial conditions, this is when we know that the ESP is respected. This is a straightforward and naive method to test the criterion. It does not require significant modifications to my code and provides a macrolevel interpretation of the ESP, though it is still imprecise.

- **Test with the Lyapunov Exponent**: The Lyapunov exponent[10] is a quantity that characterizes the separation of infinitesimally close trajectories. Quantitatively, two trajectories in phase space with initial separation vector δZ_0 diverge (provided that the divergence can be treated within the linearized approximation) at a rate given by $|\delta Z(t)| \approx e^{\lambda t}|\delta Z_0|$, where λ is the Lyapunov exponent.

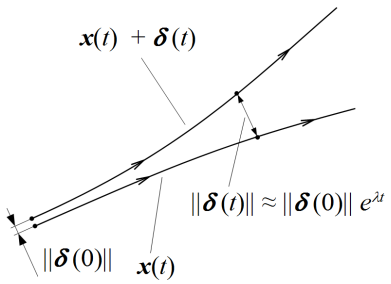


Figure 7: Explanations for the Lyapunov Exponent, characterizing the separation between two trajectories. Source[10]

And we can get the Lyapunov exponent like this :

$$|\delta Z(t)| = e^{\lambda t}|\delta Z_0| \iff e^{\lambda t} \approx \left|\frac{\delta Z(t)}{\delta Z_0}\right|$$

$$\implies \lambda t \approx \ln\left|\frac{\delta Z(t)}{\delta Z_0}\right|$$

$$\implies \lambda \approx \frac{1}{t}\ln\left|\frac{\delta Z(t)}{\delta Z_0}\right|$$

To keep the starting distance infinitesimal, we must verify:

$$\delta Z_0 \to 0$$

So we have:

$$\lambda = \lim_{t\to\infty}\lim_{\delta Z_0\to 0}\frac{1}{t}\ln\left|\frac{\delta Z(t)}{\delta Z_0}\right|$$

Despite my efforts to implement the Lyapunov exponent as a measure for the Echo State Property (ESP), the results remain unreliable as of the time I am writing this paper. Surprisingly, even in degenerate models, such as those with a high spectral radius and low sparsity, the Lyapunov exponent suggests that the ESP is respected, which contradicts my expectations. I have performed macro-level evaluations to assess the quality of the output, and based on these, I know that with a spectral radius close to $rho = 1$, the results should generally be consistent regardless of sparsity. This is particularly true when predicting the Lorenz chaotic system with a Liquid State Machine (LSM) featuring an excitatory/inhibitory ratio of 80%."

5 Conclusion

I will draw two conclusions: one regarding my research results and another reflecting on my internship experience in Professor Hosaka's laboratory.

Firstly, the last results indicate that my current implementation of the Lyapunov exponent for evaluating the Echo State Property still requires further refinement and validation.However, I successfully created a Liquid State Machine (LSM) using both a simple biological model inspired by my professor's code and a more complex, enriched model using the NEST simulator. These models are significant because they allow me to replicate the biological functioning expected from an LSM, which I hope will help my laboratory fellows in the future. I still can empirically evaluate the ESP from a macroscopic scale. Finishing implementing the Lyapunov method could be a next step for this research. I am also interested in switching the functioning of the reservoir neurons from Leaky Integrate and Fire model to Hodgkin-Huxley model, which is more realistic in the sense of biological brain neurons model.
Secondly, The results I obtained in these three months of experimentations are quite promising. I have learned a lot about Reservoir Computing, Spiking Models and Brain Information Processing—topics I never imagined understanding when I first arrived at Professor Hosaka's laboratory in May. I experimented a lot, first trying to understand and create a simple Echo State Network to studying the intricate science of the brain and spiking information with the liquid state machines. Throughout this journey, I engaged with various researchers, participated in conferences to share my findings, and learned from others. I have sharpened my skills in the field of machine learning and AI, but also in English and Japanese, and made several friends within the laboratory and outside.
I would like to conclude by thanking genuinely the ENSICAEN that gave me the opportunity to perform this internship abroad, Professor Hosaka-sensei for his patience, pedagogy and guidance towards me, the whole laboratory and the SIT that welcomed me in Japan and within their establishment, made me feel at home and even financed my participation for the IEEE WCCI Yokohama world AI conference.

References

[1] Kohei Nakajima and Ingo Fischer. *Reservoir Computing: Theory, Physical Implementations, and Applications*. Springer, 2021.

[2] Herbert Jaeger. *Echo Statne Netwrok*. `http://www.scholarpedia.org/article/Echo_state_network`. Last accessed: 2024-07-28. 2007.

[3] N. Trouvain et al. *ReservoirPy*. `https://github.com/reservoirpy/reservoirpy`. Version used: 0.3.11. 2020.

[4] Wolfgang Maass. "Liquid State Machines: Motivation, Theory, and Applications". `https://igi-web.tugraz.at/PDF/189.pdf`. Lecture.

[5] Henry Markram Wolfgang Maass Thomas Natschläger. "Real-time computing without stable states: a new framework for neural computation based on perturbations". In: *Neural Computation* (2002).

[6] Y. Zhang and P. Li. "A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition". In: *IEEE* (2015).

[7] R. de Azambuja et al. "Neurorobotic simulations on the degradation of multiple column liquid state machines". In: *IEEE* (2017).

[8] Oladipupo Gideon Gbenga. "Research on the concept of Liquid State Machines". Archive of Heriot-Watt University.

[9] A. Subramoney, M. Hoff, and Schmitt S. *LSM*. `https://github.com/IGITUGraz/LSM`. Version of NEST used: 2.17. 2017.

[10] Wikipedia. *Lyapunov Exponent Wiki*. `https://en.wikipedia.org/wiki/Lyapunov_exponent`. Accessed on 2024/07/06. last modified : 2024/07/22.

A Annex 1: Weekly Report 1

Report n°1 : Reservoir Computing

Objective :
Establish a foundational understanding of reservoir computing, with a focus on liquid state machines (LSMs).

Tasks :
1. Watch videos on reservoir computing and Take detailed notes and revisit complex sections

2. Read the sections (from p.59 to p.76) of the Springer book on reservoir computing. Try to understand the mathematics behind.

Work that have been done :

- Watching Videos on Reservoir Computing

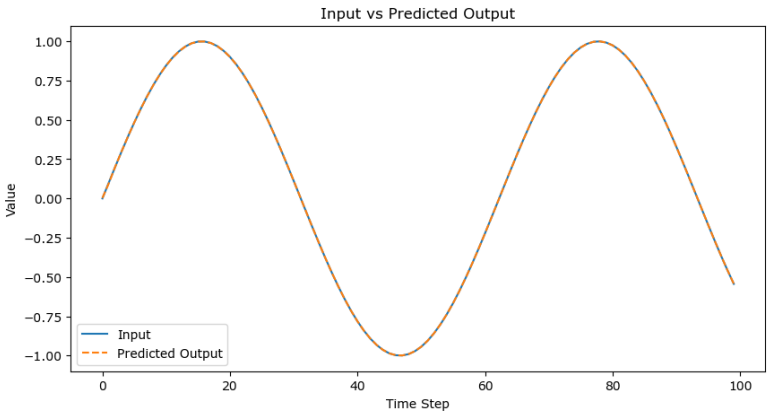
  Key points:
  - Reservoir Concept: Utilises a recurrent neural network (RNN) as a dynamic reservoir that captures temporal patterns through its internal state transitions.
  - Memory and Prediction: The reservoir's memory capabilities make it suitable for predicting chaotic phenomena and other time-dependent processes.
  - Training Simplicity: Only the readout layer is trained, which simplifies the training process compared to traditional neural networks. Also, it needs much less training data to be fully trained.

- Creating a Reservoir Computer in Python :

  Objective: Implement a basic reservoir computer using Python to reinforce theoretical knowledge with practical experience.
  - Reservoir Setup: Created a simple RNN with random weights as the reservoir.
  - Data Processing: Fed sample input data through the reservoir to observe state transitions.
  - Readout Training: Trained a linear readout layer using the states produced by the reservoir to predict outputs.

Implementation Code Snippet :

```python
import numpy as np

# Define reservoir parameters
input_size = 1
reservoir_size = 100
output_size = 1

# Create random weight matrices
W_in = np.random.rand(reservoir_size, input_size) - 0.5
W = np.random.rand(reservoir_size, reservoir_size) - 0.5

# Initialize reservoir states
state = np.zeros((reservoir_size, 1))

# Sample input data
inputs = np.sin(np.linspace(0, 10, 100)).reshape(-1, 1)

# Collect reservoir states
states = []
for u in inputs:
    state = np.tanh(np.dot(W_in, u) + np.dot(W, state))
    states.append(state.flatten())
states = np.array(states)

# Train readout layer
from sklearn.linear_model import Ridge

readout = Ridge(alpha=1.0)
readout.fit(states, inputs)

# Predict output
predicted = readout.predict(states)
```



- Areas for Improvement: Plan to explore more sophisticated reservoirs and fine-tune parameters for better performance.

Summary of Reservoir Computing Properties :
- Recurrent Neural Network (RNN): Forms the core of the reservoir, providing dynamic memory capabilities.
- Memory and Prediction: Effective for modeling and predicting chaotic and time-dependent phenomena due to its inherent memory. (Still have to test it)
- Simplified Training: Only the readout layer is trained, making the approach computationally efficient and easy to implement.

Conclusion :
This week's activities provided a robust introduction to reservoir computing, combining theoretical learning with practical implementation. The insights gained lay a strong foundation for further exploration into liquid state machines and more complex reservoir computing models.

Next Steps :
- Implement a more sophisticated model (a chaotic system as such as the Lorentz's one)
- Deep Dive into LSMs: Focus on understanding the unique properties and applications of liquid state machines.
- Advanced Python Implementation: Develop more advanced reservoir computing models and experiment with different datasets.

B Annex 2: Weekly Report 2

Report n°2 : Reservoir Computing & Echo state property
13/05/24 - 19/05/24

Work that have been done:

• **Exploring Reservoir Computing with ReservoirPy**

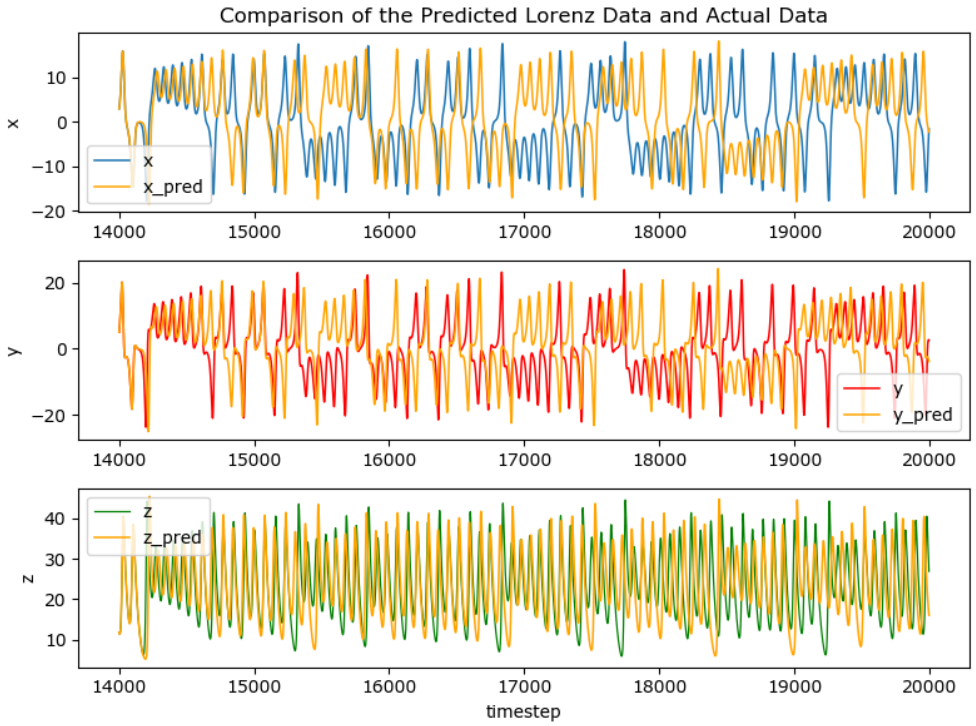
Objective: Utilize the Python library ReservoirPy to experiment with echo state networks on a chaotic system such as the Lorenz system.

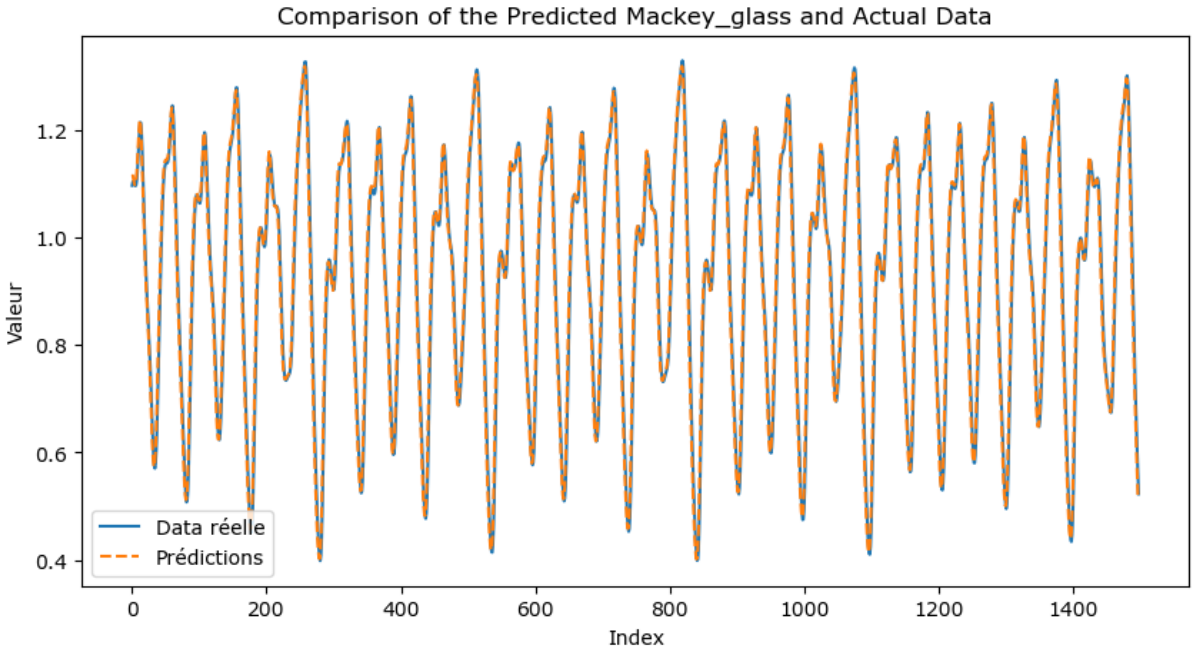
- Library Setup:

Employed the ReservoirPy library to set up and run echo state networks.

- Model Implementation:

Applied the echo state network model to forecast the dynamics of the chaotic Lorenz system.





Key points:

- Echo State Network and Chaotic Systems:

The echo state network, a type of reservoir computing model, excels at capturing the dynamics of chaotic systems initially due to its rich reservoir of randomly connected neurons. This reservoir can mirror the complex, sensitive dependence on initial conditions characteristic of chaotic systems like the Lorenz attractor.

- Limitations in Long-Term Prediction:

While echo state networks can initially model chaotic systems effectively, their performance tends to deteriorate over time. This decline occurs because the inherent predictability of chaotic systems decreases sharply with time — a phenomenon related to the system's sensitive dependence on initial conditions and the accumulation of prediction errors.

- Echo Property of Reservoirs:

The echo state property is crucial in reservoir computing; it ensures that the reservoir's response to a given input dissipates over time, which prevents previous states from indefinitely affecting future states. This fading memory characteristic is essential for the system's ability to adapt and respond to new inputs while maintaining stability.

Differences in Prediction Performance between the Lorenz and Mackey-Glass Systems using Echo State Networks :

The varying success of echo state networks (ESNs) in predicting the behavior of the Lorenz and Mackey-Glass systems can be attributed to several factors rooted in the unique characteristics of these chaotic systems and the operational dynamics of ESNs:

1. Nature of the Systems:

- Lorenz System:

The Lorenz system is a three-dimensional deterministic system that epitomizes deterministic chaos, characterized by extreme sensitivity to initial conditions. This sensitivity makes long-term prediction particularly challenging, as small inaccuracies in initial conditions can lead to vastly divergent outcomes.

- Mackey-Glass System:

Though also chaotic, featuring long-term dependencies, the Mackey-Glass system, modeled by a delay differential equation, exhibits a different type of chaos. Its dynamics may manifest more regular patterns over extended periods compared to the Lorenz system, potentially making it somewhat easier to predict using ESNs.

2. Memory Capacity and Reservoir Dynamics:

- ESNs maintain a "reservoir" of neurons, where the state is updated based on past inputs, allowing the network to "remember" past information over a certain interval. While effective for medium-term dependencies, the reservoir may struggle to maintain accuracy for the very long-term dependencies seen in highly chaotic systems like Lorenz.

Conclusion :

This past week has been highly productive in deepening my understanding of Echo State Networks (ESNs) and their capability to predict chaotic behaviors. The hands-on experience with the ReservoirPy library, combined with theoretical exploration, has provided a robust introduction to reservoir computing. These insights have laid a strong foundation for further research into not only ESNs but also their relationship with Liquid State Machines (LSMs).

Next Steps :

- Refining my skills in working with ESNs and exploring their integration with LSMs.

- Enhance my proficiency with ReservoirPy. I have reached out to the researchers responsible for developing this library to discuss ESNs in more depth and to seek guidance on constructing LSMs effectively. This interaction is expected to significantly enrich my understanding and contribute to my ongoing projects in reservoir computing.

C Annex 3: Weekly Report 3

Report n°3 : Studying echo-state property and LSM
20/05/24 - 26/05/24

Work that have been done:

• Read *Reservoir Computing* by Kohei Nakajima and Ingo Fischer -> focus on *Learn To Learn method* (p.59 - 76)

Objectives:

- Deepen my understanding of how to calculate the echo state property in Echo State Networks (ESNs)
- Distinguish the main differences between ESNs and Liquid State Machines (LSMs)

Key points:

• Reservoir Composition:

ESNs utilize a network of randomly connected non-spiking neurons with fixed weights, primarily working with continuous signals. LSMs employ spiking neurons, which simulate more complex, biologically realistic neural dynamics and handle binary spikes for processing

• Signal Processing and Learning:

ESNs focus on the amplitude of signals with learning restricted to the linear readout layer. LSMs are adept at capturing precise timing of inputs, potentially using more advanced learning rules that can adjust weights based on the timing of spikes.

• Echo State Property:

In ESNs, this property ensures that the influence of a given input fades over time, crucial for the network's stability and ability to adapt to new inputs. In LSMs, the emphasis is on how input timings influence the network state, reflecting a different approach to using neural computation.

Conclusion :

This week was less productive in terms of direct outputs due to increased focus on theoretical learning and participation in the NIT's workshop. While this meant fewer experimental results, the theoretical insights and interactions with fellow researchers provided valuable perspectives on the potential applications of reservoir computing, particularly in mimicking brain functions, which will inspire and inform my future research.

Next Steps :

- Refining my skills in working with ESNs and exploring their integration with LSMs.
- Enhance my proficiency with ReservoirPy.

D Annex 4: Weekly Report 4

Report n°4 : Studying LSM, ESN and Spiking Neurons principle
27/05/24 - 2/06/24

Work that have been done:

- Get to grips with the ReservoirPy library to master its features
- Studying lectures of RC, focused on LSMs
- First study of the principle of spiking neurons in neurology and their applications in reservoir computing

Objectives:

- Deepen my understanding of how to calculate the echo state property in Echo State Networks (ESNs)
- Understand the principle of Spiking Neurons (at least do some research on books and papers to be able to read them next week)
- Reviewing the principles of RC, and have a hands-on experience on setting ESN, choosing hyperparameters for reservoirs etc..

Readings:

- Spiking Neuron Model by W.Gerstener & W.Kistler
- Echo State Property of Deep Reservoir Computing Networks By C.Gallicchio & A.Micheli
- Reservoir Computing - Echo State Networks and Liquid State Machines by V.Sankhala

Conclusion :

This week, I have focused my research on finding papers and books to understand the underlying principles of spiking neurons, the LSM principle, and the method to define the echo state of a reservoir. My goal is to deeply understand how to determine the echo criterion of an LSM reservoir. Additionally, I am working on linking my studies with their neurological aspects.

While this focus has resulted in fewer experimental outputs, the theoretical knowledge and connections to neurology are providing valuable insights that will guide my future research.

Next Steps :

- Refining my skills in working with ESNs and exploring their integration with LSMs
- Read and study the papers i found last week
- Link my research with brain functioning and possible applications in Neurology
- Enhance my proficiency with ReservoirPy.

E Annex 5: Weekly Report 5

Report n°5 : Studying LSM, ESN and Spiking Neurones principle
2/06/24 - 16/06/24

Work that have been done:

- Get to grips with the ReservoirPy library to master its features
- Studying lectures of RC, focused on LSMs (Not Finished)
- Reading of Spiking Neuron Model by W.Gerstener & W.Kistler
- Reading of Research on the concept of Liquid State Machines by G.G.Oladipupo

Objectives:

- Understand the principle of Spiking Neurons in LSMs to understand how we can build one from scratch and how Find the echo state criterion.

Readings:

- Spiking Neuron Model by W.Gerstener & W.Kistler
- What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? By W.Gerstener,
- Reservoir Computing - Echo State Networks and Liquid State Machines by V.Sankhala

Key Points :

- Spiking Neural Networks (SNNs): LSMs use SNNs, which are biologically inspired and operate in the temporal domain, making them efficient and accurate for cognitive tasks like speech recognition and image classification.
- Reservoir Structure: The LSM reservoir contains Leaky Integrate and Fire (LIF) neurons with realistic dynamic synaptic connections, transforming input streams to higher-dimensional states, while the readout neurones, also LIF, are not interconnected.

What are LIF neurones ?

- LIF are a type of Spiking Neurones
- The « leaks » involves in a decay of the neurones electrical potential :

$$\tau_m \frac{dV_m}{dt} = -(V_m - V_{\text{resting}}) + R_m \cdot \left(I_{\text{syn}}(t) + I_{\text{background}} + I_{\text{inject}}(t) \right)$$

where $\tau_m = C_m \cdot R_m$is the membrane time constant, R_m is the membrane resistance, $I_{\text{syn}}(t)$is the current supplied by the synapses, $I_{\text{background}}$ is a constant background current, and $I_{\text{inject}}(t)$ represents currents induced by a "teacher." If V_m exceeds the threshold voltage V_{thresh}, it is reset to V_{reset} and held there for the length T_{refract} of the absolute refractory period

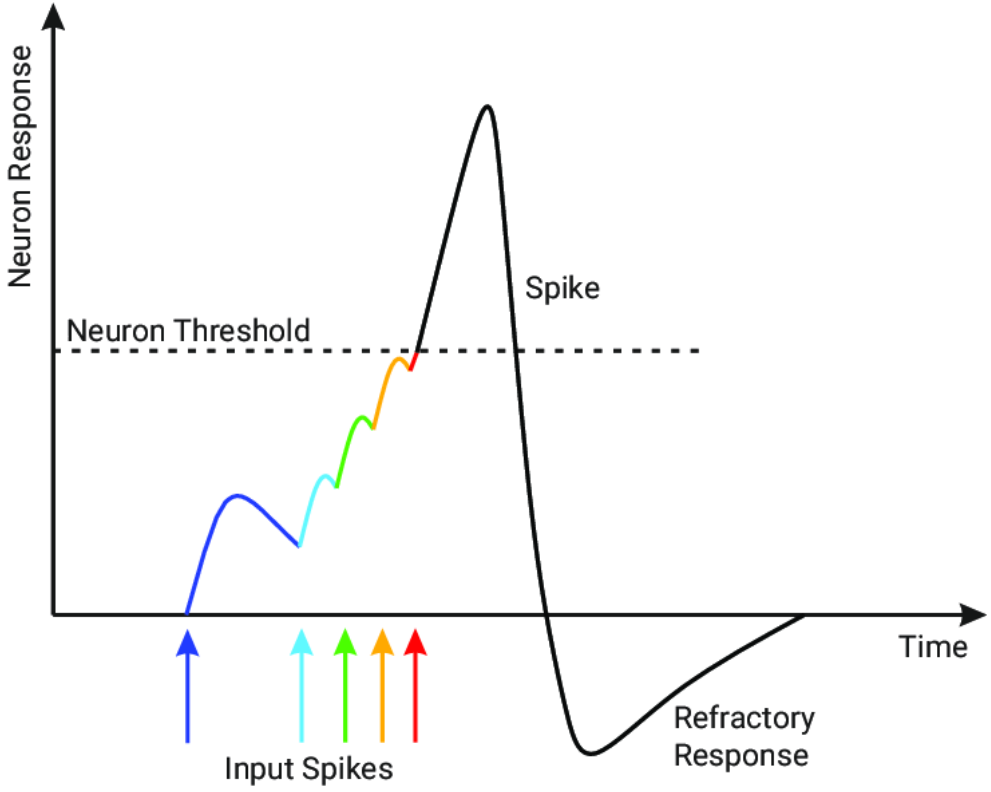


Fig.1 Depiction of the dynamical activity of a spiking neurone. The neurone receives input coming either from the data or lower layers (shown here as colored arrows), which generate bumps in the membrane voltage; we refer to this voltage in the paper as u(t). If the voltage u(t) exceeds a threshold ϑ, shown here as the dotted line, the neurone outputs a spike, and then enters a refractory phase where it is less likely to fire another spike for a short time.

- Dimensional Transformation: LSMs convert lower-dimensional inputs into higher-dimensional internal states that feed into a memoryless readout circuit, which produces the final output.
- Neocortex Inspiration: The concept of LSMs is inspired by the neocortex, a vital brain region for functions like sensory perception and conscious thought, characterised by interconnected layers of neurones developed over time.
- Current received by the readout neurone :

$$I_o(t) = \sum w_{oi} \cdot f_i(t) = \sum w_{oi} \cdot f_i[u(t)]$$

- The integrated net current over $[0,T]$:

$$\int_0^T I_o(t)\,dt = \sum w_{oi} \cdot \int_0^T f_i(t)\,dt = \sum w_{oi} \cdot \int_0^T f_i[u(t)]\,dt$$

- **Offline/Batch Learning**

  - Static Data Processing:

Offline or batch learning involves training a model on a fixed dataset all at once. The entire dataset is available at the beginning of the training process, and the model does not update its parameters based on new data after the initial training phase

  - Prior Knowledge and Complexity:

This method often requires prior knowledge and can introduce complexity, as the model must generalise from a static set of data to handle new, unseen data without real-time updates

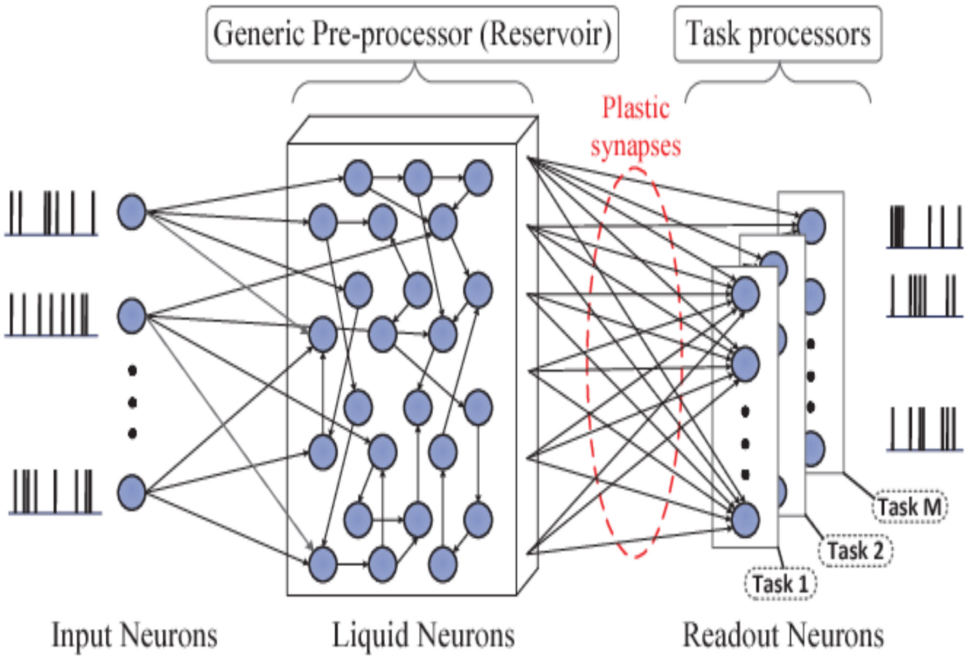


Fig.2 A Liquid State Machine supporting multiple tasks (Wang et.al, 2015)

- **Online Learning**

• Dynamic Data Processing:

Online learning continuously updates the model with new data as it arrives. The model adapts over time, making it well-suited for real-time applications where data is constantly changing

• Flexibility and Cost Efficiency:

This approach does not require prior knowledge of the data, reducing complexity and implementation costs. It allows the model to learn and adapt dynamically, providing flexibility in handling diverse and evolving datasets

Why is Online Learning better here ?

• The results suggest that accurate brain maps, which show areas of activity in the brain, can be generated using data-driven approaches. These approaches do not require prior knowledge about the expected hemodynamic response function (HRF), which is typically used to interpret BOLD (Blood-Oxygen-Level-Dependent) signals.

• Traditional methods often rely on batch or offline learning, where models are trained on a fixed dataset and require predefined information about the HRF. In contrast, the data-driven approach used here allows for more flexibility and adaptability without needing such prior knowledge.

How does it really stand in the general case?

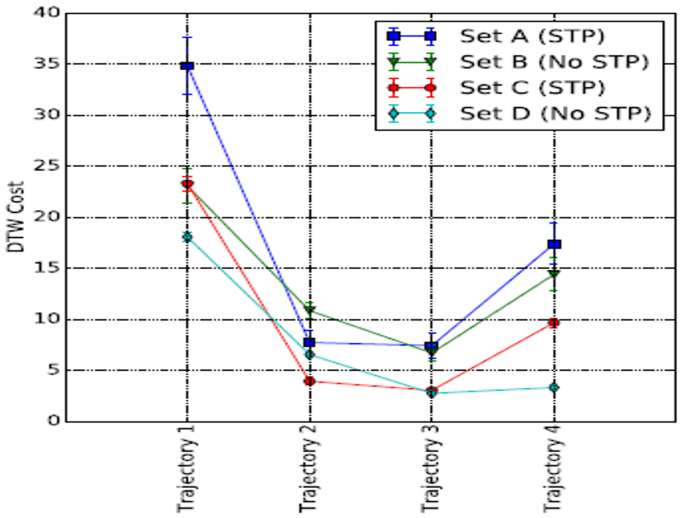


Fig.3 Dynamic Time Warping (DTW) cost considering the trajectories individually.
The lower the DTW cost, the better are the results (de Azambuja et.al 2015).

- **Short-Term Plasticity (STP)**

STP refers to temporary changes in synaptic strength that occur due to recent neural activity. These changes can either enhance (facilitate) or diminish (depress) synaptic transmission.

- Dynamic Adaptation:

STP allows the synapses to adapt quickly to changes in neural activity, which can be beneficial for tasks that require real-time processing of temporal patterns.

- Temporal Coding:

In speech recognition, for example, the ability to modify synaptic strength quickly can help the model distinguish between similar sounds or patterns that occur in quick succession.

- Memory and Learning:

By temporarily adjusting the synaptic weights, STP can play a role in short-term memory and learning, allowing the model to remember recent inputs and use this information to influence current or future outputs.

- **Dynamic and Real-Time Applications:**

• In contexts where data changes rapidly and unpredictably (e.g., real-time monitoring, speech recognition), LSMs with STP and online learning are preferred. The combination of STP's adaptive synaptic changes and online learning's continuous updates allows for high responsiveness and adaptability to new data

• Example:

In a speech recognition task, where the input data is a continuous stream of spoken words, STP helps the LSM model to adapt quickly to variations in speech, and online learning allows the model to refine its understanding with each new word or phrase.

- **Stable and Predictable Environments:**

• For tasks that require stable and consistent performance, such as batch processing of data or scenarios where the data environment is relatively static, LSMs without STP and offline learning may be more appropriate. The absence of STP reduces unnecessary complexity, and batch learning leverages prior knowledge to train the model effectively on a fixed dataset .

• Example:

In a data analysis task where the data is collected and processed in batches, and there is no need for real-time updates, offline learning and a stable LSM configuration without STP can efficiently handle the task.

Conclusion :

This week, I have further deepened my understanding of Liquid State Machines (LSMs) and the principles of spiking neurones. While I have made significant progress in grasping the core concepts, there are still several papers I need to read to fully comprehend the intricacies involved. Additionally, I plan to conduct some tests in Python to apply and validate my theoretical insights practically.

I am also exploring the possibility of participating in the Yokohama conference on AI, which includes a forum on reservoir computing and neurology. This could be an excellent opportunity to expand my knowledge and network with experts in the field.

Next Steps :

- Refining my skills in working with ESNs and exploring their integration with LSMs
- Reviewing the principles of RC, and have a hands-on experience on setting ESN, choosing hyperparameters for reservoirs etc..
- Deepen my understanding of how to calculate the echo state property in Echo State Networks (ESNs)
- Read and study the papers I found last week
- Link my research with brain functioning and possible applications in Neurology
- Enhance my proficiency with ReservoirPy.

F Annex 6: Weekly Report 6

Report n°6 : Studying LSM, ESN and ESP principle - 報告書第6号：LSM、ESN、ESPの原理を研究する
17/06/24 - 23/06/24

Work that have been done - 作業内容:

- Get to grips with the ReservoirPy library to master its features
- ReservoirPyライブラリを使いこなせるようにして、その機能を完全に理解します

- Studying lectures of RC, focused on LSMs (Not Finished)
- LSM（液体状態マシン）に重点を置いたリザーバーコンピューティングの講義を学習中です（まだ完了していません）。

- Reading of Spiking Neuron Model by W.Gerstener & W.Kistler
- W. GerstnerとW. Kistlerによるスパイキングニューロンモデルについての書籍を読んでいます。

- Reading papers about ESN's echo state property criterion
- ESNのエコーステートプロパティの基準についての論文を読んでいます。

Objectives - 目的:

- Understand how to find the echo state criterion of an ESN to apply it to a LSM .
- ESNのエコーステート基準を見つける方法を理解し、それをLSMに適用します。

Readings - 読書:

- Spiking Neuron Model by W.Gerstener & W.Kistler
- What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? By W.Gerstener,
- Reservoir Computing - Echo State Networks and Liquid State Machines by V.Sankhala
- Scholarpedia Page about Echo State Network

Key Points - 重要なポイント:

- Equation of the ESN's internal state - ESNの内部状態の方程式 :

$$(1)\ x(n+1) = f(Wx(n) + W_{\text{in}}u(n+1) + W_{\text{fb}}y(n))$$

where $x(n)$ is the N-dimensional reservoir state, f is a sigmoid function (usually the logistic sigmoid or the $tanh$ function), W is the N × N reservoir weight matrix, W_{in} is the N × K input weight matrix, $u(n)$ is the K-dimensional input signal, W_{fb} is the N × L output feedback matrix, and $y(n)$ is the L-dimensional output signal.

ここで、$x(n)$ は N 次元のリザーバー状態で、f はシグモイド関数（通常はロジスティックシグモイドまたは $tanh$ 関数）、W は N x N のリザーバー重み行列、W_{in} は N x K の入力重み行列、$u(n)$ は K 次元の入力信号、W_{fb} は N x L の出力フィードバック行列、そして $y(n)$ は L 次元の出力信号です。

In tasks where no output feedback is required, W_{fb} is nulled. The extended system state $z(n) = \begin{bmatrix} x(n) \\ u(n) \end{bmatrix}$ at time n is the concatenation of the reservoir and input states.

出力フィードバックが不要なタスクでは、W_{fb} はゼロに設定されます。時刻 n における拡張システム状態 $z(n) = \begin{bmatrix} x(n) \\ u(n) \end{bmatrix}$ は、リザーバー状態と入力状態の連結です。

The output is obtained from the extended system state by - 出力は拡張システム状態から次のように得られます:

$$(2)\ y(n) = g\left(W_{\text{out}} z(n)\right)$$

where g is an output activation function (typically the identity or a sigmoid) and W_{out} is a L×(K+N)-dimensional matrix of output weights.

ここで、 g は出力活性化関数（通常は恒等関数またはシグモイド関数）であり、 W_{out} は L x (K+N) 次元の出力重み行列です。

- Explanation - 説明 :
  - The Echo State Property (ESP) is crucial for the effective functioning of Echo State Networks (ESNs), ensuring that the network responds only to recent inputs and not its initial state
  - エコーステートプロパティ (ESP) はエコーステートネットワーク (ESN) の効果的な機能に不可欠であり、ネットワークが初期状態ではなく最近の入力にのみ応答することを保証します
  - For the ESP to hold, the reservoir's connections, particularly with additive-sigmoid neurons, must meet specific algebraic conditions related to their singular values
  - ESPを維持するためには、特に加法的シグモイドニューロンを持つリザーバーの接続が、その特異値に関連する特定の代数条件を満たす必要があります
  - A key factor is the spectral radius, the largest absolute value of the reservoir weight matrix's eigenvalues. If it exceeds one, the network retains information from its initial state, violating the ESP
  - 重要な要素はスペクトル半径であり、これはリザーバー重み行列の固有値の最大絶対値です。これが1を超えると、ネットワークは初期状態の情報を保持し、ESPに違反します
  - Conversely, if the spectral radius is less than one, the ESP is usually maintained, enabling the network to forget initial conditions and focus on new inputs
  - 逆に、スペクトル半径が1未満であれば、通常ESPは維持され、ネットワークは初期条件を忘れ、新しい入力に集中することができます
  - This principle has led to a simplified but often incorrect belief that a spectral radius below one always ensures the ESP

- この原則は、スペクトル半径が1未満であれば常にESPが保証されるという簡略化された、しかししばしば誤った認識をもたらしました
- The relationship is more complex, as the ESP can hold with spectral radii slightly above one, depending on the input signal's characteristics and amplitude
- 関係はより複雑であり、ESPは入力信号の特性や振幅に応じて、スペクトル半径が1を少し上回っても維持されることがあります

- Studies show that even with spectral radii slightly above one, the ESP can be maintained, highlighting the need for careful parameter control to ensure the ESP in ESNs.
- 研究によると、スペクトル半径が1をわずかに上回っていてもESPが維持されることが示されており、ESNにおいてESPを確実にするためには、パラメータの慎重な制御が必要であることを強調しています

Conclusion - 結論 :

This week has been particularly productive. I have made significant strides in grasping core concepts such as why weights are chosen randomly in Echo State Networks (ESNs) to avoid the vanishing gradient problem and understanding the principles underlying the echo state property. While there are still several papers I need to delve into to fully comprehend the intricate details, my understanding has become much clearer. I plan to conduct tests in Python to practically apply and validate these theoretical insights.

今週は特に生産的でした。エコーステートネットワーク (ESN) で重みが無作為に選ばれる理由（消失勾配問題を回避するため）や、エコーステートプロパティの基本原理を理解する上で大きな進展がありました。まだ詳細を完全に理解するために読むべき論文は多くありますが、私の理解はかなり明確になってきました。これらの理論的な洞察を実際に適用して検証するために、Pythonでテストを行う予定です

Additionally, I have reviewed the planning for the IEEE WCCI 2024 conference in Yokohama, focusing on the topics and tutorials I intend to attend. This includes sessions on reservoir computing and computational neurology, which will help me understand these areas more comprehensively. Participating in the conference will be an excellent opportunity to expand my knowledge and network with experts in the field, ensuring I am well-prepared to efficiently absorb and understand each topic and aspect of AI presented.

さらに、横浜で開催されるIEEE WCCI 2024会議の計画を見直し、参加予定のトピックやチュートリアルに焦点を当てました。これには、リザーバーコンピューティングや計算神経学に関するセッションが含まれており、これらの分野をより包括的に理解するのに役立ちます。会議に参加することで、知識を広げ、分野の専門家とのネットワークを構築する絶好の機会となります。これにより、AIに関する各トピックと側面を効率的に吸収し、理解するための十分な準備を整えることができます。

Next Steps - 次のステップ :

- Refining my skills in working with ESNs and exploring their integration with LSMs
- ESNを扱うスキルをさらに磨き、LSM（液体状態マシン）との統合方法を探ります。

- Read the papers I found and study ESP in LSMs

- 発見した論文を読み、LSMにおけるエコーステートプロパティ（ESP）を詳しく研究します。

- Enhance my ReservoirPy proficiency
- ReservoirPyの熟練度を高め、効果的に活用できるようにします。

- Link my research with brain functioning and possible applications in Neurology
- 自分の研究を脳の機能と神経学に応用できる可能性を探ります。

- Prepare for WCCI conference : Finalize my plans for the IEEE WCCI conference, focusing on getting ready for sessions on reservoir computing, computational neurology, and other relevant AI topics.
- IEEE WCCI会議の計画を最終調整し、リザーバーコンピューティングや計算神経学に関するセッションに向けて準備を整えます。

G Annex 7: Weekly Report 7

Report n°7 : Studying LSM, ESN and ESP principle - 報告書第7号：LSM、ESN、ESPの原理を研究する
01/07/24 - 07/07/24

Work that have been done - 作業内容:

- Attending the IEEE WCCI Yokohama's conference on Computational Intelligence
- 横浜で開催されるIEEE WCCIの計算知能に関する会議に参加する

- Searching for empiric methods to determine the ESP in a ESN and LSM
- ESNおよびLSMにおけるESPを決定するための経験的方法を探す

Objectives - 目的:

- Discuss with masters in Neural Computation/Artificial Intelligence and create a network with Japanese and worldwide researchers and professionals
- 神経計算・人工知能の専門家と議論し、日本および世界中の研究者や専門家とネットワークを作る

- Understand how to find the echo state criterion of an ESN to apply it to a LSM to have a theoretical condition
- ESNのエコーステート基準を見つける方法を理解し、それをLSMに適用します。

- Try to first find an ESP empirically if it's too hard to find a theoretical condition
- 理論的な条件を見つけるのが難しい場合は、まず経験的にESPを見つけるようにする

Readings - 読書:

- Spiking Neuron Model by W.Gerstener & W.Kistler

- What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? By W.Gerstener,

- Reservoir Computing - Echo State Networks and Liquid State Machines by V.Sankhala

- Wikipedia page about Lyapunov Exponent

- Signals to Spikes for Neuromorphic Regulated Reservoir Computing and EMG Hand Gesture Recognition by N.Garg, I.Balafrej and Y.Beilliard

- GitHub repo EMG_exp

Key Points - 重要なポイント:

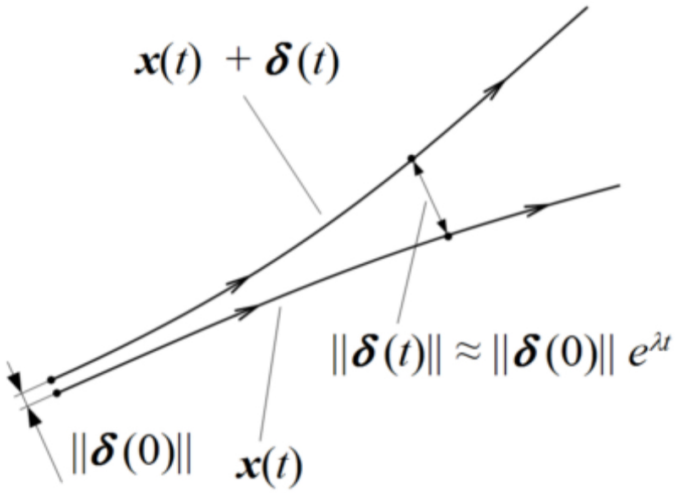
- Determine the ESP empirically - ESPを経験的に決定する：

- We can determine the ESP by giving the ESN or LSM the same Input and see if the output differs when we set the reservoir with different initial states.
- 同じ入力を与え、異なる初期状態でリザーバを設定した場合に出力が異なるかどうかを確認することで、ESNまたはLSMのESPを決定できます
- We also can use the Lyapunov Exponent - リャプノフ指数を使用することもできます：

The Lyapunov exponent is a quantity that characterise the separation of infinitesimally close trajectories. - リャプノフ指数は、微小に近い軌道の分離を特徴付ける量です。

Quantitatively, two trajectories in phase space with initial separation vector δZ_0 diverge (provided that the divergence can be treated within the linearized approximation) at a rate given by $| \delta Z(t) | \approx e^{\lambda t} | \delta Z_0 |$ where λ is the Lyapunov exponent.

定量的には、初期分離ベクトル δZ_0 を持つ位相空間内の2つの軌道は（発散が線形近似内で扱える場合に）リャプノフ指数 $| \delta Z(t) | \approx e^{\lambda t} | \delta Z_0 |$ によって与えられる速度でλ発散します。



We can find alpha proceeding that way :

$$|\delta Z(t)| \simeq e^{\lambda t} |\delta Z_0| \Leftrightarrow e^{\lambda t} \simeq \left| \frac{\delta Z(t)}{\delta Z_0} \right|$$

$$\Leftrightarrow \lambda t \simeq \ln \left| \frac{\delta Z(t)}{\delta Z_0} \right|$$

$$\Leftrightarrow \lambda \simeq \frac{1}{t} \ln \left| \frac{\delta Z(t)}{\delta Z_0} \right|$$

to keep the starting distance as infinitesimal, we must verify : $|\delta Z_0| \to 0$

So, we have

$$\lambda = \lim_{t \to \infty} \lim_{|\delta Z_0| \to 0} \frac{1}{t} \ln \left| \frac{\delta Z(t)}{\delta Z_0} \right|$$

Conclusion - 結論 :

This week has been unbelievable. I have met so many scientists during the conference and learnt about so many subjects and various topics. From BCI (Brain-Computer Interface) to reservoir computing and computational intelligence to prevent global warming or solve real societal problems. I have discussed with many famous scientists like Akira Hirose or Johan Suykens and many more. I also discussed my research with them, and they gave me some valuable advice. I have been thrilled to be a part of this event and I have expanded my network.

今週は信じられないほど素晴らしいものでした。会議中に多くの科学者に会い、多くの主題やさまざまなトピックについて学びました。BCI（脳コンピュータインターフェース）からリザーバーコンピューティングや地球温暖化を防止するための計算知能、あるいは実際の社会問題を解決するためのものまで。私は、広瀬章教授やヨハン・スイケンス教授など、多くの著名な科学者と話し合いました。また、自分の研究についても話し、彼らから貴重なアドバイスをいただきました。私はこのイベントの一員であることに興奮し、ネットワークを広げることができました。

Next Steps - 次のステップ :

For this week, I will focus on determining the ESP empirically using an LSM that I will create tomorrow. My goal is to check the changes in the output and in the Lyapunov exponent.

今週は、明日作成するLSMを使用してESPを経験的に決定することに集中します。私の目標は、出力とリャプノフ指数の変化を確認することです。

H Annex 8: Weekly Report 8

Report n°8 : Studying LSM, ESN and ESP principle - 報告書第7号：LSM、ESN、ESPの原理を研究する
08/07/24 - 14/07/24

Work that have been done - 作業内容:

- Implementing Brian Simulator and Nest Simulator to build an LSM
- ESNおよびLSMにおけるESPを決定するための経験的方法を探す

Objectives - 目的:

- Making an LSM
- LSMの作成
- Try to first find an ESP empirically
- まず経験的にESPを見つけてみてください

Readings - 読書:

- GitHub repo Brian Simulator
- GitHub repo Gratz University for LSM
- Signals to Spikes for Neuromorphic Regulated Reservoir Computing and EMG Hand Gesture Recognition by N.Garg, I.Balafrej and Y.Beilliard
- GitHub repo EMG_exp

Key Points - 重要なポイント:

1. Neuron Models and Synapses - ニューロンモデルとシナプス:

- NEST: Understand different neuron models (*iaf_psc_alpha, aeif_cond_exp*, etc.) and synapse models (*static_synapse, tsodyks_synapse*, etc.)
- NEST: 異なるニューロンモデル（iaf_psc_alpha、aeif_cond_expなど）やシナプスモデル（static_synapse、tsodyks_synapseなど）を理解する
- Brian: Review how neuron equations are defined and how synapses are implemented (*Synapses* class)
- Brian: ニューロン方程式の定義方法とシナプスの実装方法（Synapsesクラス）を確認する

2. Stimulus Generation - 刺激生成:

- Generating input spikes using Poisson processes
- ポアソン過程を使用して入力スパイクを生成する
- Creating different patterns for testing the LSM, such as XOR or more complex temporal patterns
- LSMをテストするために、XORやより複雑な時間的パターンなど、さまざまなパターンを作成する

3. State Recording and Readout Layer:

- NEST: Using *spike_recorder* to record spikes and process them
- 状態記録と読み出し層

- Brian: Utilizing *SpikeMonitor* to capture neuron states
- Brian: SpikeMonitorを利用してニューロンの状態をキャプチャする

4. Training and Evaluation - 訓練と評価 :

- Implementing linear regression or other machine learning techniques for the readout layer
- 読み出し層に対して線形回帰や他の機械学習技術を実装する
- Evaluating the performance using metrics like accuracy, precision, recall, etc
- 精度、適合率、再現率などの指標を使用してパフォーマンスを評価する

Conclusion - 結論 :

This week has been productive but not as much as I expected. I was expecting to implement an LSM model really quickly, but I was astonished by the complexity of the NEST simulator. That's why, in the first place, I wanted to start working using the Brian Simulator, but I rapidly understood that the NEST simulator was more comprehensive than Brian, and it would allow me to test the ESP criterion better.

However, I made significant progress in understanding the key features and functionalities of both the NEST and Brian simulators. By reviewing the code provided by the University of Graz and implementing an LSM class based on the principles outlined in the W. Maass paper, I was able to grasp the core concepts necessary for building and simulating neural networks.

今週は生産的でしたが、期待していたほどではありませんでした。LSMモデルをすぐに実装できると思っていましたが、NESTシミュレーターの複雑さに驚かされました。最初はBrianシミュレーターを使って作業を始めようとしましたが、NESTシミュレーターの方がBrianよりも包括的であり、ESP基準のテストをより効果的に行えることにすぐに気付きました。

しかし、NESTとBrianの両方のシミュレーターの主要な機能と特性を理解する上で大きな進展がありました。グラーツ大学から提供されたコードをレビューし、W. Maassの論文に基づいてLSMクラスを実装することで、ニューラルネットワークの構築とシミュレーションに必要な基本概念を把握することができました。

Next Steps - 次のステップ :

This week, I will focus on developing a proper LSM (Liquid State Machine) and experimenting with various settings to display the Lyapunov exponent. My goal is to refine the implementation and ensure that the LSM functions correctly. Additionally, I will explore different configurations and parameters to accurately compute and visualize the Lyapunov exponent, which is crucial for understanding the system's sensitivity to initial conditions and chaotic behavior.

今週は、適切なLSM（液体状態マシン）の開発に集中し、リアプノフ指数を表示するためのさまざまな設定をテストしてみる予定です。目標は、実装を改良してLSMが正しく機能するようにすることです。また、システムの初期条件やカオス的な挙動に対する感度を理解するために重要なリアプノフ指数を正確に計算し、視覚化するためのさまざまな構成やパラメーターを探索する予定です。

I Annex 9: Weekly Report 9

Report n°9 : Studying LSM, ESN and ESP principle - 報告書第7号：LSM、ESN、ESPの原理を研究する
15/07/24 - 21/07/24

Work that have been done - 作業内容:

- Implementing Brian Simulator and Nest Simulator to build an LSM
- ESNおよびLSMにおけるESPを決定するための経験的方法を探す
- Having an LSM model working properly
- LSMモデルが適切に動作すること

Objectives - 目的:

- Making an LSM
- LSMの作成
- Try to first find an ESP empirically
- まず経験的にESPを見つけてみてください
- Make a simple example using a xor function
- XOR関数を使用して簡単な例を作成する
- Making a better example using the Lorenz Function
- ローレンツ関数を使用してより良い例を作成する

Readings - 読書:

- GitHub repo Brian Simulator
- GitHub repo Gratz University for LSM
- Signals to Spikes for Neuromorphic Regulated Reservoir Computing and EMG Hand Gesture Recognition by N.Garg, I.Balafrej and Y.Beilliard
- GitHub repo EMG_exp

Key Points - 重要なポイント:

1. Neuron Models and Synapses - ニューロンモデルとシナプス:

- NEST: Understand different neuron models (*iaf_psc_alpha, aeif_cond_exp*, etc.) and synapse models (*static_synapse, tsodyks_synapse*, etc.)
- NEST: 異なるニューロンモデル（iaf_psc_alpha、aeif_cond_expなど）やシナプスモデル（static_synapse、tsodyks_synapseなど）を理解する
- Brian: Review how neuron equations are defined and how synapses are implemented (*Synapses* class)
- Brian: ニューロン方程式の定義方法とシナプスの実装方法（Synapsesクラス）を確認する

2. Stimulus Generation - 刺激生成:

- Generating input spikes using Poisson processes
- ポアソン過程を使用して入力スパイクを生成する
- Creating different patterns for testing the LSM, such as XOR or more complex temporal patterns

- LSMをテストするために、XORやより複雑な時間的パターンなど、さまざまなパターンを作成する

3. State Recording and Readout Layer:

- NEST: Using *spike_recorder* to record spikes and process them
- 状態記録と読み出し層
- Brian: Utilizing *SpikeMonitor* to capture neuron states
- Brian: SpikeMonitorを利用してニューロンの状態をキャプチャする

4. Training and Evaluation - 訓練と評価 :

- Implementing linear regression or other machine learning techniques for the readout layer
- 読み出し層に対して線形回帰や他の機械学習技術を実装する
- Evaluating the performance using metrics like accuracy, precision, recall, etc
- 精度、適合率、再現率などの指標を使用してパフォーマンスを評価する

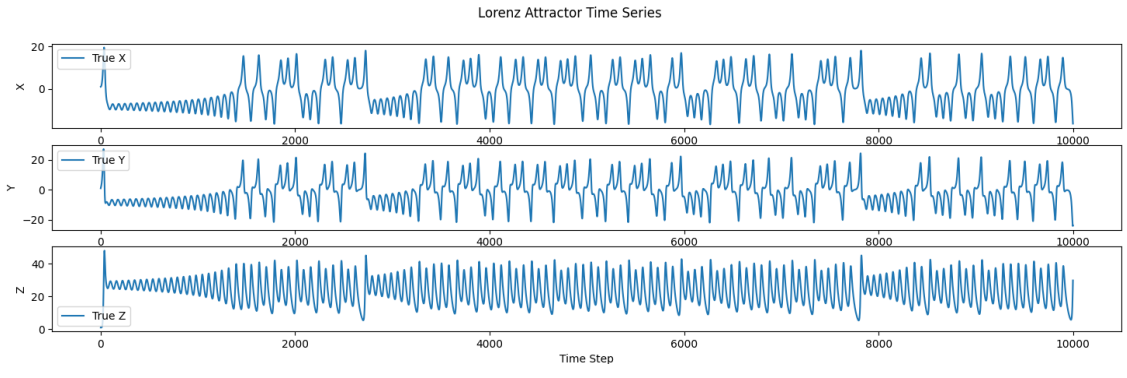
Testing the LSM -LSMのテスト:

1. Objective - 目的 :

- The goal was to test two different Liquid State Machine (LSM) models using the NEST simulator.
- 目標は、NESTシミュレーターを使用して2つの異なるリキッドステートマシン（LSM）モデルをテストすることでした。
- The first model was tested with data generated from a simple XOR function.
- 最初のモデルは、単純なXOR関数から生成されたデータでテストされました
- The second model was tested with data generated from the Lorenz attractor, a classic example of chaotic dynamics.
- 2つ目のモデルは、カオス力学の古典的な例であるローレンツアトラクタから生成されたデータでテストされました

2. Data Generation - データ生成:

- For the XOR function, I used the example provided by the LSM repository of the University of Graz. This example generated binary input states and corresponding output targets that represent the XOR logic gate.
- XOR関数については、グラーツ大学のLSMリポジトリで提供されている例を使用しました。この例では、XOR論理ゲートを表すバイナリ入力状態と対応する出力ターゲットが生成されました
- For the Lorenz attractor, I had to code the data generation from scratch. I used the solve_ivp function to generate a time series representing the chaotic behavior of the Lorenz system.
- ローレンツアトラクタについては、データ生成をゼロからコード化する必要がありました。solve_ivp関数を使用して、ロー レンツシステムのカオス的な振る舞いを表す時系列データを生成しました。



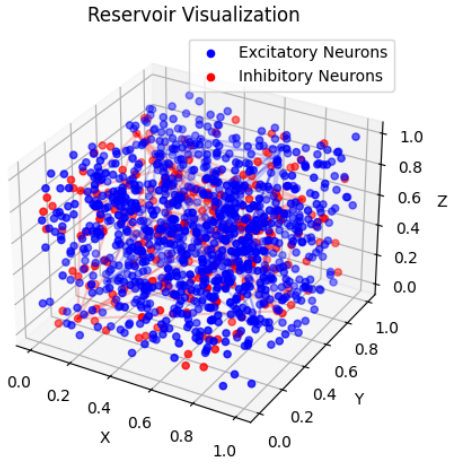
3. Spike Time Array Creation - スパイクタイム配列の作成:

- For both models, I transformed the generated data into arrays of spike times:
- 両方のモデルについて、生成されたデータをスパイクタイムの配列に変換しました。
  - For the XOR function, I generated spikes corresponding to the binary states using a fixed time grid
  - XOR関数では、固定時間グリッドを使用してバイナリ状態に対応するスパイクを生成しました。

  - For the Lorenz attractor, I scaled the continuous data and used a Poisson process to convert it into spike trains.
  - ローレンツアトラクタでは、連続データをスケーリングし、ポアソン過程を使用してスパイク列に変換しました。

4. LSM Initialization - LSMの初期化:

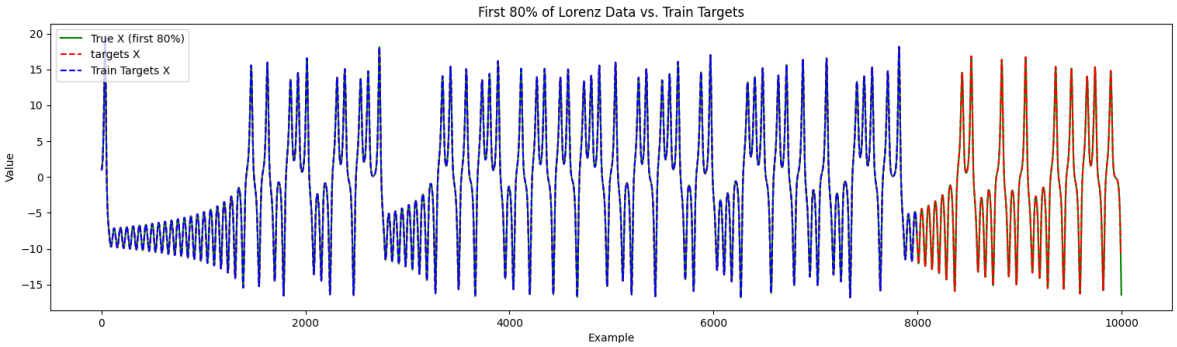
- Initialized the LSM with a specified number of excitatory, inhibitory, and recurrent neurons.
- 指定された数の興奮性ニューロン、抑制性ニューロン、および再帰性ニューロンでLSMを初期化しました。
- Set up the connectivity patterns and synapse models within the LSM.
- スパイク時刻をこれらのジェネレーターに入力し、それらをLSMの入力ニューロンに接続しました。




```
# Define parameters
n_exc = 1000  # Number of excitatory neurons
n_inh = 250   # Number of inhibitory neurons
n_rec = 500   # Number of recurrent neurons to record

sim_time = 200000  # Simulation time in ms
stim_interval = 300  # Interval between stimuli
stim_length = 50  # Length of each stimulus
stim_rate = 200  # [1/s]
readout_delay = 10  # Delay before readout
```

5. Feeding Data into LSM - データをLSMに入力する:

- For each input data set (XOR and Lorenz), I created spike generators in the NEST simulator.
- 各入力データセット（XORおよびLorenz）に対して、NESTシミュレーターでスパイクジェネレーターを作成しました。
- I fed the spike times into these generators and connected them to the input neurons of the LSM.
- スパイク時刻をこれらのジェネレーターに入力し、それをLSMの入力ニューロンに接続しました。

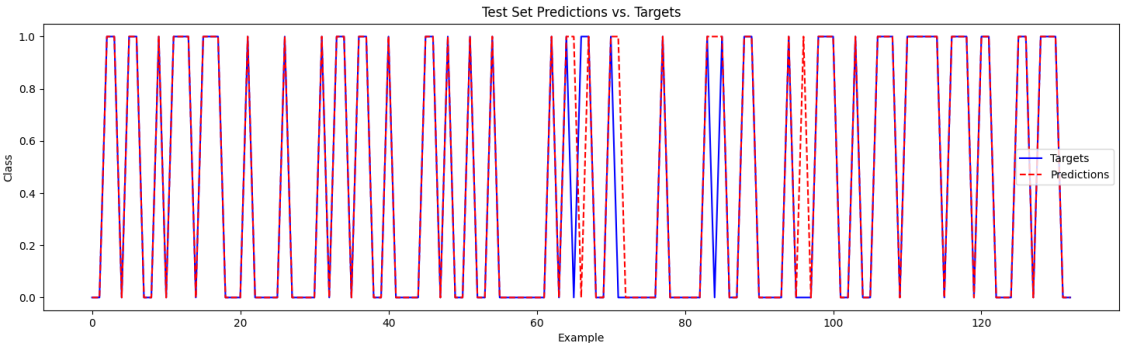


6. Simulation - シミュレーション:

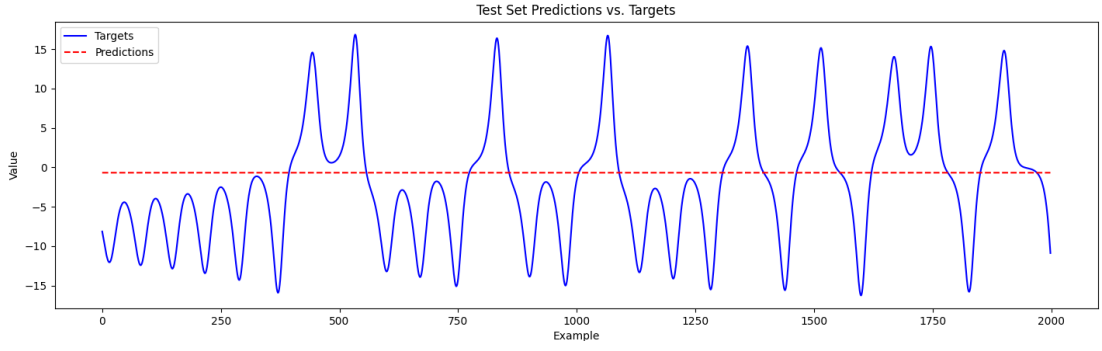
- Simulated the LSM for a defined period to process the input spike trains.
- 入力スパイク列を処理するために、LSMを一定期間シミュレーションしました。
- Used the states of the recurrent neurons as the basis for the readout layer.
- リカレントニューロンの状態をリードアウト層の基礎として使用しました。

7. Readout and Evaluation - リードアウトと評価:

- For the XOR model, the output of the readout layer was evaluated against the XOR targets to measure the model's accuracy.
- XORモデルでは、リードアウト層の出力をXORターゲットと比較し、モデルの精度を評価しました



- For the Lorenz model, the readout layer's predictions were compared to the true Lorenz attractor data to evaluate the model's predictive performance.
- ローレンツモデルでは、リードアウト層の予測を真のローレンツアトラクタデータと比較し、モデルの予測性能を評価しました。



8. Results Visualization - 結果の可視化:

- Plotted the predictions against the true targets for both models to visually assess their performance.
- 両方のモデルについて、予測結果と真のターゲットをプロットし、視覚的にパフォーマンスを評価しました。
- Visualized the spike times and the reservoir's neuron activity to better understand the internal dynamics of the LSM.
- スパイク時間とリザーバーのニューロン活動を可視化し、LSMの内部動態をよりよく理解しました。

Conclusion - 結論 :

This week has been productive and informative, although some problems remain unresolved, particularly with making the Lorenz model work properly. I was expecting to implement an LSM model quickly, but I was astonished by the complexity of the NEST simulator. Initially, I wanted to start working with the Brian Simulator, but I rapidly understood that the NEST simulator was more comprehensive than Brian, and it would allow me to test the ESP criterion better.

However, I made significant progress in understanding the key features and functionalities of both the NEST and Brian simulators. By reviewing the code provided by the University of Graz and implementing an LSM class based on the principles outlined in the W. Maass paper, I was able to grasp the core concepts necessary for building and simulating neural networks.

Despite the challenges, particularly with the Lorenz model, this week has been highly informative. I gained a lot of knowledge and understanding about the practical functioning of Liquid State Machines and their use with the NEST simulator. I now feel more confident in tackling the remaining issues and further developing my LSM models.

今週は生産的で有意義な週でしたが、特にローレンツモデルの適切な動作に苦労しているため、まだ解決されていない問題もあります。LSMモデルをすぐに実装できると思っていましたが、NESTシミュレーターの複雑さに驚かされました。最初はBrianシミュレーターを使って作業を始めようとしましたが、NESTシミュレーターの方がBrianよりも包括的であり、ESP基準のテストをより効果的に行えることにすぐに気付きました。

しかし、NESTとBrianの両方のシミュレーターの主要な機能と特性を理解する上で大きな進展がありました。グラーツ大学から提供されたコードをレビューし、W. Maassの論文に基づいてLSMクラスを実装することで、ニューラルネットワークの構築とシミュレーションに必要な基本概念を把握することができました。

ローレンツモデルに関する課題はありますが、今週は非常に有意義でした。液体状態マシンの実際の機能とNESTシミュレーターの使用方法について多くの知識と理解を得ることができました。残りの問題に取り組み、LSMモデルをさらに発展させる自信がつきました。

Next Steps - 次のステップ :

Next week, I will focus on having the Lorenz model working properly to test the ESP criterion on both the XOR and Lorenz models.

来週は、ローレンツモデルを適切に動作させ、XORモデルとローレンツモデルの両方でESP基準をテストすることに集中します。

J Annex 10: Weekly Report 10

Report n°10 : Studying LSM, ESN and ESP principle - 報告書第10号：LSM、ESN、ESPの原理を研究する
22/07/24 - 28/07/24

Work that have been done - 作業内容:

- Implementing Brian Simulator and Nest Simulator to build an LSM
- ESNおよびLSMにおけるESPを決定するための経験的方法を探す
- Having an LSM model working properly
- LSMモデルが適切に動作すること
- Searching for different libraries to create the model
- 異なるライブラリを探してモデルを作成する

Objectives - 目的:

- Making an LSM
- LSMの作成
- Try to first find an ESP empirically
- まず経験的にESPを見つけてみてください
- Making a better example using the Lorenz Function
- ローレンツ関数を使用してより良い例を作成する

Readings - 読書:

- GitHub repo Brian Simulator
- GitHub repo Gratz University for LSM
- Signals to Spikes for Neuromorphic Regulated Reservoir Computing and EMG Hand Gesture Recognition by N.Garg, I.Balafrej and Y.Beilliard
- GitHub repo EMG_exp

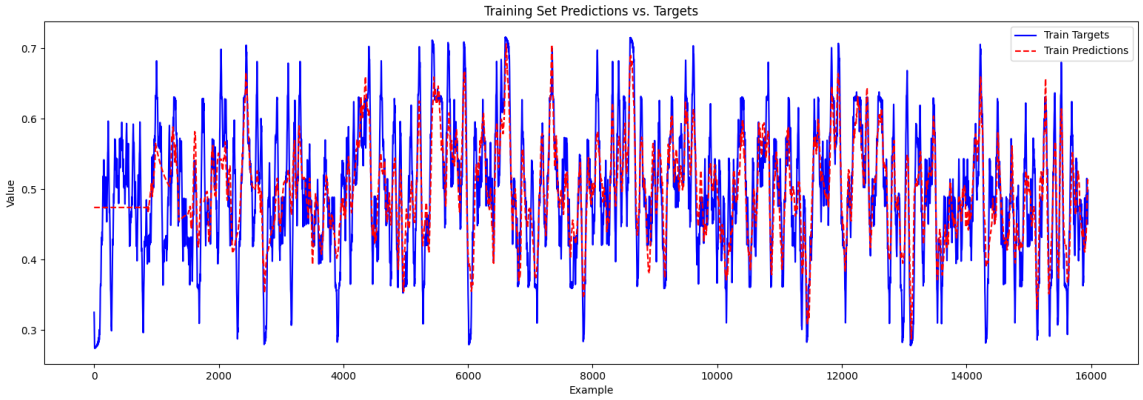
Key Points - 重要なポイント:

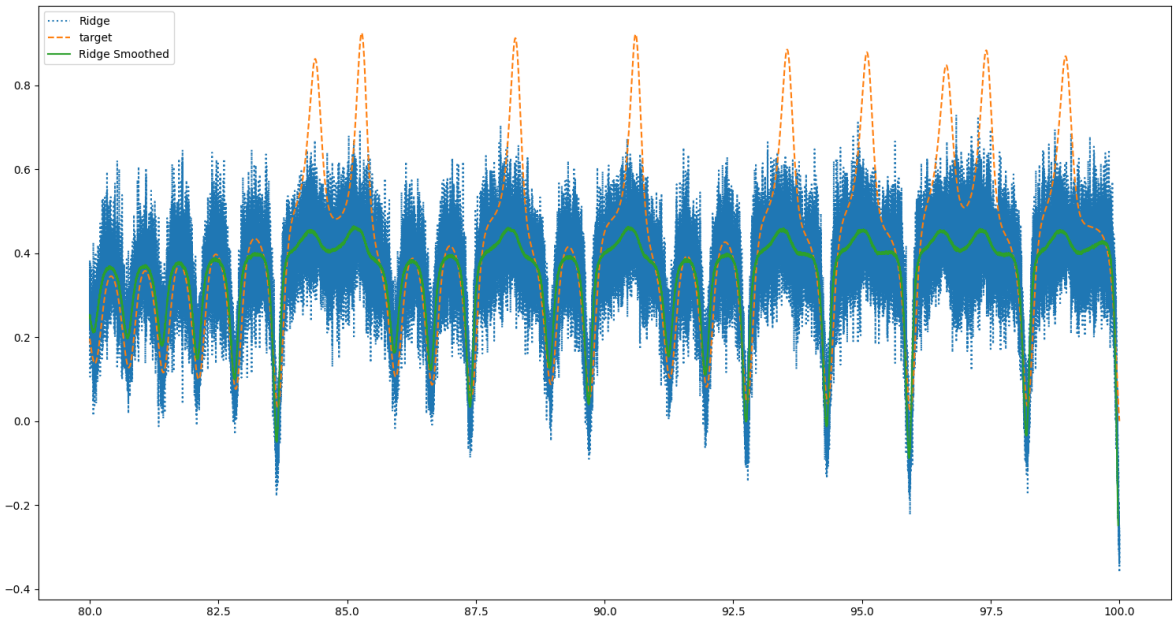
1 - Two way for data representations - データ表現の二つの方法 :

- Spike Generator : The spike_generator generates discrete spikes representing the moments when neurons emit action potentials. In this case, data is represented by individual spike times for each neuron. This means that the code must handle the generation and synchronization of spikes according to the input values. Using a spike_generator more realistically mimics biological neuron models, but it can make model training more complex, and the results may be less precise due to the noise introduced by the discrete spikes.
- スパイクジェネレーター: spike_generatorはニューロンが活動電位を発する瞬間を表す離散的なスパイクを生成します。この場合、データは各ニューロンのスパイク時間によって表されます。これは、コードが入力値に応じたスパイクの生成と同期を処理する必要があることを意味します。スパイクジェネレーターを使用すると、生物学的ニューロンモデルをよりリアルに模倣しますが、モデルのトレーニングがより複雑になり、離散的なスパイクによって導入されるノイズのために結果があまり正確でない可能性があります。

- Step Current Generator : The step_current_generator applies continuous currents to neurons, where the current amplitudes are proportional to the input data values. In this case, data is represented by continuous values, simplifying input management in the code. This type of generator tends to produce smoother and more accurate results, as it eliminates the noise associated with discrete spikes. However, it diverges from the realistic dynamics of biological neural networks.

- ステップカレントジェネレーター: step_current_generatorはニューロンに連続した電流を適用し、電流の振幅が入力データの値に比例します。この場合、データは連続値で表され、コード内の入力管理が簡素化されます。このタイプのジェネレーターは、離散的なスパイクに関連するノイズを排除するため、より滑らかで正確な結果を生成する傾向があります。ただし、生物学的な神経ネットワークのリアルな動態からは逸脱します。





Conclusion - 結論 :

This week has been productive and informative, although I am not satisfied with my results using the Lorenz function. Despite my efforts, the model did not perform as expected, and the predictions did not align well with the actual data. This has led me to the decision to stop using NEST for my simulations. The complexity and lack of transparency in NEST make it feel like a black box, which hinders my ability to fully understand and control the model. Moving forward, I plan to explore other simulation tools that offer more clarity and flexibility.

ローレンツ関数を使用した結果に満足していません。努力にもかかわらず、モデルは期待通りに機能せず、予測が実際のデータとよく一致しませんでした。このため、NESTを使用したシミュレーションを中止することにしました。NESTの複雑さと透明性の欠如は、まるでブラックボックスのように感じられ、モデルを完全に理解し制御することが困難です。今後は、より明確で柔軟なシミュレーションツールを探していく予定です。

Next Steps - 次のステップ :

This week, I will be working on creating a Liquid State Machine (LSM) by myself. Despite my previous challenges with the Lorenz function and the limitations I encountered using the NEST simulator, I am determined to develop an LSM independently. I believe this approach will provide me with a deeper understanding of the mechanics and functionalities of LSMs, allowing me to have full control over the model. My goal is to overcome the obstacles faced so far and achieve better results through hands-on experimentation and learning.

今週は、自分自身でリキッドステートマシン（LSM）の作成に取り組みます。以前、ローレンツ関数の使用で直面した課題やNESTシミュレーターの限界にもかかわらず、独自にLSMを開発する決意をしました。このアプローチにより、LSMの仕組みと機能をより深く理解し、モデルを完全に制御できるようになると信じています。これまで直面してきた障害を克服し、実践的な実験と学習を通じてより良い結果を達成することを目指しています。