Information processing in soft robots

Kohei Nakajima
University of Tokyo

2023/11/03 Friday, 16:20-17:50
立命館大学：特殊講義（ソフトロボット学）
zoom

Contents

1. 11/3: Reservoir computing

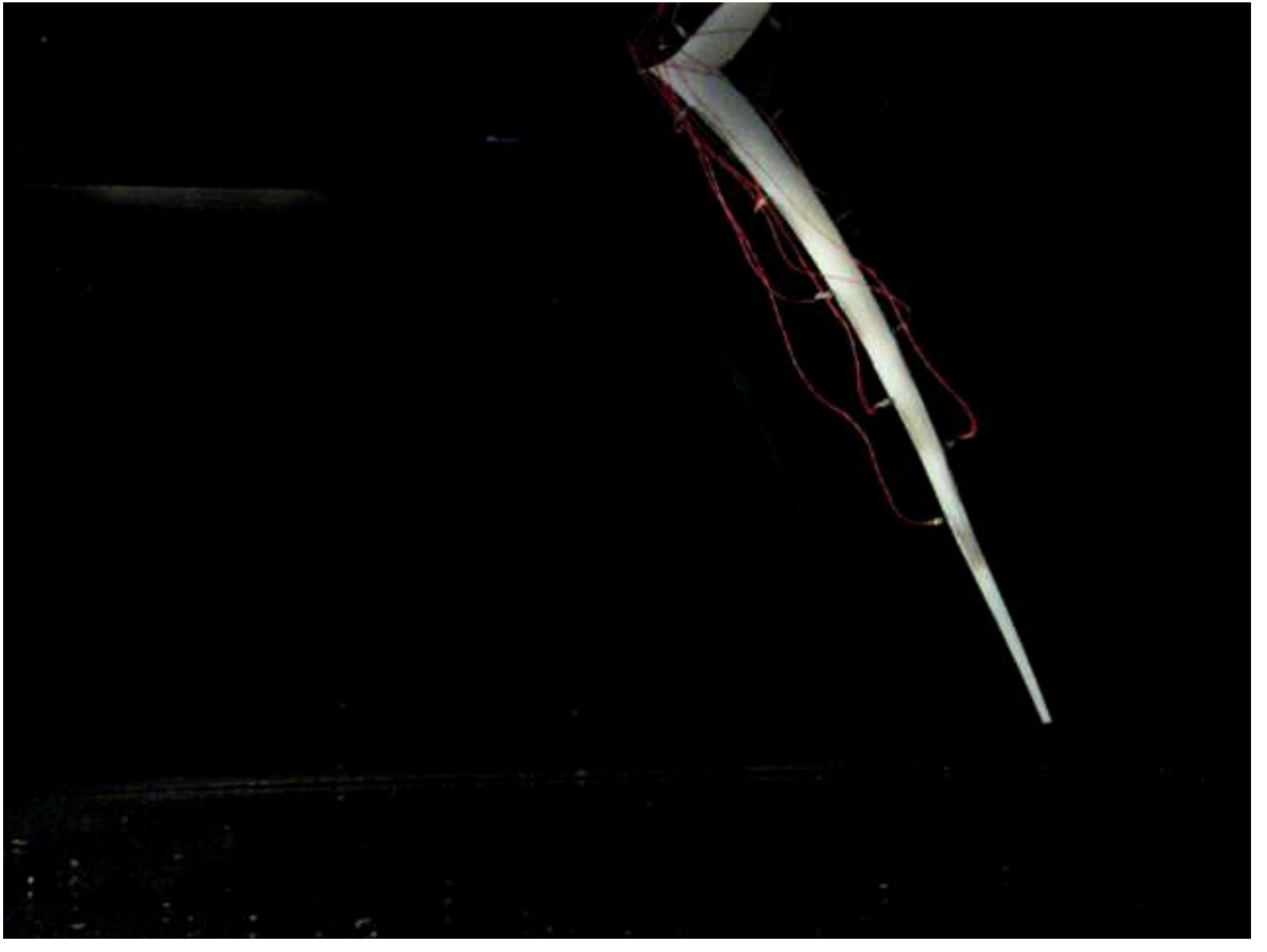
   Nonlinear dynamics as computational device

2. 11/10: Physical reservoir computing

   Soft body dynamics as computational device

The report topic will be announced at the final lecture (11/10)!

Computing with soft body dynamics?



Where I'm from…

(education)

Ph. D. The University of Tokyo, 2009.
M. S. The University of Tokyo, 2006.
B. S. The University of Tokyo, 2004.

(major)
nonlinear dynamics, chaos theory, recurrent neural network, embodiment.

Takashi Ikegami

(research experience)

2009 **Postdoctoral Researcher** (EU project: OCTOPUS)
Department of Informatics,
University of Zürich

2013 **JSPS Postdoctoral Fellow**
ETH Zürich

2014 **Assistant Professor**
The Hakubi Center for Advanced Research,
Kyoto University
(from 2015, 10, JST PRESTO Researcher)

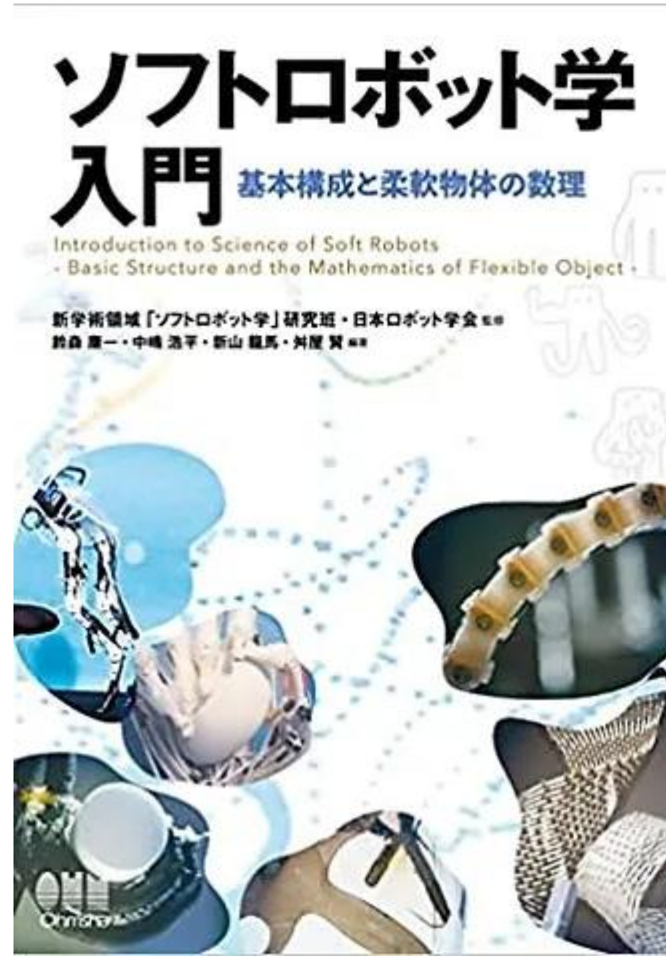
2017 **Project Associate Professor**
The University of Tokyo

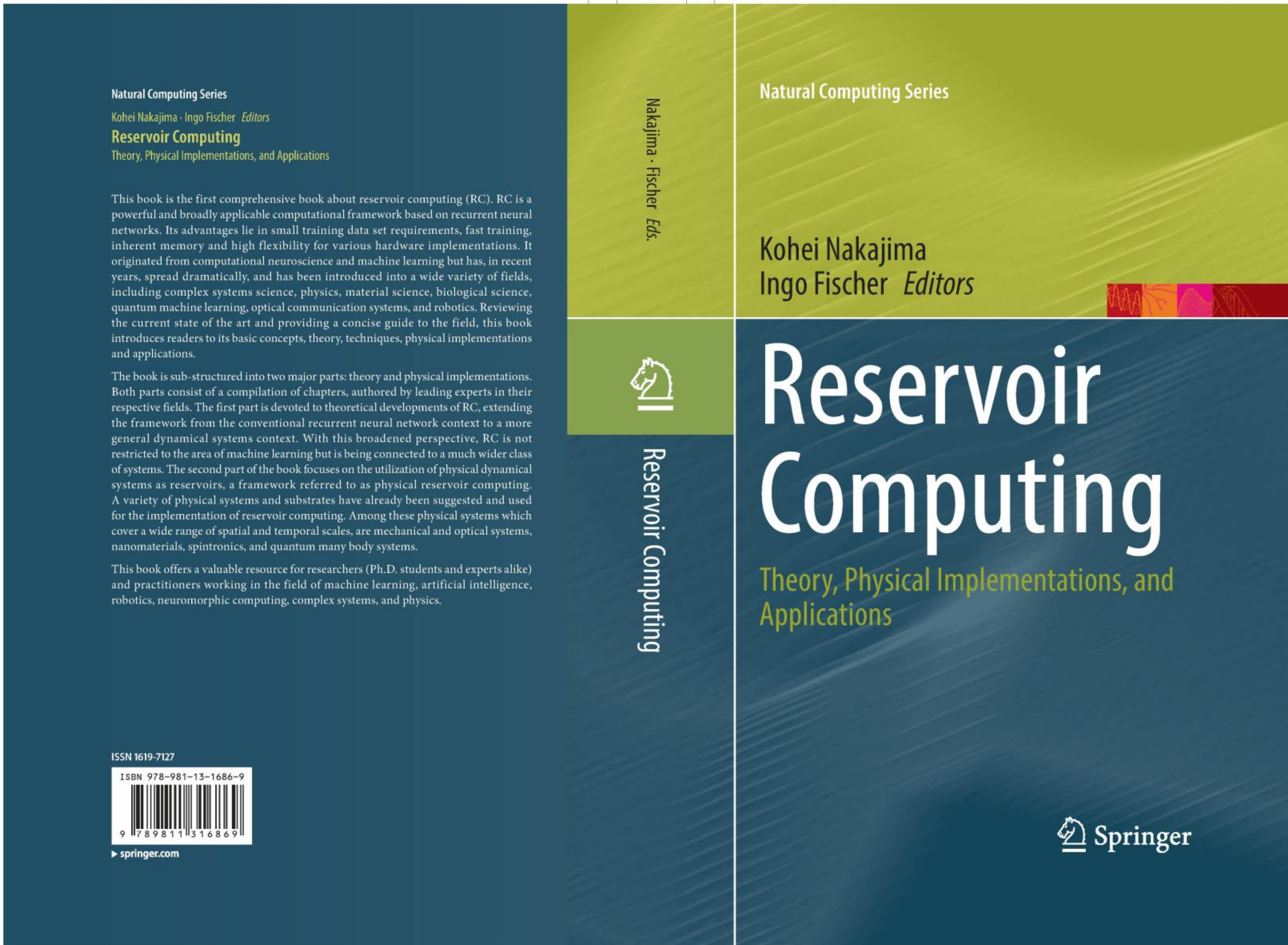
2020 **Associate Professor**
The University of Tokyo

Rolf Pfeifer

(topic)
soft robotics, morphological computation, (physical) reservoir computing.

Soft robotics text books




Natural Computing Series
Kohei Nakajima · Ingo Fischer Editors
Reservoir Computing
Theory, Physical Implementations, and Applications
This book is the first comprehensive book about reservoir computing (RC). RC is a powerful and broadly applicable computational framework based on recurrent neural networks. Its advantages lie in small training data set requirements, fast training, inherent memory and high flexibility for various hardware implementations. It originated from computational neuroscience and machine learning but has, in recent years, spread dramatically, and has been introduced into a wide variety of fields, including complex systems science, physics, material science, biological science, quantum machine learning, optical communication systems, and robotics. Reviewing the current state of the art and providing a concise guide to the field, this book introduces readers to its basic concepts, theory, techniques, physical implementations and applications.
The book is sub-structured into two major parts: theory and physical implementations. Both parts consist of a compilation of chapters, authored by leading experts in their respective fields. The first part is devoted to theoretical developments of RC, extending the framework from the conventional recurrent neural network context to a more general dynamical systems context. With this broadened perspective, RC is not restricted to the area of machine learning but is being connected to a much wider class of systems. The second part of the book focuses on the utilization of physical dynamical systems as reservoirs, a framework referred to as physical reservoir computing. A variety of physical systems and substrates have already been suggested and used for the implementation of reservoir computing. Among these physical systems which cover a wide range of spatial and temporal scales, are mechanical and optical systems, nanomaterials, spintronics, and quantum many body systems.
This book offers a valuable resource for researchers (Ph.D. students and experts alike) and practitioners working in the field of machine learning, artificial intelligence, robotics, neuromorphic computing, complex systems, and physics.
ISSN 1619-7127
ISBN 978-981-13-1686-9
springer.com
Nakajima · Fischer Eds.
Reservoir Computing
Natural Computing Series
Kohei Nakajima
Ingo Fischer Editors
Reservoir Computing
Theory, Physical Implementations, and Applications
Springer

RC textbook

Octopus arm computer

日経サイエンス「人工知能」（2020年6月）


特集　AIの身体性

体で計算するコンピューター

タコは脳を使わず，足だけで複雑な運動を制御する
同様に生物の体などの物理系の動きを利用して計算し
ローコストで機械学習を実行する新たな仕組みが登場した

古田彩（編集部）
協力：中嶋浩平（東京大学）

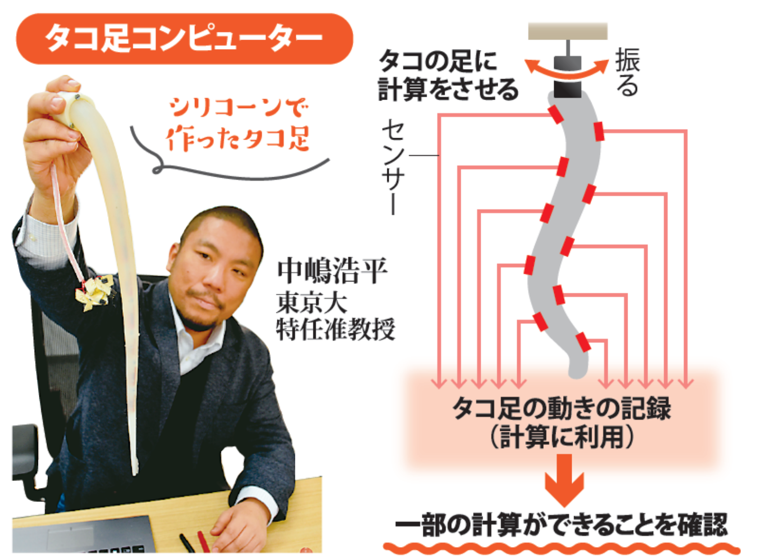
タコ足コンピューター　タコの足は脳を介さず，足だけで複雑な運動をこなす。人工のタコ足も，自らの複雑な動きを使って，様々な計算方法を学習する。

SWI swissinfo.ch　Swiss perspectives in 10 languages

Swissinfo
2013

Next generation of robots will have a gentle touch

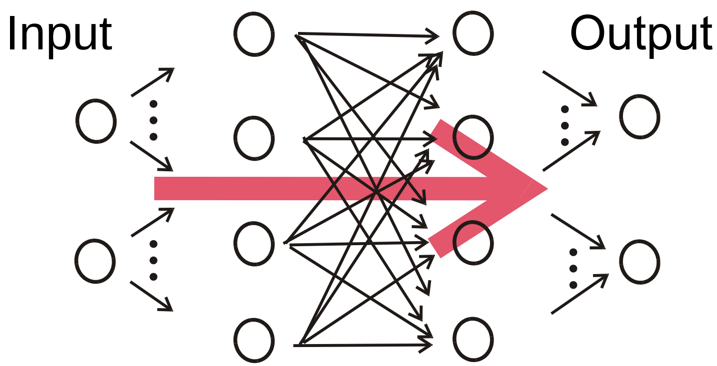
毎日新聞2020



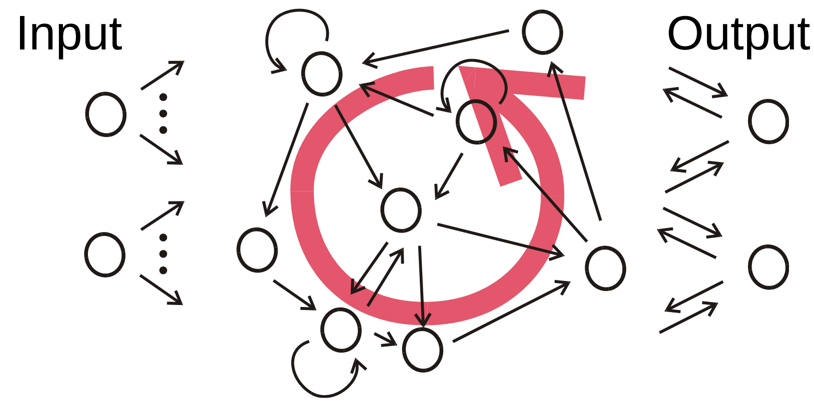
Reservoir computing: basics

Recurrent Neural Networks

Feedforward NN (FNN) vs Recurrent NN (RNN)

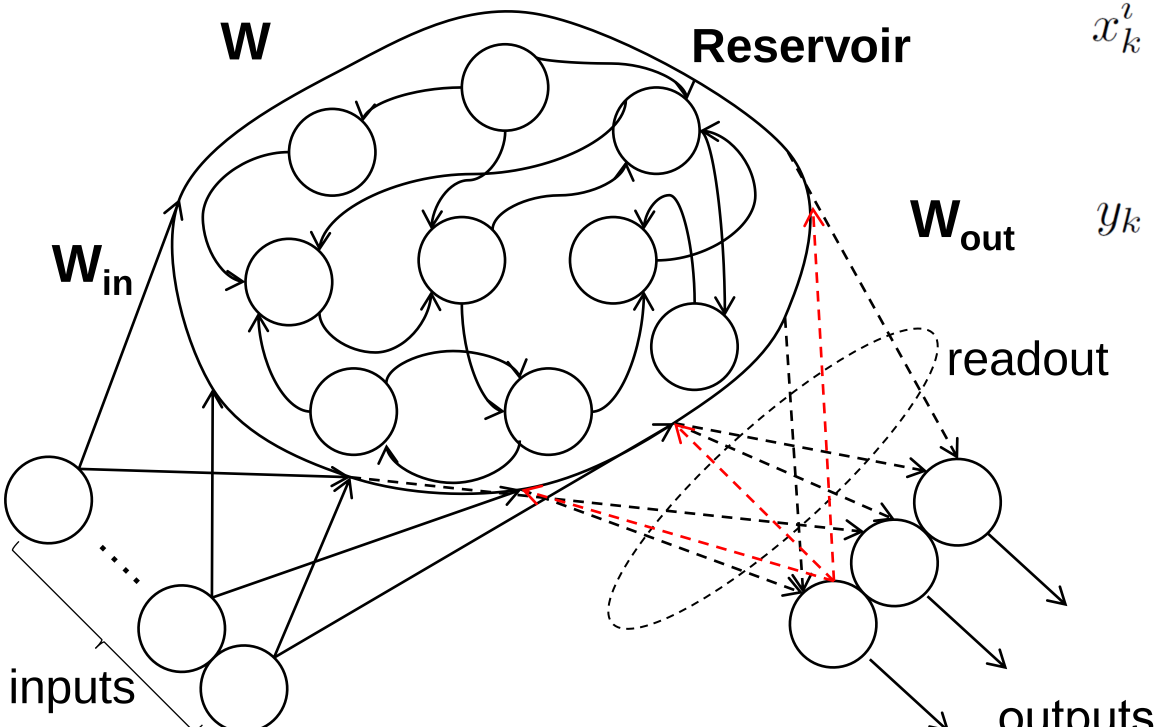


- Activation is fed forward from input to output via hidden layers
- Can approximate arbitrary nonlinear static maps with arbitrary precision
- Static (e.g., image processing)



- Has at least one cyclic path in synaptic connections (memory)
- Can approximate arbitrary nonlinear dynamical systems with arbitrary precision
- Dynamic (e.g., prediction tasks for time series)

Reservoir computing: basic settings



H. Jaeger et al., *Science*, Vol.304. no.5667, pp.78–80 (2004).

$$x_k^i = f(\sum_{j=1}^{M} w^{ij} x_{k-1}^j + w_{in}^i u_k)$$

$$y_k = \sum_{i=0}^{M} w_{out}^i x_k^i, \quad f(x) = \tanh(x)$$

Adjust only the readout!

$$W_{out} = (X^T X)^{-1} X^T y$$

Use W_{out} for information processing!

$$\hat{y} = X W_{out}$$

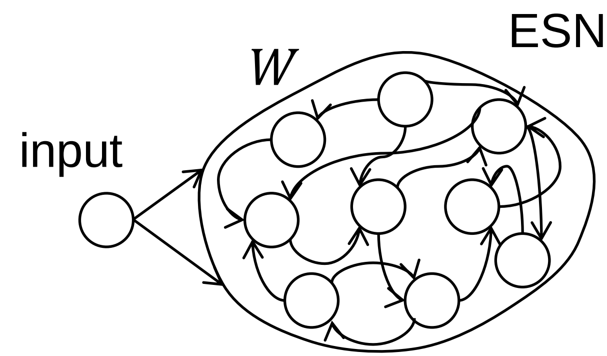
(Good points)

- Learning is fast and stable!
- No local minimum problem!
- Feasible for physical platform.

(Computational power)

- **Nonlinearity**
- **Memory**

Echo-state network (ESN)



- number of nodes : N
- state of neuron i at t : $x_i(t)$
- action potential of neuron i at t : $a_i(t)$
- input : $u(t)$
- internal weights : w_{ij}
- input weights : w_{in}
- activation function : $f(a) = \tanh(a)$
- dynamics :

spectral radius of W

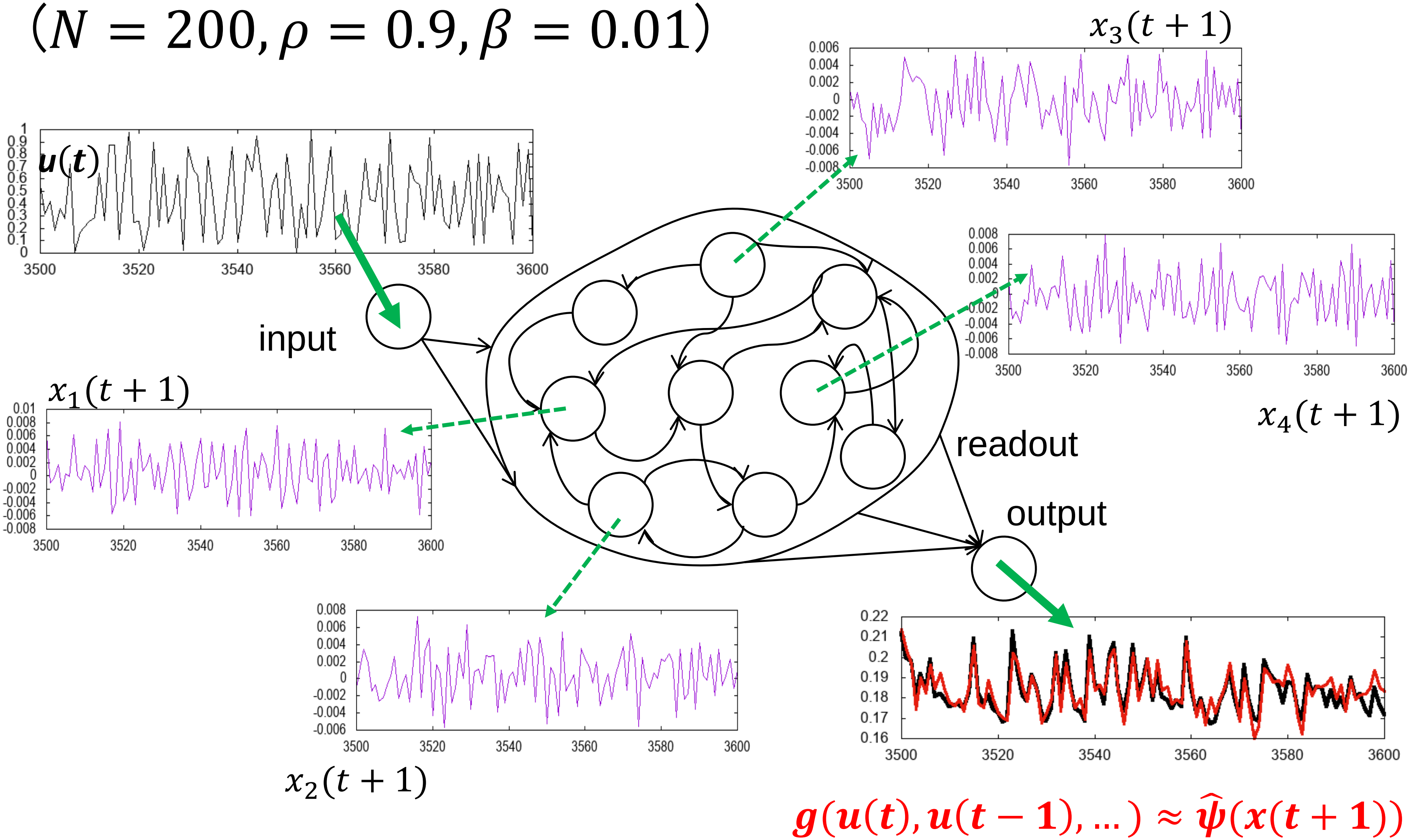
$$\rho(W) = \max\{|\lambda_1|, \ldots, |\lambda_N|\}$$

$$x_i(t+1) = f\big(a_i(t)\big), \quad a_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{in} u(t)$$

Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach (Vol. 5, p. 01). Bonn: GMD-Forschungszentrum Informationstechnik.

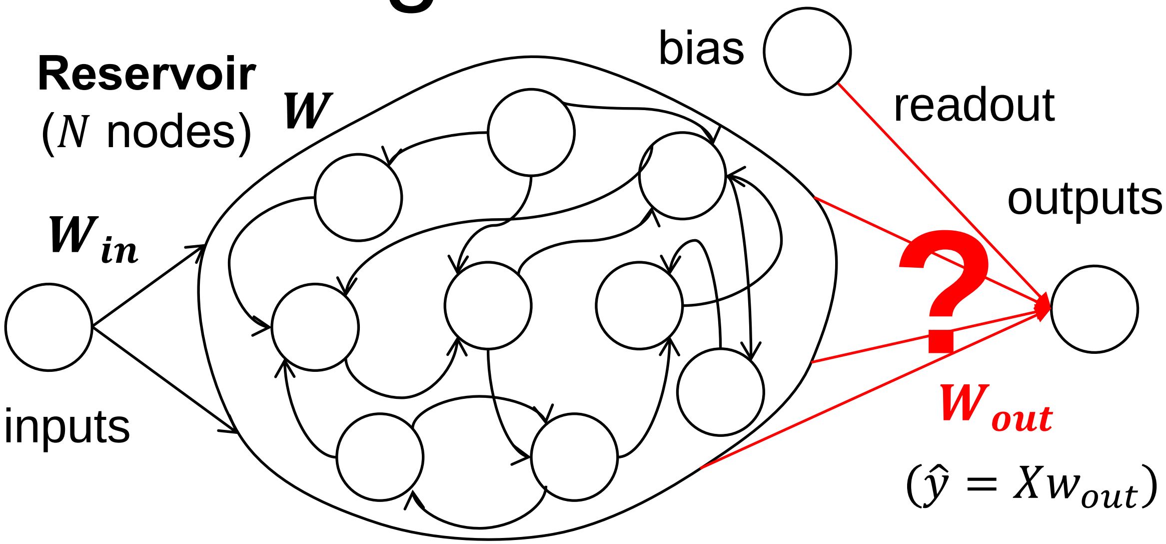
Example of ESN dynamics

$(N = 200, \rho = 0.9, \beta = 0.01)$



- Diverse dynamics within the reservoir!

Linear regression



N+1 nodes,
k sample data,
corresponding
target data is
obtained!

Data: $X = \begin{pmatrix} 1 & x_1^1 & \cdots & x_1^N \\ \vdots & & \ddots & \vdots \\ 1 & x_k^1 & \cdots & x_k^N \end{pmatrix}$ Target: $y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix}$

Weight: $w_{out} = \begin{pmatrix} w_{out}^0 \\ w_{out}^1 \\ \vdots \\ w_{out}^N \end{pmatrix}$

$$\mathrm{L} = \frac{1}{2}\sum_i (y_i - X_i w_{out})^2$$
$$= \frac{1}{2}\|y - Xw_{out}\|^2$$

Want to minimize !

$$\mathrm{L} = \frac{1}{2}\|y - Xw_{out}\|^2 = \frac{1}{2}(y - Xw_{out})^T(y - Xw_{out})$$

$$= \frac{1}{2}\left(y^T y - y^T X w_{out} - w_{out}^T X^T y + w_{out}^T X^T X w_{out}\right)$$

$$\frac{\partial L}{\partial w_{out}} = \frac{1}{2}\left(-2X^T y + 2X^T X w_{out}\right) = 0$$

$$\Rightarrow \quad \boxed{w_{out} = \left(X^T X\right)^{-1} X^T y}$$

(note)

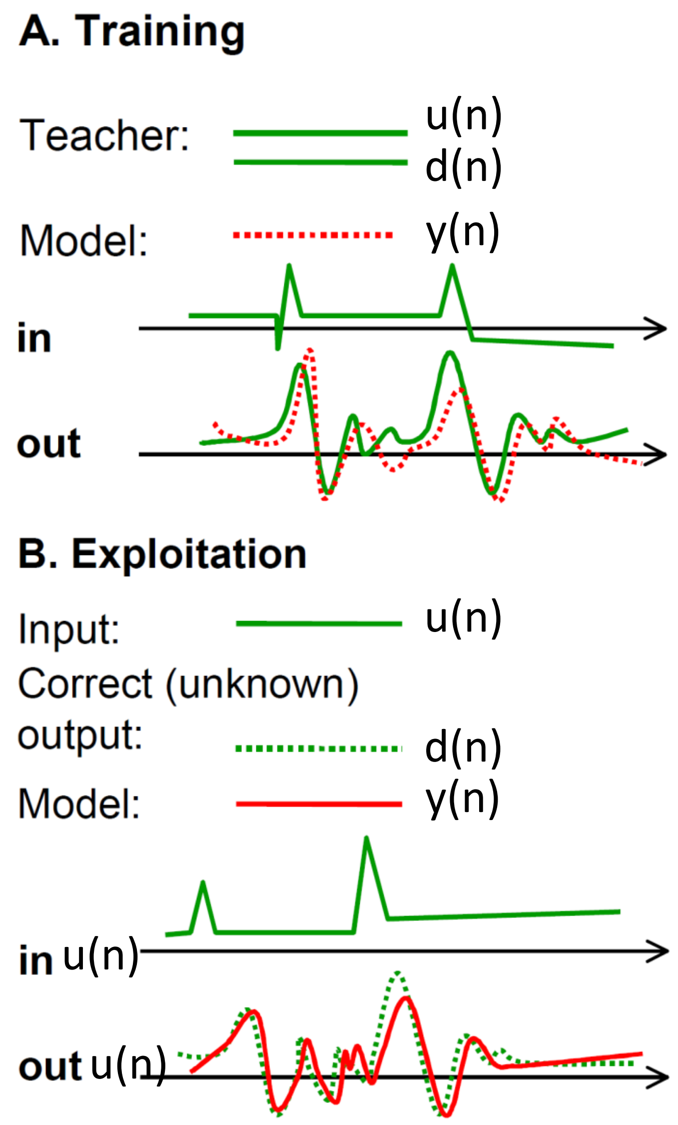
- Ridge regression is often used to avoid overfitting.
- Online learning scheme, such as recursive least squares, can be also used.

Typical learning procedures

Step 1. Data collection: Empirically observe or artificially construct input-output time series (u(n), d(n)), n = 1, 2, …, T as teacher/training data and collect corresponding reservoir states

Step 2. Training phase: Utilize teacher data to train a readout such that its output y(n) precisely reproduces/fits d(n)

Step 3. Test phase: Evaluate the generalization of the trained system, i.e., when it receives a different input sequence u(k) from the training input sequence u(n) by comparing an output y(k) with the target output d(k).



Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach (Tech. Rep. No. 159). Bremen: German National Research Center for Information Technology.

Two representative models in RC

Echo state network

H. Jaeger, Tech. Rep. No. 148. Bremen: German National Research Center for Information Technology (2001).

H. Jaeger et al., Science, Vol.304. no.5667, pp.78–80 (2004).

Liquid state machine

W. Maass et al., Neural Comput 14 (11): 2531–60, 2002.

W. Maass, & H. Markram, H. Journal of computer and system sciences, 69(4), 593-616, 2004.

Herbert Jaeger

- Randomly coupled network
- Artificial neural network (Sigmoidal function)
- Engineering oriented

∗ Similar architectures can be found at least in 1990.

Jaeger, H. (2021). In Reservoir Computing. Springer Nature.

Early 2000

Wolfgang Maass

- Often assume space
- Pulse neuron
- Neuroscience oriented

Conception in around 2005!

Let us unify the approach in the same umbrella!

Reservoir computing

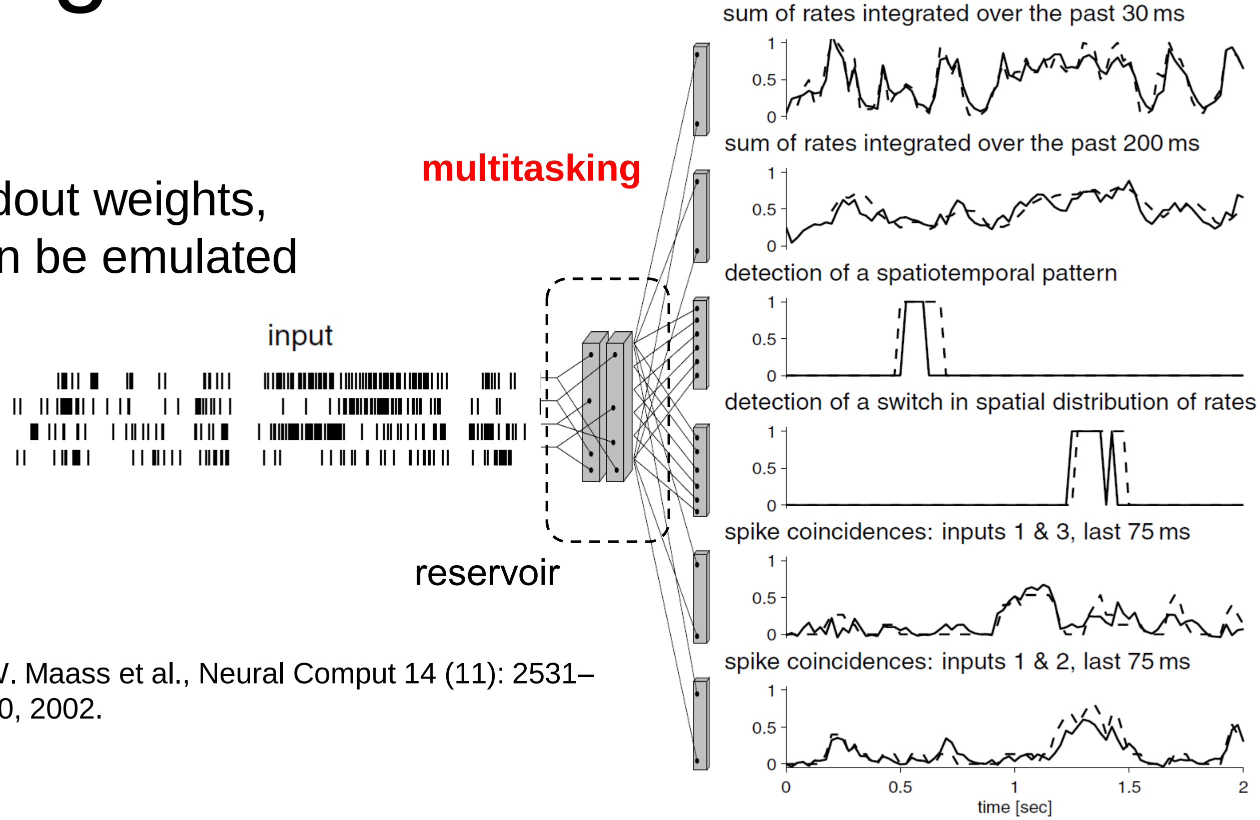
Benjamin Schrauwen,
Joni Dambre
(University of Gent)

Typical settings

- **Open-loop**

By attaching the readout weights, multiple functions can be emulated simultaneously!

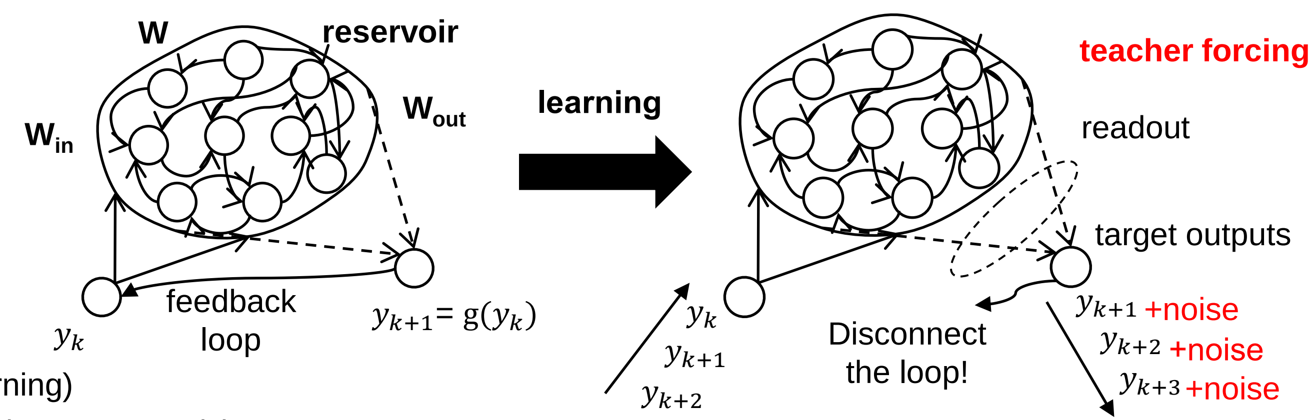
$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$
$$\hat{\boldsymbol{y}}_k = \hat{\psi}(\boldsymbol{x}_k)$$



W. Maass et al., Neural Comput 14 (11): 2531–60, 2002.

- **Close-loop**

$$\hat{\boldsymbol{x}}_{k+1} = f(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k)$$
$$\hat{\boldsymbol{u}}_k = \hat{\psi}(\hat{\boldsymbol{x}}_k)$$



*On-line learning
(e.g. FORCE learning, Innate learning)

Sussillo, D., & Abbott, L. F. (2009). *Neuron*, 63(4), 544-557.

*Prerequisite: "*reproducible response*"

(target function)

$$y_{k+1} = g(u_k, u_{k-1}, \dots)$$

(reservoir computing)

$$\begin{cases} \boldsymbol{x}_{\boldsymbol{k+1}} = f(\boldsymbol{x}_{\boldsymbol{k}}, u_k) \\ \hat{y}_{k+1} = \hat{\psi}(\boldsymbol{x}_{\boldsymbol{k+1}}) \end{cases}$$

$$\boxed{g(u_k, u_{k-1}, \dots) \approx \hat{\psi}(\boldsymbol{x}_{\boldsymbol{k+1}})}$$

We want to emulate (learn) function "g"!

(prerequisite)

- Reproducible response to the same input sequence!
- Reservoir states should not depend on the initial condition!

$$f(\boldsymbol{x}_{\boldsymbol{k}}, u_k) - f\left(\boldsymbol{x}^*_{\boldsymbol{k}}, u_k\right) \approx 0 \longleftarrow \boldsymbol{x}_{\boldsymbol{k}} = \boldsymbol{\phi}(u_{k-1}, u_{k-2}, \dots)$$

(common-signal-induced synchronization/ Negative conditional Lyapunov exponents)

(echo state property (ESP))

Z. Lu, B. R. Hunt, E. Ott, *Chaos* 28, 061104 (2018).

H. Jaeger, GMD Technical Report. 148 (2001).
I. B. Yildiz, et. al., *Neural netw.* 35 (2012).
G. Manjunath, et. al., *Neural comp.* 25 (2013).

One dimensional example

E.g.) $x_{k+1} = -\frac{1}{2}x_k + u_k$

$$x_{\boldsymbol{k}} = \left(-\frac{1}{2}\right)^{k-1} x_0 + u_{k-1} - \frac{1}{2}u_{k-2} + \cdots + \left(-\frac{1}{2}\right)^{k-1} u_0$$

$\Downarrow$ $\mathrm{k} \to \infty$

$$x_{\boldsymbol{k}} = u_{k-1} - \frac{1}{2}u_{k-2} + \cdots$$

Independent of k

E.g.) $x_{k+1} = -x_k + u_k$

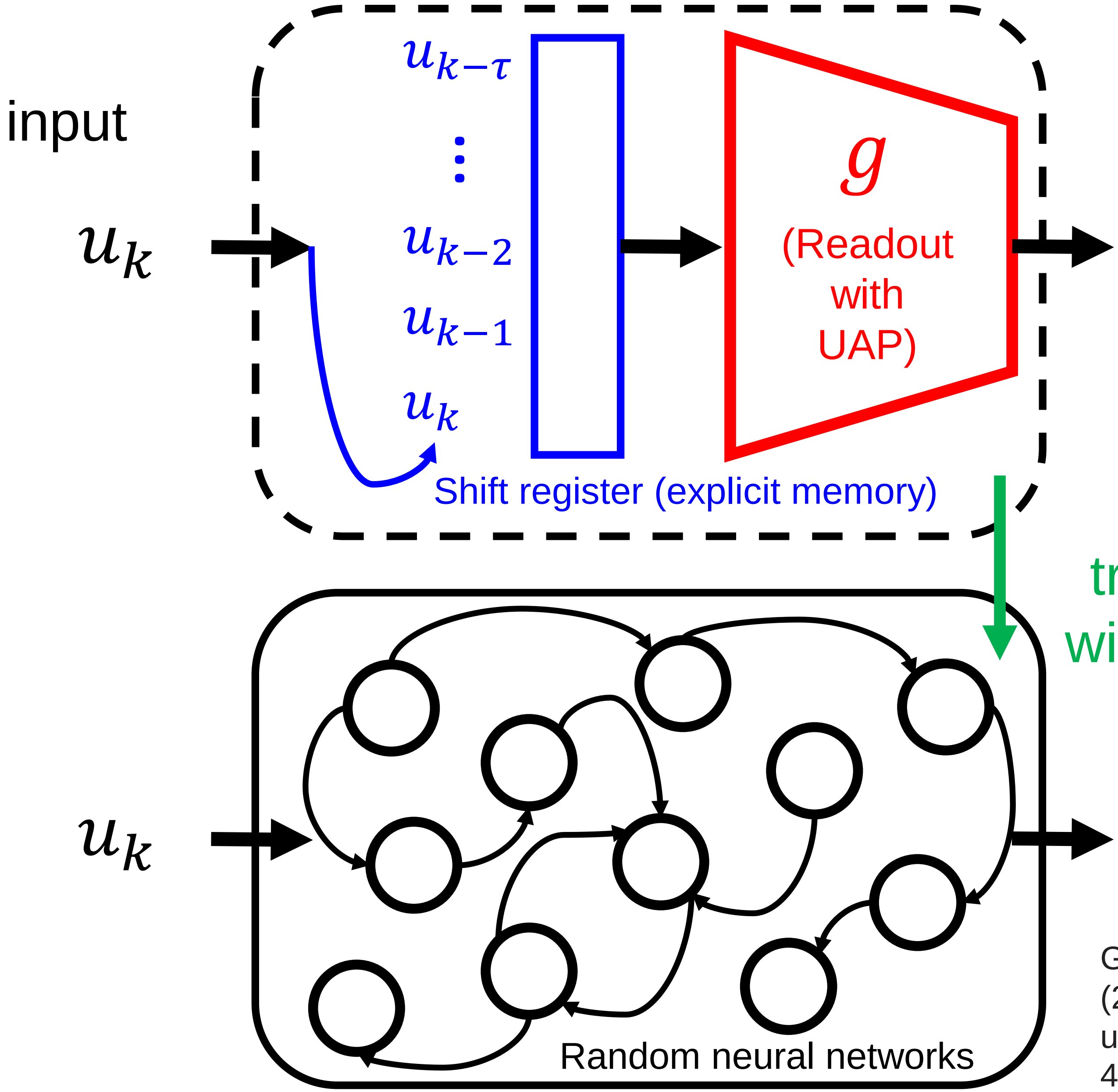
$$x_{\boldsymbol{k}} = (-1)^{k-1}x_0 + u_{k-1} - u_{k-2} + \cdots + (-1)^{k-1}u_0$$

No ESP

(No ESP examples)

- Limit cycles, chaos (several techniques to use them as a reservoir)
- Non-stationary systems, systems having trends

Encoding memory as dynamics: $y_{k+1} = g(u_k, u_{k-1}, \ldots)$



Memory and nonlinear transformation are within the dynamics!

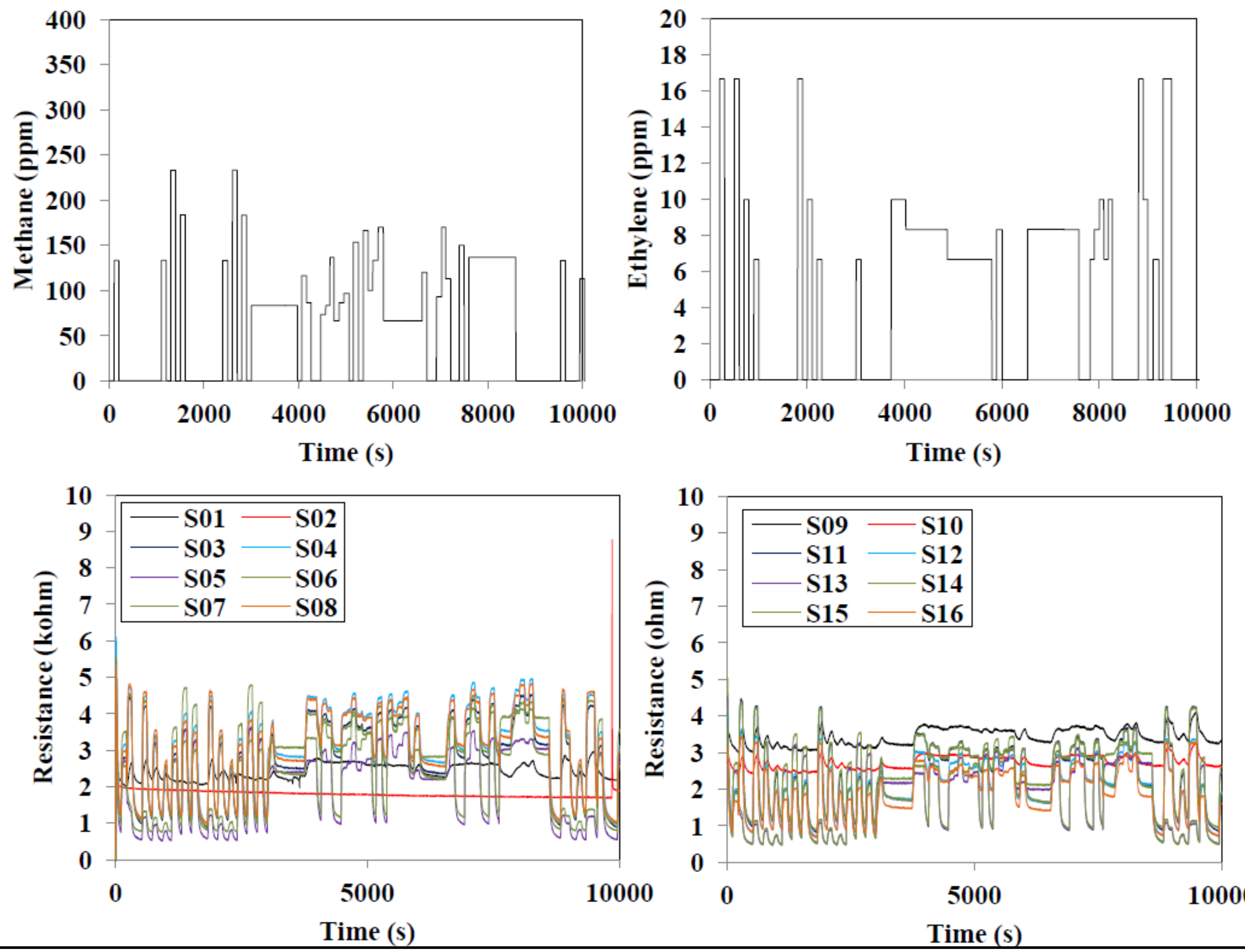
Grigoryeva, L., & Ortega, J. P. (2018). Echo state networks are universal. *Neural Networks*, *108*, 495-508.

Case 1: Sensor emulations

J. Fonollosa, et.al., Sensors and Actuators B **215** (2015) pp.618-629

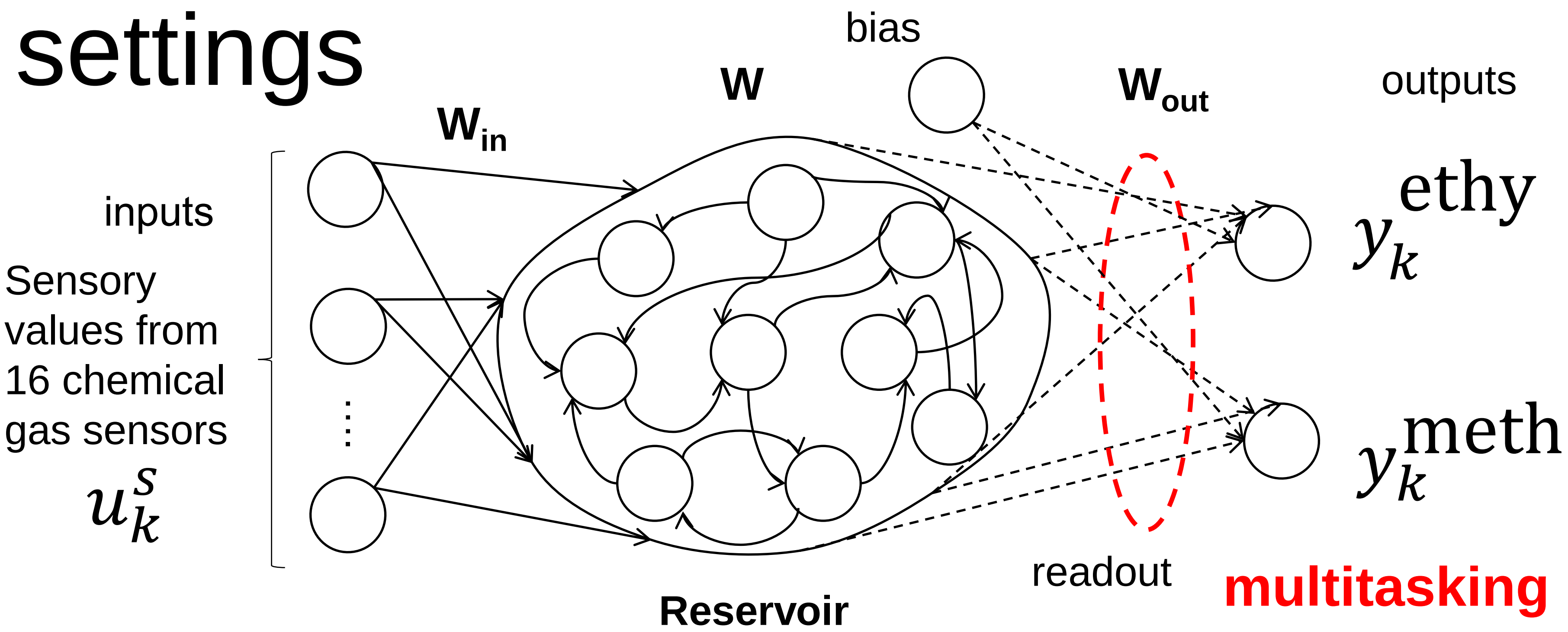
Gas concentration of Methane and Ethylene

16 chemical gas sensors (not so sensitive to Methane and Ethylene)



Emulate Methane and Ethylene gas sensors using 16 chemical gas sensors!

settings



- Task is to emulate Methane and Ethylene sensors out of 16 chemical gas sensors.
- Node number is 100, W and W_{in} are determined random. Only W_{out} is adjusted.
- If Methane and Ethylene were functions of 16 chemical gas, then there is a possibility that the task can be performed successfully!

$$x_k^i = f\left(\sum_{j=1}^{M} w^{ij} x_{k-1}^j + \sum_{s=1}^{16} w_{in}^{is} u_k^s\right),$$

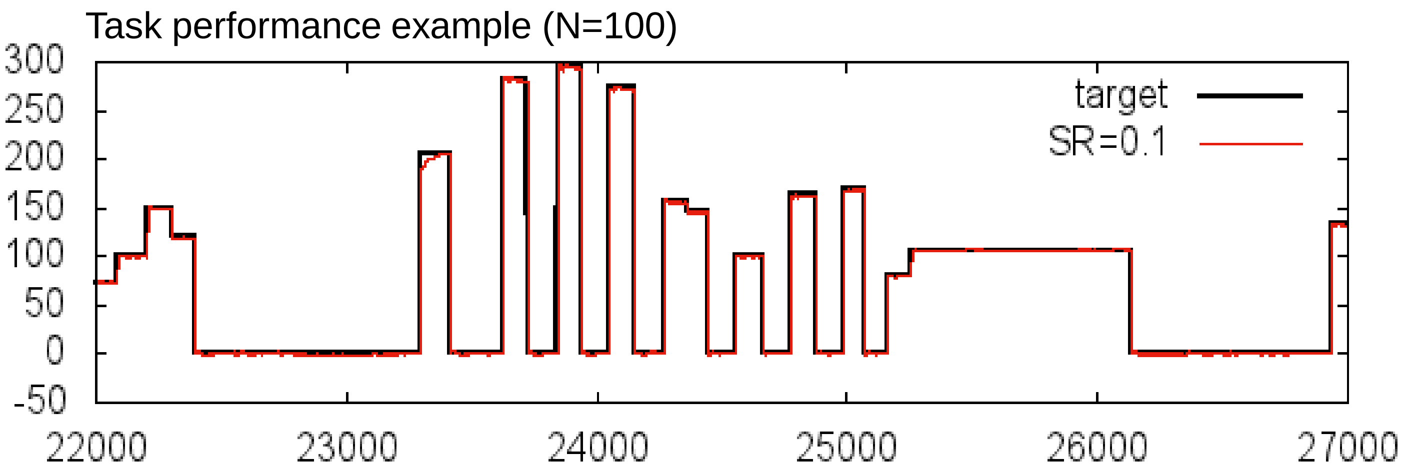
$$y_k^{ethy} = \sum_{i=0}^{M} w_{out,ethy}^i x_k^i,$$

$$y_k^{meth} = \sum_{i=0}^{M} w_{out,meth}^i x_k^i,$$

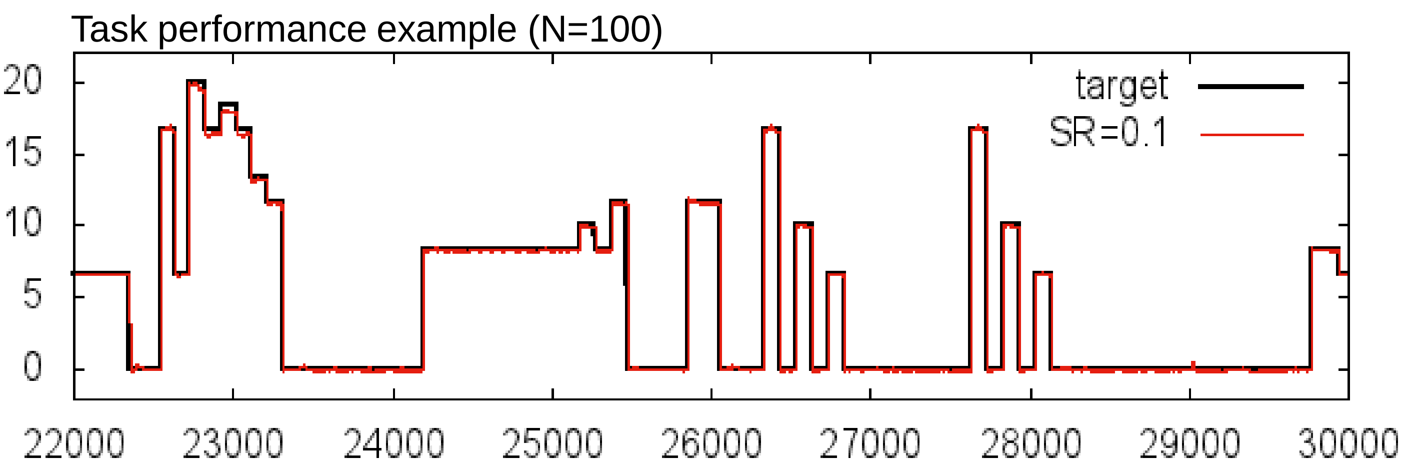
$$f(x) = \tanh(gx)$$

Task performance

y_k^{ethy}

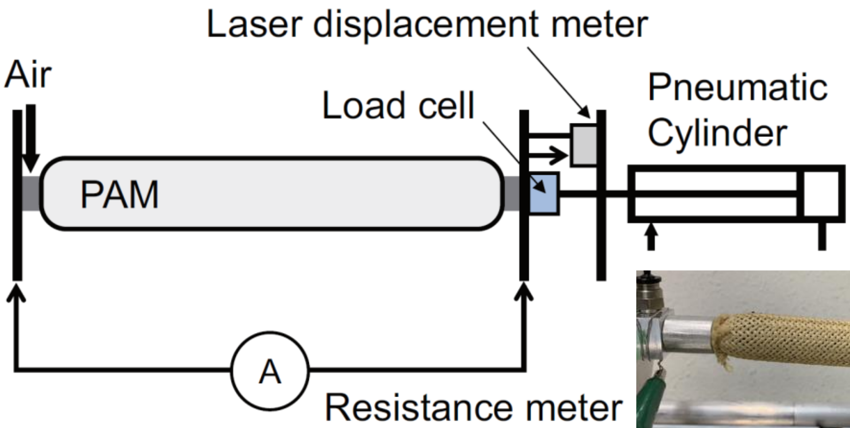


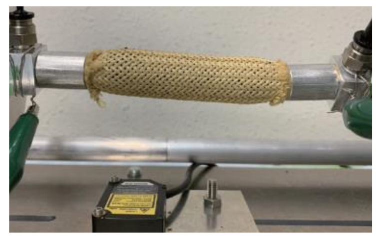
y_k^{meth}



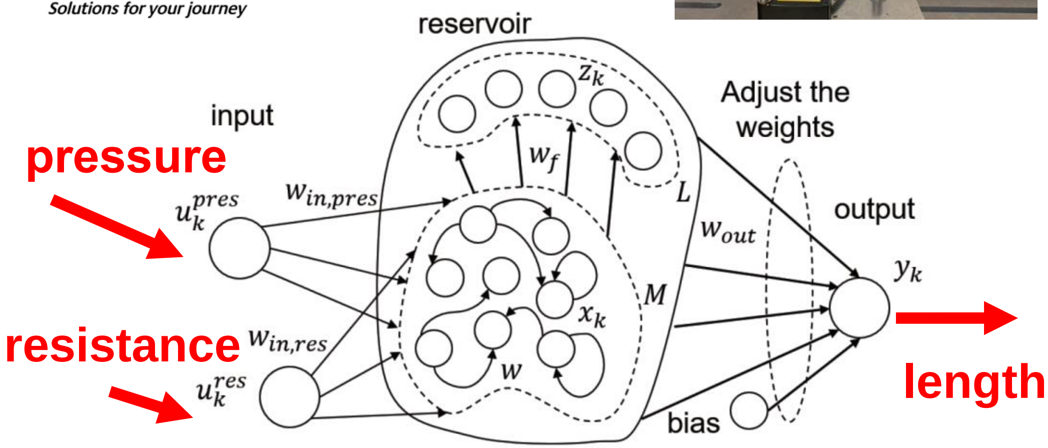
- Learning is quick and performance is good.
- Can be used for edge computing device!

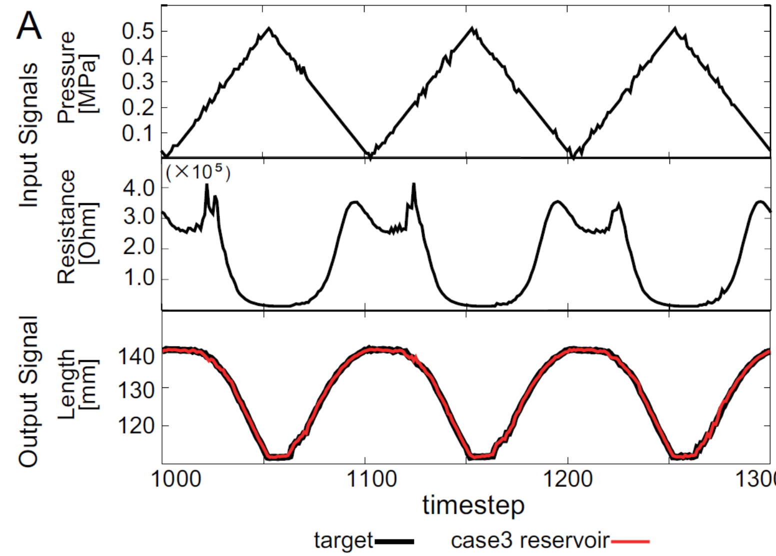
Soft sensing using material dynamics

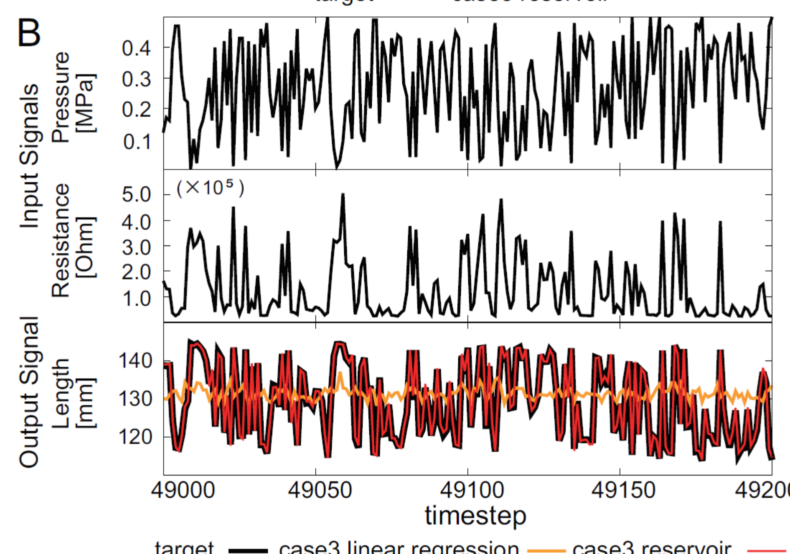




BRIDGESTONE
Solutions for your journey





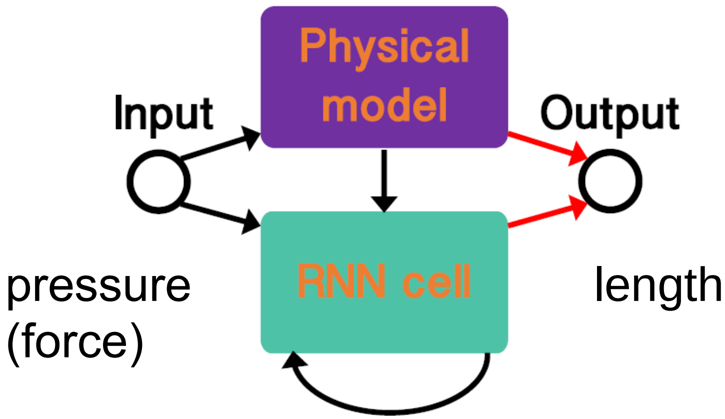


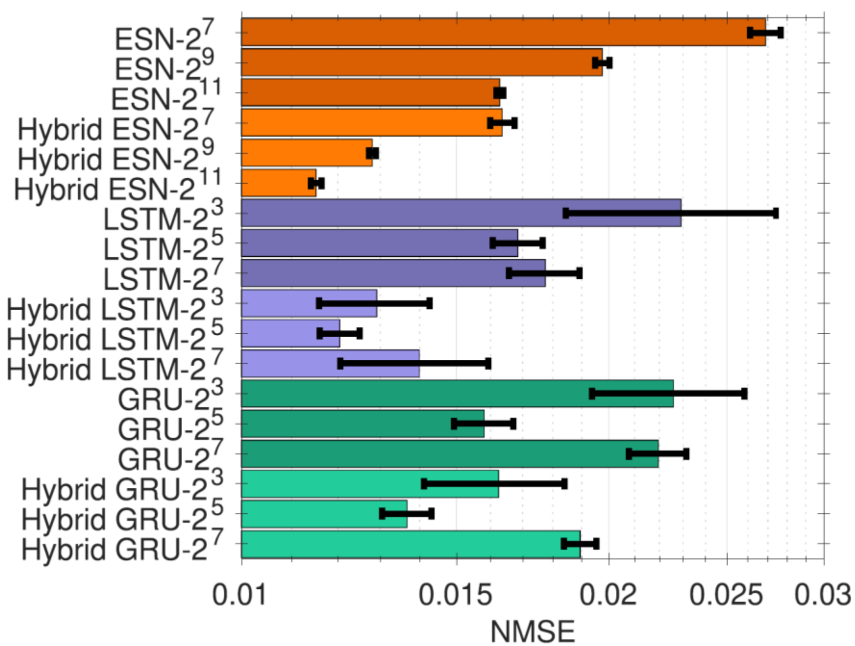
R. Sakurai, et. al., Proc. of 2020 3rd IEEE RoboSoft, pp. 710-717, 2020.
W. Sun, et. al., Proc. of 2022 IEEE 5th RoboSoft, pp. 409-415, 2022.
N. Akashi, et. al., Adv. Intell. Syst. 4: 2200123 (2022).

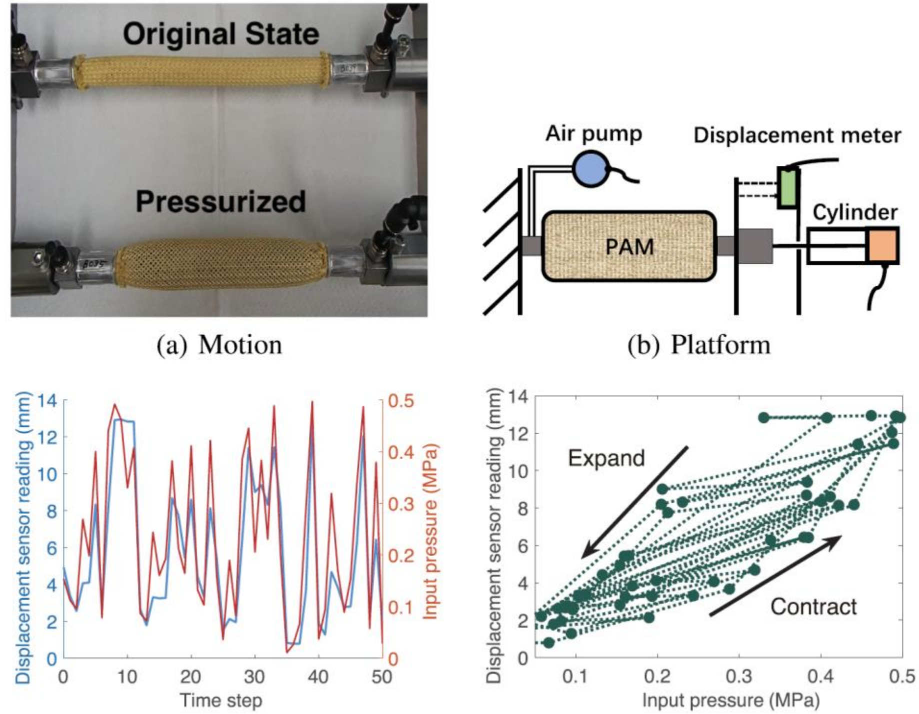
- Emulating a laser displacement meter in a high precision!
- Using conducting rubbers and do not need to attach the rigid sensors!

Physics-informed RNN for indirect sensing

$$B\dot{y} + K(y - y_0) + c \cdot \mathrm{sgn}\{\dot{y}\} = u$$







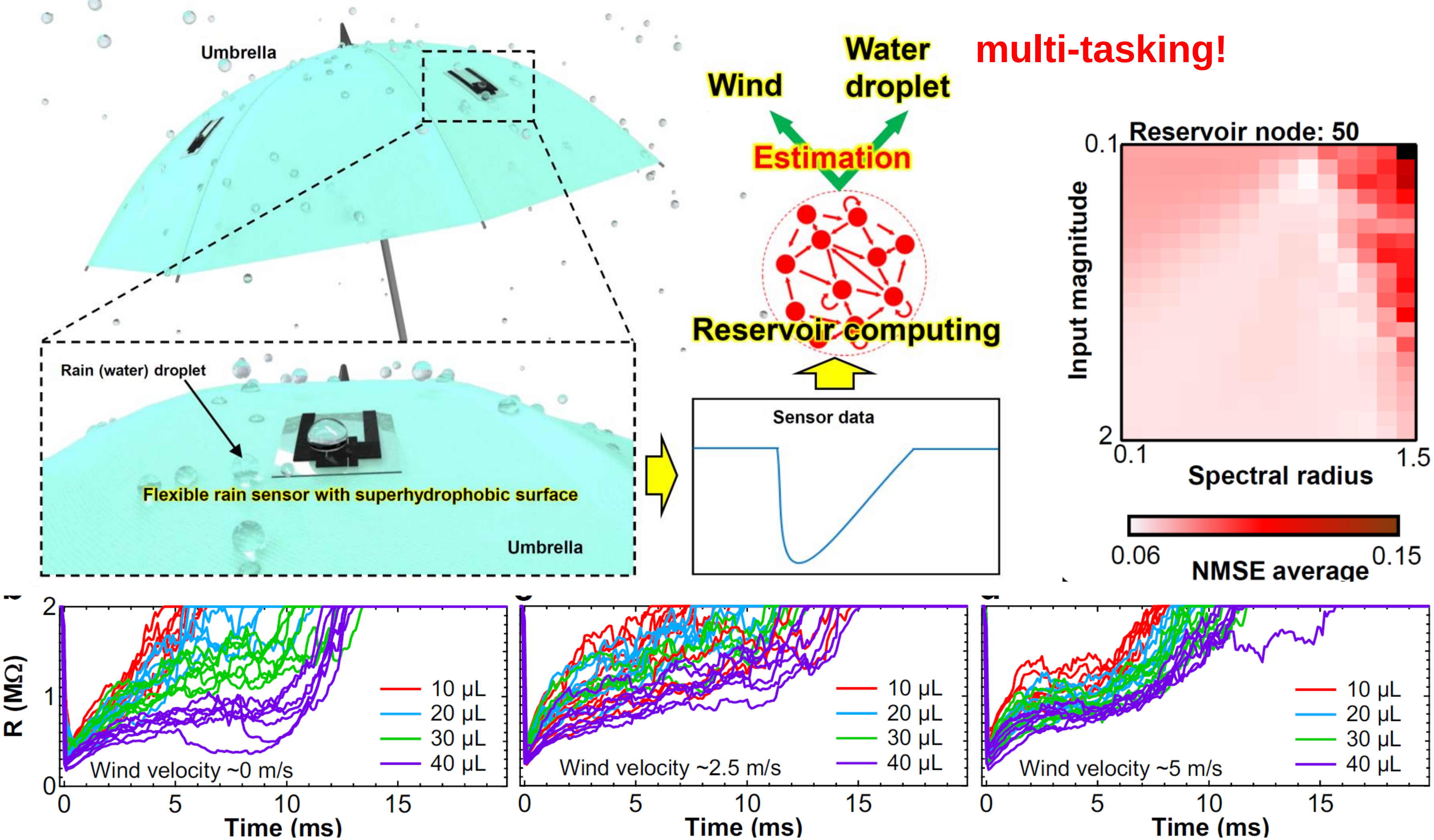
(a) Motion (b) Platform

(c) Time sequence (d) Hysteresis loop

Sun, W., Akashi, N., Kuniyoshi, Y., & Nakajima, K. (2022). Physics-informed recurrent neural networks for soft pneumatic actuators. *IEEE Robotics and Automation Letters*.

Knowledge-based approach improves the prediction accuracy in any kind of RNN!

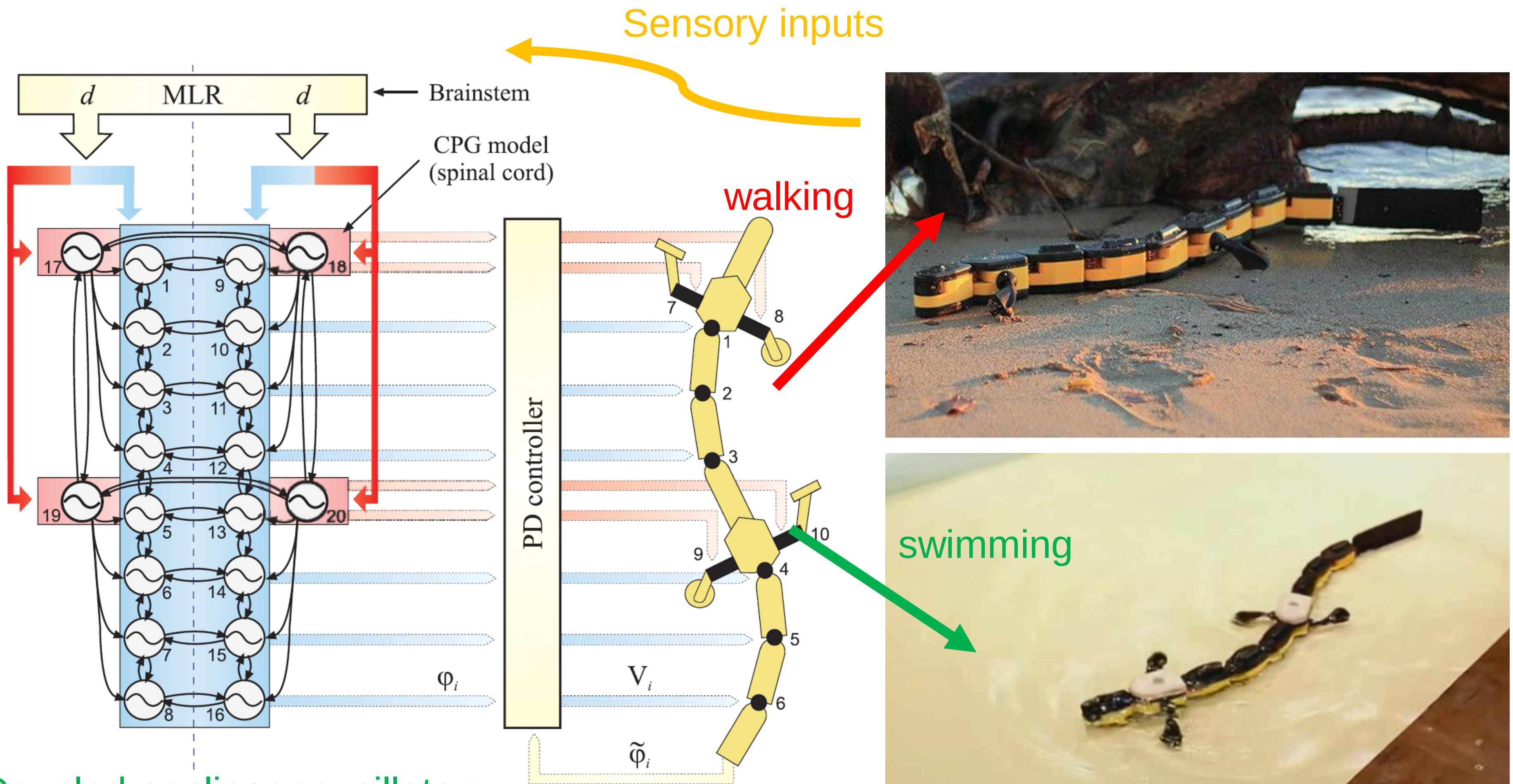
Reservoir computing meets flexible sensors



S. Wakabayashi, T. Arie, S. Akita, K. Nakajima, K. Takei, A multi-tasking flexible sensor via reservoir computing, ***Advanced Materials***, 2201663, 2022.

Multi-tasking is easy and learning is quick!

Case 2: Locomotion control

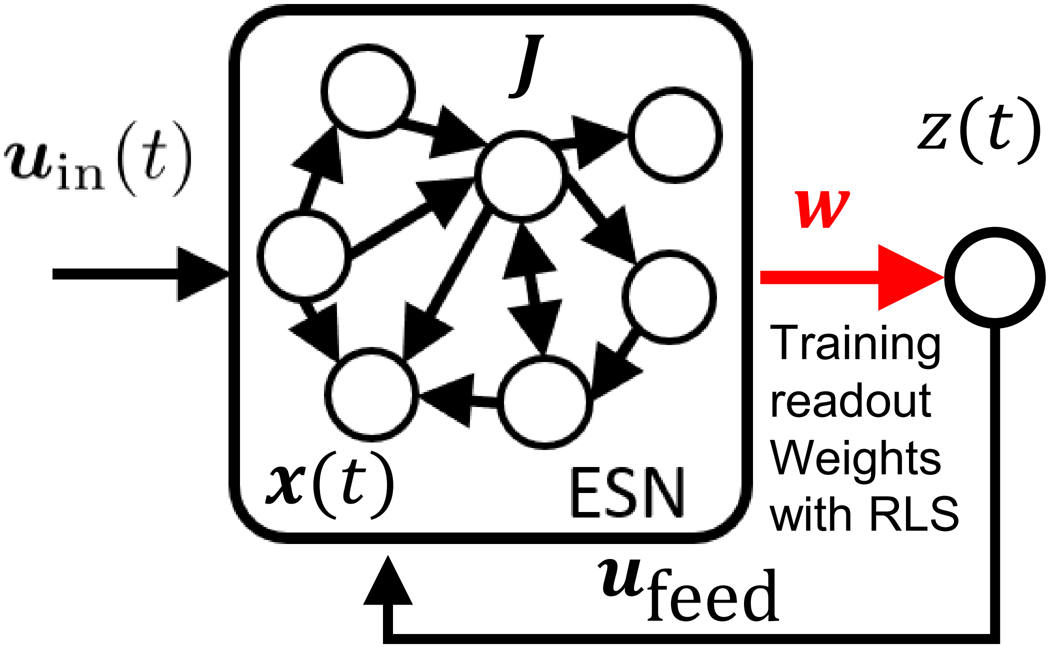


Coupled nonlinear oscillators

Ijspeert, A. J., Crespi, A., Ryczko, D., & Cabelguen, J. M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. Science, 315(5817), 1416-1420.

- Locomotion through central pattern generators!
- Switching the locomotion patterns via external stimuli! (e.g., sensors or external controllers)

Reservoir settings: continuous time ESN



τ: time constant,

$\boldsymbol{x}$: network state, $\boldsymbol{x} \in \mathbb{R}^N$

g: nonlinearity parameter ($g > 1$: chaotic)

$\boldsymbol{J}$: $N \times N$ matrix, $\boldsymbol{J} \sim \mathcal{N}\left(0, \frac{1}{N}\right)$

$\boldsymbol{u}_{\mathrm{in}}$: input function, $\boldsymbol{u}_{\mathrm{in}}: \mathbb{R} \to \mathbb{R}^N$

$\boldsymbol{u}_{\mathrm{feed}}$: $N \times 1$ matrix, $\boldsymbol{u}_{\mathrm{feed}} \sim [-1, 1]$

$\boldsymbol{w}$: readout weights, $N \times 1$ matrix

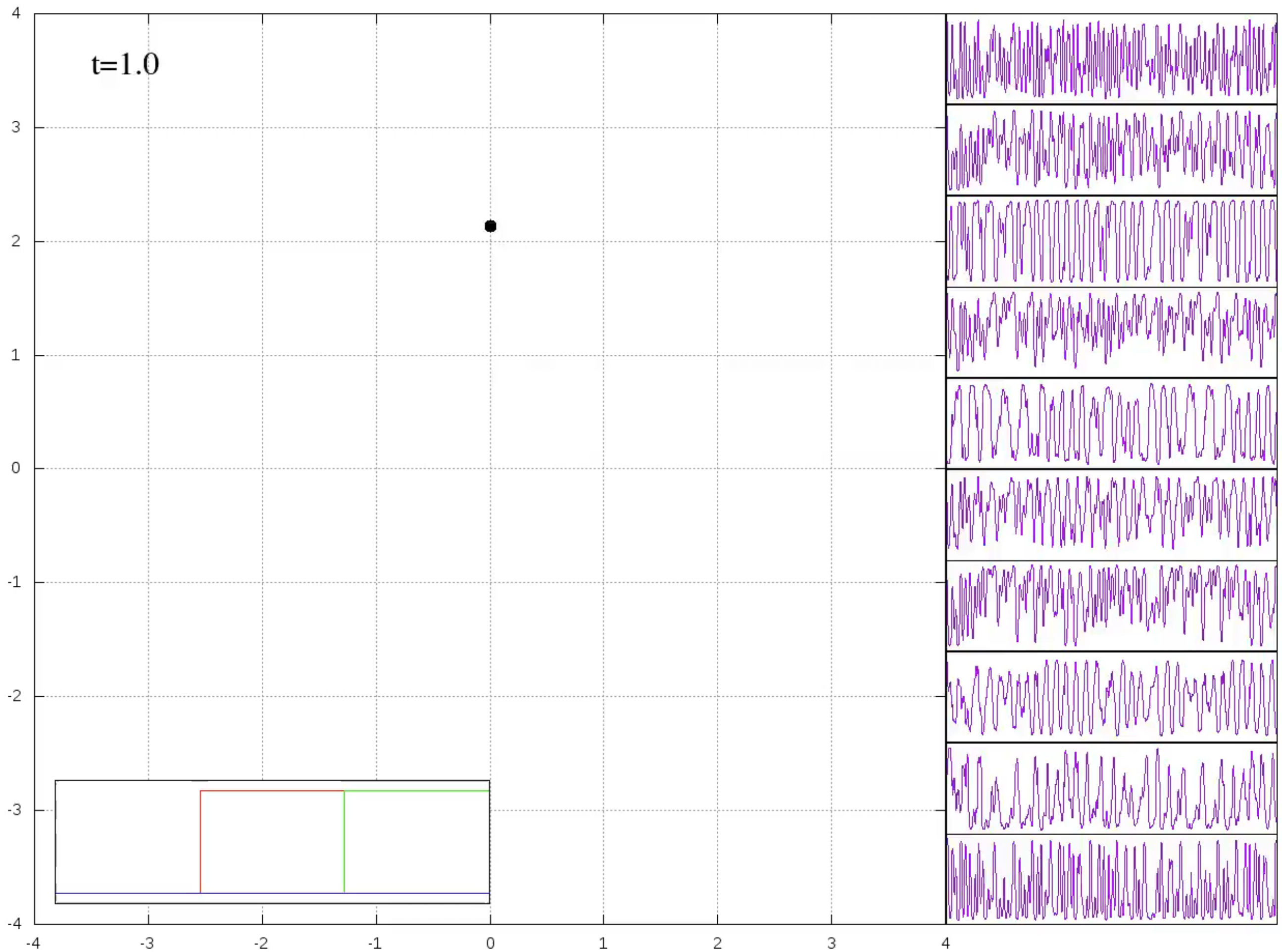
K. Inoue, K. Nakajima, Y. Kuniyoshi, Proceedings of NOLTA2018, pp. 412-414, 2018.

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \tanh\left(g\boldsymbol{J}\boldsymbol{x}(t) + \boldsymbol{u}_{\mathrm{feed}} z(t) + \boldsymbol{u}_{in}(t)\right)$$

$$z(t) = \boldsymbol{w}^T \boldsymbol{x}(t)$$

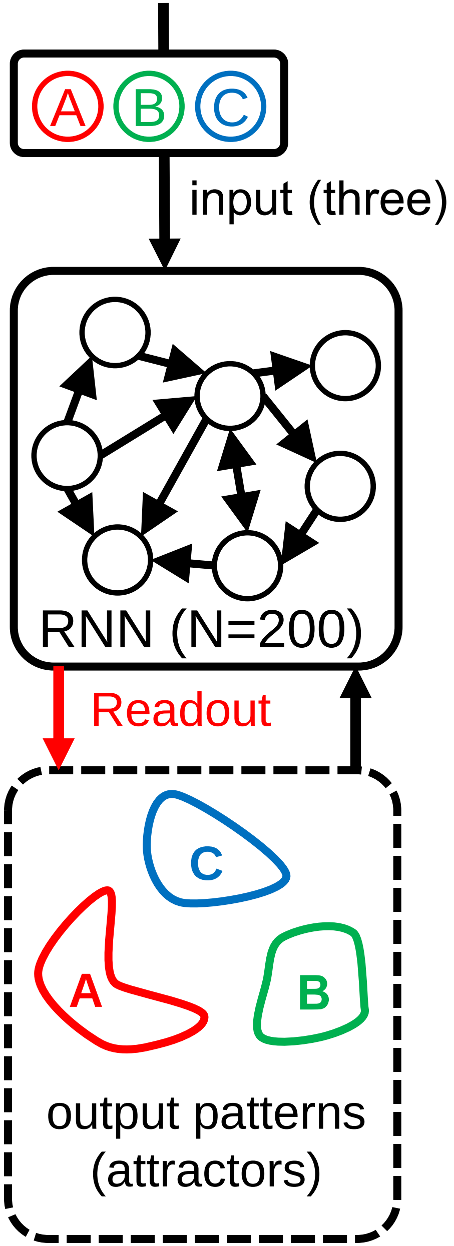
Embedding three limit cycles according to corresponding inputs

Output patterns



Input commands (three)

ESN dynamics (from 10 nodes)

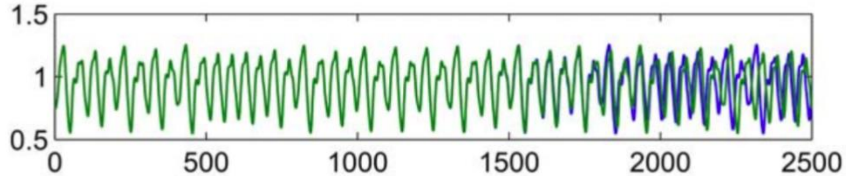


High Operability!

K. Inoue, K. Nakajima, Y. Kuniyoshi, Proceedings of NOLTA2018, pp. 412-414, 2018.

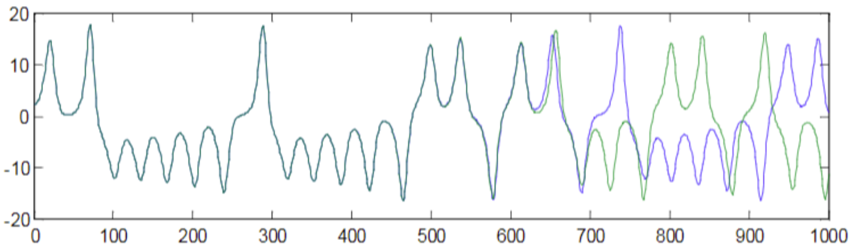
Emulating chaos and spatiotemporal dynamics

- Mackey-Glass attractor



$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10} - 0.1x(t)}$$

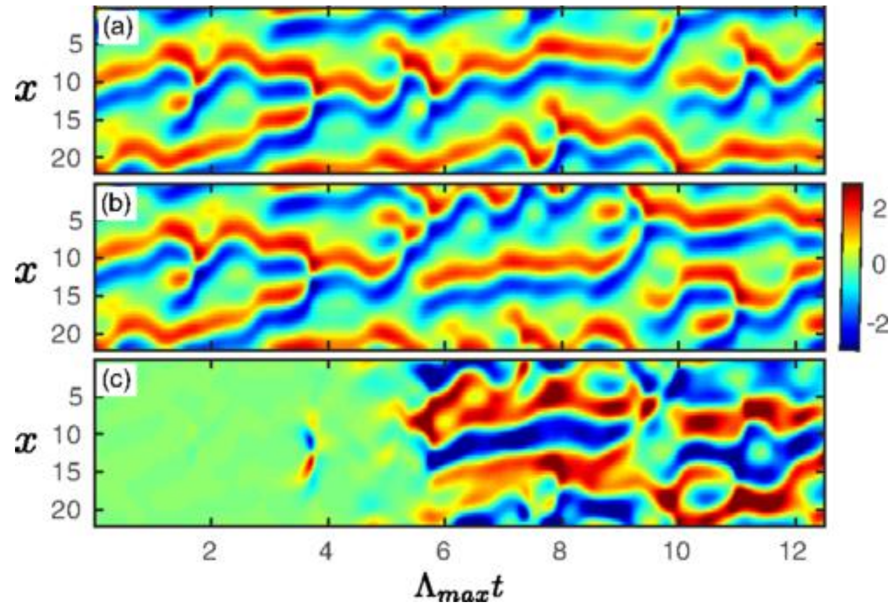
- Lorenz attractor (first coordinate)



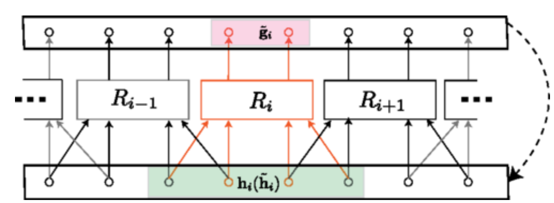
$$\frac{dx}{dt} = p(y-x)$$
$$\frac{dy}{dt} = -xz + rx - y$$
$$\frac{dz}{dt} = xy - bz$$

[H. Jaeger+, 2004, *Science*]

- Kuramoto-Sivashinsky equation



$$\frac{dy}{dt} = -y\frac{dy}{dx} - \frac{d^2y}{dx^2} - \frac{d^4y}{dx^4} + \mu\cos\left(\frac{2\pi x}{\lambda}\right)$$



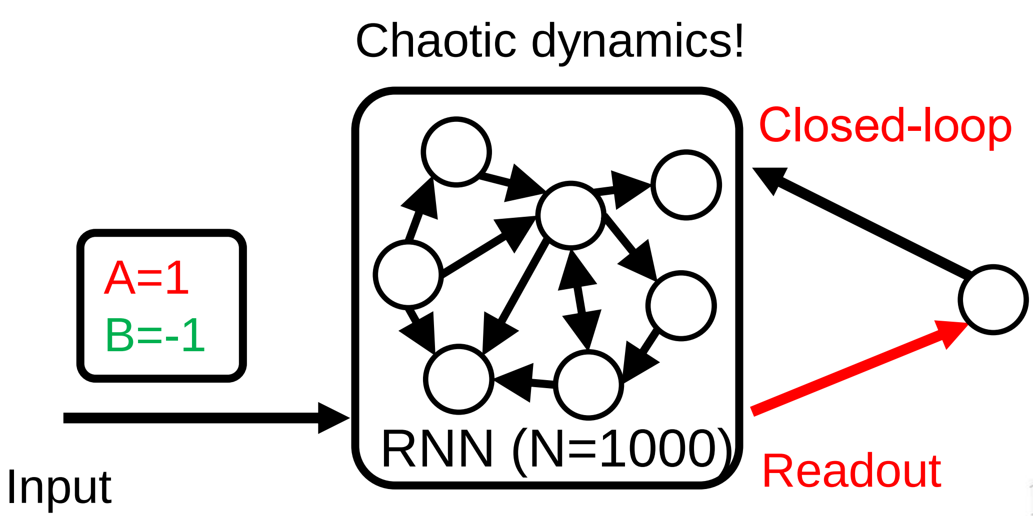
[J. Pathak+, 2018, *PRL*]

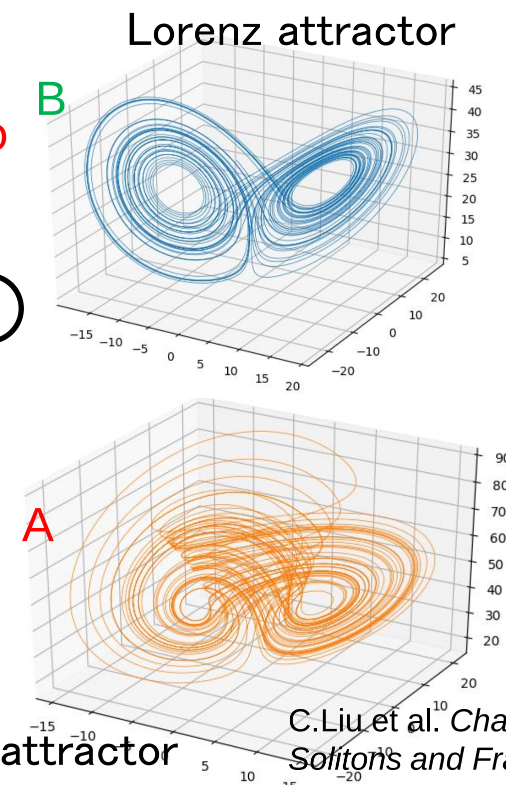
Switching chaotic attractors via sensory inputs

Deadlock avoidance with chaos！

S. Steingrube, M. Timme, F. W¨org¨otter, P. Manoonpong, Nature physics 6, 224 (2010).

- Step1: design two chaotic attractors
- Step2: switch them according to inputs



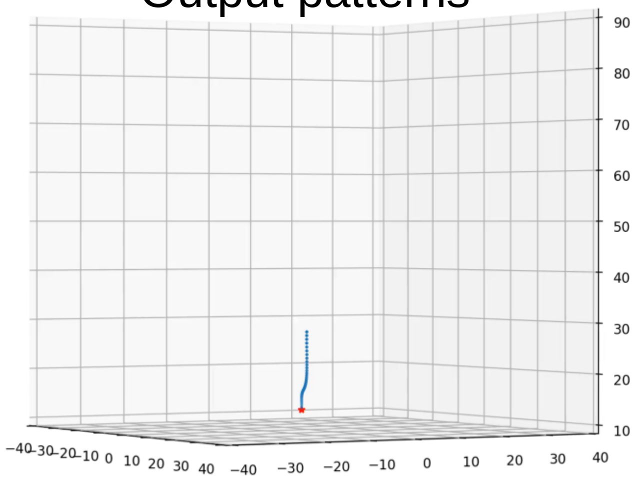


C.Liu et al. *Chaos, Solitons and Fractals* 22 (2004) 1031–1038

Check what is generalized in different inputs!

Lorenz attractor

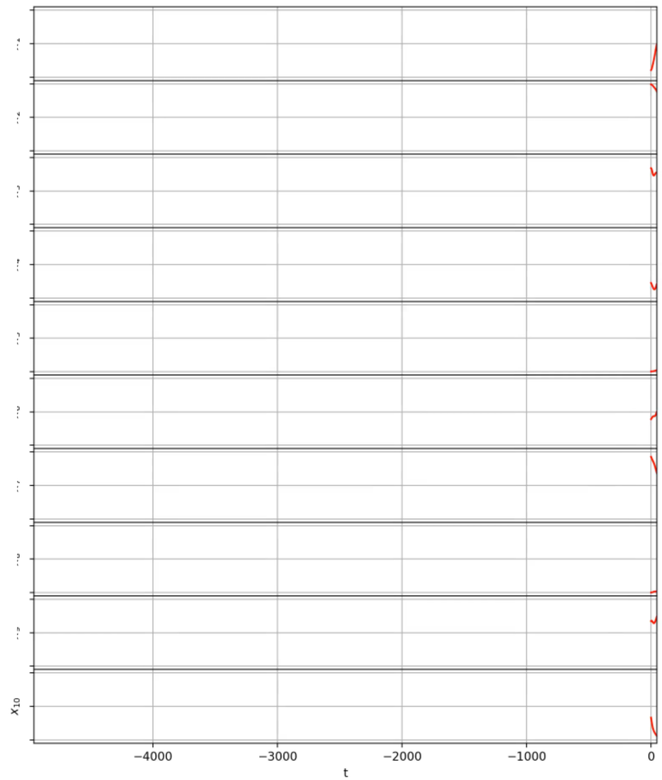
Output patterns



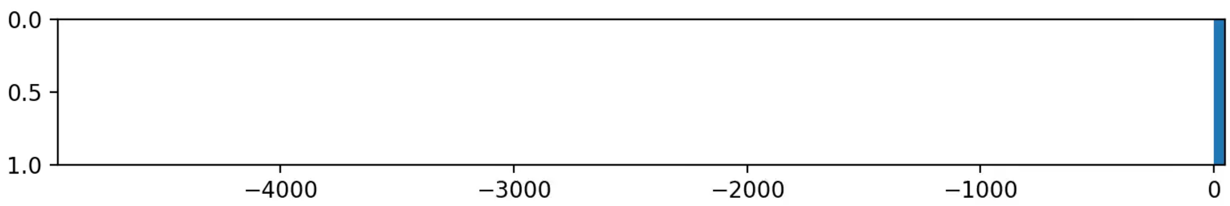
input: -1.00

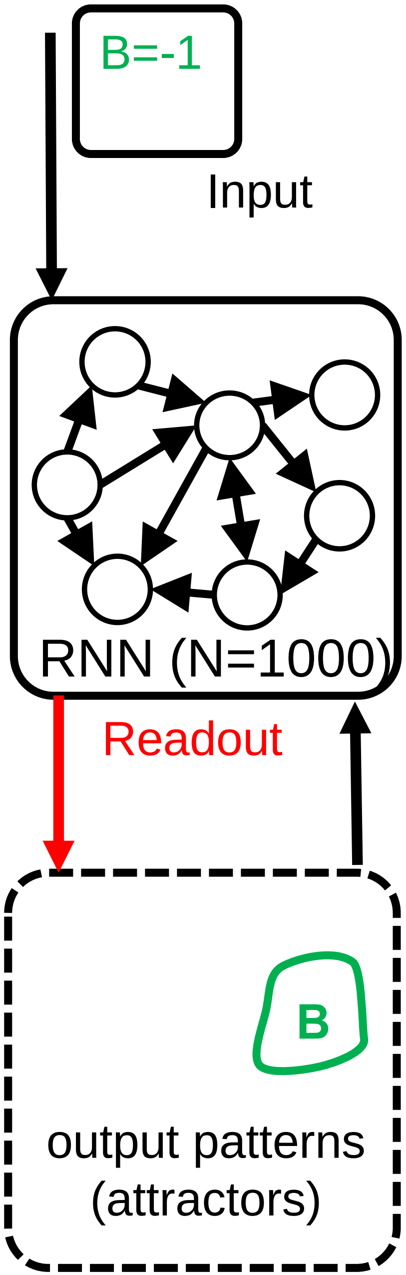
t = 47

ESN dynamics (from 10 nodes)



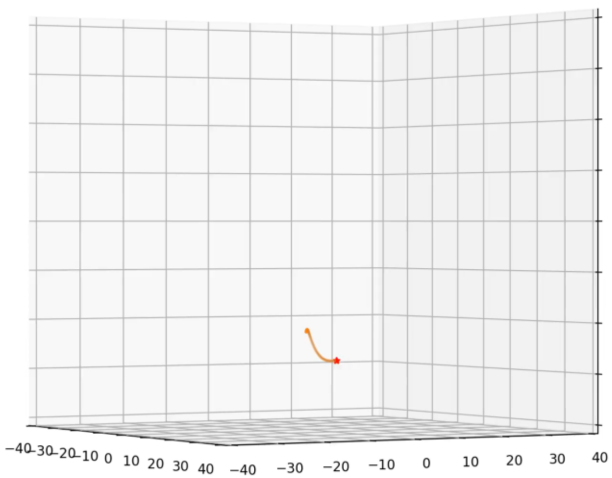
Input commands (-1: Lorenz)





Liu attractor

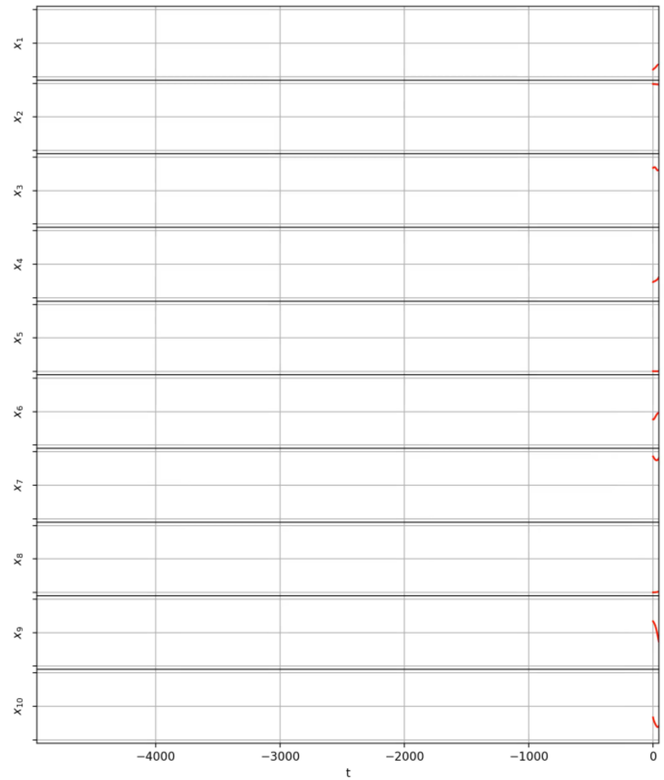
Output patterns



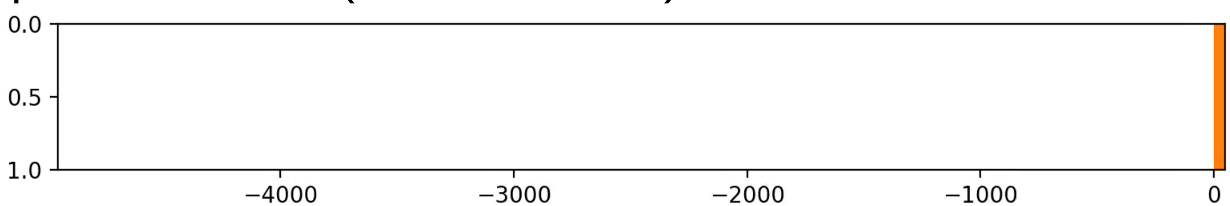
input: 1.00

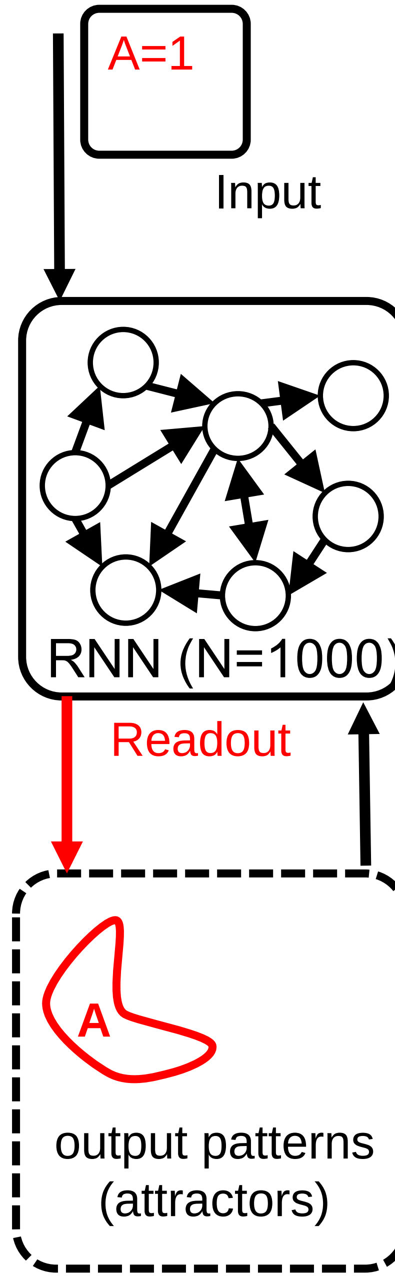
t = 47

ESN dynamics (from 10 nodes)



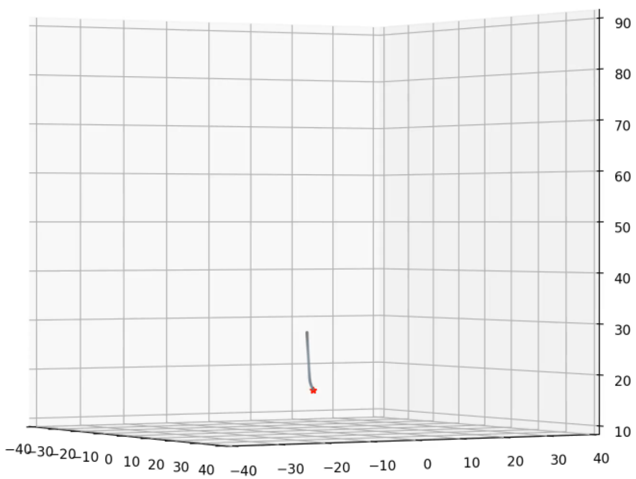
Input commands (1: Liu-attractor)



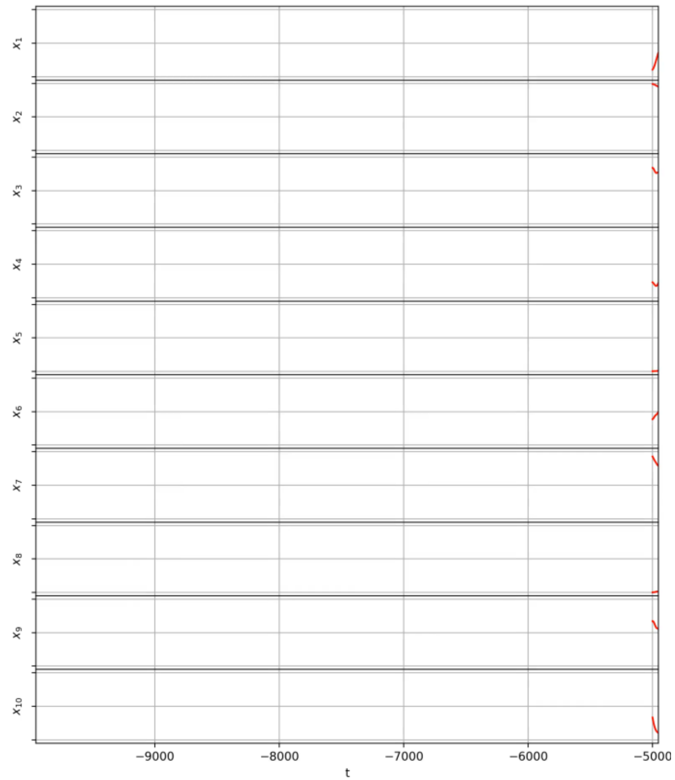


Interpolation and generalization

Output patterns

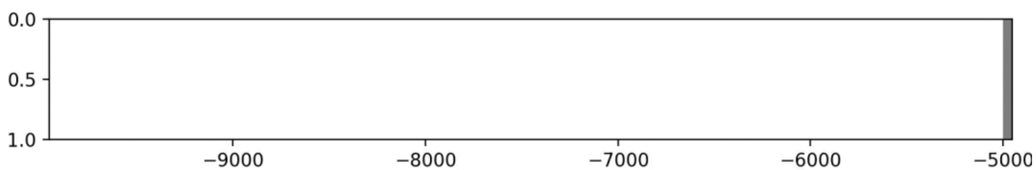


ESN dynamics (from 10 nodes)

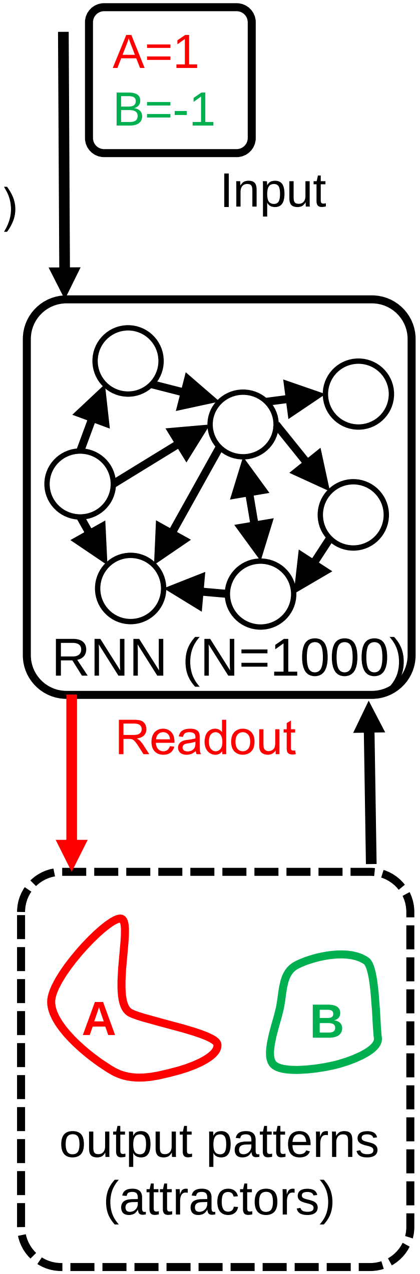


input: 0.00

t = -4953

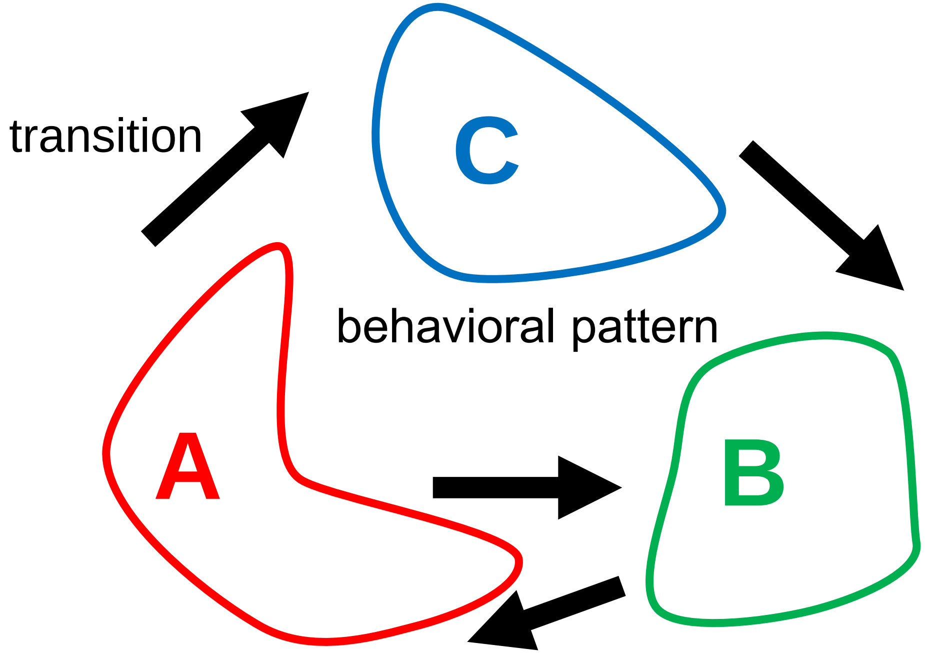


Input commands (-1: Lorenz, 0: no input, 1: Liu-attractor)



Flexible control of attractors

Designing spontaneous behavioral switching



~~External control~~

Internal dynamics

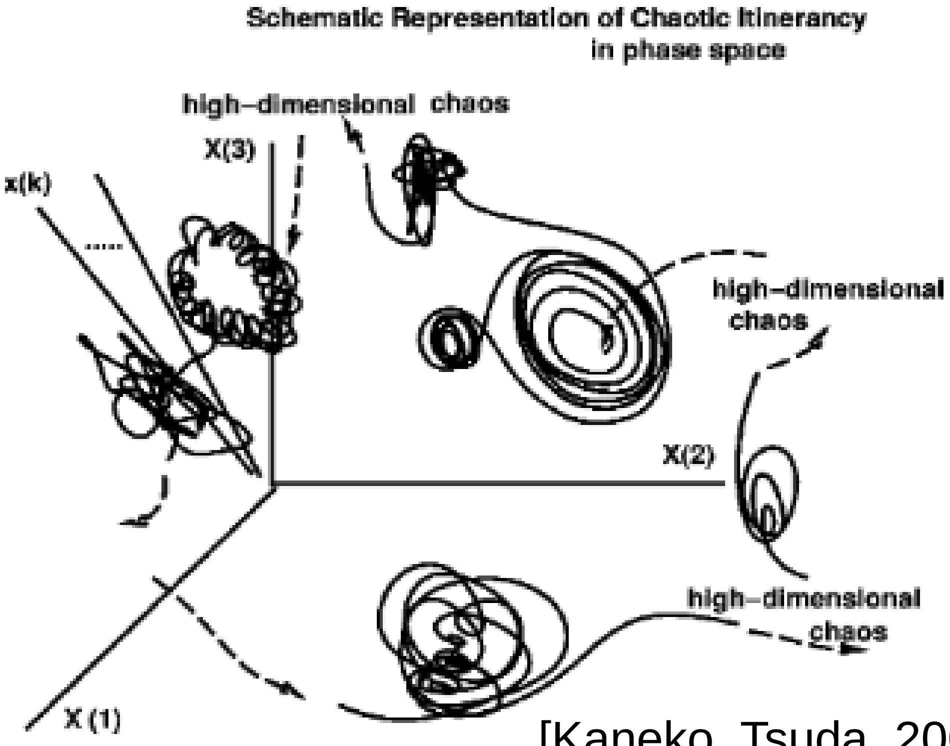
- **Step 1**: Behavioral patterns
- **Step 2**: Periodic transitions among the patterns

  (challenge: **embed timer**)
- **Step 3**: Random transitions among the patterns

  (challenge: **embed timer + random number generators**)

Step 3 is related to chaotic itinerancy!

What is chaotic itinerancy?



[Kaneko, Tsuda, 2003]

FIG. 1. Schematic representation of chaotic itinerancy.

First found in…

- Optical turbulence
  [K. Ikeda et. al., 1989]
- A globally coupled chaotic system
  [K. Kaneko, 1990; 1991]
- Nonequilibrium neural networks
  [I. Tsuda, 1991; 1992]

(Features)

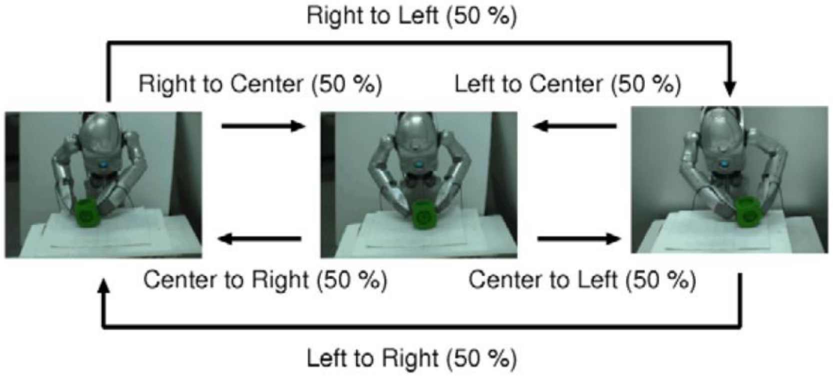
- Frequently observed in high-dimensional nonlinear dynamical systems
- Seemingly random transitions among quasi-attractors

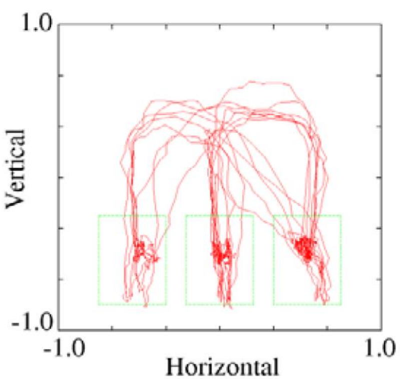
Propose a scheme to design CI!

Spontaneous behavioral switching in robots

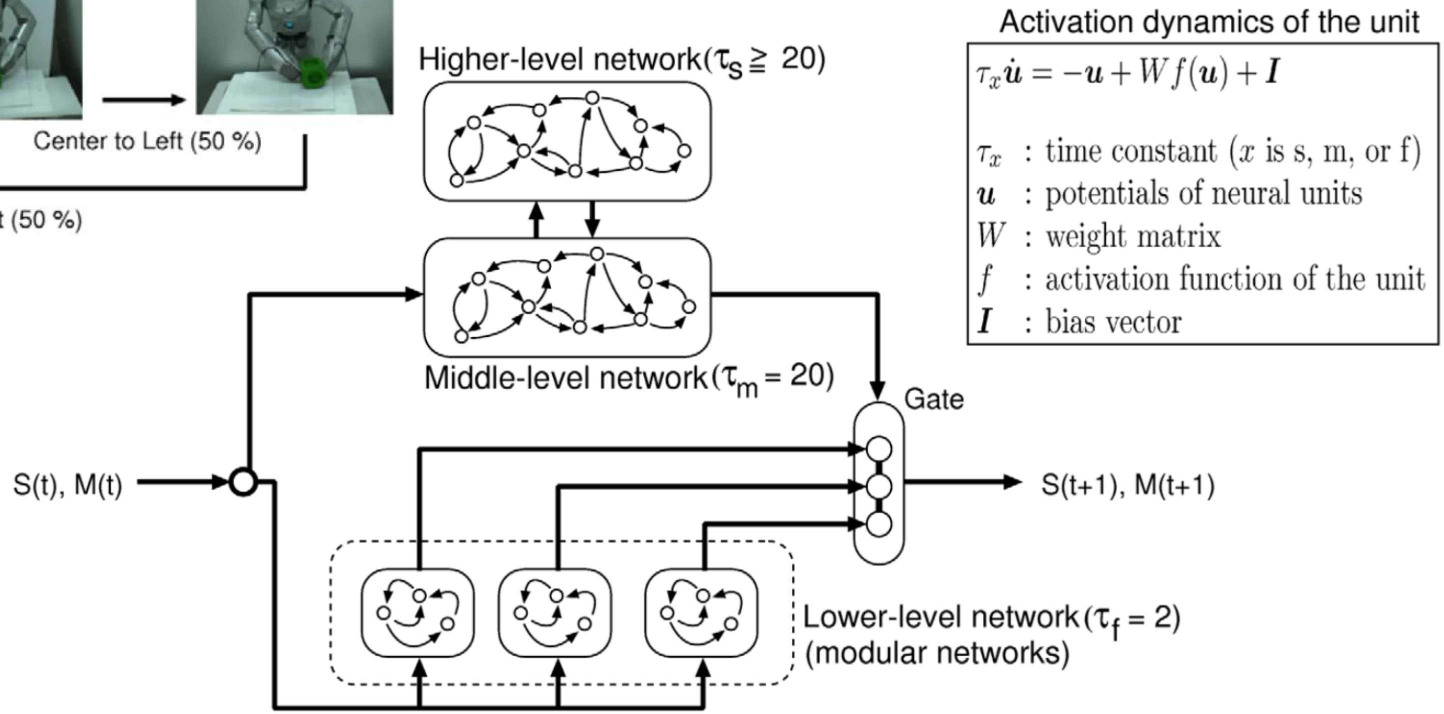
Behavioral switching

Namikawa, J., Nishimoto, R., & Tani, J. (2011). A neurodynamic account of spontaneous behaviour. *PLoS computational biology*, *7*(10), e1002221.



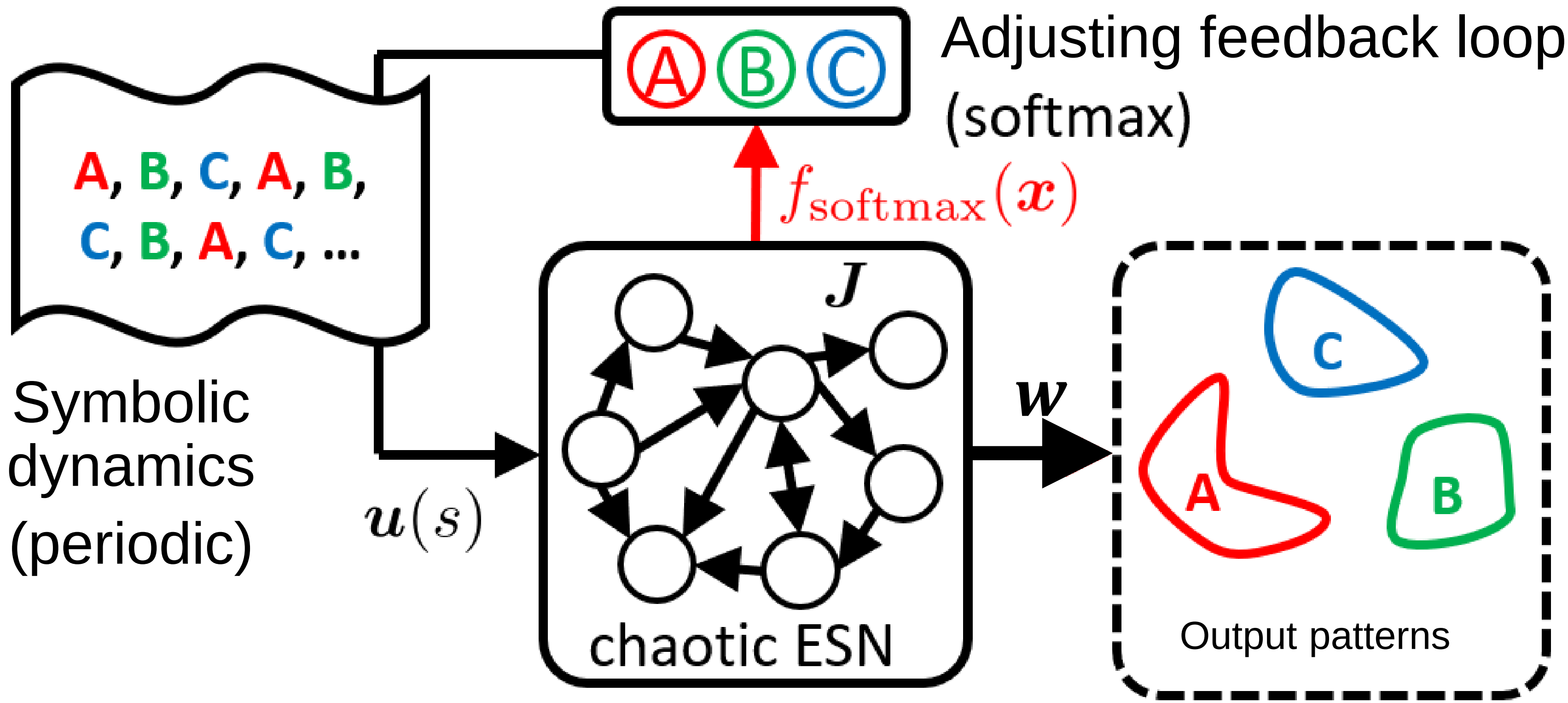


Network architecture



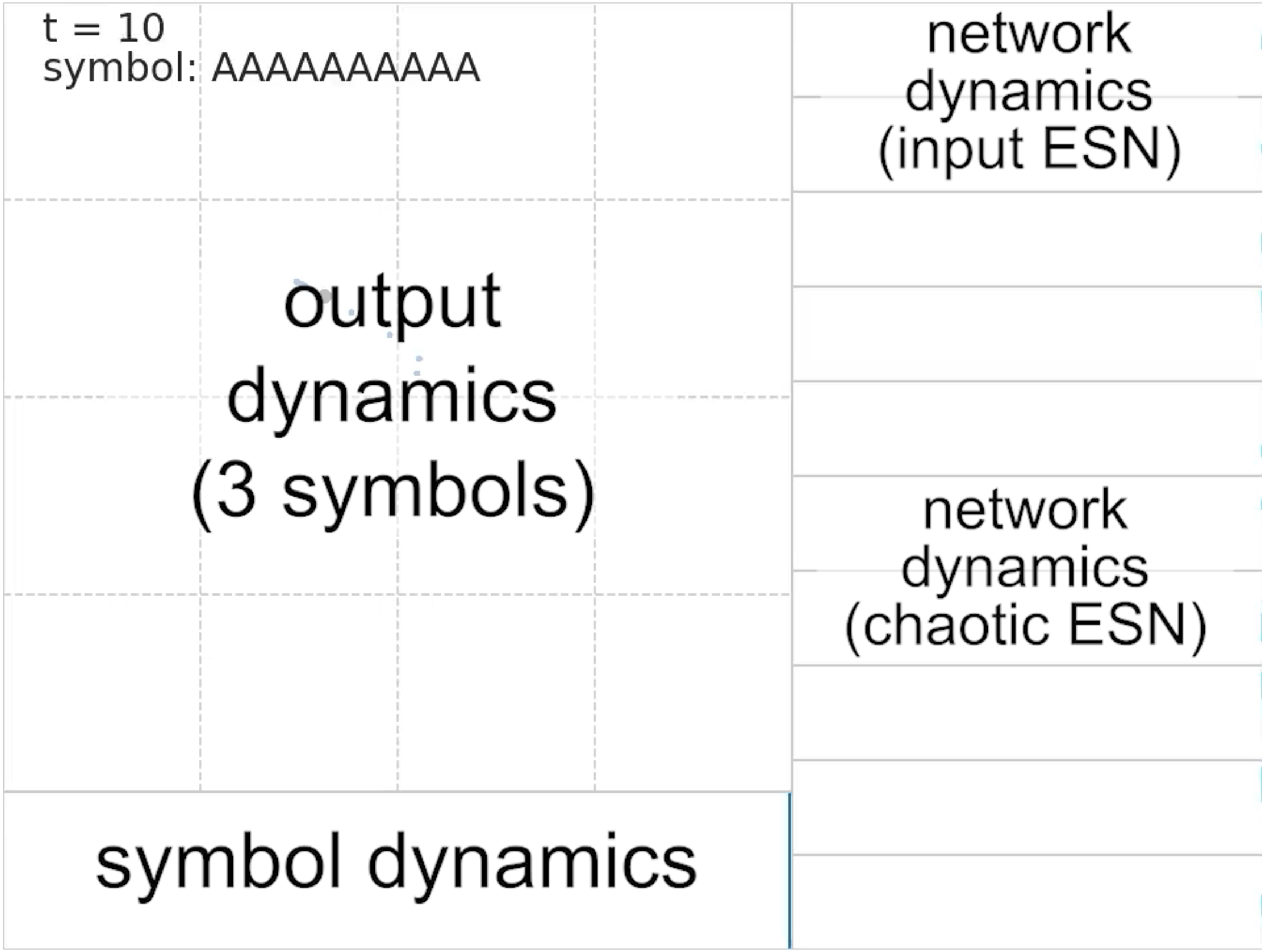
- Deterministic chaos self-organized to generate stochastic processes.
- Using hierarchical modules beforehand!

Switching patterns autonomously



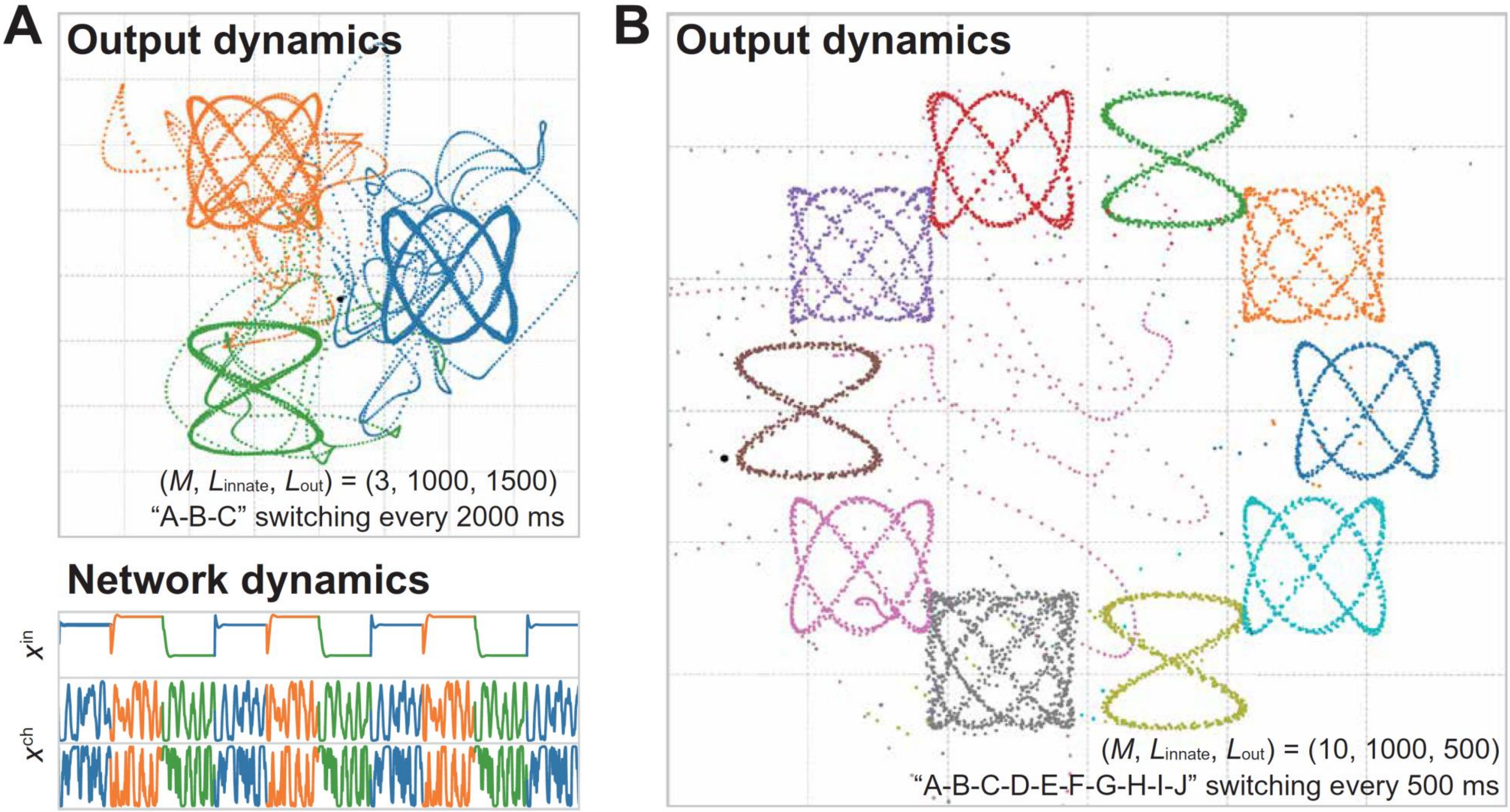
- Internalizing the external control through feedback loop!
- Should estimate the duration of time for each pattern (switch in appropriate timing) using the same reservoir!

Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.



Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

Periodic transitions are embedded!

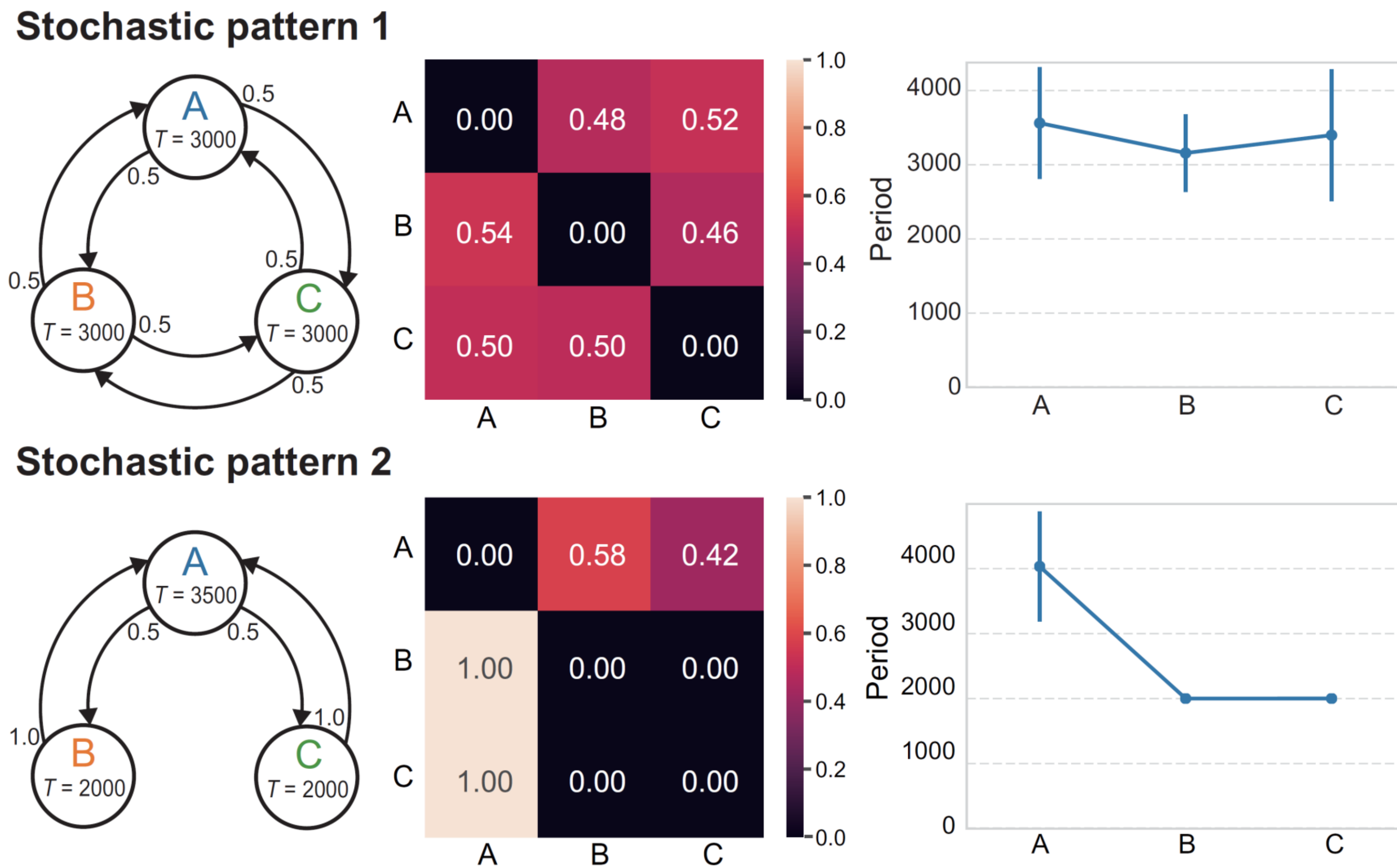


Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

Can harness complex dynamics and design patterns!

Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

Designing transition probabilities

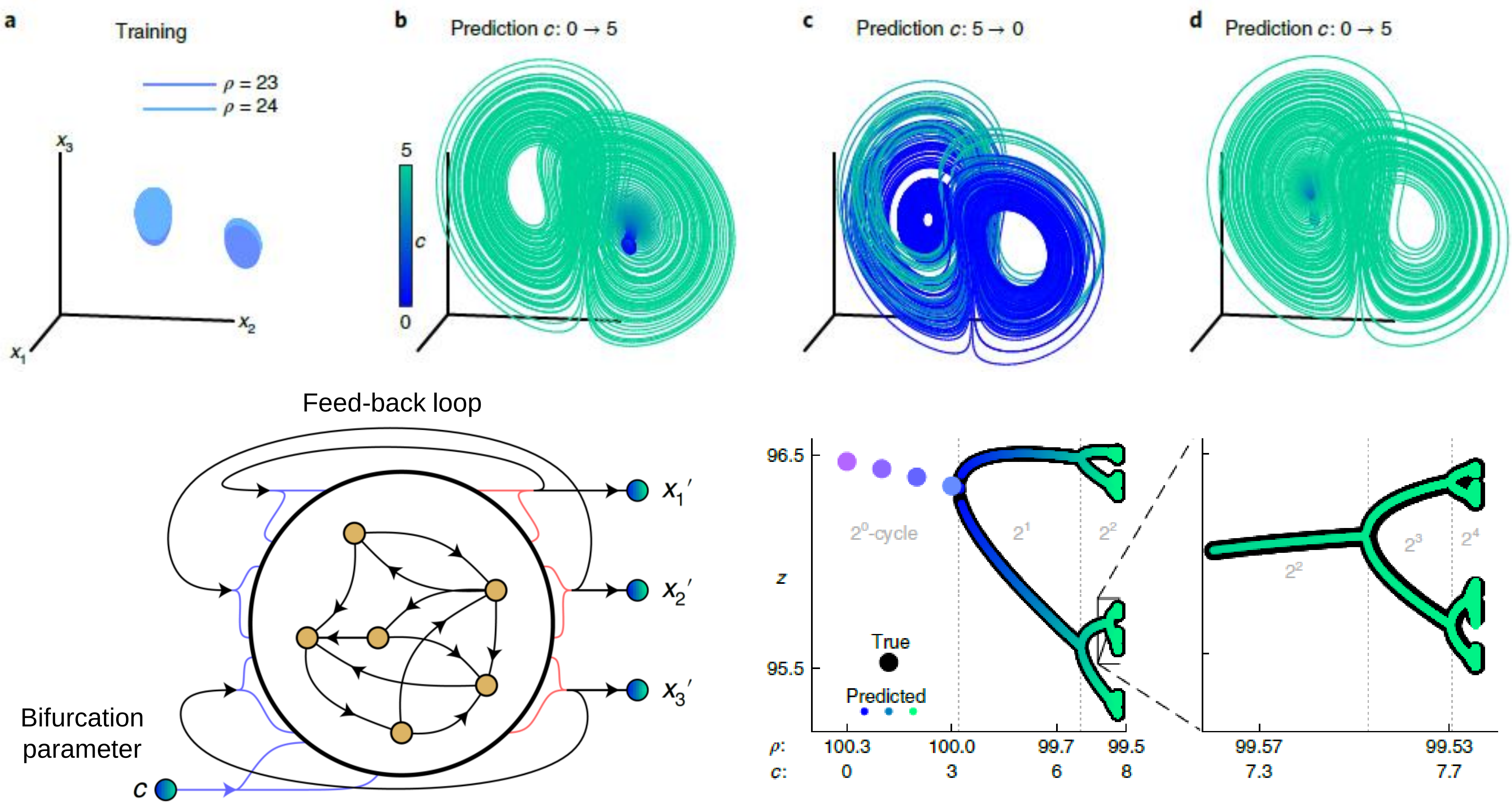


Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

Transition probability and duration for each pattern are designed successfully!

Learning bifurcation

Kim, J. Z., et. al., (2021). Nature Machine Intelligence, 3(4), 316-323.

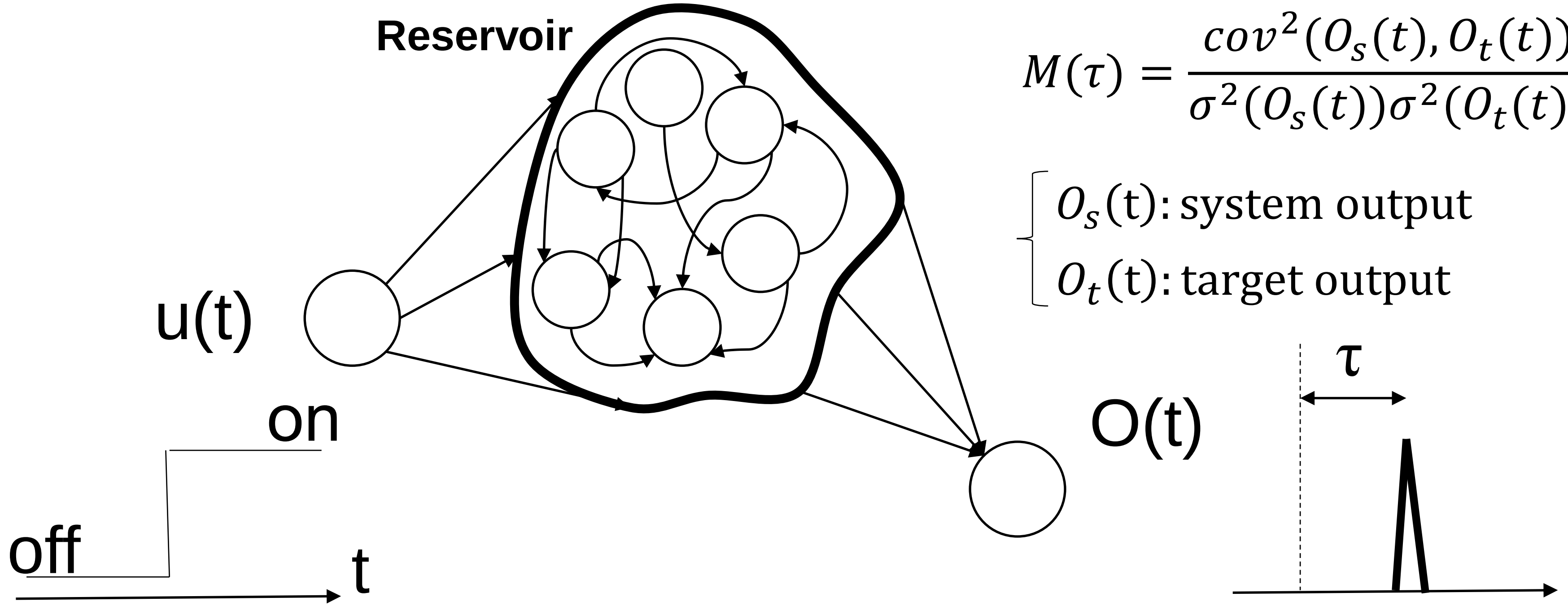


- Reservoir can learn bifurcation structures only by presenting training data from limited parameter range!

Case 3: Timer task

R. Laje et al. Nature Neurosci. 16: 925–933 (2013).
H. Jaeger GMD Report 152 (60 pp.) (2001).
H. Jaeger GMD Report 148 (43 pp.) (2001).

If the input (u(t)) is switched “on”, should output “1” after τ timestep!

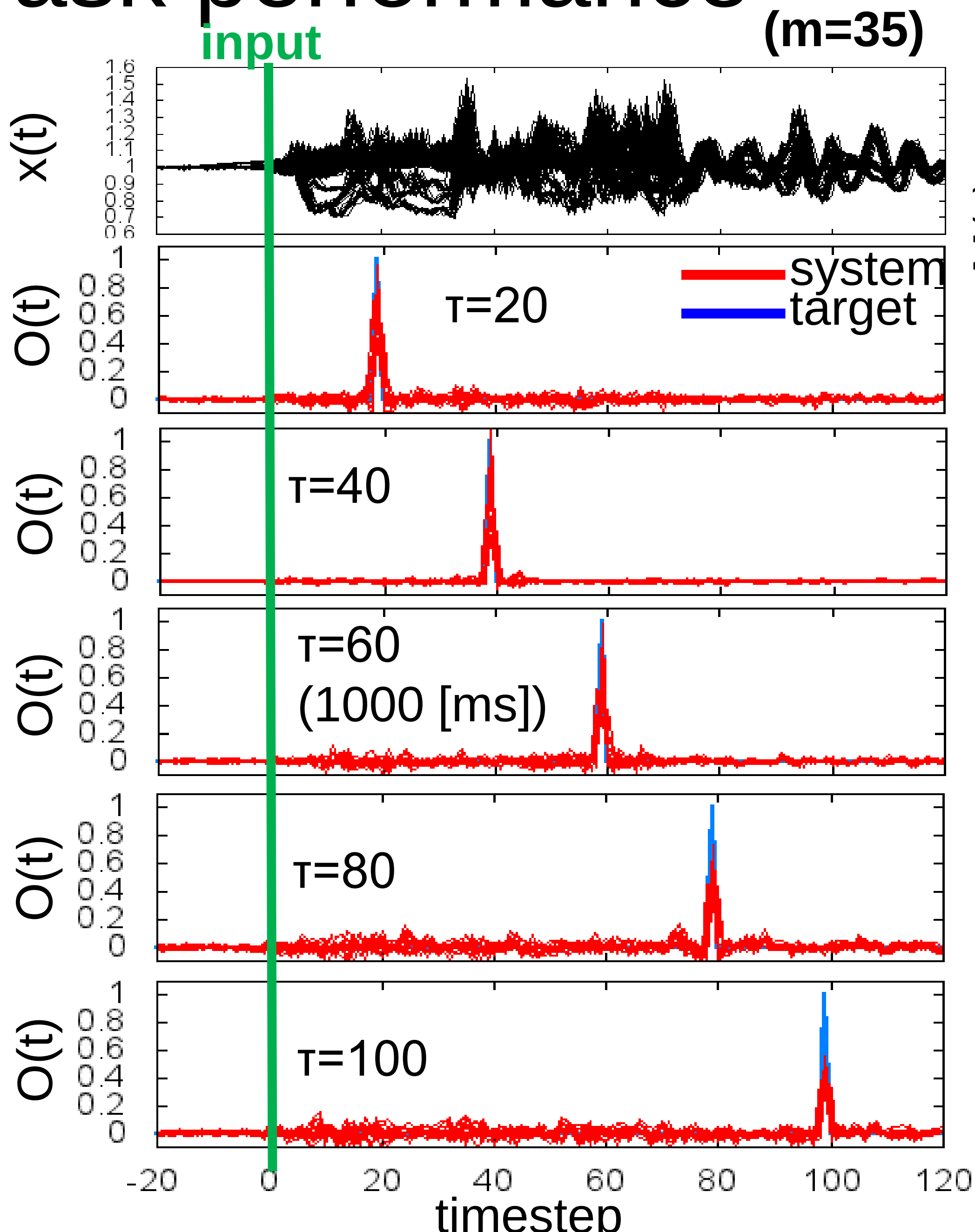


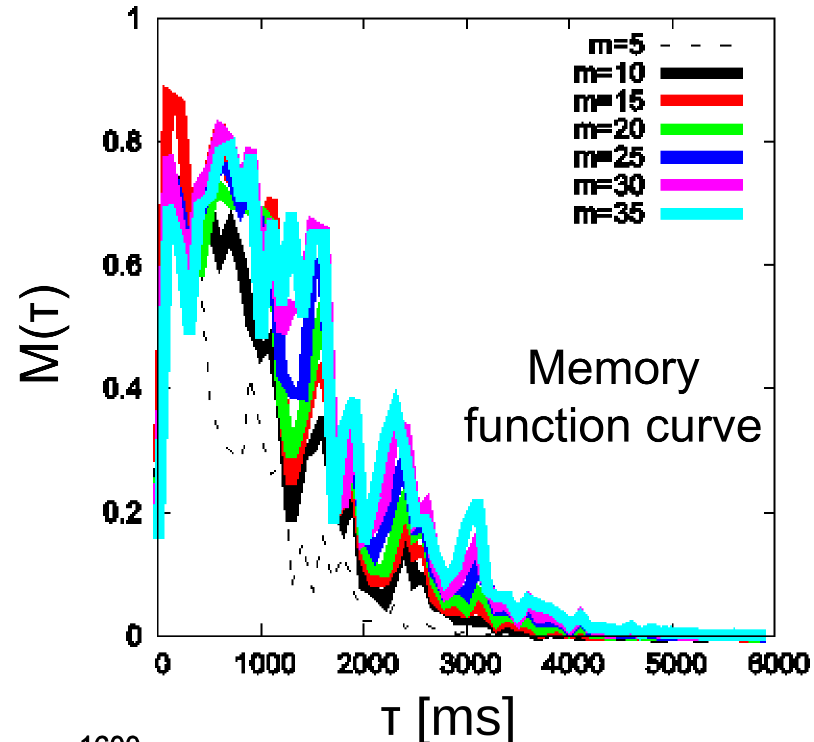
$$M(\tau) = \frac{cov^2(O_s(t), O_t(t))}{\sigma^2(O_s(t))\sigma^2(O_t(t))}$$

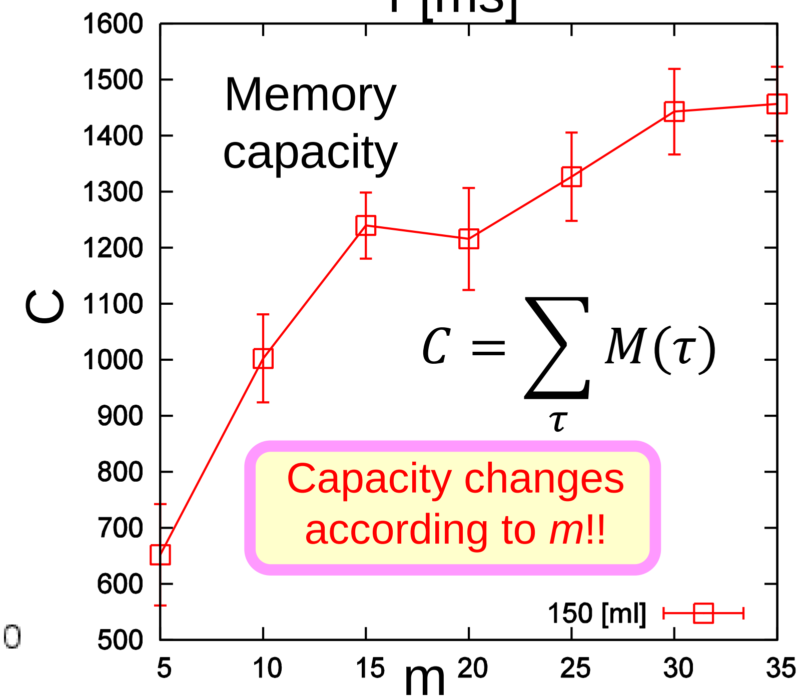
$O_s(\mathrm{t})$: system output
$O_t(\mathrm{t})$: target output

- Press the stopwatch in exact timing with your eyes closed!
- Should recognize a duration of time!
- Requires memory to perform the task!

Task performance







Water surface as a reservoir!

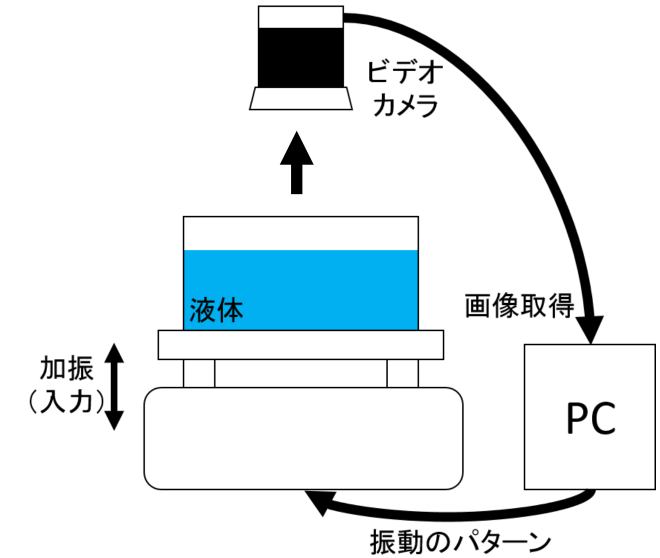
Cornstarch (Non-Newtonian fluid)

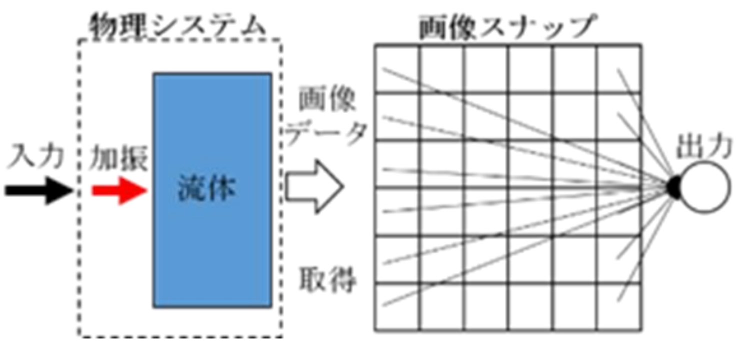
“Physical reservoir computing”

"Physical liquid state machine"

(Overall system)

K. Nakajima, T. Aoyagi, The Memory Capacity of a Physical Liquid State Machine, IEICE Technical Report vol.115. No.300, pp.109-112, 2015.

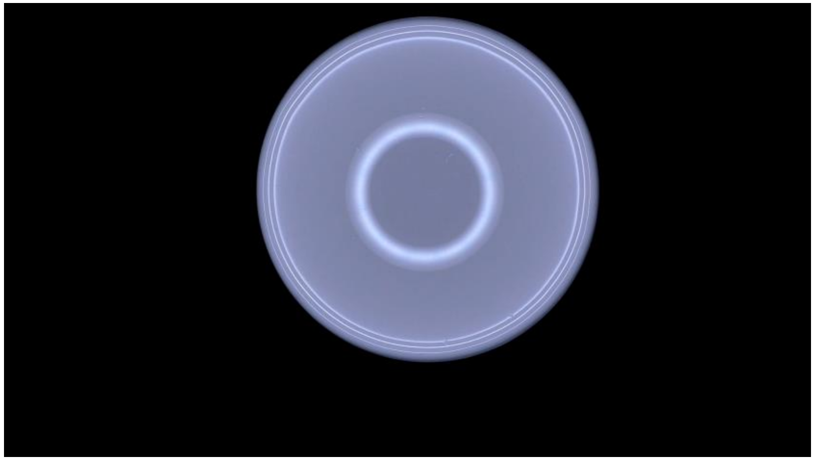




情報処理の過程

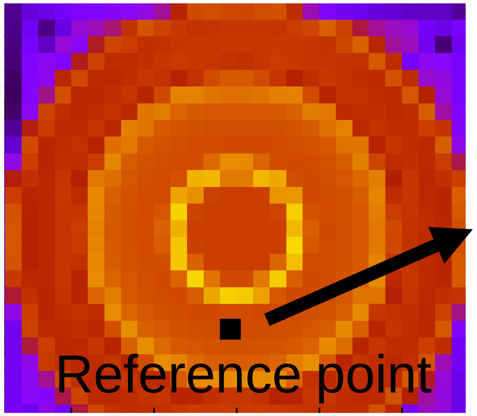
Dynamics of the water surface as computational resources!

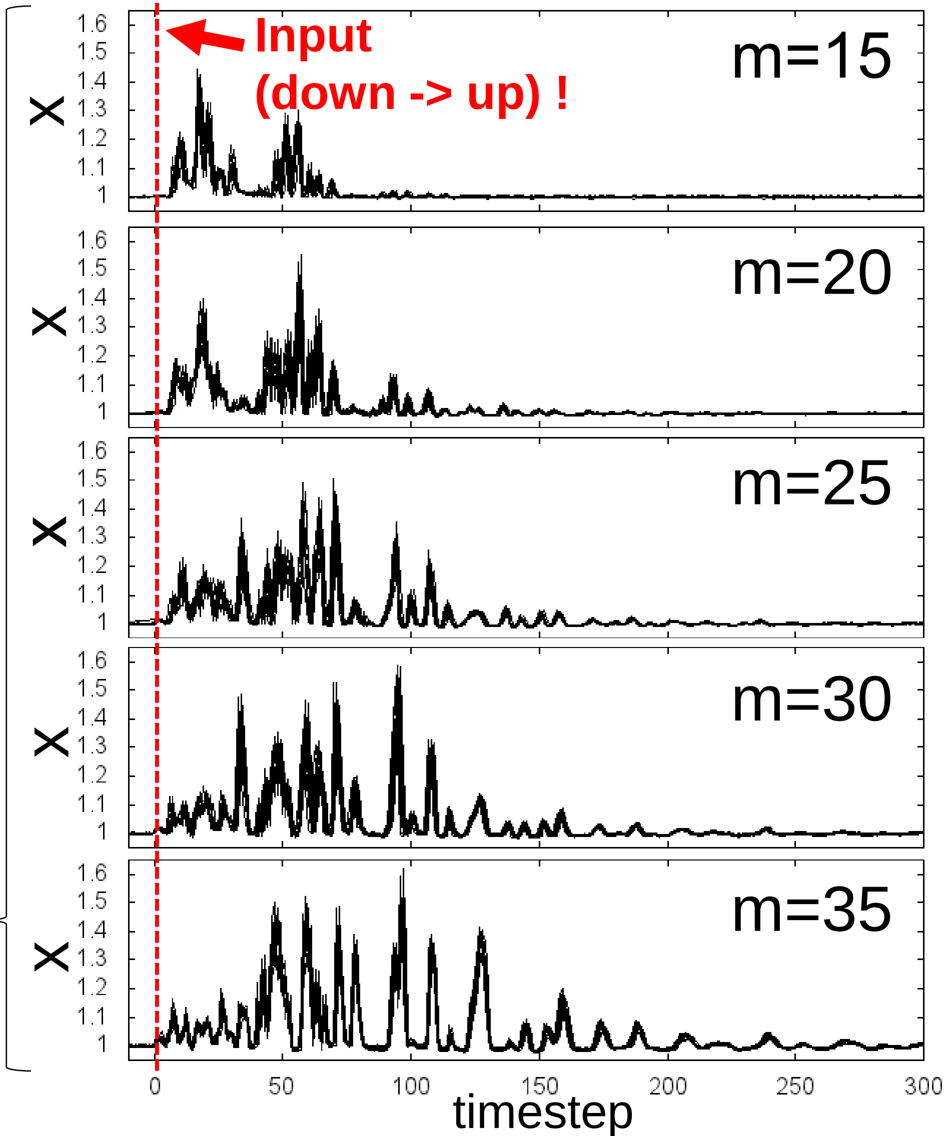
(Step response)



m=35

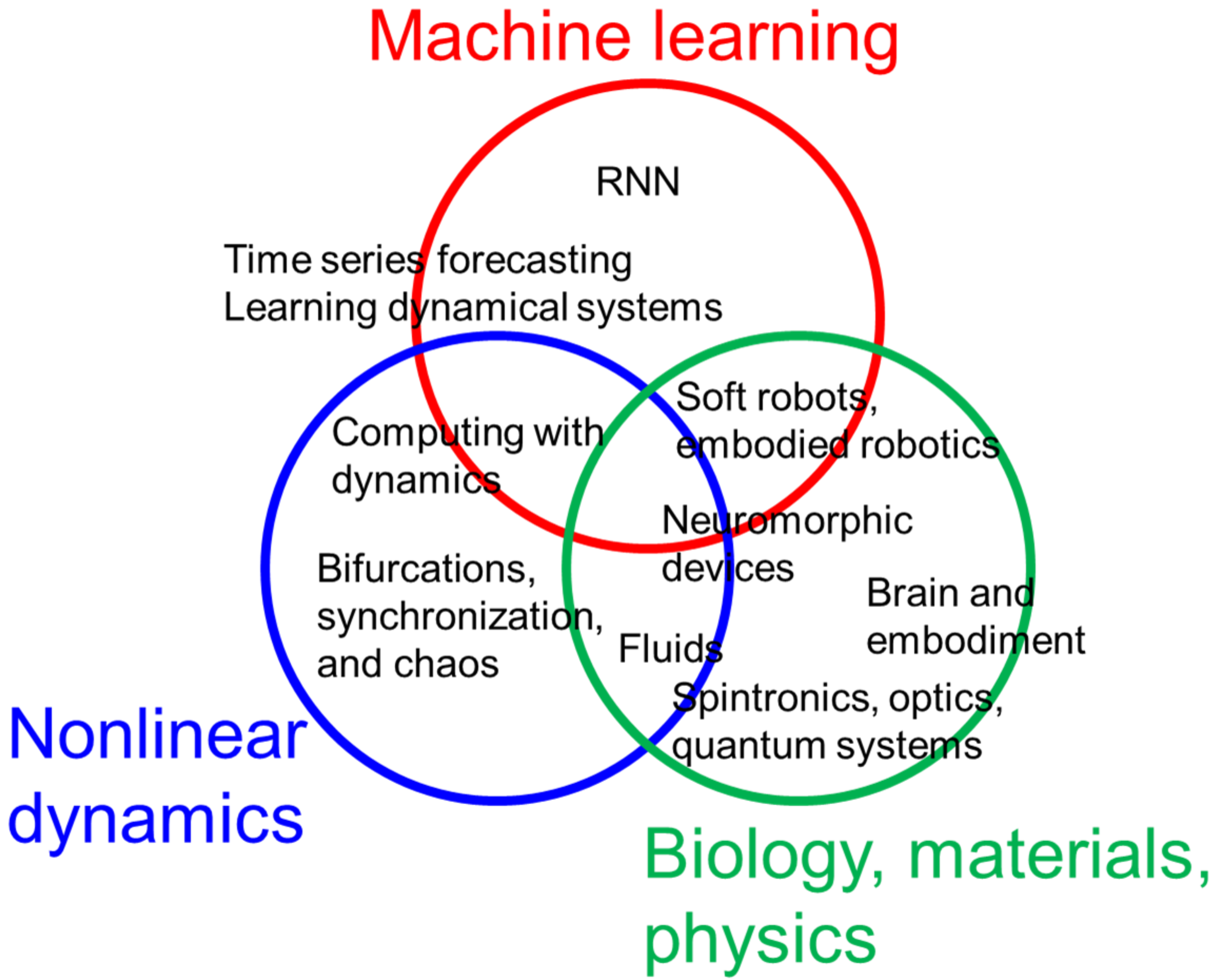
Response curve according to one shot from down to up with motor intensity *m*!





- For each grid, the summation of all the rgb values of the pixels are used for grid state *x*!
- Time series depends on the strength of the motor command!

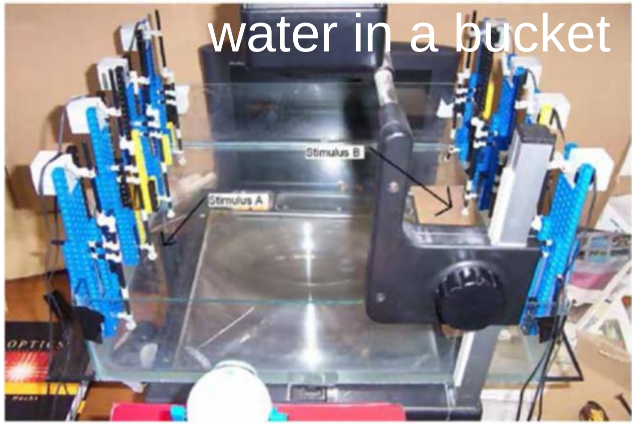
PRC as an interdisciplinary field



Physical reservoirs …

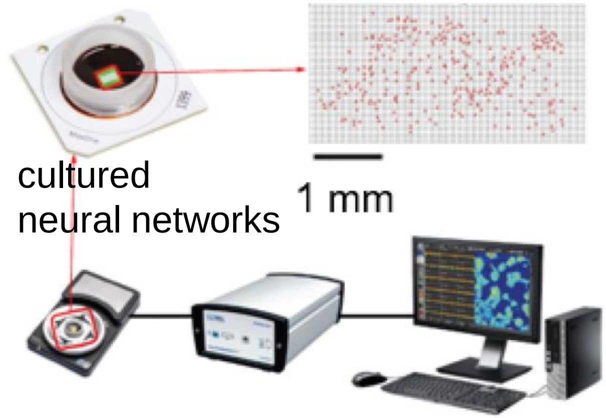
K. Nakajima, Physical reservoir computing---an introductory perspective, Jap. J. Appl. Phys. 59: 060501, 2020.

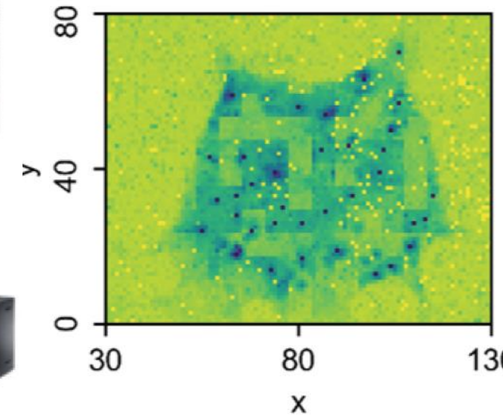
Liquid brain



C. Fernando et. al., Lec. Comp. Sci. 2801 (2003).

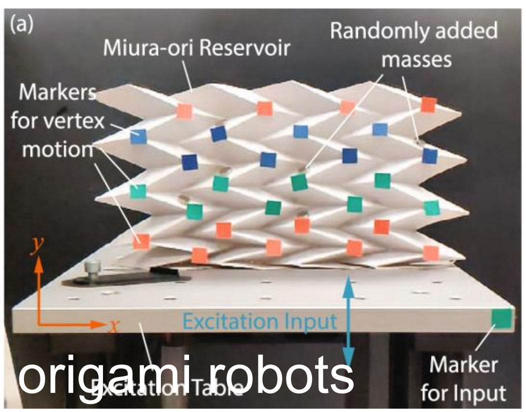
Cultured neural networks

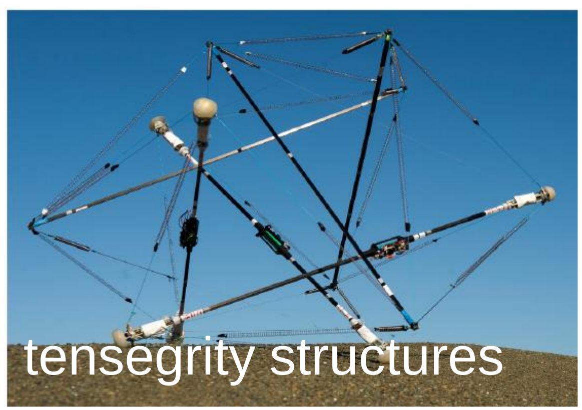


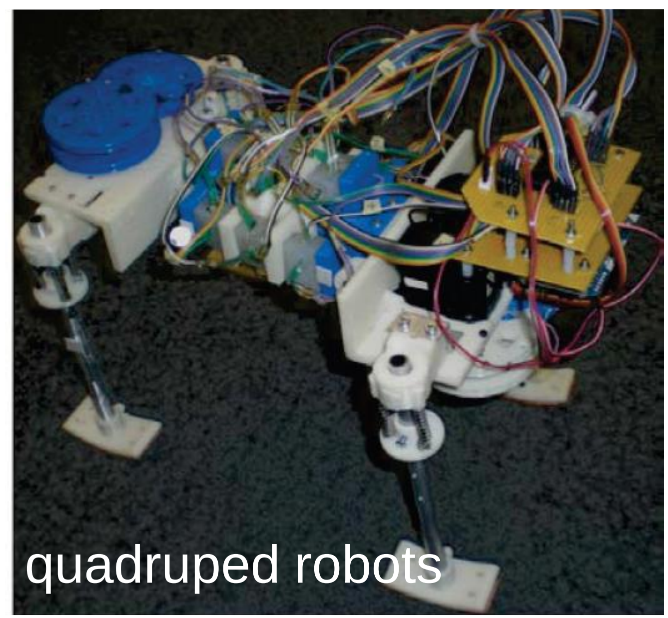


M. R. Dranias, et. al., J. Neurosci. 33, 1940 (2013).
T. Kubota, et. al., Lect. Comp. Sci. 11731 (2019).
Y. Yada, et. al., Appl. Phys. Lett., 119(17), 173701 (2021).

Soft robots



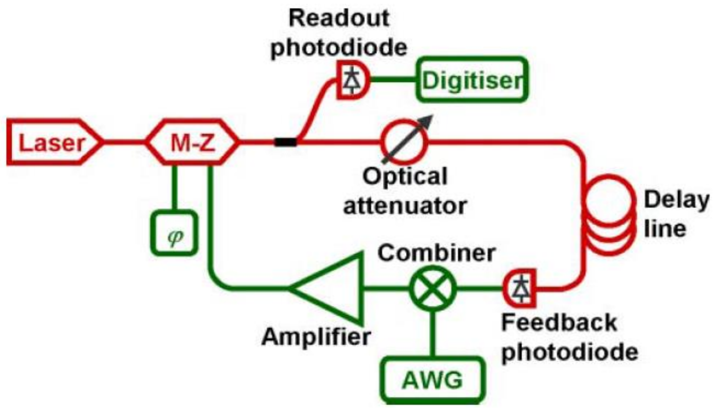




Q. Zhao et al., Proceedings of IROS, pp. 1445-1451 (2013).
K. Nakajima et al. J. R. Soc. Interface. 11: 20140437 (2014).
K. Caluwaerts et al. J. R. Soc. Interface 11:98 (2014).
K. Nakajima et al. Sci. Rep. 5: 10487 (2015).
K. Nakajima et al. Soft Robotics 5: 10487 (2018).
P. Bhovad, et. al., Sci. Rep. 11(1), 1-18 (2021).

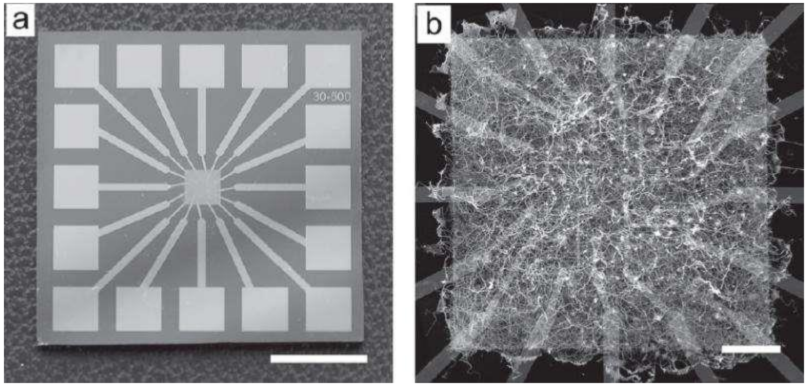
PRC for neuromorphic devices

Photonic reservoirs



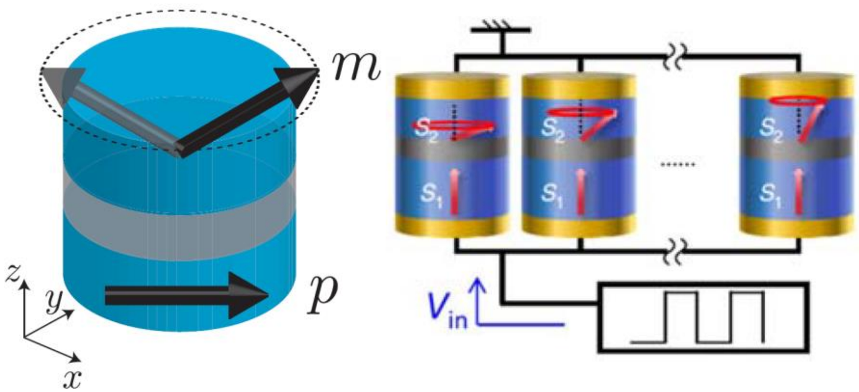
L. Larger, et. al., Opt. Express 20, 3241 (2012).
D. Brunner, et. al., Nat. Commun. 4, 1364 (2013).
K. Vandoorne, et. al., Nat. Commun. 5, 3541 (2014).
L. Larger, et. al., Phys. Rev. X 7, 011015 (2017).

In-Materia reservoirs



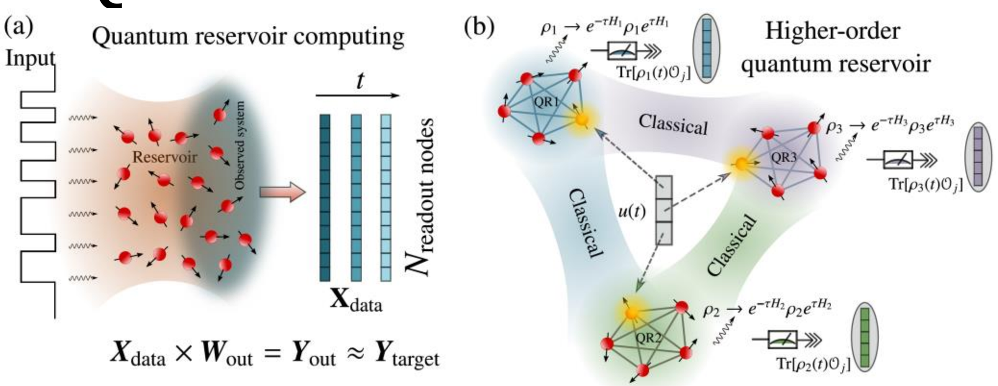
E. C. Demis et al. Nanotechnology 26:204003 (2015).
A. Z. Stieg et al. Adv. Mater. 24:286-293 (2012).
M. Cucchi, et. al., Science Advances, 7(34), eabh0693 (2021).
Y. Usami, et. al., Adv. Mater. (2021).

Spintronics reservoirs



J. Torrejon et al., Nature 547, 428 (2017).
T. Furuta, et. al., Phys. Rev. Appl. 10, 034063 (2018).
S. Tsunegi, et. al., Appl. Phys. Lett. 114, 164101 (2019).
N. Akashi, et. al., Phys. Rev. Res. 2: 043303 (2020).

Quantum reservoirs



K. Fujii, K. Nakajima, Phys. Rev. Appl. 8: 024030 (2017).
K. Nakajima, et. al., Phys. Rev. Appl. 11: 034021 (2019).
S. Ghosh, et. al., Adv. Quantum Technol. 4: 2100053 (2021).
Q. H. Tran, K. Nakajima, Phys. Rev. Lett. 127: 260401 (2021).

To be continued on 11/10