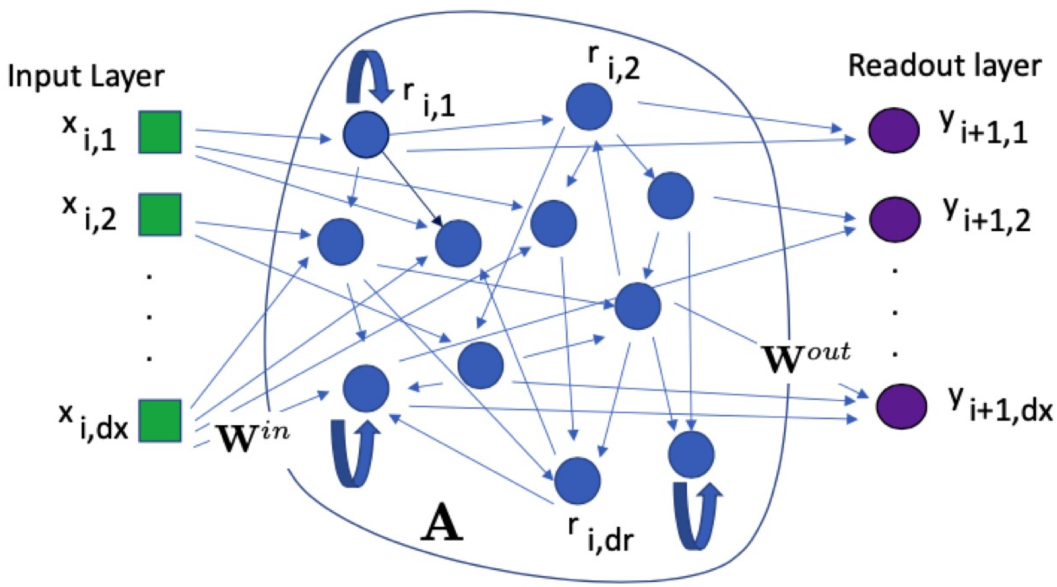
An RC is meant to **– Predict Future from the Past –** (supervised) trained on example data
A Reservoir Computer is some kind of **crazy random RNN** – for time series forecasting -
It works GREAT!

My question is – why does it work at all with all sorts of random parameters



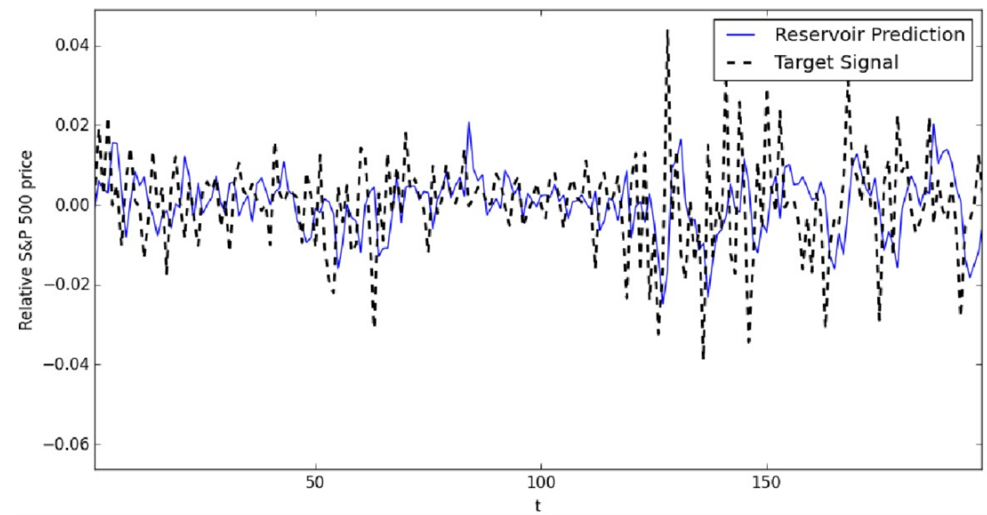


Figure 5.12: Target data ytarget(n) compared to the outcome y(n) from the reservoir.

Published in 2015
Reservoir Computing in Forecasting Financial Markets
J. Su

A Neural Net but
-no distinct layers
-has a memory

Turns out that A Reservoir Computer is some kind of crazy - a random RNN – but it is actually something very classical - a classical VAR(k) – *a star from Econometrics* - *and stochastic processes*

an autoregressive model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

So what? There is a very well developed theory for AR and VAR – notably an existence of representation Theorem by WOLD

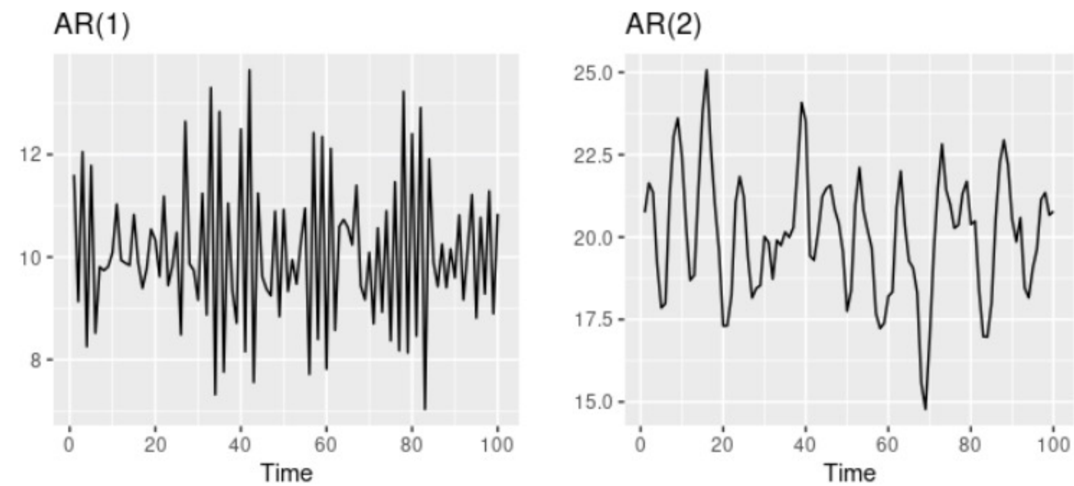


Figure 8.5: Two examples of data from autoregressive models with different parameters. Left: AR(1) with $y_t = 18 - 0.8y_{t-1} + \varepsilon_t$. Right: AR(2) with $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \varepsilon_t$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

For an AR(1) model:

- when $\phi_1 = 0$, y_t is equivalent to white noise;
- when $\phi_1 = 1$ and $c = 0$, y_t is equivalent to a random walk;
- when $\phi_1 = 1$ and $c \neq 0$, y_t is equivalent to a random walk with drift;
- when $\phi_1 < 0$, y_t tends to oscillate around the mean.

What-Why – **Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process**

Statistical analysis and time-series models for minimum/maximum temperatures in the Antarctic Peninsula

By Gillian L. Hughes[1], Suhasini Subba Rao[2] and Tata Subba Rao[1,*]

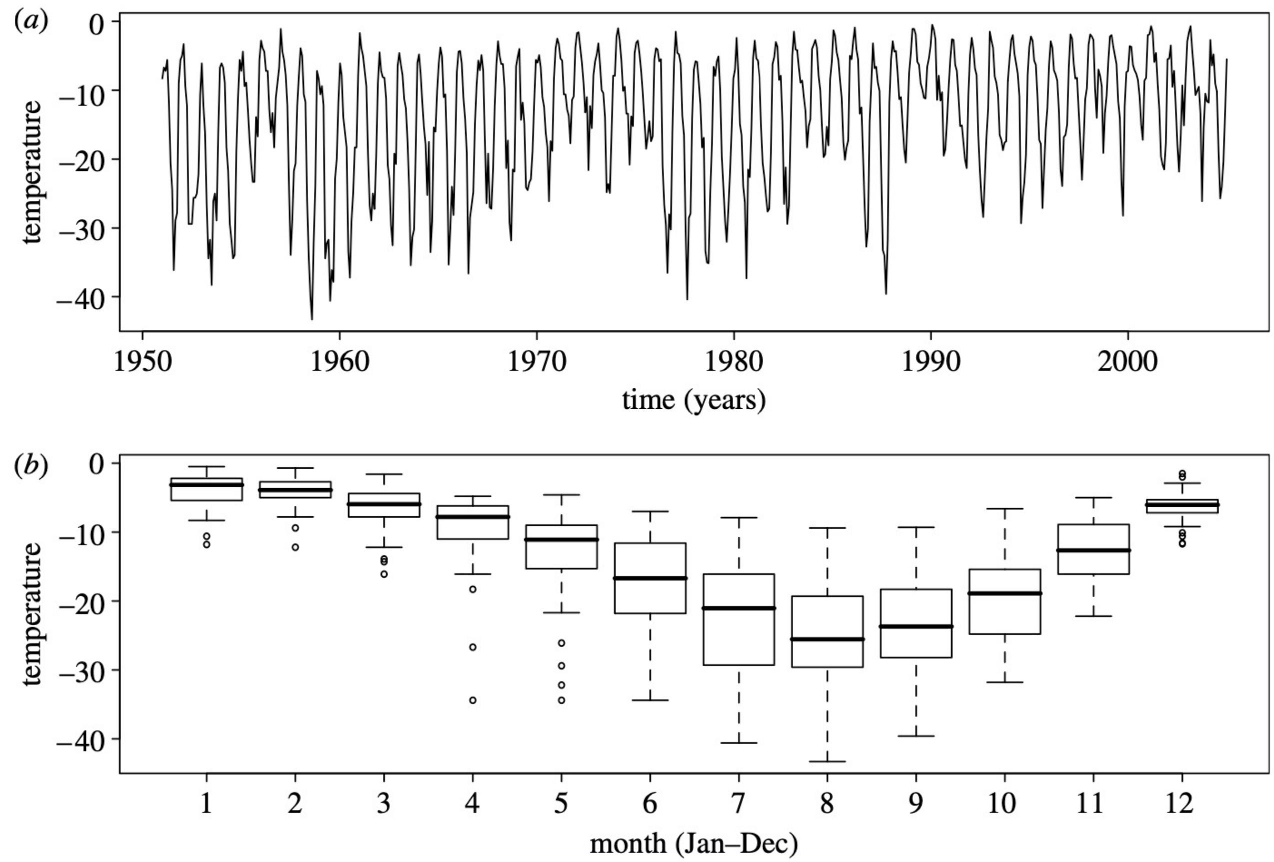


Figure 1. (a) Minimum monthly temperatures and (b) boxplot of minimum monthly temperatures at the Faraday station (January 1951–December 2004).

What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process

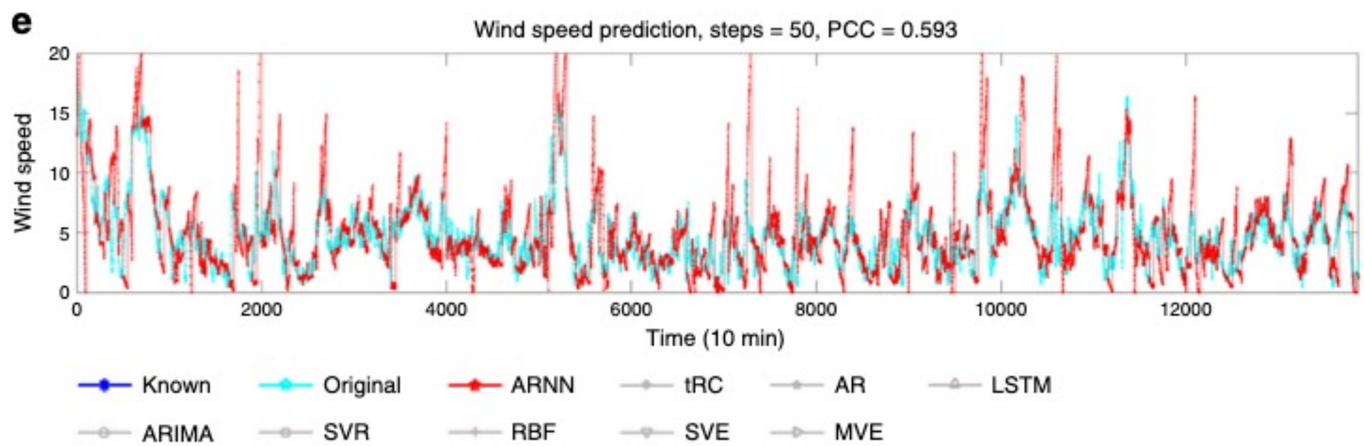
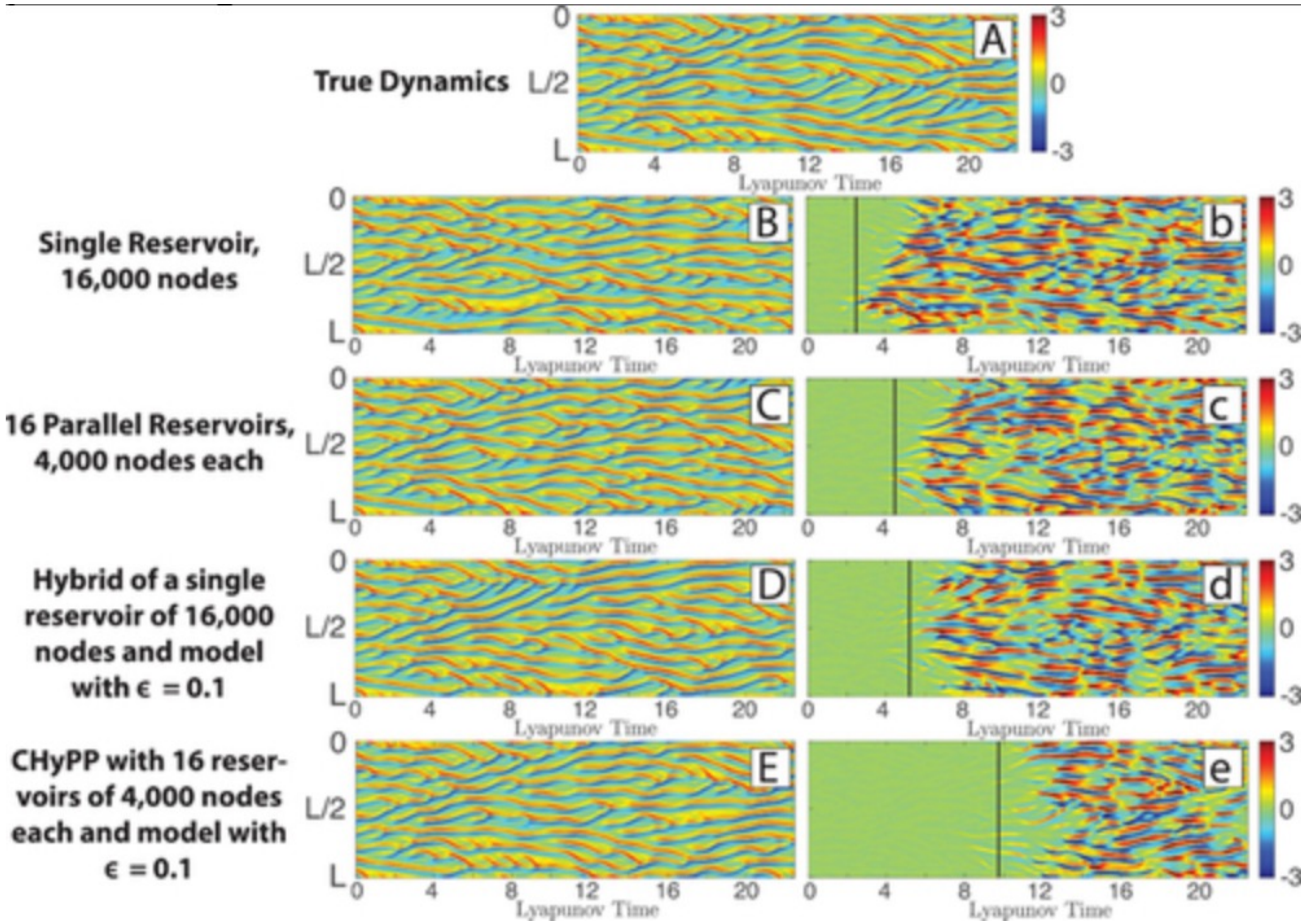


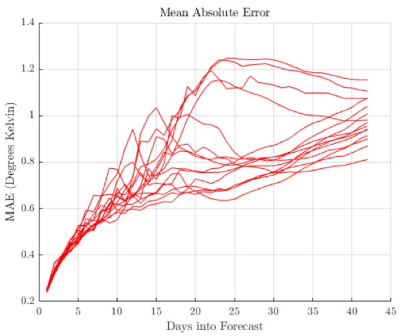
Fig. 3 Wind speed prediction in Wakkanai, Japan. Based on the time-course data of $D = 155$ sampling sites in Wakkanai, Japan, ARNN was applied to

What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process



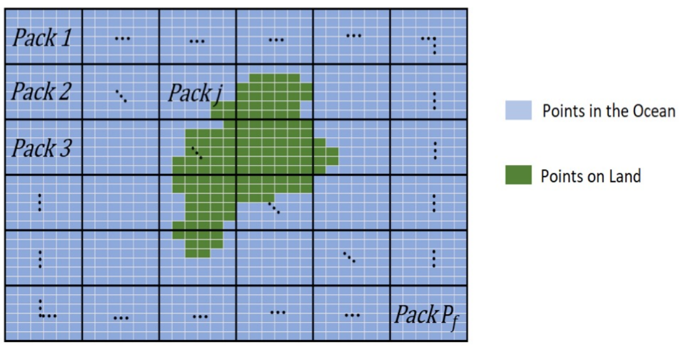
Wikner, Alexander, et al. "Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems." *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.5 (2020): 053111.

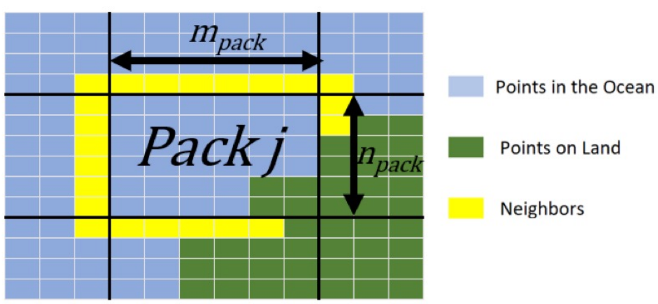
What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process

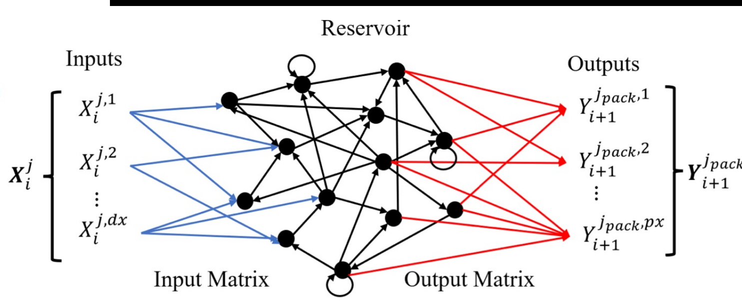


The mean absolute error for the 6 week forecast

A substantial and spatiotemporally complex data set of significance – SST Earth







Walleshauser, Bollt. "Predicting Sea Surface Temperatures with Coupled Reservoir Computers." *Nonlinear Processes in Geo Disc*(2022): 1-19.

What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process

Market Analysis

Search the site ...

REPORTS ABOUT SPONSOR CONTACT

Reservoir Computing Market Analysis

December 25, 2019

The report covers reservoir computing niche market segments in selected verticals where reservoir computing is a game changer. 2020 is set to shatter reservoir computing spending records across R&D, applications, verticals and especially government sector.

Interested in more of this research and a chance to learn about opportunities in the reservoir computing market? The report features data-driven insights from our market intelligence platform.

Reservoir Computing Market Analysis, December 2019, Single User License: $5,950.00
Reports are delivered in PDF format within 48 hours.
The report provides quantitative market analysis in a concise tabular format. The tables/charts present a focused snapshot of market dynamics.
Buy from 2CO
2CheckOut.com Inc. (Ohio, USA) is an authorized retailer for goods and services provided by Market Research Media Ltd.

Reservoir Computing Market Analysis, December 2019, Global Site License: $9,950.00
Reports are delivered in PDF format within 48 hours.
The report provides quantitative market analysis in a concise tabular format. The tables/charts present a focused snapshot of market dynamics.
Buy from 2CO
2CheckOut.com Inc. (Ohio, USA) is an authorized retailer for goods and services provided by Market Research Media Ltd.

Filed Under: Reports
Tagged With: AI, deep learning, machine learning, machine learning technology, neural network, Reservoir Computing

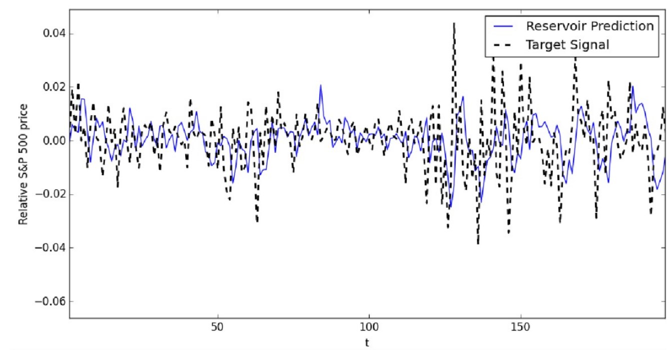


Figure 5.12: Target data ytarget(n) compared to the outcome y(n) from the reservoir.

Published in 2015
Reservoir Computing in Forecasting Financial Markets
J. Su

What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process

Market Analysis

REPORTS ABOUT SPONSOR CONTACT

Reservoir Computing Market Analysis

December 25, 2019

The report covers reservoir computing niche market segments in selected verticals where reservoir computing is a
computing spending records across R&D, applications, verticals and especially government sector.

Interested in more of this research and a chance to learn about opportunities in the reservoir computing market? Th
market intelligence platform.

Reservoir Computing Market Analysis, December 2019, Single User License: $5,950.00
Reports are delivered in PDF format within 48 hours.
The report provides quantitative market analysis in a concise tabular format. The tables/charts present a focused s
Buy from 2CO
2CheckOut.com Inc. (Ohio, USA) is an authorized retailer for goods and services provided by Market Research Medi

Reservoir Computing Market Analysis, December 2019, Global Site License: $9,950.00
Reports are delivered in PDF format within 48 hours.
The report provides quantitative market analysis in a concise tabular format. The tables/charts present a focused s
Buy from 2CO
2CheckOut.com Inc. (Ohio, USA) is an authorized retailer for goods and services provided by Market Research Medi

Filed Under: Reports
Tagged With: AI, deep learning, machine learning, machine learning technology, neural network, Reservoir Computing

Market Research Media Ltd

USD English

Reservoir Computing Market Analysis, Single User License	1	$5,950.00
Reservoir Computing: Government Markets, Single User License	1	$5,950.00

Total : $11,900.00

Pay securely with Credit card PayPal

Billing details

Personal Company

Email address*

Country* United States of America

City*

Zip / Postal code* State* Search

Card details

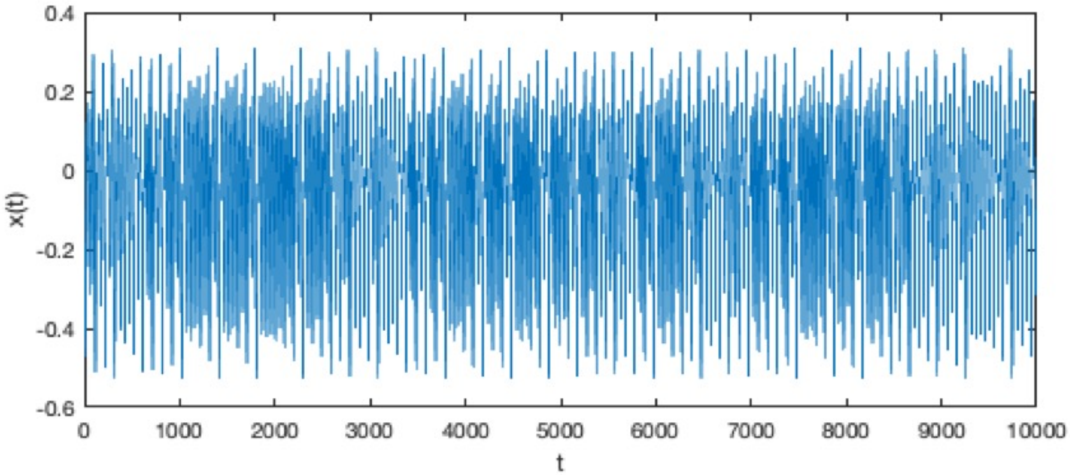
Card number*

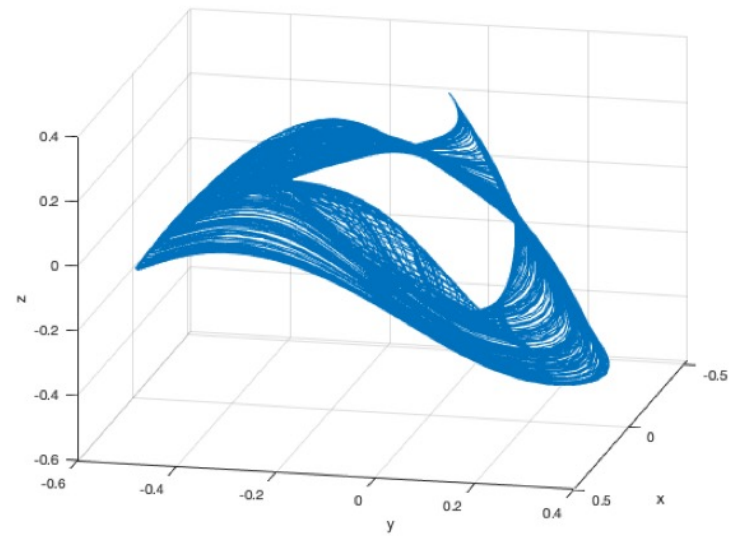
Expiration date* Security code*

Name on card*

Place order

What-Why – **Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process**





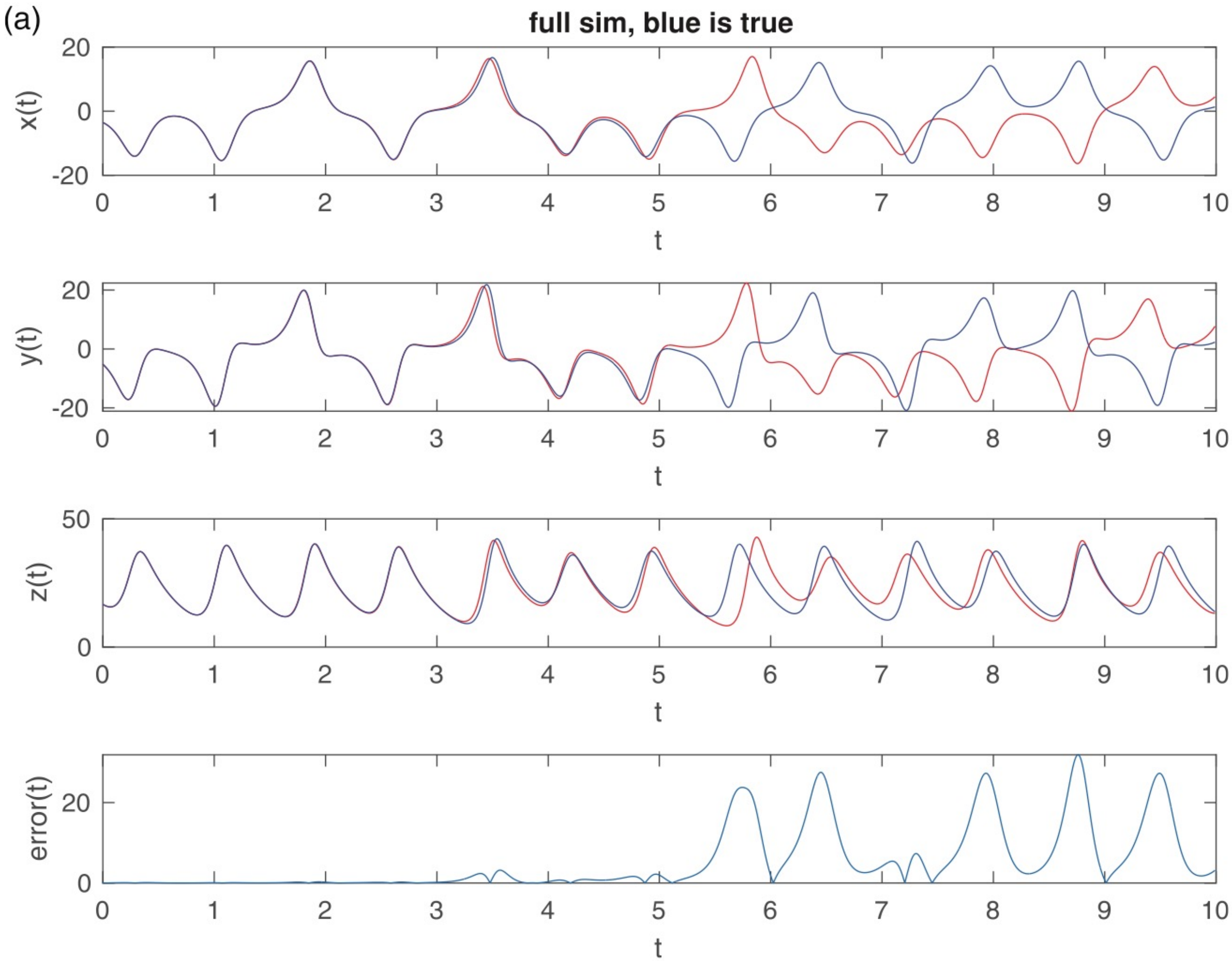
$\{X_t : t \in T\}$

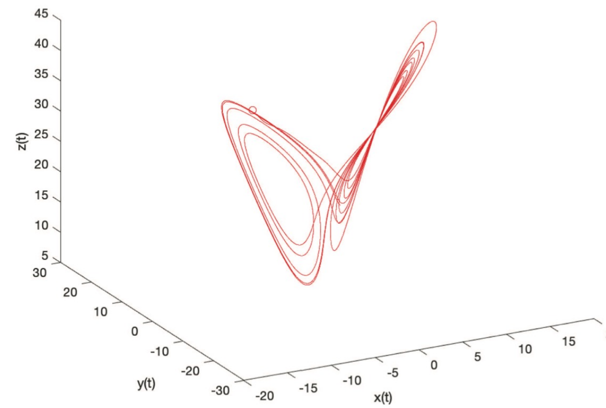
Example 1. First we numerically simulate a "noisy" dynamical system by adding white noise to the Mackey–Glass differential delay equations [Mackey & Glass, 1977],

$$x'(t) = \frac{ax(t - t_d)}{1 + [x(t - t_d)]^c} - bx(t) + \varepsilon\,, \qquad (18)$$

which has become a standard example in time-series

What-Why – Reservoir computing –forecast the future from time series data from some chaotic process or stochastic process





$\{X_t : t \in T\}$ **Dynamical System as Stochastic Process**

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = rx - y - xz$$
$$\dot{z} = xy - bz$$

Lorenz63

Reservoir computing – **a special case of RNN**
spec case ANN, Jaeger-Hass 2004, ESN-Jaeger 2001.

$\{\mathbf{x}_i\}_{i=1}^{N} \subset \mathbb{R}^{d_x}$ $\qquad d_r > d_x$

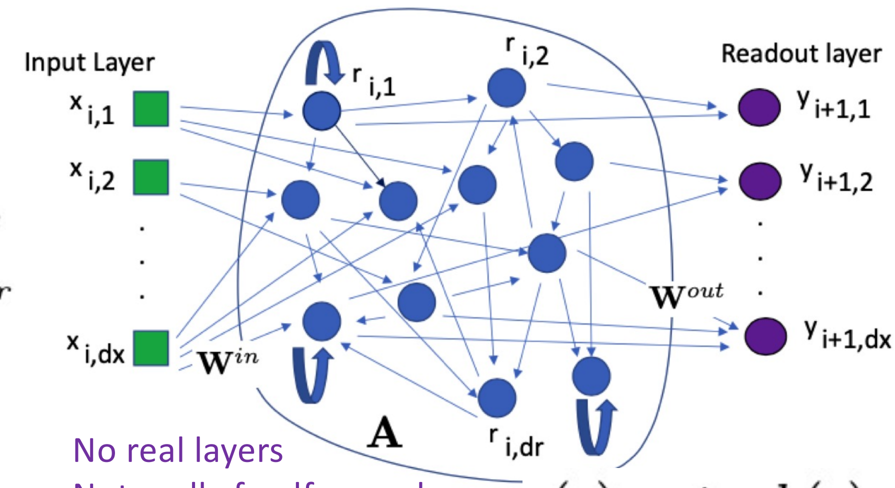
$$\mathbf{u}_i = \mathbf{W}^{in}\mathbf{x}_i, \qquad \mathbf{r}_i \in \mathbb{R}^{d_r}$$

$$\mathbf{r}_{i+1} = (1-\alpha)\mathbf{r}_i + \alpha q(\mathbf{A}\mathbf{r}_i + \mathbf{u}_i + \mathbf{b}),$$
$$\mathbf{y}_{i+1} = \mathbf{W}^{out}\mathbf{r}_{i+1}.$$

$\mathbf{W}^{in}_{i,j}$ $d_r \times d_x$ read in matrix

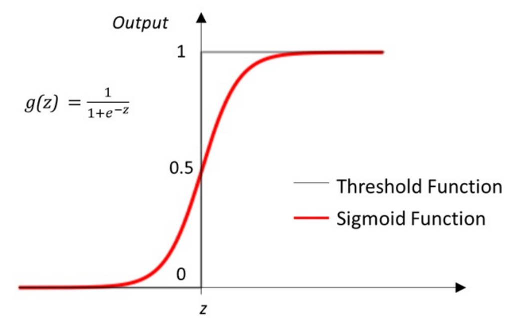
$\mathbf{A}_{i,j}$

$d_x \times d_r$ $\underline{\textit{trained read-out matrix} \text{ matrix } \mathbf{W}^{out}}$



No real layers
Not really feedforward

$$q(s) = tanh(s)$$
$$q(s) = s$$



Question – HOW can randomly chosen A and randomly chosen $\mathbf{W}^{in}$ but ONLY trained $\mathbf{W}^{out}$
Still have enough flexibility/freedom to yield a successful method?!

Reservoir computing – a special case of RNN
spec case ANN, Jaeger-Hass 2004, ESN-Jaeger 2001.

$$\{\mathbf{x}_i\}_{i=1}^{N} \subset \mathbb{R}^{d_x} \qquad\qquad d_r > d_x$$

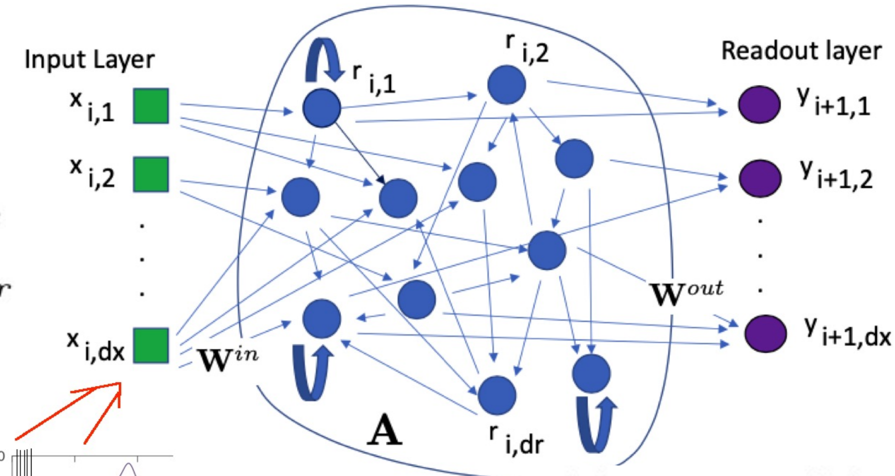
$$\mathbf{u}_i = \mathbf{W}^{in}\mathbf{x}_i, \qquad\qquad \mathbf{r}_i \in \mathbb{R}^{d_r}$$

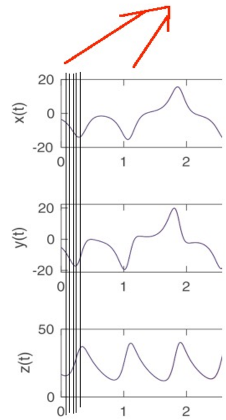
$$\begin{aligned} \mathbf{r}_{i+1} &= (1-\alpha)\mathbf{r}_i + \alpha q(\mathbf{A}\mathbf{r}_i + \mathbf{u}_i + \mathbf{b}), \\ \mathbf{y}_{i+1} &= \mathbf{W}^{out}\mathbf{r}_{i+1}. \end{aligned}$$

$\mathbf{W}^{in}_{i,j}$ $d_r \times d_x$ read in matrix

$\mathbf{A}_{i,j}$

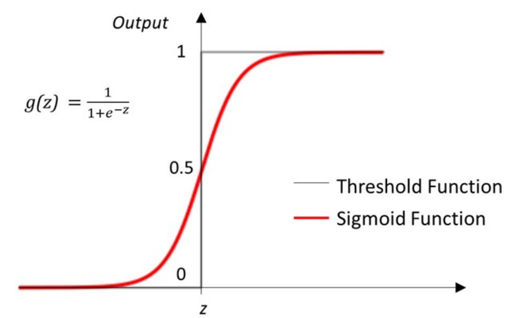
$d_x \times d_r$ *trained read-out matrix* matrix $\mathbf{W}^{out}$





$$q(s) = tanh(s)$$

$$q(s) = s$$



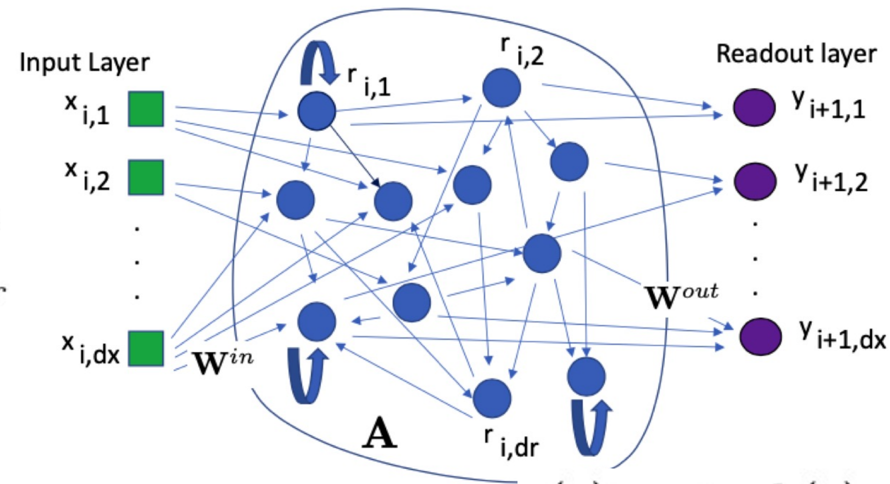
Question – HOW can randomly chosen A and randomly chosen $\mathbf{W}^{in}$ but ONLY trained $\mathbf{W}^{out}$
Still have enough flexibility/freedom to yield a successful method?!

Reservoir computing – a special case of RNN
spec case ANN, Jaeger-Hass 2004, ESN-Jaeger 2001.

$$\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^{d_x} \qquad d_r > d_x$$

$$\mathbf{u}_i = \mathbf{W}^{in}\mathbf{x}_i, \qquad \mathbf{r}_i \in \mathbb{R}^{d_r}$$

$$\begin{aligned} \mathbf{r}_{i+1} &= (1-\alpha)\mathbf{r}_i + \alpha q(\mathbf{A}\mathbf{r}_i + \mathbf{u}_i + \mathbf{b}), \\ \mathbf{y}_{i+1} &= \mathbf{W}^{out}\mathbf{r}_{i+1}. \end{aligned}$$

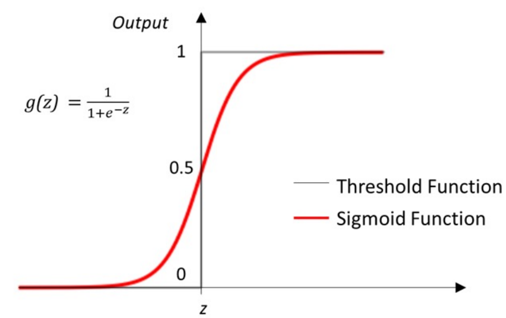


$$q(s) = tanh(s)$$

$$q(s) = s$$

$$\mathbf{W}_{i,j}^{in} \sim U(0,\gamma) \qquad d_r \times d_x \text{ read in matrix}$$

$$\mathbf{A}_{i,j} \sim U(-\beta,\beta), \text{ with } \beta \text{ to scale the spectral radius}$$



$$d_x \times d_r \ \underline{\textit{trained read-out matrix} \text{ matrix } \mathbf{W}^{out}}$$

Question – HOW can randomly chosen A and randomly chosen $\mathbf{W}^{in}$ but ONLY trained $\mathbf{W}^{out}$
Still have enough flexibility/freedom to yield a successful method?!

What-why? A Reservoir "Computer" – a special case of RNN
An RNN – a special case of an ANN – but good for time/sequential data processes

First a little background about ANN and Deep Learning

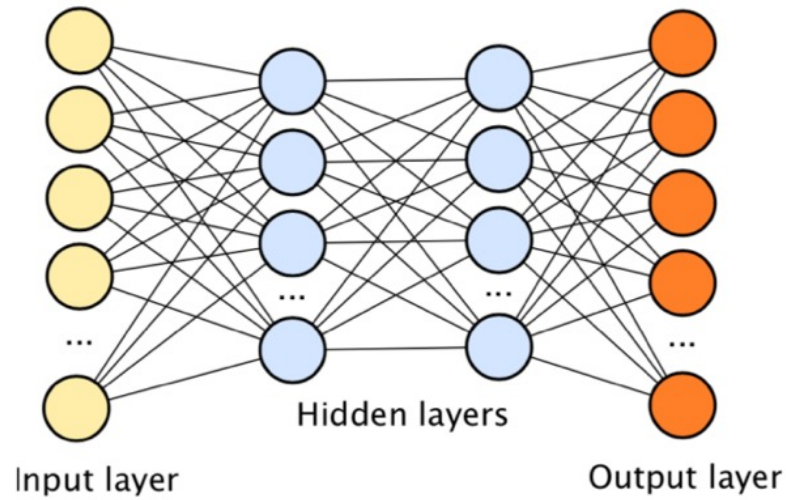
-what is an ANN? – useful for some kind of weird regression (supervised learning).

Given input data – predict output. $y=f(x)$.

What-why? A Reservoir "Computer" – a special case of RNN
An RNN – a special case of an ANN – but good for time/sequential data processes
-what is an ANN? – some kind of weird regression (supervised learning).

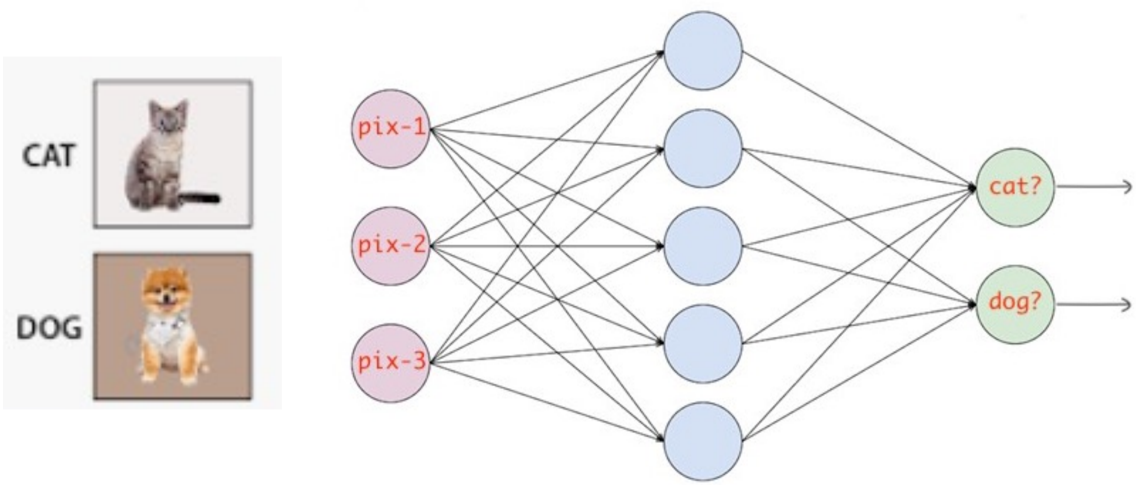
A little backround about Neural Nets/Deep Learning

Given input data – predict output. y=f(x).



"Deep" Feedforward Neural Network

A Classic Supervised Learning Problem

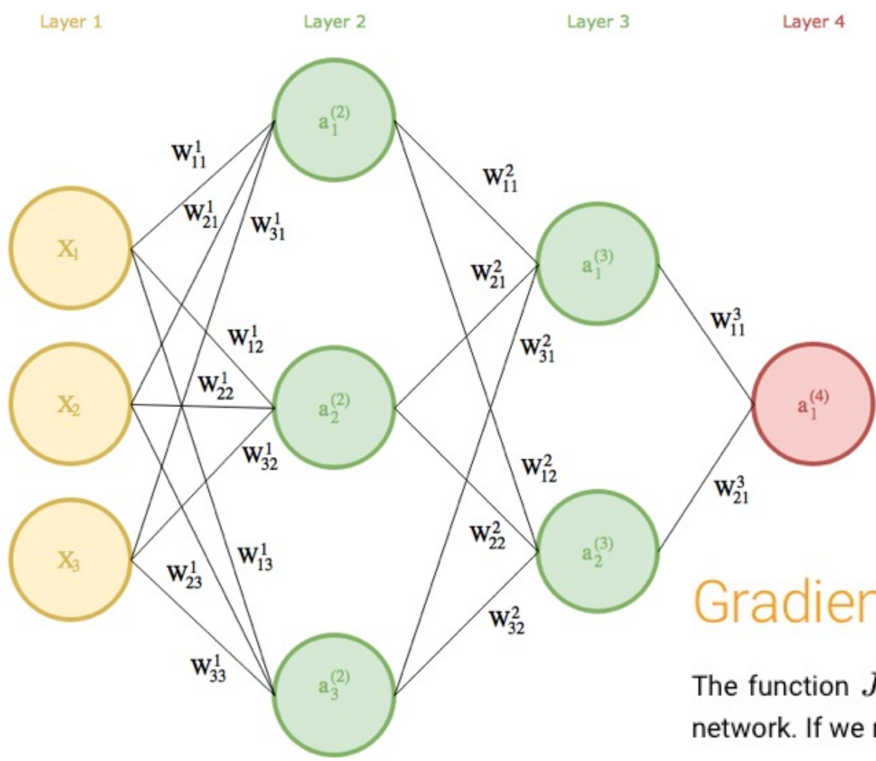


SLFN – Single Layer Feedforward Net

What does that graph notation mean – and how do you "train?"

A little backround about Neural Nets/Deep Learning

Training means finding the best weights to accommodate your given data



$$a_1^{(2)} = \tanh(Z_1^{(2)}) = \tanh(X_1 \times W_{11}^1 + X_2 \times W_{21}^1 + X_3 \times W_{31}^1 + b)$$

$$a_2^{(2)} = \tanh(Z_2^{(2)}) = \tanh(X_1 \times W_{12}^1 + X_2 \times W_{22}^1 + X_3 \times W_{32}^1 + b)$$

$$a_3^{(2)} = \tanh(Z_3^{(2)}) = \tanh(X_1 \times W_{13}^1 + X_2 \times W_{23}^1 + X_3 \times W_{33}^1 + b)$$

Gradient descent

The function $J(W)$ gives us the error of our network regarding our inputs X and the weights of our network. If we replace $\hat{y}$ by its calculations, our function is:

$$J(W) = \sum_{1}^{n} \frac{1}{2}(y - \tanh(\tanh(\tanh(X.W_1).W_2).W_3))^2$$

$$W_{n+1} \leftarrow W_n - \delta \nabla J_W(W_n)$$

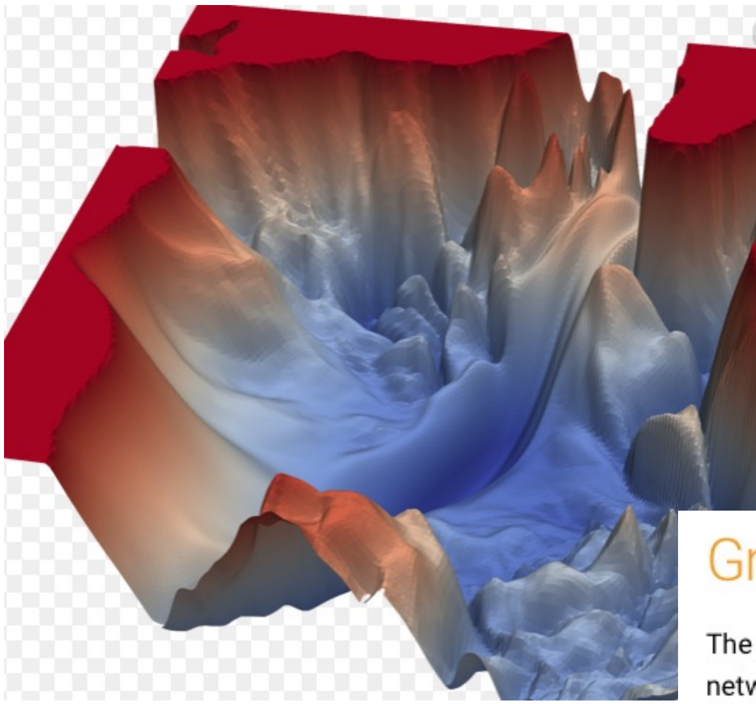
3*3+3+3*2+2+2+1=23 parameters here vs Say 2 (or 3).

ANN may have MANY weights – so a very high dimensional space of weights and a crazy loss function landscape to navigate

With your optimization method – can be very very expensive.

Two technologies to the rescue –

1) GPU
2) Stochastic gradient descent



Gradient descent

The function $J(W)$ gives us the error of our network regarding our inputs X and the weights of our network. If we replace $\hat{y}$ by its calculations, our function is:

$$J(W) = \sum_{1}^{n} \frac{1}{2}(y - \tanh(\tanh(\tanh(X.W_1).W_2).W_3))^2$$

$$W_{n+1} \leftarrow W_n - \delta \nabla J_W(W_n)$$

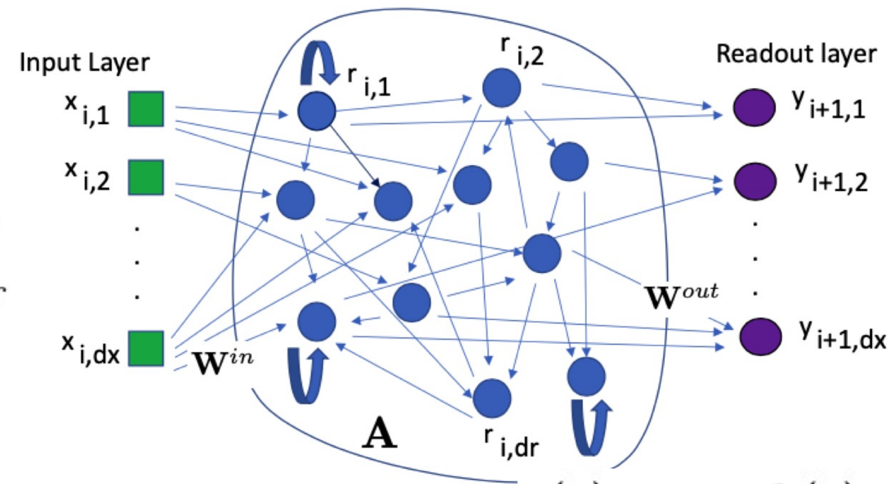
Now back to our main story about reservoir computing

Reservoir computing – **a special case of RNN**
spec case ANN, Jaeger-Hass 2004, ESN-Jaeger 2001.

$$\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^{d_x} \qquad d_r > d_x$$

$$\mathbf{u}_i = \mathbf{W}^{in}\mathbf{x}_i, \qquad \mathbf{r}_i \in \mathbb{R}^{d_r}$$

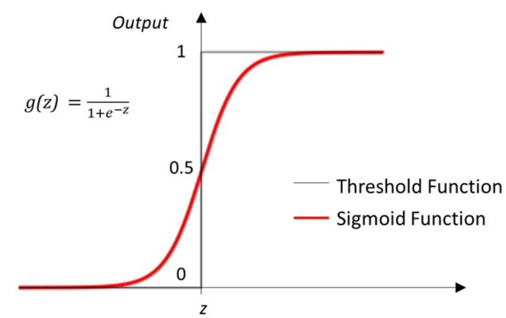
$$\mathbf{r}_{i+1} = (1-\alpha)\mathbf{r}_i + \alpha q(\mathbf{A}\mathbf{r}_i + \mathbf{u}_i + \mathbf{b}),$$
$$\mathbf{y}_{i+1} = \mathbf{W}^{out}\mathbf{r}_{i+1}.$$



$$q(s) = tanh(s)$$
$$q(s) = s$$

$$\mathbf{W}^{in}_{i,j} \sim U(0,\gamma) \quad d_r \times d_x \text{ read in matrix}$$

$$\mathbf{A}_{i,j} \sim U(-\beta,\beta), \text{ with } \beta \text{ to scale the spectral radius}$$

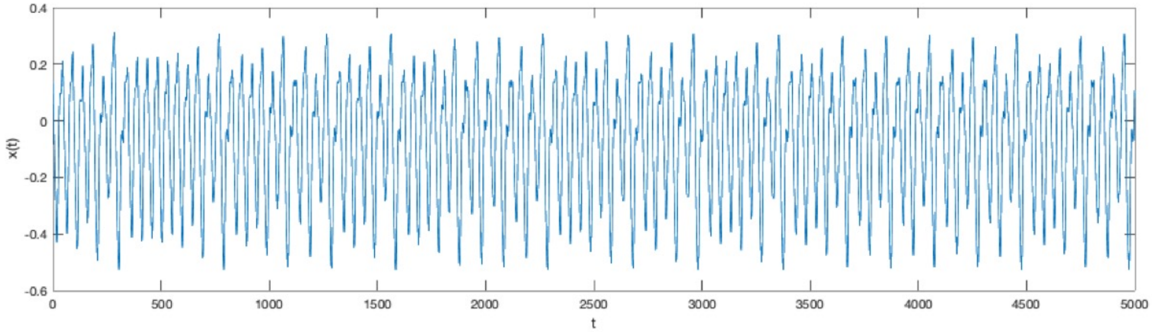


$$d_x \times d_r \textit{ trained read-out matrix} \text{ matrix } \mathbf{W}^{out}$$

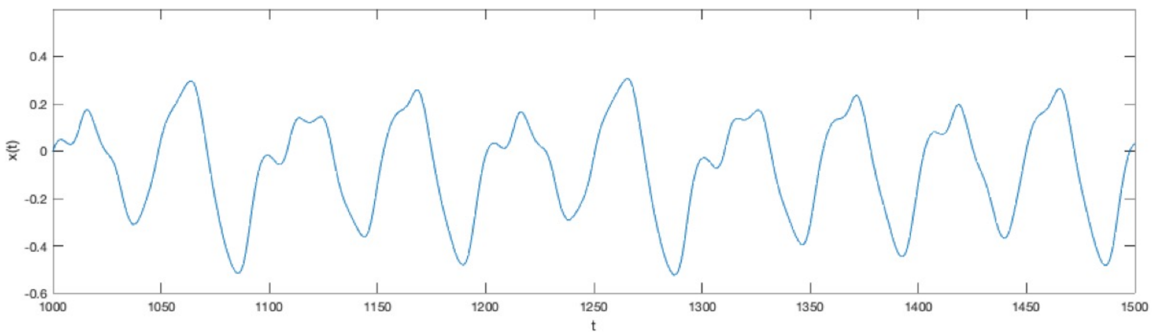
Question – HOW can randomly chosen A and randomly chosen $\mathbf{W}^{in}$ but ONLY trained $\mathbf{W}^{out}$
Still have enough flexibility/freedom to yield a successful method?!

Example RC for Mackey-Glass

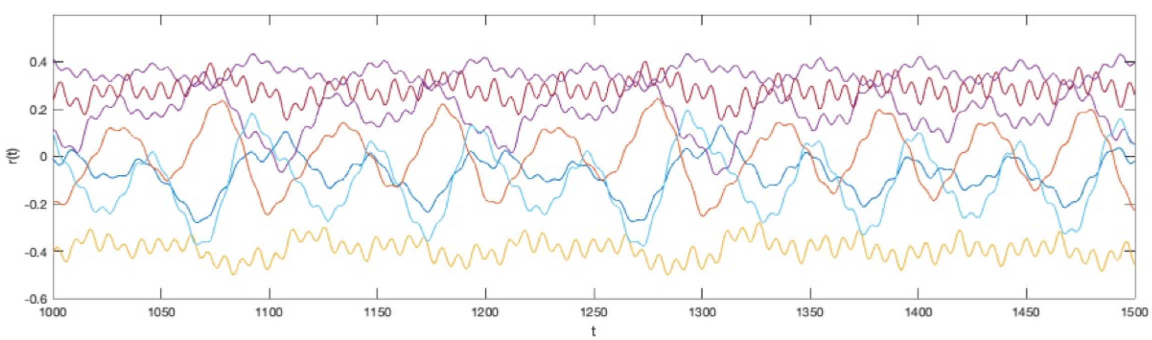
X(t)

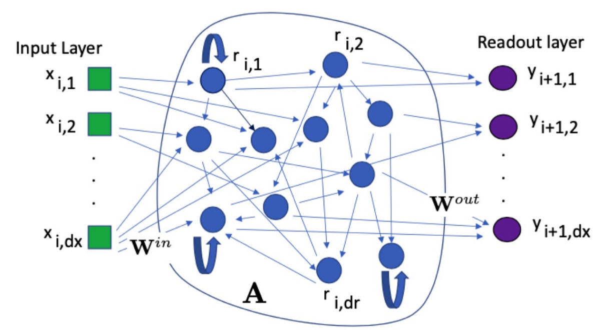


X(t)



r_i(t)





My question is – why does it work at all with all sorts of random parameters
Answer: time soaks up the random

Things people do ad-hoc to make it work better

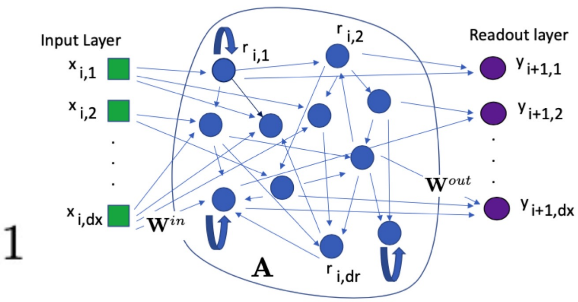
-distribution of A (e.g. by sparsity and scaling) to control spectral radius
-better read in distribution
-better read out matrix fitting method
-better threshold function q(s).

we will allow ourselves to make it worse! But in a way we can analyze.

Strip away as much of the idea as possible so while it still works to some degree to interpret what is happing more analytically.
-choose simple distributions for W_in and A. -We choose *a linear - identity threshold q(s)=s*

Punchline is now it become directly comparable to a vector autoregressive process – VAR –
-and with the VAR comes VMA which allows a representation theorem by WOLD
-also it has a bit like DMD-Koopman.

Fitting the readout matrix by (regularized) least squares



$$\mathbf{R} = [\mathbf{r}_{k+1}|\mathbf{r}_{k+1}|\ldots|\mathbf{r}_N],\ \ k \geq 1.$$

$$\mathbf{X} = [\mathbf{x}_{k+1}|\mathbf{x}_{k+1}|\ldots|\mathbf{x}_N] = [\mathbf{V}\mathbf{r}_{k+1}|\mathbf{V}\mathbf{r}_{k+2}|\ldots|\mathbf{V}\mathbf{r}_N] = \mathbf{V}\mathbf{R},\ \ k \geq 1$$

$$\mathbf{W}_{out} = \underset{\mathbf{V}\in\mathbb{R}^{d_x \times d_r}}{\arg\min} \|\underline{X} - \mathbf{V}\mathbf{R}\|_F = \underset{\mathbf{V}\in\mathbb{R}^{d_x \times d_r}}{\arg\min} \sum_{i=k}^{N} \|\mathbf{x}_i - \mathbf{V}\mathbf{r}_i\|_2,$$

(Tikhonov regularized – ridge regression) least squares solution

$$\mathbf{W}^{out} := \mathbf{X}\mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \lambda\mathbf{I})^{-1}$$

pseudo-inverse with the notation,

$$\mathbf{R}_\lambda^\dagger := \mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \lambda\mathbf{I})^{-1}$$

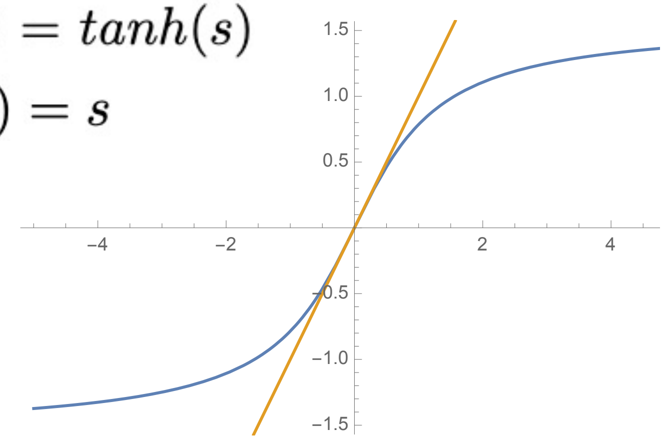
regularized singular value decomposition (SVD) in terms of regularized singular values such as $\sigma_i/(\sigma_i^2 + \lambda)$

RC With A Fully Linear Activation, $q(s) = s$, Yields a VAR(k)

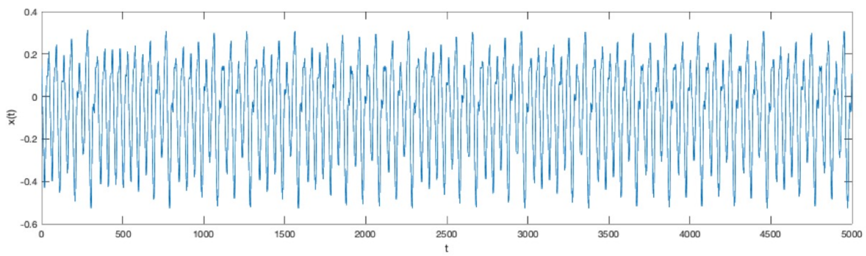
Note: $q(s) = tanh(s) \approx s - s^3/3 \ldots,$

$q(s) = tanh(s)$

$\dashrightarrow q(s) = s$

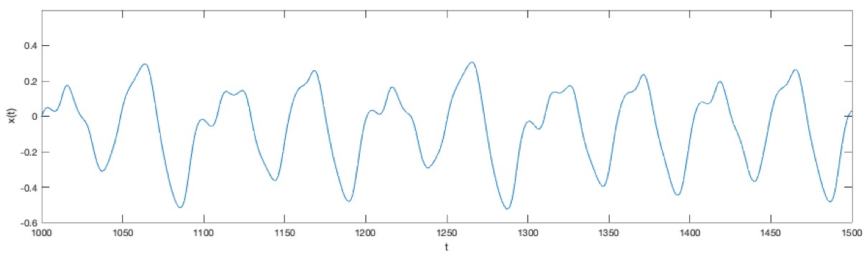


For small s, for small r, what if we just choose? $q(s) = s.$

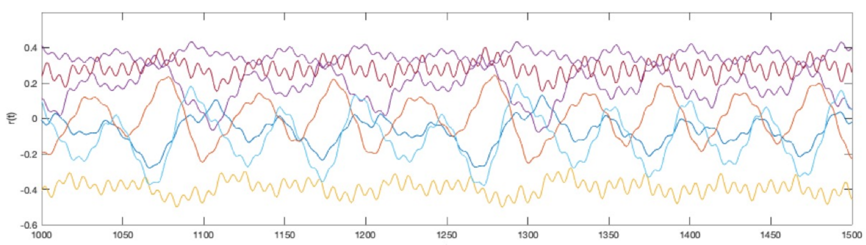


x(t)

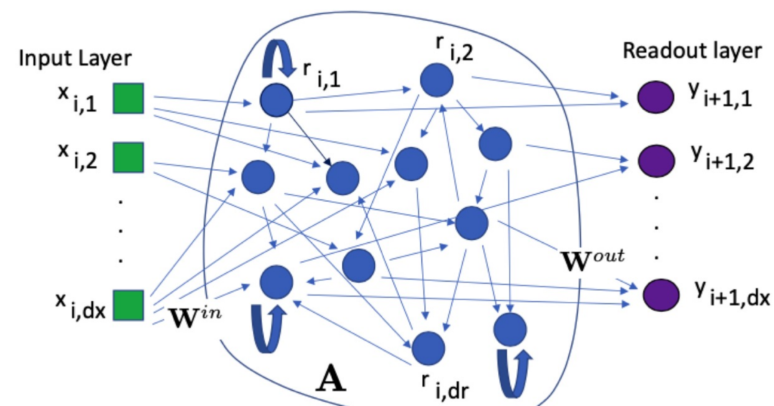
Example RC for Mackey-Glass



x(t) - zoom



r_i(t)



$$\mathbf{r}_{i+1} \quad = \quad (1-\alpha)\mathbf{r}_i + \alpha q(\mathbf{A}\mathbf{r}_i + \mathbf{u}_i + \mathbf{b})$$

$\mathbf{u}_1 = \mathbf{W}^{in}\mathbf{x}_1$, but also we choose, $\mathbf{r}_1 = 0$. Then just iterate– RC is a simple linear iteration with q(s)=s activation

With just linear activation
q(s)=s

Then just iterate
That hidden variable

$$
\begin{aligned}
\mathbf{r}_2 &= \mathbf{A}\mathbf{r}_1 + \mathbf{u}_1 = \mathbf{u}_1 = \mathbf{W}^{in}\mathbf{x}_1 \\
\mathbf{r}_3 &= \mathbf{A}\mathbf{r}_2 + \mathbf{u}_2 \\
&= A\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{in}\mathbf{x}_2 \\
\mathbf{r}_4 &= \mathbf{A}\mathbf{r}_3 + \mathbf{u}_3 \\
&= \mathbf{A}(\mathbf{A}\mathbf{r}_2 + \mathbf{u}_2) + \mathbf{u}_3 \\
&= \mathbf{A}^2\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{A}\mathbf{W}^{in}\mathbf{x}_2 + \mathbf{W}^{in}\mathbf{x}_3 \\
&\vdots \\
\mathbf{r}_{k+1} &= \mathbf{A}\mathbf{r}_k + \mathbf{u}_k \\
&= \mathbf{A}(\mathbf{A}\mathbf{r}_{k-1} + \mathbf{u}_{k-1}) + \mathbf{u}_k \\
&\vdots \\
&= \mathbf{A}^{k-1}\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{A}^{k-2}\mathbf{W}^{in}\mathbf{x}_2 + \ldots + \mathbf{A}\mathbf{W}^{in}\mathbf{x}_{k-1} + \mathbf{W}^{in}\mathbf{x}_k \\
&= \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{u}_{k-j+1} = \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{W}^{in}\mathbf{x}_{k-j+1}, \quad \mathbf{A}^0 = I
\end{aligned}
$$

*A linear RC, linear readout **=** implicit vector autoregressive*

$$
\begin{aligned}
\mathbf{y}_{k+1} &= \mathbf{W}^{out}\mathbf{r}_{k+1} \\
&= \sum_{j=1}^{k} \mathbf{A}^{j-1}\mathbf{W}^{in}\mathbf{x}_{k-j+1} \\
&= \mathbf{W}^{out}\mathbf{A}^{k-1}\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{out}\mathbf{A}^{k-2}\mathbf{W}^{in}\mathbf{x}_2 + \ldots + \mathbf{W}^{out}\mathbf{A}\mathbf{W}^{in}\mathbf{x}_{k-1} + \mathbf{W}^{out}\mathbf{W}^{in}\mathbf{x}_k \\
&= a_k\mathbf{x}_1 + a_{k-1}\mathbf{x}_2 + \ldots + a_2\mathbf{x}_{k-1} + a_1\mathbf{x}_k,
\end{aligned}
$$

with notation,

$$\boxed{a_j = \mathbf{W}^{out}\mathbf{A}^{j-1}\mathbf{W}^{in}, \; j = 1, 2, \ldots, k.}$$

coefficients a_j are $d_x \times d_x$ matrices

Conclude:

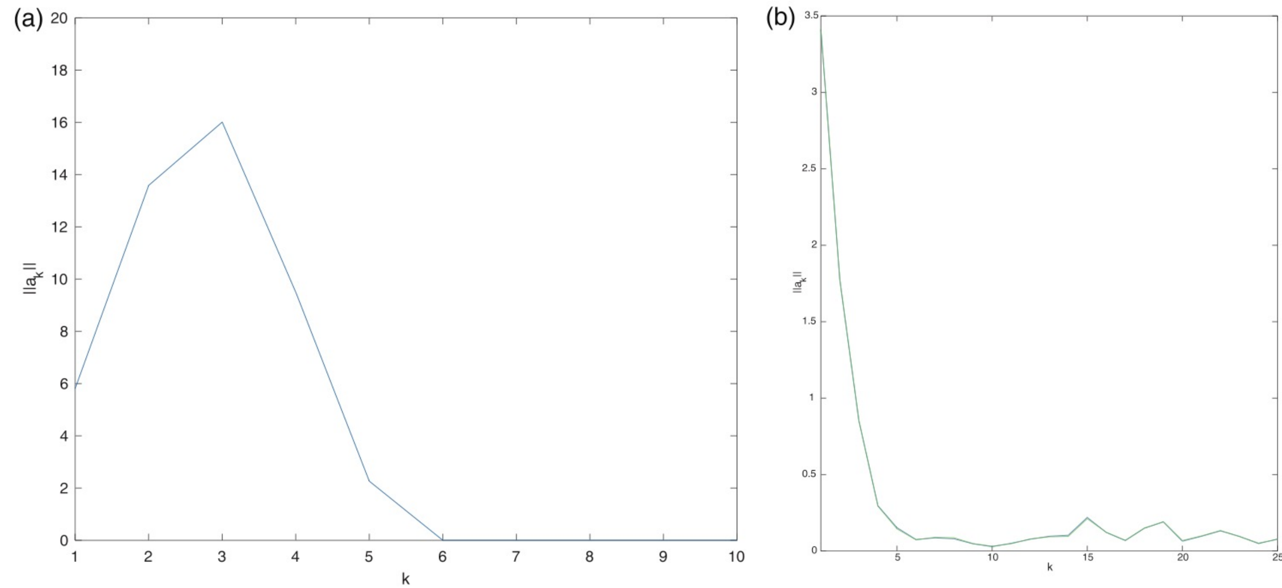
A linear RC - linear readout **=** vector autoregressive of k-delays estimator of a stochastic process – a classical VAR(k) – *a star from Econometrics and stochastic processes*

$$\boxed{\mathbf{y}_{k+1} = c + a_k\mathbf{x}_1 + a_{k-1}\mathbf{x}_2 + \ldots + a_2\mathbf{x}_{k-1} + a_1\mathbf{x}_k + \boldsymbol{\xi}_{k+1}}$$

And this already this works "**pretty well**"

Naturally – Fading memory – time scale re A

$$\begin{aligned}\|a_j\|_\star &= \|\mathbf{W}^{out}\mathbf{A}^{j-1}\mathbf{W}^{in}\|_\star \\ &\leq \|\mathbf{W}^{out}\|_\star\|\mathbf{A}^{j-1}\|_\star\|\mathbf{W}^{in}\|_\star \\ &\leq \|\mathbf{W}^{out}\|_\star\|\mathbf{A}\|_\star^{j-1}\|\mathbf{W}^{in}\|_\star.\end{aligned}$$



The explicit Bridge:

RC= A Lovely VAR(k)

a star from Econometrics

$$\begin{bmatrix} | & | & | & | \\ \mathbf{y}_{k+1} & \mathbf{y}_{k+2} & \dots & \mathbf{y}_N \\ | & | & | & | \end{bmatrix} = [[a_1] \quad [a_2] \quad \dots \quad [a_k]] \begin{bmatrix} | & | & \vdots & | \\ \mathbf{x}_k & \mathbf{x}_{k+1} & \dots & \mathbf{x}_{N-1} \\ | & | & \vdots & | \\ \mathbf{x}_{k-1} & \mathbf{x}_k & \dots & \mathbf{x}_{N-2} \\ | & | & \vdots & | \\ \vdots & \vdots & \vdots & \vdots \\ | & | & \vdots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-k-1} \\ | & | & \vdots & | \end{bmatrix}.$$

$$\mathbf{Y} = \mathbf{a}\mathbb{X} = \mathbf{v}\mathbb{A}\mathbb{X}.$$

$$\mathbb{A} = [\mathbf{W}^{in}|\mathbf{A}\mathbf{W}^{in}|\dots|\mathbf{A}^{k-2}\mathbf{W}^{in}|\mathbf{A}^{k-1}\mathbf{W}^{in}]$$

Randomly stir operator – with delays for memory

$$\mathbf{a}^* = \mathbf{X}\mathbb{X}^T(\mathbb{X}\mathbb{X}^T + \lambda I)^{-1} := \mathbf{X}\mathbb{X}_\lambda^\dagger$$

The directly fitted VAR coefficients

$$\mathbf{W}^{out} := \mathbf{v}^* = \mathbf{a}^*\mathbb{A}_\lambda^\dagger = \mathbf{X}\mathbb{X}_\lambda^\dagger\mathbb{A}_\lambda^\dagger$$

The Relationship between var coefficients and RC readout

EXISTENCE of the representation: **Wold theory** about zero mean covariance stationary vector processes
-there is a VMA - possibly infinite history => for invertible delay processes described by a VAR and approx by a VAR(k).

Theorem 1 **(Wold Theorem,** *A zero mean covariance stationary vector process* $\{\mathbf{x}_t\}$ *admits a representation,*

$$\mathbf{X}_t = C(L)\boldsymbol{\xi}_t + \boldsymbol{\mu}_t,$$

where $C(L) = \sum_{i=0}^{\infty} C_i L^i$ *is a polynomial delay operator polynomial, the* C_i *are the moving average matrices, and* $L^i(\boldsymbol{\xi}_t) = \boldsymbol{\xi}_{t-i}$. *The term* $C(L)\boldsymbol{\xi}$ *is the stochastic part of the decomposition. The* $\boldsymbol{\mu}_t$ *term is the deterministic (perfectly predictable) part as a linear combination of the past values of* $\mathbf{X}_t$. *Furthermore,*

- $\boldsymbol{\mu}_t$ *is a d-dimensional linearly deterministic process.*
- $\boldsymbol{\xi}_t \sim WN(0, \Omega)$ *is white noise.*
- *Coefficient matrices are square summable,*

$$\sum_{i=0}^{\infty} \|C_i\|^2 < \infty.$$

- $C_0 = I$ *is the identity matrix.*
- *For each t,* $\boldsymbol{\mu}_t$ *is called the innovation or the linear forecast errors.*

$$\mathbf{X}_t = C(L)\boldsymbol{\xi}_t \implies B(L)\mathbf{X}_t = \boldsymbol{\xi}_t,$$

Clarifying notation of the delay operator polynomial, with an example, let

$$C(L) = \begin{bmatrix} 1 & 1+L \\ -\frac{1}{2}L & \frac{1}{2} - L \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1}{2} \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -\frac{1}{2} & -1 \end{bmatrix}$$

$$L = C_0 + C_1 L, \text{ and } C_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ if } i > 1;$$

therefore if, for example, $\mathbf{x}_t \in \mathbb{R}^2$,

$$C(L)\mathbf{x}_t = \begin{bmatrix} 1 & 1+L \\ -\frac{1}{2}L & \frac{1}{2} - L \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} = \begin{bmatrix} x_{1,t} + x_{2,t} + x_{2,(t-1)} \\ \frac{1}{2}x_{1,(t-1)} + \frac{1}{2}x_{2,t} - x_{2,(t-1)} \end{bmatrix}$$

Koopman Konnection - The RC can be written as a DMD regression

$$
\begin{bmatrix}
| & | & \vdots & | \\
\mathbf{x}_{k+1} & \mathbf{x}_{k+2} & \dots & \mathbf{x}_N \\
| & | & \vdots & | \\
\mathbf{x}_k & \mathbf{x}_{k+1} & \dots & \mathbf{x}_{N-1} \\
| & | & \vdots & | \\
\vdots & \vdots & \vdots & \vdots \\
| & | & \vdots & | \\
\mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_{N-k} \\
| & | & \vdots & |
\end{bmatrix}
= \mathcal{K}
\begin{bmatrix}
| & | & \vdots & | \\
\mathbf{x}_k & \mathbf{x}_{k+1} & \dots & \mathbf{x}_{N-1} \\
| & | & \vdots & | \\
\mathbf{x}_{k-1} & \mathbf{x}_k & \dots & \mathbf{x}_{N-2} \\
| & | & \vdots & | \\
\vdots & \vdots & \vdots & \vdots \\
| & | & \vdots & | \\
\mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-k-1} \\
| & & &
\end{bmatrix},
$$

$$\mathcal{K} = \underset{K}{\arg\min} \|\mathbb{X}' - K\mathbb{X}\|_F,$$

$$\mathcal{K} = \mathbb{X}'\mathbb{X}^\dagger,$$

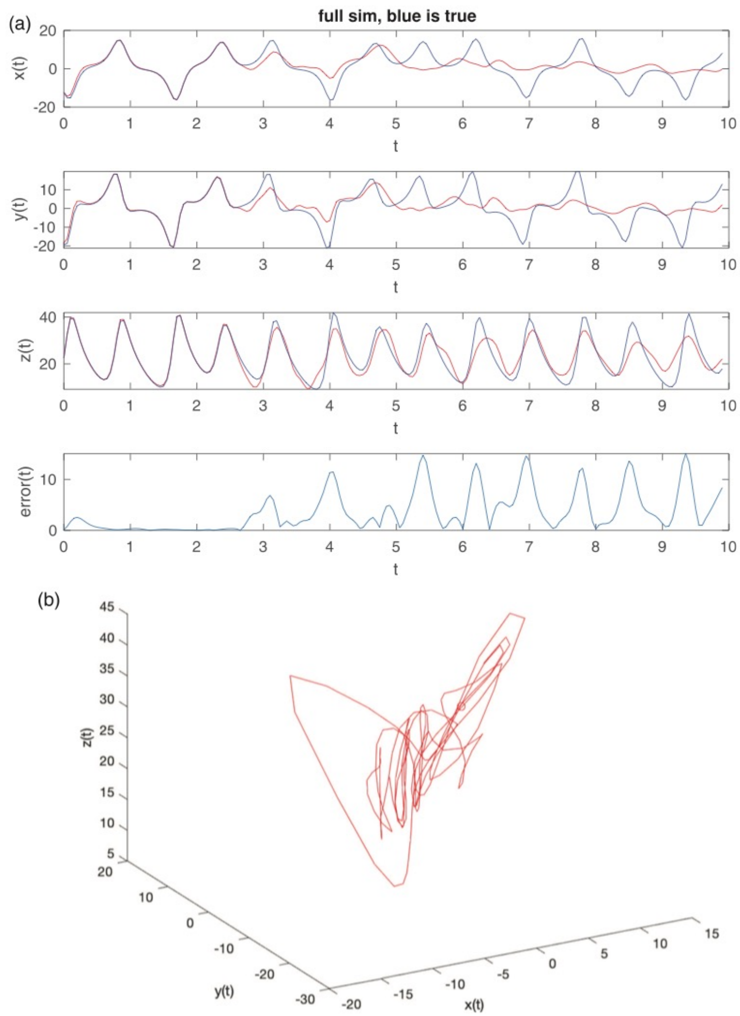
"exact DMD" solution

$$\mathbb{X}' = \mathcal{K}\mathbb{X},$$

vs

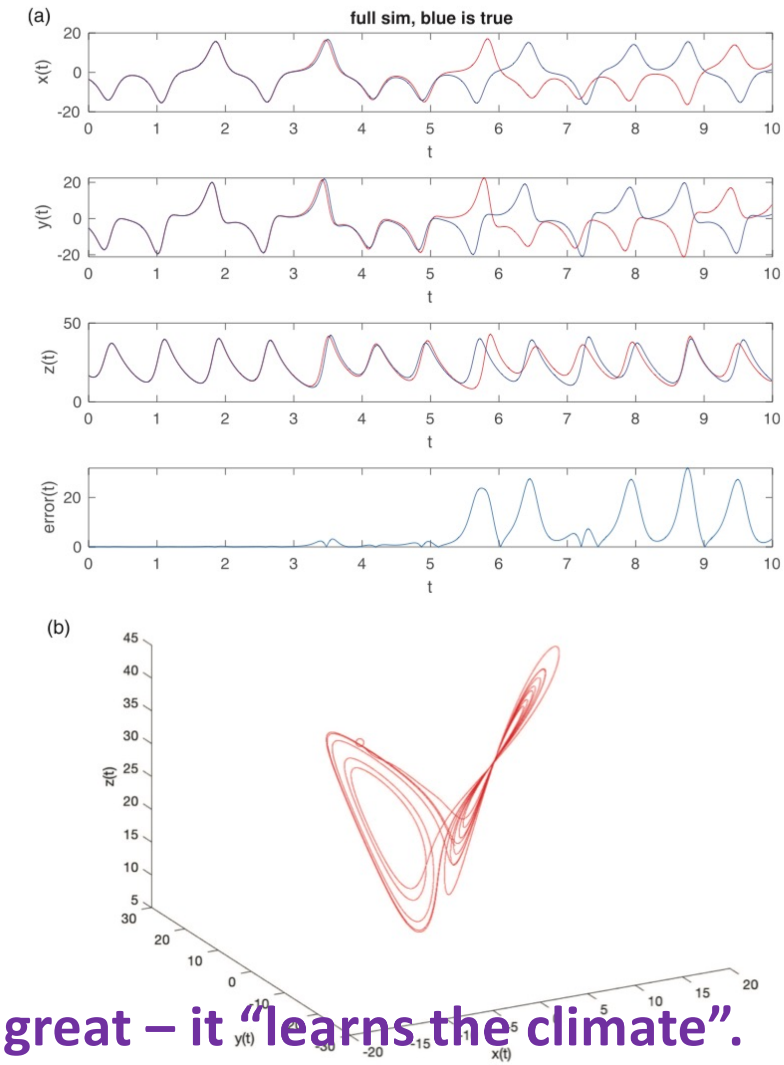
$$
\begin{bmatrix}
| & | & | & | \\
\mathbf{y}_{k+1} & \mathbf{y}_{k+2} & \dots & \mathbf{y}_N \\
| & | & | & |
\end{bmatrix}
= \begin{bmatrix} [a_1] & [a_2] & \dots & [a_k] \end{bmatrix}
\begin{bmatrix}
| & | & \vdots & | \\
\mathbf{x}_k & \mathbf{x}_{k+1} & \dots & \mathbf{x}_{N-1} \\
| & | & \vdots & | \\
\mathbf{x}_{k-1} & \mathbf{x}_k & \dots & \mathbf{x}_{N-2} \\
| & | & \vdots & | \\
\vdots & \vdots & \vdots & \vdots \\
| & | & \vdots & | \\
\mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-k-1} \\
| & | & \vdots & |
\end{bmatrix}
$$

And already this works "**pretty well**"



Fully linear RC, q(x)=x, d_r=1000

Works Great! – linear RC training with nonlinear readout

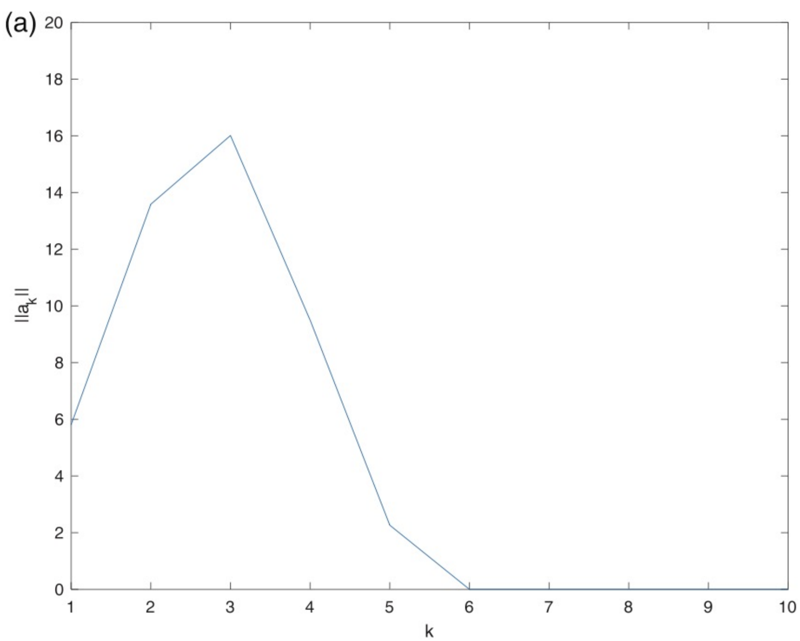


by works great – it "learns the climate".

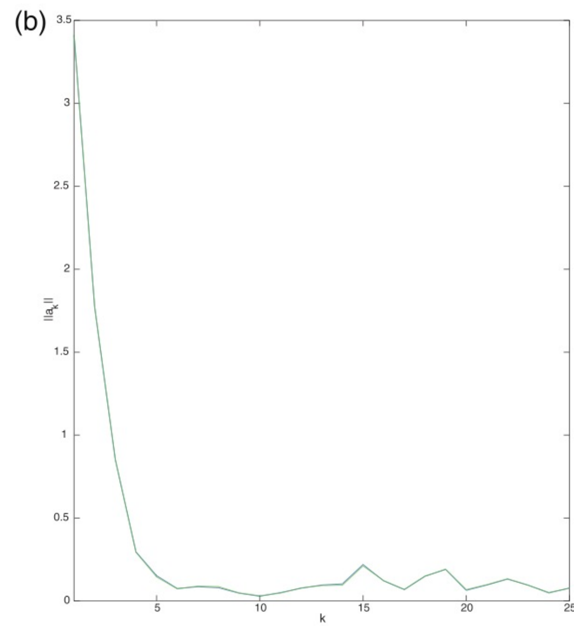
Naturally – **Fading memory** – time scale re A

$$
\begin{aligned}
\|a_j\|_\star &= \|\mathbf{W}^{out}\mathbf{A}^{j-1}\mathbf{W}^{in}\|_\star \\
&\le \|\mathbf{W}^{out}\|_\star \|\mathbf{A}^{j-1}\|_\star \|\mathbf{W}^{in}\|_\star \\
&\le \|\mathbf{W}^{out}\|_\star \|\mathbf{A}\|_\star^{j-1} \|\mathbf{W}^{in}\|_\star .
\end{aligned}
$$

Mackey-Glass



Lorenz63



In practice – train the linear RC to polynomial readout and not just to hidden variable **r**

$$\mathbf{R}_1 = \left[\mathbf{r}_k \quad |\mathbf{r}_{k+1} \quad | \cdots \quad |\mathbf{r}_N\right],$$

Hadamard product

$$\mathbf{R}_2 = \left[\mathbf{r}_k \circ \mathbf{r}_k \quad |\mathbf{r}_{k+1} \circ \mathbf{r}_{k+1} \quad | \cdots \quad |\mathbf{r}_N \circ \mathbf{r}_N\right]$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}. \qquad \mathbf{W}^{out} = \begin{bmatrix} \mathbf{W}_1^{out} \\ \mathbf{W}_2^{out} \end{bmatrix} \qquad \mathbf{W}^{out} := \mathbf{X}\mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \lambda\mathbf{I})^{-1}$$

Turns out this yields not a VAR but an **NVAR** – works much better! – Just like before – iterate……

$$B\mathbf{w} \circ B\mathbf{w} = (w_1\mathbf{b}_1 + w_2\mathbf{b}_2 + \cdots + w_n\mathbf{b}_n) \circ (w_1\mathbf{b}_1 + w_2\mathbf{b}_2 + \cdots + w_n\mathbf{b}_n)$$

"key trick"

$$= [\mathbf{b}_1 \circ \mathbf{b}_1 | \mathbf{b}_1 \circ \mathbf{b}_2 | \cdots | \mathbf{b}_n \circ \mathbf{b}_n] \begin{bmatrix} w_1^2 \\ w_1 w_2 \\ \vdots \\ w_1 w_n \\ w_2 w_1 \\ w_2^2 \\ \vdots \\ w_n^2 \end{bmatrix} := P_2(B, B)p_2(\mathbf{w}, \mathbf{w}).$$

$$P_2 : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n^2},$$

$m \times n^2$ matrix of Hadamard products

$$\begin{aligned}
\mathbf{r}_2 \circ \mathbf{r}_2 &= (\mathbf{W}^{in}\mathbf{x}_1) \circ (\mathbf{W}^{in}\mathbf{x}_1) \\
&= P_2(\mathbf{W}^{in}, \mathbf{W}^{in})p_2(\mathbf{x}_1), \\
\mathbf{r}_3 \circ \mathbf{r}_3 &= (A\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{in}\mathbf{x}_2) \circ (A\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{in}\mathbf{x}_2) \\
&= P_2(A\mathbf{W}^{in}, A\mathbf{W}^{in})p_2(\mathbf{x}_1, \mathbf{x}_1) + P_2(A\mathbf{W}^{in}, \mathbf{W}^{in})p_2(\mathbf{x}_1, \mathbf{x}_2) \\
&\quad + P_2(\mathbf{W}^{in}, A\mathbf{W}^{in})p_2(\mathbf{x}_2, \mathbf{x}_1) + P_2(\mathbf{W}^{in}, \mathbf{W}^{in})p_2(\mathbf{x}_2, \mathbf{x}_2), \\
&\vdots
\end{aligned}$$

$p_2(\mathbf{v}, \mathbf{w}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n^2}$, $n^2 \times 1$ vector of quadratic monomials

$$(\mathbf{v}, \mathbf{w}) \mapsto [v_1 w_1 | v_1 w_2 | \cdots | v_1 w_n | v_2 w_1 | v_2 w_2 | \cdots | v_n w_n]^T,$$

$$
\begin{aligned}
\mathbf{r}_2 \circ \mathbf{r}_2 &= (\mathbf{W}^{in}\mathbf{x}_1) \circ (\mathbf{W}^{in}\mathbf{x}_1) \\
&= P_2(\mathbf{W}^{in}, \mathbf{W}^{in}) p_2(\mathbf{x}_1), \\
\mathbf{r}_3 \circ \mathbf{r}_3 &= (A\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{in}\mathbf{x}_2) \circ (A\mathbf{W}^{in}\mathbf{x}_1 + \mathbf{W}^{in}\mathbf{x}_2) \\
&= (A\mathbf{W}^{in}\mathbf{x}_1) \circ (A\mathbf{W}^{in}\mathbf{x}_1) + (A\mathbf{W}^{in}\mathbf{x}_1) \circ (\mathbf{W}^{in}\mathbf{x}_2) \\
&\quad + (\mathbf{W}^{in}\mathbf{x}_2) \circ (A\mathbf{W}^{in}\mathbf{x}_1) + (\mathbf{W}^{in}\mathbf{x}_2) \circ (\mathbf{W}^{in}\mathbf{x}_2) \\
&= P_2(A\mathbf{W}^{in}, A\mathbf{W}^{in}) p_2(\mathbf{x}_1, \mathbf{x}_1) + P_2(A\mathbf{W}^{in}, \mathbf{W}^{in}) p_2(\mathbf{x}_1, \mathbf{x}_2) \\
&\quad + P_2(\mathbf{W}^{in}, A\mathbf{W}^{in}) p_2(\mathbf{x}_2, \mathbf{x}_1) + P_2(\mathbf{W}^{in}, \mathbf{W}^{in}) p_2(\mathbf{x}_2, \mathbf{x}_2), \\
&\vdots \\
\mathbf{r}_{k+1} \circ \mathbf{r}_{k+1} &= \sum_{i=1}^{k} (A^{i-1}\mathbf{W}^{in}\mathbf{x}_{k+1-i}) \circ \left(\sum_{j=1}^{k} A^{j-1}\mathbf{W}^{in}\mathbf{x}_{k+1-j} \right) \\
&= \sum_{i,j=1}^{k} P_2(A^{i-1}\mathbf{W}^{in}, A^{j-1}\mathbf{W}^{in}) p_2(\mathbf{x}_{k+1-i}, \mathbf{x}_{k+1-j}) \\
&:= \mathbb{A}_2[\mathbb{X}_2]_k.
\end{aligned}
$$

The iteration thing again,
Now gives monomials

$$
\begin{aligned}
\mathbb{A}_2 = [&P_2(\mathbf{W}^{in}, \mathbf{W}^{in}) | P_2(A\mathbf{W}^{in}, \mathbf{W}^{in}) | P_2(A^2\mathbf{W}^{in}, \mathbf{W}^{in}) | \cdots \\
&\cdots | P_2(A^{k-1}\mathbf{W}^{in}, \mathbf{W}^{in}) | P_2(\mathbf{W}^{in}, A\mathbf{W}^{in}) | P_2(A\mathbf{W}^{in}, A\mathbf{W}^{in}) \\
&\times | P_2(A^2\mathbf{W}^{in}, A\mathbf{W}^{in}) | \cdots \\
&\cdots | P_2(A^{k-2}\mathbf{W}^{in}, A^{k-1}\mathbf{W}^{in}) | P_2(A^{k-1}\mathbf{W}^{in}, A^{k-1}\mathbf{W}^{in})]
\end{aligned}
$$

is a $d_r \times kd_x^2$ matrix.

Now explicit connection between NVAR=linear RC w' nonlinear readout

Stack the monomials

$$\mathbb{X}_1 = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{x}_k & \mathbf{x}_{k+1} & \cdots & \mathbf{x}_{N-1} \\ | & | & \vdots & | \\ \mathbf{x}_{k-1} & \mathbf{x}_k & \cdots & \mathbf{x}_{N-2} \\ | & | & \vdots & | \\ \vdots & \vdots & \vdots & \vdots \\ | & | & \vdots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{N-k} \\ | & | & \vdots & | \end{bmatrix}, \mathbb{X}_2 = \begin{bmatrix} | & | & \vdots & | \\ p_2(\mathbf{x}_k, \mathbf{x}_k) & p_2(\mathbf{x}_{k+1}, \mathbf{x}_{k+1}) & \cdots & p_2(\mathbf{x}_{N-1}, \mathbf{x}_{N-1}) \\ | & | & \vdots & | \\ p_2(\mathbf{x}_{k-1}, \mathbf{x}_k) & p_2(\mathbf{x}_k, \mathbf{x}_{k+1}) & \cdots & p_2(\mathbf{x}_{N-2}, \mathbf{x}_{N-1}) \\ | & | & \vdots & | \\ \vdots & \vdots & \vdots & \vdots \\ | & | & \vdots & | \\ p_2(\mathbf{x}_1, \mathbf{x}_k) & p_2(\mathbf{x}_2, \mathbf{x}_{k+1}) & \cdots & p_2(\mathbf{x}_{N-k-1}, \mathbf{x}_{N-1}) \\ | & | & \vdots & | \\ p_2(\mathbf{x}_k, \mathbf{x}_{k-1}) & p_2(\mathbf{x}_{k+1}, \mathbf{x}_k) & \cdots & p_2(\mathbf{x}_{N-1}, \mathbf{x}_{N-2}) \\ | & | & \vdots & | \\ p_2(\mathbf{x}_{k-1}, \mathbf{x}_{k-1}) & p_2(\mathbf{x}_{k+1}, \mathbf{x}_{k-1}) & \cdots & p_2(\mathbf{x}_{N-2}, \mathbf{x}_{N-2}) \\ | & | & \vdots & | \\ \vdots & \vdots & \vdots & \vdots \\ | & | & \vdots & | \\ p_2(\mathbf{x}_1, \mathbf{x}_1) & p_2(\mathbf{x}_2, \mathbf{x}_2) & \cdots & p_2(\mathbf{x}_{N-k}, \mathbf{x}_{N-k}) \\ | & | & \vdots & | \end{bmatrix}.$$

$$p_2(\mathbf{v}, \mathbf{w}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n^2},$$

$$(\mathbf{v}, \mathbf{w}) \mapsto [v_1 w_1 | v_1 w_2 | \cdots | v_1 w_n | v_2 w_1 | v_2 w_2 | \cdots | v_n w_n]^T,$$

Then again we get a version of

$$\mathbb{X} = \begin{bmatrix} \mathbb{X}_1 \\ \mathbb{X}_2 \end{bmatrix} \qquad \mathbf{Y} = \mathbf{a}\mathbb{X}$$

said as NVAR

$$\begin{aligned} \mathbf{y}_{\ell+1} &= a_\ell \mathbf{x}_1 + a_{\ell-1}\mathbf{x}_2 + \cdots + a_2 \mathbf{x}_{\ell-1} + a_1 \mathbf{x}_\ell + a_{2,(\ell,\ell)} p_2(\mathbf{x}_1, \mathbf{x}_1) \\ &+ a_{2,(\ell-1,\ell)} p_2(\mathbf{x}_2, \mathbf{x}_1) + \cdots + a_{2,(1,1)} p_2(\mathbf{x}_\ell, \mathbf{x}_\ell), \end{aligned}$$

Specifically - NVAR coeff relate to RC parameters

$$\underline{a_j = \mathbf{W}_1^{out} \mathbf{A}^{j-1} \mathbf{W}^{in}}, j = 1, 2, \ldots, \ell, \qquad \underline{a_{2,(i,j)} = \mathbf{W}_2^{out} P_2(A^{i-1}\mathbf{W}^{in}, A^{j-1}\mathbf{W}^{in})}, i, j = 1, \ldots, \ell.$$

nature COMMUNICATIONS

ARTICLE

https://doi.org/10.1038/s41467-021-25801-2 OPEN

Next generation reservoir computing

Daniel J. Gauthier[1,2], Erik Bollt[3,4], Aaron Griffith[1] & Wendson A. S. Barbosa[1]

Linear RC with nonlinear readout **=** implicit NVAR ===> NG-RC

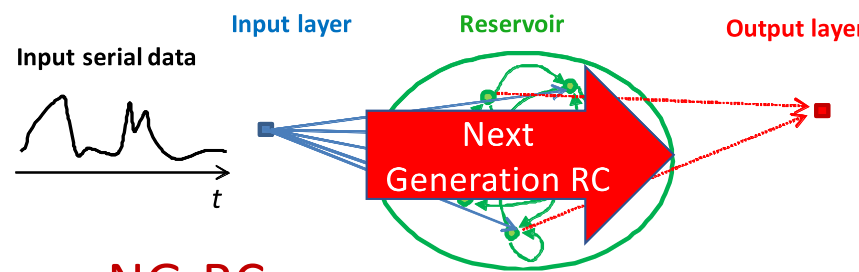
An implicit RC means we can skip it - NG-RC more efficient

– less data hungry – skips the middle man –

Less parameters and hyperparameters to worry about.

Leads to a more general concept

NG-RC – Next Generation RC.



Facts: a good NVAR has an implicit RC

a good RC implies a good NVAR – collect as a NG-RC

Choose Linear Features vector

$$\mathbb{O}_{lin} = [x(t), x(t-dt), y(t), y(t-dt), z(t), z(t-dt)]$$

An efficient notation collects all unique terms of high order monomials

$$\mathbb{O}_{nonlin}(t) = \mathbb{O}_{lin} \lceil \otimes \rceil \mathbb{O}_{lin} \text{ term of quadratics monomials}$$

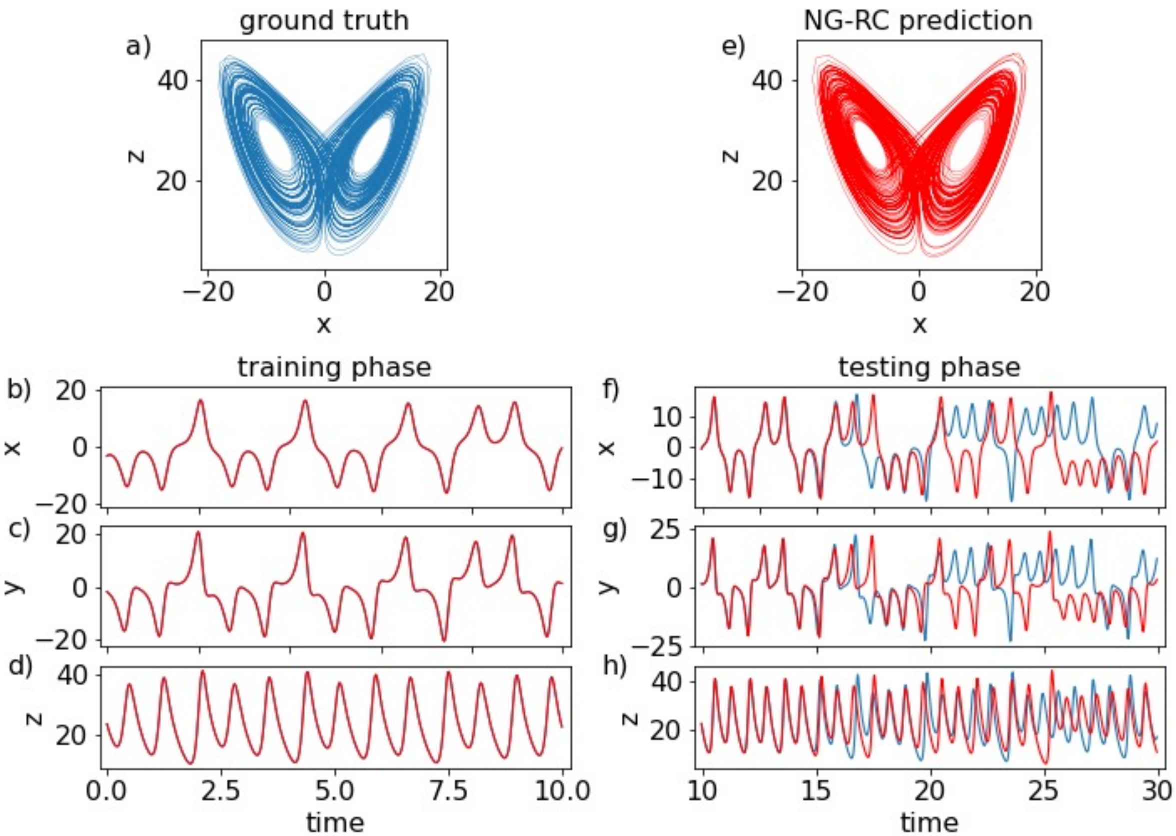
$$\lceil \otimes \rceil_k \mathbb{O}_{lin} := \lceil \mathbb{O}_{lin} \otimes \ldots \otimes \mathbb{O}_{lin} \rceil, \; k\text{-times repeating the } \otimes$$

$$\mathbb{O}_{total}(t) = [\mathbb{O}_{lin}; \lceil \otimes \rceil_2 \mathbb{O}_{lin}; \ldots; \lceil \otimes \rceil_p \mathbb{O}_{lin}](t)$$

Conclude: Works really really well – and drastically MUCH less data hungry

-linear RC with nonlinear readout = implicit NVAR **AND this leads to** NG-RC

-VAR vs VMA which follows classic representation theorem by WOLD thm - also relates to DMD-Koopman



Notable from Literature

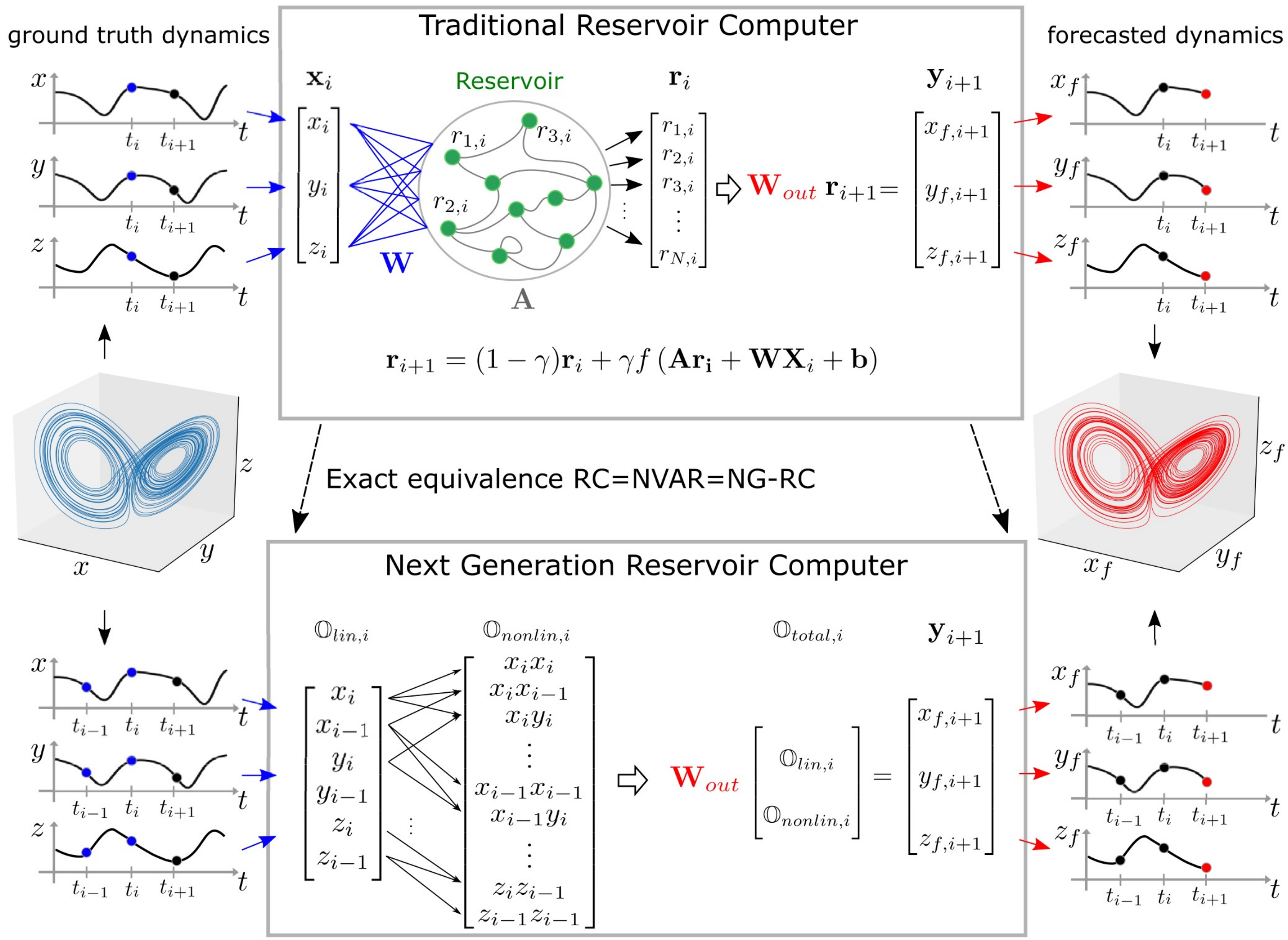
-Billings and NARX

-Gonan-Ortega universal approx thm.

Even for linear RC with nonlinear readout

NG-RC is 1. simple – 2. MUCH less data hungry – 3. few parameters – 4. flexible feature

A traditional RC is implicit in an NG-RC



Ortega, and also Bollt, move the nonlinearity from the activation function instead to a feature vector of inner state.

A linear reservoir with nonlinear output is equivalently powerful as a universal approximator with similar performance as a Standard RC - but with reliability and simplicity advantages.

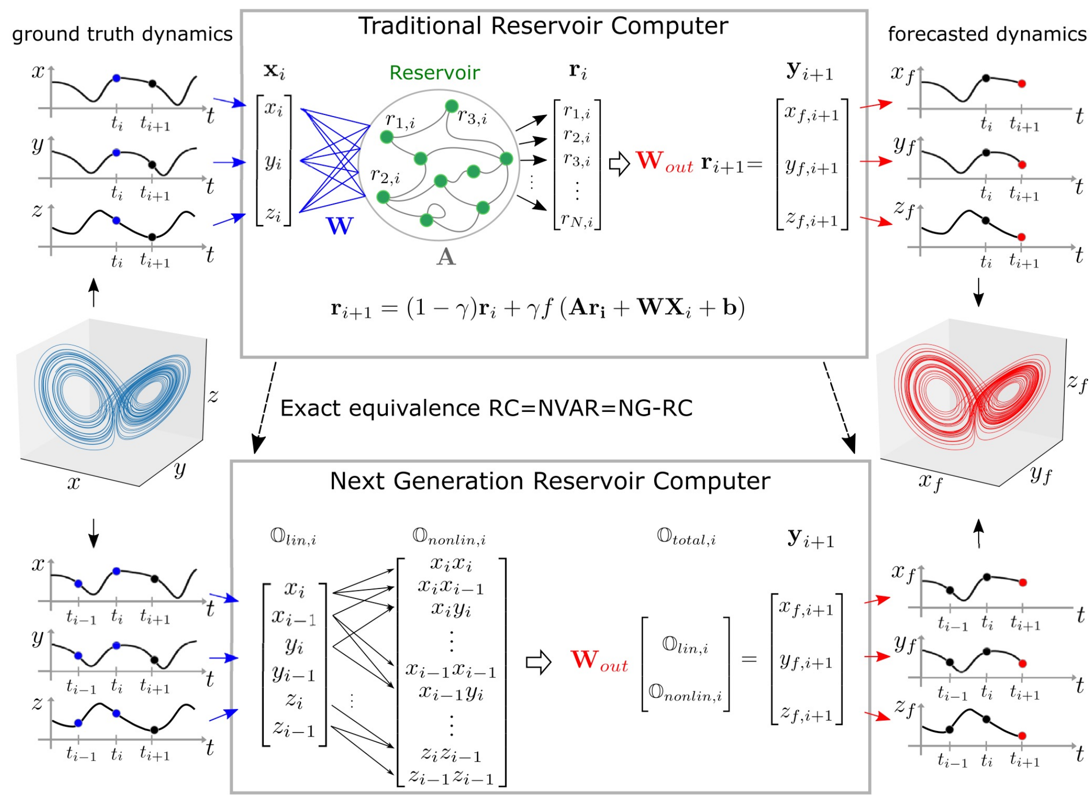
$$\mathbf{Y}_{i+1} = \mathbf{W}_{\text{out}} \mathbb{O}_{\text{total},i+1},$$

$$\mathbf{W}_{\text{out}} = \mathbf{Y}_d \mathbb{O}_{\text{total}}{}^T \left(\mathbb{O}_{\text{total}} \mathbb{O}_{\text{total}}{}^T + \alpha \mathbf{I}\right)^{-1},$$

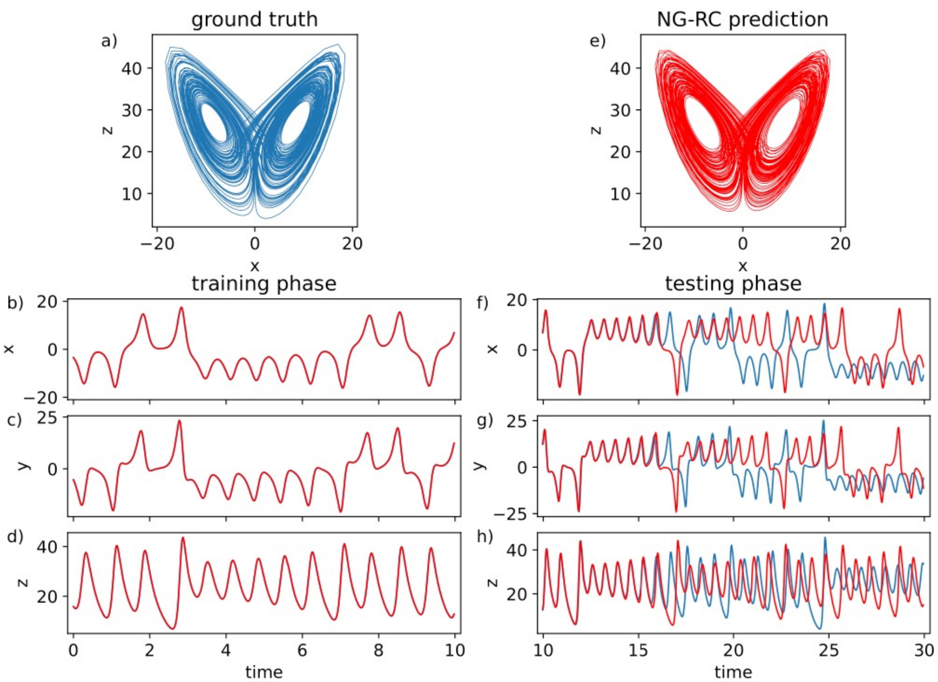
Hadamard product $\mathbf{r} \odot \mathbf{r} = \left[r_1^2, r_2^2, \ldots, r_N^2\right]^T$

$$\mathbb{O}_{\text{total}} = \mathbf{r} \oplus (\mathbf{r} \odot \mathbf{r}) = \left[r_1, r_2, \ldots, r_N, r_1^2, r_2^2, \ldots, r_N^2\right]^T,$$

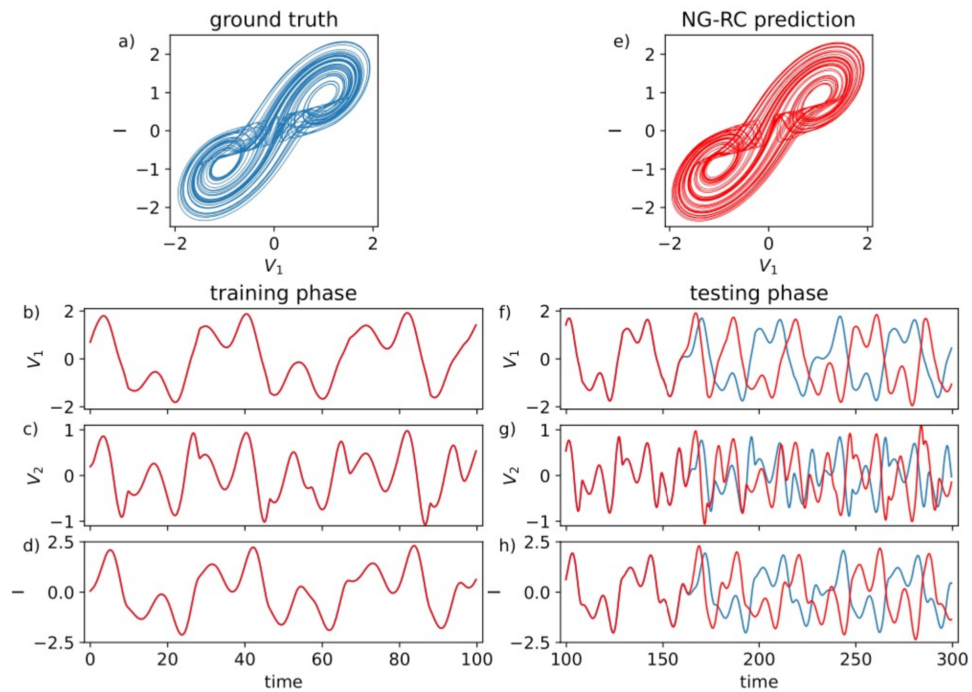
$\oplus$ represents the vector concatenation operation.



NG-RC works very well, with very few points, almost no tunable parameters



Forecasting a dynamical system using the NG-RC.
Lorenz63 strange attractors.



Forecasting the double-scroll system using the NG-RC

Another fun task – *look Ma! – no z*!

Inference using an NG-RC. a–c Lorenz63 variables during the training phase (blue) and prediction (c, red)