# Reservoir computing

Karol Bednarz

September 14, 2025

## Contents


**1 Introduction** **1**
- 1.1 Principles of reservoir computing . . . 1
- 1.2 Echo state property . . . 2

**2 Learning algorithm** **2**


# 1 Introduction

## 1.1 Principles of reservoir computing

The state of reservoir dynamics can be expressed as:

$$h_{\mathrm{t}} = f\big((1-k)\cdot u_{\mathrm{t}}\cdot W_{\mathrm{in}} + k\cdot h_{\mathrm{t-1}}\cdot W_{\mathrm{h}} + y_{\mathrm{t-1}}\cdot W_{\mathrm{1b}} + b\big) \tag{1}$$

Where:

$h_{\mathrm{t-1}}$ – are the reservoir state respectively, from the previous time step,
$u_{\mathrm{t}}$ – is the observed data at time step $t$,
$y_{\mathrm{t-1}}$ – is the the predicted output state $t-1$,
$W_{\mathrm{in}} \in \mathbb{R}^{N_{\mathrm{u}}\times N_{\mathrm{h}}}$ – is the input weight matrix,
$W_{\mathrm{h}} \in \mathbb{R}^{N_{\mathrm{h}}\times N_{\mathrm{h}}}$ – is the internal weight matrix,
$W_{\mathrm{1b}} \in \mathbb{R}^{N_{\mathrm{h}}\times N_{\mathrm{y}}}$ – is the output fedback weight matrix,
$b \in \mathbb{R}^{N_{\mathrm{h}}}$ – is the bias vector.
$f$ – is the activation function, typically tanh or sigmoid,
$k$ – is the leaking rate, typically $k \in [0.1, 0.3]$.

With the computed reservoir dynamics, the output can be then obtained by:

$$y_{\mathrm{t}} = h_{\mathrm{t}}\cdot W_{\mathrm{out}} \tag{2}$$

Where:

$W_{\mathrm{out}} \in \mathbb{R}^{N_{\mathrm{h}}\times N_{\mathrm{y}}}$ – is the output weight matrix.

### 1.2 Echo state property

Any system that changes in a nonlinear way can work as the reservoir. However, starting a nonlinear system with random connection strengths creates problems. The reservoir is a system that feeds its outputs back into itself. This can make it unstable if the connection strengths aren't set up correctly. For example, if the internal connections are too strong, the system might get stuck giving the same output regardless of what input it receives. The random connection strengths must be chosen so the system doesn't grow out of control. For the system to work well, it must follow something called the "echo state property." This rule ensures that the reservoir's behavior eventually depends on the input signal rather than just its starting conditions. To meet this requirement, the internal connection matrix $W_{\text{h}}$ is first set up using random values between -1 and 1. This matrix is then adjusted one time according to the echo state property rule:

$$W'_{\text{h}} = \alpha \odot W_{\text{h}} \tag{3}$$

$$W^{\dagger}_{\text{h}} = \frac{\rho W_{\text{h}}}{|\lambda_{\max}(W_{\text{h}})|} \tag{4}$$

Where:

$\rho \in (0, 1)$ – is the spectral radius, typically $\rho \in [0.9, 1]$
$\lambda_{\max}(W_{\text{h}})$ – is the largest eigenvalue of $W_{\text{h}}$.
$\alpha \in (0, 1)$ – is the sparsity coefficient, typically $\alpha \in [0.1, 0.3]$.

The spectral radius is a parameter that determines the amount of nonlinear interaction of input components through time.

Due to the recursive nature of the reservoir layer, such dynamics reflect trajectories of the past historical inputthe short-term memory (known as the fading memory). As another critical property for computing the RC principle, short-term memory can be quantitative measured by the coefficient of memory capacity

$$MC = \sum_{\text{k}=1}^{\infty} MC_{\text{k}} = \sum_{\text{k}=1}^{\infty} d^2(u_{\text{t}-\text{k}}, y_{\text{t}}) = \sum_{\text{k}=1}^{\infty} \frac{\text{cov}^2(u_{\text{t}-\text{k}}, y_{\text{t}})}{\sigma^2(u_{\text{t}})\sigma^2(y_{\text{t}})} \tag{5}$$

Where:

$d^2(u_{\text{t}-\text{k}}, y_{\text{t}})$ – is the square of the correlation coefficient between the output $y_{\text{t}}$ and the input $u_{\text{t}-\text{k}}$ with a delay of $k$ time steps,

According to the Lyapunov stability analysis, a large memory capacity is needed to compute the RC principle, which can be achieved at the asymptotically stable region.

## 2 Learning algorithm

The training of the reservoir computing model involves adjusting only the output weights $W_{\text{out}}$. The input weights $W_{\text{in}}$, internal weights $W_{\text{h}}$, and feedback weights $W_{\text{1b}}$ are typically initialized randomly and remain fixed during training. The training process can be summarized in the following steps:

$$Y = H \cdot W_{\text{out}} \tag{6}$$

Where:

$Y \in \mathbb{R}^{T \times N_{\text{y}}}$ – is the matrix of target outputs for all time steps,

$H \in \mathbb{R}^{T \times N_\mathrm{h}}$ – is the matrix of reservoir states for all time steps,
$T$ – is the total number of time steps.

In general, the $W_\mathrm{out}$ can be directly obtained by calculating the Moore-Penrose pseudoinverse of the reservoir states matrix H with respect to the target outputs matrix Y:

$$W_\mathrm{out} = Y \cdot H^\dagger \cdot (H \cdot H^\dagger + \eta I)^{-1} \tag{7}$$

Where:

$H^\dagger$ – is the Moore-Penrose pseudoinverse of matrix H,
$\eta$ – is the regularization parameter, typically $\eta \in [10^{-6}, 10^{-2}]$,
$I$ – is the identity matrix of size $N_\mathrm{h} \times N_\mathrm{h}$.