Delay embedding of Rossler system

Name	:	Rishi Vora
Reg. No.	:	MS21113
Code	:	https://github.com/Vortriz/IDC402

I. Introduction

Delay coordinate embedding is a technique used in nonlinear dynamics to reconstruct the state space of a system when only a single time series of measurements is available. Often, in real-world systems or experiments, it's not possible to measure all the variables that define the system's state. Instead, we might only have data for one observable quantity over time, which might even be a function of the original state variables rather than one of the variables itself.

It might seem that reducing a complex, high-dimensional system to a single time series would cause a loss of crucial information about the system's dynamics. However, delay embedding allows us to recover many dynamical properties of the original system from this single time series. The core idea stems from the deterministic nature of these systems, where the past influences the future.

II. The Method

The procedure involves creating new, multi-dimensional state vectors by combining time-delayed values from the original time series $w(t)$. A vector v_n in this new, reconstructed space is formed as follows:

$$\boldsymbol{v}_n = (w[n], w[n+\tau], \ldots, w[n+(d-1)\tau])$$

Here:

- $w[n]$ is the measurement at time step n.
- τ is the delay time (an integer representing time steps).
- d is the embedding dimension (the dimension of the reconstructed space).

This process essentially uses the time series' history to create new dimensions. The collection of all such vectors $\{v_n\}$ forms the reconstructed state space, often denoted as $\mathbb{R}$.

III. Why it works

The delay embedding relies on the fundamental properties of deterministic dynamical systems and is mathematically grounded by embedding theorems, most famously Takens' Theorem.

In deterministic systems, the current state uniquely determines the future state. This implies that the history of an observable $w(t)$ contains implicit information about the unobserved variables it interacts with. The values of the hidden variables influence the trajectory of the observed variable $w(t)$. By including time-delayed values ($w[n+\tau]$, $w[n+2\tau]$, etc.) in the reconstructed vector, we are essentially capturing snapshots of this influence over time, allowing us to infer the state of the hidden variables.

Mathematically, the process defines a transformation (a map) F that takes a state $x(t)$ from the original system's attractor A (the set of states the system settles into) and maps it to a vector $v(t)$ in the reconstructed d-dimensional space $\mathbb{R}$. This map can be written as:

$$\begin{aligned} v(t) = \big(h(x(t)), h(\Phi^{\tau_d}(x(t))), \ldots, \\ h\big(\Phi^{(d-1)\tau d}(x(t))\big)\big) = F(x(t)) \end{aligned}$$

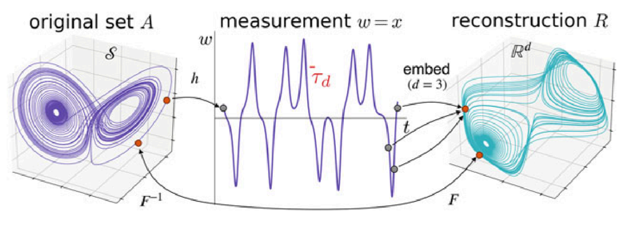

Figure 1: Delay coordinates embedding of a time series generated by the Lorenz-63 system [1]

where h is the measurement function and Φ^τ represents the evolution of the original system over time τ.

Takens' Theorem (and related results) guarantees that if the original system's attractor A has a fractal dimension Δ, then for almost all choices of smooth measurement functions h and delay times τ_d, the map F is an embedding of the original attractor A into the reconstructed space $\mathbb{R}$, provided the embedding dimension d is sufficiently large (specifically, $d > 2\Delta(A)$is a sufficient condition for preserving dynamics, while $d > \Delta(A)$ can be enough for just calculating dimensions).

An embedding in this context means the map F is a diffeomorphism between the original attractor A and the reconstructed set $\mathbb{R} = F(A)$. A diffeomorphism is a smooth, invertible map with a smooth inverse and diffeomorphisms preserve topological properties. This means:

- The geometric structure of the attractor is preserved in a topological sense (though it might look stretched or twisted). Points close together on the original attractor remain close in the reconstruction, and vice-versa. Trajectories that don't cross in the original space won't cross in a proper reconstruction.
- Dynamical invariants like fractal dimensions (e.g., correlation dimension, capacity dimension) and Lyapunov exponents are the same for both the original attractor A and the reconstructed set $\mathbb{R}$.

the theorem holds for "almost all" h and τ_d means that while specific choices might fail (e.g., observing a variable unaffected by others, or choosing a delay equal to a system period, or using a function that collapses symmetries), such cases are exceptional. A generic (typical or randomly chosen) measurement function and delay time will work.

A sufficiently large embedding dimension d is needed to unfold the dynamics properly in the reconstructed space, ensuring that trajectories don't falsely intersect due to projection into too few dimensions. The condition $d > 2\Delta(A)$ guarantees this unfolding.

IV. Optimal time delay and embedding dimension

A. *Choosing the Time Delay (τ):*

The goal is to choose a τ such that the delayed coordinate $w[n+\tau]$ adds the most new information relative to $w[n]$, meaning they should have minimal similarity.

A common approach is to calculate the self-mutual information (SMI) between the time series w and its τ-delayed version w_τ. The optimal τ is often chosen as the value where the SMI reaches its first minimum. This indicates the point where, on average, $w[n]$ and $w[n+\tau]$ share the least information.

Alternatively, one might use the first zero or first minimum of the autocorrelation function of the time series, though this is often less reliable, especially if the autocorrelation decays slowly.

B. *Choosing the Embedding Dimension (d):*

The goal is to find the smallest dimension d that is large enough to fully unfold the dynamics of the attractor in the reconstructed space, eliminating false crossings of trajectories that occur due to projection into too low a dimension.

A widely used method is the False Nearest Neighbors (FNN) algorithm (or variations like AFNN). This method checks how the distance between nearest neighbors in the reconstructed space changes as the dimension is increased from d to $d+1$.

If two points are close in dimension d simply because the attractor projection is squashed, they will often jump far apart when the next dimension $(d+1)$ is added. These are false neighbors.

The optimal embedding dimension d is typically chosen as the dimension where the percentage of false nearest neighbors drops significantly (e.g., close to zero) or where a related statistic (E_d) stops changing substantially and saturates near 1.

V. Rossler attractor

The equations defining Rossler system [2] are:

$$\dot{x} = -y - z$$
$$\dot{y} = x + ay$$
$$\dot{z} = b + z(x - c)$$

Properties of this system with $a = 0.1$, $b = 0.1$, and $c = 14$ have been studied extensively.

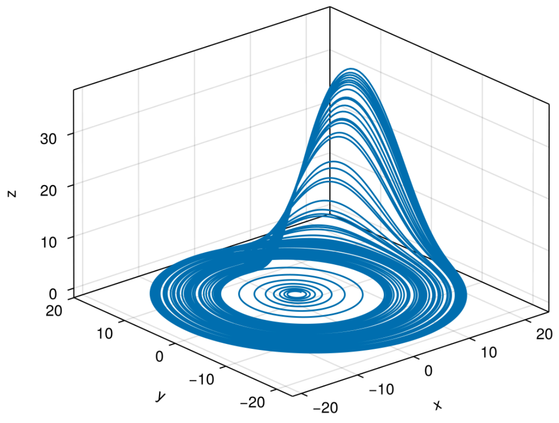

Figure 2: Rossler attractor with $a = 0.1$, $b = 0.1$, and $c = 14$

For embedding, we use:

$$h(x) = x + \text{random noise}$$
$$\tau = 100$$
$$d = 3$$

These values of τ and d were obtained with DynamicalSystems.jl [1]. While we earlier stated that $d > 2\Delta(A)$ is sufficient for embedding, it is not necessary. For Rossler system $d = 3$ suffices. This particular τ was obtained at time step $\Delta t = 0.01$.

VI. Results

We measured Lyapunov exponent of the original system and the delay embedded one, as it is invariant under diffeomorphism.

- Lyapunov (original) = 0.048657
- Lyapunov (delay embedded) = 0.048981

So, we were able to get back our original system with error of just 0.666%

VII. Credits

The simulations were doing using Julia language [3] with the help of following packages:

- OrdinaryDiffEq [4]
- CairoMakie [5]
- DataFrames [6]
- DataFramesMeta
- IntervalArithmetic [7]
- CoordinateTransformations
- LinearAlgebra

Bibliography

[1] G. Datseris and U. Parlitz, *Nonlinear Dynamics*. Springer International Publishing, 2022. doi: 10.1007/978-3-030-91032-7.

[2] O. Rössler, “An equation for continuous chaos,” *Physics Letters A*, vol. 57, pp. 397–398, 1976, doi: 10.1016/0375-9601(76)90101-8.

[3] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017, doi: 10.1137/141000671.

[4] C. Rackauckas and Q. Nie, “Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia,” *Journal of Open Research Software*, vol. 5, no. 1, p. 15, 2017.

[5] S. Danisch and J. Krumbiegel, “Makie.jl: Flexible high-performance data visualization for Julia,” *Journal of Open Source Software*, vol. 6, no. 65, p. 3349, 2021, doi: 10.21105/joss.03349.

[6] M. Bouchet-Valat and B. Kamiński, “DataFrames.jl: Flexible and Fast Tabular Data in Julia,” *Journal of Statistical Software*, vol. 107, no. 4, pp. 1–32, 2023, doi: 10.18637/jss.v107.i04.

[7] D. P. Sanders and L. Benet, “IntervalArithmetic.jl.” [Online]. Available: https://github.com/JuliaIntervals/IntervalArithmetic.jl