# Using Circles to Detect Bifurcations in Dynamical Systems with Persistent Homology

Chase Demick
Skidmore College Department of Mathematics

Dr. Junyi Tu
Salisbury University Department of Computer Science

## Abstract

Dynamical systems and topology are fields that often interact. Topology often deals with the overall properties of a space, while the study of dynamical systems deals with the overall behavior of a system without necessarily being preoccupied with numerical details. Key in the study of global behavior is the analysis of bifurcations, which are quantifiable changes to the system's behavior with respect to some set of parameters. We use tools from the growing field of persistent homology in order to develop a procedure to analyze dynamical systems by comparing the homologies of their trajectories, known as orbits, to those of circles approximating the orbit's long-term behavior. We then implemented the procedure into Python principally using the `giotto-tda` module.

## Outline of the Circle Comparison Method

We compare the orbit of the time series to that of an approximating circle using persistent homology. If the two are close homologically, then the system likely has some periodicity. Otherwise, the system may have other types of behavior.

1. Compute an orbit of the dynamical system.
2. Pick a sample set from this orbit to represent the end behavior.
3. Choose a midpoint of the sample set.
4. Choose a radius of a circle to approximate the orbit.
5. Compute persistence diagrams of the sample set and circle.
6. Find the distance between the persistence diagrams.

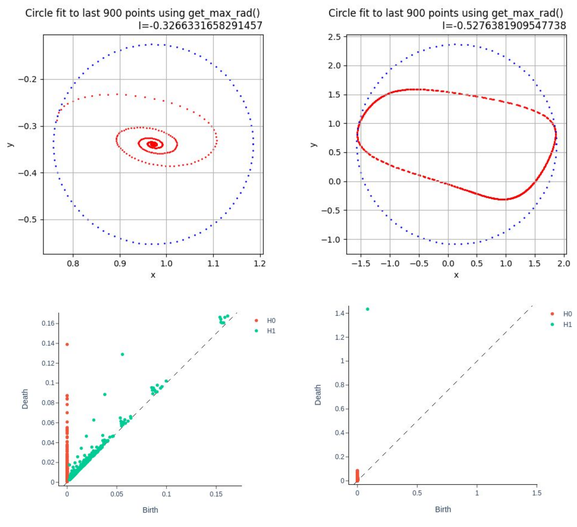


*Figure 1: The FitzHugh-Nagumo system, corresponding circles, and persistent diagrams at two different parameter values – one gives a fixed point attractor (left column) while another gives a limit cycle (right column).*

## Choosing the Circle Radius

- Maximum radius : $R_{\max}(X_s) = \max_{x \in X_s} \|x - m(X_s)\|$
- Minimum radius : $R_{\min}(X_s) = \min_{x \in X_s} \|x - m(X_s)\|$
- Average of minimum and maximum radii : $R_{\text{avg minmax}}(X_s) = \dfrac{R_{\min}(X_s) + R_{\max}(X_s)}{2}$
- Average of all radii : $R_{\text{avg overall}}(X_s) = \operatorname{avg}_{x \in X_s} \|x - m(X_s)\|$

## Topological Data Analysis (TDA) Tools

- We calculate the simplicial complex for our time series and circle separately using the Vietoris-Rips complex and persistence:

$$VR_s(X) = \left\{ [v_0, \ldots, v_n] \;\middle|\; \forall i, j \; d(v_i, v_j) \leq s \right\}.$$

  We choose d(-, -) to be the standard Euclidean metric.
- Then, we use the Wasserstein Metric to give a distance between the persistent diagrams of the time series and the circle.

$$W_q(X, Y) = \left[ \inf_{\eta : X \to Y} \sum_{x \in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}$$

- We plot the distance of the system to its circle as one of the system's parameters changes.

## Application to FitzHugh-Nagumo System

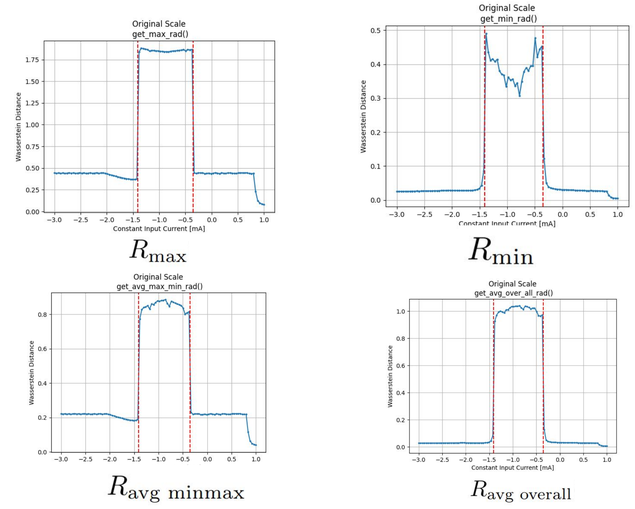


*Figure 2: The circle comparison method applied to the FitzHugh-Nagumo system using each radius function.*

## Application to Lorenz and Rössler Systems

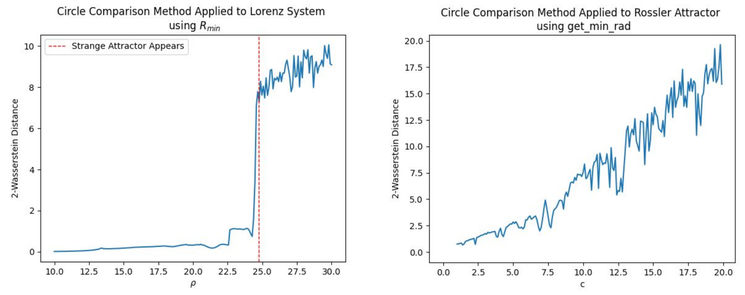


*Figure 3: The circle comparison method applied to the Lorenz system (left) and the Rössler system (right). The Rössler system, having few clear bifurcations between chaotic and periodic, rapidly switches between the two, generating noise in the graph.*

## Implementation in Python

A snippet of the Python code for generating the parameter-vs-Wasserstein distance plots:

```
def rho_to_PD(rho, pos, rad_func):
    # calculate trajectory for each rho
    t, ts = dynamical_system(rho)

    # take two coordinates, here choose (x,y)
    chosen_coords = ts[:, 0:2]

    # Get samples from when ts has settled into its end behavior
    ts_sampled, limit = get_limit_cycle_samples(chosen_coords)

    # center the time series
    ts_centered, midpoint = get_midpoint_center(ts_sampled, limit)

    # Get original radius of circle
    radius, rad_func_name = get_radius(ts_centered, rad_func=rad_func)

    if pos == 'Normalized':
        # Scale the data to fit unit circle
        ts_final, circle = centered_data_normalized(ts_centered, radius)
    elif pos == 'Original':
        # don't do anything to the time series
        ts_final = ts_sampled
        circle = create_circle(midpoint, radius)

    # calculate ts_final persistence
    FH_diagram = VR.fit_transform(ts_final[None, :, :])

    return ts_final, FH_diagram, circle, rad_func_name, att_type
```

## Future Work

We would like to reduce the noise in the parameter-vs-distance plots in order to better distinguish state change locations. Also, we want to generalize the circle comparison to $n$-tori in order to analyze higher-dimensional systems without resorting to plane projection.

## References

- Ghrist, R. (2008). Barcodes: The persistent topology of data. Bulletin of the American Mathematical Society, 45 (1), 61–75.
- Myers, A., Munch, E., & Khasawneh, F. A. (2019). Persistent homology of complex networks for dynamic state detection. Phys. Rev. E, 100, 022314. https://doi.org/10.1103/PhysRevE.100.022314

This project was a part of the EXERCISE REU at Salisbury University, MD, NSF grant #2149591. Special thanks to P.I. Dr. Enyue Lu.