





RESEARCH ARTICLE | JULY 27 2023

# Framework for global stability analysis of dynamical systems

George Datservis   ; Kael Luiz Rossi  ; Alexandre Wagemakers 



Chaos 33, 073151 (2023)

<https://doi.org/10.1063/5.0159675>



CrossMark



**APL Quantum**  
Bridging fundamental quantum research with technological applications

**Now Open for Submissions**  
No Article Processing Charges (APCs) through 2024

**Submit Today**



# Framework for global stability analysis of dynamical systems

Cite as: Chaos 33, 073151 (2023); doi: 10.1063/5.0159675

Submitted: 25 May 2023 · Accepted: 3 July 2023 ·

Published Online: 27 July 2023






View Online



Export Citation



CrossMark

George Datsaris,<sup>1,a)</sup>  Kaeli Luiz Rossi,<sup>2</sup>  and Alexandre Wagemakers<sup>3</sup> 

## AFFILIATIONS

<sup>1</sup>Department of Mathematics and Statistics, University of Exeter, North Park Road, Exeter EX4 4QF, United Kingdom

<sup>2</sup>Theoretical Physics/Complex Systems, ICBM, Carl von Ossietzky University of Oldenburg, Carl-von-Ossietzky-Straße 9-11, 26111 Oldenburg, Germany

<sup>3</sup>Nonlinear Dynamics, Chaos and Complex Systems Group, Departamento de Física, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain

<sup>a)</sup>Author to whom correspondence should be addressed: [g.datseris@exeter.ac.uk](mailto:g.datseris@exeter.ac.uk)

## ABSTRACT

Dynamical systems that are used to model power grids, the brain, and other physical systems can exhibit coexisting stable states known as attractors. A powerful tool to understand such systems, as well as to better predict when they may “tip” from one stable state to the other, is global stability analysis. It involves identifying the initial conditions that converge to each attractor, known as the basins of attraction, measuring the relative volume of these basins in state space, and quantifying how these fractions change as a system parameter evolves. By improving existing approaches, we present a comprehensive framework that allows for global stability analysis of dynamical systems. Notably, our framework enables the analysis to be made efficiently and conveniently over a parameter range. As such, it becomes an essential tool for stability analysis of dynamical systems that goes beyond local stability analysis offered by alternative frameworks. We demonstrate the effectiveness of our approach on a variety of models, including climate, power grids, ecosystems, and more. Our framework is available as simple-to-use open-source code as part of the DynamicalSystems.jl library.

© 2023 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>). <https://doi.org/10.1063/5.0159675>

Dynamical systems are ubiquitous to model natural phenomena ranging from climate, ecosystems, and more. Global stability analysis is the study of their attracting states over the full state space. It goes beyond local bifurcation analysis and provides insight into the resilience of the states, i.e., how close they are to “tip” to a different, perhaps undesirable stable state such as population death in ecosystems or circulation shutdown in climate. In this work, we develop an accurate and flexible framework for global stability analysis of dynamical systems that we believe will accelerate multistability and tipping point research in many fields. We also provide a software implementation that is fast and easy to use in the open-source DynamicalSystems.jl library.

## I. INTRODUCTION

Multistable dynamical systems exhibit two or more co-existing stable states, formally called *attractors*. Multistability is ubiquitous

in nature and in mathematical models,<sup>1,2</sup> with examples ranging from power grids,<sup>3–5</sup> the climate,<sup>6,7</sup> ecosystems like the Amazon rain forest,<sup>8,9</sup> the brain and neuronal circuits therein,<sup>10–12</sup> or metabolic systems.<sup>13–15</sup> Some attractors of these systems can be desirable, such as synchronized oscillations in power grids, crucial for their proper functioning.<sup>16</sup> However, they can also be undesirable, as, for example, the extinction of a certain species in ecological models or the collapse of circulation in climate models.<sup>17</sup> In a multistable system, the attractor at which the system ends up depends on the initial conditions (ICs), but perturbations of the state may enforce switching between attractors, a phenomenon called “tipping.”<sup>1,9,18</sup> Alterations in the parameters of a dynamical system can trigger tipping. Hence, it becomes important to evaluate how “resilient” attractors are to perturbations, either to parameters or to the system’s variables. This is a crucial problem of practical importance in several areas of research.<sup>9,19</sup>

A traditional solution to this problem is the *continuation-based bifurcation analysis* (CBA). It identifies fixed points and (under

some requirements) limit cycles and describes their *linear* (also called *exponential*) stability dependence on a system parameter via the eigenvalues of the Jacobian.<sup>20</sup> One major downside is that, by definition, it cannot be applied to chaotic attractors.<sup>20</sup> In the majority of cases, this analysis must be done numerically via one of several software, e.g., AUTO,<sup>21</sup> MATCONT,<sup>22</sup> CoCo,<sup>23</sup> or Bifurcationkit.jl.<sup>24</sup> Continuation tools can be combined with other numerical techniques to enrich the exploration of the parameter space as shown in Ref. 25. The information provided by these frameworks is useful but incomplete: rigorously speaking, local stability only conveys information about the system's response to infinitesimally small perturbations. It cannot yield insight into the response to finite perturbations in the state space, which are predominant in practice.

For such responses, it is necessary to study the *global stability*<sup>26</sup> of the system's attractors, which involves the nonlinear dynamics over the full state space of the system.<sup>27</sup> A proxy for global stability of an attractor is the portion of all possible initial conditions ending up at this attractor, i.e., the *fraction* of the state space that is in the *basin* of said attractor. When the state space is infinite, the concept of the state space fraction becomes a pragmatic one: we need to define a finite-volume box of *physically plausible* initial conditions for the system under study, and we are concerned about the fractions of these plausible initial conditions. In this analysis, attractors with larger basins fractions are globally more stable because stronger perturbations are typically needed to switch the system to another attractor.<sup>26</sup> Frequently, this measure is also a much better indicator of the loss of stability as a system parameter is varied when compared to the local stability analysis of the system (see, e.g., Ref. 26 or Chap. 12 of Ref. 20).

Analyzing global stability as a function of a parameter demands extensive effort from researchers, as it requires the creation of algorithms that can find system attractors and their global stability, “continue” them across a parameter and also perform the expensive numerical simulations required for such algorithms. In the literature, the only framework so far that can aid this analysis is the *featurizing and grouping* approach, proposed first by Gelbrecht *et al.*<sup>28</sup> as MCBB (Monte Carlo Basin Bifurcation Analysis) and then later very similarly by Stender and Hoffmann<sup>29</sup> as bSTAB (basin stability). The method integrates randomly sampled initial conditions (ICs) of a dynamical system for a preset time span. The trajectories of these ICs are then mapped onto *features*, numbers describing the trajectories, such as the mean or standard deviation of some of the system variables. All the feature vectors are clustered using the DBSCAN algorithm<sup>30</sup> so that ideally each cluster corresponds to an attractor of the system. The fractions of ICs in each cluster approximate the basin fractions and, hence, the global stability of each attractor. More details on the method are given in Sec. IV.

This method works well in a variety of circumstances and can also be applied across a parameter range. However, it comes with significant downsides. One is that it is not clear *a priori* which features should be chosen to correctly separate the attractors into clusters, requiring a lot of trial and error. Another downside is that the method cannot guarantee that the clusters of features really correspond to unique attractors and that it is not mixing two or more attractors together. An alternative method for finding attractors and their basins of attraction is the *recurrence-based* approach recently proposed.<sup>31</sup> The method locates attractors by finding recurrences in

the system's state space, assuming the Poincaré recurrence theorem holds for the system attractors. The input to this method is a state space box, and its tessellation, defining a grid to search for recurrences. Hence, the method will only find attractors within the given box, although the box can initially be arbitrarily large. We describe the method in more detail in Sec. IV B and provide a comparison between the two techniques in Sec. III B. The main advantage of the recurrences method is that it locates the actual system attractors and only requires as an input a state space box that may contain the attractors. So far, however, it has not been clear how to “continue” attractors across a parameter range with this method.

In this work, in Sec. III A, we present a novel global stability analysis and *continuation* algorithm that utilizes the recurrences-based method for finding attractors.<sup>31</sup> In Sec. III C, we apply it to exemplary models of climate, ecosystem dynamics, and more. As detailed in Sec. III, this novel continuation algorithm is the most accurate in finding the actual attractors of a dynamical system, the most transparent in matching attractors across parameter values, and requires the least amount of guesswork from the researcher. We believe that this novel continuation of global stability, much like global stability analysis itself,<sup>26</sup> is a crucial new tool for the analysis of dynamical systems. In some cases, it supersedes CBA, and in others, it can be complemented by CBA, as we discuss in Sec. III A.

This continuation method is part of a novel automated framework that performs global stability analysis and continuation, which we present in Sec. III B. Our framework significantly advances existing methodology, including the featurizing methods, thereby including all upsides of current literature while addressing most downsides (Secs. IV C and IV D describe the improvements in detail). Its design is based on modular components that can be configured or extended independently. This allows researchers to simply compose the methodology that is best suited to their problem and then let an automated algorithm execute the process. The framework is accompanied by an extensively tested, well documented, and highly optimized open-source software implementation, part of the DynamicalSystems.jl<sup>32</sup> general purpose library for nonlinear dynamics (see Sec. IV for code example and documentation).

## II. RESULTS

### A. Novel global stability continuation algorithm

A major contribution of our framework is the novel algorithm for global stability analysis and continuation that we name *recurrences-based attractor find-and-match continuation*, RAFM for short. This algorithm can be applied to any system whose attractors satisfy the Poincaré recurrence theorem. As illustrated in Fig. 1(a), it works as follows.

- Step 0: the starting point of the algorithm. Attractors and their basins, or basins fractions, are already known at a parameter  $p = p_1$  using the recurrences-based algorithm.<sup>31</sup>
- Step 1: new initial conditions are seeded from the existing attractors. Then, we set the system parameter to  $p = p_2$ .
- Step 2: evolve the seeded initial conditions according to the dynamic rule of the system. The seeds are evolved until they converge to an attractor using the recurrences-based method (the grid reflects the tessellation of the state space

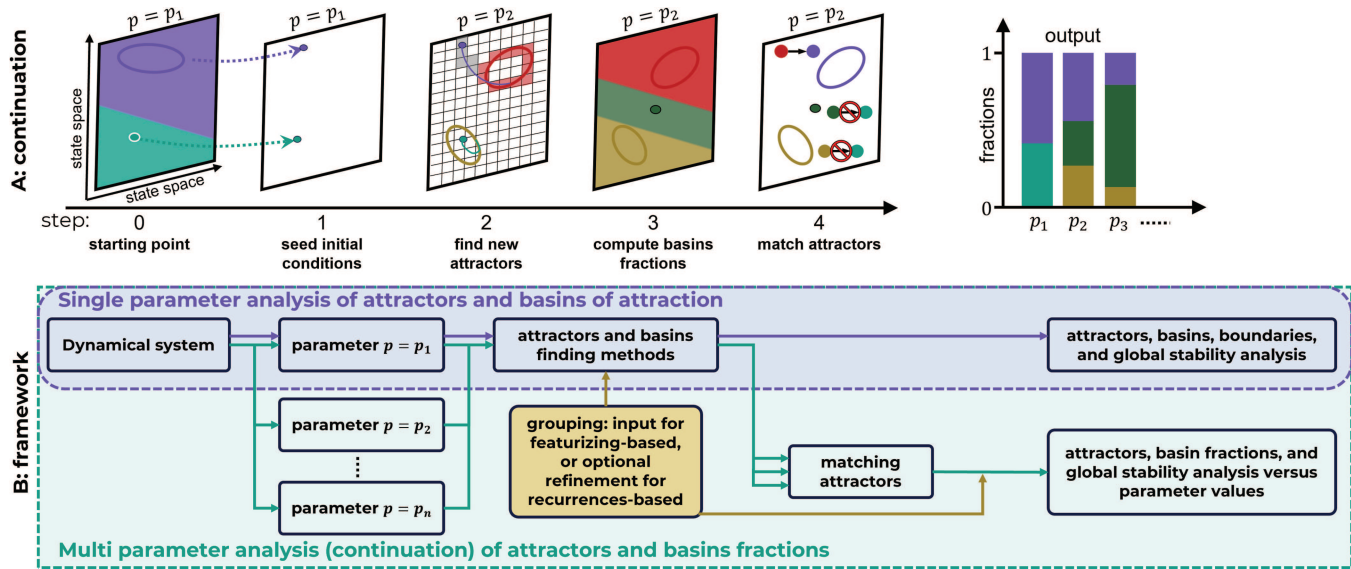


FIG. 1. (a) The recurrences-based find-and-match (RAFM) algorithm for global stability continuation described in Sec. III A. (b) Schematic illustration of the modular framework for global stability analysis and continuation described in Sec. III B.

that is decided by the user; the finer the grid, the more accurate the results<sup>31</sup>). The main performance bottleneck of the recurrences-based method is finding the attractors. Once found, convergence of other initial conditions is generally much faster.<sup>31</sup> To address this in the continuation, we use the observation that, unless a bifurcation is occurring, attractor size, shape, and position, typically depend smoothly on  $p$ . Hence, the seeded initial conditions at each new parameter will most likely converge the fastest to the new attractors.

- Step 3: with the main bottleneck of the algorithm (finding the attractors) being taken care of, now compute the basins fractions by formally applying the recurrence-based algorithm<sup>31</sup> to randomly sampled initial conditions. It can be arbitrarily configured how to randomly sample initial conditions, but typically the samples are drawn uniformly from a state space box. Importantly, the algorithm may still find new attractors during this step (here the “dark green” one) that did not exist before.
- Step 4: match attractors in current parameter  $p_2$  to those in parameter  $p_1$ . Matching is arguably the most sophisticated part of the algorithm. Attractors are matched by their “distance” in state space, with distance any arbitrary metric on the space of state space sets, see Sec. III D for more details. In this illustration, the “red” attractor is matched to the “purple” one of Step 0, while neither the “yellow” or “dark green” attractors match to the previous “teal” one, because their state space distance is beyond a pre-defined threshold.

The end result is the (matched) system attractors, and their basins fractions (or full basins if computationally feasible), as functions of the parameter. The attractors and basins are labeled

with the positive integers (enumerating the different attractors), and the basins always sum to 1. Note that because RAFM works on a parameter-by-parameter basis, it can be used to perform continuation across any number of parameters, not just one (we present one here as the simplest conceptual example).

### B. Global stability continuation framework

To perform global stability analysis, several tasks need to be taken in sequence [see Fig. 1(b) for an overview]. We have abstracted and generalized the tasks to allow researchers different possibilities of how to achieve them.

The first task is the creation of a dynamical system for the global stability analysis. For our framework, this is achieved for free simply by making the implementation part of the DynamicalSystems.jl library<sup>32</sup> (see Sec. IV E).

The second task is the creation of a mechanism to find attractors and map initial conditions to them. Possibilities for this mechanism are: (1) *featurizing and grouping* initial conditions into attractors (as discussed in the introduction), (2) finding attractors using the *recurrences algorithm*,<sup>31</sup> or (3) mapping initial conditions to *previously known attractors by proximity*: once the evolution of an initial condition comes close enough to a pre-determined attractor, the initial condition is mapped to that attractor. Note that in (1) or (2), attractors are found via random sampling in the state space. For the typically used uniform sampling, the probability to find an attractor with basin fraction  $f$  after  $n$  samples is  $1 - (1 - f)^n$ . However, users may use system-specific knowledge to adjust the sampling into something that may find attractors with better scaling with respect to sample number  $n$ .

Mechanism (1) is paired with instructions on how to group features. Currently, the possibilities are: (a) *clustering* features into groups using DBSCAN (as done in MCBB or bSTAB), (b) grouping features by *histograms in the feature space*, so that features that end up in the same histogram bin belong to the same group (novel grouping approach), or (c) mapping features to their *nearest feature* in feature space, from a set of pre-defined features (as done in bSTAB). Having more grouping possibilities than only clustering can be useful, as discussed in Sec. III D.

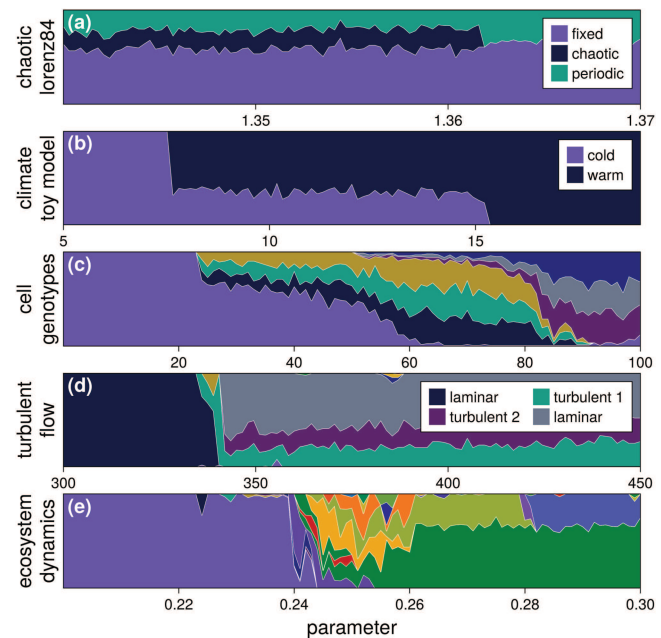
At this point, one can analyze global stability at a given parameter and also analyze the basin boundaries of attractors for fractal properties. From here, the third task is to “continue” the attractors and basins across a parameter range. Our framework currently has two continuation algorithms; however, due to an extendable design, more may be added in the future. The first continuation algorithm is what has been employed so far by the MCBB or bSTAB algorithms, but with significantly increased accuracy (see MatMeth), and extended to allow any kind of instructions for how to group features. We will call this “Featurize and Group Across Parameter” (FGAP). In this approach, trajectories from the dynamical system are generated by sampling random ICs across all parameter values of interest. All these trajectories are mapped to features, and all these feature vectors are then grouped using one of currently three grouping instructions (clustering, histogram, nearest feature). Each group is representing an attractor. The grouped ICs are then re-distributed into the parameter slices they came from, providing the fractions of each group at each parameter value. The second continuation algorithm is the RAFM algorithm that we described in Sec. II A. These two approaches are compared in detail in Sec. III B.

### C. Application on exemplary systems

In Fig. 2, we apply RAFM on some exemplary systems. We stress that we could characterize the different attractors in accurate detail because RAFM finds the actual system attractors, not some incomplete representation of them (i.e., features used in the featurizing-and-grouping approach).

To illustrate a concrete application of the method, we discuss how the example of the Lorenz 84 dynamical system in Fig. 2(a) was generated. Along with the following step by step “tutorial,” we provide the computer code for this example in Listing 1. First, a continuous time dynamical system object `ds` (whose equations are provided in Sec. IV F) is created, and we choose the ODE integrator according to the system specifics (here we use the Verner 9th order solver<sup>38</sup>). Next, we choose the state space box, and its tessellation, that will be used both for the recurrences-algorithm<sup>31</sup> as well as for sampling random initial conditions when estimating the basin fractions. For this example, the box ranges from  $-3$  to  $+3$  divided into 600 points along each dimension.

With the grid and the dynamical system, we construct a `mapper` object that maps initial conditions to attractors using the recurrences method.<sup>31</sup> To decide the meta-parameters of this algorithm, we consult our previous publication specifically about the algorithm.<sup>31</sup> For this example, the presence of a chaotic attractor involves longer recurrence times, so we set high values for the parameters `mx_chk_fnd_att` and `mx_chk_loc_att` to improve the precision of the characterization of the attractors.



**FIG. 2.** Basins fraction continuation for exemplary dynamical systems using the novel recurrence-based continuation algorithm. The fractions of the basins of attraction are plotted as stacked band plots (hence, summing to 1). Each color corresponds to a unique attractor that is found and continued (but not plotted here). Each simulation scanned 101 parameter values, and in each, it sampled randomly 100 initial conditions. The fractions fluctuate strongly vs parameter not due to lack of convergence, but because the basin boundaries are fractal in all systems considered. The systems used are: (a) three-dimensional paradigmatic chaotic model by Lorenz (Lorenz84<sup>33</sup>) with a co-existence of a fixed point, limit cycle, and chaotic attractor, undergoing a crisis with the chaotic attractor merging into the limit cycle; (b) 33-dimensional climate toy model<sup>34</sup> featuring bistability of chaotic attractors; (c) three-dimensional multistable cell-division model,<sup>35</sup> where each cell type is considered to be a distinct attractor in the gene activity state space; (d) nine-dimensional model for turbulent shear flow model in which the fluid between two walls experiences sinusoidal body forces;<sup>36</sup> (e) eight-dimensional ecosystem competition dynamics model<sup>37</sup> featuring extreme multistability (due to the number of attractors, we made no effort to label them further).

Sometimes, transient trajectories may spend some time outside the defined grid. In this case, we can either provide a larger state space box or set the parameter `mx_chk_lost` to a higher value. Very small time steps may lead to the identification of false attractors if the orbit stays too long in a grid cell. To prevent these artifacts, we specify that during time integration, a non-adaptive method with a fixed time step of  $\Delta t$  should be used (note that the value of  $\Delta t$  will depend on the system’s internal timescale).

We chose 100 values of the parameter  $G$  in the range  $[1.34; 1.37]$  and construct a continuation object `rsc`, which instructs how to perform the continuation (in this case, this corresponds to using the RAFM algorithm). During the construction of `rsc`, we could also specify how to match attractors (which is something we only discuss in Sec. II D and in this example, we used the default matching). We also specify how to sample random initial conditions via the `sampler` object, which in this case is randomly

drawn initial conditions within the limits of the provided state space box. The continuation output is obtained after the call of the `continuation` function. This workflow is identical for the other examples used, in addition to the obvious changes of the dynamical system, state space box, parameter range, and possible tuning of meta-parameters. We also stress that the best place to learn how to choose optimal meta-parameters of all these algorithms is the documentation of the associated software, which is constantly updated with best practices.<sup>39</sup>

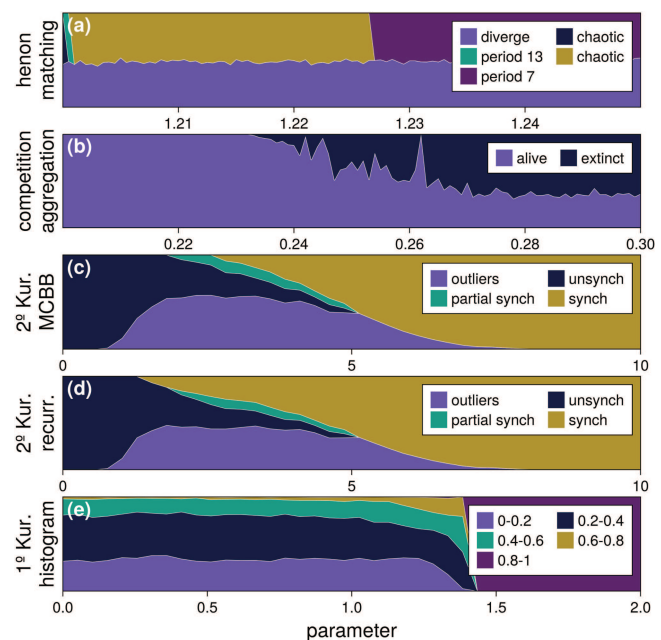
#### D. Matching and grouping attractors

Traditional CBA has a rigid “matching” procedure: it always matches the next point found along a “continuation curve” to the previous point. This is often correct for infinitesimal perturbations of fixed points but becomes problematic for global stability analysis, which attempts to find all attractors in a state space box and then continue them. In this case, matching attractors from one parameter to the next becomes a crucial part of the algorithm. For instance, the analysis presented in Fig. 2 is only coherent because of the powerful matching procedure implemented in our framework. Without it, the colors would alternate arbitrarily at each parameter value.

In the featurize-and-group algorithm, matching and grouping are the same process. In RAFM, matching is rather sophisticated and operates on a parameter-by-parameter basis. Each time the parameter is incremented and the new attractors are found, a matching sub-routine is launched. The distance between attractors before and after the parameter change is estimated, with “distance” being any symmetric positive-definite function defined on the space of state space sets. By default, the Euclidean distance of the attractor centroids is used because it is extremely fast and in the majority of cases, it works very well. A more rigorous metric is the Hausdorff distance,<sup>40</sup> which is also provided out of the box. Additionally, “distance” is not limited to state space distances. It could be the distance across dynamic invariants. For example, one can track the Lyapunov spectrum or the fractal dimension<sup>20</sup> of each attractor and define a distance in terms of their absolute difference. This is easily possible in our code implementation because it is part of `DynamicalSystems.jl`, which offers algorithms for computing Lyapunov spectra or fractal dimensions out of the box.

After the distance is computed between all new-old attractor pairs, the new attractor labels are matched to the previous attractor labels that have the smallest distance to them, prioritizing pairs with the smallest distance. The matching respects uniqueness, so that once an attractor from the previous parameter has been matched, this attractor is removed from the matching pool and cannot be matched to an additional new attractor. Additionally, a distance threshold value can be provided, so that old-new pairs of attractors whose distance is larger than this threshold are guaranteed to get assigned different IDs. Note that in principle finding the attractors and matching them are two completely independent processes. If after the continuation process is finished the user decides that the chosen matching procedure was unsuitable, they can launch a “re-matching” algorithm with different matching “distance” function, without having to re-do any computations for finding the attractors or their fractions (i.e., matching only renames attractor labels but leaves the attractors themselves untouched).

The last thing to highlight in this section is the desirable post-processing of grouping similar enough attractors. This happens automatically if one uses the featurize-and-group continuation method. However, taking as an example Fig. 2(e), the RAFM method finds countless individual attractors. For the researcher, the individual attractors may be useful for careful analysis, but it is sometimes desirable to group similar enough attractors. In our framework, it is possible to use exactly the same grouping infrastructure utilized by the featurizing-and-grouping continuation, but now applied to the outcome of RAFM as a post-processing step. In Fig. 3,



**FIG. 3.** Highlights of the matching or grouping components of the framework. (a) Matching attractors of the Hénon map based on their period. In the chosen parameter range, an attractor is transformed from chaotic, to period 3, 7, and 14. The attractor stays in approximately the same state space location, so whether we consider the centroid distance or the Hausdorff distance, the attractor would be matched to itself in all parameter values due to the very small distance evaluation. Here, however, we use as distance  $f(A, B) = |\log_2(\text{len}(A)) - \log_2(\text{len}(B))|$ , with  $\text{len}$  measuring the amount of cells (of the state space tessellation) the attractor covers, and threshold  $t = 0.99$ . This effectively means that matched attractors must have periods with a ratio less than 2. (b) Grouping attractors of Fig. 2(e) so that attractors are grouped into those whose third species has a population of less than 0.01 or more. (c) A replication of the MCBB<sup>28</sup> results for a second-order Kuramoto oscillator network representing a power grid, using the featurize-and-group continuation implementation from our framework. Features extracted from sampled trajectories are the means of the frequencies. (d) Same system as (c), but using the recurrence continuation and matching attractors by their centroid distance (i.e., as in Fig. 2). The only extra step was to post-process the results so that all attractors with basin fractions less than 4% are aggregated [as was done in Ref. 28 and in panel (c)]. (e) Attractor basin fractions for a network of first-order Kuramoto oscillators; the attractors here are found and matched using the recurrences continuation and then grouped via a histogram of their synchronization order parameter  $R$  (Chap. 9 of Ref. 20). Attractors whose order parameter  $R$  fall in the same histogram bin are aggregated.

we highlight examples that utilize the powerful matching and/or grouping components offered by our framework.

### III. DISCUSSION

#### A. Comparison with traditional continuation-based bifurcation analysis

In Table I and Fig. 4, we provide a careful comparison between CBA and RAFM. A direct comparison of the two approaches is

difficult, since their operational basis, main output, and even the way stability is quantified, are fundamentally different. On one hand, CBA finds and continues the curves of individual fixed points or limit cycles across the joint state-parameter space on the basis of Newton's method. The stability is quantified in terms of local stability (also called exponential stability via the Jacobian eigenvalues). On the other hand, RAFM first finds attractors at all parameter values using the original system equations and then *matches* appropriately similar attractors across different parameters, giving the illusion of continuing them individually. Additionally, the curves of stable

**TABLE I.** A comparison between CBA and RAFM as tools for analyzing the stability of a dynamical system vs a parameter. Entries are colored blue when they contain advantages of CBA over RAFM, and green when they contain advantages of RAFM over CBA.

#	Traditional continuation-based bifurcation analysis (CBA)	Recurrences-based attractor find-and-match continuation (RAFM)
1	Provides curves of unstable fixed points/limit cycles.	Only finds attracting sets in the formal attractor sense (i.e., saddles, stable/unstable manifolds are excluded).
2	Several possibilities for how to continue bifurcation curves.	Does not explicitly detect bifurcation points.
3	Does not put limits on state space extent.	Needs as an input a state space box that may contain attractors.
4 <sup>a</sup>	Likely to find fixed points/limit cycles with small or even zero basins.	Probability to find attractor is proportional to its basins fraction.
5	Detects and classifies local bifurcation points.	Does not compute Jacobian eigenvalues at all.
6 <sup>b</sup>	Finds and continues fixed points and periodic orbits.	Finds and continues any kind of attractors, including quasiperiodic or chaotic.
7 <sup>c</sup>	User must manually search for multistability.	Different attractors are automatically detected (via random sampling).
8	Does not compute the basins of attraction or their fractions.	Computes the fractions and, if computationally feasible, also the full basins.
9 <sup>d</sup>	Limited use in indicating loss of stability.	More likely to indicate loss of stability as the basin fraction approaches 0.
10 <sup>e</sup>	Parameter change may not affect linear stability of all fixed points.	Parameter change is more likely to affect global stability of all attractors.
11	No sophistication on matching fixed points.	Sophisticated, user-configurable matching of attractors.
12	Requires expertise and constant interventions.	Conceptually straightforward even in advanced use-cases.

<sup>a</sup>Newton's method transforms the dynamical system into a discrete time system with different basins, making attractors with very small or zero basin sizes have much larger ones instead.

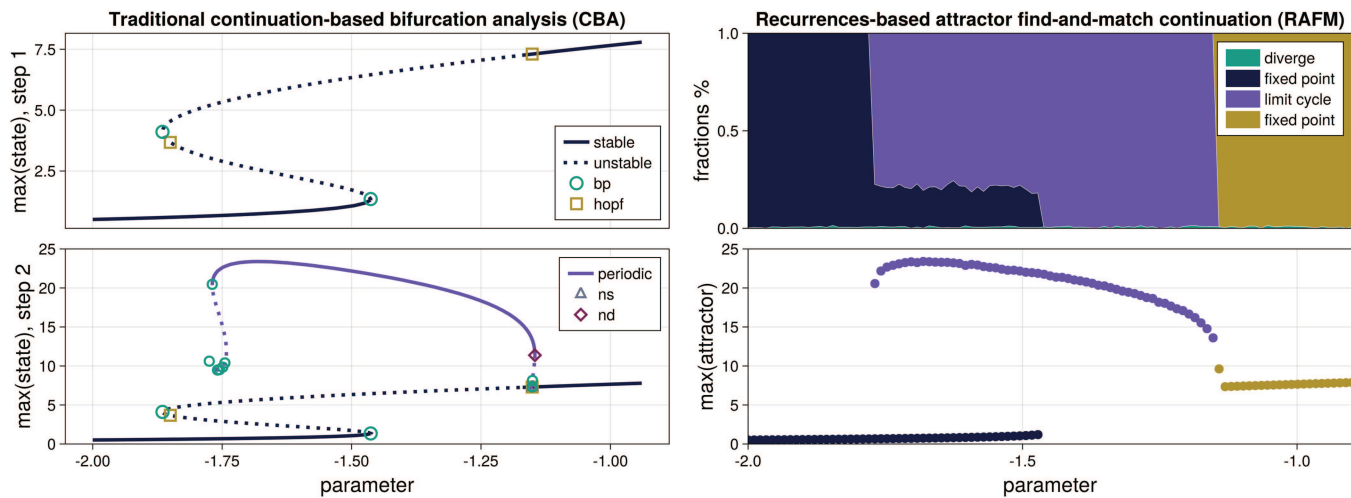
<sup>b</sup>The CBA method may also find tori but this scenario is highly specific to the exact system and bifurcations it undergoes and not applicable in the general sense. Additionally, the word "finds" should be taken with a grain of salt. The user needs to provide an initial condition that would be in the Newton-method-transformed basin of attraction of, e.g., the limit cycle the method has to find. This basin cannot be related with the real system dynamics in any obvious way, requiring an arbitrary degree of trial-and-error to find an initial condition leading to a limit cycle. Alternatively, the user must find the limit cycle prior to using CBA and provide a point on the cycle for the continuation.

<sup>c</sup>In RAFM, attractors that are not being continued from a previously found one, are found via random sampling of initial conditions in the given state space box. The probability to find an attractor is equal to  $1 - (1 - f)^n$  with  $f$  the basins fraction of the attractor and  $n$  the amount of sampled initial conditions.

<sup>d</sup>Changing a parameter often does not meaningfully increase the unstable eigenvalues of the Jacobian matrix, which would indicate loss of stability (Chap. 12 of Ref. 20). On the other hand, basin fractions typically decrease smoothly toward zero as an attractor loses stability,<sup>26</sup> although this is not guaranteed to be the case,<sup>41</sup> in which scenario, neither method indicates loss of stability.

<sup>e</sup>Change of a parameter may affect the local (exponential) stability of a single fixed point, not all, providing flat lines in the bifurcation diagram for the unaffected fixed points. On the contrary, loss of global stability of any attractor affects (typically increases) the global stability of all other attractors.

<sup>f</sup>Advanced applications of traditional bifurcation analysis software require several manual interventions during the process and tuning of several configuration options, many of which do not have an immediately transparent role, requiring an expert user to make several decisions. The simplicity of our approach comes in part because of the brute-force nature of mapping individual initial conditions to attractors to collect the fractions, the intuitive nature of how attractors are matched (which is also user configurable), and the lack of necessity of interventions: after the configuration is decided, the framework runs automatically.



**FIG. 4.** Stability analysis of a three-dimensional neural dynamics model,<sup>42</sup> plotting as information the maximum of the first variable of the dynamical system. For the parameters used, the model undergoes saddle-node and Hopf bifurcations and features the bistability of a limit cycle and a fixed point. Left: analysis using the BifurcationKit.jl<sup>24</sup> Julia package for CBA. The analysis must happen in two steps; first, the branch of a fixed point is found and continued, and then, using a different algorithm, the branch of the limit cycle is continued from the Hopf bifurcation. We scatterplot special points found and labeled by the process. The computation required  $\sim 19$  s on an average laptop. Right: analysis of the same model using our framework. The analysis happens in one fully automated step, after deciding the state space box and other meta-parameters. For the analysis, we purposefully demanded unnecessarily high accuracy, using a ninth-order ODE solver<sup>38,43</sup> with tolerances of  $10^{-9}$ , and requiring 1000 recurrences before claiming convergence in the recurrence algorithm.<sup>31</sup> The process integrated in total 101 000 initial conditions, yet required  $\sim 16$  s on the same laptop. Attractor matching utilized a threshold: attractors whose distance in terms of their maximum value of the first variable (i.e., same information plotted in the figure) exceeding 3.0 are not matched.

fixed points in the joint parameter space (Fig. 4) are only a small part of the information our framework provides. Important, provided information is the basin fractions and how they change, which is completely absent in CBA.

Based on this comparison, we argue that the RAFM algorithm and the global stability framework we provide is an essential tool for stability analysis of dynamical systems. We further believe that in some application scenarios, RAFM will supersede CBA, especially given the difference in required user expertise, required interventions, and steepness of the learning curve that CBA has over RAFM. In other scenarios, we envision that RAFM can be used as the default analysis method, providing the majority of information, and CBA then becomes a more in-depth analysis of fixed points, limit cycles, bifurcations, and even stable/unstable manifolds, if such analysis is required.

## B. Comparison between attractor-finding methods

Our framework provides two radically different methods for finding attractors: the recurrence-based and the featurize-and-group approach. Generally speaking, the recurrence-based method should be preferred when possible, because of its accuracy (finding the actual attractors), and the possibility for follow-up analysis of found attractors. However, the featurize-and-group method should be preferred when the recurrence-based method fails, because, e.g., the provided state space tessellation is ill-defined, or because computational demands exceed what is available, such as in higher-dimensional systems with chaotic attractors. In Table II, we provide a comprehensive comparison between the two methods.

## IV. MATERIAL AND METHODS

### A. Featurizing methods for finding attractors

We group together two similar methods that have been recently proposed in the literature for finding attractors. One is called Monte Carlo basin bifurcation analysis (MCBB),<sup>28</sup> the other is basin stability analysis (bSTAB).<sup>29</sup> Both methods work by identifying attractors as clusters of user-defined-features of trajectories. Their first step is to integrate  $N$  randomly chosen initial conditions inside a certain box in state space. The integration is done for some time  $T$ , after a transient  $T_{tr}$ . Both  $T$  need to be sufficiently large so that the trajectories correspond to the systems' long-term behavior and also  $T_{tr}$  needs to be large to avoid the transient regime. Each trajectory  $\vec{x}(t)$  is then transformed into a vector of  $K$  features  $\vec{F}$ , specified by some featurizer function  $\vec{f}$  such that  $\vec{f}(\vec{x}(t)) = \vec{F}$  that has to be defined by the user. Each vector of features  $\vec{F}$  describes a point in the  $K$ -dimensional space of features  $f_1 \times f_2 \times \dots \times f_K$ . The key idea is that features belonging to the same attractor cluster together in state space, so that each attractor forms a distinct cluster (a cloud of points) in feature space. The final step in the method is to therefore cluster the features. The clustering algorithm chosen for this is the Density-based Spatial Clustering of Applications with Noise (DBSCAN).<sup>30</sup> It first classifies two points as neighbors if their distance is smaller than a radius  $\epsilon$ . Then, it clusters together points with many neighbors (equal or more than a parameter  $\text{minPts}$ ), and leave as outliers points with too few neighbors (less than  $\text{minPts}$ ). The radius  $\epsilon$  is a crucial parameter for the algorithm and often needs fine tuning for proper clustering. The methods by Refs. 28 and 29 use two different ways to identify a value for  $\epsilon$ . Authors in Ref. 28



TABLE II. Comparison the two main methods of finding and continuing attractors.

Aspect	Recurrences-based attractors find and match (RAFM)	Featurize and group across parameter (FGAP)
Valid systems	Dynamical systems whose attractors satisfy the Poincaré recurrence theorem, irrespective of their basin structure.	Anything, including stochastic systems, provided the user has found features that can distinguish different “attractors” (which could be actual attractors or something similar in the case of stochastic systems).
Accuracy	Highly accurate: finds actual attractors using their unique property of their state space location.	Less accurate, as trajectories are transformed into features, and attractors correspond to groups of features. The correspondence is not guaranteed to be unique or reversible.
Info stored	Stores samples of points on the found attractors.	Stores a user-provided function of the group of features (by default: the centroid of the group).
Speed	Very fast for low-dimensional systems and for systems whose attractors are fixed points or periodic orbits. Becomes slow for attractors with long recurrence times, such as high-dimensional systems ( $\gtrsim 80$ -D) with chaotic attractors or very fine state space tessellations.	Performance is independent of system attractors. It linearly scales with the number of features, the amount of initial conditions, the transient integration time, and the total integration time. Parallelizable. In addition is the cost of the grouping process, which is huge for clustering but trivial for histograms or nearest feature. See also the benchmark comparison in Sec. IV.
Memory	Memory allocation scales as $(1/\varepsilon)^\Delta$ , with $\varepsilon$ the state space tessellation size and $\Delta$ the capacity dimension of the attractor, which is often much lower than the state space dimension. <sup>20</sup>	Memory allocation of the trajectories scales linearly with integration time and sampling rate. Additionally, specifically for clustering used as the grouping mechanism, the total memory allocated is proportional to the square of (initial conditions $\times$ parameter values) which, if one attempts to obtain accurate results, is often beyond the available memory on a typical computer (in the software implementation, we offer the possibility of an on-disk allocation in this case).
Necessary input (guesswork)	A state space box that may contain the attractors and a state space tessellation that is fine enough to differentiate the location of attractors.	A state space box that may contain the attractors; a function mapping attractors to features, such that different attractors produce different features; how much transient time to discard from time evolution; how much time to evolve and record the trajectory for, after transient.
Meta-parameters	The parameters of the finite state machine of the recurrences algorithm <sup>31</sup> and the time stepping $\Delta t$ . All are crucial, but all are conceptually straightforward.	The integration time, sampling rate, and all parameters of the grouping procedure (such as those for DBSCAN or the histogram bin in feature space). Integration parameters are straightforward, but optimal parameters related to the grouping are much harder to guess.
Trouble-shooting	Easy to troubleshoot. At any point, the actual trajectories and attractors are accessible and why a failure occurs is typically easy to find out. Matching of attractors happens parameter-by-parameter; hence, individual parameter slices can be isolated and analyzed to identify where matching has failed and why (distances between attractors is also provided information).	Difficult to troubleshoot. It does not find the actual system attractors, so the user must always reason in terms of features. When grouping using clustering (DBSCAN), the grouping process essentially operates as a black box after the features have been computed, making it harder to comprehend failures. Matching of attractors happens at the same time as grouping, making it nearly impossible to understand why an expected matching failed during the continuation process.
Failures	Algorithm may fail if state space tessellation is not fine enough and a grid cell may contain points from different attractors. Chaotic saddles and other sticky sets generate very long transients <sup>44</sup> and the algorithm can interpret them as attractors. Additionally, the algorithm is sensitive to the time step $\Delta t$ and the used integrator. For limit cycles, it is often the case that a non-adaptive integrator needs to be used.	Sticky sets that are formally not attractors will be interpreted as attractors. Additionally, for ill-defined features, any group of trajectories could be interpreted as an attractor, which is not desirable in the context of this paper. Clustering via DBSCAN may fail unexpectedly or finding the optimal radius for the clustering may yield incorrect results.

use a method that looks at the ordered distance of the  $k$  nearest neighbors to each point in the dataset and finds the first knee (high derivative point).<sup>30,45</sup> Authors in Ref. 29 iteratively search for the  $\varepsilon$  that maximizes a criterion of clustering quality. To calculate this criterion, they evaluate the silhouette of each cluster, which measures how similar each point is to the cluster it currently belongs to, compared to the other clusters, and ranges from  $-1$  (worst matching) to  $+1$  (ideal matching). This leads to one silhouette value per feature; the authors then take the minimum value as the representative for the clustering quality for each radius. The chosen radius is, thus, the value that leads to the highest minimum silhouette. In both methods, the clusters found by DBSCAN are considered then as attractors.

## B. Recurrence-based method for finding attractors

The inputs to this method are a dynamical system, a state space box that may contain the attractors (although initially it may be arbitrarily large) and a tessellation of the given box into cells. An initial condition of the system is evolved step-by-step with time step  $\Delta t$ . At each step, the location of the trajectory in state space is mapped to its cell and that the cell is labeled as visited. If the dynamical system has attractors and they satisfy the Poincaré recurrence theorem (Chap. 9 of 20), the trajectory is guaranteed to revisit cells it has visited before. Once a pre-decided number of recurrences  $n_f$  have been accumulated consecutively (i.e., enough previously visited cells are visited again), the method claims to have found an attractor. It then proceeds to locate the attractor accurately, by collecting a pre-decided number  $n_l$  of points on the attractor [Fig. 1 of Datsoris and Wagemakers<sup>31</sup> and panel (3) of Fig. 1]. A finite state machine formulation keeps track of coexisting attractors, so that each attractor is unique. It also keeps track of divergence to infinity by counting steps  $n_d$  outside the box, ensures algorithm termination by setting a total  $n_m$  of max amount of  $\Delta t$  iterations, and makes convergence faster by utilizing information already encoded in the grid: if the trajectory visits consecutively a relatively small number  $n_r$  of cells already labeled as an attractor, convergence is already eagerly decided, i.e., converging to an already found attractor is much faster than finding that attractor for the first time. Hence,  $\Delta t$ ,  $n_f$ ,  $n_l$ ,  $n_d$ ,  $n_m$ , and  $n_r$  are the meta-parameters of the algorithm and have sensible default values that work in most cases. More information on the method can be found in Ref. 31. Notice that the recurrence method is different at a fundamental level from the Global Analysis of Invariant Objects (GAIO)<sup>46</sup> and other cell mapping techniques.<sup>47</sup> We expand more on this in Sec. IV. Also, note that the method is not perfect; it may identify two attractors when only one exists due to, e.g., choosing too low convergence criteria  $n_l$ ,  $n_f$ , or due to a commensurate period of attractor and integrator time step. However, once again the importance of finding the “actual” attractors becomes apparent: further analysis by, e.g., plotting the attractors, immediately highlights such a failure and how to deal with it, and we also provide several tips in the documentation of our method in the code implementation.<sup>39</sup>

## C. Improvements to the recurrences method

A large drawback of the recurrences method was that it scaled poorly with the dimension  $D$  of the dynamical system. If an  $\varepsilon$ -sized tessellation of the state space is chosen, then memory allocated

scaled as  $1/\varepsilon^D$ . We now use sparse arrays to store accessed grid locations. This changes the memory scaling to  $1/\varepsilon^\Delta$ , with  $\Delta$  the capacity dimension<sup>30</sup> of the attractor, which is typically much smaller than  $D$  (and is only 1 for limit cycles).

## D. Improvements to the featurizing methods

First, we have changed the criterion used for finding the optimal radius in the clustering method. We have found the knee method consistently more unreliable than the iterative search. We have also found that the mean, instead of the minimum, silhouette value as the measure of clustering quality leads to better clustering. For instance, this leads to correct clustering in the Lorenz86 system, whereas the minimum value criterion did not. Furthermore, our method searches for the optimal radius with a bisection method, instead of the linear method used by authors in Ref. 29. This significantly speeds up the code. Another simple modification we introduced is to rescale the features in each dimension ( $f_1, f_2, \dots, f_K$ ) into the same interval, for instance,  $[0, 1]$ . We noticed that the clustering method performs poorly if the features span different ranges of values, and this simple modification proved to be a very powerful solution. Third, we allow the integration of all initial conditions to be done in parallel, using several computer cores, which speeds up the solution. Last, grouping in our framework can also happen based on a histogram in feature space.

## E. Code implementation

The code implementation of our framework is part of the DynamicalSystems.jl library<sup>32</sup> as the Attractors.jl package.<sup>39</sup> The code is open-source code for the Julia language, has been developed following best practices in scientific code,<sup>48</sup> is tested extensively, and is accompanied by a high quality documentation. An example code snippet is shown in Listing 1.

In addition to the quality of the implementation, three more features of the code are noteworthy. First, that it is part of DynamicalSystems.jl instead of an isolated piece of code. This integration makes the simplicity and high-levelness of Listing 1 possible and makes the input for the code easy to set up. Moreover, the direct output of the code can be used with the rest of the library to further analyze attractors in terms of, e.g., Lyapunov exponents or fractal dimensions. Indeed, in the provided code example Listing 1, we compute the Lyapunov spectra of all found attractors, across all parameter values, in only two additional lines of code. Second, utilizing the Julia language’s multiple dispatch system,<sup>49</sup> the code is extendable. It establishes one interface for how to map initial conditions to attractors and one for how to group features, both of which can be extended, and yet readily be usable by the rest of the library such as the continuation methods. Third, a lot of attention has been put into user experience, by establishing a short learning curve via a minimal user interface, by carefully considering how to provide the output in an intuitive format, as well as providing easy-to-use plotting functions that utilize the code output. More overview and information on the code or its design can be found in its online documentation or source code.<sup>39</sup>

### F. Simulated systems

In this section, we state the dynamic rules (equations of motion), parameter values, and state space boxes used for all systems in the main text.

#### 1. Chaotic Lorenz84

This model due to Lorenz<sup>33</sup> is an extremely simplified representation of atmospheric flow as a low-dimensional dynamical system with equations,

$$\begin{aligned} \dot{x} &= -y^2 - z^2 - ax + aF, \\ \dot{y} &= xy - y - bxz + G, \\ \dot{z} &= bxy + xz - z, \end{aligned}$$

and parameters  $F = 6.846, a = 0.25, b = 4.0$ , and  $G$  ranging from 1.34 to 1.37. The state space box and tessellation was from  $-3$  to  $+3$  discretized into 600 points in each dimension.

#### 2. Climate toy model

The high-dimensional toy model of global climate is due to Gelbrecht *et al.*,<sup>34</sup>

$$\begin{aligned} \dot{X}_n &= (X_{n+1} - X_{n-2})X_{n-1} - X_n + F \left( 1 + \beta \frac{T - \bar{T}}{\Delta T} \right), \\ n &= 1, \dots, N; \quad n \pm N \equiv i, \end{aligned}$$

#### 4. Turbulent flow

We reproduce here the equations resulting of a Galerkin projection of a stream function of a fluid limited to a finite cell volume, describing a low-dimensional turbulent shear flow model by<sup>36</sup>

$$\begin{aligned} \frac{da_1}{dt} &= \frac{\beta^2}{Re} - \frac{\beta^2}{Re} a_1 - \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}} a_6 a_8 + \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\beta\gamma}} a_2 a_3, \\ \frac{da_2}{dt} &= - \left( \frac{4\beta^2}{3} + \gamma^2 \right) \frac{a_2}{Re} + \frac{5\sqrt{2}\gamma^2}{3\sqrt{3}\kappa_{\alpha\gamma}} a_4 a_6 - \frac{\gamma^2}{\sqrt{6}\kappa_{\alpha\gamma}} a_5 a_7 - \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}} a_5 a_8 - \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\beta\gamma}} a_1 a_3 - \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\beta\gamma}} a_3 a_9, \\ \frac{da_3}{dt} &= - \frac{\beta^2 + \gamma^2}{Re} a_3 + \frac{2}{\sqrt{6}} \frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}} (a_4 a_7 + a_5 a_6) + \frac{\beta^2(3\alpha^2 + \gamma^2) - 3\gamma^2(\alpha^2 + \gamma^2)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}} a_4 a_8, \\ \frac{da_4}{dt} &= - \frac{3\alpha^2 + 4\beta^2}{3Re} a_4 - \frac{\alpha}{\sqrt{6}} a_1 a_5 - \frac{10\alpha^2}{3\sqrt{6}\kappa_{\alpha\gamma}} a_2 a_6 - \sqrt{\frac{3}{2}} \frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}} a_3 a_7 - \sqrt{\frac{3}{2}} \frac{\alpha^2\beta^2}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}} a_3 a_8 - \frac{\alpha}{\sqrt{6}} a_5 a_6, \\ \frac{da_5}{dt} &= - \frac{\alpha^2 + \beta^2}{Re} a_5 + \frac{\alpha}{\sqrt{6}} a_1 a_4 + \frac{\alpha^2}{\sqrt{6}\kappa_{\alpha\gamma}} a_2 a_7 - \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}} a_2 a_8 + \frac{\alpha}{\sqrt{6}} a_4 a_9 + \frac{2\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}} a_3 a_6, \\ \frac{da_6}{dt} &= - \frac{3\alpha^2 + 4\beta^2 + 3\gamma^2}{3Re} a_6 + \frac{\alpha}{\sqrt{6}} a_1 a_7 + \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}} a_1 a_8 + \frac{10(\alpha^2 - \gamma^2)}{3\sqrt{6}\kappa_{\alpha\gamma}} a_2 a_4 - 2\sqrt{\frac{2}{3}} \frac{\alpha\beta\gamma}{\kappa_{\alpha\gamma}\kappa_{\beta\gamma}} a_3 a_5 + \frac{\alpha}{\sqrt{6}} a_7 a_9 + \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}} a_8 a_9, \\ \frac{da_7}{dt} &= - \frac{\alpha^2 + \beta^2 + \gamma^2}{Re} a_7 - \frac{\alpha}{\sqrt{6}} (a_1 a_6 + a_6 a_9) + \frac{\gamma^2 - \alpha^2}{\sqrt{6}\kappa_{\alpha\gamma}} a_2 a_5 + \frac{\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}} a_3 a_4, \end{aligned}$$

$$\dot{T} = S \left( 1 - a_0 + \frac{a_1}{2} \tanh(T - \bar{T}) \right) - \sigma T^4 - \alpha \left( \frac{\mathcal{E}(X)}{0.6F^{\frac{4}{3}}} - 1 \right),$$

$$\mathcal{E}(X) = \frac{1}{2N} \sum_{n=1}^N X_n^2,$$

with parameter values identical to those in Table 1 of Ref. 34; however, we used  $N = 32$   $X$  variables. The parameter we varied was the solar constant  $S$  from 5 to 19. The initial dynamical system above was transformed to a projected dynamical system to the space of  $T, \mathcal{E}$  and  $M = \sum_n X_n/N$ , as also done in Ref. 34. In this projected space, the box and tessellation we used was from  $-2$  to  $10$  for  $M$ ,  $0$  to  $50$  for  $\mathcal{E}$ , and  $230$  to  $350$  for  $T$  with 101 points in each dimension.

#### 3. Cell genotypes

The cell differentiation model MultiFate proposed in Ref. 13 is given by

$$\dot{A}_i = \alpha + \beta \frac{B_i^n}{1 + B_i^n} - A_i, \quad i = 1, 2, 3,$$

with, for each  $i$ ,

$$B_i = \frac{2A_i^2}{K_d + 4(A_1 + A_2 + A_3) + \sqrt{K_d^2 + 8(A_1 + A_2 + A_3)K_d}},$$

with parameters  $\alpha = 0.8, \beta = 20, K_d = 1, n = 1.5$ . The state space grid ranged the interval  $[0, 100]$  for all 3 dimensions, with 100 grid cells per dimension.

$$\begin{aligned} \frac{da_8}{dt} &= -\frac{\alpha^2 + \beta^2 + \gamma^2}{Re} a_8 + \frac{2\alpha\beta\gamma}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\alpha\beta\gamma}} a_2 a_5 + \frac{\gamma^2(3\alpha^2 - \beta^2 + 3\gamma^2)}{\sqrt{6}\kappa_{\alpha\gamma}\kappa_{\beta\gamma}\kappa_{\alpha\beta\gamma}} a_3 a_4, \\ \frac{da_9}{dt} &= -\frac{9\beta^2}{Re} a_9 + \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\beta\gamma}} a_2 a_3 - \sqrt{\frac{3}{2}} \frac{\beta\gamma}{\kappa_{\alpha\beta\gamma}} a_6 a_8, \\ \kappa_{\alpha\gamma} &= \sqrt{\alpha^2 + \gamma^2}, \\ \kappa_{\beta\gamma} &= \sqrt{\beta^2 + \gamma^2}, \\ \kappa_{\alpha\beta\gamma} &= \sqrt{\alpha^2 + \beta^2 + \gamma^2}. \end{aligned}$$

Global parameters are  $L_x = 1.75\pi$ ,  $L_z = 1.2\pi$ ,  $\alpha = 2\pi/L_x$ ; and  $\beta = \pi/2$ ,  $\gamma = 2\pi/L_z$ .

### 5. Ecosystem dynamics

The population dynamics model of competing species is described by the following equations<sup>37</sup> for  $n$  species and three resources:

$$\begin{aligned} \dot{N}_i &= N_i[\mu_i(R_1, R_2, R_3) - m], \quad i = 1, \dots, n, \\ \dot{R}_j &= D(S - R_j) - \sum_{i=1}^n c_{ji}\mu_i(R_1, R_2, R_3)N_i, \quad j = 1, 2, 3. \end{aligned}$$

The term  $\mu_i(R_1, R_2, R_3)$  is given by, for each  $i = 1, \dots, n$ ,

$$\mu_i(R_1, R_2, R_3) = \min\left(\frac{rR_1}{K_{1i} + R_1}, \frac{rR_2}{K_{2i} + R_2}, \frac{rR_3}{K_{3i} + R_3}\right).$$

The parameters used in the figure are:  $n = 5$ ,  $m = 0.25$ ,  $S = 10$ ,  $r = 1.0$ ,

$$K = \begin{bmatrix} 0.20 & 0.05 & 1.00 & 0.05 & 1.20 \\ 0.25 & 0.10 & 0.05 & 1.00 & 0.40 \\ 0.15 & 0.95 & 0.35 & 0.10 & 0.05 \end{bmatrix},$$

$$c = \begin{bmatrix} 0.20 & 0.10 & 0.10 & 0.10 & 0.10 \\ 0.10 & 0.20 & 0.10 & 0.10 & 0.20 \\ 0.10 & 0.10 & 0.20 & 0.20 & 0.10 \end{bmatrix},$$

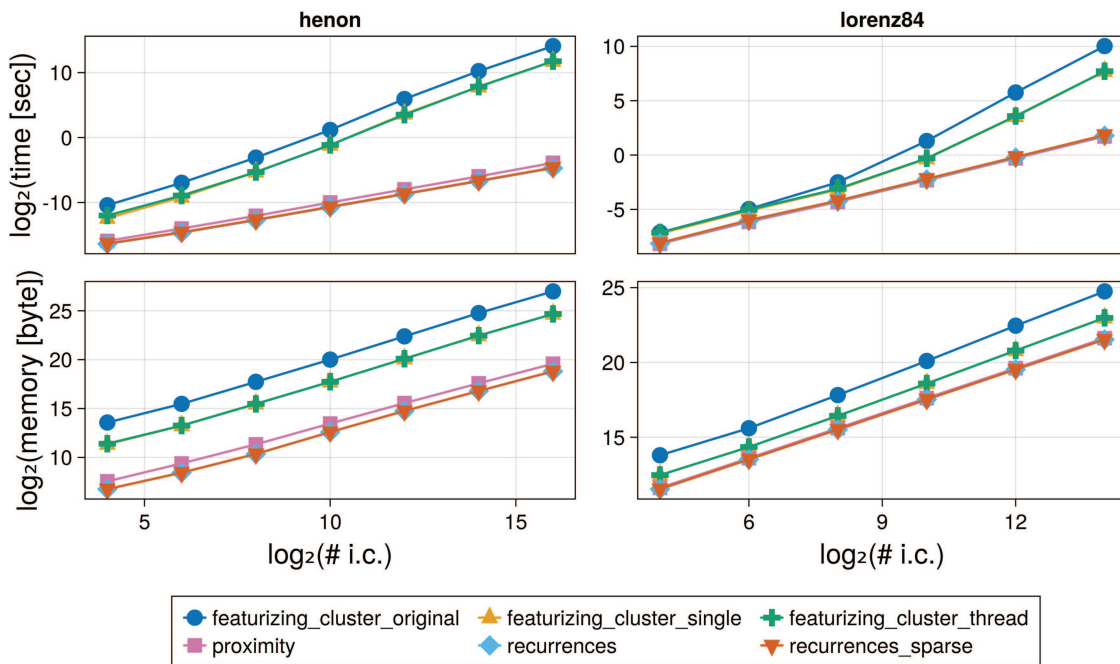


FIG. 5. Benchmark comparison between all methods for finding attractors and their basins in DynamicalSystems.jl. Note that the featurizing method scales quadratically with the number of initial conditions, because the DBSCAN algorithm scales quadratically.

and  $D$  is given in the axis of the figure. The grid in state space was defined in the interval  $[0, 60]$  for all dimensions and discretized to include 300 grid squares. The initial conditions in a grid in the same interval, but with only two grid squares to dimension, rendering  $2^8$  initial conditions for the eight-dimensional state space.

## 6. Hénon map

A simple yet famous two-dimensional discrete time map with constant Jacobian due to Hénon

$$\begin{aligned}x_{n+1} &= 1 - ax_n^2 + y_n, \\ y_{n+1} &= bx_n,\end{aligned}$$

with  $b = 0.3$  and  $a$  ranging from 1.2 to 1.25. We used the state space box from  $-2.5$  to  $2.5$  with 500 cell points in each dimension.

## 7. Second-order Kuramoto oscillators on networks

This model of coupled oscillators reproduces the dynamics of synchronous generators coupled over a power grid. The equations of the nodes are

$$\begin{aligned}\dot{\phi}_n &= \omega_n, \\ \dot{\omega}_n &= \pm 1 - 0.1\omega - K \sum_j A_{ij} \sin(\Phi_i - \Phi_j),\end{aligned}$$

where  $\phi_n$  and  $\omega_n$  are the phase and the frequency of the oscillator  $n$ . The adjacency matrix  $A_{ij}$  contains all the information about the coupling of the system and is taken from a random regular graph of degree 3. The leading coefficient of the second equation is 1 when  $n$  is odd and  $-1$  otherwise.  $K$  is the coupling coefficient between oscillators that is used as a parameter for the study of the basins fractions. The panel (d) of Fig. 3 of the article has been processed such that basins with less than 4% of the basins fractions are aggregated into the cluster called “outliers.”

## 8. Kuramoto coupled oscillators on networks

This is the classical network of  $N$  phase oscillator with a global coupling,

$$\dot{\phi}_n = \omega_n - \frac{K}{N} \sum_j \sin(\phi_i - \phi_j).$$

Frequencies of the individual oscillators  $\omega_n$  are spread evenly within the interval  $[-1, 1]$ .

```
1 using DynamicalSystems # our framework implementation
2 using OrdinaryDiffEq # high-accuracy ODE solvers
3
4 # create Lorenz84 within DynamicalSystems.jl
5 function lorenz84_rule(u, p, t)
6     F, G, a, b = p
7     x, y, z = u
8     dx = -y^2 - z^2 - a*x + a*F
9     dy = x*y - y - b*x*z + G
10    dz = b*x*y + x*z - z
11    return SVector(dx, dy, dz)
12 end
13 u0 = ones(3) # init. state
14 p0 = [6.886, 1.347, 0.255, 4.0] # init. parameters
15 # ODE solver:
16 diffeq = (alg = Vern9(), )
17 # Main object of the library: a 'DynamicalSystem'
18 ds = CoupledODEs(lorenz84_rule, u0, p0; diffeq)
19
20 # Provide state space box tessellation to search in
21 xg = yg = zg = range(-3, 3; length = 600)
22 grid = (xg, yg, zg)
23 # initialize recurrences-based algorithm
24 # and choose its metaparameters
25 mapper = AttractorsViaRecurrences(ds, grid;
26     mx_chk_fnd_att = 1000, mx_chk_loc_att = 2000,
27     mx_chk_lost = 100, mx_chk_safety = 1e8,
28     Dt = 0.05, force_non_adaptive = true,
29 )
30
31 # find and continue attractors across a given
32 # parameter range for the 'pidx'-th parameter
33 prange = range(1.34, 1.37; length = 101)
34 pidx = 2 # index of parameter
35 sampler = statespace_sampler(grid)[1]
36 rsc = RAFM(mapper)
37 # main output:
38 fractions_cont, attractors_cont = continuation(
39     rsc, prange, pidx, sampler
40 )
41
42 # Estimate Lyapunov spectra for all attractors
43 # by looping over the parameter range
44 lyapunovs_curves = map(eachindex(prange)) do index
45     set_parameter!(ds, pidx, prange[index])
46     attractor_dict = attractors_info[index]
47     exponents = Dict{
48         id => lyapunovspectrum(ds, 10000; u0 = A[1])
49         for (id, A) in attractor_dict
50     }
51 end
```

Listing 1: Julia code snippet showcasing the usage of the DynamicalSystems.jl implementation of our framework. The code produces panel (a) of Fig. 2. The main output of the code are two vectors, containing the basins fractions and attractors at each parameter value, respectively. The fractions and attractors are formulated as dictionaries, mapping attractor labels (the integers) to basin fractions and sets of points on the attractor, respectively. At its end, the code snippet computes the Lyapunov spectra of all found attractors by using the first point on each attractor as initial condition for the computation of the Lyapunov spectrum.

## G. Computational performance comparison

The benchmarks presented in Fig. 5 provide a comparison between techniques for finding attractors for a discrete and continuous dynamical system.

## H. Comparison with GAIO and cell mapping techniques

The numerical approximations of the attractors and their basins of attraction can be achieved with other well known numerical tools. The cell mapping and the GAIO algorithms rely on a subdivision of the state space. In its simplest form,<sup>47</sup> the cell mapping technique transforms the dynamical system into a discrete mapping. Each cell of the state space is mapped to another cell following the dynamics during a fixed time  $T$ . The new representation of the dynamical system is a directed graph where each node represents an initial condition on the tessellated phase space and a single edge starts from this node to another one in the graph. Once the full mapping has been obtained, efficient search algorithms inspired from graph theory approximate the basins and the attractors. The computational effort is centered around the construction of the mapping and depends directly on the discretization of the state space. The computational complexity explodes with the system dimension and, hence, limits its practical use to low-dimensional systems.

The Global Analysis of Invariant Objects (GAIO) algorithm<sup>50</sup> is also based on the discretization on a region but only a subset of cells is subdivided following the dynamics of the system. An iterative process allows us to approximate accurately the attractor manifold and also other invariant sets embedded in the state space. The complexity only depends on the dimension of the manifold not on the dimension of the state space, which is a considerable gain over the cell mapping. Since the method is focused on global attracting sets, it is not usable for multistable systems, which are the main focus of our work.

These two techniques are hard to apply in estimating the basins fractions across a parameter value. The cell mapping technique requires a full description of the state space for each parameter, making the random sampling of the phase space impossible. The GAIO technique can continue a global attractor but will fail if two or more exist at any parameter value.<sup>46</sup>

We should mention another algorithm using interval arithmetic for continuation of the dynamics.<sup>51</sup> The continuation is performed in a database space using Conley indices and Morse decomposition. Although very effective for low-dimensional maps, it lacks the flexibility and ease of use of our proposed method.

## ACKNOWLEDGMENTS

The authors would like to thank Ulrike Feudel, Peter Ashwin, Ulrich Parlitz, and Harry Dankowicz for helpful discussions. G.D. was supported by the Royal Society International Newton Fellowship. K.L.R. was supported by the German Academic Exchange Service (DAAD). A.W. was supported by the Spanish State Research Agency (AEI) and the European Regional Development Fund (ERDF) under Project No. PID2019-105554GB-I00 (MCIN/AEI/10.13039/501100011033).

## AUTHOR DECLARATIONS

### Conflict of Interest

The authors have no conflicts to disclose.

## Author Contributions

G.D. coordinated the project. All authors contributed in compiling the results, creating the code, writing, and revising the manuscript.

**George Datsis:** Conceptualization (lead); Formal analysis (equal); Software (equal); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Kalel Luiz Rossi:** Conceptualization (equal); Formal analysis (equal); Software (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Alexandre Wagemakers:** Conceptualization (equal); Formal analysis (equal); Software (equal); Writing – original draft (equal); Writing – review & editing (equal).

## DATA AVAILABILITY

The code we used to create the figures of this article is fully reproducible and available online, Ref. 52.

## REFERENCES

- <sup>1</sup>U. Feudel, A. N. Pisarchik, and K. Showalter, “Multistability and tipping: From mathematics and physics to climate and brain—Minireview and preface to the focus issue,” *Chaos* **28**, 33501 (2018).
- <sup>2</sup>A. N. Pisarchik and A. E. Hramov, *Multistability in Physical and Living Systems* (Springer International Publishing, 2022).
- <sup>3</sup>F. Hellmann, P. Schultz, P. Jaros, R. Levchenko, T. Kapitaniak, J. Kurths, and Y. Maistrenko, “Network-induced multistability through lossy coupling and exotic solitary states,” *Nat. Commun.* **11**, 592 (2020).
- <sup>4</sup>H. Kim, S. H. Lee, J. Davidsen, and S.-W. Son, “Multistability and variations in basin of attraction in power-grid systems,” *New J. Phys.* **20**, 113006 (2018).
- <sup>5</sup>L. Halekotte, A. Vanselow, and U. Feudel, “Transient chaos enforces uncertainty in the British power grid,” *J. Phys.: Complex.* **2**, 035015 (2021).
- <sup>6</sup>J. Marotzke, P. Welander, and J. Willebrand, “Instability and multiple steady states in a meridional-plane model of the thermohaline circulation,” *Tellus A* **40**, 162–172 (2016).
- <sup>7</sup>T. M. Lenton, “Environmental tipping points,” *Annu. Rev. Environ. Resour.* **38**, 1–29 (2013).
- <sup>8</sup>M. Hirota, M. Holmgren, E. H. V. Nes, and M. Scheffer, “Global resilience of tropical forest and savanna to critical transitions,” *Science* **334**, 232–235 (2011).
- <sup>9</sup>V. Dakos, B. Matthews, A. P. Hendry, J. Levine, N. Loeuille, J. Norberg, P. Nosil, M. Scheffer, and L. D. Meester, “Ecosystem tipping points in an evolving world,” *Nat. Ecol. Evol.* **3**, 355–362 (2019).
- <sup>10</sup>J.-L. Schwartz, N. Grimault, J.-M. Hupé, B. C. J. Moore, and D. Pressnitzer, “Multistability in perception: Binding sensory modalities, an overview,” *Philos. Trans. R. Soc. B: Biol. Sci.* **367**, 896–905 (2012).
- <sup>11</sup>J. A. S. Kelso, “Multistability and metastability: Understanding dynamic coordination in the brain,” *Philos. Trans. R. Soc. B: Biol. Sci.* **367**, 906–918 (2012).
- <sup>12</sup>A. Kleinschmidt, P. Sterzer, and G. Rees, “Variability of perceptual multistability: From brain state to individual trait,” *Philos. Trans. R. Soc. B: Biol. Sci.* **367**, 988–1000 (2012).
- <sup>13</sup>R. Zhu, J. M. D. Rio-Salgado, J. Garcia-Ojalvo, and M. B. Elowitz, “Synthetic multistability in mammalian cells,” *Science* **375**, eabg9765 (2022).
- <sup>14</sup>T. Khazaei, R. L. Williams, S. R. Bogatyrev, J. C. Doyle, C. S. Henry, and R. F. Ismagilov, “Metabolic multistability and hysteresis in a model aerobic-anaerobic microbiome community,” *Sci. Adv.* **6**, eaba0353 (2020).
- <sup>15</sup>C. Geiß, E. Salas, J. Guevara-Coto, A. Régner-Vigouroux, and R. A. Mora-Rodríguez, “Multistability in macrophage activation pathways and metabolic implications,” *Cells* **11**, 404 (2022).
- <sup>16</sup>K. Padiyar, *Power System Dynamics: Stability and Control* (Wiley, 1999).
- <sup>17</sup>J. Lohmann and P. D. Ditlevsen, “Risk of tipping the overturning circulation due to increasing rates of ice melt,” *Proc. Natl. Acad. Sci. U. S. A.* **118**, e2017989118 (2021).
- <sup>18</sup>P. Ashwin, S. Wieczorek, R. Vitolo, and P. Cox, “Tipping points in open systems: Bifurcation, noise-induced and rate-dependent examples in the climate

- system,” *Philos. Trans. R. Soc. A: Math., Phys. Eng. Sci.* **370**, 1166–1184 (2012). [arXiv:1103.0169](https://arxiv.org/abs/1103.0169).
- <sup>19</sup>L. Halekotte and U. Feudel, “Minimal fatal shocks in multistable complex networks,” *Sci. Rep.* **10**, 11783 (2020).
- <sup>20</sup>G. Datsoris and U. Parlitz, “Nonlinear dynamics,” in *Undergraduate Lecture Notes in Physics*, 1st ed. (Springer Nature, Cham, 2022).
- <sup>21</sup>E. J. Doedel, “Auto: A program for the automatic bifurcation analysis of autonomous systems,” *Congr. Numer.* **30**, 25–93 (1981).
- <sup>22</sup>A. Dhooge, W. Govaerts, and Y. A. Kuznetsov, “MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs,” *ACM Trans. Math. Softw. (TOMS)* **29**, 141–164 (2003).
- <sup>23</sup>H. Dankowicz, “Recipes for continuation,” in *Computational Science and Engineering* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013).
- <sup>24</sup>R. Veltz (2020). “BifurcationKit.jl.” HAL Open Science. <https://hal.archives-ouvertes.fr/hal-02902346>.
- <sup>25</sup>K. Pusuluri, H. Meijer, and A. Shilnikov, “Homoclinic puzzles and chaos in a nonlinear laser model,” *Commun. Nonlinear Sci. Numer. Simul.* **93**, 105503 (2021).
- <sup>26</sup>P. J. Menck, J. Heitzig, N. Marwan, and J. Kurths, “How basin stability complements the linear-stability paradigm,” *Nat. Phys.* **9**, 89–92 (2013).
- <sup>27</sup>The term “global stability” is also used when a dynamical system has a single global attractor, which is different from our use of the term here.
- <sup>28</sup>M. Gelbrecht, J. Kurths, and F. Hellmann, “Monte Carlo basin bifurcation analysis,” *New J. Phys.* **22**, 033032 (2020).
- <sup>29</sup>M. Stender and N. Hoffmann, “bSTAB: An open-source software for computing the basin stability of multi-stable dynamical systems,” *Nonlinear Dyn.* **107**, 1451–1468 (2021).
- <sup>30</sup>M. Ester, H. P. Kriegel, J. Sander, and X. Xiaowei, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Springer, 1996).
- <sup>31</sup>G. Datsoris and A. Wagemakers, “Effortless estimation of basins of attraction,” *Chaos* **32**, 023104 (2022). [arXiv:2110.04358](https://arxiv.org/abs/2110.04358).
- <sup>32</sup>G. Datsoris, “DynamicalSystems.jl: A Julia software library for chaos and non-linear dynamics,” *J. Open Source Softw.* **3**, 598 (2018).
- <sup>33</sup>E. N. Lorenz, “Irregularity: A fundamental property of the atmosphere,” *Tellus A: Dyn. Meteorol. Oceanogr.* **36**, 98–110 (1984).
- <sup>34</sup>M. Gelbrecht, V. Lucarini, N. Boers, and J. Kurths, “Analysis of a bistable climate toy model with physics-based machine learning methods,” *Eur. Phys. J.: Spec. Top.* **123**, 3121–3131 (2021).
- <sup>35</sup>S. Huang, “Multistability and multicellularity: Cell fates as high-dimensional attractors of gene regulatory networks,” in *Computational Systems Biology* (Elsevier, 2006), pp. 293–326.
- <sup>36</sup>J. Moehlis, H. Faisst, and B. Eckhardt, “A low-dimensional model for turbulent shear flows,” *New J. Phys.* **6**, 56 (2004).
- <sup>37</sup>J. Huisman and F. J. Weissing, “Fundamental unpredictability in multispecies competition,” *Am. Nat.* **157**, 488–494 (2001).
- <sup>38</sup>J. H. Verner, “Numerically optimal Runge–Kutta pairs with interpolants,” *Numer. Algorithms* **53**, 383–396 (2010).
- <sup>39</sup>G. Datsoris, K. L. Rossi, and A. Wagemakers (2023). “github.com/juliadynamics/attractors.jl: v1.2.5.” [Zenodo](https://zenodo.org/record/7811111).
- <sup>40</sup>F. Hausdorff, *Grundzuge Der Mengenlehre* (American Mathematical Society, Providence, RI, 1949).
- <sup>41</sup>P. Schultz, P. J. Menck, J. Heitzig, and J. Kurths, “Potentials and limits to basin stability estimation,” *New J. Phys.* **19**, 023005 (2017).
- <sup>42</sup>J. M. Cortes, M. Desroches, S. Rodrigues, R. Veltz, M. A. Muñoz, and T. J. Sejnowski, “Short-term synaptic plasticity in the deterministic Tsodyks–Markram model leads to unpredictable network dynamics,” *Proc. Natl. Acad. Sci. U.S.A.* **110**, 16610–16615 (2013).
- <sup>43</sup>C. Rackauckas and Q. Nie, “DifferentialEquations.jl—A performant and feature-rich ecosystem for solving differential equations in Julia,” *J. Open Res. Softw.* **5**, 15 (2017).
- <sup>44</sup>Y.-C. Lai and T. Tél, “Transient chaos: Complex dynamics on finite-time scales,” in *Applied Mathematical Sciences* (Springer, New York, 2009).
- <sup>45</sup>M. Hahsler, M. Piekenbrock, and D. Doran, “dbscan: Fast density-based clustering with R,” *J. Stat. Softw.* **91**, 1–30 (2019).
- <sup>46</sup>R. Gerlach, A. Zießler, B. Eckhardt, and M. Dellnitz, “A set-oriented path following method for the approximation of parameter dependent attractors,” *SIAM J. Appl. Dyn. Syst.* **19**, 705–723 (2020).
- <sup>47</sup>J.-Q. Sun, F.-R. Xiong, O. Schütze, and C. Hernández, *Cell Mapping Methods* (Springer, 2018).
- <sup>48</sup>G. Datsoris (2023). “Good scientific code workshop.” [Zenodo](https://zenodo.org/record/7811111).
- <sup>49</sup>J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *SIAM Rev.* **59**, 65–98 (2017).
- <sup>50</sup>M. Dellnitz, G. Froyland, and O. Junge, “The algorithms behind gaio—set oriented numerical methods for dynamical systems,” in *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems* (Springer, 2001), pp. 145–174.
- <sup>51</sup>Z. Arai, W. Kalies, H. Kokubu, K. Mischaikow, H. Oka, and P. Pilarczyk, “A database schema for the analysis of global dynamics of multiparameter systems,” *SIAM J. Appl. Dyn. Syst.* **8**, 757–789 (2009).
- <sup>52</sup>G. Datsoris (2023). “github.com/datsoris/frameworkglobalstability: v0.1,” [Zenodo](https://zenodo.org/record/7811111).