Visualization of Set-valued Dynamical Systems

Nam Do
nam.do@student.oulu.fi
University of Oulu

Jyri Ollanketo
jyri.ollanketo@student.oulu.fi
University of Oulu

Kalle Timperi
Kalle.Timperi@oulu.fi
University of Oulu

Abstract

Set-valued dynamical systems model the long-run behavior of nonlinear maps under additive bounded noise by replacing each image point with a ball of radius ε. The minimal invariant set $M(\varepsilon)$ is the exact outer limit of where the system can reach under worst-case noise of amplitude ε, and the noise level at which $M(\varepsilon)$ undergoes a sudden topological change marks how much perturbation the system can tolerate before its long-run geometry breaks down. Computing $M(\varepsilon)$ reduces to a deterministic problem via the boundary map E: the unstable manifolds of periodic orbits of E trace $\partial M(\varepsilon)$ exactly, converting an intractable stochastic problem into a classical manifold computation. We present **B**ounded **I**nvariant **S**et **T**oolbox (BIST), a browser-native visualization tool for this class of systems, built on a Rust computation core compiled to WebAssembly for near-native performance without any installation. BIST supports real-time boundary map visualization, periodic orbit detection, automated parameter sweeps for bifurcation analysis, stationary measure approximation via Ulam's method, and user-defined symbolic maps and differential equations. We demonstrate BIST on the Hénon map and Duffing oscillator, and validate the numerical results against published theoretical benchmarks and an extensive automated test suite of more than 100 tests.

Keywords

set-valued dynamical systems, invariant sets, bifurcation, bounded noise, WebAssembly, real-time visualization, Hénon map, Duffing equation, transfer operator, periodic orbits

1 Introduction

A central question in dynamical systems theory is how the qualitative behavior of a system changes when it is perturbed. The concept of *structural stability*, introduced by Andronov and Pontryagin and developed by Smale and others, captures this precisely: a system is structurally stable if small perturbations of its defining rule produce a topologically equivalent flow, in the sense that attractors, periodic orbits, and their connectivity are preserved. Uniformly hyperbolic systems enjoy this property, but most physically relevant systems do not–they sit near bifurcation points where qualitative structure, most importantly the geometry of attractors and invariant sets, can change discontinuously under parameter variation. Understanding and computing these transitions is the central problem motivating this work.

When noise is present, the structural stability question becomes: how large a perturbation can the system tolerate before its long-term geometry changes? The theory of random dynamical systems, developed by Arnold [11], provides one answer through the notion of a *random attractor*–a compact, time-varying set $A(\omega)$ indexed by the noise realization ω that attracts all trajectories in a pullback sense. This framework is powerful but requires a probability distribution on the noise. In many settings–bounded sensor error, quantized control inputs, model uncertainty–one knows only that disturbances are bounded in magnitude, with no distributional hypothesis. The natural framework for this regime is *set-valued dynamics*: rather than tracking a single orbit, one tracks the set of all states reachable under some noise realization.

Concretely, given a map $f : \mathbb{R}^d \to \mathbb{R}^d$ and noise amplitude $\varepsilon > 0$, the set-valued map $F_\varepsilon(x) = B_\varepsilon(f(x))$ captures all one-step successors of x under bounded noise. The key geometric object is the *minimal invariant set* $M(\varepsilon)$–the smallest compact set satisfying $F_\varepsilon(M) = M$. This set has three fundamental interpretations that make it worth computing: (i) it is the exact set of all states reachable from within itself under any bounded noise; any trajectory starting in the basin of attraction converges to $M(\varepsilon)$ and stays there; (ii) under mild conditions, $M(\varepsilon)$ is the support of the unique stationary invariant measure of the random system [4], so its geometry IS the long-term statistical behavior; and (iii) the critical noise amplitude ε_c at which $M(\varepsilon)$ undergoes a topological bifurcation–a discontinuous change in Hausdorff metric–is precisely the *structural stability radius* of the deterministic attractor, measuring its resilience to bounded perturbations [1–3]. In the deterministic limit $\varepsilon \to 0$, the minimal invariant set collapses to the classical attractor, which for systems like the Hénon map coincides with the closure of the unstable manifold of its saddle fixed point. For $\varepsilon > 0$, $M(\varepsilon)$ fattens around this structure, and its *boundary* $\partial M(\varepsilon)$ is the geometric object of primary interest–it records the outermost extent of the noisy system's long-term behavior.

Computing set-valued dynamics and visualizing them directly is prohibitive: iterating sets leads to exponentially growing representations. The boundary map approach of Tey [1] overcomes this by converting the intractable stochastic problem into a full deterministic one. Instead of sampling noise randomly, the method models noise as an explicit worst-case perturbation at each step: the set-valued map F_ϵ replaces every image point $f(x)$ with the closed ball $B_\epsilon(f(x))$, so the maximum noise amplitude ϵ is added directly into the dynamics. This construction gives $M(\epsilon)$ a *precise confinement guarantee*: any trajectory that starts on the boundary of $M(\epsilon)$ remains inside for all future time, therefore, the boundary is not an approximation – it is the exact outer limit of long-term reachability under the maximum noise amplitude ϵ. To compute this, one exploits the differential-geometric fact that the outward unit normal bundle of $\partial M(\varepsilon)$ is invariant under the extended boundary map E, so $\partial M(\epsilon)$ can be recovered by computing the unstable manifolds of periodic orbits of E on the space $R^d \times S^{d-1}$. For the complementary statistical picture, a GAIO-based Ulam discretization of the Perron–Frobenius operator approximates the stationary measure supported on $M(\varepsilon)$ [4, 6, 7].

In this paper we introduce BIST, an interactive browser-native tool that brings both computational strategies–boundary map iteration and Ulam's method–to researchers and students without

requiring any local software installation. It supports user-defined symbolic maps, both discrete and continuous systems, real-time parameter sweeps to locate bifurcation thresholds, and session recording. Existing tools address parts of this problem: GAIO [6] provides set-oriented numerics but requires MATLAB, and the reference Python implementation in [1] covers only the Hénon boundary map. BIST unifies and extends these capabilities in a zero-install, extensible platform accessible at https://namvdo.github.io/set-valued-viz.

2 Related Work

Set-valued dynamics and topological bifurcations. Topological bifurcations of minimal invariant sets for dynamical systems with bounded additive noise were defined and studied by Lamb, Rasmussen, and Rodrigues [3], who established conditions under which the geometry of a minimal invariant set changes discontinuously under parameter variation. A closely related earlier contribution is Homburg and Young [12], who defined and studied "hard bifurcations" in dynamical systems with bounded random perturbations – a class of qualitative transitions that corresponds to the topological bifurcations later identified in the set-valued framework. Kuehn, Malavolta, and Rasmussen [13] subsequently studied one-dimensional dynamics in this setting and identified early-warning indicators that signal the proximity of an approaching bifurcation.

Boundary map and invariant set theory. The present work is based on the concept of the boundary map, which was discovered jointly by Timperi, Tey, Lamb, and Rasmussen during Timperi's and Tey's PhD projects at Imperial College London. This concept is formally established in Kourliouros, Lamb, Rasmussen, Tey, Timperi, and Turaev [14], which introduces the boundary map and proves that, under the assumption of normal hyperbolicity, the boundaries of the minimal invariant sets persist smoothly under small perturbations of the system parameters. Tey's PhD thesis [1] builds on this concept and provides theoretical and numerical analyses of the properties of system attractors in the case of linear dynamics. The recent study by Lamb, Rasmussen, and Tey [2] gives explicit bifurcation results for the Hénon map with additive bounded noise, and directly informs our evaluation benchmarks for the discrete case.

Set-oriented numerical methods. Efficient approximation of invariant manifolds and invariant measures in dynamical systems has been developed through the set-oriented framework originating with Dellnitz and Hohmann [5], who introduced a subdivision algorithm for computing unstable manifolds and global attractors. The GAIO software package and its underlying theory were presented by Dellnitz, Froyland, and Junge [6], establishing a general framework for set-oriented computation including transition-matrix approximations of the Perron-Frobenius operator. Ulam [7] originally proposed the discretization scheme that now bears his name, and Peschke [4] extended the GAIO-based Ulam approach to the set-valued bounded-noise setting that we implement. Together, these works define the numerical toolkit for our density estimation and bifurcation detection modules.

Periodic orbit computation. Unstable periodic orbits organize the structure of chaotic attractors but are notoriously hard to find numerically. Davidchack and Lai [10] developed a stabilized Newton-type iteration that improves convergence for highly unstable saddle orbits by introducing a regularization parameter β_k that prevents divergence during the root-finding process. We implement this scheme directly in our Rust/WebAssembly backend for both standard maps and the extended boundary map.

Interactive scientific visualization. Web-based tools for interactive exploration of dynamical systems have historically been limited by the computational demands of the underlying mathematics. The advent of WebAssembly and its integration with modern JavaScript runtimes now makes it feasible to run performance-critical numerical code in the browser. Our tool contributes to this emerging direction by demonstrating that complicated set-oriented algorithms, previously limited to desktop scientific computing environments, can be made accessible through a browser-native interface.

3 Mathematical Background

3.1 Dynamical Systems with Bounded Noise

3.1.1 Discrete-Time Dynamical System. A discrete deterministic dynamical system is defined by a map

$$x_{n+1} = f(x_n), \qquad f : \mathbb{R}^d \to \mathbb{R}^d. \tag{1}$$

Throughout this work, a primary example is the Hénon map:

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

where a and b are real parameters controlling the degree of nonlinearity and area contraction, respectively.

With additive bounded noise, the state update becomes

$$x_{n+1} = f(x_n) + \xi_n, \qquad \xi_n \in W_\varepsilon, \tag{3}$$

where the noise is constrained to the closed ε-ball $W_\varepsilon := B_\varepsilon(0) = \{z \in \mathbb{R}^d : \|z\| \le \varepsilon\}$. The set of all states reachable from x in one step is

$$F_\varepsilon(x) = \{f(x) + \xi : \xi \in W_\varepsilon\} = B_\varepsilon(f(x)), \tag{4}$$

a closed ball of radius ε centered at the deterministic image $f(x)$.

Let $\mathcal{K}(\mathbb{R}^d)$ denote the family of non-empty compact subsets of $\mathbb{R}^d$, equipped with the Hausdorff metric. The induced set-valued map acts on compact sets by

$$F_\varepsilon : \mathcal{K}(\mathbb{R}^d) \to \mathcal{K}(\mathbb{R}^d), \qquad F_\varepsilon(A) = \bigcup_{x \in A} B_\varepsilon(f(x)). \tag{5}$$

This is the fundamental set-valued model used in the discrete setting [1, 2].

3.1.2 Continuous-Time Dynamical System. For continuous-time dynamics, one considers the controlled ODE

$$\dot{x}(t) = g(x(t), t) + \eta(t), \qquad \|\eta(t)\| \le \varepsilon, \tag{6}$$

where η is a measurable noise signal bounded uniformly in norm. A canonical example used in this project is the Duffing oscillator [8]:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma \cos(\omega t). \tag{7}$$

For browser-based computation, continuous flow is approximated by discretizing time with step Δt using a fourth-order Runge-Kutta

(RK4) scheme. For $\dot{x} = g(x,t)$, a single RK4 step reads

$$\begin{aligned} k_1 &= g(x_n, t_n), \\ k_2 &= g\big(x_n + \tfrac{\Delta t}{2} k_1,\ t_n + \tfrac{\Delta t}{2}\big), \\ k_3 &= g\big(x_n + \tfrac{\Delta t}{2} k_2,\ t_n + \tfrac{\Delta t}{2}\big), \\ k_4 &= g(x_n + \Delta t\, k_3,\ t_n + \Delta t), \\ x_{n+1} &= x_n + \tfrac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{8}$$

Additive bounded noise is then applied at each discrete step, giving

$$x_{n+1} \in \big\{ \Phi_{\Delta t}^{\mathrm{RK4}}(x_n) + \xi_n : \xi_n \in W_\varepsilon \big\}, \tag{9}$$

which provides a practical discrete approximation of the continuous noisy flow.

3.2 Extended Boundary Mappings

A key observation is that one need not track all points in a noisy region: since the topological boundary of the perturbed set determines its maximal extent, tracking only the boundary suffices to characterize the minimal invariant set geometry. For a diffeomorphism $f : \mathbb{R}^d \to \mathbb{R}^d$, the *boundary map* $E : \mathbb{R}^d \times S^{d-1} \to \mathbb{R}^d \times S^{d-1}$ was introduced by Kourliouros, Lamb, Rasmussen, Tey, Timperi, and Turaev [14] and is defined as

$$E(z,n) = \left(f(z) + \varepsilon \frac{J(z)^{-\top} n}{\|J(z)^{-\top} n\|},\ \frac{J(z)^{-\top} n}{\|J(z)^{-\top} n\|} \right), \tag{10}$$

where $J(z) = Df(z)$ is the Jacobian of f at z, and $J(z)^{-\top} := (J(z)^{\top})^{-1}$ denotes the inverse transpose. The geometric rationale is that outward unit normals to a smooth surface transform under f by the inverse-transpose of the Jacobian (the standard pushforward for co-vectors), so both components of E share the same normalized direction $\frac{J(z)^{-\top} n}{||J(z)^{-\top} n||}$. The outward unit normal bundle $UN^+\partial M$ of any F_ε-invariant set M with C^1 boundary is invariant under E, making periodic orbits of E the key objects for computing the minimal invariant set boundary and its unstable structures [1, 14].

3.3 Transfer Operator and GAIO-based Ulam's Method

To approximate statistical properties of the bounded-noise system, we discretize the *Perron-Frobenius transfer operator*. Assuming noise is uniform on $W_\varepsilon = B_\varepsilon(0)$ with Lebesgue measure m, the transfer operator acting on densities $\rho \in L^1(X)$ over a compact domain $X \subset \mathbb{R}^d$ is

$$(\mathcal{P}_\varepsilon \rho)(y) = \int_X k_\varepsilon(x,y)\, \rho(x)\, dx, \qquad k_\varepsilon(x,y) = \frac{\mathbf{1}_{B_\varepsilon(f(x))}(y)}{m(W_\varepsilon)}. \tag{11}$$

Equivalently, for a probability measure μ on X:

$$(\mu \mathcal{P}_\varepsilon)(A) = \int_X \frac{m(A \cap B_\varepsilon(f(x)))}{m(W_\varepsilon)}\, d\mu(x), \qquad A \subset X \text{ measurable}. \tag{12}$$

Ulam's method [7] approximates $\mathcal{P}_\varepsilon$ by partitioning X into boxes $\{B_1, \ldots, B_N\}$ and using piecewise-constant basis functions. The resulting transition matrix has entries

$$P_{ij} = \frac{1}{m(B_i)} \int_{B_i} \frac{m\big(B_j \cap B_\varepsilon(f(x))\big)}{m(W_\varepsilon)}\, dx. \tag{13}$$

Following the GAIO framework [4, 6], this integral is approximated by Monte Carlo sampling over test points $\{p_{i,\ell}\}_{\ell=1}^{M_i} \subset B_i$:

$$\widehat{P}_{ij} = \frac{1}{M_i} \sum_{\ell=1}^{M_i} \frac{m\big(B_j \cap B_\varepsilon(f(p_{i,\ell}))\big)}{m(W_\varepsilon)}. \tag{14}$$

The stationary invariant measure is then the left eigenvector π of P at eigenvalue 1:

$$\pi P = \pi, \qquad \pi_i \geq 0, \qquad \sum_{i=1}^{N} \pi_i = 1. \tag{15}$$

The right eigenstructure of P is additionally used to identify attracting and repelling regions in set-oriented bifurcation analysis [4].

3.4 Periodic Orbit Detection

A point $z^* \in \mathbb{R}^d$ is a *p-periodic point* of f if $f^p(z^*) = z^*$ and $f^m(z^*) \neq z^*$ for all proper divisors $m \mid p$, $m < p$. Finding periodic orbits reduces to solving

$$G_p(z) := f^p(z) - z = 0, \tag{16}$$

whose Jacobian is

$$DG_p(z) = Df^p(z) - I, \qquad Df^p(z) = \prod_{r=0}^{p-1} Df(f^r(z))\,. \tag{17}$$

Standard Newton iteration,

$$z^{(k+1)} = z^{(k)} - \big[DG_p(z^{(k)})\big]^{-1} G_p(z^{(k)}), \tag{18}$$

often diverges for highly unstable periodic orbits because the eigenvalues of $DG_p = Df^p - I$ grow exponentially with the instability of the orbit, making DG_p ill-conditioned. The stabilized scheme of Davidchack and Lai [10] replaces this with

$$z^{(k+1)} = z^{(k)} + \big[\beta_k \|G_p(z^{(k)})\| \cdot I - DG_p(z^{(k)})\big]^{-1} G_p(z^{(k)}), \qquad \beta_k > 0, \tag{19}$$

where the residual-scaled term $\beta_k \|G_p\| \cdot I$ regularizes the inversion away from convergence, then gracefully vanishes as $\|G_p\| \to 0$, recovering standard Newton locally. Periodic boundary points of the extended map E are found analogously by solving

$$H_p(z,n) := E^p(z,n) - (z,n) = 0 \tag{20}$$

with the same stabilized framework on the augmented state space (z,n). Stability is classified from the eigenvalues $\{\lambda_i\}$ of $Df^p(z^*)$: **stable** if all $|\lambda_i| < 1$, **saddle** if some $|\lambda_i| < 1$ and some $|\lambda_i| > 1$, and **unstable** if all $|\lambda_i| > 1$.

3.5 Topological Bifurcation

Let λ be a continuation parameter and let $M(\lambda) \in \mathcal{K}(\mathbb{R}^d)$ denote the minimal invariant set of F_λ. For compact sets $A, B \subset \mathbb{R}^d$, the Hausdorff distance is

$$d_H(A,B) = \max\left\{ \sup_{a \in A} \inf_{b \in B} \|a - b\|,\ \sup_{b \in B} \inf_{a \in A} \|a - b\| \right\}, \tag{21}$$

which metrizes $\mathcal{K}(\mathbb{R}^d)$ [9]. A *topological bifurcation* at parameter value λ_c is characterized by a discontinuity of $\lambda \mapsto M(\lambda)$ in this metric [1, 2]:

$$\limsup_{\lambda \to \lambda_c} d_H(M(\lambda), M(\lambda_c)) > 0. \tag{22}$$

The domain of attraction of $M(\lambda)$ is

$$\mathcal{A}(M(\lambda)) = \left\{ x \in \mathbb{R}^d : \operatorname{dist}\left(F_\lambda^n(\{x\}), M(\lambda)\right) \to 0 \text{ as } n \to \infty \right\}, \quad (23)$$

and the complementary repeller is $M^*(\lambda) = \mathbb{R}^d \setminus \mathcal{A}(M(\lambda))$. A necessary condition for bifurcation at λ_c is the collision

$$M(\lambda_c) \cap M^*(\lambda_c) \neq \varnothing. \quad (24)$$

In planar examples, splitting or merging events are detected additionally by monitoring changes in the connected-component count $\beta_0(M(\lambda))$, i.e., $\beta_0(M(\lambda_c^-)) \neq \beta_0(M(\lambda_c^+))$ [1–3].

4 Design

4.1 Use Cases and Requirements

The primary use case for BIST is a researcher or student who wants to explore set-valued dynamical systems with bounded noise interactively–adjusting parameters, observing how invariant set geometry changes, and identifying topological bifurcation thresholds–without installing any software beyond a modern web browser.

Existing tools address parts of this problem but leave significant gaps. GAIO [6], the standard set-oriented numerics package, requires a MATLAB license and a local installation, and is focused on invariant measure approximation rather than the geometric boundary-map approach. Tey's reference Python implementation [1] visualizes the extended boundary map, but is restricted to the Hénon map and does not support user-defined maps, continuous-time systems, or interactive parameter exploration. Neither tool provides parameter sweeping, session recording, or a zero-install deployment. BIST addresses all of these gaps with the following concrete requirements:

- **Zero-install, browser-native**: the tool runs at https://namvdo.github.io/set-valued-viz with no local setup.
- **User-defined maps**: some discrete map expressible in standard mathematical notation can be entered symbolically and analyzed immediately.
- **Discrete and continuous support**: both iterated maps and continuous ODEs (with RK4 discretization) are supported.
- **Boundary map and invariant measure**: the tool covers both the geometric (boundary map) and statistical (Ulam's method) viewpoints.
- **Parameter sweep**: automated sweeps over one or two parameters locate bifurcation-relevant regions; results can be exported for further analysis.
- **Orbit detections**: users can change the grid resolutions to detect more periodic orbits.
- **Session recording**: the evolution of the system can be captured as a video directly from the browser.

4.2 System Design

Figure 10 shows the three-layer architecture of BIST. The **Visualization Interface** layer is built with React, which manages all application state and user interaction, and Three.js, which renders the scientific viewport using WebGL for smooth display of boundary curves, orbit clouds, and density heatmaps. React's component model allows each analysis panel (unstable manifolds, periodic orbits, Ulam density, parameter sweep, animation) to be independently developed and tested.

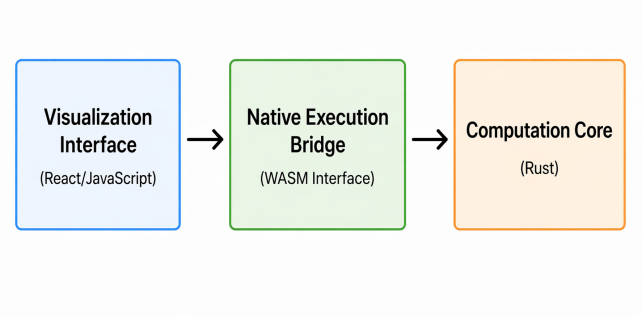

Figure 1: BIST system architecture with three layers: Visualization Interface, Native Execution Bridge and Computation Core

The **Native Execution Bridge** connects the Visualization Interface to the Computation Core via `wasm-bindgen`. All heavy numerical computations–boundary map iteration, periodic orbit solving, transition matrix assembly–are compiled to WebAssembly, which delivers near-native execution speed within the browser's sandboxed memory model. A dedicated Web Worker hosts the WASM module, so computations run on a separate thread and never block the rendering pipeline.

The **Computation Core** implements all mathematical algorithms as a native Rust library, enabling the same code to be tested natively and compiled to WASM for deployment. Performance-critical paths use `nalgebra` for linear algebra, `rayon` for parallel iteration on the native build, and `evalexpr` for symbolic expression parsing.

We also use Github Actions that will trigger tests and builds and deploy the applications to general audience online at: https://namvdo.github.io/set-valued-viz

4.3 Design Analysis

The architecture reflects three deliberate trade-offs. First, placing all numerical logic in Rust not only improves the performance – but also it allows the same code to be unit tested natively against known analytic results before being compiled to WASM, so the mathematical corectness is verified independently of the browser environment.

Second, the Web Worker isolation guarantees that long-running computations – orbit searches, adaptive boundary refinement, Ulam matrix calculations – never block the main UI thread so that the visualization interface stays responsive.

Third, the `DynamicalSystem` trait abstraction makes the system genuienly extensible: any new map, whether built-in or user-defined, plugs into the full analysis pipeline – boundary map iteration, periodic orbit detection, and stationary measure approximation – without touching the front-end or bridge layer.

The zero-install deployment compounds these benefits for the primary use case. A researcher examining a new map can open BIST, enter the symbolic expressions, and observe the minimal invariant set geometry within seconds, without configuring a Python environment or compiling a native binary. The main cost of this

approach is the browser memory ceiling and the absence of multithreading in the WASM context, which constrains the maximum grid resolution and orbit period that are practical; these trade-offs are discussed further in Section 7.1.

5 Implementation

5.1 Computation Core

The computation core is written in Rust and organized into modules that map directly onto the mathematical components described in Section 3. The `DynamicalSystem` trait is the main abstraction: it requires implementors to provide `map` and `jacobian`, and in return inherits default implementations of `transform_normal` – the inverse-transpose pushforward $n \mapsto \frac{J(z)^{-\top} n}{\|J(z)^{-\top} n\|}$ that appears in both components of the boundary map E – as well as `extended_map` and `extended_map_inverse`, which iterate E forward and backward on the augmented space $\mathbb{R}^2 \times S^1$. Any type implementing this trait is immediately usable by all higher-level solvers without modification.

Concrete implementations are provided for both discrete and continuous systems. `HenonSystem` hard-codes the analytic Jacobian and its inverse-transpose for maximum numerical efficiency. `UserDefinedDynamicalSystem` wraps a parsed symbolic expression and computes the Jacobian by centered finite differences with step $h = 10^{-5}$, accurate to $O(h^2)$ for smooth maps. For continuous flows, `DuffingSystem` models the Duffing oscillator; its time evolution is discretized using a fourth-order Runge–Kutta (RK4) integrator, converting the ODE into an iterated map that the rest of the pipeline can consume uniformly.

The remaining modules are organized by mathematical function. `unstable_manifold.rs` performs forward boundary map iteration to grow the unstable manifold from a seed orbit. `boundary_periodic.rs` runs the stabilized Davidchack–Lai solver on the 4D extended state space $(z, n) \in \mathbb{R}^2 \times S^1$ to find periodic orbits of E; these orbits seed the unstable manifold computation that traces $\partial M(\varepsilon)$. `ulam.rs` handles box partition, sparse transition matrix assembly, and stationary distribution recovery via power iteration, for both discrete maps and continuous flows. `hausdorff.rs` computes the Hausdorff distance between successive $M(\varepsilon)$ approximations, which is used to detect topological bifurcation events during parameter sweeps. On native builds, performance-critical loops in the manifold and Ulam routines are parallelized with `rayon`; the same code compiles to single-threaded WASM for browser deployment without any conditional compilation.

5.2 Generic Expression Evaluation Pipeline

Besides the built-in implementations of dynamical systems, we also want to have the software extensible enough so that it can let users enter arbitrary equations to visualize the evolution. To support this need, we create a generic pipeline that runs from the symbol input to numerical evaluation as follows:

(1) **Text input**: the user types two expressions f_x and f_y (e.g., `"1-a*x^2+y"` and `"b*x"`) and a list of named parameter values in the sidebar.
(2) **Preprocessing**: `preprocess_abs` rewrites absolute-value notation $|\cdot|$ into `abs(·)` to comply with the expression grammar, handling nested cases correctly.
(3) **Parsing**: `evalexpr::build_operator_tree` compiles each preprocessed string into an operator tree (AST). Parsing errors are returned immediately to the frontend as user-readable messages.
(4) **Context binding**: a thread-local `HashMapContext` registers the mathematical functions `sin`, `cos`, `tan`, `abs`, `sqrt`, `exp`, `ln` and exposes the current values of x, y, and all named parameters as named variables.
(5) **Evaluation**: `ParsedEquations::eval(x,y)` binds x and y in the context, evaluates both ASTs, and returns $(f_x, f_y) \in \mathbb{R}^2$.
(6) **Integration**: the resulting `UserDefinedDynamicalSystem` implements the `DynamicalSystem` trait, so it is passed directly into the same boundary-map, periodic-orbit, and Ulam routines used for built-in systems.

This design means that adding support for a new built-in map (e.g., the Lozi map) requires only registering a new `DynamicalSystem` implementor; the entire visualization and analysis pipeline is inherited automatically.

5.3 Native Execution Bridge

The Rust computation core is compiled into WebAssembly (WASM) – a binary instruction format that browse can execute directly at near-native speed, without any server round-trip. The compilation pipeline works as follows: the Rust source is compiled via `wasm-bindgen`, which generates both a `.wasm` binary and a thin JavaScript layer that exposes the Rust functions as callable JavaScrip APIs. When the page loads, the browser fetches and instantiates the `.wasm` module into a sandbox linear memory space. Data is passed across the boundary by serializing Rust structs to JSON using `serde-wasm-bindgen`, transferring the bytes through shared memory, and deserializing on the JavaScript side.

To prevent the heavy numerical computations that could freeze the visualization interface such as orbit detections, boundary map iterations, unstable manifold calculations – the WASM module is loaded inside a Web Worker, a seperate background thread provided by the web browser. The React frontend communicates with the worker via message passing: it posts a computation request, the worker runs the WASM function, and posts the result back, at which point Three.js re-renders the visualization.

5.4 Visualization Interface

The visualization interface is divided into two components: the main viewport and the sidebar. The main viewport is rendered by Three.js via WebGL and displays all geometric and statistical outputs of the analysis — boundary curves tracing $\partial M(\varepsilon)$, unstable manifold segments, periodic orbit clouds, and Ulam density heatmaps — updated in real time as parameters change. The viewport also supports animated playback of system evolution and can record the session directly to video from the browser.

The sidebar is organized into collapsible panels, each dedicated to one aspect of the analysis. *SystemPicker* lets the user select between built-in systems (Hénon map, Duffing oscillator) or enter

a custom map. *EquationDisplay* provides a symbolic input field where the user types the two component expressions f_x and f_y; *ParametersPanel* exposes sliders and text inputs for system parameters (a, b, ε, and any user-defined parameters), with support for named presets. *UnstableManifoldPanel* controls the boundary map computation — seed orbit selection, iteration depth, and refinement level. *PeriodicOrbitsPanel* configures the grid resolution and maximum orbit period for the Davidchack–Lai search. *UlamPanel* sets the box partition resolution for stationary measure approximation. *ParameterSweepPanel* defines the sweep range and step size for automated bifurcation scans, with export support. Together these panels give the user full control over every layer of the analysis without leaving the browser.

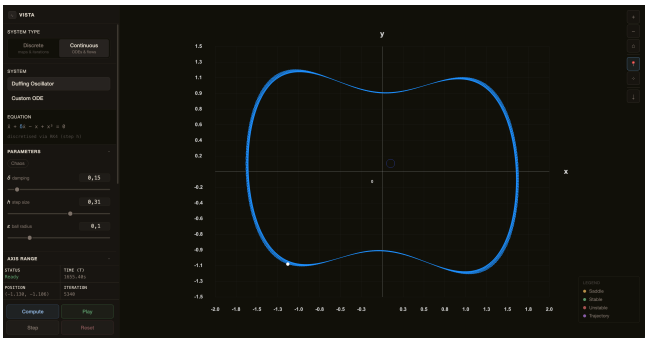

Figure 2: Main viewport showing Duffing equation with bounded noise visualization and the sidebar

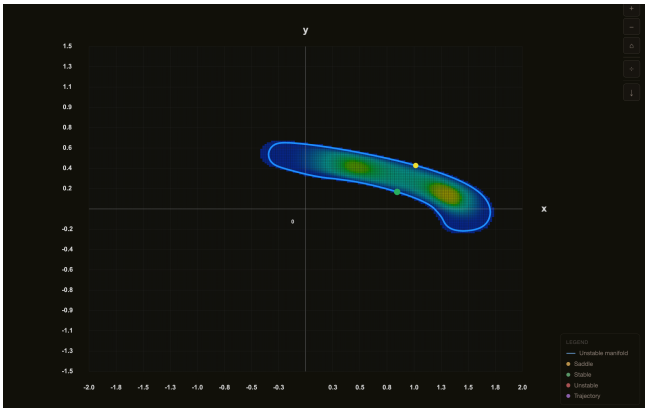

Figure 3: Bounded noise Hénon map with MIS approximation and ULAM grid visualization

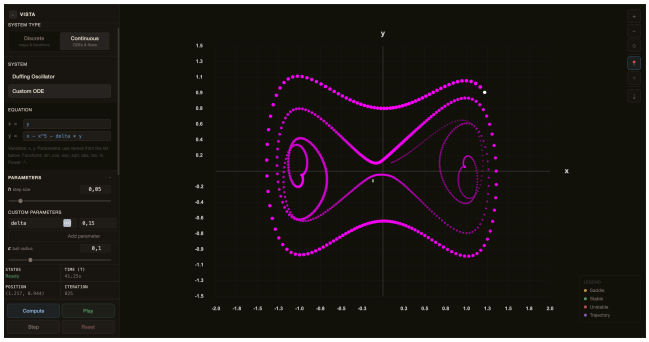

Figure 4: Custom continuous system visualization

5.5 Deployment

Every push to the master branch triggers a GitHub Actions workflow with two parallel jobs: a Rust job that compiles the library and runs `cargotest` (covering all unit tests for the numerical solvers, expression evaluator, and Hausdorff distance routines), and a frontend job that installs dependencies and runs the Vitest component test suite. On success, a deployment job compiles the Rust library to WASM, builds the Vite/React bundle, and publishes the result to GitHub Pages. The live tool is accessible at https://namvdo.github.io/set-valued-viz without any user-side configuration.

6 Evaluation

6.1 Evaluation Plan

Our evaluation addresses three questions: (1) Do the computed minimal invariant sets and topological bifurcations match the numerical results reported in the literature? (2) Do the automated test suites confirm correctness of the individual components? (3) What are the concrete performance characteristics of the major computational routines under realistic workloads?

6.2 Experimental Setup

All experiments were run on an Apple M1 Pro (8 cores, 16 GiB RAM) running macOS 14.6.1. The software stack comprised:

- **Rust / Cargo** 1.81.0 – computation core and test runner
- **Node.js** v24.7.0, **npm** 11.5.1 – frontend toolchain
- **React** 19.1.1, **Three.js** 0.180.0 – visualization interface
- **Vite** 7.1.2, **Vitest** 3.2.4 – frontend build and test

Backend correctness tests were run with `cargo test --release`. Frontend tests were run with `vitest run`. Performance measurements were produced by the benchmark binary `src/bin/eval_perf.rs`, which times each routine with `std::time::Instant` and writes results to JSON; each case was run once in release mode.

6.3 Part 1: Mathematical Correctness

Single connected MIS (a=0.4, b=0.6, ε=0.065). At these parameters the Hénon map has a stable fixed point near (1.16, 0.69), and $M(\varepsilon)$ forms a single compact connected region around the attractor. BIST traces $\partial M(\varepsilon)$ via the unstable manifold of the boundary map

periodic orbit; the result is a single closed curve, consistent with the qualitative picture in Tey [1].

Figure 5: Saddle fixed point found at (1.158, 0.695)

Topological bifurcation: single region splits into two (a=0.6, b=0.3, ε=0.0625). Lamb, Rasmussen, and Tey [2] report that for b=0.3, ε=0.0625, increasing a past a critical value causes $M(\varepsilon)$ to split from one connected component into two disjoint regions. BIST's bifurcation sweep over $a \in [0.1, 2.0]$ detects this transition via a discontinuity in the Hausdorff distance between consecutive $M(\varepsilon)$ approximations, correctly identifying the bifurcation region reported in [2].

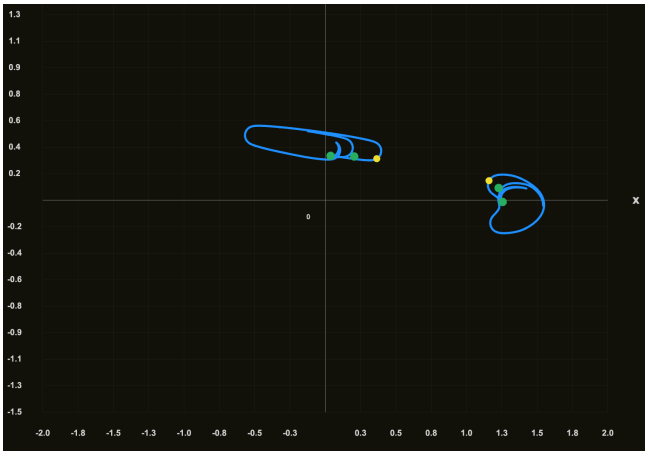

Figure 6: Topological bifurcation happens at a = 0.6, b = 0.3, eps = 0.0625 when a single region splits into two

Zero-noise recovery of the deterministic map. When ε=0, F_0 collapses to the point map f, so E should reduce to f. The unit test `test_boundary_map_reduces_to_henon_at_zero_epsilon` asserts that $E(x, y, n_x, n_y)\big|_{\varepsilon=0} = f(x, y)$ to within 10^{-10} for all inputs. For a=1.4, b=0.3, ε=0, BIST's trajectory panel reproduces the standard Hénon strange attractor, visually matching the published figure on Wikipedia.

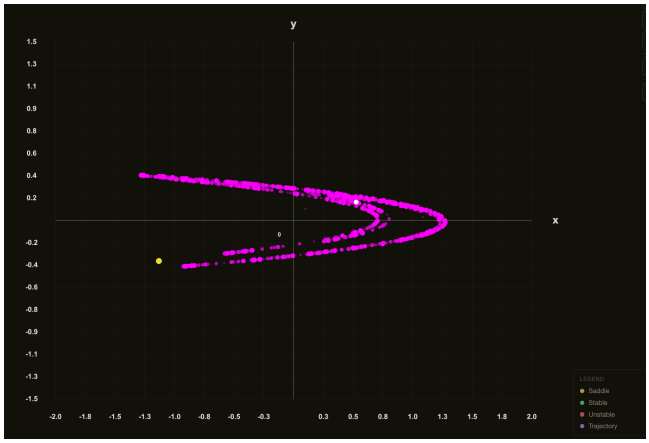

Figure 7: The standard strange attractor of Hénon map with a = 1.4, b = 0.3, and zero noise amplitude

User-defined maps: Lozi map. The Lozi map $f(x, y) = (1 - a|x| + y,\ bx)$ uses absolute value, handled by the `preprocess_abs` pipeline step. Entering `1-1.4*|x|+y` and `0.3*x` with ε=0 produces the Lozi strange attractor, matching the standard shape for a=1.4, b=0.3. The unit test `test_user_defined_henon_matches_generic` confirms that a user-defined Hénon expression matches the hard-coded `HenonSystem` with residuals below 10^{-10}.

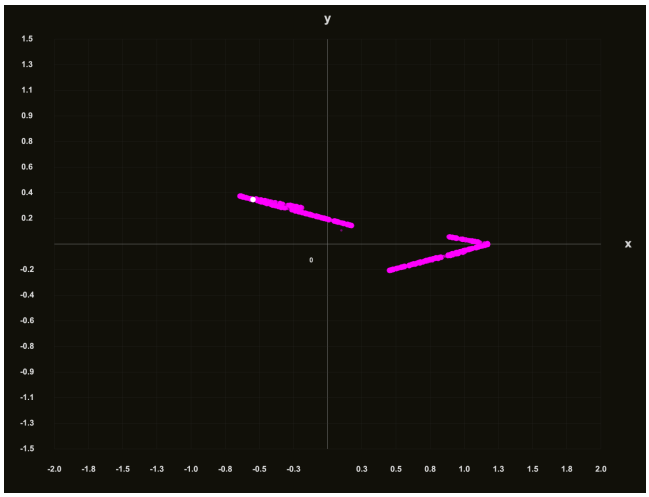

Figure 8: The standard Lozi map with a = 1.4, b = 0.3 and zero noise amplitude

6.4 Part 2: Automated Test Suite

The backend suite comprises 92 unit tests across boundary_periodic.rs, unstable_manifold.rs, `dynamical_systems.rs`, user_defined.rs, and `range.rs`, covering Jacobian arithmetic, eigenvalue computation, Davidchack–Lai solver correctness, Hausdorff distance properties, manifold invariance, and stability classification. All 92 tests passed in 137.81s.

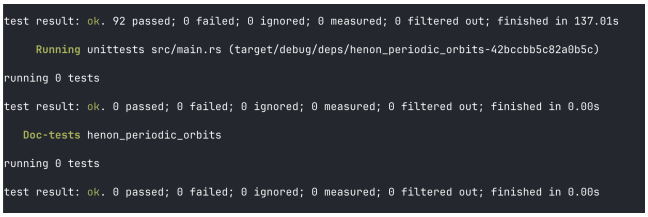

Figure 9: Backend test results

The frontend suite (Vitest) comprises 72 tests across 15 files covering `BifurcationPanel`, `ParameterSweepPanel`, `ManifoldsPanel`, `PeriodicOrbitsPanel`, `ManifoldRendering`, `Sidebar`, `Viewport`, `ControlsBar`, `ParametersPanel`, and utility modules. All 77 tests passed in 2.86 s.

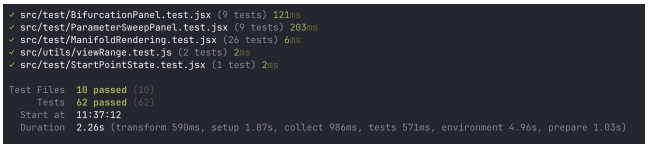

Figure 10: Front-end test results

6.5 Part 3: Computational Performance

Tables 1 and 2 report wall-clock times on the M1 Pro host. WASM execution is typically 1.5–3× slower due to the absence of SIMD and `rayon` parallelism.

Table 1: Backend performance (Apple M1 Pro).

Routine	Parameters	ms
Periodic orbits	a=1.4, b=0.3, $p \le 7$, grid 15^2	1651
Periodic orbits	a=0.55, b=0.3, $p \le 7$, grid 15^2	2575
Periodic orbits	a=0.60, b=0.3, $p \le 7$, grid 15^2	2592
Periodic orbits (stress)	$p \le 8$, grid 20^2	4891
Unstable manifold	a=1.4, 4522 pts, Converged	3.1
Unstable manifold	a=0.55, 1570 pts, SelfIntersect	1.5
Ulam method	48^2 grid, 147 456 samples	32
Ulam method	64^2 grid, 409 600 samples	120
Ulam method (stress)	80^2 grid, 921 600 samples	297
RK4 (Duffing)	50 000 steps, h=0.05	1.6
RK4 (stress)	200 000 steps	6.4

The unstable manifold routine is the fastest — under 5 ms even at high iteration budgets — because it is a sequential forward iteration that terminates early on convergence or self-intersection. Periodic orbit search dominates total computation time; at max period 8 with a dense grid it takes roughly 5s, acceptable for a one-shot interactive trigger but too slow for continuous sweeping. Ulam grid construction scales as $O(n^2)$: going from a 48^2 to an 80^2 grid increases time roughly 9×, consistent with the quadratic growth in both box count and total sample count. Frontend geometry preparation is negligible at typical resolutions, staying below 3 ms for up to 24 000 manifold points.

Table 2: Frontend geometry-preparation performance (Node.js)

Task	Configuration	avg ms
Orbit payload prep	40 orbits × 80 pts	1.1
Orbit payload prep (stress)	250 orbits × 240 pts	18.1
Manifold geometry	12 branches × 2000 pts	2.9
Manifold geometry (stress)	40 branches × 5000 pts	11.8
Ulam overlay	64^2 grid (4096 boxes)	0.3
Ulam overlay (stress)	128^2 grid (16 384 boxes)	0.6

6.6 Data Analysis

Several observations emerge from the combined correctness and performance results.

The periodic orbit search is the computational bottleneck. At the typical configuration ($p \le 7$, grid 15^2), it accounts for over 99% of total wall-clock time per analysis run, with the unstable manifold and Ulam routines contributing less than 1% combined. This asymmetry arises because the Davidchack–Lai solver must launch from every grid cell and every normal direction in the θ-grid, making its cost $O(g^2 \cdot \theta \cdot p)$ where g is the spatial grid size, θ the angular grid size, and p the maximum period. In contrast, the unstable manifold computation is a single forward trajectory from a known seed orbit and terminates as soon as it self-intersects or converges, explaining the sub-5 ms timings even at high iteration budgets.

The Ulam method shows clean $O(n^2)$ scaling. Moving from a 48^2 to a 64^2 grid multiplies time by $\approx 3.8\times$ (31 ms to 120 ms), and from 48^2 to 80^2 by $\approx 9.4\times$ (31 ms to 297 ms). These ratios are consistent with the theoretical $(64/48)^2 \approx 1.78$ and $(80/48)^2 \approx 2.78$ scaling of box count, with the additional factor coming from proportionally more samples per box in the stress cases.

The near-bifurcation case (a=0.55, b=0.3, ε=0.0625) reveals a qualitatively different manifold topology: the unstable manifold computation terminates with `SelfIntersection` rather than `Converged`, and produces only 1570 points compared to 4522 for the post-bifurcation case at a=1.4. This is expected — near the bifurcation threshold the boundary $\partial M(\varepsilon)$ is geometrically more complex and the manifold branches fold back on themselves earlier. The periodic orbit search also takes $\approx 56\%$ longer at the near-bifurcation parameter (a=0.55, 2575 ms) than at the classical Hénon attractor (a=1.4, 1651 ms).

7 Discussion

7.1 Limitations

Computational Performance. Dense grid searches and high-period orbit detection scale poorly with resolution and orbit period; adaptive boundary refinement for unstable manifold approximation compounds this cost at fine tolerances; and Ulam's method degrades with large grid sizes as the transition matrix grows quadratically in the number of boxes.

Modeling Expressiveness. Support for user-defined symbolic maps is restricted to two-dimensional systems with a limited expression syntax; higher-dimensional definitions are not yet supported.

Platform Constraints. Browser memory limits and single-threaded JavaScript execution cap the problem sizes that are practical, and users on low-end hardware may encounter slowdowns on demanding parameter sweeps.

Visualization Scope. All visualizations are currently restricted to two-dimensional phase space; extending to three dimensions would require significant changes to both the rendering pipeline and the boundary map computation.

7.2 Reflection on Initial Target Setting

The original project scope was substantially more modest than what was ultimately delivered. At the outset, the plan was to implement a rough Python prototype that visualized the Hénon map with additive bounded noise added, with no particular emphasis on interactivity, extensibility, or performance. Through continuous feedback from Dr. Kalle Timperi at the University of Oulu and from researchers at Imperial College London, the target evolved significantly. The scope expanded to a complete browser-native application covering both discrete iterated maps and continuous ODEs, a generic user-defined equation pipeline, multiple complementary analysis methods (boundary map, Ulam stationary measure, periodic orbit detection, parameter sweep), and a Rust/WebAssembly architecture capable of near-native performance without any installation. The final tool substantially exceeds the initial specification in both breadth and technical depth.

7.3 Reflection on the State of the Art

Existing research tools for set-oriented numerical dynamics occupy two distinct niches. GAIO [6], the most mature tool in this space, provides a full suite of subdivision and transfer operator algorithms but requires a local MATLAB installation and familiarity with its specialized API, placing it out of reach for students or researchers who want quick exploratory access. Python-based implementations such as the one developed in Tey's thesis [1] are more accessible but are single-purpose scripts with no interactive interface, requiring the user to modify source code to change parameters. BIST addresses the gap between these two extremes. It matches the mathematical scope of specialized research implementations – boundary map iteration, Davidchack–Lai periodic orbit detection, Ulam's method, Hausdorff-distance bifurcation detection – while being immediately accessible to anyone with a modern web browser at https://namvdo.github.io/set-valued-viz. The zero-install deployment and interactive parameter panels make it suitable as a teaching tool, while the user-defined equation pipeline and JSON export for sweep results keep it useful for research workflows.

7.4 Future Work

Several directions follow naturally from the current limitations. On the modeling side, the most impactful extension would be support for *non-uniform noise*, where the noise amplitude $\varepsilon(x)$ varies with the system state rather than being globally constant; this would allow modeling systems where uncertainty is concentrated in particular regions of phase space. A complementary direction is to generalize beyond the minimal invariant set to visualize *arbitrary invariant sets* and their nested structure, enabling a more complete picture of the long-term geometry of the system. On the computational side, offloading heavy routines to a backend server – rather than running everything client-side in WASM – would remove the browser memory ceiling and enable extensive parameter sweeps that are currently impractical; this is particularly relevant for researchers who need high-resolution Ulam grids or exhaustive periodic orbit searches. On the platform side, WebGPU could parallelize the transition matrix assembly across thousands of shader threads, eliminating the $O(n^2)$ Ulam bottleneck for large grids. The longer-term goal is for BIST to serve a dual role: an intuitive, self-contained learning environment for students encountering set-valued dynamics for the first time, and a versatile, extensible research instrument for studying the qualitative geometry of dynamical systems with bounded noise.

8 Conclusions

In this project, we developed a browser-based visualization tool for set-valued dynamical systems with bounded additive noise. The tool integrates extended boundary mappings for minimal invariant set computation, a GAIO-based Ulam's method for stationary measure approximation, and a stabilized periodic orbit solver, all implemented in Rust/WebAssembly and rendered in real time via Three.js. Validation against published theoretical benchmarks confirmed correctness, and the interactive performance demonstrates the viability of high-performance mathematical computation delivered through standard web technologies.

References

[1] W. H. Tey, "Boundary mappings of minimal invariant sets," Ph.D. dissertation, Imperial College London, 2023.

[2] J. S. W. Lamb, M. Rasmussen, and W. H. Tey, "Bifurcations of the Hénon map with additive bounded noise," arXiv preprint arXiv:2504.02776, 2025.

[3] J. S. W. Lamb, M. Rasmussen, and C. S. Rodrigues, "Topological bifurcations of minimal invariant sets for set-valued dynamical systems," *Proc. Amer. Math. Soc.*, vol. 143, no. 9, pp. 3927–3937, 2015.

[4] G. Peschke, "Computational analysis of invariant sets and their bifurcation in set valued dynamical systems," Diplomarbeit, Technische Universität Dresden, 2015.

[5] M. Dellnitz and A. Hohmann, "A subdivision algorithm for the computation of unstable manifolds and global attractors," *Numer. Math.*, vol. 75, no. 3, pp. 293–317, 1997.

[6] 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*, B. Fiedler, Ed. Springer, 2001, pp. 145–174.

[7] S. M. Ulam, *A Collection of Mathematical Problems*. New York: Interscience Publishers, 1960.

[8] G. Duffing, *Erzwungene Schwingungen bei Veränderlicher Eigenfrequenz und ihre Technische Bedeutung*. Braunschweig: Vieweg, 1918.

[9] D. Burago, Y. Burago, and S. Ivanov, *A Course in Metric Geometry*. Providence, RI: American Mathematical Society, 2001.

[10] R. L. Davidchack and Y.-C. Lai, "Efficient algorithm for detecting unstable periodic orbits in chaotic systems," *Phys. Rev. E*, vol. 60, no. 5, pp. 6172–6175, 1999.

[11] L. Arnold, *Random Dynamical Systems*. Berlin: Springer-Verlag, 1998.

[12] A. J. Homburg and T. Young, "Hard bifurcations in dynamical systems with bounded random perturbations," *Regul. Chaotic Dyn.*, vol. 11, no. 2, pp. 247–258, 2006.

[13] C. Kuehn, G. Malavolta, and M. Rasmussen, "Early-warning signals for bifurcations in random dynamical systems with bounded noise," *J. Math. Anal. Appl.*, vol. 464, no. 1, pp. 58–77, 2018.

[14] K. Kourliouros, J. S. W. Lamb, M. Rasmussen, W. H. Tey, K. Timperi, and D. Turaev, "Smooth persistence of attractors for set-valued dynamical systems: A boundary map approach," arXiv preprint arXiv:2303.01895, 2023.

Acknowledgments

To the University of Oulu, and in particular the ACP2 course coordinators and supervisors (Prof. Timo Ojala and Dr. Kalle Timperi), for

guiding the project requirements and providing feedback throughout.

A Group Members' Contributions

- **Nam Do** (*nam.do@student.oulu.fi*) – 300 hours: gathered and analyzed user requirements, designed and built the entire BIST system end-to-end: the Rust computation core, the WebAssembly bridge, the visualization interface, the CI/CD deployment pipeline via GitHub Actions, implemented the evaluation process, and the final report.
- **Jyri Ollanketo** (*jyri.ollanketo@student.oulu.fi*) – 200 hours: Python prototyping, implemented the basic GUI and visualization for Hénon map and some custom equations.
- **Kalle Timperi** (*Kalle.Timperi@oulu.fi*) – Supervisor. Provided continuous feedback on mathematical correctness, scope, and software design throughout the project.

B Demo Screenshots

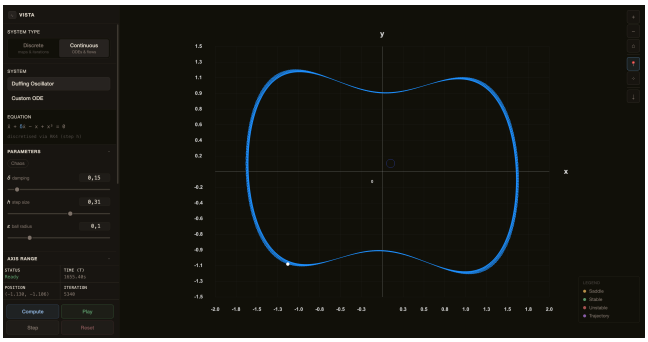

Figure 11: Duffing oscillator with bounded noise in BIST. The sidebar shows the collapsible panel layout; the main viewport renders the ODE trajectory and the boundary of the minimal invariant set in real time.

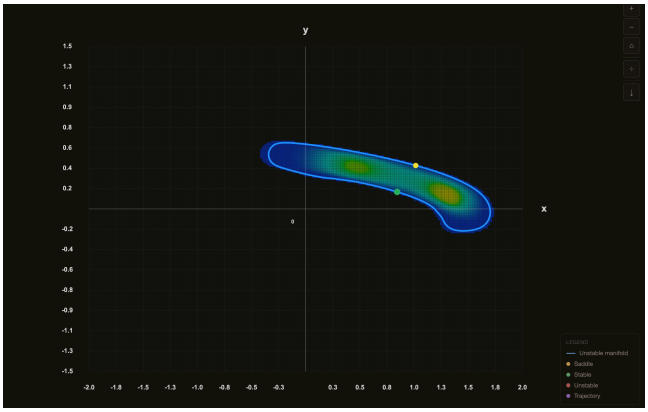

Figure 12: Hénon map with bounded noise (a=1.4, b=0.3, ε=0.0625). The boundary $\partial M(\varepsilon)$ is traced by the unstable manifold (red/blue curves) and the Ulam stationary measure is overlaid as a heatmap.

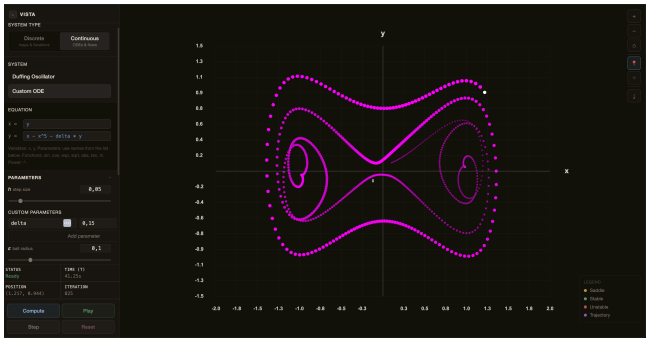

Figure 13: User-defined continuous system entered via the symbolic equation panel. This demonstrates BIST's generic expression evaluation pipeline handling a custom ODE without any code changes.

Figure 14: Single connected minimal invariant set for a=0.4, b=0.6, ε=0.065. The saddle fixed point of the boundary map E is found near (1.158, 0.695).

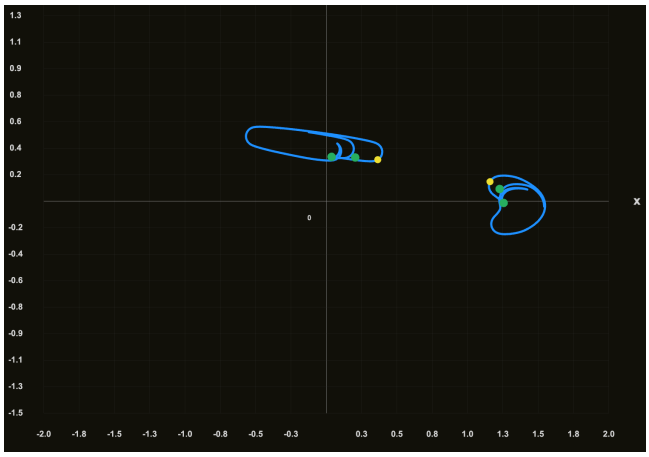

Figure 15: Topological bifurcation at a=0.6, b=0.3, ε=0.0625: the minimal invariant set splits from one connected component into two disjoint regions.

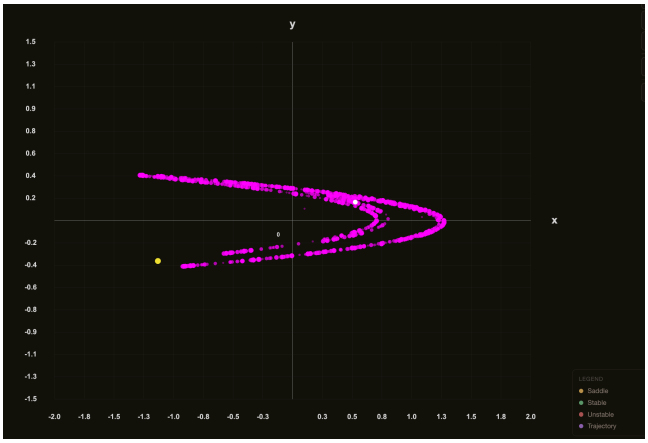

Figure 16: Standard Hénon strange attractor at a=1.4, b=0.3, ε=0 (zero noise). The trajectory panel reproduces the classical attractor, confirming that the set-valued dynamics reduce correctly to the deterministic map.

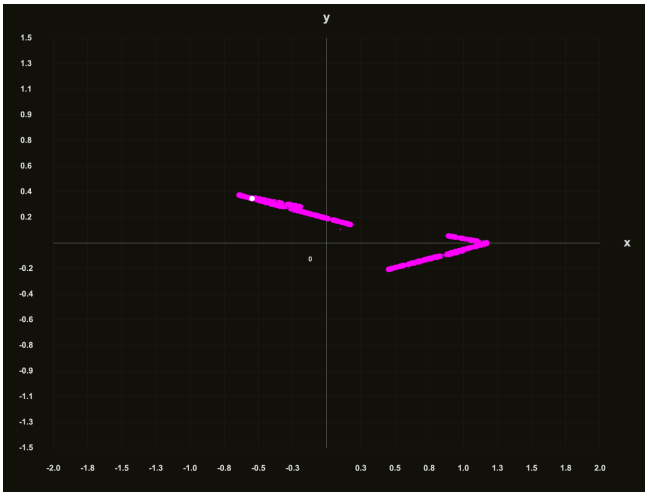

Figure 17: Lozi map strange attractor (a=1.4, b=0.3, ε=0) produced via the user-defined equation interface using `1-1.4*|x|+y` and `0.3*x`.

```
test result: ok. 92 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 137.01s
     Running unittests src/main.rs (target/debug/deps/henon_periodic_orbits-42bccbb5c82a0b5c)
running 0 tests
test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
   Doc-tests henon_periodic_orbits
running 0 tests
test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Figure 18: Backend test results

```
✓ src/test/BifurcationPanel.test.jsx (9 tests) 121ms
✓ src/test/ParameterSweepPanel.test.jsx (9 tests) 203ms
✓ src/test/ManifoldRendering.test.jsx (26 tests) 6ms
✓ src/utils/viewRange.test.js (2 tests) 2ms
✓ src/test/StartPointState.test.jsx (1 test) 2ms

Test Files  10 passed (10)
     Tests  62 passed (62)
  Start at  11:37:12
  Duration  2.26s (transform 590ms, setup 1.87s, collect 986ms, tests 571ms, environment 4.96s, prepare 1.03s)
```

Figure 19: Frontend test results.