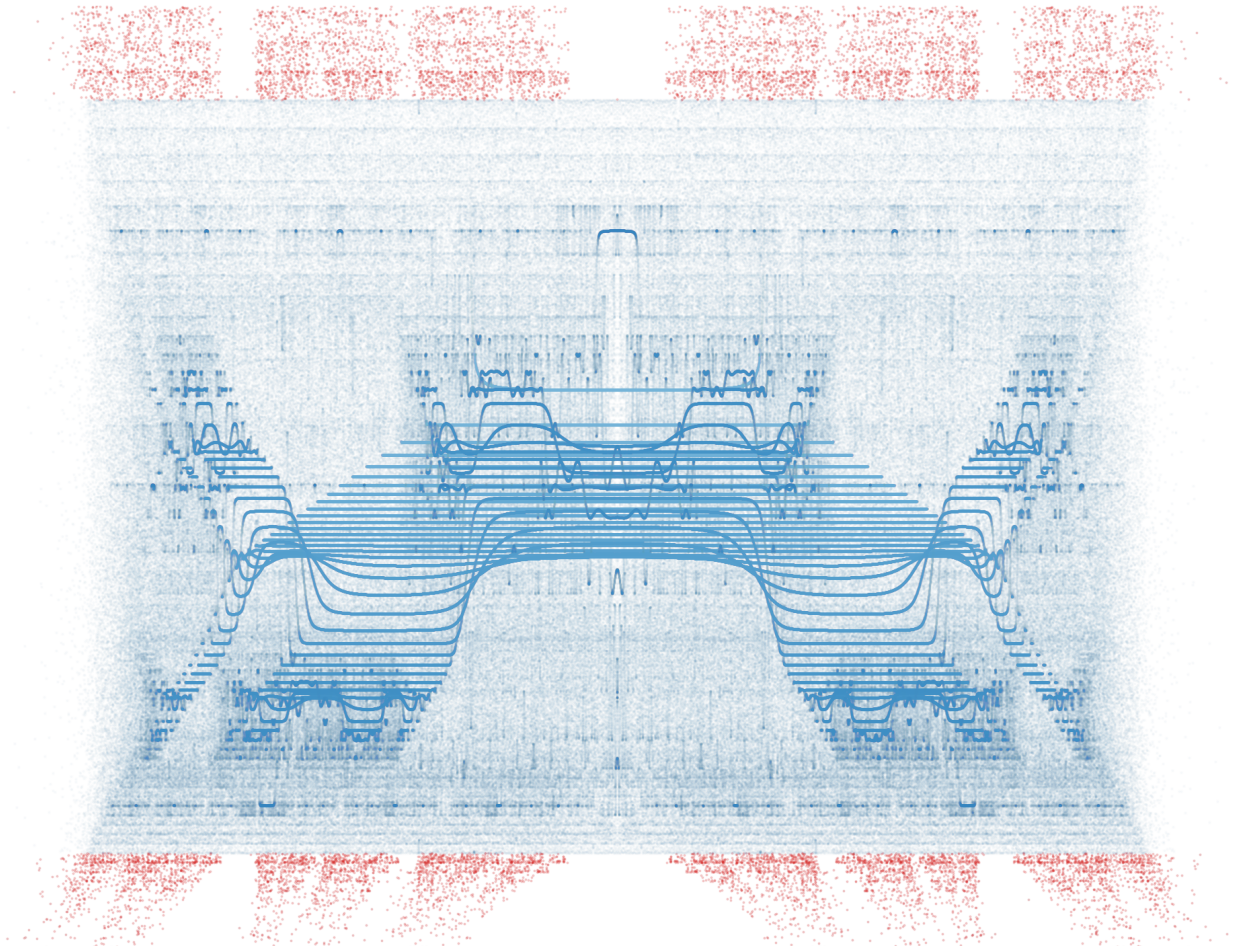


Order and Chaos in the Quadratic Family $f_c(x) = x^2 + c$

Leto Cavé

February 2, 2026



Abstract

In this study we investigate the transition from deterministic stability towards chaotic dynamics within the quadratic family $f_c(x) = x^2 + c$. We explore both the theoretical analysis of fixed and periodic points and numerical simulations of the resulting dynamics and bifurcations. We lay out the process of a bifurcation cascade and identify subsequent stable periodic windows. Our results cover a universal limiting ratio of bifurcation intervals known as the Feigenbaum constant and the existence of the Prisoner Set. In this short study we conclude that while the map f_c is deterministic, its sensitivity to parameters leads to unpredictable long-term behavior. We touch upon the intricate interplay between local stability and global dynamical behaviors, revealing how simple polynomial maps can give rise to complex fractal geometries and unpredictable orbits leading to fascinating visualizations.

Keywords: Quadratic map f_c , Bifurcation point, Period-doubling cascade, Feigenbaum constant δ , Prisoner Set P_c , Julia Set ∂P_c , Deterministic chaos.

Titel: Order and Chaos in the Quadratic Family $f_c(x) = x^2 + c$

Auteur: Leto Cavé, leto.cave@gmail.com

Einddatum: February 2, 2026

Written for the course Programmeren en Experimenteren,
BSc Mathematics, University of Amsterdam, February 2026.

Contents

1	Introduction	4
2	Stability Analysis of the Quadratic Map	5
2.1	Fixed Point Dynamics and Local Stability	5
2.2	The Period-Doubling Cascade and Feigenbaum Universality	9
3	Global Dynamics and the Prisoner Set	11
3.1	The Prisoner Set and Julia Boundary	11
3.2	Analysis of High-Iteration Visualizations	11
4	Conclusion	15
	Bibliography	16
	Python Code	17

1 Introduction

Deterministic chaos comes as a surprising phenomenon in dynamical systems, where simple deterministic rules can lead to complex and unpredictable behavior. It is a fascinating area of study that from the origins of natural sciences bridges mathematics, physics, engineering, economics and beyond [12, 1].

Here we study the evolution of a system over time, governed by a set of deterministic equations. We will iterate the relation many times to which the state progresses. Many forms of dynamical systems exist such as: discrete-, real-, cellular-, and multidimensional dynamical systems. As parameters within these equations are varied, the system can undergo changes in stability at critical points formally known as bifurcations. These bifurcations can lead to the emergence of chaotic dynamics, characterized by extreme sensitivity to initial conditions and an apparent randomness despite the underlying determinism [2]. The stability of these models has been of particular importance. In the 19th century, primarily through the work of Henri Poincaré on the Three-Body Problem, mathematicians began to rigorously analyze the field. As yet bounded by the constraints of theory he showed groundbreaking observations on non-periodic, divergent behaviours. Due to the fast growing complexities findings remained largely theoretical.

It was only with the rise of computers in the 1960s and 1970s that the field expanded rapidly. The ability to perform extensive numerical simulations allowed researchers like Edward Lorenz (1963) to pioneer on deterministics of non-periodic systems such as the "butterfly effect" providing one of first visualizations of chaotic attractors [7]. This era transformed dynamical systems from an abstract branch of topology into a science capable of modeling real-world turbulence across all sciences.

We will touch upon this field of study through the quadratic family of maps, defined by $f_c(x) = x^2 + c$ for a real parameter c . At first glance some would consider the function f_c too elementary for any serious analysis; we hope to demonstrate the profound complexity inherent in this foundational model. This fundamental model opens the door from order to the realm of chaos. During this report we delve into the stability analysis of fixed points, the bifurcation structure leading to chaos, and the emergence of fractal sets such as the Prisoner Set.

2 Stability Analysis of the Quadratic Map

2.1 Fixed Point Dynamics and Local Stability

Before we can touch upon the more complex dynamics of chaos theory, we must first understand the stability of fixed points within the quadratic family. Consider an initial point $x_0 \in \mathbb{R}$ which we iterate by the function f_c to obtain a sequence of points defined by $x_{n+1} = f_c(x_n)$ for $n = 0, 1, 2, \dots$. A fixed point of the first iterate is defined as an initial value x^* such that $f_c^{o1}(x^*) = x^*$. Our first objective is to determine the parameter values c for which the sequence (x_n) converges to a fixed point as $n \rightarrow \infty$. To do so we will increase the order of iterations step by step.

Consider the first iterate $f_c^{o1} \equiv f_c$ [4]. It is also called the orbit of order 1, e.g. $x_0 \xrightarrow{f_c} x_0$. Deriving the fixed points of f_c implies finding the points $x_1 = f(x_0)$, and $x_2 = f(x_1)$ and so on. Therefore the criterion is met when for a particular point $x^* \in \mathbb{R}$ we have that $(x^*)^2 + c = x^*$. This condition implies that fixed points of f are also fixed points of f^{on} , $n = 2, 3, \dots$. One could prove this by induction. For $n = 1$, we have that $f(x^*) = x^*$. Now assume $f^{ok}(x^*) = x^*$ is true, then $f^{o(k+1)}(x^*) = f(f^{ok}(x^*)) = f(x^*) = x^*$ is also true. By induction we proved that it is true for all k . To obtain an expression for the set of fixed points we isolate x^* ,

$$x_{\pm}^* = \frac{1}{2}(1 \pm \sqrt{1 - 4c}) \quad (\text{Fixed Points of the First Orbit}) \quad (2.1)$$

The two points are visualized in Figure 2.1. We observe that the fixed points usually come in pairs of 2 (x_-^*, x_+^*) due to the symmetry of the quadratic map. Note however that in such a pair not both points need to be attracting or repelling. For example the plots show that for $c = 0$ we have a repelling fixed point at $x_+^* = 1$ and an attracting fixed point at $x_-^* = 0$. At $c = 1/4$ the two fixed points $x_{\pm}^*$ merge into a single point $x^* = 1/2$, which serves as an example of neutral stability.

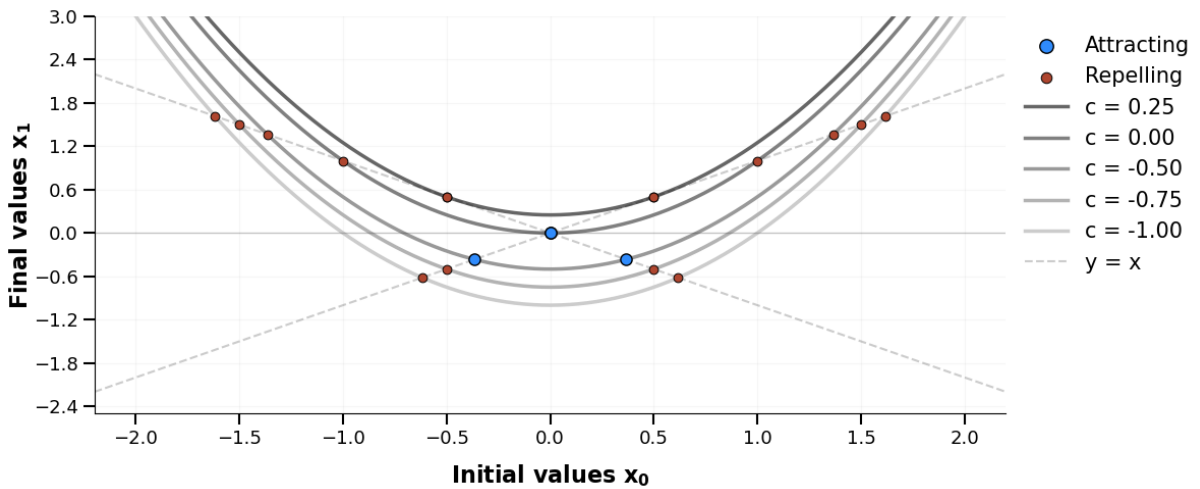


Figure 2.1: Period-1 analysis of $f_c(x)$.

Continuing the investigation of the set of parameters for c we recognise the familiar quadratic formula in $x_{\pm}^*$ (2.1). From the discriminant we know that only real solutions exist whenever $c < 1/4$. If c passes this boundary (i.e. $c > 1/4$) we are certain that x_0 is not fixed in $\mathbb{R}$.

Throughout the project we must carefully distinguish between a fixed point and existence of stability that such a point could possess in the long-term behaviour of the system. Under a small positional change of the initial value x , stability is characterised by whether the point acts as a "sink", pulling it towards an equilibrium or if it acts as a "source" pushing it away. If such an equilibrium does not exist we call the fixed point repelling. Mathematically the

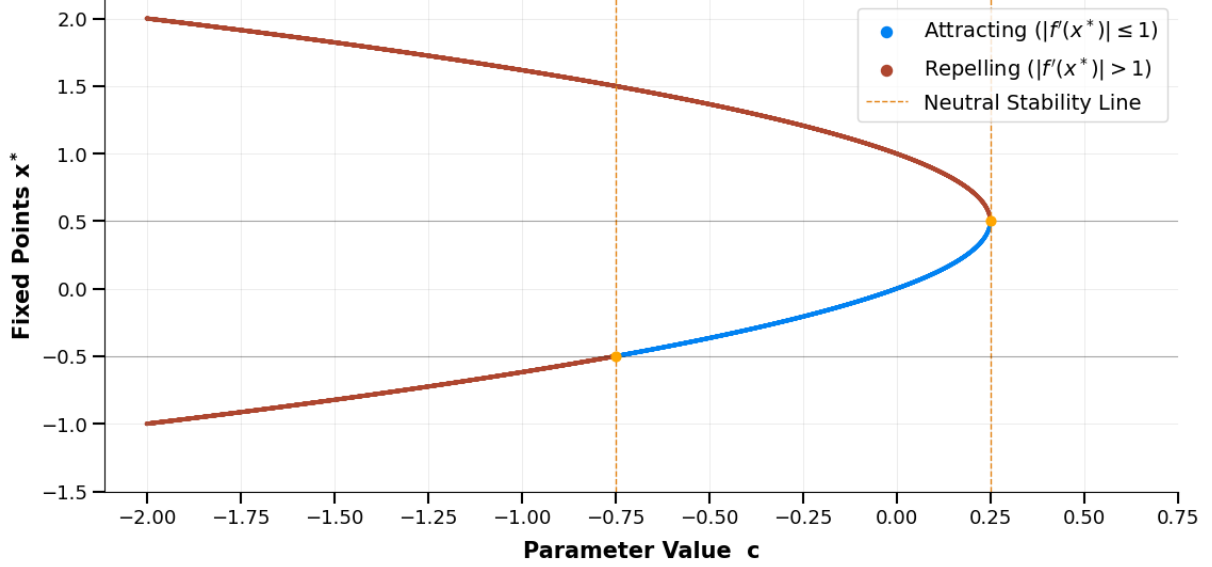


Figure 2.2: Fixed points of the first orbit for $f_c(x)$ plotted as a function of c .

stability criterion is defined as $|f'(x^*)| < 1$ where x^* is a fixed point. Thus it ensures that small perturbations around the fixed point decay over iterations, leading to convergence towards x^* . For $|f'_c(x_-^*)| = |2x_-^*| = |1 - \sqrt{1 - 4c}| < 1$, when we substitute the lower fixed point in stability criterion we find that:

$$|f'_c(x_-^*)| = |2x_-^*| = |1 - \sqrt{1 - 4c}| < 1 \implies c \in (-3/4, 1/4)$$

Thus we find that the lower bound for stable fixed points is $-3/4$ which can be observed in Figure 2.2. As seen in the plot, no fixed points for $c > 0.25$ exist and the orbits x_n diverge to infinity for all x_0 . When $c \leq 0.25$, two fixed points exist (intersecting at $c = 0.25$). The numerical plot aligns perfectly with the theoretical curve $x_{\pm}^* = \frac{1 \pm \sqrt{1 - 4c}}{2}$. Regarding stability, the plot confirms that the lower fixed point x_-^* is attracting (colored blue) on the interval $-0.75 < c < 0.25$ where the derivative condition $|f'(x_-^*)| < 1$ is satisfied, while the upper fixed point x_+^* remains repelling (colored red) for all $c < 0.25$. For $c < -0.75$, both fixed points are repelling, implying that x_n no longer converges to a stationary point.

We call $c = -0.75$ a bifurcation point. It marks the fundamental transition from a stationary equilibrium to a periodic oscillation within the quadratic map $f_c(x) = x^2 + c$. We call this a period-doubling bifurcation, as the system transitions from a fixed point (period-1) to a stable cycle of period-2 [10].

Before further exploring the realm of periodic bifurcation, we find a similar criterion for c in the second iterate $f_c^{\circ 2}$, revealing complexities that span many orders of magnitude as n increases. A periodic orbit of period 2 (or 2-cycle) has the form:

$$x_{2,-} \xrightarrow{f_c} x_{2,+} \xrightarrow{f_c} x_{2,-}$$

To find the fixed points we find the roots of:

$$f_c^{\circ 2}(x) = (x^2 + c)^2 + c = x^4 + 2cx^2 + c^2 + c =: x$$

Consider the general form $f_c^{\circ 2}(x) - x = 0$ obtained when subtracting x on both sides. When solving the orbits of the second iterate we need to observe that the fixed points of the first order must also be fixed points of the second order, e.g. $f_c(x^*) = x^* \implies f_c^{\circ 2}(x^*) = x^*$. It thus will help if we factor out $x^2 - x + c$ by polynomial division in order to find the roots of the expression above.

$$\begin{array}{r}
 x^2 - x + c \overline{) x^4 + 0x^3 + 2cx^2 - x + (c^2 + c)} \\
 \underline{-(x^4 - x^3 + cx^2)} \\
 x^3 + cx^2 - x \\
 \underline{-(x^3 - x^2 + cx)} \\
 (c+1)x^2 - (c+1)x + (c^2 + c) \\
 \underline{-(c+1)x^2 + (c+1)x - (c^2 + c)} \\
 0
 \end{array}$$

The condition after factorization is then:

$$x^4 + 0x^3 + 2cx^2 - x + (c^2 + c) = (x^2 - x + c)(x^2 + x + c + 1) = 0$$

As the roots of $(x^2 - x + c)$ determine the fixed points for period-1 $x_{\pm}^* = \frac{1}{2}(1 \pm \sqrt{1 - 4c})$, those for period-2 are obtained when solving $x^2 + x + c + 1 = 0$. Doing this we obtain:

$$x_{2,\pm} = \frac{1}{2}(-1 \pm \sqrt{-3 - 4c}) \quad (\text{Fixed Points of a Periodic Orbit of Period-2})$$

Figure 2.3 as shown below provides a visual interpretation to the period-2 fixed points.

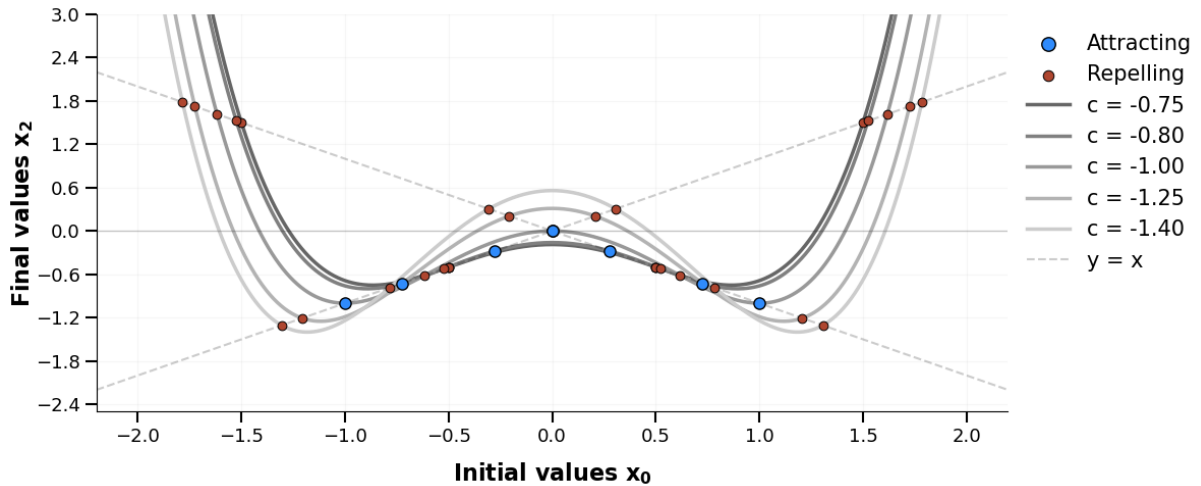


Figure 2.3: Analysis of the orbits of order 2, i.e. $f_c^{\circ 2}(x)$.

To determine the values that guarantee stability we first apply the chain rule on the condition of stability:

$$\left| \frac{d}{dx} f(f(x)) \right| = |f'(f(x)) \cdot f'(x)| < 1 \quad (\text{Chain Rule of } |f'| < 1 \text{ for } f^{\circ 2})$$

For distinct points x_0, x_1 in a cycle $x_0 \xrightarrow{f} x_1 \xrightarrow{f} x_0$, the orbit of period 2 is attracting if:

$$|f'(f(x_{2,+})) \cdot f'(x_{2,-})| = |f'(x_{2,+}) \cdot f'(x_{2,-})| = 4|x_{2,+}x_{2,-}| < 1 \quad (\text{Stability of } f_c^2)$$

As the stability condition yields $|4x_{2,-}x_{2,+}| < 1$, we find the bounds for c by utilizing the fact that the period-2 points are roots of the quadratic factor $x^2 + x + (c + 1) = 0$. As we already found these roots, we obtain:

$$x_{2,-} \cdot x_{2,+} = \left(\frac{-1 + \sqrt{-3 - 4c}}{2} \right) \cdot \left(\frac{-1 - \sqrt{-3 - 4c}}{2} \right) = 1 + c \quad (\text{Product of Roots})$$

Thus we require $|4(1 + c)| < 1$ which implies that $-1.25 < c < -0.75$. Therefore the period-2 orbit is attracting with certainty whenever $c \in (-1.25, -0.75)$. At $c = -0.75$, the stability condition of the fixed point x_-^* reaches -1.0 , marking its transition from an attractor to a repeller. Since x_+^* is repelling for all $c < 0.25$, the loss of stability in x_-^* renders all period-1 orbits unstable. Yet a new stable orbit with double the period may still be formed. This mechanism is not an immediate descent into chaos but a cascade of period-doublings ($2 \rightarrow 4 \rightarrow 8 \rightarrow \dots$) which accumulates at a finite parameter value. The effect of periodicity for orders 1 and 2 can be observed in Figure 2.4.

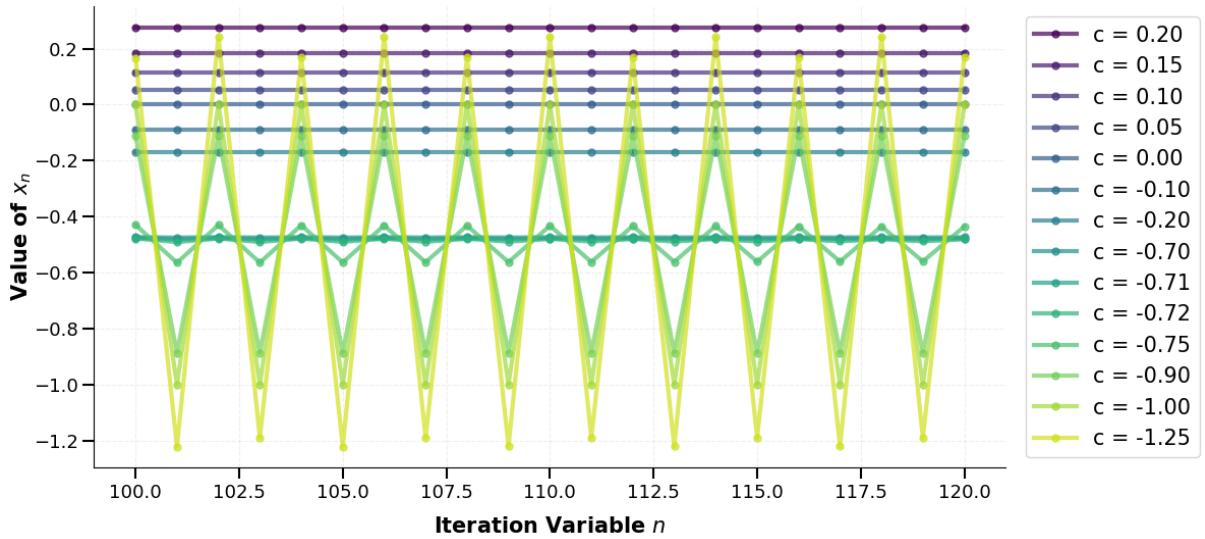


Figure 2.4: Orbit tails x_n against the iteration index n for $n = 100$ to 120 , started from $x_0 = 0$, at fourteen parameter values ranging from $c = 0.2$ down to $c = -1.25$. Above the period-doubling threshold $c = -0.75$ the orbit has settled on a single fixed point and traces a flat line, with the values closest to the threshold still visibly converging; below it the orbit alternates between two values, tracing the period-2 cycle.

So far we have seen that the factorization of $f_c^{\circ 2}(x) - x$ remains analytically solvable. However, consider the next iterated composition $f_c^{\circ 3}(x)$. For the orbits of period-3 we find, as before, the roots of the polynomial $f_c^{\circ 3}(x) - x = 0$:

$$x^8 + 4cx^6 + (6c^2 + 2c)x^4 + (4c^3 + 4c^2)x^2 - x + (c^4 + 2c^3 + c^2 + c) = 0 \quad (2.2)$$

The third iterate $f_c^{\circ 3}(x)$ is a polynomial of degree $2^3 = 8$, and thus could have up to 8 roots. From this it becomes obvious that once we ascend to higher orbits, solving the roots of higher order polynomials becomes analytically intractable.

Therefore, from here we only solve the roots through numerical approximation methods. We used `scipy.optimize.root` to hunt for roots of the polynomial $f_c^{\circ 3} - x$ (2.2) on a small set of c -values as visualized in Figure 2.5.

To obtain more insights we show several interesting cycles of period-2 and period-3 in Table 2.1. The data illustrates the period-doubling cascade very clearly. At $c = -1.25$, $|f'_c|$ hits 1.0, marking the neutral stability threshold where the period-2 cycle bifurcates into a period-4 cycle. Further

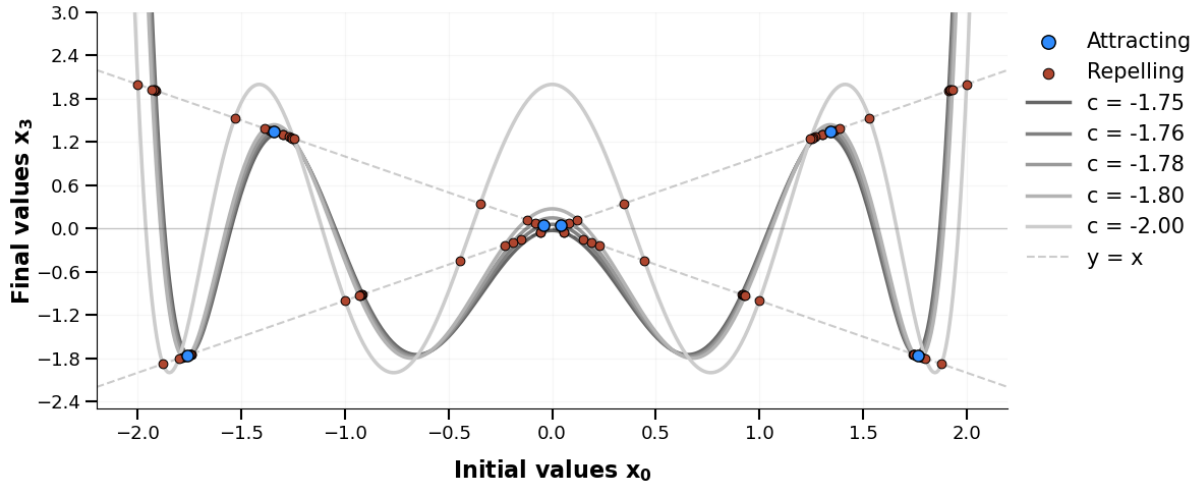


Figure 2.5: Period-3 analysis of $f_c(x)$.

along, at $c \approx -1.75$, a period-3 cycle becomes attracting and a new stable window emerges. The appearance of this period-3 window, after the period-doubling cascade has already passed into the chaotic regime, is a distinct phenomenon known as a tangent (saddle-node) bifurcation. We identify the transition by $|f'_c| = 1.0$, at which a stable and an unstable periodic orbit are created simultaneously; traversing the parameter in the opposite direction, the same pair collides and annihilates.

2.2 The Period-Doubling Cascade and Feigenbaum Universality

The period-3 cycle born at $c \approx -1.75$ remains attracting as c decreases further; at $c = -1.7650$ its multiplier is still 0.7893. The cycle satisfies $|(f_c^3)'(x)| < 1$ until $c \approx -1.7685$, where the multiplier crosses 1 and the period-3 cycle in turn period-doubles into a period-6 cycle [8].

Sharkovskii's Theorem does not predict where this window appears, but it constrains what must accompany it [4]. Once a period-3 orbit exists at $c \approx -1.75$, periodic orbits of every natural period $n \in \mathbb{N}$ exist simultaneously within the system, even if all but one are unstable. The related result of Li and Yorke shows that a period-3 orbit further implies sensitive dependence on initial conditions, which is the formal sense in which period three implies chaos [6].

c-Value	Stability	$ (f_c^{on})' $	Orbit Cycle ($x_0 \rightarrow x_1 \rightarrow x_2$)
-0.8000	Attracting (P2)	0.8000	-0.7236 $\rightarrow$ -0.2764
-1.2500	Neutral (P2 $\rightarrow$ P4)	1.0000	-1.2071 $\rightarrow$ 0.2071
-1.7500	Tangent (P3)	1.0000	-1.7470 $\rightarrow$ 1.3019 $\rightarrow$ -0.0550
-1.7650	Attracting (P3)	0.7893	-1.7633 $\rightarrow$ 1.3441 $\rightarrow$ 0.0416
-1.8000	Repelling (P3)	4.0199	-1.7473 $\rightarrow$ 1.2532 $\rightarrow$ -0.2295
-2.0000	Repelling (P3)	8.0000	-1.8019 $\rightarrow$ 1.2470 $\rightarrow$ -0.4450

Table 2.1: Stability parameters and orbit coordinates across critical parameter regimes. For $c = -1.7650$ the tangent bifurcation has produced two period-3 orbits: the attracting one listed here, and a repelling partner with multiplier 2.6693. By $c = -1.8000$ both orbits are repelling and the period-3 window has closed.

From our analysis so far we have established a cascade of stable windows to higher orbits of order 2^n . It therefore becomes interesting to explore the lengths of the intervals between successive bifurcation points. Let c_n denote the parameter value at which the 2^{n-1} cycle loses

stability to a 2^n cycle, and define the interval $L_n = c_n - c_{n+1}$. Only period-doubling bifurcations enter this sequence: the transition at $c = 0.25$ is a saddle-node bifurcation, at which the two fixed points merge rather than a cycle doubling, and it is therefore excluded. The first two thresholds are known analytically, $c_1 = -0.75$ and $c_2 = -1.25$, so $L_1 = 0.5$. We determine the intervals for the orbits that cascade from order 4 to 8, from order 8 to 16, and so on, numerically. The results of the calculations are summarized in Table 2.2.

Bifurcation	Threshold ($c \approx$)	Interval L_n	Ratio L_n/L_{n+1}
$1 \rightarrow 2$	-0.7500	0.50000 (L_1)	4.23374 (L_1/L_2)
$2 \rightarrow 4$	-1.2500	0.11810 (L_2)	4.55151 (L_2/L_3)
$4 \rightarrow 8$	-1.3681	0.02595 (L_3)	4.64581 (L_3/L_4)
$8 \rightarrow 16$	-1.3940	0.00559 (L_4)	4.66394 (L_4/L_5)
$16 \rightarrow 32$	-1.3996	0.00120 (L_5)	—

Table 2.2: Numerical thresholds of the period-doubling sequence showing the convergence to the Feigenbaum constant $\delta \approx 4.6692016 \dots$.

Table 2.2 shows that the intervals L_i between bifurcation points c_i, c_{i+1} shrink rapidly as we progress through the period-doubling cascade. The bifurcation sequence $1 \rightarrow 2 \rightarrow 4 \rightarrow \dots \rightarrow 2^n$ represents a geometric contraction of stability intervals in the parameter space c . Each transition is governed by condition $|f'_c| = 1$, which marks the critical threshold where a 2^n cycle loses stability to a 2^{n+1} cycle. Here a remarkable universal constant exists known as the Feigenbaum constant δ which describes the limiting ratio of these intervals.

$$\delta = \lim_{n \rightarrow \infty} \frac{L_n}{L_{n+1}} = 4.669\,201\,609\,102\,990 \dots \quad (\text{Feigenbaum Constant [5]})$$

By numerical approximation of the bifurcation points we obtained the following set of values for L_n/L_{n+1} :

$$\{4.23374, 4.55151, 4.64581, 4.66394\} \quad (\text{Numerical Ratios of Bifurcation Intervals})$$

From these results we therefore suspect that our sequence of ratios converges to the theoretical constant as the error decreases after every subsequent ratio. The thresholds themselves accumulate at the Feigenbaum point $c_\infty \approx -1.4011552$, beyond which the cascade is complete and the chaotic regime begins.

3 Global Dynamics and the Prisoner Set

3.1 The Prisoner Set and Julia Boundary

Beyond local stability, we define the so called Prisoner Set P_c as the set of initial conditions x_0 whose orbits remain bounded under infinite iteration, e.g. $n \rightarrow \infty$ [4, 11].

$$P_c = \{x_0 \in \mathbb{R} : \limsup_{n \rightarrow \infty} |f_c^{on}(x_0)| < \infty\} \quad (\text{Prisoner Set})$$

The complement of P_c is the Basin of Attraction of Infinity, consisting of points that diverge once they exceed a critical escape radius R . Thus the Basin of Attraction of Infinity determines the set of points that have stepped outside the zone P_c such that they escape and shoot off to infinity. For the parameter range $c \in [-2, 0.25]$, this threshold is $R = 2$ [4], which can be proven by the domination of the quadratic term and using the reverse triangle inequality:

$$|x_n| > 2, |c| \leq 2 \implies |x_{n+1}| \geq |x_n|^2 - |c| > 2|x_n| - 2 = |x_n| + (|x_n| - 2) \quad (\text{Escape criterion})$$

The excess $|x_n| - 2$ is positive and strictly increases at every step, so the increments do not shrink and $|x_n| \rightarrow \infty$. The boundary between these sets, ∂P_c , constitutes the Julia Set, which exhibits complex fractal geometry. [9, 3]

3.2 Analysis of High-Iteration Visualizations

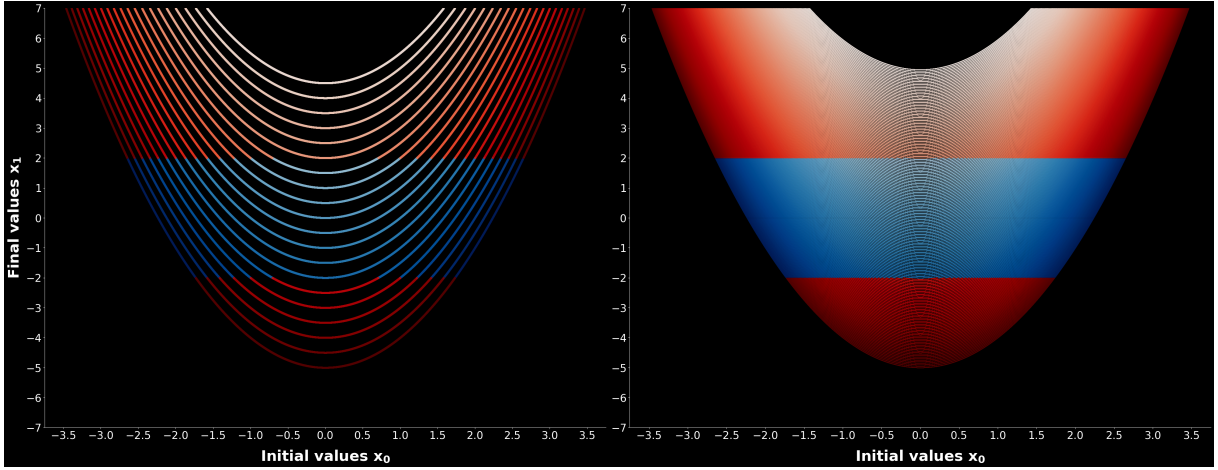
To visualize the global structure of the Prisoner Set P_c , we examine the behavior of orbits across a grid of initial conditions x_0 and a range of parameters c . The initial values are drawn on an adaptive grid over $x_0 \in [-10, 10]$: a dense core spanning the plotted window, and a coarse step of 0.05 outside it, so that resolution is spent where the structure lies. The dense step is tightened as the iteration count grows, from 0.0001 for $n \leq 9$, to 0.00005 for $n = 10$ and $n = 20$, and 0.00001 for $n = 50$ and $n = 200$. The parameter c is sampled over $c \in [-5, 5]$, and the sequence of plots maps the evolution of x_0 for iteration counts $n \in \{1, 2, 3, 5, 10, 20, 50, 200\}$ using two distinct sampling densities in c :

1. Discrete (Sparse) Sampling: a step of 0.5 in c , giving 20 c -values, to provide a clear view of individual iterate curves and the primary bifurcation structure.
2. Continuous (Dense) Sampling: a step of 0.05 in c , giving 200 c -values, to reveal the solid horizontal density of the Prisoner Set and its emergent fractal patterns

In these visualizations, points remaining within the escape radius $R = 2$ are colored blue, while those satisfying the escape criterion $|x_n| > 2$ are colored red. These plots show that bounded behaviour is not restricted to the stable fixed-point interval $c \in (-0.75, 0.25)$. While local stability of the fixed point is lost at $c = -0.75$, the Prisoner Set persists down to $c = -2.0$, revealing that orbits remain bounded within the chaotic regime. Figure 3.1 through Figure 3.8 illustrate the evolution of the Prisoner Set across increasing iteration counts. As the iteration count n increases, a clear transition from deterministic order to complex chaotic regions is observed. While the sparse iterates (observed at the left figures) visualize the discrete mechanism of the quadratic map clearly, the dense sampling (at the right plots) reveals the density of the set revealing the nuance of the map f_c .

At high iterates ($n = 200$) (shown at Figure 3.8), the system reveals horizontal strata of stability lines slicing through the chaotic background. This visualization was made possible by reducing the opacity to very low values with a counterbalance of a large number of points. These bands represent stable periodic windows, such as the Period-3 window at $c \approx -1.75$, where the map converges to a repeating cycle despite being surrounded by chaotic parameters. In the chaotic regions between these bands, the orbits seem to exhibit a never-settling repeated pattern, yet remain trapped within the blue bounded regions.

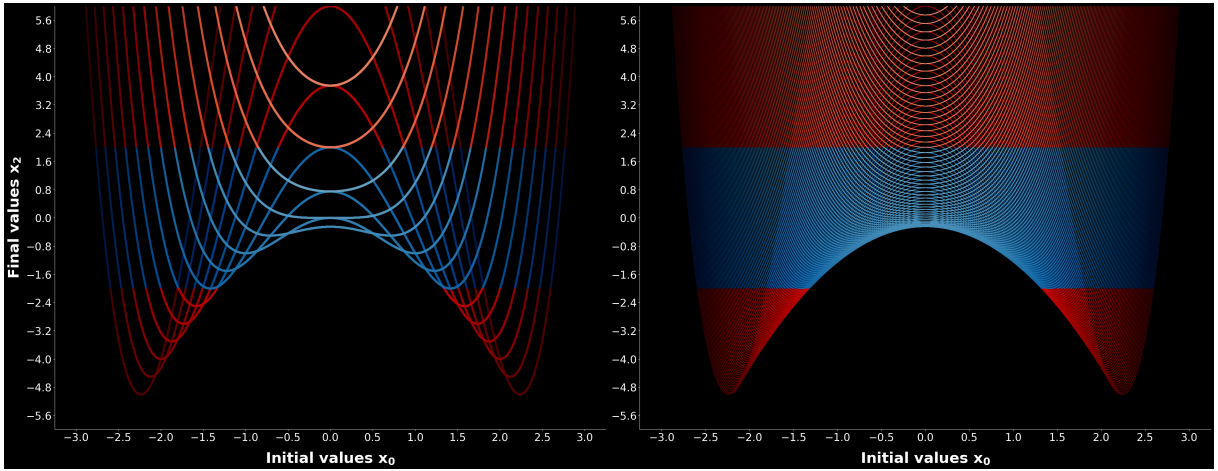
To conclude the discussion on the prisoner set, it is worth noting that at lower orbits, the escaping points (shown in red) diverge rapidly compared to the handful of points still visible in red as of the 50th iterate (Figure 3.7). These points escape relatively slowly, but as we move to $n = 200$ (Figure 3.8), none remain visible in the plotted window.



(a) Sparse steps of the 1st Iterate

(b) Dense steps of the 1st Iterate

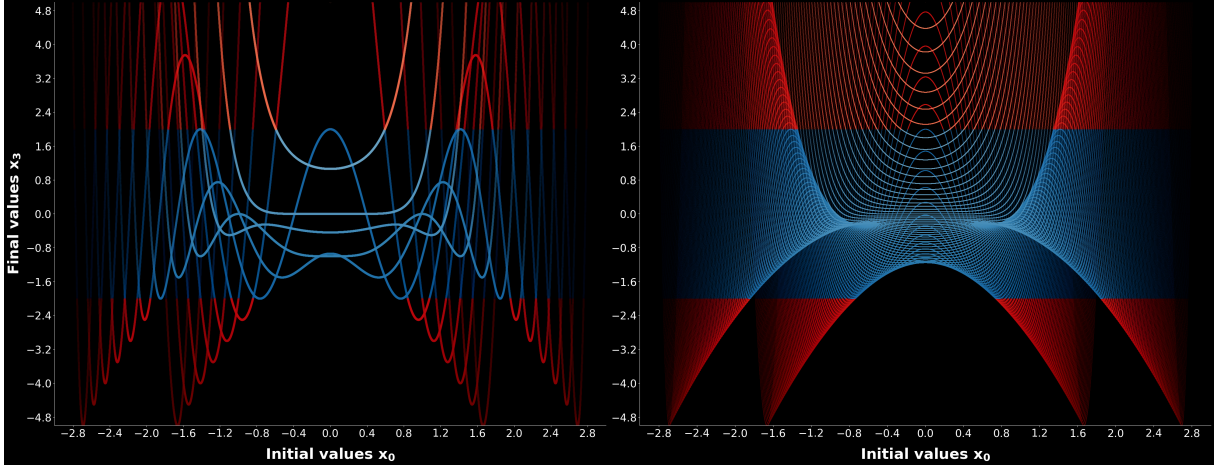
Figure 3.1: Comparison of sparse and dense iterate for $f_c(x)$.



(a) Sparse steps of the 2nd Iterate

(b) Dense steps of the 2nd Iterate

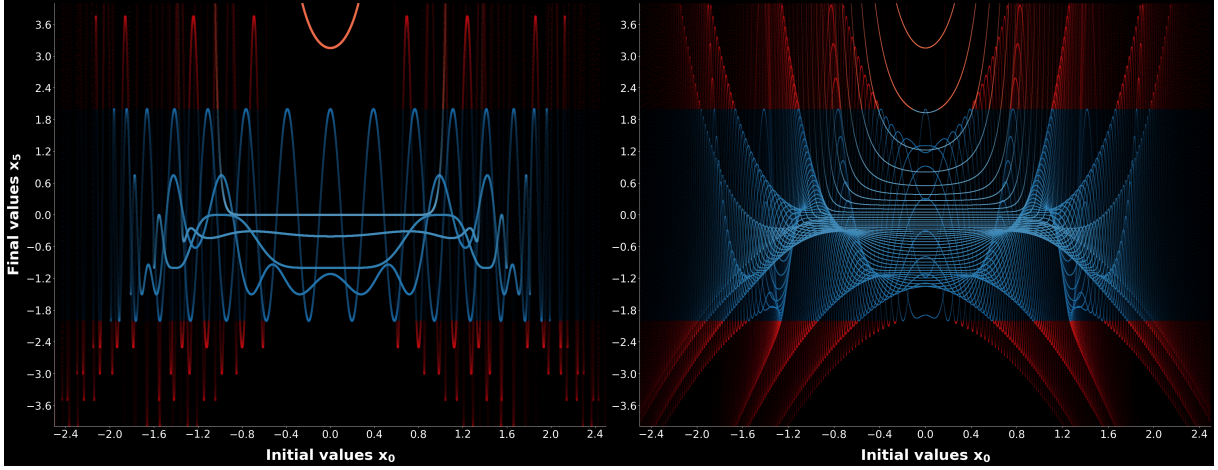
Figure 3.2: Comparison of sparse and dense iterated composition $f_c^2(x)$.



(a) Sparse steps of the 3rd Iterate

(b) Dense steps of the 3rd Iterate

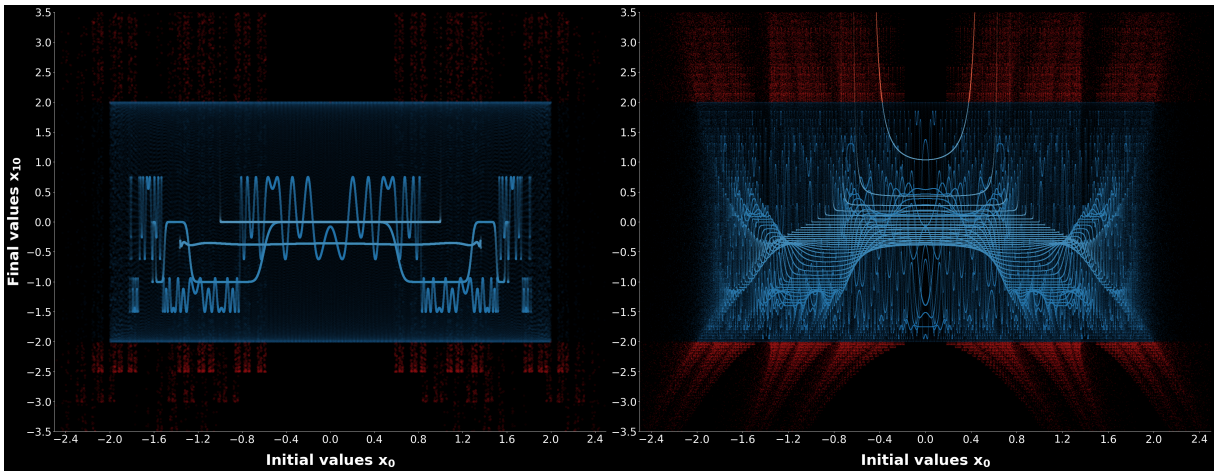
Figure 3.3: Comparison of sparse and dense iterated composition $f_c^3(x)$.



(a) Sparse steps of the 5th Iterate

(b) Dense steps of the 5th Iterate

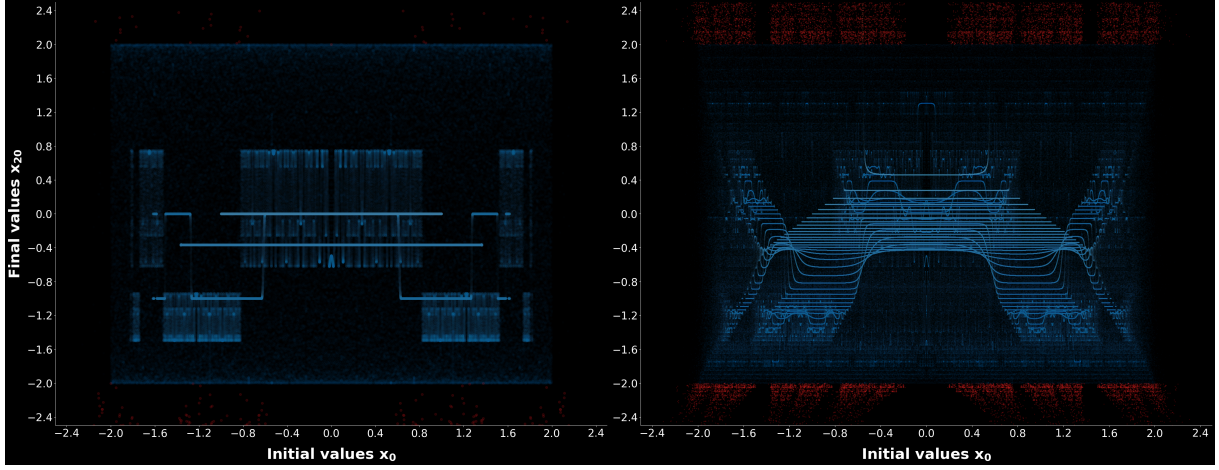
Figure 3.4: Comparison of sparse and dense iterated composition $f_c^5(x)$.



(a) Sparse steps of the 10th Iterate

(b) Dense steps of the 10th Iterate

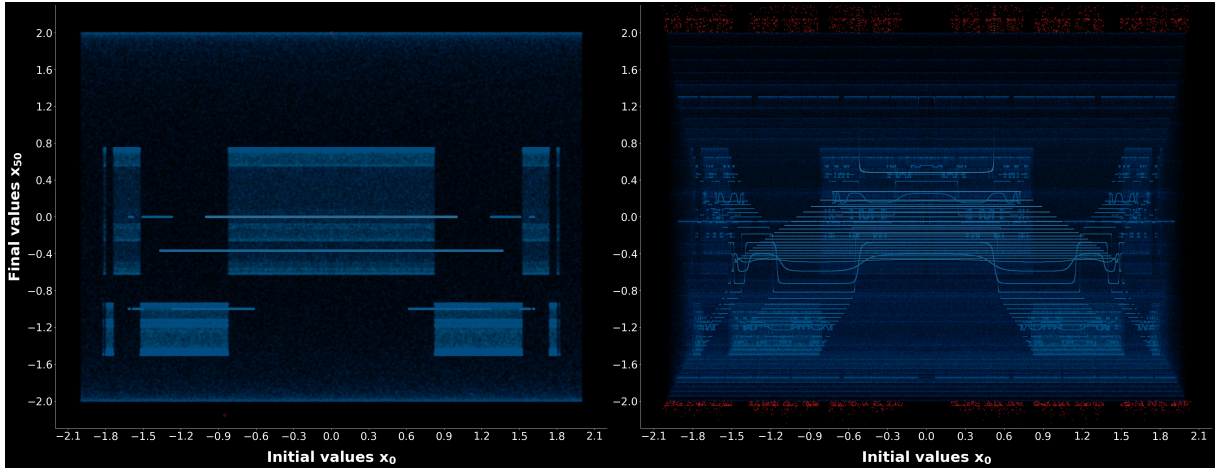
Figure 3.5: Comparison of sparse and dense iterated composition $f_c^{10}(x)$.



(a) Sparse steps of the 20th Iterate

(b) Dense steps of the 20th Iterate

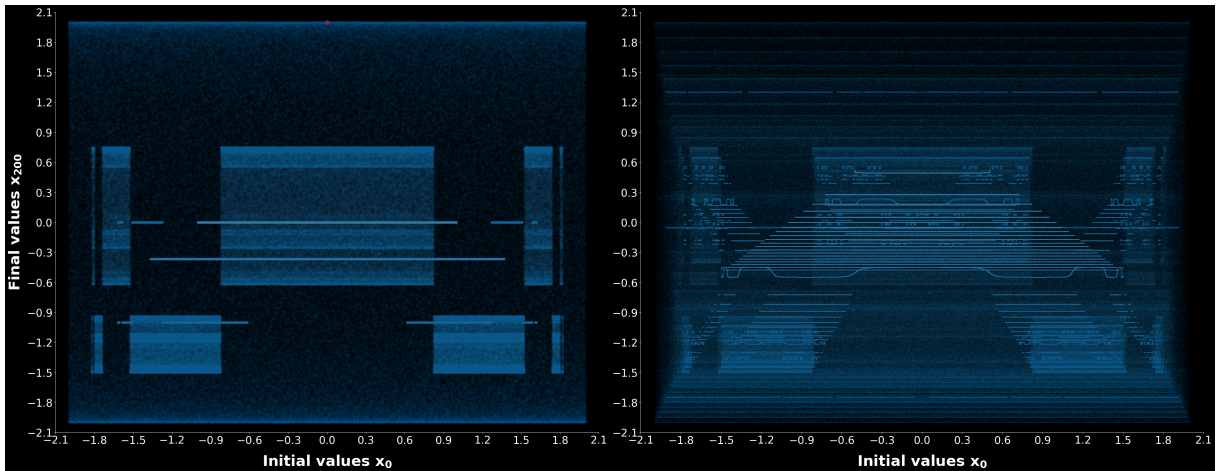
Figure 3.6: Comparison of sparse and dense iterated composition $f_c^{o20}(x)$.



(a) Sparse steps of the 50th Iterate

(b) Dense steps of the 50th Iterate

Figure 3.7: Comparison of sparse and dense iterated composition $f_c^{o50}(x)$.



(a) Sparse steps of the 200th Iterate

(b) Dense steps of the 200th Iterate

Figure 3.8: Comparison of sparse and dense iterated composition $f_c^{o200}(x)$.

4 Conclusion

The investigation into the quadratic family $f_c(x) = x^2 + c$ reveals a deterministic system with profound complexity. By mapping the local stability of the fixed point branch x_-^* , we established that the system maintains a stationary equilibrium within the interval $c \in (-0.75, 0.25)$. The transition at $c = -0.75$ represents a bifurcation point where a violation of the stability condition $|f'_c(x_-^*)| < 1$ initiates a period-doubling bifurcation cascade. This cascade, defined by the sequence $2^n \rightarrow 2^{n+1}$, is constrained by the universal scaling law known as the Feigenbaum Constant:

$$\lim_{n \rightarrow \infty} \frac{L_n}{L_{n+1}} = \delta \approx 4.669$$

Our numerical results verify that the parameter space for c undergoes such a geometric scaling governed by δ .

The emergence of the period-3 window at $c \approx -1.75$ through a tangent bifurcation serves as a critical anchor. According to Sharkovskii's Theorem, the existence of a period-3 cycle necessitates the presence of periodic orbits for all $n \in \mathbb{N}$, marking the formal transition to chaos.

The global dynamics are bounded by the Prisoner Set P_c , which serves as the locus of initial conditions x_0 that do not diverge. The escape of orbits exceeding the critical radius $R = 2$ is a consequence of the reverse triangle inequality. This ensures that the growth of the quadratic term $|x_n|^2$ dominates the parameter c :

$$|x_n| > R = 2, |c| \leq 2 \implies |x_{n+1}| = |x_n^2 + c| \geq |x_n|^2 - |c| > 2|x_n| - 2 = |x_n| + (|x_n| - 2)$$

so the increments grow at every step and $|x_n| \rightarrow \infty$. The boundary ∂P_c between bounded and escaping orbits forms the Julia Set, a fractal structure revealing detailed images as shown in the sequence Figure 3.1 through Figure 3.8. The study of these fractal boundaries demonstrates that the interplay between local instability and global boundedness creates a self-organizing landscape where order and chaos coexist [2, 1].

Within this study we remained in the domain of $\mathbb{R}$. For future research we could extend the parameter space to complex plane, e.g $c \in \mathbb{C}$ which would lead us to the Mandelbrot set and beyond.

Bibliography

- [1] Marcello Barylli. Could the future of artificial intelligence be self-organising? The Network Pages, 2026. Accessed: 2026-01-28.
- [2] Job Briet. Structure is everywhere. so is chaos. The Network Pages, 2026. Accessed: 2026-01-28.
- [3] Jordana Cepelewicz. 'entropy bagels' and other complex structures emerge from simple rules. *Quanta Magazine*, February 2024.
- [4] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Addison-Wesley, 2 edition, 1989.
- [5] Mitchell J. Feigenbaum. Quantitative universality for a class of nonlinear transformations. *Journal of Statistical Physics*, 19(1):25–52, 1978.
- [6] Tien-Yien Li and James A. Yorke. Period three implies chaos. *The American Mathematical Monthly*, 82(10):985–992, 1975.
- [7] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.
- [8] Robert M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459–467, 1976.
- [9] Valisoa Fenohasina Rabodonandrianandraina. Critical points and their role in the dynamics of rational functions. Master's thesis, University of Amsterdam, 2022.
- [10] Steven H. Strogatz. *Nonlinear Dynamics and Chaos*. Westview Press, 2 edition, 2015.
- [11] Marijke van Harten. The dynamics of the one-dimensional tent map family and quadratic family. Bachelor's thesis, University of Groningen, 2018.
- [12] Wikipedia contributors. Dynamical system — Wikipedia, the free encyclopedia, 2026. Online; accessed 2026-01-26.

```

# Code used to generate Figures 3.1-3.8
# =====
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Dark background improves visibility of the low-opacity scatter
plt.style.use("dark_background")

# == CONFIGURATIONS of the Zooming Iterated Functions ==
# =====
"""
We have here a dictionary of the configurations for every zoom. We set
the
iteration count by "iter", the center of the zoom by ("center_x", "
center_y"),
the width of the zoom by ("width_x", "width_y"), the grid resolution by
"step", and the styling parameters for the red and blue points.
These settings allows us for flexibility in the plotting process such
that we
can have the best view depending on the iteration and zoom.
"""

# == Sampling bounds and step size for c ==
# =====
C_BOUND = [-5, 5] # the bounds where c is sampled
C_STEP_COUNT = 0.5 # the step size for c; the dense figures use 0.05
instead

# =====

# Set the zoom configurations per iterate.
ZOOM_CONFIG = [
    {
        "iter": 1,
        "center_x": 0,
        "center_y": 0.0,
        "width_x": 3.75,
        "width_y": 7,
        "step": 0.0001,
        "alpha_red": 0.05,
        "alpha_blue": 0.05,
        "markersize_blue": 0.7,
        "markersize_red": 0.7,
    },
    {
        "iter": 2,
        "center_x": 0.0,
        "center_y": 0.0,
        "width_x": 3.25,
        "width_y": 6,
        "step": 0.0001,
        "alpha_red": 0.04,
        "alpha_blue": 0.04,
        "markersize_blue": 0.8,
        "markersize_red": 0.8,
    },
    {

```

```

    "iter": 3,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 3.00,
    "width_y": 5.0,
    "step": 0.0001,
    "alpha_red": 0.06,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 4,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.75,
    "width_y": 4.50,
    "step": 0.0001,
    "alpha_red": 0.06,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 5,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 4.0,
    "step": 0.0001,
    "alpha_red": 0.07,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 6,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 4.0,
    "step": 0.0001,
    "alpha_red": 0.08,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 7,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 4.0,
    "step": 0.0001,
    "alpha_red": 0.09,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
}

```

```

},
{
    "iter": 8,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 4.0,
    "step": 0.0001,
    "alpha_red": 0.1,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 9,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 4.0,
    "step": 0.0001,
    "alpha_red": 0.11,
    "alpha_blue": 0.04,
    "markersize_blue": 0.8,
    "markersize_red": 0.8,
},
{
    "iter": 10,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.5,
    "width_y": 4.0,
    "step": 0.00005,
    "alpha_red": 0.11,
    "alpha_blue": 0.04,
    "markersize_blue": 0.9,
    "markersize_red": 0.9,
},
{
    "iter": 20,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.75,
    "width_y": 3.5,
    "step": 0.00005,
    "alpha_red": 0.2,
    "alpha_blue": 0.02,
    "markersize_blue": 1.3,
    "markersize_red": 1.3,
},
{
    "iter": 50,
    "center_x": 0.0,
    "center_y": 0.0,
    "width_x": 2.50,
    "width_y": 2.5,
    "step": 0.00001,
    "alpha_red": 0.3,
    "alpha_blue": 0.015,
}

```



```

        "markersize_blue": 1,
        "markersize_red": 2,
    },
    {
        "iter": 100,
        "center_x": 0.0,
        "center_y": 0.0,
        "width_x": 2.2,
        "width_y": 2.2,
        "step": 0.00001,
        "alpha_red": 0.3,
        "alpha_blue": 0.02,
        "markersize_blue": 1.5,
        "markersize_red": 3,
    },
    {
        "iter": 200,
        "center_x": 0.0,
        "center_y": 0.0,
        "width_x": 2.1,
        "width_y": 2.1,
        "step": 0.00001,
        "alpha_red": 0.5,
        "alpha_blue": 0.02,
        "markersize_blue": 1,
        "markersize_red": 4,
    },
]

# === GLOBAL OVERWRITE ===
# We define here an global overwriting of the zoom parameters to focus
# if we
# for example want to change center we want to focus on. This allows us
# to
# not adjust every single iteration settings in the dictionary.
OVERWRITE_ENABLED = False
TARGET_CENTER_X = 1.0
TARGET_CENTER_Y = 0.0
TARGET_WIDTH_X = 1.0 # +/- on both sides
TARGET_WIDTH_Y = 1.0 # +/- on both sides

# If we overwrite the zooming, we loop over all configurations and set
# the desired center and width.
if OVERWRITE_ENABLED:
    for step in ZOOM_CONFIG:
        step["center_x"] = TARGET_CENTER_X
        step["center_y"] = TARGET_CENTER_Y
        step["width_x"] = TARGET_WIDTH_X
        step["width_y"] = TARGET_WIDTH_Y

# =====

# === Define the functions ===
# =====
# We first define the mapping we iterate
def f_c(x, c):

```

```

    return x**2 + c

# Define the function that iterates f_c for a number of composition_run
and
# returns the initial and final values
def iterating_function(func, c, composition_run, x_0_values):
    """
    Iterate f_c for a total of composition times. As we extended the
    process
    over time we need to handle potential overflows.
    """
    # We copy the input to avoid modifying the original array.
    # Then we can safely return both the input and output
    x = x_0_values.copy()

    # We specify an escaping radius to avoid overflow errors, we choose
    1e6
    escape_radius = 1e6

    # Iterate over the number of composition_run desired
    for _ in range(composition_run):
        # Only iterate values that haven't escaped to avoid overflow
        active = np.abs(x) < escape_radius # active is a boolean array
        if not np.any(active): # check if any points are left to
            iterate
            break # all points have escaped, exit early (before #
                composition_run)
        # Apply the function only to non-escaped points
        x[active] = func(x[active], c)

    return x_0_values, x # return both initial and final values

# We will zoom in as the iterated composition_run increase. Since we
plot
# a very large amount of starting values we require an adaptation in the
# grid resolution where it matters. This function creates a grid
function
# that generates an adaptive grid used in plotting.
def generate_adaptive_grid(focus, width, dense_step, outer_bound,
    sparse_step=0.05):
    """
    Generates high density points in [focus-width, focus+width]
    and low density points elsewhere.
    """
    # Define the core bandwidth where we like to have a detailed plot.
    core_min = focus - width
    core_max = focus + width

    # High resolution for the center where chaotic and interesting
    behaviour
    # occurs.
    core_x = np.arange(core_min, core_max, dense_step)

    # Low resolution for the region where less interesting behaviour is
    found.
    if core_min > -outer_bound:

```

```

        left_x = np.arange(-outer_bound, core_min, sparse_step)
    else:
        left_x = np.array([]) # empty array if no left side is existing

    if core_max < outer_bound:
        right_x = np.arange(core_max, outer_bound, sparse_step)
    else:
        right_x = np.array([]) # empty array if no right side is
                                existing

    return np.concatenate([left_x, core_x, right_x])

# === Part 3: Execution of the iterations and corresponding plotting ===
# =====
"""
We have a nested loop:
1. Loop over c_step_count to define the granularity of c values
2. Loop over composition_run to define the number of iterations
    Within these loops we have the following process:
    - (a) Unpack the Zoom Configurations
    - (b) Generate an Adaptive Grid for x_0 Values
    - (c) Define the Plot Presettings
    - (d) Loop over all Values of c in the Preset Range Set
        (i) Execute iterations by using the def_iterating_function
        (ii) Filter points within the plotting view -
        (iii) Split the values into a bounded and escaping array
        (iv) Plot the points
    - (e) Final Plot Settings and Present the Figure
"""
# Run over the different zoom_configurations set in the dictionary above
.
# Enumerate creates an index of the configs in ZOOM_CONFIG as the config
  itself
for level_idx, config in enumerate(ZOOM_CONFIG):

    # --- (a) Unpack the Zoom Configurations ---
    composition_run = config["iter"]
    center_x = config["center_x"]
    center_y = config["center_y"]
    width_x = config["width_x"]
    width_y = config["width_y"]
    stepsize_x0 = config["step"]

    # --- (b) Generate an Adaptive Grid for the Values x0 ---
    # Calculate explicit view limits
    x_min, x_max = center_x - width_x, center_x + width_x
    y_min, y_max = center_y - width_y, center_y + width_y

    # Generate grid of starting values x0. We use the preset function
    # that for the given center and width creates a dense grid to obtain
    # a high
    # resolution in the area of interest, and a sparse grid outside such
    # that
    # the computer is able to cope with the number of computations.
    x_0_input = generate_adaptive_grid(
        focus=center_x, width=width_x, dense_step=stepsize_x0,
        outer_bound=10

```

```

)

# --- (c) Obtain Plot Presettings ---
# We create a figure before plotting to avoid multiple figures.
plt.figure(figsize=(20, 14))

# Create the values of c based on the bound and step count in the
CONFIG
c_values = np.arange(C_BOUND[0], C_BOUND[1], C_STEP_COUNT)
c_min_val, c_max_val = C_BOUND[0], C_BOUND[1] # for normalization

# We create an automatic adjustment of the markersize in the scatter
plot
# based on the number of c_values. Since as the total number of
c_values
# decrease their visible presence decreases. Without the adjustment
the
# plots would be difficult to see.

# Marker scale is based on a default 500 sample values of c
markersize_scale = 500 / max(1, len(c_values))
markeralpha_scale = 0.95

# --- (d) Loop over all Values of c in the Preset Range Set ---
for c_val in c_values:

    # - (i) Execute iterations by using the def_iterating_function
    -
    # For the adapted list of x_0_input values we find the final
    values
    x_0, x_final = iterating_function(
        func=f_c, c=c_val, composition_run=composition_run,
        x_0_values=x_0_input
    )

    # - (ii) Filter points within the plotting view -
    """We only pass points to matplotlib that are actually inside
    the
    plotting view. Due to the millions of values plotted. If we did
    not set
    a criterion as such, at some point matplotlib would refuse.
    """
    # Explicitly check bounds against center_x/y +/- width_x/y
    # We ask whether x_0 and x_final are inside the limit of x_axes
    # and limit of y_axes. # It creates a boolean array of the
    points
    # within the view.
    in_view_mask = (
        (x_0 >= x_min) & (x_0 <= x_max) & (x_final >= y_min) & (
            x_final <= y_max)
    )
    # Select the x_0 and x_final values we will plot in the figure
    x_0_view = x_0[in_view_mask]
    x_final_view = x_final[in_view_mask]

    # Only continue the algorithm if there are actually points to
    plot
    if len(x_0_view) == 0:

```

```

        continue

# - (iii) Split the values into a bounded and escaping array -
# Create a boolean array of all bounded and escaping values.
# The bounds are set based on the final values after iterations.
# We have numerically determined that this value is 2, see
# Section 3.1.
is_bounded = np.abs(x_final_view) < 2
is_escaping = ~is_bounded # escaping points are those not
    bounded

# - (iv) Plot the points -
# We now plot the bounded and escaping points in different
    colors
# Based on a value c we choose a color in a color map. We
    require
# a value inbetween (0, 1)
# In the plotting we use the standard matplotlib colormaps
# plt.cm.Blues_r, plt.cm.Red_s_r for blue and red.
normed_color_value = (c_val - c_min_val) / (c_max_val -
    c_min_val)

# Bounded points are in blue.
# Note that we apply the scaling at the markersize and its
    opacity.
if np.any(is_bounded):
    plt.scatter(
        x_0_view[is_bounded],
        x_final_view[is_bounded],
        color=plt.cm.Blues_r(normed_color_value),
        s=markersize_scale * config["markersize_blue"],
        alpha=markeralpha_scale * config["alpha_blue"],
        marker="o",
    )

# Escaping points are in red.
# Note that we apply the scaling at the markersize and its
    opacity.
if np.any(is_escaping):
    plt.scatter(
        x_0_view[is_escaping],
        x_final_view[is_escaping],
        color=plt.cm.Red_s_r(normed_color_value),
        s=markersize_scale * config["markersize_red"],
        alpha=markeralpha_scale * config["alpha_red"],
        marker="o",
    )

# --- (e) Final plot settings ---
plt.axhline(0, color="black", lw=1, alpha=0.1)
plt.title(
    rf"$\bf{{f_c^{{{composition\_run}}}}(x)}}$ for $\bf{{c \in [{}$
        c_min_val}, {c_max_val}]}}$",
    fontsize=30,
    fontweight="bold",
    pad=30,
)
plt.xlabel(rf"Initial values $\bf{{x_0}}$", fontsize=20, fontweight=

```

```

    "bold")
plt.ylabel(
    rf"Final values $\bf{x}_{\{\{\{composition\_run\}\}\}}$",
    fontsize=20,
    fontweight="bold",
)

# Set explicit Limits for the zoom level as determined at the start.
plt.xlim(x_min, x_max) # (center_x - width_x, center_x + width_x)
plt.ylim(y_min, y_max) # (center_y - width_y, center_y + width_y)

# Obtain the axes which we require by using an adjustable ticks
ax = plt.gca()
# here we adjust tick density based on zoom level converted to nice
# floating
# numbers that are easy to grasp. We use MaxNLocator from the plt
# library.
# nbins = 8 indicates that we are fitting 8 ticks in the given range
# and
# steps defines the allowed intervals.
ax.xaxis.set_major_locator(ticker.MaxNLocator(nbins=15, min_n_ticks
=5))
ax.yaxis.set_major_locator(ticker.MaxNLocator(nbins=15, min_n_ticks
=5))
ax.tick_params(axis="both", labelsize=15, rotation=45)
# Adjust the spines and grid for better visibility and visual
# preference
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
ax.spines["left"].set_color("white")
ax.spines["bottom"].set_color("white")
# plt.grid(alpha=0.2)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 1.10])
plt.show()

# END OF CODE

```