Tese apresentada para obtenção do título de Doutor em Matemática pelo
Instituto de Matemática Pura e Aplicada

On interval numerical methods for dynamical systems in the plane

por

José Eduardo de Almeida Ayres

Orientador: Luiz Henrique de Figueiredo

Rio de Janeiro
January 19, 2022

PREFACE

Context. The theory of interval analysis arises from the need for a better computational representation of errors, and in his pioneering work, Moore [35, 36, 37] developed the basics of this tool. Moore's simple idea has become a powerful tool, which today has many applications in applied mathematics, computer science, and engineering.

If one thinks about classical arithmetic and how it operates on real numbers, is not too weird how interval arithmetic defines a set of operations on intervals. Interval arithmetic allows using a computer to get validated results, ensuring that the result interval contains the true solution. Tucker [43] says that interval arithmetic justifies the extension of the real arithmetic with intervals and provides an elegant means of computing with inequalities.

Ramon E. Moore conceived interval arithmetic in 1957, while an employee of Lockheed Missiles and Space Company, as an approach to bound rounding errors in mathematical computations. Forty years later, in the kick-off meeting of Sun Microsystems interval arithmetic university research and development program, he explained his thinking as follows: in 1957 he was considering how scientists and engineers represent measurements and computed results as $\tilde{x} \pm \varepsilon$, where $\tilde{x}$ is the measurement and ε is the error tolerance. While representing fallible values using the $\tilde{x} \pm \varepsilon$ notation is convenient, computing with them is not, even in a case as simple as calculating the area of a room. If the errors due to finite precision arithmetic are simultaneously taken into account, complexity increases further. Error analyses of large scientific, engineering and commercial algorithms are sufficiently complex and labor-intensive that they are often not conducted [25].

The idea to solve the problem of $\tilde{x} \pm \varepsilon$ notation developed by Moore was great. Given that the error is composed of two different numbers grouped into $\tilde{x} \pm \varepsilon$, why not use an interval defined by the two numbers to represent exactly the same information? Instead of using the $\tilde{x} \pm \varepsilon$, use $X = [\tilde{x} - \varepsilon, \tilde{x} + \varepsilon]$, which define the endpoints of an interval containing the exact quantity in question. It was this simple, yet the profound, idea that started interval arithmetic and interval analysis, the branch of applied mathematics developed to numerically analyzing interval algorithms.

Affine arithmetic is a more complex and expensive computation model, designed to give tighter and more informative bounds than interval arithmetic in certain situations where the latter is known to perform poorly. Affine arithmetic is similar to standard interval

arithmetic developed by Moore, to the extent that it automatically keeps track of rounding and truncation errors for each computed quantity. In addition, it keeps track of first-order correlations between those quantities. Thanks to this extra information, affine arithmetic is able to provide much tighter bounds for the computed quantities, with errors that are approximately quadratic in the uncertainty of the input variables. This advantage of affine arithmetic is especially noticeable in computations of great arithmetic depth or subject to cancellation errors [40].

As one may expect, the affine arithmetic model is more complex and expensive than ordinary interval arithmetic. However, for the applications developed in the first part of this thesis, we shown that using interval arithmetic is not possible, as the error bounds increase very fast and explode, even in very simple dynamical systems.

Contributions of this thesis. The objective of this thesis is to develop the application of reliable numerical methods for dynamical systems in the plane, and reliable methods in general. The thesis contains two parts, written as papers:

- Idealization, implementation, and experimentation of a range of condensation strategies to solve a trade-off in applications of affine arithmetic: the computational cost of operating with affine forms versus the precision of affine forms in terms of overestimation and wrapping effect.
- Development of a novel rigorous numerical method, based on a branch and bound algorithm and interval arithmetic, to find all fixed points of a dynamical system, with an application to finding periodic points of a complex dynamical system.

The first part of the thesis is called "Condensation strategies for affine arithmetic". It looks at the two extremes problem in affine arithmetic and gives solutions to how to operate with affine forms getting much smaller errors than using interval arithmetic without suffering from the excessive computational cost of the full affine arithmetic. The full affine arithmetic uses the total power that affine arithmetic has to offer, but this comes with unlimited use of symbols, which requires high costs. Interval arithmetic is very efficient in computational cost, but it generates a lot of wrapping effect with its axis-aligned boxes, which has a lot of problems to represent even simple geometries like a rotation map.

We can see in the literature many efforts to address this trade-off, but as yet no one has addressed this problem fully. With the work described in this part we expect to give a kick-off in the solution

for this trade-off problem, giving a range of tested condensation strategies to work with affine forms in a simple but typical application. As a final result, we suggest specific strategies that works in a wide range of problems.

The second part of the thesis is called "Interval methods for fixed and periodic points: development and visualization". It was published in the *Journal of Universal Computer Science* in 2020. In this part, we describe the development of rigorous numerical methods for finding all fixed points of a map. Our main contribution is a novel hybrid algorithm that switches to plain fixed-point iteration once it establishes the existence of an attracting fixed point using Banach's criterion before finding a small certified enclosure for the fixed point. The certified enclosure for the fixed point is constructed using a numerical method based on a branch and bound algorithm and interval arithmetic, it gives a guarantee that we have the smallest possible interval which contains the unique fixed point.

Related works. Turner's thesis [44] from 2020 (probably developed at the same time as our work which was started in 2018) develops a library for AA and uses the radical strategy to address the problem of the high-computational cost of AA. Turner calls his strategy Mixed Trimmed AA/IA (it can be called a condensation strategy by definition). It is essentially a radical strategy that simultaneously computes AA and IA versions of each range, and use the range information from either method to complement that of the other ($F(X) \cap [F(\widehat{x})]$), with the AA error trimmed by the IA range when $F(X) \subseteq [F(\widehat{x})]$, that is, when the IA range is contained in the radical condensation of the AA zonotope.

Turner's strategy was already suggested in the original AA work from Brazilian Mathematics Colloquium [40]. That work goes beyond Turner's and says that there is more in the mixed AA/IA model (AAIA) than just performing the same computation in AA and IA and intersecting the resulting ranges. The two models can and should interact synergistically at each step, with each model using the other's information to improve its own accuracy.

Messine [32, 33] tried to cut the affine forms to reduce the computational costs, using what looks like the radical strategy and the truncation strategy (see below). He also worked on a quadratic form, introducing a new error term ε^2.

Suggestions for future work. The two parts are deeply connected by a natural improvement in the second part. Using IA to solve the

existence and uniqueness rules for fixed-point generate to us a phenomenon called *wrapping effect*, which describes the lack of capability of a box to represent complex geometry and should be discussed in more detail in the work. Using AA condensation strategies from the first part, we should be able to reduce the wrapping and generate gains in computer efficiency. The new problem will have at its core developments in zonotope intersections and generating a new zonotope from them, which could be started using the oriented minimum bounding box (OMBB). This problem is very rich geometrically and connects the two pieces in an algorithm.

Acknowledgments. First of all, I want to express my gratitude to Luiz Henrique de Figueiredo for all the support in the orientation throughout my research for this thesis. He encouraged me to not give up in some troubled moments in the development of this work (which were so many), always with useful advice. I have greatly appreciated his friendly advice, and if I was able to become a better researcher and professional, he has a big part in that development.

I am grateful to Diego Nehab and Luiz Velho for the warm welcome to the Visgraf laboratory. For me, the Visgraf laboratory looks like a family composed of people from lots of places, and those three professors (including Luiz Henrique) are responsible for that. I enjoyed each moment in Visgraf, even some tough ones, like the qualification exam.

I would like to thank the long date friends that arrived with me at the Visgraf laboratory Pedro Souza and Caio Souza, as well all the new friends that I made there Daniel Yukimura, Julia Giannella, Vitor Guerra, Santiago Guisasola, Lenka Ptackova, and Ezequiel Soto — it was a pleasure to be working with them for five years. Also, Djalma Lucio was always supportive and helped me in a lot of technical problems at IMPA.

I am grateful to Francisco Moura Neto, for being my professor in undergraduate and my advisor in master's degree. He presented me to IMPA and encouraged me to flight new horizons.

My love to Vanessa Cabral, always supportive in these 15 years together. If I made it here, it was thanks to her help in many non enumerate aspects.

I lost two of the most important people for me in those three years working in this thesis, and I would like to remember them here.

My beloved grandfather raised me in such an exemplary way and gave me all the support I needed to get here today, doing next to impossible so that I had what I needed to develop well.

I'm sorry to lose my brother at such a young age, but the 24 years we were able to live together were incredible, even the final days in the hospital. He was always very visionary and a great supporter that I've had in my life, perhaps the person who believed in me more than myself. I dedicate this thesis to him.

Financial support. The author was partially supported by CNPq and FAPERJ doctoral scholarships. In addition, McKinsey and Company, who kindly offered a 2-month summer internship in the doctoral program. This research was done in the Visgraf Computer Graphics laboratory at IMPA. Visgraf is supported by the funding agencies FINEP, CNPq, and FAPERJ, and also by gifts from IBM Brasil, Microsoft, NVIDIA, and other companies.

Condensation strategies for affine arithmetic

CONDENSATION STRATEGIES FOR AFFINE ARITHMETIC

JOSÉ EDUARDO DE ALMEIDA AYRES

CONTENTS

ABSTRACT. We develop several condensation strategies for affine arithmetic and compare their performance in the numerical simulation of some discrete dynamical systems in the plane.

1. INTRODUCTION

Numerical simulations of discrete dynamical systems are subject to rounding errors in floating-point arithmetic that can affect their reliability in complex ways (§2). Interval arithmetic provides reliable computational methods, even in the presence of rounding errors, but is often subject to severe overestimation (§3). Affine arithmetic is designed to reduce overestimation in interval methods, but is subject to increasing complexity in long computations (§4). In this work, we develop strategies for reducing this complexity in the same that we avoid the introduction of overestimation (§5). We evaluate these strategies by comparing their performance in the numerical simulation of some discrete dynamical systems in the plane, generation the first roadmap in condensation strategies for AA, contributing here with a begging of a benchmark (§8).

2. DISCRETE DYNAMICAL SYSTEMS

The natural sciences seek to understand and ideally predict the evolution of a system from a given initial condition. A mathematical formulation is given by a *discrete dynamical system*: the goal is to study the long-term behavior of the iterates of a map $f\colon \mathbf{R}^d \to \mathbf{R}^d$. The evolution of the system is the *orbit* $\mathcal{O}(p)$ of the initial point $p \in \mathbf{R}^d$:

$$\mathcal{O}(p) = \{p, f(p), f(f(p)), f(f(f(p))), \ldots\}$$

Discrete dynamical systems are typically studied by numerical simulation using computers; this is the realm of computational science. An important example is numerical weather prediction [31]. Numerical simulation uses floating-point arithmetic, which is subject to rounding errors [19]. For chaotic dynamical systems, which exhibit sensitivity to initial conditions, rounding errors are potentially serious: orbits starting at nearby points can diverge from each other exponentially. Precisely this phenomenon led to the discovery of *deterministic chaos* by Lorenz [29]: even though the system evolves deterministically, its long-term behavior is unpredictable. Here is his own account of that discovery [30]:

> At one point I decided to repeat some of the computations in order to examine what was happening in greater detail. I stopped the computer, typed in a line of numbers that it had printed out a while earlier, and set it running again. I went down the hall for a cup of coffee and returned after about an hour, during which time the computer had simulated about two months of weather. The numbers being printed were

> nothing like the old ones. I immediately suspected a weak vacuum tube or some other computer trouble, which was not uncommon, but before calling for service I decided to see just where the mistake had occurred, knowing that this could speed up the servicing process. Instead of a sudden break, I found that the new values at first repeated the old ones, but soon afterward differed by one and then several units in the last decimal place, and then began to differ in the next to the last place and then in the place before that. In fact, the differences more or less steadily doubled in size every four days or so, until all resemblance with the original output disappeared somewhere in the second month. This was enough to tell me what had happened: the numbers that I had typed in were not the exact original numbers, but were the rounded-off values that had appeared in the original printout. The initial round-off errors were the culprits; they were steadily amplifying until they dominated the solution. In today's terminology, there was chaos.

Rounding errors affect numerical simulations of dynamical systems in very complex ways [6]. Well-conditioned dynamical systems may display chaotic numerical behavior [1]. Conversely, numerical methods can suppress chaos in chaotic dynamical systems [7]. On the other hand, sometimes strong sensitivity to initial conditions does not affect the overall picture, because numerically computed orbits are "shadowed" by exact orbits that capture the typical behavior of the system [22]. These phenomena raise doubts about the reliability of numerical simulations, despite the immense success of computational science. Nevertheless, there are computational methods for studying and simulating dynamical systems that work reliably even in the presence of rounding errors. The main tool for these methods is interval analysis, which we now review briefly.

3. Interval analysis

Interval analysis is the main tool for rigorous numerical computation [43]. It is based on interval arithmetic (IA), an extension of ordinary arithmetic operations and standard elementary functions to intervals [35, 36, 37].

Motivation. One of the original motivations for the creation of interval arithmetic was automatic, a posteriori, rounding error analysis of numerical computations. Starting with small intervals representing input quantities within a given precision, the output intervals are guaranteed to contain the exact output values from the given input

values. In the original thinking, if the output intervals are too wide, then the computation was contaminated with rounding errors. However, as we shall see below, rounding errors are not the main reason for large output intervals.

Interval extensions. The fundamental fact in interval analysis is that for each function $f\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}$ expressed by a formula or an algorithm, there is a computable function F automatically built from the expression of f, called the *natural interval extension* of f, such that $F(X)$ is an interval that estimates the whole range of values taken by f on a box $X \subseteq \Omega$:

$$F(X) \supseteq f(X) = \{f(x) : x \in X\}$$

Moreover, the estimate $F(X)$ approaches the exact range $f(X)$ as X shrinks to a point, in the sense that $F(\{x\}) = \{f(x)\}$ for every $x \in \Omega$. Thus, F coincides with f on degenerate boxes. Software libraries for interval arithmetic take rounding errors into account to ensure that the inclusion $F(X) \supseteq f(X)$ holds even when computed with floating-point arithmetic. It is in this sense that interval arithmetic supports rigorous numerical computation.

For a map $f\colon \mathbf{R}^2 \to \mathbf{R}^2$ given by $f = (f_1, f_2)$, with $f_i\colon \mathbf{R}^2 \to \mathbf{R}$, we get interval estimates for f by combining interval estimates for each component of f. More precisely, to estimate the values that f takes on a box $X = X_1 \times X_2 \subseteq \mathbf{R}^2$, let F_i be an interval extension of f_i. Then

$$F(X) := F_1(X) \times F_2(X) \supseteq f_1(X) \times f_2(X) \supseteq f(X)$$

and so F is an interval extension of f; it maps boxes to boxes.

Primitive operations. The primitive operations are the elementary arithmetic operations as follows

$$\begin{aligned}
X + Y &= [\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}],\\
X - Y &= [\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}],\\
X \times Y &= [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] = [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})],\\
X \div Y &= [\underline{x}, \overline{x}] \div [\underline{y}, \overline{y}] = [\underline{x} + \overline{x}] \times \left[\tfrac{1}{\overline{y}} + \tfrac{1}{\underline{y}}\right].
\end{aligned}$$

For extensions of the elementary functions to intervals, can be used the tecnics provided by Stolfi and de Figueiredo [40].

Overestimation. Finding the exact range $f(X)$ is a hard problem in general [41]. Therefore, the inclusion $F(X) \supseteq f(X)$ is usually proper and interval estimates are usually *overestimates*.

Overestimation is the main obstacle in the widespread use of interval methods in mainstream numerical computing. In long iterative

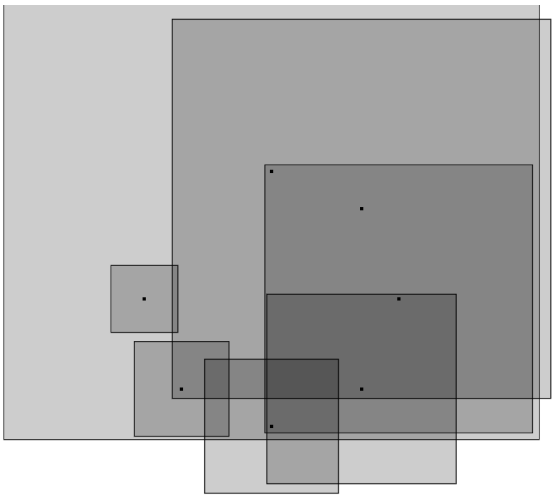


FIGURE 1. Rotation map of 45° using AA.

computations, overestimates may compound steadily and lead to uselessly large intervals.

The main source of overestimation is the occurrence of multiple instances of one or more variables in an expression. Indeed, the primitive interval operations give exact ranges only when the operands are independent. When the operands partially depend on each other, not all combinations of values in their intervals are valid and the estimated range is typically larger than the exact range. This is called the *dependency problem*. A simple example lies in the fact that, in general, $X - X \neq [0, 0]$:

$$X = [1, 2] \implies X - X = [1 - 2, 2 - 1] = [-1, 1]$$

Wrapping. To compute the orbit of a point p under f numerically using interval arithmetic, we start with a small box containing p and iterate F. The result is a sequence of boxes, each box containing the image of the previous one under f: $X_{n+1} = F(X_n) \supseteq f(X_n)$. Even when the initial box is quite small, these boxes grow in size because the box $F(X_n)$ cannot represent the curved geometry of the exact range $f(X_n)$. This phenomenon is called the *wrapping effect*. It is a facet of overestimation that comes from the inability to represent curved geometries with boxes; it does not come directly from multiple instances of variables in expressions. The wrapping effect happens even when the map f is *conservative*, that is, preserves area.

A striking example is the 45° rotation, which could be seen in the Figure 1. The initial box is the smaller square on the figure, which

has an area $A_0 = l_0^2$. If we apply AA into this square, we will rotate this until the diagonal of the initial square became the side of the second square, the new area $A_1 = l_1^2 = (\sqrt{2}l_0)^2 = 2l_0^2$. Applying sequentially AA on this rotation will double the area in each iteration, and quickly we will have a so big square that is useless in representing the expected quantity.

Success. Despite these conceptual and practical hurdles, there have been numerous successful applications of interval methods in the numerical study of discrete dynamical systems [13, 14, 15, 34, 12, 20], including striking computer-aided proofs [42].

4. Affine arithmetic

Affine arithmetic (AA) [5, 40, 10] is a model for rigorous numerical computation designed to reduce overestimation by explicitly representing first-order correlations among quantities. Like interval arithmetic, AA provides arithmetic operations and elementary functions that work on special representations of real quantities and it is able to extract estimates for the range of computed quantities from their representations.

Representation. Affine arithmetic represents a quantity q with an *affine form*:

$$\widehat{q} = q_0 + q_1\varepsilon_1 + q_2\varepsilon_2 + \cdots + q_n\varepsilon_n$$

where q_i are real numbers and ε_i are *noise symbols* which vary in the interval $[-1, 1]$ and represent independent sources of uncertainty. We call q_0 the *central value* and the other q_i *partial deviations*. Different quantities may use different sets of noise symbols. Affine forms that share noise symbols represent partially dependent quantities.

An affine form $\widehat{q}$ implies an interval estimate for the value of q:

$$q \in [\widehat{q}] := [q_0 - \delta, q_0 + \delta]$$

where $\delta = |q_1| + \cdots + |q_n|$ is the *total deviation* of $\widehat{q}$. Thus, AA generalizes interval arithmetic.

Computing in AA. There are simple formulas for operating with affine forms. Affine forms support affine operations (addition, subtraction, scalar multiplication, and scalar translation) exactly (except for rounding errors in floating-point arithmetic). The formulas for non-affine operations (multiplication, integer powers, square root, and other elementary functions) combine a good affine approximation with an explicit error term, as explained in detail elsewhere [40, 10].

Computing with affine forms is more costly than with intervals. Nevertheless, AA frequently provides better interval estimates that converge quadratically as the diameters of the input intervals shrink to zero. (IA estimates typically converge linearly.) This feature yields more efficient methods in several cases, especially when the geometry of AA approximations is exploited [9, 11, 38, 8].

Part of the cost in AA arises because AA creates a new noise symbol for each primitive operation to account for approximation errors (including rounding errors). The theme of this paper is how to manage the number of noise symbols in long computations, such as orbit computations in numerical simulations of a discrete dynamical system.

Geometry of joint ranges. Consider two quantities x and y represented by the affine forms

$$\begin{aligned}\widehat{x} &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ \widehat{y} &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n\end{aligned}$$

For convenience, we have used the same set of noise symbols in both affine forms by adding zero coefficients when necessary. Considered separately, these affine forms give interval estimates for x and y: $x \in X = [\widehat{x}]$ and $y \in Y = [\widehat{y}]$. However, their *joint range* implied by AA is the subset of plane given by

$$Z = \{(x, y) : \varepsilon_i \in [-1, 1],\ i = 1, \ldots, n\}$$

In other words, Z is the image of the hypercube $H = [-1, 1]^n$ under the affine transformation $\mathbf{R}^n \to \mathbf{R}^2$ given by

$$\begin{pmatrix}\varepsilon_1 \\ \vdots \\ \varepsilon_n\end{pmatrix} \mapsto \begin{pmatrix}x_0 \\ y_0\end{pmatrix} + \begin{pmatrix}x_1 & \cdots & x_n \\ y_1 & \cdots & y_n\end{pmatrix} \begin{pmatrix}\varepsilon_1 \\ \vdots \\ \varepsilon_n\end{pmatrix}$$

Therefore, Z is the convex hull of the images of the vertices of H and so Z is a *zonotope*: a convex polygon that is centrally symmetric with respect to the point (x_0, y_0), the image of the origin $(0, \ldots, 0) \in \mathbf{R}^n$.

The partial dependency between x and y implied by the shared noise symbols shows that Z is not simply the box $X \times Y$, but a subset of often much smaller area. More importantly, Z can have different orientations in the plane. In this sense, AA helps to mitigate the wrapping effect. In orbit computations with AA, we get a sequence of zonotopes enclosing each point of the orbit.

It would seem expensive to compute the vertices of Z explicitly, because the hypercube H has 2^n vertices. Nevertheless, Z admits a simple description of linear size: it has only $2n$ vertices, which are

found as follows [11]. Order the $2n$ vectors $\pm(x_i, y_i)$ for $i = 1, \ldots, n$ circularly around the origin. Let $u_0, \ldots, u_{2n-1}$ be the sorted list. Then, the $2n$ vertices of Z are $p_0, \ldots, p_{2n-1}$, where $p_j = (x_0, y_0) + \sum_{k=0}^{n-1} u_{j+k}$, with $j + k$ computed modulo $2n$.

Complexity. AA creates a noise symbol for each primitive operation. In long computations, the number of terms in the produced affine forms increases without bound as the computation progresses, affecting the performance of primitive operations. Thus, in orbit computations, the zonotopes increase in complexity.

Most terms in a given affine form correspond to approximation errors in previous operations. It is not worth tracking all those errors separately, especially those with small deviations; these terms imply small sides in zonotopes. Thus, it is desirable to simplify the affine forms while still ensuring that they provide reliable enclosures for the quantities they represent. The main theme of this paper is when and how to do this simplification.

Condensation. A point in an orbit of a discrete dynamical system on the plane is represented by the joint range of two affine forms corresponding to its coordinates:

$$\begin{aligned} \widehat{x} &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ \widehat{y} &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \end{aligned}$$

Individual simplification of $\widehat{x}$ and $\widehat{y}$ would destroy their correlation. Instead, we must simplify *conservatively* the zonotope Z that represents their joint range. Ideally, we would find the smallest zonotope Z^* that contains Z and has a suitably smaller number of sides. This is a delicate geometric problem, which is feasible in 2D but not in higher dimensions [21]. We prefer a simpler algebraic solution. While Z^* would correspond to the joint range of two new affine forms in potentially all new noise symbols, it is simpler to identify and keep the *important* noise symbols in the original affine forms and combine the remaining symbols. This process is called *condensation* and is detailed below. The core of this paper is the proposal and comparison of several strategies for identifying important noise symbols.

A point in Z is represented by a 2D affine form:

$$\begin{pmatrix} \widehat{x} \\ \widehat{y} \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \varepsilon_1 + \cdots + \begin{pmatrix} x_n \\ y_n \end{pmatrix} \varepsilon_n$$

which we shall write as

$$\widehat{v} = v_0 + v_1\varepsilon_1 + \cdots + v_n\varepsilon_n$$

where $v_i = (x_i, y_i)$. After reordering, we assume that the important noise symbols are the first m symbols. Then condensation is the simplification of $\widehat{x}$ and $\widehat{y}$ to

$$\begin{aligned}\widehat{x} &= x_0 + x_1\varepsilon_1 + \cdots + x_m\varepsilon_m + \rho_x\varepsilon_x\\ \widehat{y} &= y_0 + y_1\varepsilon_1 + \cdots + y_m\varepsilon_m + \rho_y\varepsilon_y\end{aligned}$$

where

$$\rho_x = |x_{m+1}| + \cdots + |x_n|, \quad \rho_y = |y_{m+1}| + \cdots + |y_n|,$$

and ε_x and ε_y are new noise symbols. The terms $\rho_x\varepsilon_x$ and $\rho_y\varepsilon_y$ condense *independently* all the information contained in the remaining terms in the original forms. Thus, $\widehat{v}$ is condensed to

$$\widehat{v} = v_0 + v_1\varepsilon_1 + \cdots + v_m\varepsilon_m + v_{m+1}\varepsilon_{m+1} + v_{m+2}\varepsilon_{m+2}$$

where we have reused $\varepsilon_{m+1} = \varepsilon_x$ and $\varepsilon_{m+2} = \varepsilon_y$ and set $v_{m+1} = (\rho_x, 0)$ and $v_{m+2} = (0, \rho_y)$. Geometrically, condensation adds (in the sense of Minkowski sums) a (presumably small) *remainder box* $[-\rho_x, \rho_x] \times [-\rho_y, \rho_y]$ to the zonotope corresponding to the first m symbols.

5. Condensation strategies

We shall now propose and discuss several strategies for performing condensation in AA. We shall compare their relative performance using numerical experiments in §8.

Our methodology is to set a *budget* N for the maximum number of noise symbols that a strategy may keep. Based on the budget, the strategy selects a *cutoff* $m \leq N$, the number of symbols it will keep, and condenses the remaining $n - m$ symbols as described in §4. By performing condensation with a fixed budget, most AA computations are restricted to affine forms of bounded length and so have bounded complexity. The various condensation strategies differ in how they choose m and in how they select m important noise symbols to keep. Moreover, since condensation has a cost, each strategy also decides *when* condensation is performed. Typically, but not always, condensation is performed for affine forms whose length n is above the budget N. For complicated maps, this may happen after every evaluation of the map; for simple maps, it may take several evaluations until the budget is exceeded.

The various condensation strategies try to balance the cost of condensation with the benefit of performing primitive AA operations on short affine forms. We start with two basic condensation strategies: use the budget indiscriminately (unbounded complexity) or use the

minimum budget required (operating the symbols in AA and then converting to IA).

Full. This strategy ignores the budget N and sets $m = \infty$. Thus, in the full strategy no condensation is ever performed, and we are dealing with unmodified AA, in which the number of terms increases without bound as the computation progresses. We shall use the full strategy as ground truth for comparing the relative performance of the strategies that involve condensation.

procedure *full*($\widehat{v}$, N)
  return $\widehat{v}$
end

Radical. This strategy sets $m = 0$ and so always satisfies the budget N. Radical condensation condenses all symbols, reducing the joint range to a box. When performed after every primitive operation, this strategy converts AA to IA. When performed less frequently, for instance at the end of an evaluation of the map f, it may yield smaller boxes than the ones computed with IA because AA will have exploited the correlations among subexpressions in f.

procedure *radical*($\widehat{v}$, N)
  return *condense*($\widehat{v}$, 0)
end

procedure *condense*($\widehat{v}$, m)
  if $m < n$ **then**
    return *docondense*($\widehat{v}$, m) {as in §4}
  else
    return $\widehat{v}$
  end
end

Trade-off. The *full* and *radical* strategies are frequently used in literature but they do not choose well inside the trade-off of complexity versus error. The *full* strategy uses the total power that AA has to offer, but this comes with indiscriminate use of symbols and potentially a high cost in very long computations. The *radical* strategy takes advantage of AA only inside the operations, but always stores an IA-like form. This strategy is efficient in the computational cost but does not use fully exploit the potential of AA. We see in literature a few efforts to address this trade-off [32, 33, 44].

Turner's thesis [44] from 2020 (probably developed at the same time as our work which was started in 2018) develops a library for AA and uses the radical strategy to address the problem of the high-computational cost of AA. Turner calls his strategy Mixed Trimmed AA/IA (it can be called a condensation strategy by definition). It is essentially a radical strategy that simultaneously computes AA and IA versions of each range, and use the range information from either method to complement that of the other ($F(X) \cap [F(\widehat{x})]$), with the AA error trimmed by the IA range when $F(X) \subseteq [F(\widehat{x})]$, that is, when the IA range is contained in the radical condensation of the AA zonotope.

Turner's strategy was already suggested in the original AA work from Brazilian Mathematics Colloquium [40]. That work goes beyond Turner's and says that there is more in the mixed AA/IA model (AAIA) than just performing the same computation in AA and IA and intersecting the resulting ranges. The two models can and should interact synergistically at each step, with each model using the other's information to improve its own accuracy.

Messine [32, 33] tried to cut the affine forms to reduce the computational costs, using what looks like the radical strategy and the truncation strategy (see below). He also worked on a quadratic form, introducing a new error term ε^2.

Here we want to discuss condensation strategies that go deeper on the affine forms and use the budget efficiently, not just using the first symbols. Our condensation strategies are divided into two main groups; each group is defined by how we select the cutoff m. The cutoff can be statically selected based on the budget N. These strategies are called *truncate* variants. On the other hand, we will select the cutoff dynamically using a heuristic called cascade [26, 27, 28]. These strategies are called *cascade* variants.

6. Truncate variants

These variants select a static number of symbols m as the cutoff per iteration and perform the condensation. We shall construct a strategy that is a generalization of the two extreme strategies (*full* and *radical*). The first strategy has a cutoff between $m = 0$ and $m = \infty$. Yet it looks simple, this use of the cutoff variable is insightful, and a kick start to solve the trade-off of complexity versus error.

Truncation. The simplest condensation strategy is to set $m = N$ and naively select the first m symbols as important. This naive selection has zero cost, but it does not take into account the relative importance

of the terms. In particular, the remainder box may dominate the resulting zonotope.

procedure *truncate*($\widehat{v}$, N)
  if $n > N$ **then**
    return *condense*($\widehat{v}$, N)
  else
    return $\widehat{v}$
  end
end

Delayed truncation. We expect that plain truncation will quickly suffer the wrapping effect, if performed every time $n > N$. To make it more competitive, we reduce the cutoff to $m = \lceil N/2 \rceil$ to delay the next truncation, thus reducing the frequency of condensations and the size of the remainder box. This strategy allows affine forms to grow undisturbed until they reach the budget again.

procedure *truncateDelayed*($\widehat{v}$, N)
  if $n > N$ **then**
    return *condense*($\widehat{v}$, *ceil*(N/2))
  else
    return $\widehat{v}$
  end
end

Maximum norm. Truncation is cheap, but naive, because it selects the first m symbols as important, disregarding the relative importance of the terms. We expect that the important terms are the largest ones. More precisely, in $\widehat{v} = v_0 + v_1\varepsilon_1 + \cdots + v_n\varepsilon_n$ the important terms are the ones where $\lambda_i = \|v_i\|$ is large. (Since all norms are equivalent, the choice of norm does not really matter. We use the 1-norm because it is easiest to compute.) The motivation here is that the shape of the zonotope is mostly determined by its longer sides, which correspond to the longer vectors in the 2D affine form.

The maximum norm strategy sets $m = N$ and reorders the terms so that $\lambda_1, \ldots, \lambda_m$ are the largest m lengths. A simple way is ensure this is to order all terms in decreasing length, so that $\lambda_1 \geq \cdots \geq \lambda_n$. Then it truncates the form at the first m symbols as before. This strategy tends to generate much smaller remainder boxes and smaller resulting zonotopes with larger sides. The additional cost is ordering the terms, which is a non-trivial cost. Although sorting is not strictly

necessary for selecting the largest m lengths, it is quite practical for moderate budgets.

procedure *maxnorm*($\widehat{v}$, N)
  if $n > N$ **then**
    return *condense*(*order_by_norm*($\widehat{v}$), N)
  else
    return $\widehat{v}$
  end
end

Delayed maximum norm. To avoid the cost of sorting the terms at every condensation, this variant proceeds as in delayed truncation and sets $m = \lceil N/2 \rceil$. It also has the same rationale: to reduce the frequency of condensations and the size of the remainder box.

procedure *maxnormDelayed*($\widehat{v}$, N)
  if $n > N$ **then**
    return *condense*(*order_by_norm*($\widehat{v}$), *ceil*(N/2))
  else
    return $\widehat{v}$
  end
end

7. Cascade variants

Apparently unaware of affine arithmetic, Kühn [26, 27, 28] proposed the use of zonotopes for controlling the wrapping effect. He used ad hoc methods to compute functions on zonotopes getting zonotopes as results. In contrast, AA does that automatically. The main contribution of Kühn is the *cascade heuristic* for reducing the complexity of zonotopes.

Cascade. The *cascade heuristic* serves as direct inspiration for a condensation strategy. Given a 2D affine form $\widehat{v} = v_0 + v_1\varepsilon_1 + \cdots + v_n\varepsilon_n$, consider the relative sizes of the $\lambda_i = \|v_i\|$ as follows. Given the budget N and looking at the affine form from back to front, let l be the smallest integer such that $1 \le l \le N$ and $\lambda_l < \lambda_{l+1} + \cdots + \lambda_n$. If no such integer exist, take $l = N + 1$. Having found l, set $m = l - 1$, thus condensing the terms $v_l\varepsilon_l + \cdots + v_n\varepsilon_n$.

procedure *cascade*($\widehat{v}$, N)
  $\lambda \leftarrow 0$
  $l \leftarrow N + 1$
  for $i = n$ **downto** 1 **do**

```
    if i ≤ N and λ > λ_i then
      l ← i
    end
    λ ← λ + λ_i
  end
  return condense(v̂, l − 1)
end
```

Consider the extreme case $l = N + 1$. When $n \leq N$ and we are within budget, no condensation is performed because $m = l - 1 = N \geq n$. When $n \geq N$ and we are over budget, we get $m = l - 1 = N \leq n$ and cascade condensation is similar to the truncation strategy. When l gets a value assigned in the loop, we have $l \leq n$ and so cascade condensation is again similar to truncation.

While cascade condensation performs no sorting, it can create an increasing order in the terms after some iterations; the partial deviations in the condensed terms will be the least significant ones. Intuitively, symbols are combined and moved to the left, like a cascading waterfall, if we look at the iterations as time that water flows. We can see the water flow in Figure 2, where this flow can pass each cascade if the heuristic is satisfied. Each cascade surpassed accumulates a new pressure λ_i to the flow, so it can be used in the next attempt to overcome a new cascade, until it stops in a lake.

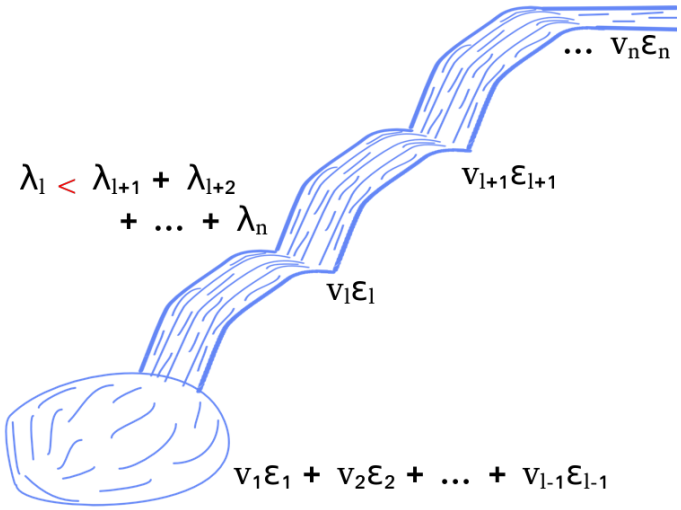


FIGURE 2. Ilustration of the cascade strategy as a waterfall and the water flow.

Relaxation cascade. The cascade heuristic always applies the same rule in the cutoff selection $\lambda_l < \lambda_{l+1} + \cdots + \lambda_n$. This rule has a bad effect on budget usage, making the cutoff $l - 1$ with a superior boundary independent of the budget N, when N is sufficient large.

The idea here is force a usage of the mean budget (using a cutoff of $\approx N/2$) and this is done with a *relaxation factor* μ, which acts over the heuristic as

$$\mu\,\lambda_l < \lambda_{l+1} + \cdots + \lambda_n$$

This idea was borrowed from the numerical linear algebra method of successive over-relaxation (SOR), which is a variant of the Gauss-Seidel method for solving a linear system of equations, resulting in faster convergence. Here we do not want faster convergence, but control on choosing an m that makes good use of the given budget. In the end we want to be next to the mean budget; thus we divide our search parameter space for μ using three rules:

(1) $N/3 \leq n \leq 2N/3$: the size of affine form is in the mean budget zone, that is we are close to $m = N/2$, then we perform no alteration on the μ factor.
(2) $n < N/3$: the size of affine form is getting smaller than mean budget zone, then we update the relaxation factor $\mu = \mu + \delta\mu$.
(3) $n > 2N/3$: the size of affine form is getting larger than mean budget zone, then we update the relaxation factor $\mu = \mu - \delta\mu$.

Here, $\delta\mu$ is a parameter for the variation on μ in each iteration of heuristic, and the starting μ is equal to one. This strategy with generalizes the cascade strategy when $\delta\mu = 0$, then no variation on μ will be performed.

procedure *relaxCascade*$(\widehat{v}, N, \delta\mu)$
  $\lambda \leftarrow 0$
  $l \leftarrow N + 1$
  for $i = n$ **downto** 1 **do**
    if $i \leq N$ **and** $\lambda > \mu\lambda_i$ **then**
      $l \leftarrow i$
    end
    $\lambda \leftarrow \lambda + \lambda_i$
  end
  if $n < N/3$ **then**
    $\mu \leftarrow \mu + \delta\mu$
  else if $n > 2N/3$ **then**
    $\mu \leftarrow \mu - \delta\mu$
  end

```
  return condense(v̂, l − 1)
end
```

Magnitude cascade. The cascade heuristic is somewhat sensitive to the actual lengths of the partial deviations. In this variant, we compare orders of magnitude instead, by replacing $\lambda_l < \lambda_{l+1} + \cdots + \lambda_n$ with

$$\lfloor \log_{10}(\lambda_l) \rfloor < \lfloor \log_{10}(\lambda_{l+1} + \cdots + \lambda_n) \rfloor$$

We expect that this variant is more stable, in the sense that it condenses symbols less often and condensations are frequently generated with fewer symbols.

```
procedure magCascade(v̂, N)
  λ ← 0
  l ← N + 1
  for i = n downto 1 do
    if i ≤ N and ⌊log10(λ)⌋ > ⌊log10(λi)⌋ then
      l ← i
    end
    λ = λ + λi
  end
  return condense(v̂, l − 1)
end
```

Computing $\lfloor \log_{10}(\lambda) \rfloor$ is somewhat costly in C. To make this more efficient, and using a similar logic, we use $\mathrm{frexp}(\lambda)$ instead. The frexp function breaks a floating point number into its normalized binary fraction, i.e. a floating point with an absolute value in interval $[0.5, 1.0)$, and an integer exponent for base 2. Thus, instead of using the base 10 exponent, we will get the base 2 exponent. The function for magnitude cascade is rewritten as follows:

```
procedure magCascade(v̂, N)
  λ ← 0
  l ← N + 1
  for i = n downto 1 do
    if i ≤ N and frexp(λ) > frexp(λi) then
      l ← i
    end
    λ = λ + λi
  end
  return condense(v̂, l − 1)
end
```

This change can be easily combined with the relaxation factor μ as follows.

procedure *RelaxMagCascade*($\widehat{v}$, N, $\delta\mu$)
  $\lambda \leftarrow 0$
  $l \leftarrow N + 1$
  for $i = n$ **downto** 1 **do**
    if $i \leq N$ **and** frexp(λ) $>$ frexp($\mu\lambda_i$) **then**
      $l \leftarrow i$
    end
    $\lambda = \lambda + \lambda_i$
  end
  if $n < N/3$ **then**
    $\mu \leftarrow \mu + \delta\mu$
  else if $n > 2N/3$ **then**
    $\mu \leftarrow \mu - \delta\mu$
  end
  return *condense*($\widehat{v}$, $l - 1$)
end

Semi-adaptive cascade. In this variant, we perform cascade condensation only when $n > N$. Thus, we can have from radical condensation, when $m = 0$, to truncation, when $m = N$. This strategy generalizes the delayed truncation strategy, because it can delay to any number of symbols, instead of fixing the delay to $N/2$.

procedure *cascadeSA*($\widehat{v}$, N, $\delta\mu$)
  if $n > N$ **then**
    return *relaxCascade*($\widehat{v}$, N, $\delta\mu$)
  else
    return $\widehat{v}$
  end
end

Semi-adaptive magnitude cascade. The same restriction can be applied in the magnitude cascade strategy to give a semi-adaptive magnitude cascade strategy.

procedure *magCascadeSA*($\widehat{v}$, N, $\delta\mu$)
  if $n > N$ **then**
    return *relaxMagCascade*($\widehat{v}$, N, $\delta\mu$)
  else
    return $\widehat{v}$
  end

end

Ordered cascade. The natural ordering that cascade induces on the symbols is efficient and shows good results. Nevertheless, if we force a full ordering before performing the condensation in semi-adaptive cascade, we expect to create a small remainder box, and frequently ordered cascade generates less wrapping effect than the semi-adaptive cascade.

When the condensation is performed, we can have from a radical condensation, with $m = 0$, to a maximum norm condensation with $m = N$. This strategy is a generalizes the delayed maximum norm strategy, because it can delay to any number of symbols, instead of fixing the delay to $N/2$.

procedure *ordCascade*($\widehat{v}, N, \delta\mu$)
  if $n > N$ **then**
    return *relaxCascade*(*order_by_norm*($\widehat{v}$), $N, \delta\mu$)
  else
    return $\widehat{v}$
  end
end

Ordered magnitude cascade. The same ordering from ordered cascade can be performed in the semi-adaptive magnitude cascade to give an ordered magnitude cascade strategy.

procedure *magOrdCascade*($\widehat{v}, N, \delta\mu$)
  if $n > N$ **then**
    return *relaxMagCascade*(*order_by_norm*($\widehat{v}$), $N, \delta\mu$)
  else
    return $\widehat{v}$
  end
end

How to choose the best strategy? Sadly there is not a simple answer to this question, for we developed more than two digits in the number of condensation strategies. The best way to set a good strategy for a specific problem will be experimenting with these problem conditions, and this is practical only if you have many instances of the problem you want to solve. The objective of the next sections of this work is to generate a long and complete (but not exhaustive) set of experiments to work as a first guideline on this fundamental question.

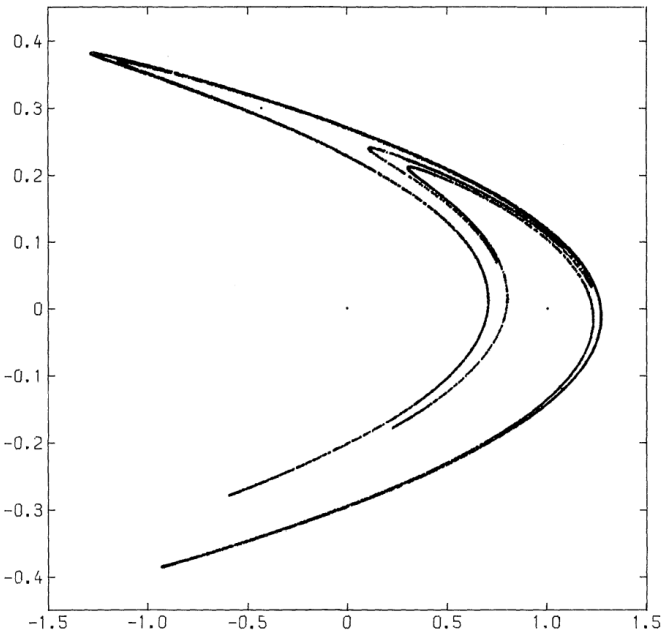


FIGURE 3. 10000 successive points obtained by iteration of the Hénon map, starting from $x = 0.0, y = 0.0$ [24]

8. NUMERICAL EXPERIMENTS

Performing numerical experiments is crucial to generate insights about the behavior of dynamical systems. However, a central question must be faced when utilizing numerical results: In what sense do the numerical experiments, with their inherent computer rounding errors, reflect the true dynamics of the actual system? [22]

Simple dynamical systems are easy and fast to be simulated in computers with floating point, but we do not expect reliability in all cases, even for simple systems. Even when the numerical orbit is shadowed by a true orbit, we cannot fully trust the numerical solution.

The problem reaches another level of difficulty when we have chaotic behavior, and the rounding errors are amplified very fast by the simulation. Different orbits, even when starting close together (sometimes differing by a rounding error, as we have seen in the Lorenz account), will move apart rapidly. This is the nature of chaotic dynamical systems, and the classic example is the strange attractor discovered by Hénon [24] in 1976. In this case, we do not even have the shadowed orbits.

IA and AA can reliably simulate a chaotic orbit in a dynamical system, but it raises other problems to us. A first observation is that IA suffers a lot with overestimation and wrapping; we will see this in experiment 9.3. When we try to use AA to overcome the IA problems, we run into other problems. To follow a long orbit for a dynamical system using AA, we generate a lot of new symbols for the affine forms (one for each non-affine operation). Then, this problem quickly gets a computer cost higher than we are willing to pay. Condensation strategies aim to keep solutions with AA at bounded complexities. Analyzing the performance on controlling wrapping effect and time efficiency of the strategies is a good way to validate the strategies.

The first set of experiments (§9) considers the computation of long orbits for a map, and here we put to test all the strategies developed in §6 and §7. In the second set of experiments (§10), we work with the Hénon map (§10.1), which is a contraction map. Here, we have an objective to find periodic orbits and evaluate the strategies for a contraction map.

The numerical experiments were executed in a Dell G7 laptop with the following configurations:

- OS: Ubuntu 18.04.2 LTS 64-bit
- CPU: Intel Core i7-8750H @ 2.20GHz x 12
- GPU: NVIDIA GeForce GTX 1060 (6GB GDDR5)
- RAM: 16GB (2x8GB) 2666MHz DDR4
- SSD: 256GB
- HD: 1TB 5400 rpm

To perform this comparative analysis, we need a set of tools to measure the efficiency of each strategy. These tools will make it easier to verify the usability of strategies and to clarify which strategy to use in each situation.

Total deviation extension. An easy and fast way to compare the evolution of a 1-dimensional affine form $\widehat{q} = q_0 + q_1\varepsilon_1 + q_2\varepsilon_2 + \cdots + q_n\varepsilon_n$ is to analyze its total deviation $\delta(q) = |q_1| + \cdots + |q_n|$. However, it is not clear how to use it for a higher dimensional affine form.

First, we need to define the norm for a 2-dimensional affine form, which was used for construction of maximum norm strategy, and is defined as

$$\lambda_i = \|v_i\|_1 = \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\|_1 .$$

Based on the 2-dimensional norm, we can extend the definition of total deviation. This construction is made in the sense that it will

reflect all diameter variation in the zonotope construction. We extend the total deviation δ for 2-dimensional affine form, as follows:

$$\delta = \lambda_1 + \lambda_2 + \cdots + \lambda_n = \sum \lambda_i = \|\lambda_i\|_1 ,$$

which gives us a clear notion of which affine form is bigger. Using the l_1 norm here gives to us a linearity on the function where we could decompose

$$\delta(\widehat{v}) = \delta(\widehat{x}, \widehat{y}) = \delta(\widehat{x}) + \delta(\widehat{y})$$

and extend to higher dimensions.

β-*competitiveness.* The total deviation is effective to verify, in an absolute way, how a strategy is performing. Nevertheless, based on Kuhn work [28], we can construct a relative metric that takes into account the performance of the full strategy, which is our ground truth.

Given a map f, suppose δ^i_{full} as the total deviation for the full condensation strategy, and δ^i_{cond} as the total deviation for a given condensation strategy, in f^i, the ith iterate of f. Then, we say that the condensation strategy is β-*competitive* with the full strategy, at iteration i, when

$$\delta^i_{\text{cond}} \leq \beta \delta^i_{\text{full}},$$

where $\log_{10}(\beta)$ is called the *competitive ratio*. If β does not exist, i.e., the condensation strategy blows up, then this strategy is not competitive at iteration i.

Note that the full strategy has strictly $\beta = 1$. The full strategy, by definition, has the smallest zonotope between all reliable strategies, then $\beta \geq 1$. The lower the values of β, the more competitive a condensation strategy is. With the full strategy as ground truth, we can perform a fairer comparison between two strategies.

9. Numerical results with long orbits

In the first set of experiments, we want to validate which AA strategy are useful in finding long orbits for dynamical systems. In this section, we will see the main results for all the condensation strategies and why they are our choice for each budget range. We performed a large set of numerical experiments to validate the choice for best strategies; they are described in detail in the appendix (note that the three appendix sections are sequential and have interdependence):

- Appendix A details the experiments with truncate variants, looking for the best strategies.

- Appendix B details the experiments with cascade variants (with delayed maximum norm strategy, the best truncate variant strategy), looking for the best strategies.
- Appendix C details the experiments with best condensation strategies, validating them.

As we have developed a large number of theoretical condensation strategies, we need to determine which ones are worth using in practice. For these experiments, we follow Hénon [23] and chose the Cremona map, a simple map that exhibits typical complex behaviors of dynamical systems. Since this map is conservative, the expectation is to see a constant size of the initial interval over the iterations. However, we know that we suffer from wrapping effect, and condensation strategies normally amplify that.

9.1. **Cremona map.** Our test problem is the quadratic map given by

$$\begin{aligned} x &\leftarrow x\cos\alpha - (y - x^2)\sin\alpha \quad \text{and} \\ y &\leftarrow x\sin\alpha + (y - x^2)\cos\alpha, \end{aligned} \tag{1}$$

where $\alpha \in \mathbf{R}$ is a constant. This is an area-preserving map $\mathbf{R}^2 \to \mathbf{R}^2$. It is arguably the simplest nontrivial mapping, and yet it is typical [23]. The mapping is called an "entire Cremona transformation". For sake of simplicity, we shall call this quadratic map as Cremona map C.

The Cremona map C is a product of two simple maps. First, a vertical shearing map S by a factor $-x$:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y - x^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -x & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

A rotation R by the angle α:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x'\cos\alpha - y'\sin\alpha \\ x'\sin\alpha + y'\cos\alpha \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

Then, the Cremona map $C = RS$, which leads us to Equation 1. As we shall see in the numerical experiments, even though the Cremona map looks simple, it exhibits the complicated properties of typical dynamical systems.

Normally, near the origin, the quadratic term will be dominated by the linear terms and we will have something similar to a rotation, leading to bounded orbits. However, for some values of α we will not have this behavior. At great distances from the origin, quadratic

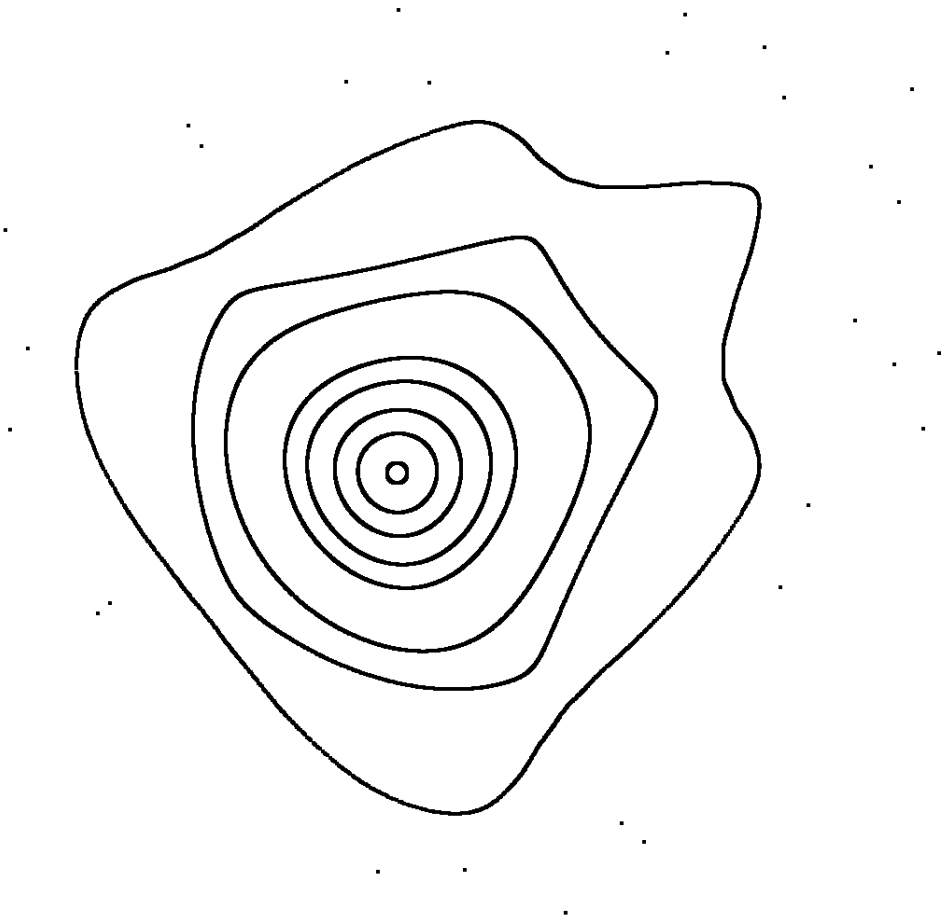


FIGURE 4. Reliable level curves for Cremona map with $\cos(\alpha) = 0.24$.

terms become dominant [23] and there we have approximately

$$x_{i+1} \approx x_i^2 \sin\alpha$$

$$x_{i+j} \sin\alpha \approx (x_i^2 \sin\alpha)^{2^j}.$$

Then, if $|x_i| > 1/\sin\alpha$, we have $x_{i+j} \to \infty$ as $j \to \infty$. This means that the orbit will escape to infinity. Given the square exponent in this escape motion, the distance to the origin is practically squared by every iteration. We can see an example of these behaviors in Figure 4[1],

[1]All orbit figures (if not explicitly stated another procedure of construction) were generated using the AA condensation procedure with delayed maximum norm strategy. The zonotopes are small as a point, it is the reason that then looks as

where, next to origin, we got a behavior similar to a rotation, and when $|x|$ is sufficient large, the orbit escapes.

Given the rotation parameter α, we have a one-parameter family. Instead of considering just one value of α, we have a lot of different maps to experiment with by varying α. Then, this map provides a variety of situations to test different interesting phenomena of dynamical systems. We define Cremona α-map as the Cremona map with a specific value for α defined.

Besides the variation of α for Cremona α-map, we have to look at the initial points for the iterations, as this will impact the behavior of the orbits. To reduce the degrees of freedom in the experiments, we will fix a value for α and change the initial values. Even with a fixed value for α, we will create a large set of different behaviors to test the condensation strategies.

9.2. **Orbits in Cremona α-map with $\cos(\alpha) = 0.24$.** For $\cos(\alpha) = 0.24$, the Figure 5 shows the Cremona map reliable level curves, that is the curve was generated by AA condensation procedure using delayed maximum norm strategy, but the zonotopes are so small that they look like points. In this figure, we can see some characteristics of this Cremona α-map.

First, near the origin, the map really looks like a rotation map. When the distance from the origin grows, the map loses the rotation aspect. This will be the first example, with initial point $x_0 = -0.38, y_0 = 0.0$, which can be seen in Figure 6.

On each curve, the arrangement of the points is topologically the same as if the mapping were a simple rotation by a given angle ω. Hénon [23] says:

> When $\omega/2\pi$ goes through a rational value q_1/q_2, with q_1 and q_2 integers, the curve breaks into a string of q_2 closed curves or "islands"; Inside each island, there is a stable invariant point of C^{q_2}, and between two neighbouring islands, where they almost join, there is an unstable invariant point of C^{q_2}. Successive points jump from one island to another by application of the mapping. If we consider C^{q_2} as the elementary mapping instead of C, each island will present the same structure, on a reduced scale, as the whole mapping; in particular it will contain second-order islands, and so on, so that the whole picture is of incredible complexity.

---

floating-point iteration, but they were constructed with guarantee that the true orbit point lies in the small generated zonotope

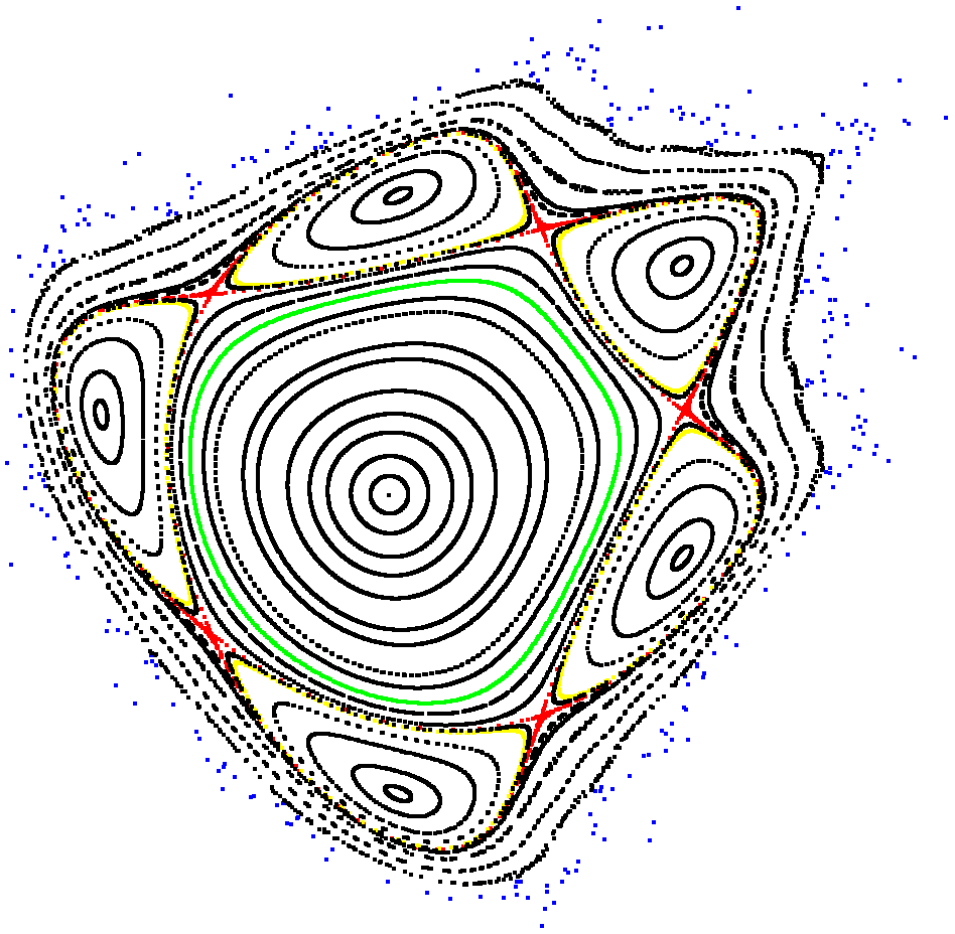


FIGURE 5. Reliable level curves (denser than Figure 4) for Cremona map with $\cos(\alpha) = 0.24$. The colored orbits represent the interest behaviors that will be analyses in the experiments.

An example can be seen in Figure 5 and the level curve alone in Figure 7, where we clearly see 5 first order islands. Here, we have $q_1 = 1$ and $q_2 = 5$. Inside the islands, there is a stable periodic point with period $p = 5$, which means that C^5 has a fixed point. Also, we have second-order islands in Figure 8 and third-order islands in Figure 10.

These islands are surrounded by a region of scattered points, next to the unstable fixed points. This scattered region can be seen in Figure 11, and has a higher accumulation of points next to this fixed

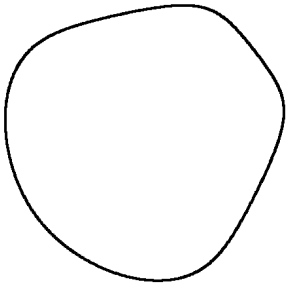


FIGURE 6. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = -0.38, y_0 = 0.0$.

points. This scattered region does not diverge, for it is constrained to remain in a region of the plane, by an outside curve.

As the last experiment, in Figure 12 we can see an escaping orbit. It is proved by AA that this orbit touches the escape region after 286 iterations, as $|x| > 1/\sin(\alpha)$ in the entire zonotope.

In summary, we have six experiments listed as:

- Deformed rotation with the initial point $x_0 = -0.38, y_0 = 0.0$;
- First-order islands with the initial point $x_0 = 0.57, y_0 = 0.12$;
- Second-order islands with the initial point $x_0 = 0.57, y_0 = 0.125$;
- Third-order islands with the initial point $x_0 = 0.57, y_0 = 0.13$;
- Bounded scatter region with the initial point $x_0 = 0.718, y_0 = 0.0$;
- Escaping region with the initial point $x_0 = 0.82, y_0 = 0.0$.

Which will be described with more details in the next sections.

9.2.1. *Deformed rotation experiment.* The initial point $x_0 = -0.38, y_0 = 0.0$, which is relatively close to origin, has an orbit that resembles a rotation, but with some deformation. Our distance to origin in this initial point is not very small, and so the effect of quadratic term begins to deform the rotation, as we can see in Figure 6.

9.2.2. *First-order islands experiment.* Looking at Figure 5, we can see a region with a very different behavior from a deformed rotation. This region is composed by five closed curves, or islands, that can be seen in Figure 7. Now the effect of quadratic terms are higher, and we expect a bigger wrapping effect due to the non-affine approximations. We shall not see long orbits as we have seen before, as this new problem is harder.

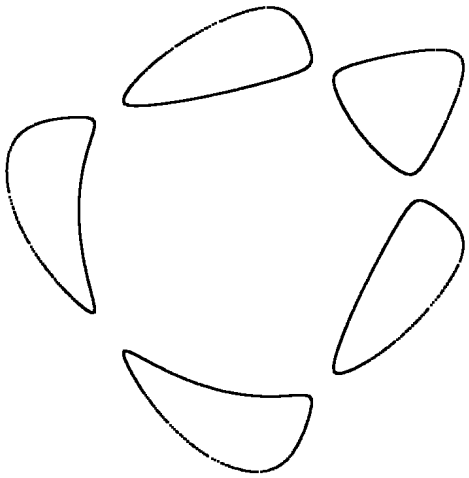


FIGURE 7. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.57, y_0 = 0.12$.

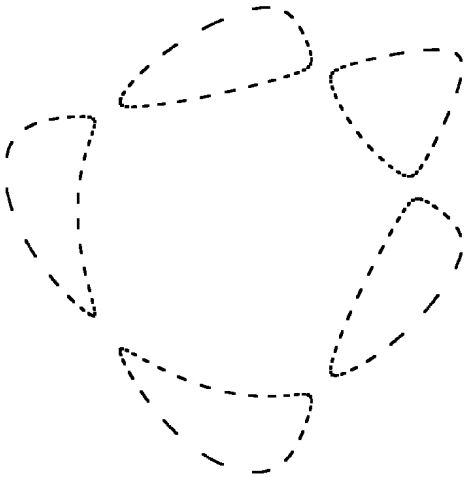


FIGURE 8. Zoomed Cremona map with $\cos(\alpha) = 0.24$.

9.2.3. *Second-order islands experiment.* Now we shall work with the second order islands, the orbit is on Figure 8. As mentioned by Hénon [23], we could see a fractal behavior in this problem. If we consider the mapping as C^5 instead of C, we will see each island surrounded by the second order islands, as we can see in the zoomed Figure 9. The red curve is the bottom first order curve in Figure 7, which is clearly surrounded by the second orders curves from Figure 8.

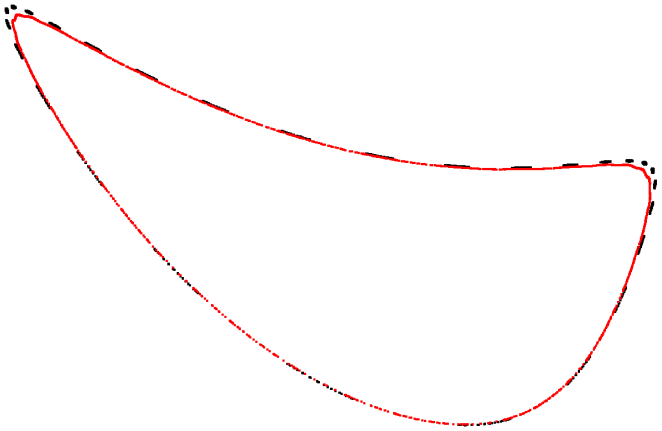


FIGURE 9. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.57, y_0 = 0.125$, in red the first-order island.

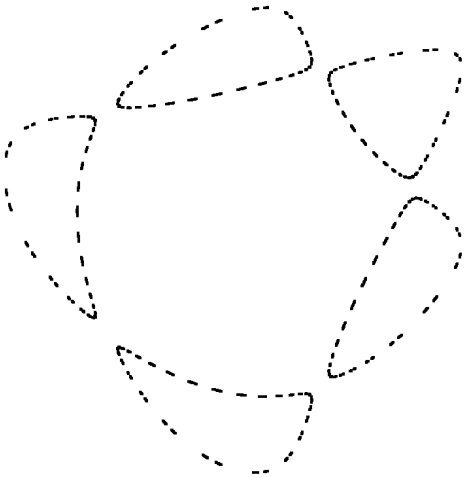


FIGURE 10. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.57, y_0 = 0.13$.

9.2.4. *Third-order islands experiment.* The orbit is in Figure 10 looks like the orbits that are in Figure 8, but we have more islands. This initial point generates the orbit composed by the third-order islands.

9.2.5. *Bounded scatter region experiment.* Now, we will work with the scattered region that can be seen in Figure 11. According to Hénon [23], this is interesting because there are invariant curves surrounding the five islands, so that the scattered set cannot escape to infinity: it

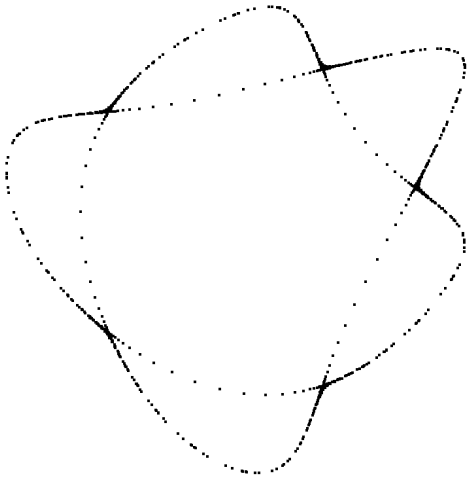


FIGURE 11. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.718, y_0 = 0.0$.

FIGURE 12. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.82, y_0 = 0.0$.

is constrained to remain in a finite region of the plane, limited by an inner curve, an outer curve, and the islands.

9.2.6. *Escaping region experiment.* The final experiment with this Cremona α-map, we shall explore an escaping region, as we can see in

Figure 12. In this escaping orbit, it was proved by AA that this orbit escapes after 286 iterations, details in §C.5.

9.3. **Difference between IA and AA radical condensation strategy.** First of all, in Figure 13, we can see the behavior of the problem by removing the shear map S, and working only with the rotation map R. Since R is an affine transformation, the AA operations are exact in this map.

Starting with a large interval, we can see in Figure 13 (b) how AA explores geometry, by perfectly rotating our interval. As IA has a fixed geometry of AABB, we have wrapping effect, and rapidly the problems diverge to another orbit, even with a much better initial interval precision (Figure 13 (a)).

Now looking at the deformed rotation experiment §9.2.1, if we take a rotation, or something very close to that, it is expected that AA does not generate wrapping (or only a tiny amount).

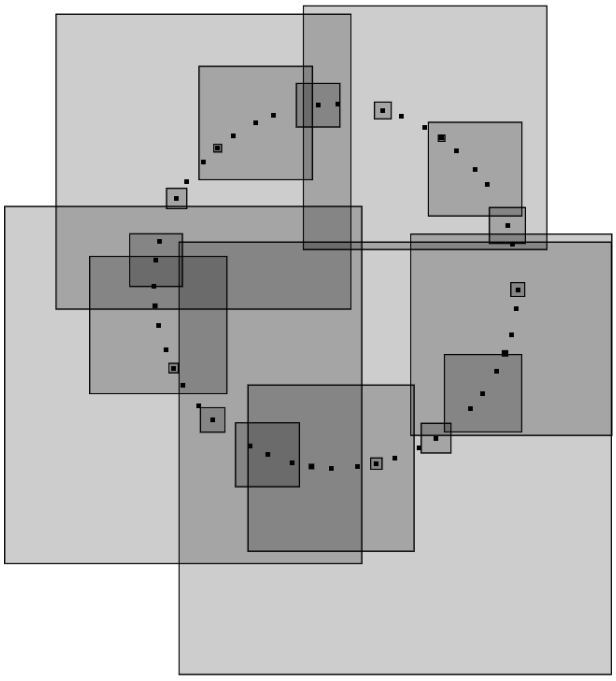
If we look at Figure 14 (a), we can see how IA performs in the problem of generating long orbits. In this figure, the problem evolved with IA until map iteration 72. At iteration 74, the IA orbit already touched the escape region.

Using AA with the radical condensation strategy is better than using IA. What we are doing in this strategy is transforming our map iteration computed with AA into an interval, but we are taking advantage of correlations between variables in the non-affine operations. That is, we are performing operations in AA and returning a result in IA, so we can call this strategy AA to IA.

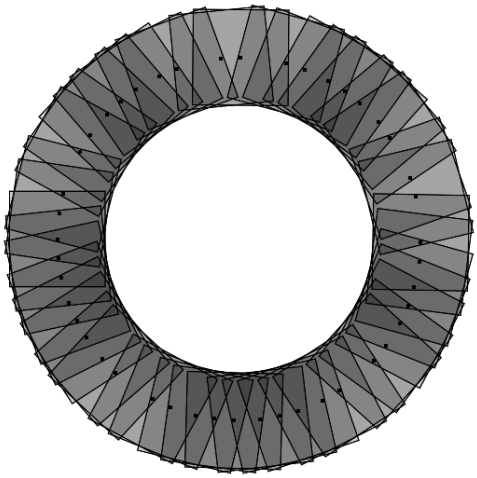
In Figure 14 (b), we can see how the radical condensation strategy performs. In this figure, the problem evolved with AA to IA until map iteration 97. The map iteration constructed an orbit with 101 steps when it touched the escape region, that is approximately 36% longer than the IA orbit.

This first AA orbit is much better than the IA orbit, but not good enough, as expected for the radical condensation. Therefore, we can conclude that with the IA and AA radical condensation strategy wrapping effects are too strong in this kind of problem, and a naive condensation strategy is not suitable for this problem. The value of the new condensation strategies will be seen in the next experiments, constructing longer orbits than the IA and AA radical condensation strategy.

9.4. **Numerical results for truncate variants.** This first set of tests (Appendix §A.1) was done on the truncate strategy with two objectives:

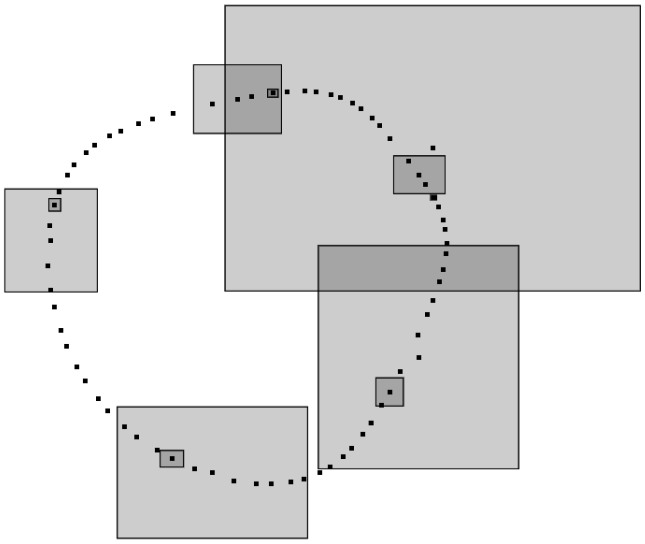


(a) IA with initial interval precision 0.0001.

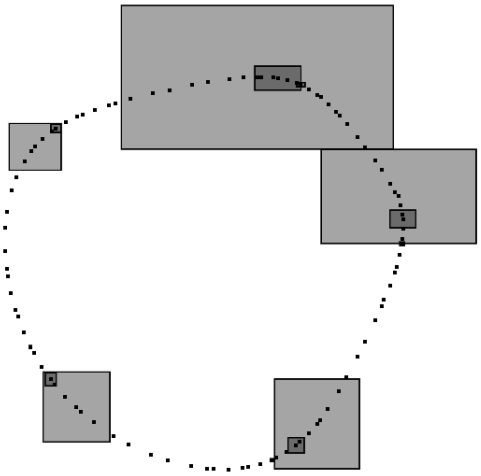


(b) AA with initial interval precision 0.1. It is hard to see, but the initial box was basically replicated and rotated by the map.

FIGURE 13. First 45 steps of IA and AA to IA orbits for Rotation map with $\cos(\alpha) = 0.24$ and initial point $x_0 = -0.38, y_0 = 0.0$.



(a) First 72 steps with IA.



(b) First 97 steps with radical condensation strategy.

FIGURE 14. IA and AA to IA orbits with initial interval precision 10^{-14} for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = -0.38, y_0 = 0.0$.

(1) The first one, and the most important, is to look at the condensation strategies with a critical point of view and observe if this idea makes sense at its core. We want to see if our most basic strategy can dominate the results of the radical strategy (which is largely used in literature);

(2) The second objective is to test whether the *truncation strategy* is useful in practice, or if we will need a more robust strategy to compute long orbits in non-linear dynamical systems.

The first conclusion on condensation strategies that we can see in general is that the truncate strategy is much better than the radical strategy. Even using only 5 symbols (Figure 18 (a)) in the affine form, the truncation strategy achieved a much better orbit than the radical strategy. The truncation strategy fails when it uses a large number of symbols, the lack of sophistication of this is translated into a cost of more than one symbol per new iteration (this trade-off is not worth it). The truncation strategy relies only on brute force, in the sense of using symbols without any intelligence in the back-end, leading to a high inefficiency.

Back to our two initial objectives, the answers are the following:

(1) Truncation is a better strategy than the radical in the sense that truncation strategy keeps competitive with full strategy for a longer part of the orbit;
(2) The lack of efficiency in noise symbol usage of the truncation strategy raises a red flag, which says: "we need a more sophisticated algorithm to work".

A small change in the way that we generate the *cutoff* will be the first update on the truncation strategy, yielding the delayed truncation strategy (§6).

The lack of efficiency of the truncation strategy is seen to be linked with the naive usage of the remainder box. The delayed truncation strategy (the experiment in Appendix §A.2) was developed exactly to delay the next truncation. By delaying the next truncation, we should impact the procedure much less with a big remainder box, as this should be effectively condensed in a much smaller number of iterations.

By reducing the frequency of condensations, we expect to insert a lower amount of error into the remainder box, and reduce its size. Until we achieve the budget again, the delayed truncation strategy will grow like a full strategy, with a difference of the small remainder box from the truncation step when it reached the budget. This small change on the remainder box usage leads us to better orbits which achieved at least four times the truncation orbit size with the same budget $N = 1280$.

As expected in the *first-order islands* experiment, the delayed truncation strategy suffers from the wrapping effect. The truncation and delayed truncation strategies performances are poor. These strategies

are very naive, and for generating long orbits, this condensation is expected to rapidly amplify the wrapping effect. Even with the change in the usage of the remainder box, the delayed truncation strategy relies still on brute force, in the sense of using symbols without any intelligence in the back-end, leading to inefficiency. The objective of the next strategy is to use order to create a savvy strategy.

The lack of efficiency of the truncation variants is seen to be linked with the naive usage of the remainder box. Even with the delayed truncation strategy, where we will use less the remainder box, we can see rapid growth of the competitive ratio. We expect that the important terms are the largest ones, and in the maximum norm strategy (experiments in Appendix §A.3), we want to keep them far from the remainder box.

The results for maximum norm strategy were good. Our prioritization on choosing which symbols will be in the remainder box leads us to much better orbits in each budget tested. However, the time performance of the maximum norm strategy is very poor, and we have a strategy here that should not be used with a larger budget.

The solution for time performance of the maximum norm strategy comes from the behavior of the delayed truncation strategy, where we can delay the truncation, by using a smaller cutoff number, which constructed a faster strategy than the truncation, so we will construct now a delayed maximum norm strategy (experiments in Appendix §A.4). The delayed maximum norm will keep the full strategy behavior a large part of the time, and avoid unnecessary sorting on affine form, keeping the strategy faster.

The delayed maximum norm is a condensation strategy that depends on a reasonable budget to work well. When we have large values for the budget is where the delayed maximum norm strategy shows its value. Even the cascade variants will not be able to overcome the delayed maximum norm strategy when the budget is large enough. In delayed maximum norm, we constructed a great strategy with the simple truncation variants. Now we need to compare it with the cascade as a more sophisticated heuristic and see where we can use this.

9.5. **Numerical results for cascade variants.** The relaxation cascade raised a new problem to deal with. This new heuristic introduced a new parameter called *relaxation factor* μ. The relaxation factor is a new degree of freedom in our problem. We did some experiments reported in Appendix §B.1 to select a good value for $\delta\mu$. We will use

the value for $\delta\mu = 3$, which showed good metrics for cascade and magnitude cascade strategies in the experiments.

We constructed a very stable strategy with the delayed maximum norm, and this strategy is very efficient with a large number of symbols in the budget. Now we want to look at the cascade heuristic (the experiments are in Appendix §B.2). The first conclusion is that the cascade strategy and the magnitude cascade strategy are great in time efficiency. The time performance of the delayed maximum norm strategy is good, but the cascade variants are better; the cascade strategy is close to the radical strategy. The magnitude cascade strategy is great for a small budget: with only 5 symbols in the affine form, this strategy achieved a better orbit than the other ones.

Looking at what we have done in the delayed truncate strategy, the first change on the cascade algorithm will be essentially the same, where we will apply the cascade heuristic only when the budget is violated. These are the semi-adaptive cascade variants, which are not fully adaptive anymore as they do not select a cutoff in every iteration, but only in the budget violation.

We have good strategies for some ranges of the budget now, magnitude cascade strategy for a small budget, and delayed maximum norm for a large budget. In the semi-adaptive cascade variants, we had our first failure in improving the results (experiment Appendix §B.3). However those semi-adaptive variants are a step to the ordered variants, which was a great impact in truncated variants, and we should see the same for the cascade heuristic.

The next step for developing the cascade variants is the introduction of the ordering, as we have done in the truncation variants. Even with the natural ordering introduced by the cascade heuristic, we will force a reordering before the cascade heuristic when we have a budget violation. This creates the ordered cascade strategy for us, which is a good strategy for medium budget size (shown in experiments in Appendix §B.4.

Based on all experiments done, we have a magnitude cascade strategy for small budget size, ordered cascade strategy for medium budget size, and delayed maximum norm for large-budget size. Now we need to exhaust the experiments that we have and look at the behavior of those three selected strategies, from that we can conclude that we have good strategies for each range of budget size.

With the results pointing to a better strategy for each budget range, now we need to validate if they work well in more experiments. The objective of the experiments in Appendix §C is to take a deeper look at the best-selected strategies applied in a wide range of examples,

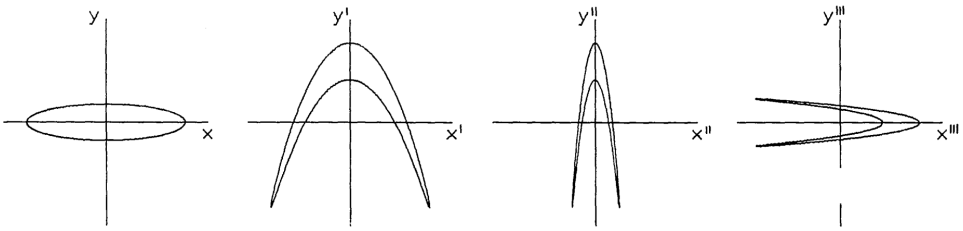


FIGURE 15. Sequence of three mappings H', H'', H''' to construct Hénon map H [24]

and confirm the hypothesis that the best strategies are magnitude cascade strategy for small budget size, ordered cascade strategy for medium budget size, and delayed maximum norm for large-budget size.

We were able to confirm a stable behavior of the strategies in all experiments listed in this section, confirming the hypothesis that we have a good strategy for each range of budget.

10. NUMERICAL RESULTS WITH PERIODIC POINTS

As the second set of experiments, we want to validate which AA strategy is useful in finding periodic points for dynamical systems. In this section, we will see the main results for all the condensation strategies and why they are useful for this task. The details of numerical experiments to validate the choice could be seen in the Appendix §D.

In these experiments, we can compare the strategies, even the full strategy, searching for periodic attractors. With AA it is easy to prove the existence of a fixed point, and we extend it to a periodic point (we will see details in §10.2). These experiments analyze the generation of wrapping effect for strategies, and how the initial rounding error interferes in a proof.

10.1. **Hénon map.** Hénon [24] presented a model problem which is as simple as possible, yet it exhibited the same essential properties as the Lorenz system. His objectives were to make the numerical exploration faster and more accurate, so that solutions could be followed for a longer time. These classical map equations are

$$\begin{aligned} x &= 1 + y - ax^2 \quad \text{and} \\ y &= bx, \end{aligned} \tag{2}$$

The Hénon map H is a product of three maps, which are defined in $R^2 \to R^2$. Consider a region elongated along the x axis. The first map H' is a folding, defined by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y + 1 - ax^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ -ax & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$

that is a vertical shearing followed by a translation. Subsequently, we apply a contraction map H'', along the x axis, defined by

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \begin{pmatrix} bx' \\ y' \end{pmatrix} = \begin{pmatrix} b & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

The final map H''' is a change of coordinates, that brings back to the orientation along the x axis, defined by

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y'' \\ x'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x'' \\ y'' \end{pmatrix}.$$

Then, as can be seen in Figure 15, the Hénon map $H = H'''H''H'$, which leads us to Equation 2. The parameter $b \in (0, 1)$, and its Jacobian is a constant $-b$, then we have a contraction map.

10.2. **Fixed points.** We shall use only the classical results on fixed points, and they are discussed with more details in the second part of the thesis, as it develops exactly this theme. A wealth of results on fixed points can be found in the books by Agarwal et al. [2] and by Berinde [4], among many others. We should use the Brouwer's fixed-point theorem to guarantee the existence of a fixed point and Banach's fixed-point theorem to guarantee the uniqueness of a fixed point.

Affine arithmetic allow us to check the hypotheses of the fixed-point theorems rigorously in a computer. Let the zonotope Z be the joint range of the affine form $\widehat{v}$. The existence of fixed points in a zonotope Z guaranteed by Brouwer's theorem follows whenever $F(\widehat{v}) = F(Z) \subseteq Z$ because then $f(Z) \subseteq F(Z)$ implies $f(Z) \subseteq Z$.

The existence of a unique fixed point in a box X guaranteed by Banach's theorem follows whenever $F(Z) \subseteq Z$ and $\|F'(Z)\| < 1$ because these imply that f is a contraction in Z, thanks to the mean value inequality. Here, F' is an interval extension of the Jacobian matrix of f. In this case, we will only need to prove

$$F(Z) \subseteq Z,$$

as the Hénon map is a contraction map.

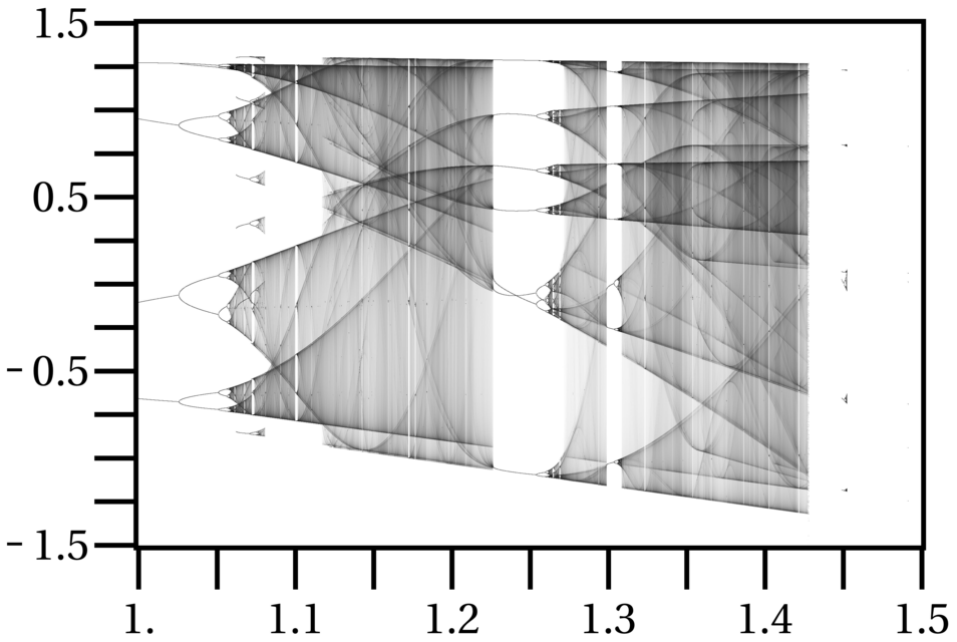


FIGURE 16. Hénon map bifurcation diagram, stationary x in function of a with $b = 0.3$ [39]

10.3. **Best condensation strategies for periodic points.** Our objective is to prove the existence of periodic orbits, in different configurations of the Hénon map, using the condensations strategies for AA. We will achieve that by proving the Brouwer's theorem. As Hénon map is a contraction, we will prove Banach's theorem too, and that the fixed point is unique.

The algorithm is simple. Fixed a bugdet[2] $N = 20$, the methodology is performing 5000 iterations with each strategy, and try to prove every period, until the period 256. We will keep the first iteration when the smallest period was proved as the result for a determined strategy.

In Table 1 of the Appendix §D are listed the experiments that will be performed in this section. From the Hénon map bifurcation diagram (Figure 16), we can select some values for a that have a periodic attractor. However, there are some interesting parameters for the Hénon map that Galias and Tucker [17] found in their paper. The results

[2]Here we are working with variation of relaxation factor $\delta\mu = 0$, that is a realaxation factor $\mu = 1$, and we are trying to be fair with cascade and magnitude cascade, for we know how these strategies work adaptively with budget.

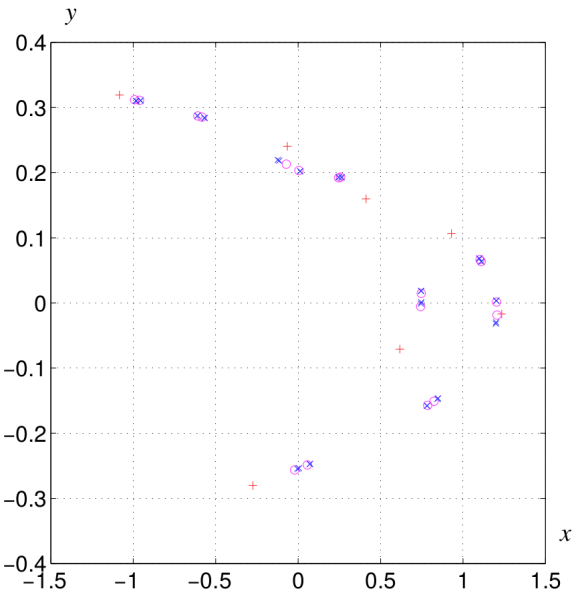


FIGURE 17. Hénon map with a = 1.3562, b = 0.2586 and attractor orbits: period-7 (+), period-18 (o) and period-36 (x) [17]

of their computation show some examples of parameters for which three coexisting attractor orbits were found, in only one configuration. We can see in Figure 17 an example of the three coexisting periodic points in experiment e08 from Table 1.

The first notable thing in the experiments, by observing some periods in Table 2, is that we do not know some of them. These Hénon map parameters have possibly chaotic attractors [17], and we cannot prove a periodic orbit for them if it is true. Of course, they may have a periodic steady state hidden by a very long transient dynamic, but this kind of study demands lots of computer power, and this is not the objective of this paper.

Example e01 shows the famous Hénon strange attractor, which is suspected to be chaotic. Even if we look at very long orbits, we cannot tell whether it is really chaotic or not, as Galias and Tucker said recently [18]:

> By performing a systematic study of the Hénon map, we find low period sinks for parameter values extremely close to the classical ones. This raises the question whether or not the well known Hénon attractor (the attractor of the Hénon map existing for the classical parameter values) is a strange attractor, or simply a stable periodic orbit. Using results

> from our study, we conclude that even if the latter were true, it would be practically impossible to establish this by computing trajectories of the map.

Trying to prove this, Galias and Tucker [16] found a steady state for Hénon map with $a = 1.399999486944$, $b = 0.3$, which is very close to the strange attractor. However, they took 6 billions of iterations using IA Newton's method to prove that these parameters have a periodic orbit with period 33.

Concentrating our efforts on testing the condensation strategies for AA, let us work with the points where we know the period of attractor orbits, even if they are long. However, working with these possibly chaotic orbits could be an interesting future work.

Looking at the truncation strategy in Tables 2 and 3, we can see that this is useless in our objective of proving periodic points. With the rough initial interval precision (2), this cannot prove anything. Looking at the refined initial interval precision (3), we can see two cases where we can prove something, but it is a very bad result. We prove the e08 with the first initial point as $p = 56$, which has $p = 7$, i.e., we proved a multiple period 8 times bigger than the original. We got the worst result when proving the e02 as $p = 52$, so this time we proved a multiple period 13 times bigger than the original. Then, we will exclude the truncation strategy from the following analysis.

Experiment e02 has a simple $p = 4$ periodic orbit, but it is surprising that there is no strategy which can prove the real period. Every strategy proved the second multiple $p = 8$, in a similar number of iterations (delayed truncation was a little better with 173 iterations). A probable reason is that the wrapping effect is stronger than the attraction of the periodic orbit, so we need a second turn for the zonotope $F^p(Z)$ to be small enough, and to be contained in Z.

On the other hand, experiment e03 was a very stable one, along all the strategies. We can see every proof similar with $p = 8$, regardless of the initial interval precision (looking at Tables 2 and 3), or the used strategy. Note that on Table 2, the delayed truncation achieves the proof a little faster, with rough initial interval precision.

Experiment e03 behavior repeats on experiment e04. In Table 2 we can see every strategy proving $p = 16$, but now we have the maximum norm strategy a little better than other ones. However, the delayed truncation has a very good performance, which is practically half of the iterations used by other strategies, even the full one. In Table 3, the results are similar for every strategy, and we suspect that a refined initial interval precision makes the computations more stable

regardless of the used strategy. And experiment e05 with $p = 32$ confirms this suspicion, where we see the exactly same behavior of the experiment e04.

In the e06 and e07 experiments we see a very different behavior, and it is the first time that the full strategy fails to prove anything, even a multiple period. In these experiments we can see the value of the more elaborated condensation strategies, which are stable in Table 3. Here, we can see the strategies proving very long periodic orbits, even a 128 sized one, and with the same number of iterations for each of the 7 strategies, which are the cascade variants and delayed maximum norm. To prove these very long orbits we will expect a lot of iterations, as it eventually needs transient states lasting longer.

Looking at Table 2 e06 and e07 experiments, we can see that a rough initial precision really interferes in the stability between strategies. We can see that it is fast to prove in this configuration, but every strategy is achieving the proof with a different number of iterations. In the two examples we can see that magnitude cascade is faster in proving the orbit, then it is a good choice if we have a rough initial interval precision and a small budget $N = 20$.

The e08 experiment is more complicated to work with, there are three coexisting periodic orbits. Note that, if our initial interval (or the orbit) touches the basin of attraction of another periodic orbit, it can make the orbit diverge (in the sense of increasing wrapping effect). Looking at Table 2, we see that we can only prove the $p = 7$ orbit, and, again, magnitude cascade as the best strategy. We cannot prove the other two attractor orbits, and it is probably based on interactions of the basins and the rough initial interval precision.

With a refined initial precision (Table 3), the behavior of strategies are better in experiment e08. We can see that in $p = 7$ attractor orbit, every strategy performs exactly in the same way. Now, we can prove the second periodic point too. In the $p = 18$ attractor orbit, we can see every strategy with similar performance, and they are proving only a multiple with $p = 36$, except for maximum norm, which was best in this experiment with refined precision and proved $p = 18$.

In the e09 experiment we could prove the three coexisting attractor orbits. For the $p = 8$ attractor orbit, only the delayed truncation could prove the correct period, every other strategy, except full and maximum norm, proved $p = 16$. In the $p = 12$, the delayed truncation, again, had a good performance. However, in $p = 20$, the delayed truncation fails, and we can see a better strategy. The delayed maximum norm can prove every coexisting attractor orbit with the correct

period (except for $p = 20$, but no other method can do better than $p = 40$, regardless of the initial interval precision).

In experiment e10 we cannot prove the possible chaotic attractor, but we can work with the other two, even the possible chaotic attractor's basin of attraction can interfere with the other two orbits. The attractor orbit with $p = 12$ does not look tough to find in Table 2, but in Table 3 only the delayed truncation could find it with $p = 12$ (the other strategies, including the full, get $p = 36$). The $p = 16$ attractor orbit is hard to find, with a refined initial interval precision only delayed maximum norm could find it. When a rough precision was used, some cascade variant strategies could find it, and we can see that in Table 2.

As the last experiment, we have a coexisting periodic point with two possible chaotic attractors. In experiment e11, only delayed truncation could prove this attractor orbit with $p = 8$, the others strategies proved a $p = 16$ or failed.

We can see that a lot of strategies can be useful in this kind of experiment, and the complete strategy, surprisingly, is not one of those. The delayed truncation can be fast sometimes in proving the periodic point, but fails in other experiments. We can use the magnitude cascade in some cases, as it is very fast, but we need to be careful when using it, since sometimes it encounters multiples of the attracting orbits. A more stable strategy is the delayed maximum norm, and it can be the best first try, if we do not know how the behavior of the dynamical system is.

11. Conclusion

In principle, the discussion in this paper applies to AA computations in any dimension d. In particular, the proposed condensation strategies work without change. However, the geometry of zonotopes in high dimensions is complex. Therefore, our focus was on performing the analysis of experiments in $\mathbf{R}^2$.

We proposed several condensation strategies, with the expectation that we would have a good choice for each situation in the experiments. With this expectation in mind, our condensation strategies were based on three principles:

- *How* they choose the cutoff m.
- *How* they select the m important noise symbols to keep.
- *When* the condensation is performed, to reduce the computational costs.

Each strategy tries to balance the time spent in condensation and the generated wrapping effect. If we look closely at our condensation strategies, we can see a chain of relations between them. This chain is clear, as we have the truncate and cascade class of strategies as a base. As an example, we have the truncation strategy modified in the first principle (*how* it choose the cutoff m), and created delayed truncation (with a fixed $m = N/2$), and its generalization semi-adaptive cascade (with a m selected by cascade heuristic). In the second principle (*how* strategies select the m important noise symbols to keep), we can see the evolution from maximum norm strategy (selecting the larger λ_i symbols) to delayed maximum norm strategy (selecting the larger λ_i symbols but with a cutoff at the median) and its generalization ordered cascade (forced ordered and an m selected by the cascade heuristic). Finally, the cascade heuristic created a new class of strategies in the sense of the third principle (*when* the condensation is performed).

Experimenting with the condensation strategies in two classical dynamical systems (Cremona and Hénon maps) was an interesting and relevant path to see how useful the strategies could be. As a first insight, we see that with bounded complexity, condensed AA is a reliable substitute for IA, as IA has severe wrapping problems even when it iterates a simple 45° rotation.

Experimenting with long orbits, we see some very useful strategies in determining situations:

- The magnitude cascade strategy is very efficient in computer time. Furthermore, its performance in suppressing the wrapping effect surpassed the cascade strategy in experiments. When we observe only the magnitude of the partial deviations, we constructed a new heuristic that expects to condense the symbols less often, and frequently performs condensations with fewer symbols. Then, our belief is that we have a more stable heuristic. This strategy worked well for small values of the budget N; with a budget in the range of 5 to 20 symbols, this strategy is the best choice.
- When we forced a sorting in cascade strategy, ignoring its natural order, we created the ordered cascade strategy. The sorting step in the algorithm compensated the computational cost, as we created a strategy that overcame the others with medium budgets. This strategy is our suggestion for a medium range of values for the budget; with a budget in the range of 21 to 320 symbols, this strategy is the best choice.

- The delayed maximum norm strategy, which looks simple and does not have a heuristic in the cutoff, computed the longest orbits that we have seen in this work, including a reliable orbit with many more than 100000 symbols in experiments §C.2 and §C.3. This strategy shows its value only by having a sufficient number of symbols to work; with a budget in the range of 321 to 1280 symbols, this strategy is the best choice.

When trying to prove the attractor orbits, the full strategy was not a good one. Here, we could see some condensation strategies very useful in this problem:

- Even the delayed truncation shows its value when it can prove quickly some periodic points, yet it fails in some experiments.
- Of course, we must use the faster magnitude cascade strategy carefully, since it encounters only multiples of real orbits in many experiments.

As a first try in a new dynamical system, that we know nothing about, the suggestion is to use the delayed maximum norm strategy. It is one of the most stable strategies and worked very well in the vast range of experiments that we have done. The second advantage is the simplicity, it is much easier to implement and validate than the cascade variant strategies.

Future work. The first idea for future work is constructing a heuristic based on hybrid strategies. It is expected to construct a self-correcting algorithm for condensation strategies. The idea is in essence very simple, basically, if a strategy is not doing well, switch to a more robust one. For that, we need to develop a novel algorithm to schedule the strategies and organize the rules of substitution.

An interesting future work should be based on the recent works of Galias and Tucker [17, 16, 18]. As it was discussed in §D, we can develop new methodologies based on their work, and use AA condensation strategies to study the possible chaotic attractors.

As Galias and Tucker concluded using their results, even if the Hénon strange attractor[3] is a stable periodic orbit, it would be practically impossible to establish this by computing trajectories of the map. As they use methodologies based on IA, our AA condensation strategies could be efficient in iterating these extremely long transient states (billions of map iterations) and help to study this problem.

[3]The strange attractor of the Hénon map exists for the classical parameter values $a = 1.4, b = 0.3$

As discussed in the preface, there is a natural connection between this part and the second part of the work, where we developed a new approach based on an elegant way to search a domain and find periodic points based on spatial subdivisions (quadtrees) and IA [3]. An idea for future work is combining the two parts of this thesis in a novel algorithm that uses AA to search for the fixed points.

We can reduce the wrapping effect with the use of AA in the search of periodic points based on spatial subdivisions, and we expect to reduce the computer power expended in our second part of this thesis. For this, as the geometry of zonotope intersections is complicated (it is harder with floating-point arithmetic in very high precisions of zonotopes partial deviations), we will need to structure our methodology with a new condensation strategy, this time based on geometry. We know that the radical strategy condenses the affine form into a zonotope, like an AABB. We can explore the geometry of search intersections in the quadtree and condense our zonotope in an oriented minimum bounding box (OMBB). With this strategy, we could explore geometry to perform a fast search, and work with a simple geometry of rectangle intersections.

Acknowledgements. The author is partially supported by CNPq and FAPERJ doctoral scholarships. This research was done in the Visgraf Computer Graphics laboratory at IMPA. Visgraf is supported by the funding agencies FINEP, CNPq, and FAPERJ, and also by gifts from IBM Brasil, Microsoft, NVIDIA, and other companies.

REFERENCES

[1] E. Adams, W. F. Ames, W. Kühn, W. Rufeger, and H. Spreuer. Computational chaos may be due to a single local error. *Journal of Computational Physics*, 104(1):241–250, 1993.

[2] R. P. Agarwal, M. Meehan, and D. O'Regan. *Fixed point theory and applications*. Cambridge University Press, 2001.

[3] J. E. Ayres and L. H. De Figueiredo. Interval methods for fixed and periodic points: Development and visualization. *JUCS - Journal of Universal Computer Science*, 26(10):1312–1330, 2020. https://doi.org/10.3897/jucs.2020.068.

[4] V. Berinde. *Iterative approximation of fixed points*, volume 1912 of *Lecture Notes in Mathematics*. Springer, 2007.

[5] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *Proceedings of SIBGRAPI'93*, pages 9–18, October 1993.

[6] R. M. Corless. What good are numerical simulations of chaotic dynamical systems? *Computers & Mathematics with Applications*, 28(10–12):107–121, 1994.

[7] R. M. Corless, C. Essex, and M. A. H. Nerenberg. Numerical methods can suppress chaos. *Physics Letters A*, 157(1):27–36, 1991.

[8] F. de Carvalho Nascimento, A. Paiva, L. H. de Figueiredo, and J. Stolfi. Approximating implicit curves on plane and surface triangulations with affine arithmetic. *Computers & Graphics*, 40:36–48, 2014.
[9] A. de Cusatis Jr., L. H. de Figueiredo, and M. Gattass. Interval methods for ray casting implicit surfaces with affine arithmetic. In *Proceedings of SIBGRAPI'99*, pages 65–71. IEEE Computer Press, 1999.
[10] L. H. de Figueiredo and J. Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.
[11] L. H. de Figueiredo, J. Stolfi, and L. Velho. Approximating parametric curves with strip trees using affine arithmetic. *Computer Graphics Forum*, 22(2):171–179, 2003.
[12] J. B. S. de Oliveira, J. Stolfi, D. Nehab, and L. H. de Figueiredo. Rigorous bounds for polynomial julia sets. *Journal of Computational Dynamics*, 3(2):113–137, 2016.
[13] Z. Galias. Rigorous numerical studies of existence of periodic orbits from Hénon map. *Journal of Universal Computer Science*, 4(2):114–124, 1997.
[14] Z. Galias. Interval methods for rigorous investigations of periodic orbits. *International Journal of Bifurcation and Chaos*, 11(09):2427–2450, 2001.
[15] Z. Galias. Rigorous investigation of the Ikeda map by means of interval arithmetic. *Nonlinearity*, 15(6):1759–1779, 2002.
[16] Z. Galias and W. Tucker. Combination of exhaustive search and continuation method for the study of sinks in the hénon map. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2751–2754. IEEE, 2013.
[17] Z. Galias and W. Tucker. Numerical study of coexisting attractors for the hénon map. *International Journal of Bifurcation and Chaos*, 23(07):1330025, 2013.
[18] Z. Galias and W. Tucker. Is the hénon attractor chaotic? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(3):033102, 2015.
[19] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.
[20] A. Golmakani, C. E. Koudjinan, S. Luzzatto, and P. Pilarczyk. Rigorous numerics for critical orbits in the quadratic family. *Chaos*, 30(7):073143, 2020.
[21] L. J. Guibas, A. T. Nguyen, and L. Zhang. Zonotopes as bounding volumes. In *SODA*, volume 3, pages 803–812, 2003.
[22] S. M. Hammel, J. A. Yorke, and C. Grebogi. Do numerical orbits of chaotic dynamical processes represent true orbits? *Journal of Complexity*, 3(2):136–145, 1987.
[23] M. Hénon. Numerical study of quadratic area-preserving mappings. *Quarterly of Applied Mathematics*, 27(3):291–312, 1969.
[24] M. Hénon. A two-dimensional map with a strange attractor. *Communications in Mathematical Physics*, 50(1):69–77, 1976.
[25] Y. Hijazi, H. Hagen, C. D. Hansen, and K. I. Joy. Why interval arithmetic is so useful. *Visualization of large and unstructured data sets*, 2008.
[26] W. Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61(1):47–67, 1998.
[27] W. Kühn. Zonotope dynamics in numerical quality control. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 125–134. Springer, 1998.
[28] W. Kühn. Towards an optimal control of the wrapping effect. In T. Csendes, editor, *Developments in Reliable Computing*, pages 43–51. Springer, 1999.

[29] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.
[30] E. N. Lorenz. *The Essence of Chaos*. Taylor & Francis, 1995.
[31] P. Lynch. *The Emergence of Numerical Weather Prediction: Richardson's Dream*. Cambridge University Press, 2014.
[32] F. Messine. Extentions of affine arithmetic: Application to unconstrained global optimization. *J. UCS*, 8(11):992–1015, 2002.
[33] F. Messine and A. Touhami. A general reliable quadratic form: An extension of affine arithmetic. *Reliable Computing*, 12(3):171–192, 2006.
[34] D. Michelucci and S. Foufou. Interval-based tracing of strange attractors. *International Journal of Computational Geometry & Applications*, 16(1):27–39, 2006.
[35] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
[36] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, 1979.
[37] R. E. Moore, R. Baker Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.
[38] A. Paiva, L. H. de Figueiredo, and J. Stolfi. Robust visualization of strange attractors using affine arithmetic. *Computers & Graphics*, 30(6):1020–1026, 2006.
[39] J. Pierce. Henon bifurcation map b=0.3. https://commons.wikimedia.org/wiki/File:Henon_bifurcation_map_b%3D0.3.png, 2011. Accessed: 2020-10-15.
[40] J. Stolfi and L. H. de Figueiredo. *Self-validated numerical methods and applications*. Monograph for 21st Brazilian Mathematics Colloquium. IMPA, 1997.
[41] B. Traylor and V. Kreinovich. A bright side of NP-hardness of interval computations: interval heuristics applied to NP-problems. *Reliable Computing*, 1(3):343–359, 1995.
[42] W. Tucker. The Lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 328(12):1197–1202, 1999.
[43] W. Tucker. *Validated numerics: A short introduction to rigorous computations*. Princeton University Press, 2011.
[44] J. P. Turner. *Analysing and Bounding Numerical Error in Spiking Neural Network Simulations*. PhD thesis, University of Sussex, 2020.

Appendices

APPENDIX A. LONG ORBITS EXPERIMENTS WITH TRUNCATE VARIANTS

This work has as objective to develop AA condensation strategies. To see whether the strategies are good enough to be used in applications, we need to put all strategies to the proof, and select the more useful ones. We will explore the diverse set of behaviors for the Cremona map to select the most useful strategies. Moreover, we want to verify which strategies are good with a large initial rounding error, how strategies perform with low computer power to spend, their capability to construct very large orbits, their relative performance based on the full strategy, etc.

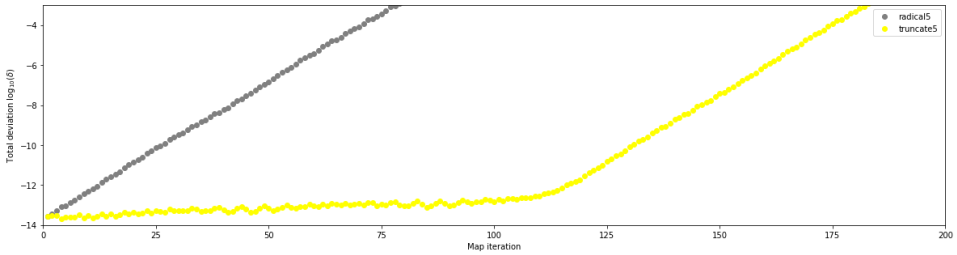
In this section we will look at the truncate variants (§6) to understand these strategies well. A first test is try to work with the already developed tools in the literature, i.e., AA and AA with the radical condensation strategy.

A.1. **Truncation strategy experiments.** Let's work with the first experiment, the *deformed rotation*, which has the simplest behavior in our set of problems. Here we will see something close to a rotation map, but with some deformation. We already saw in the last section that radical strategy touches an escape region very fast, even on this map. Now we will look at this in a more structured way, and observe the *total deviation* of the strategy. Starting always with a initial interval with *total deviation* equals to 10^{-14}, when it turns higher than 10^{-3}, we will say that the strategy has a very large perimeter in the zonotope, and the iteration *blows up*.

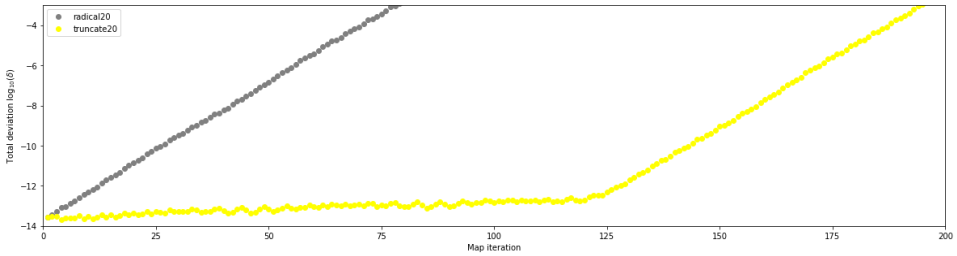
Looking at Figure 18, we can see in general, the truncate strategy is much better than the radical strategy. Even using only 5 symbols (Figure 18 (a)) in the affine form, the truncation strategy achieved a much better orbit than radical strategy.

The three extra symbols generated an orbit that blows up at iteration 183, which is larger than the radical strategy 79 iterations orbit. That is, the 5 symbols truncation strategy (which generalizes the radical strategy) is way better than the radical strategy. With this we can see a basic strategy with a small number of symbols is dominating the radical strategy.

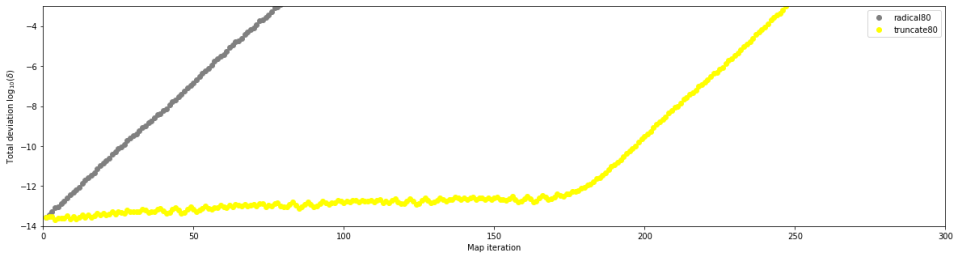
The problem of the radical strategy is never using more than two symbols. Then, the budget means nothing for it. The objective of the truncation strategy is simple growth until it touches the budget, there is no complex logic behind it.



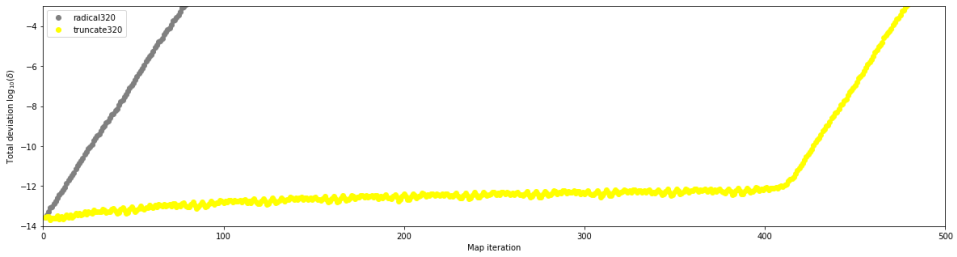
(a) N = 5



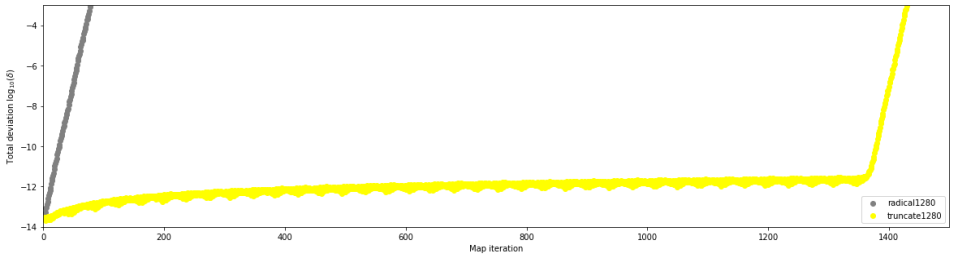
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280

FIGURE 18. Performance of radical and truncation strategies based on total deviation for *deformed rotation* example in Cremona map.

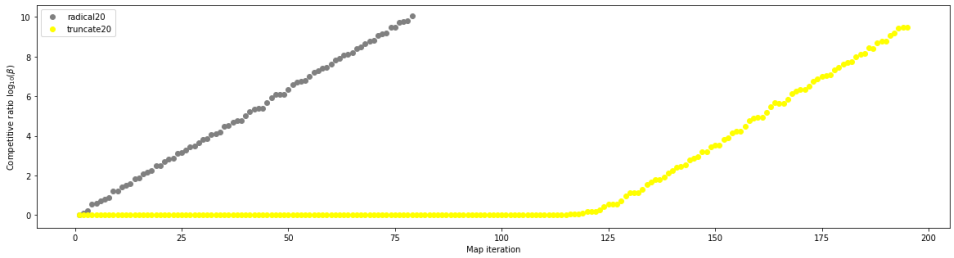


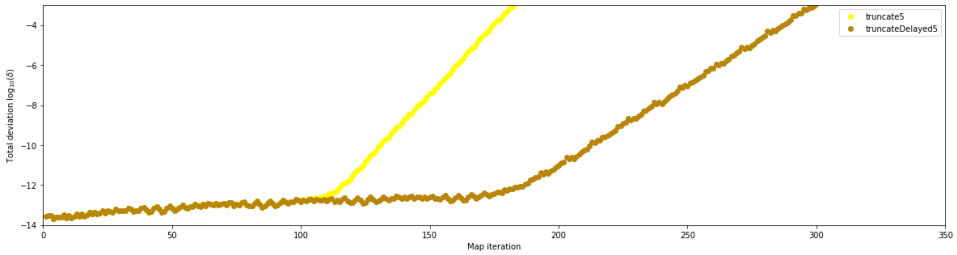
FIGURE 19. Competitive ratio of radical and truncation strategies for *deformed rotation* example in Cremona map with $N = 20$.

The lack of sophistication on truncation strategy is seen in Figure 18 (b), (c), (d) and (e). The increase of the budget to 20 in (b) helps us to achieve 195 steps before blow up, that is, the 15 extra symbols compared with (a), generated only 12 steps on the map iteration, which can be translated to a cost of more than one symbol per new iteration (this trade-off really is not worth it). In the next step (c), we have 80 symbols, and an orbit with 247 symbols (again paying more than one symbol per new iteration). In step (d), we have 320 symbols and an orbit with length 479, and the same inefficiency behavior happens again. Finally, the final step in (e) has 1280 symbols and 1430 steps in the map iteration, and there is no change in the behavior.

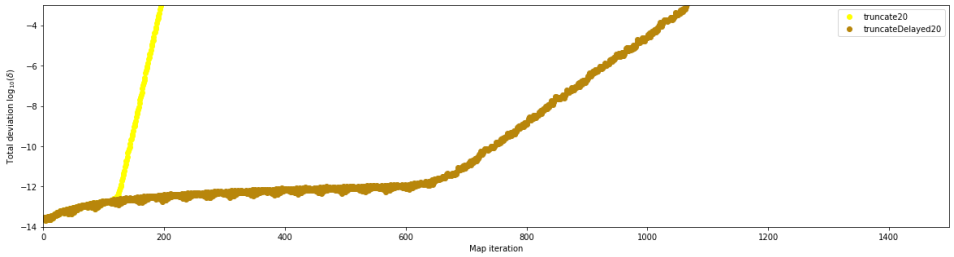
This strategy relies only on brute force, in the sense of using symbols without any intelligence in the back-end, leading to a high inefficiency. At least, if we take a look at Figure 19, we can see that the gain on β-competitiveness is very high compared with the radical strategy. The truncation strategy stops having a competitive rate close to zero only around 120 iterations. On the other hand, the radical strategy misses the full strategy zonotope perimeter quickly, the wrapping effect in the remainder box is seen in the first set of iterations.

A.2. **Delayed truncation strategy experiments.** Let's observe the performance difference of truncation and delayed truncation strategies looking at the *deformed rotation*. Looking at Figure 20, we can see in general, the delayed truncation strategy is much better than the truncation strategy.

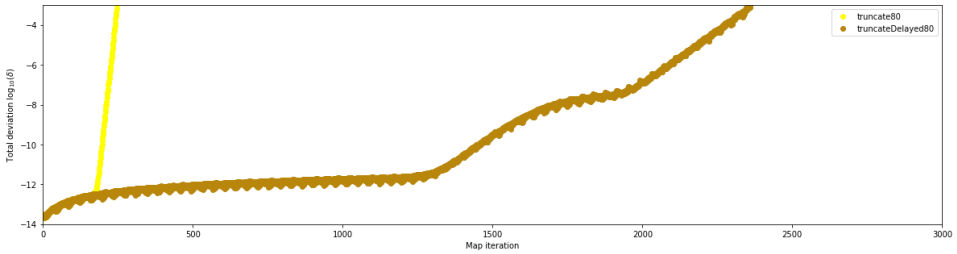
Even using only 5 symbols (Figure 20 (a)) in the affine form, the delayed truncation strategy achieved a much better orbit than the truncation strategy. The truncation strategy orbit blows up at iteration 183, which is smaller than the delayed truncation strategy orbit, which



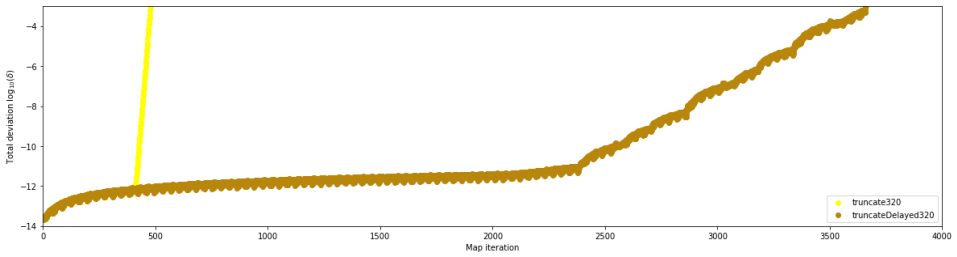
(a) N = 5



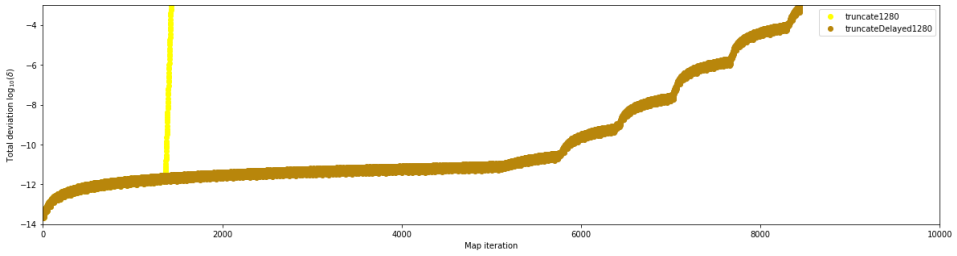
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280

FIGURE 20. Performance of truncation and delayed truncation strategies based on total deviation for *deformed rotation* example in Cremona map.

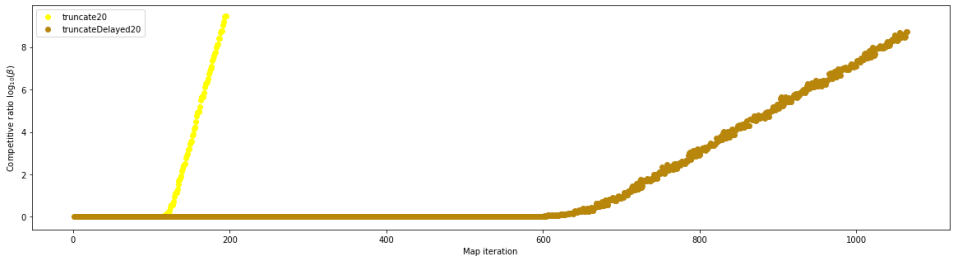


FIGURE 21. Competitive ratio of truncation and delayed truncation strategies for *deformed rotation* example in Cremona map with $N = 20$.

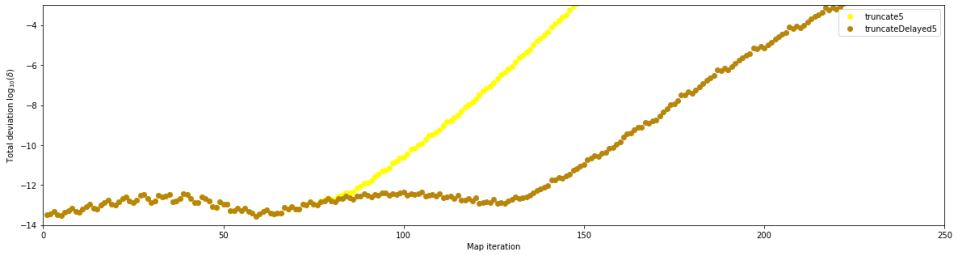
blows up after 300 iterations (i.e., we have an orbit approximately 64% longer).

This small change on the remainder box usage leads us to better orbits in each graph of Figure 20. Looking at Figure 20 (e) we were able to achieve an orbit with more than 8000 iterations in the delayed truncation strategy, which is at least four times better than the truncation orbit with exactly the same budget.

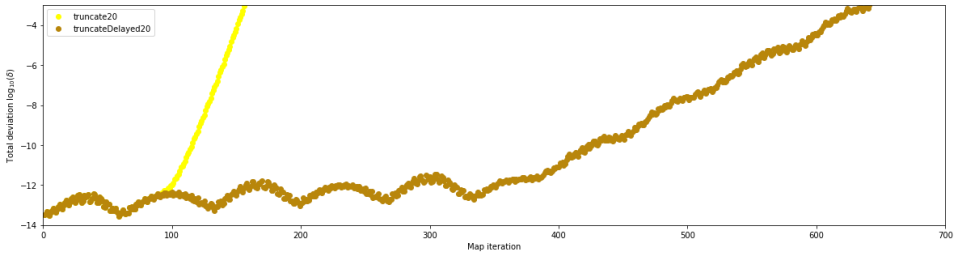
If we take a look at Figure 21, we can see that the gain on β-competitiveness for delayed truncation strategy is very high compared with the truncation strategy. The delayed truncation strategy stops having a competitive rate close to zero only around 600 iterations. On the other hand, the truncation strategy misses the full strategy zonotope perimeter around 120 (as we saw in the last section), and again we can see delayed truncation more than four times better than truncation.

Now we should observe the delayed truncation strategy when it is applied to a complex behave in the dynamical system. Looking at Figure 5, we can see a region with very different behavior from a deformed rotation. This region is composed of five closed curves, or islands, that can be seen in Figure 7, which is our *first-order islands* example. Now the effect of quadratic terms is higher, and we expect a bigger wrapping effect due to the non-affine approximations. We shall not see long orbits as we have seen before in *deformed rotation*, as this new problem is harder.

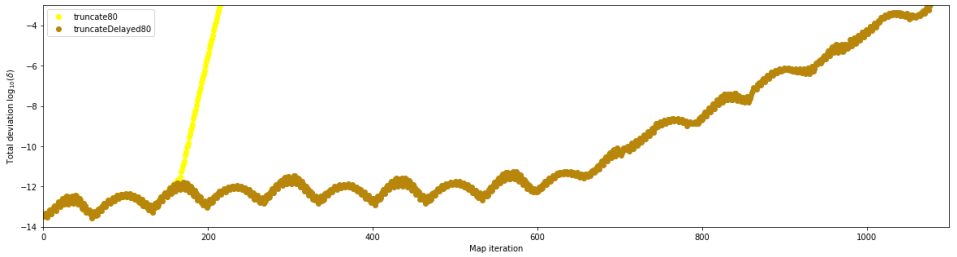
But looking at Figures 22 and 23 we can see that delayed truncation is really a better strategy than truncation. There is no moment when truncation strategy gets close to perform equal delayed truncation strategy.



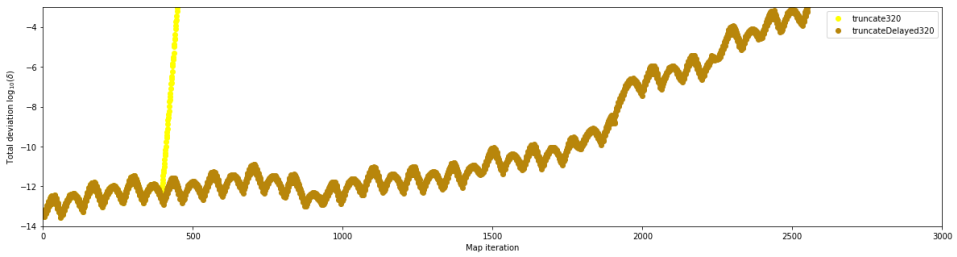
(a) N = 5



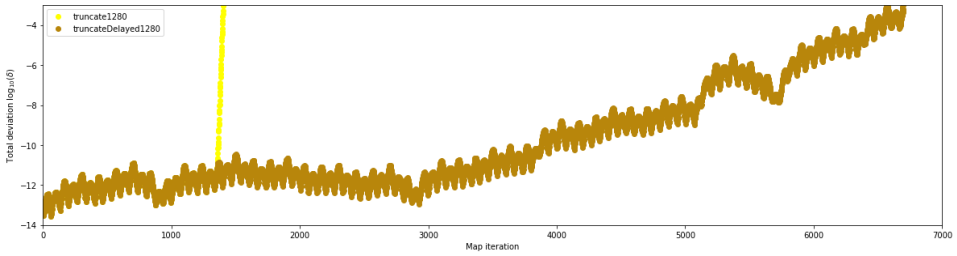
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280

FIGURE 22. Performance of truncation and delayed truncation strategies based on total deviation for *first-order islands* example in Cremona map.

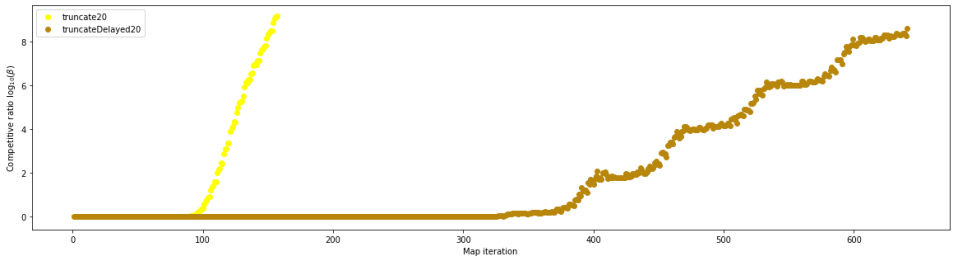
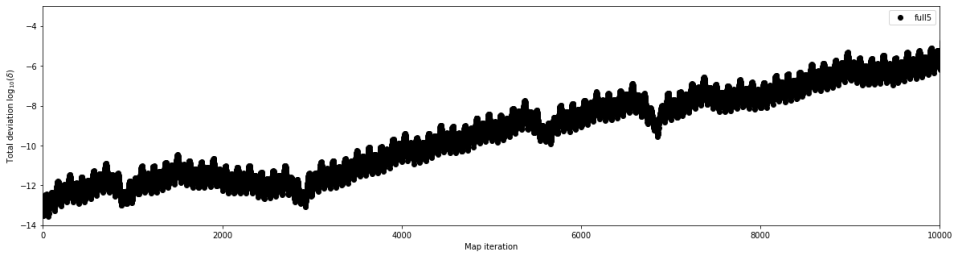
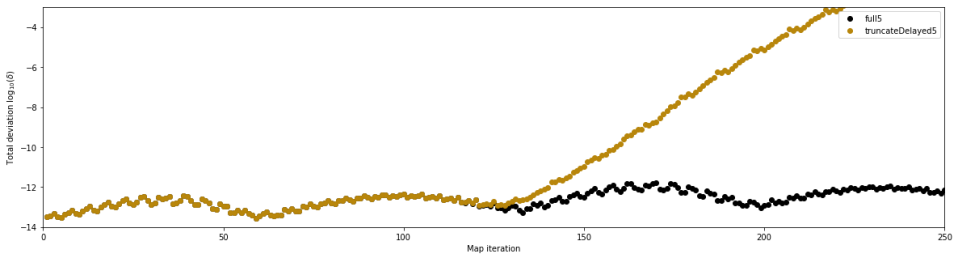


FIGURE 23. Competitive ratio of truncation and delayed truncation strategies for *first-order islands* example in Cremona map with N = 20.



(a) Full strategy



(b) Zoom of full and delayed truncation strategies

FIGURE 24. Performance of full and delayed truncation strategies based on total deviation for *first-order islands* example in Cremona map.

Figure 24 shows how the full strategy performs in this problem until 10000 map iterations. Here, if we compare it with *deformed rotation* example, we can see an oscillatory behavior on all the total deviation graphs. This behavior is not due to the condensation strategies. As we can see in Figure 24 (a), the full strategy also suffers from that behavior. This behavior comes from the frequent change of closed curves, where, in each island, we have a different outcome of non-affine operations in the wrapping effect.

As expected, we can not get long orbits as in the *deformed rotation* experiment. In this experiment we have a strong wrapping effect, even the full strategy loses precision quickly. We can see this in Figure 24 (a), where the full strategy loses a lot of precision, and the total deviation is at the order of 10^{-6}.

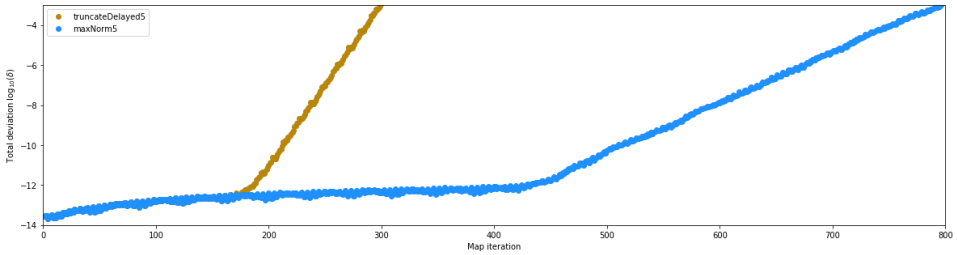
A.3. **Maximum norm strategy experiments.** Let's observe the performance difference of delayed truncation and maximum norm strategies looking at the *deformed rotation*. Looking at Figure 25, we can see in general, the maximum norm strategy is much better than the delayed truncation strategy.

Even using only 5 symbols (Figure 25 (a)) in the affine form, the maximum norm strategy achieved a much better orbit than the delayed truncation strategy. The delayed truncation strategy orbit blows up in iteration 300, which is much smaller than the maximum norm strategy orbit, which blows up after 797 iterations (i.e. we have an orbit approximately 166% bigger).

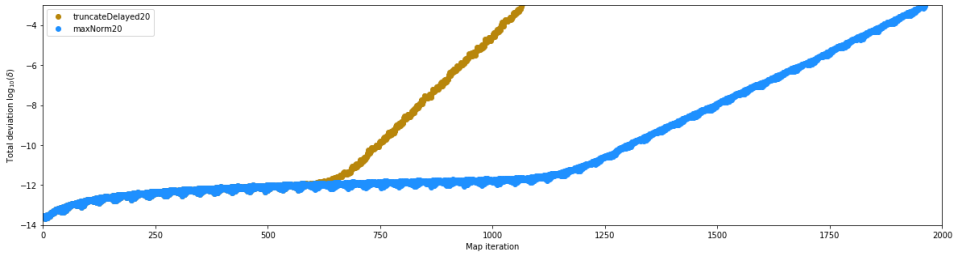
Our prioritization on choosing which symbols will be in the remainder box leads us to better orbits in each graph of Figure 25. Looking at the 25 (e) we were able to achieve an orbit with more than 27000 symbols in the maximum norm strategy, which is at least three times better than the delayed truncation orbit with exactly the same budget.

If we take a look at Figure 26, we can see a great gain on β-competitiveness for maximum norm strategy, and it is much better compared with the truncation variants. On Figure 26 (a), the maximum norm strategy stops to have a competitive rate close to zero only around 1100 iterations. On the other hand, the truncation strategy loses the full strategy zonotope perimeter around 120, and the delayed truncation strategy loses it around 600 (as we saw in the last section), and we can see the maximum norm strategy close to two times better than delayed truncation (more than nine times better than truncation).

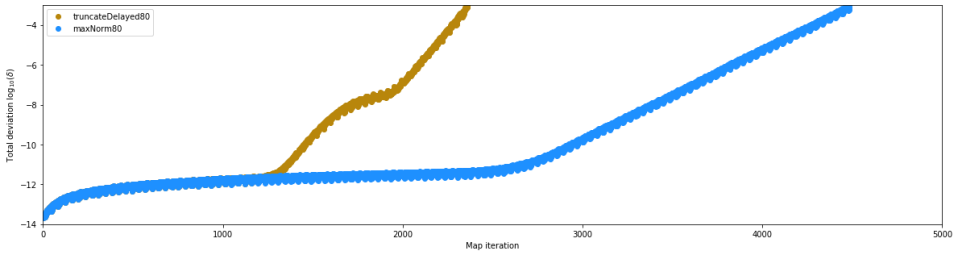
Looking at a bigger budget ($N = 320$ on Figure 26 (b)) we can see maximum norm strategy close to full strategy for approximately 6000 strategies, when the delayed truncation is losing full close to 2000 iterations. However, the best characteristic of the maximum norm strategy that we can see in this graph is the growth rate of the competitive ratio, which is much smaller for the maximum norm than the other two truncation variants. Here, after 6000 iterations we have a $\beta = 1.48$, that is the maximum norm after 6000 iterations is yet competitive with full strategy, and the zonotone has a total deviation less than twice greater than the full strategy.



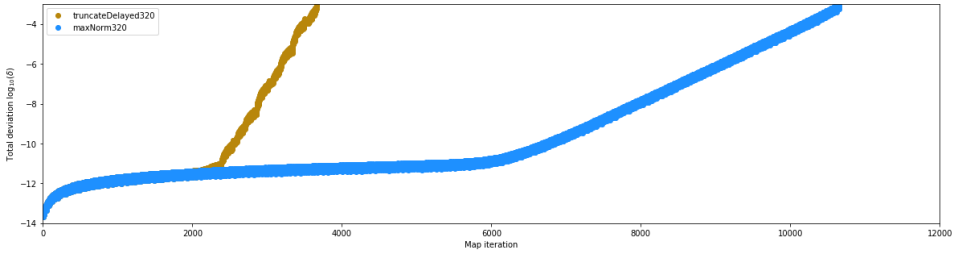
(a) N = 5



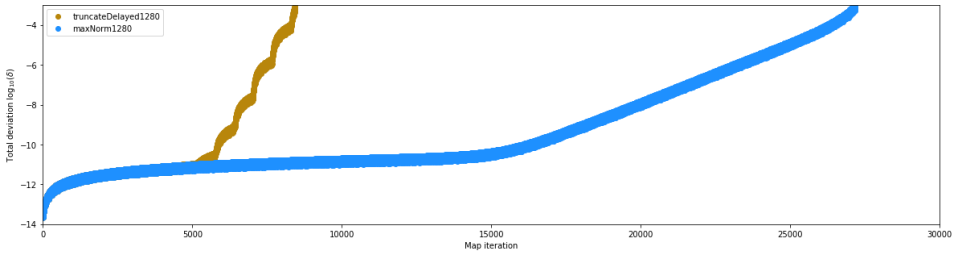
(b) N = 20



(c) N = 80

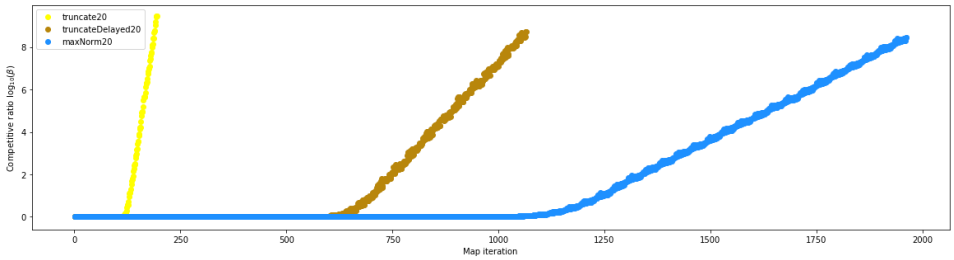


(d) N = 320

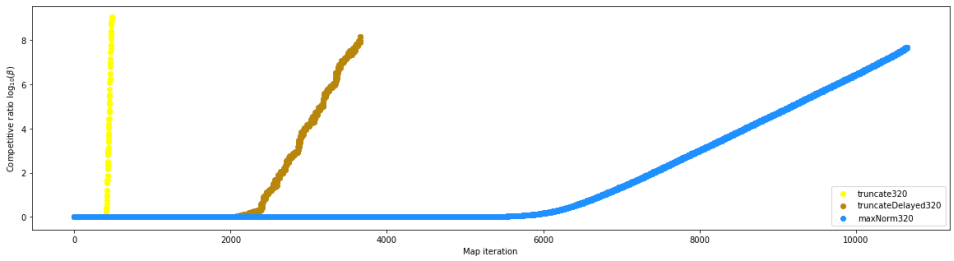


(e) N = 1280

FIGURE 25. Performance of delayed truncation and maximum norm strategies based on total deviation for *deformed rotation* example in Cremona map.

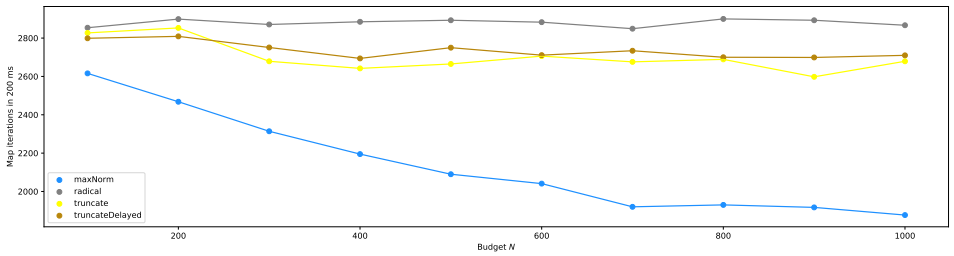


(a) N = 20

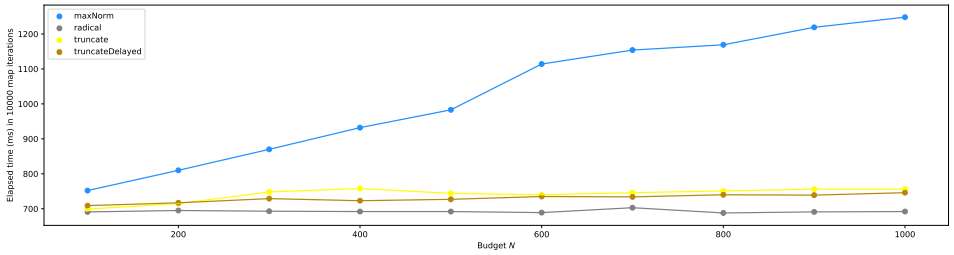


(b) N = 320

FIGURE 26. Competitive ratio of truncation, delayed truncation and maximum norm strategies for *deformed rotation* example in Cremona map.



(a) Number of iterations for fixed time



(b) Elapsed time for fixed iteration number

FIGURE 27. Time performance of radical, truncation, delayed truncation and maximum norm strategies for *deformed rotation* example in Cremona map.

This time we have a good strategy, which constructed a long orbit, close to 30000 iterations, with 1280 symbols on the budget (Figure 25 (e)). However, we now have an additional cost in ordering the terms, which is a non-trivial cost. And looking at this important dimension of performance, the time performance is where the things are worst for maximum norm strategy.

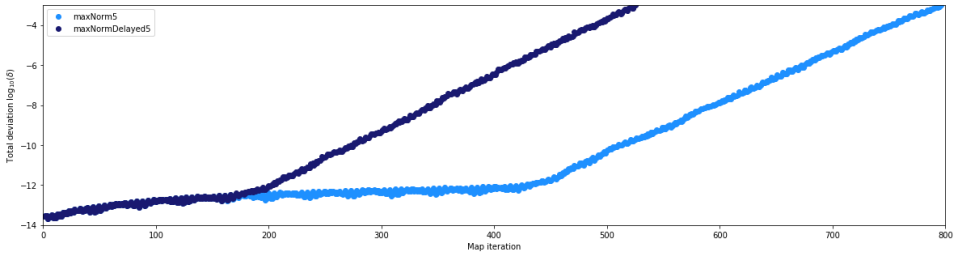
First of all looking at Figure 27, as expected, the radical strategy is the fastest one, as it uses only two symbols, independently of the budget. The delayed truncation strategy is a little faster than truncation, so it is probably connected to the fact that delayed truncation performs a much smaller number of condensations, working with the full strategy behavior for long periods.

The time performance of the maximum norm strategy is very poor, the strategy is costing next to the double of time of the other strategies with 1000 symbols (Figure 27 (b)). The behavior of this time performance graph is growing fast with the number of symbols, then we have a strategy here that should not be used with a bigger budget.

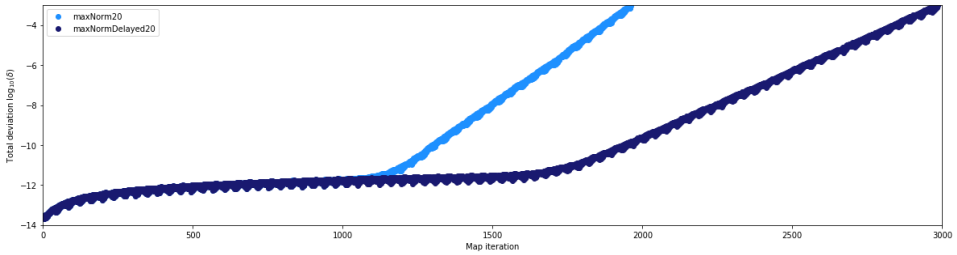
A.4. **Delayed maximum norm strategy experiments.** We constructed a more stable strategy with the maximum norm strategy, but it is time inefficient. The delayed modification of this strategy is expected to solve the time problem (as we will reduce strongly the number of ordinations). What we need to look at is if the delayed maximum norm strategy is stable and efficient in generating long orbits.

Let's observe the performance difference of maximum norm and delayed maximum norm strategies looking at the *deformed rotation*. Looking at Figure 28, we can not see a general better behavior anymore, but the delayed maximum norm strategy is much better than the maximum norm strategy (looking at big budgets) and is our best truncation variant strategy.

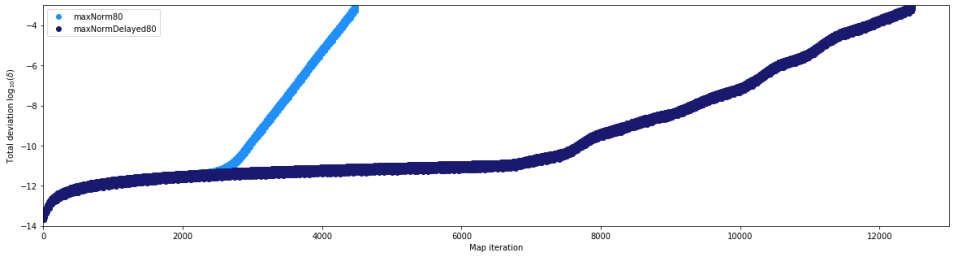
Only with 5 symbols (Figure 28 (a)) in the affine form, the maximum norm strategy achieved a better orbit than the delayed maximum norm strategy. The delayed maximum strategy orbit blows up in iteration 525, which is smaller than the maximum norm strategy orbit, which blows up after 797 iterations (i.e. we have an orbit approximately 50% bigger). However, this is the only budget where the maximum norm strategy is better, and it is a very small quantity to use only 5 symbols on budget, where the continuous ordination in each iteration is better than using full behavior. If we have a problem where we can use few symbols, the maximum norm could be useful as we have to order a smaller.



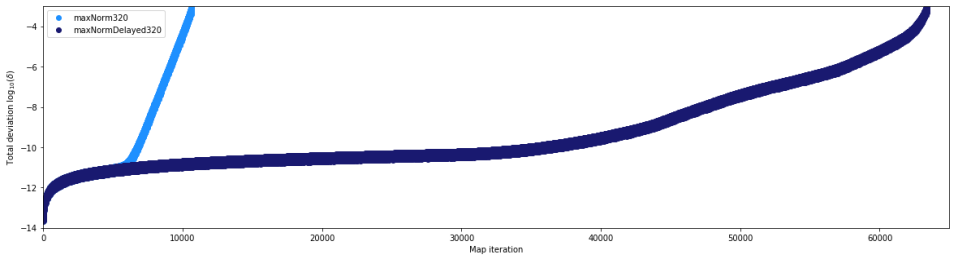
(a) N = 5



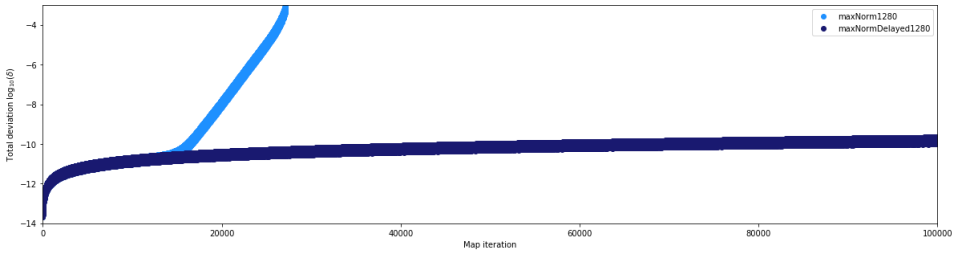
(b) N = 20



(c) N = 80

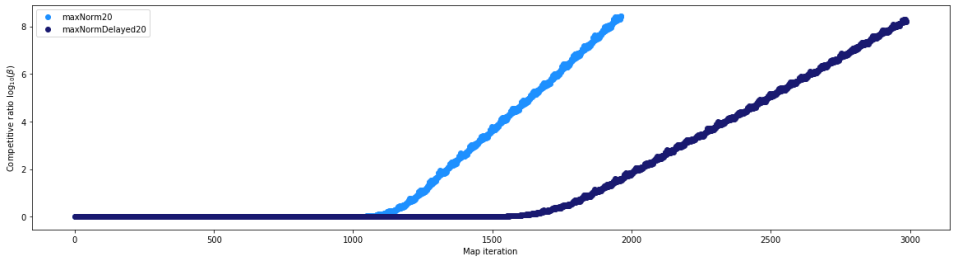


(d) N = 320

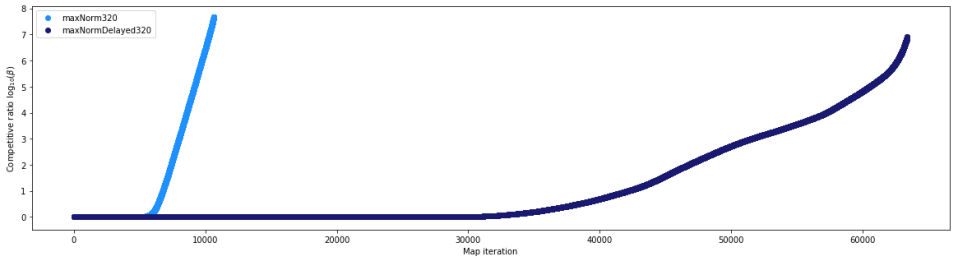


(e) N = 1280

FIGURE 28. Performance of maximum norm and delayed maximum norm strategies based on total deviation for *deformed rotation* example in Cremona map.



(a) N = 20



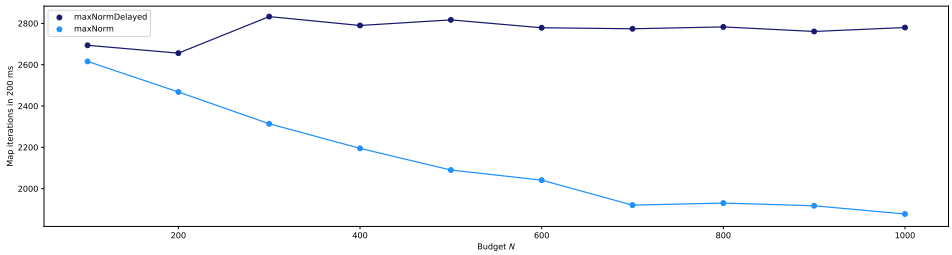
(b) N = 320

FIGURE 29. Competitive ratio of maximum norm and delayed maximum norm strategies for *deformed rotation* example in Cremona map.

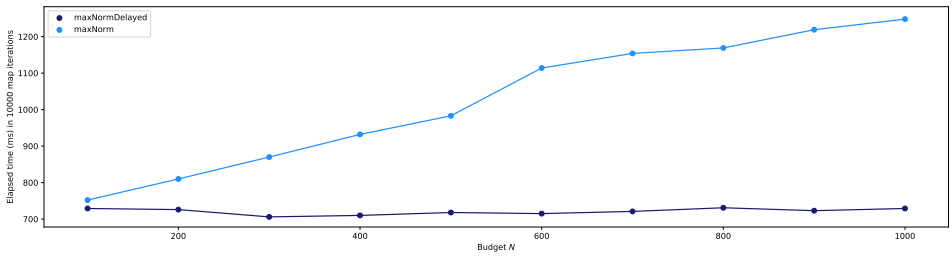
Our prioritization on choosing which symbols will be in the remainder box, and using the full behavior a big part of the time (where the affine form size n is smaller than the budget N) leads us to better orbits in each graph of Figure 28 (b)-(e). Looking at the 28 (e) we were able to achieve an orbit with more than 100000 symbols in the delayed maximum norm strategy, which is at least four times better than the maximum norm orbit with the same budget.

The delayed maximum norm is a condensation strategy that depends on a reasonable budget to work well. When we have a large budget (Figure 28 (e)), we can see the potential of this condensation strategy. With a budget equals to 1280 symbols, we can build an orbit with size 100000 without suffering a blow-up, and we have a perimeter of 10^{-10} order for total deviation. That is, we had a little increased in the affine form due to the wrapping effect introduced in the remaining box. This is a much more stable strategy than the maximum norm strategy, which blows up with fewer than 30000 map iterations.

If we take a look at Figure 29, we can see a great gain on β-competitiveness for delayed maximum norm strategy, and it is much better compared with the maximum norm strategy. On Figure 29



(a) Number of iterations for fixed time



(b) Elapsed time for fixed iteration number

FIGURE 30. Time performance of maximum norm and delayed maximum norm strategies for *deformed rotation* example in Cremona map.

(a), the delayed maximum norm strategy stops to have a competitive rate close to zero only around 1700 iterations. On the other hand, the maximum norm strategy loses the full strategy zonotope perimeter around 1100 (as we saw in the last section). With only 20 symbols the delayed maximum norm is not much better than maximum norm strategy.

Looking at a bigger budget ($N = 320$ on Figure 29 (b)) we can see delayed maximum norm strategy close to full strategy for approximately 35000 iterations, when the maximum norm is losing full close to 6000 iterations. However, the best characteristic of the maximum norm strategy that we can see in this graph is the growth rate of the competitive ratio, which is much smaller for the delayed maximum norm strategy. Here, after 35000 iterations we have a $\beta = 1.58$, that is the maximum norm after 35000 iterations is yet competitive with full strategy, and the zonotone has a total deviation greater than the full strategy, but not the double yet.

With the delayed maximum norm strategy, now we have our best strategy from truncate variants, which constructed a very long orbit with 1280 symbols on the budget (Figure 28 (e)), with 100000 iterations (the hard stop point in the code). However, we have an additional

cost in ordering the terms yet, but not the same as in the maximum norm strategy. Now we are ordering in each $m = N/2$ iterations, which decreases the total cost expend in ordering with the budget increase.

First of all looking at Figure 30, as expected, we constructed a time-efficient strategy with the delayed maximum norm strategy, and we can see that it is a little faster with a bigger budget (as the time to order the affine form surpass the time operating with more symbols).

The time performance of the delayed maximum norm strategy is very good, the strategy is costing next to the half of time of the maximum norm strategy with 1000 symbols (Figure 30 (b)). The behavior of this time performance graph is getting a slow decrease in time cost with the number of symbols, then we have a strategy here that should not be used with a very small budget.

APPENDIX B. LONG ORBITS EXPERIMENTS WITH CASCADE VARIANTS

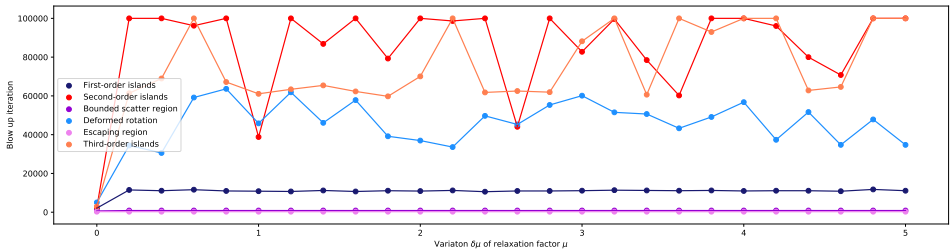
We will continue testing strategies in this section, but here looking at the cascade heuristic. We will continue to explore the diverse set of behaviors for Cremona map to select the most useful strategies. In this section we will look at the cascade variants (§7), to understand these strategies well. A first task is selecting the best relaxation factor for the cascade heuristic.

B.1. **Relaxation factor for cascade variants.**

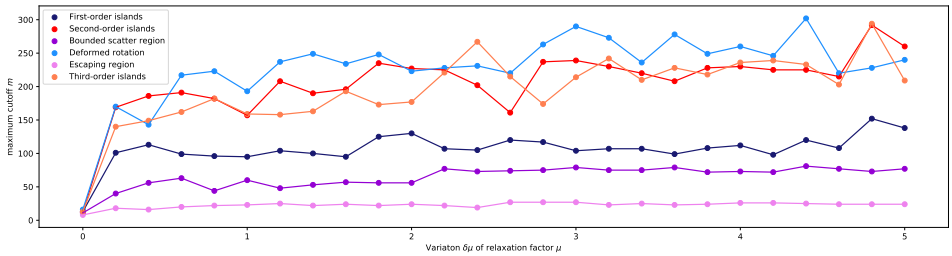
The relaxation cascade looks promising, but a new problem to deal with has come along with this new heuristic: we have introduced a new parameter called *relaxation factor* μ. The relaxation factor is a new degree of freedom in our problem, and the objective of this section is to experiment on this and try to bound a good value for the variation $\delta\mu$. As the relaxation cascade heuristic is a generalization of the *vanilla* cascade heuristic, will use the only cascade, from that point, to mention the use of the relaxation cascade.

Here we experimented with all cases defined in Section §9.2, and selected a good value for relaxation faction, to be used in the following sections.

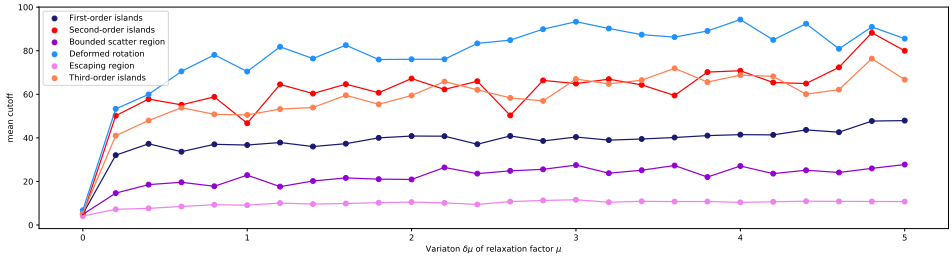
The first and the most important behavior which the relaxation factor introduces could be seen in Figure 31 (a) and Figure 33 (a). If we look at the $\delta\mu = 0$ in both graphs, we can see that the vanilla cascade strategy (i.e. relaxation cascade strategy with $\delta\mu = 0$) blows up very fast if compared with any other point on the graphs. That is, we have for every example a better condensation strategy now. In



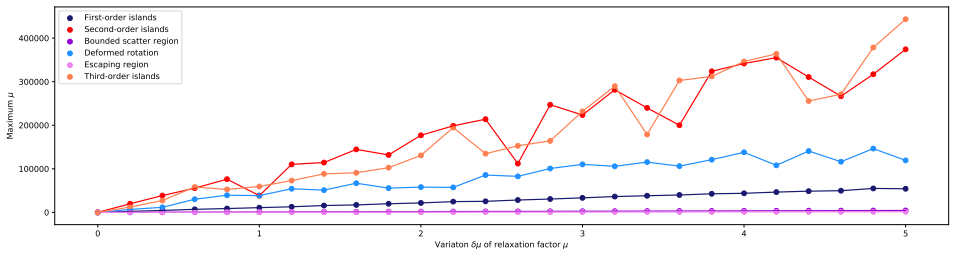
(a) Blow up iteration



(a) Maximum cutoff



(b) Mean cutoff



(c) Maximum relaxation factor

FIGURE 31. Variation of $\delta\mu$ in cascade strategy for Cremona map examples defined in Section §9.2.

conclusion, we have a really big improvement of the vanilla cascade, in the sense of budget utilization, as we can iterate a lot more, even with a small $\delta\mu = 0.2$.

This improvement on the cascade heuristic comes at a very small cost if compared with the benefit. The change in the algorithm is a

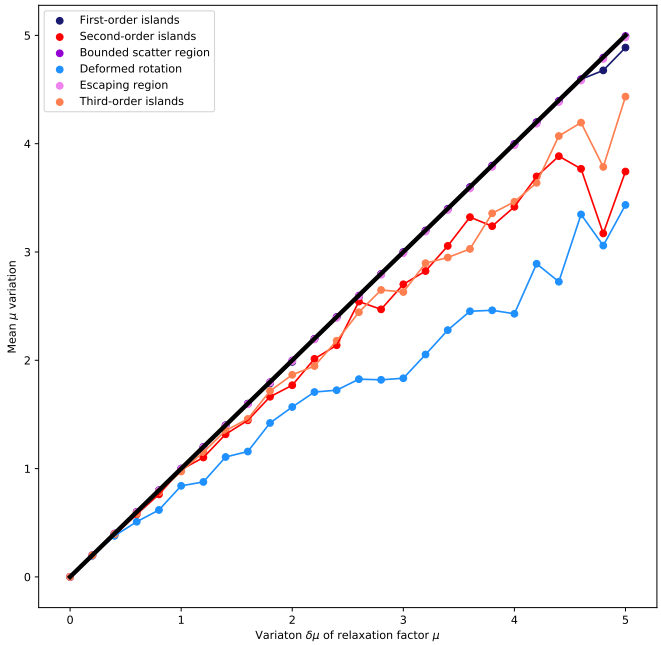
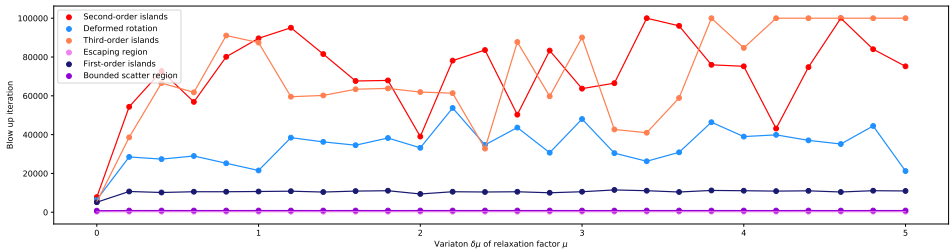


FIGURE 32. Mean μ variation in cascade strategy.

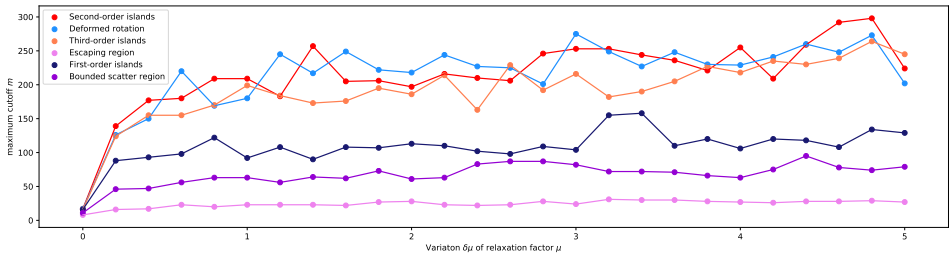
new constant multiplication and a decision block to set the new value for μ. The improvement looking at the examples which do not blow up fast in any strategy (we do not consider here the bounded scatter and the escaping regions) was of five times in the first-order islands, and getting to at least 40 times in the second-order islands.

For the cascade strategy, to set a good value for δμ let's take a look at the other results. Looking at Figure 31, we can see that the maximum and mean values for the cutoff grows up until around $\delta\mu = 3$. The values for the relaxation factor continue to grow up (specifically on second-order and third-order islands), but around $\delta\mu = 3$ it is not so big yet. Looking at the blow-up iteration, around $\delta\mu = 3$ we have very good values for blow-up iteration. With a holistic view of all graphs in Figure 31, we can chose a value around $\delta\mu = 3$.

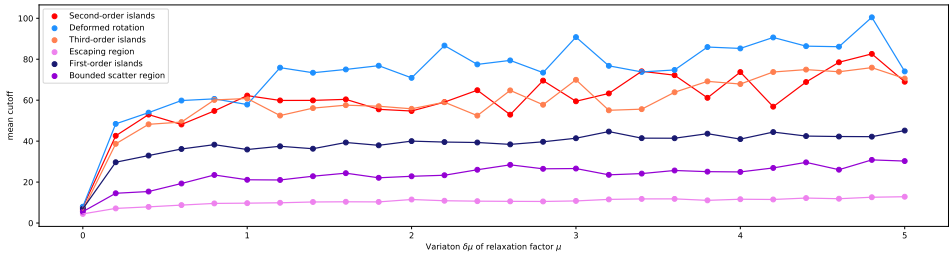
Figure 32 and Figure 34 show to us the relation between the variation of the δμ and the mean variation of the relaxation factor μ. in the line $x = y$, we have a continuous growth rate of μ, and this is not what we expect from this strategy. What we want here is to find a sweet spot where the δμ is not so big, but it is in the region of relaxation cascade where we have subtractions on μ value, that is the region next to the mean budget. In practical terms, we want the first



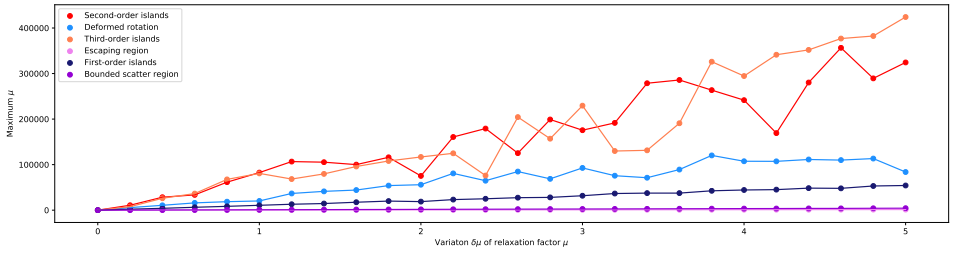
(a) Blow up iteration



(a) Maximum cutoff



(b) Mean cutoff



(c) Maximum relaxation factor

FIGURE 33. Variation of $\delta\mu$ in magnitude cascade strategy for Cremona map examples defined in Section §9.2.

value of $\delta\mu$ where the example curves are with a reasonable distance from the line $x = y$.

Looking at Figure 32, we can get empirically our value for $\delta\mu$. And again, $\delta\mu = 3$ looks a good value, as it is the small value where we have a good distance from the reference line.

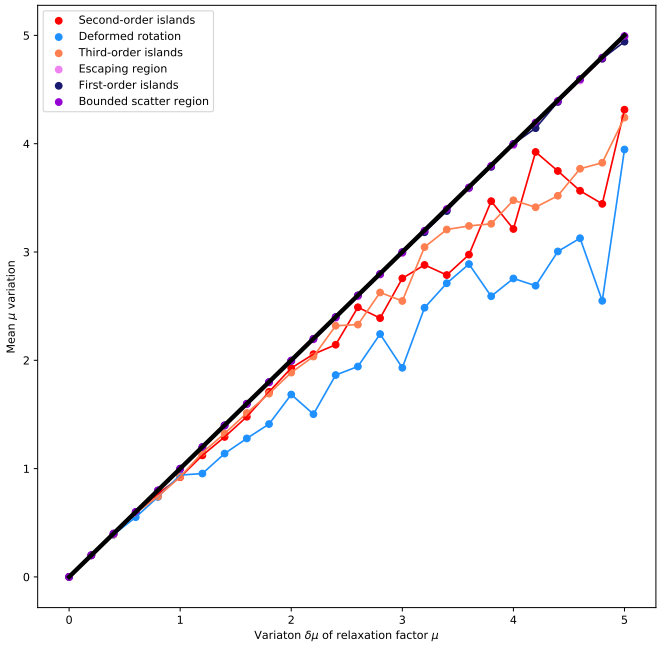
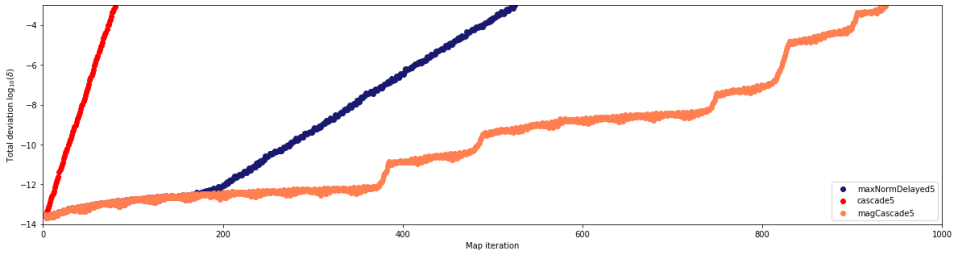


FIGURE 34. Mean μ variation in magnitude cascade strategy.

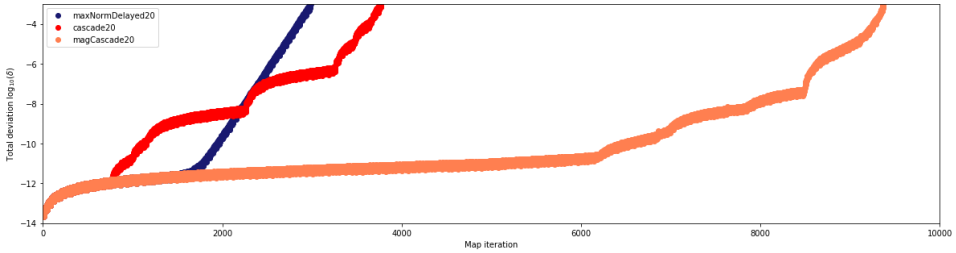
Looking at Figures 33 and 34, now for the magnitude cascade strategy, the same analysis done for cascade can be repeated. The conclusions for magnitude cascade strategy are the same as for cascade strategy, and the graphs now have much more spikes around $\delta\mu = 3$. Finally, we will use for the rest of the long orbits tests the value for $\delta\mu = 3$, which showed good metrics for cascade and magnitude cascade strategies.

B.2. **Cascade and magnitude cascade strategies experiments.** Let's observe the performance difference of delayed maximum norm, cascade, and magnitude cascade strategies looking at the *deformed rotation*. Looking at Figure 35, we can not see a general better behavior anymore, now each strategy is performing well in a range of the budget. In addition, this is not bad behavior, as we have strategies that are useful for a different range of budgets, depending on the application.

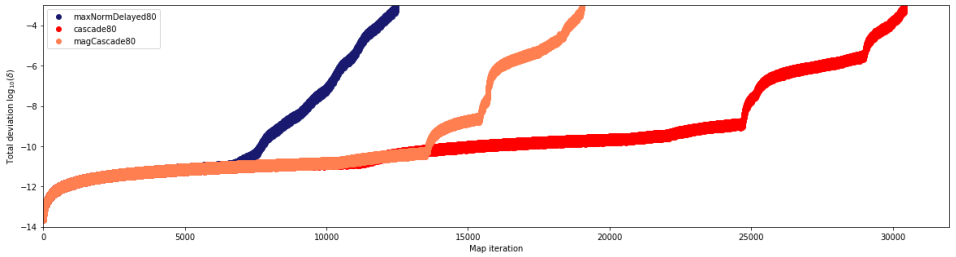
Only with 5 symbols (Figure 35 (a)) in the affine form, the magnitude cascade strategy achieved a better orbit than the other ones. The delayed maximum norm strategy orbit blows up in iteration 525, which is smaller than the magnitude cascade strategy orbit, which blows up after 938 iterations (i.e. we have an orbit approximately



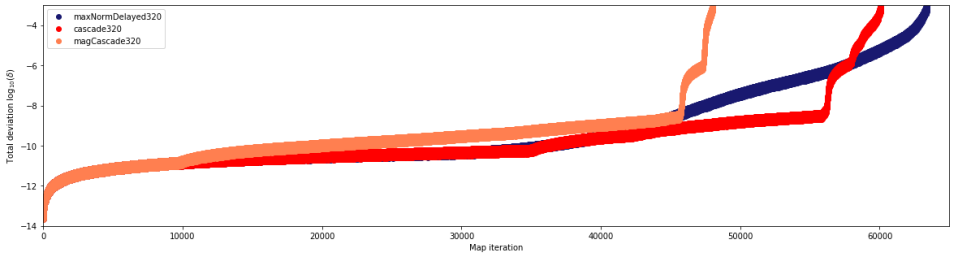
(a) N = 5



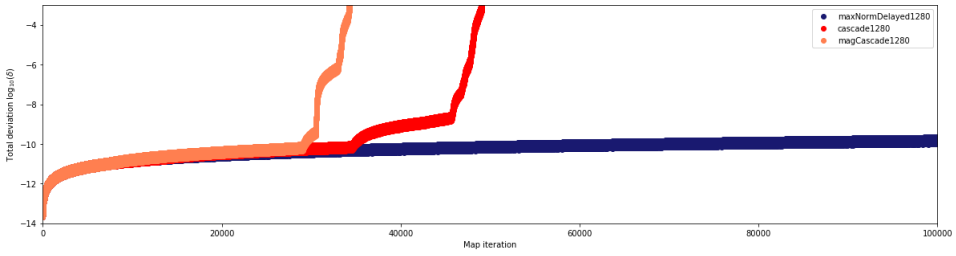
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280

FIGURE 35. Performance of delayed maximum norm, cascade and magnitude cascade strategies based on total deviation for *deformed rotation* example in Cremona map.

80% bigger), and for cascade strategy orbit it is lots of time bigger. With 20 symbols (Figure 35 (b)) we can see similar behavior, with the magnitude cascade strategy better, but now the cascade strategy is getting better than the delayed maximum norm.

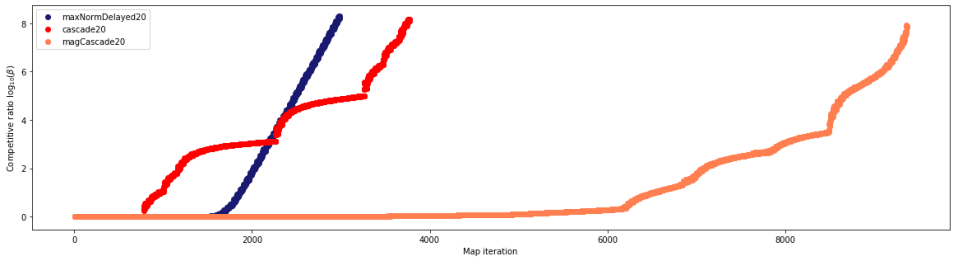
With 80 symbols (Figure 35 (c)) the cascade strategy shows more value and constructs an orbit with more than 30000 symbols and a relatively small budget. In addition, this orbit is close to three times bigger than the delayed maximum norm strategy orbit (which is already a good orbit with more than 10000 symbols).

The delayed maximum norm is a condensation strategy that depends on a reasonable budget to work well. When we have a large budget (Figure 35 (d-e)), we can see the potential of this condensation strategy even when compared with two other elaborate strategies, and it consolidates as a very good strategy. With a budget equal to 1280 symbols, we can build an orbit with size 100000 without suffering a blow-up, and we have a perimeter of 10^{-10} order for total deviation. That is, we had a little inflation in the affine form due to the wrapping effect introduced in the remaining box. This is a much more stable strategy than the cascade strategy (which blows up close to 50000 map iterations) and magnitude cascade strategy (which blows up with more than 30000 map iterations).

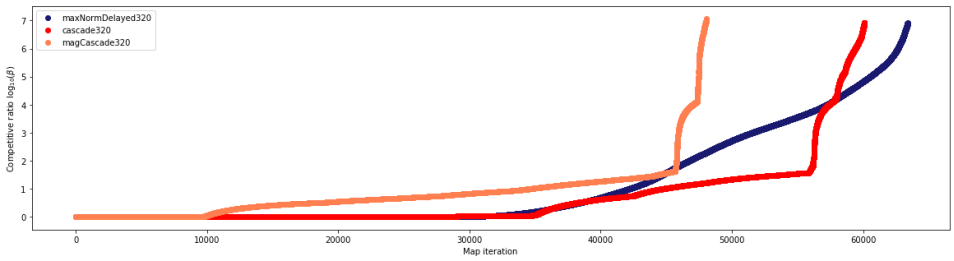
If we take a look at Figure 36 (a), we can see a great gain on β-competitiveness for the magnitude cascade strategy, and it is much better compared with the delayed maximum norm strategy. On the graph, the delayed maximum norm strategy loses a competitive rate close to zero only around 1700 iterations. On the other hand, the magnitude cascade strategy loses the full strategy zonotope perimeter at around 4000. With only 20 symbols the magnitude cascade strategy is great.

Looking at a bigger budget ($N = 320$ on Figure 36 (b)) we can see now a big difference in the behaviors for the β-competitiveness, while the truncate variants had something close to a rate of loss, the cascade variants had a change on that based on the heuristic. As the waterfall flows, we touch on the most important symbols, and this causes a strong increase in the β-competitiveness, which grows more stable until this behavior happens again. This keeps us close to the full strategy wrapping effect much longer, but when we have big total deviation symbols the strategy amplifies fast the remainder box and blows up.

First of all looking at Figure 37, the cascade heuristic is very efficient, as it is based only on the summation of the magnitude of the symbols.

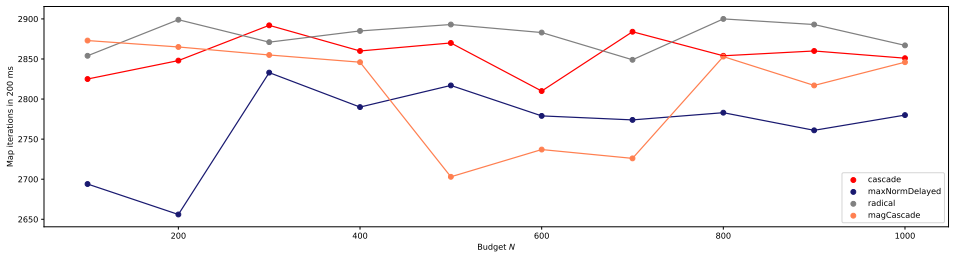


(a) N = 20

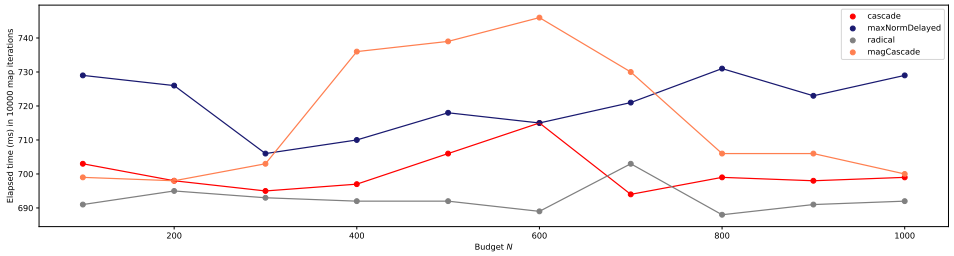


(b) N = 320

FIGURE 36. Competitive ratio of delayed maximum norm, cascade and magnitude cascade strategies for *deformed rotation* example in Cremona map.



(a) Number of iterations for fixed time



(b) Elapsed time for fixed iteration number

FIGURE 37. Time performance of radical, delayed maximum norm, cascade and magnitude cascade strategies for *deformed rotation* example in Cremona map.

We constructed a time-efficient strategy with all strategies in the graph. The time performance of the delayed maximum norm strategy is good, but the cascade variants are better. The cascade strategy in mean is close to the radical strategy (which is a little faster in Figure 37 (b)).

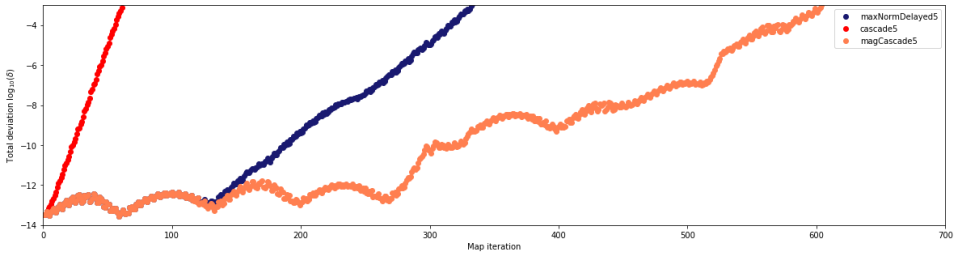
Now we should observe the new strategies when they are applied to a complex behavior in the dynamical system. Looking at Figure 5, we can see a region with very different behavior from a deformed rotation. This region is composed of five closed curves, or islands, that can be seen in Figure 7, which is our *first-order islands* example. Now the effect of quadratic terms is higher, and we expect a bigger wrapping effect due to the non-affine approximations. We shall not see long orbits as we have seen before in *deformed rotation*, as this new problem is harder.

But looking at Figure 38 we can see the same behavior on which strategy is better as in the deformed rotation example. The different behavior is on the full strategy, which blows up with 12022 map iterations, and now we have this upper boundary for the strategies. Here we have the delayed maximum norm strategy reaching an orbit with 11757 symbols, which is very close to the full orbit, and it shows that the delayed maximum norm strategy is very stable.

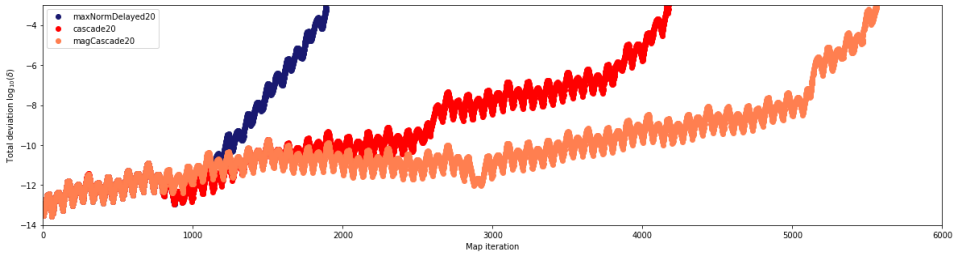
Figure 39 shows how the β-competitiveness is for the delayed maximum norm strategy, and we are very close to the full strategy orbit. As an example, with 10000 map iterations, the strategy has a $\beta = 1.08$. That is, the delayed maximum norm strategy after 10000 iterations is yet competitive with full strategy, and the zonotone has a total deviation only 8% greater than the full strategy. Now with 11000 map iterations, the strategy has a $\beta = 1.23$. We can conclude that the strategy is great here, and the natural wrapping effect of the AA is impacting here.

Now we want to look at the semi-adaptive cascade variants and see if this change on the heuristic could construct good results.

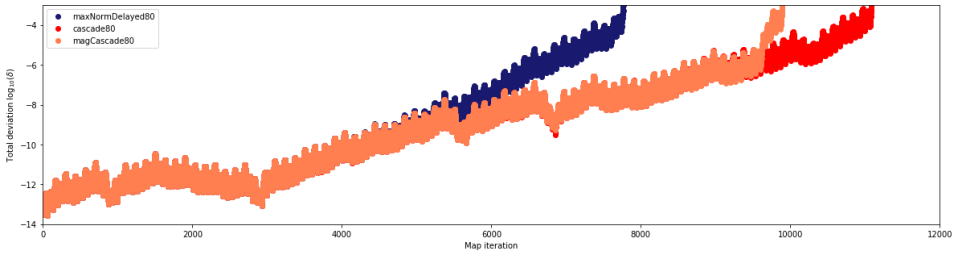
B.3. **Semi-adaptive cascade variants.** We have good strategies for some ranges of the budget now, magnitude cascade strategy for a small budget, cascade for a medium budget, and delayed maximum norm for a big budget. Now we want to look at this first change on cascade heuristic and compare it with the best strategy for each range of budget size. We will look in this section at the semi-adaptive cascade and semi-adaptive magnitude cascade strategies using the variation for relaxation factor $\delta\mu = 3$.



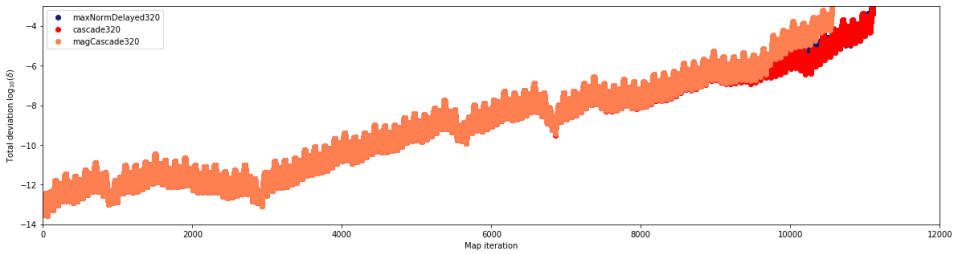
(a) N = 5



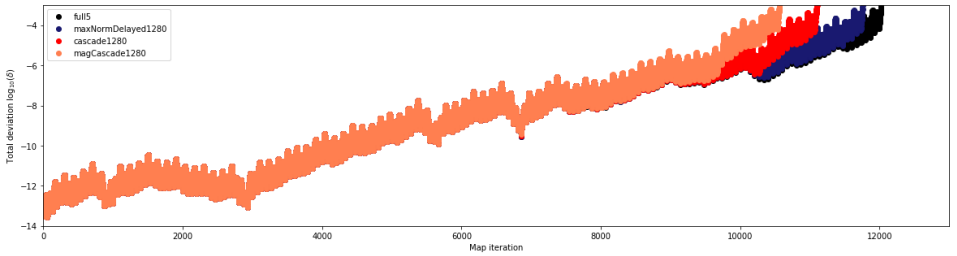
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280 with full strategy

FIGURE 38. Performance of delayed maximum norm, cascade and magnitude cascade strategies based on total deviation for *first-order islands* example in Cremona map.

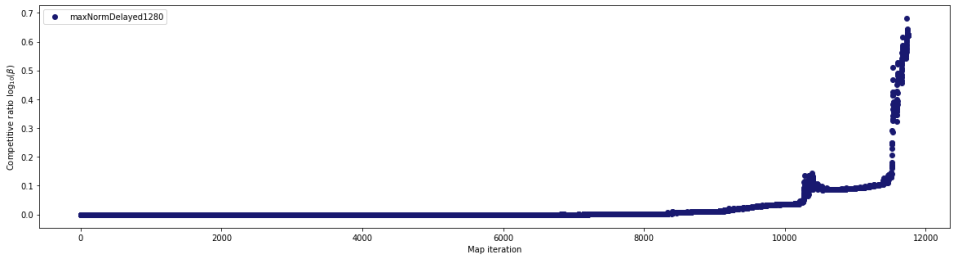


FIGURE 39. Competitive ratio of delayed maximum norm for *first-order islands* example in Cremona map with $N = 1280$.

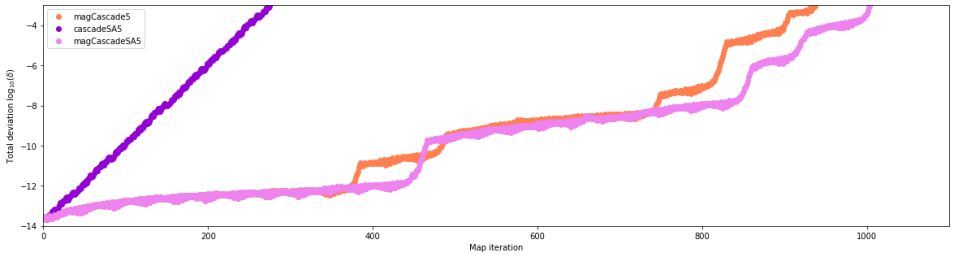
Let's observe the performance difference of semi-adaptive cascade and semi-adaptive magnitude cascade strategies from the best strategy that we have for each budget value looking at the *deformed rotation*. Looking at Figure 40, we can not see a improvement in general of the semi-adaptive variants.

Even with 5 symbols (Figure 40 (a)) in the affine form, the semi-adaptive magnitude cascade strategy does not achieve a relevant improvement in the orbit compared with the magnitude cascade strategy. The magnitude cascade strategy orbit blows up in iteration 938, which is smaller than the semi-adaptive magnitude cascade strategy orbit, which blows up after 1004 iterations. Here we have an orbit only 7% bigger, which is a disposable improvement.

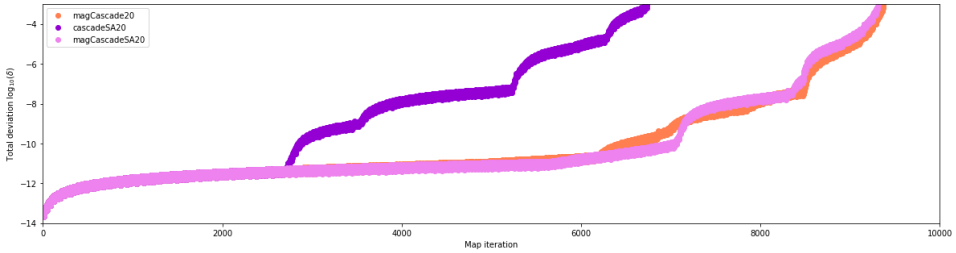
B.4. **Ordered cascade variants.** We still have the same set of good strategies for some ranges of the budget, as the semi-adaptive variants did not work well. The magnitude cascade strategy for a small budget, cascade for a medium budget, and delayed maximum norm for a big-budget are the actual choices. Now we want to look at this final change on cascade heuristic and compare it with the best strategy for each range of budget size. We will look in this section at the ordered cascade and ordered magnitude cascade strategies using the variation for relaxation factor $\delta\mu = 3$.

Let's observe the performance difference of ordered cascade and ordered magnitude cascade strategies from the best strategy that we have for each budget value looking at the *deformed rotation*.

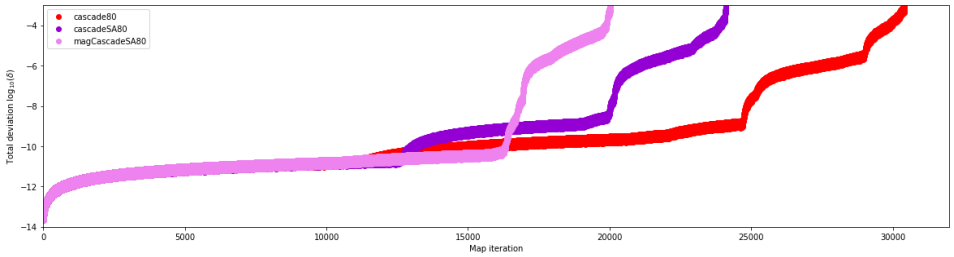
Looking at Figure 41 (a) and (b), we can not see an improvement in these small budget experiments, the magnitude cascade strategy will be selected as a choice for this type of problem.



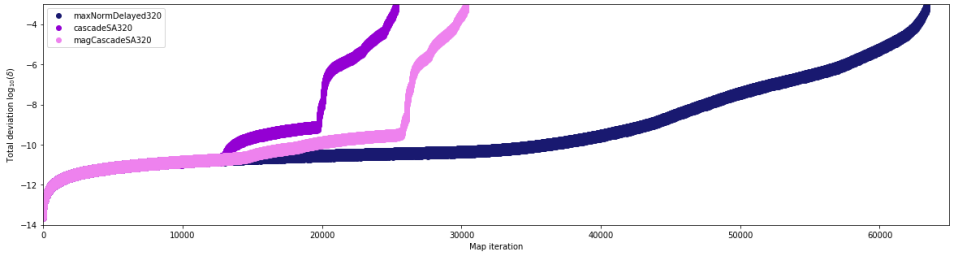
(a) N = 5



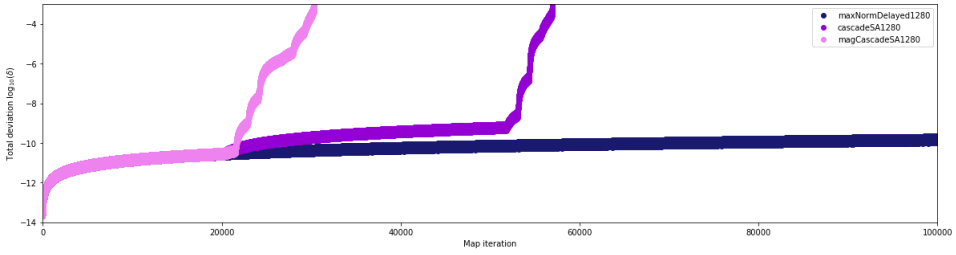
(b) N = 20



(c) N = 80

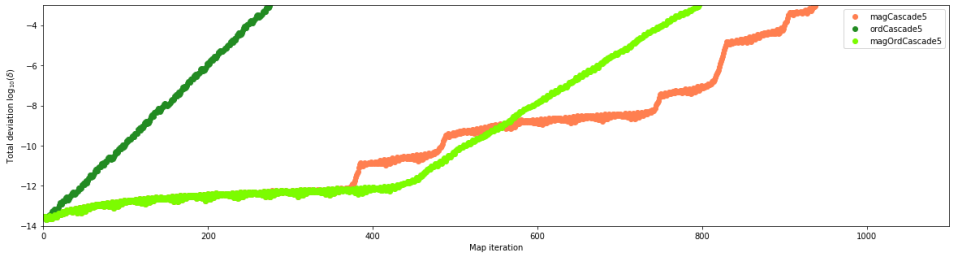


(d) N = 320

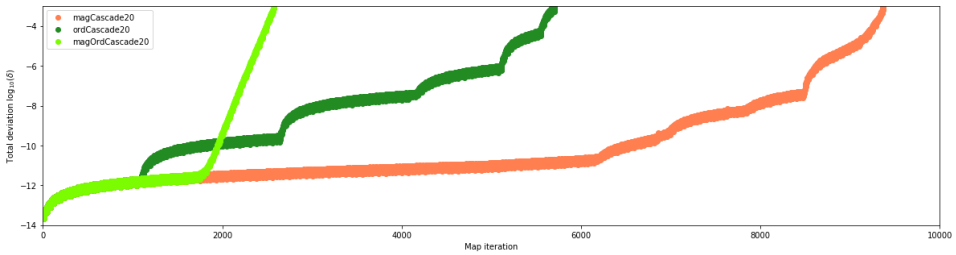


(e) N = 1280

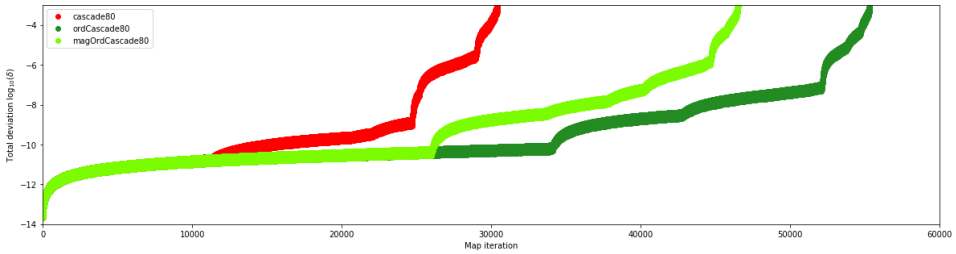
FIGURE 40. Performance of semi-adaptive cascade and semi-adaptive magnitude cascade compared with best strategies for each budget size based on total deviation for *deformed rotation* example in Cremona map.



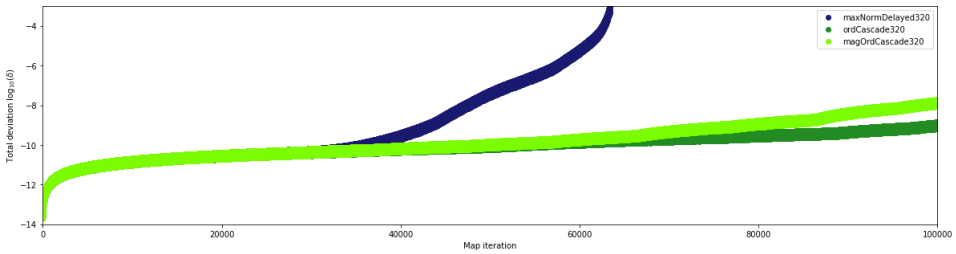
(a) N = 5



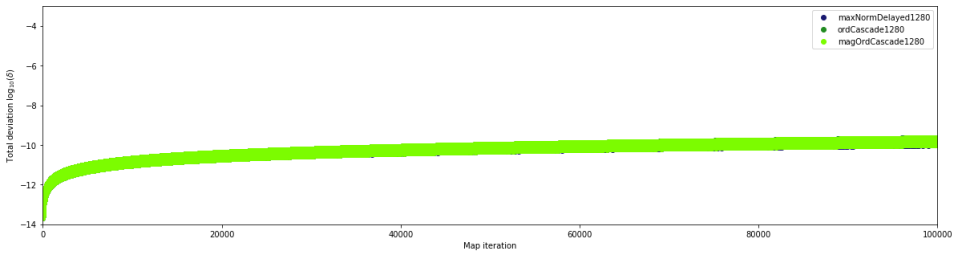
(b) N = 20



(c) N = 80

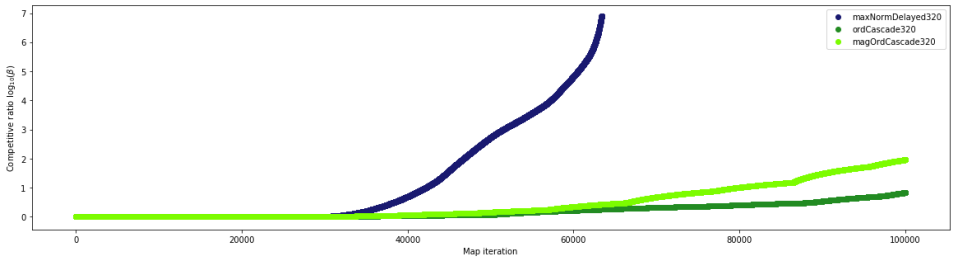


(d) N = 320

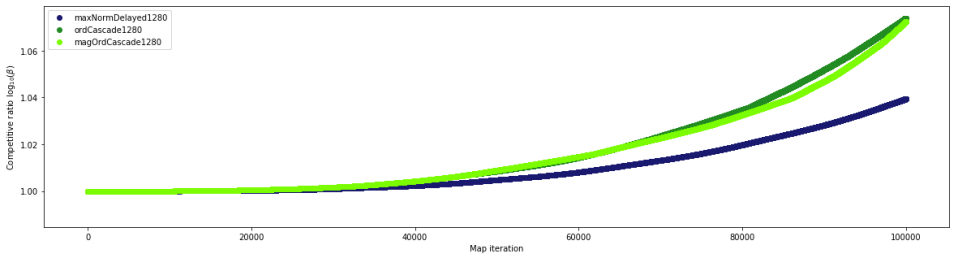


(e) N = 1280

FIGURE 41. Performance of ordered cascade and ordered magnitude cascade compared with best strategies for each budget size based on total deviation for *deformed rotation* example in Cremona map.



(a) N = 320



(b) N = 1280

FIGURE 42. Competitive ratio of delayed maximum norm, ordered cascade and ordered magnitude cascade strategies for *deformed rotation* example in Cremona map.

With 80 symbols on budget (Figure 41 (c)) the ordered cascade strategy shows more value, and constructs an orbit close to 55000 iterations with a relatively small budget. In addition, this orbit is close to two times bigger than the cascade strategy orbit (which is already a great orbit with more than 30000 iterations, that surpassed the delayed maximum norm strategy).

In 320 symbols (Figure 41 (d)), the ordered cascade strategy was able to be better than the delayed maximum norm strategy, and this was a great achievement because the maximum norm works greatly with this big budget. We can see that the orbit easily achieved 100000 iterations, which is a great result. Looking at Figure 41 (e), it is inconclusive which is the best strategy for a big budget, but the three strategies show a good behavior constructing an orbit with more than 100000 iterations.

Looking at the first budget where our best strategy delayed maximum norm lost (N = 320 on Figure 42 (a)) we can see now a big difference in the behaviors for the β-competitiveness. Looking at 30000 iterations, where is clear in the graph that delayed maximum norm is losing the competitiveness, it has approximately 0.0142 of

competitive ratio, which is next to two times greater (worst) than ordered cascade strategy (where the competitive ratio is close to 0.0075). At this moment the perimeter of the delayed maximum norm has already 1.6% more in absolute terms looking at the full strategy and shows why the delayed maximum norm strategy blows up sooner.

Looking at Figure 42 (a), we can see that the ordered cascade strategy has a better competitive ratio than the delayed magnitude cascade strategy (which the value is 0.0119). The ordered magnitude cascade strategy is blowing up half of the delayed maximum norm strategy and the ordered cascade strategy.

The test looking only at the blow-up iteration was inconclusive for 1280 symbols in the budget, but looking at the competitive ratio, we can see the value of the delayed maximum norm strategy with big budgets. This achieves 100000 map iterations with a better competitive ratio, which is 1.04, better than the other two strategies with more than 1.07.

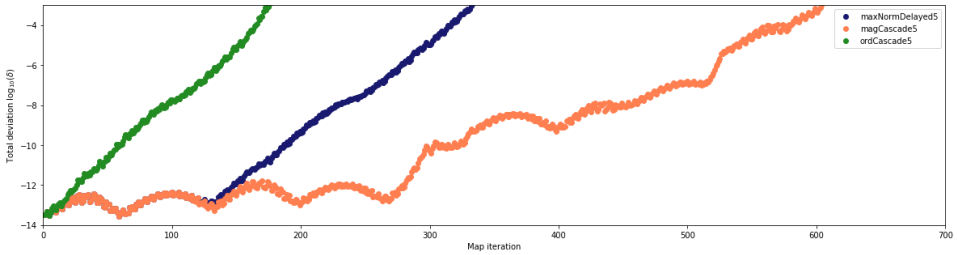
APPENDIX C. BEST STRATEGIES EXPERIMENTS

The objective now is to take a deep look at the best selected strategies applied in a wide range of examples, and confirm the hypothesis that the best strategies are magnitude cascade strategy for small budget size, ordered cascade strategy for medium budget size, and delayed maximum norm for big-budget size.

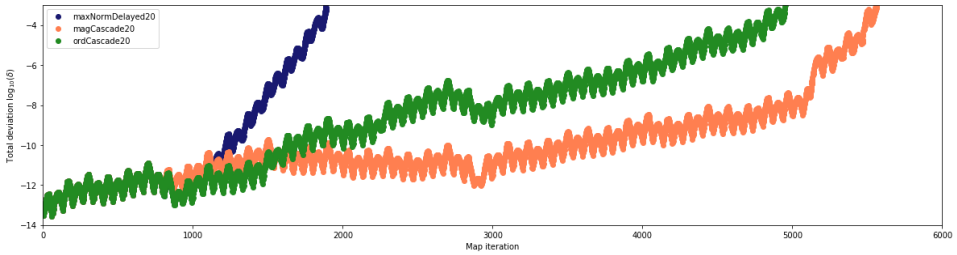
C.1. **First-order islands.** Figure 43 shows how each strategy performs in this problem until the map iteration diverges. Here we can see an oscillatory behavior on all the total deviation graphs, and this behavior is not due to the condensation strategies. As we can see in Figure 43 (e), the full strategy also suffers from that behavior. This behavior comes from the frequent change of closed curves, where, in each island, we have a different outcome of non-affine operations in the wrapping effect.

As expected, we can not get long orbits as in the deformed rotation experiment. In this experiment we have a strong wrapping effect, even the full strategy loses precision fast. We can see this in Figure 43 (e), where the full strategy loses a lot of precision and is also blowing up with 12022 map iterations.

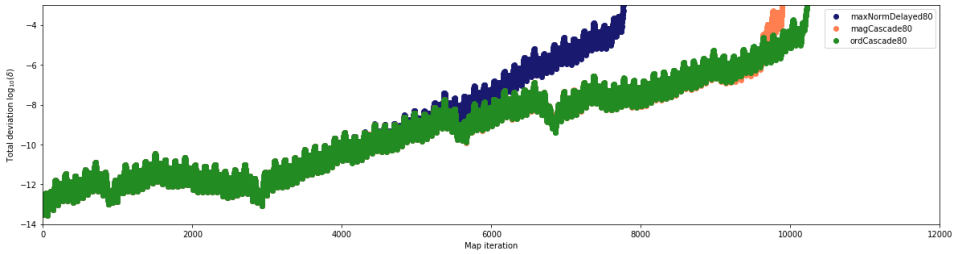
With a small number of symbols on a budget (Figure 43 (a) and (b)), we can see an outstanding performance of the magnitude cascade. The magnitude heuristic made a lot of difference with this small number of symbols, creating a reliable strategy to work when we do not have a budget to spend using a reasonable number of symbols.



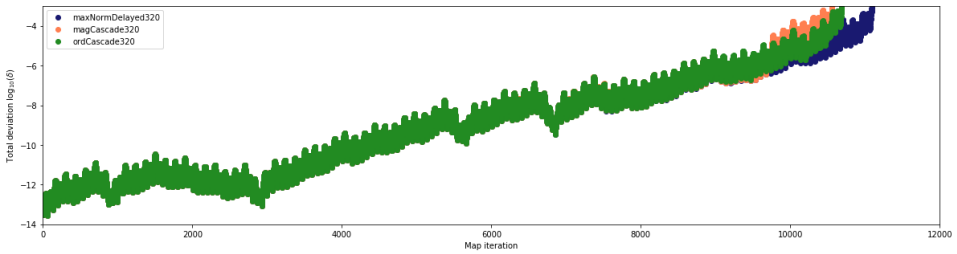
(a) N = 5



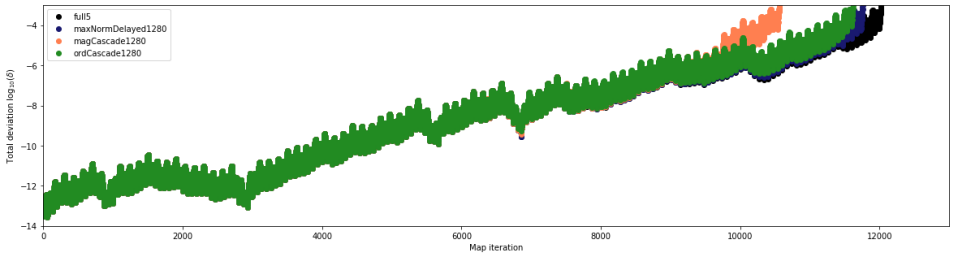
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280 with full strategy

FIGURE 43. Performance of delayed maximum norm, magnitude cascade and ordered cascade strategies based on total deviation for *first-order islands* example in Cremona map.

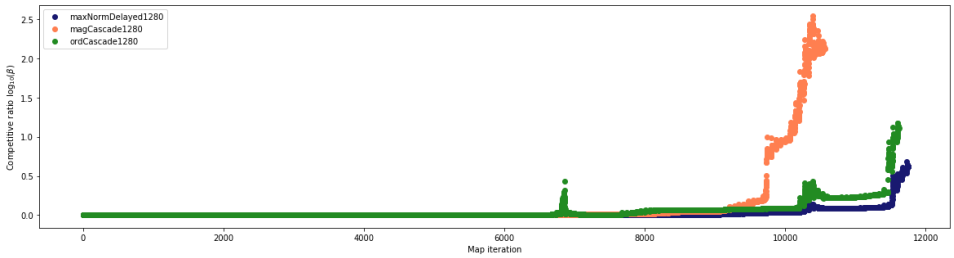


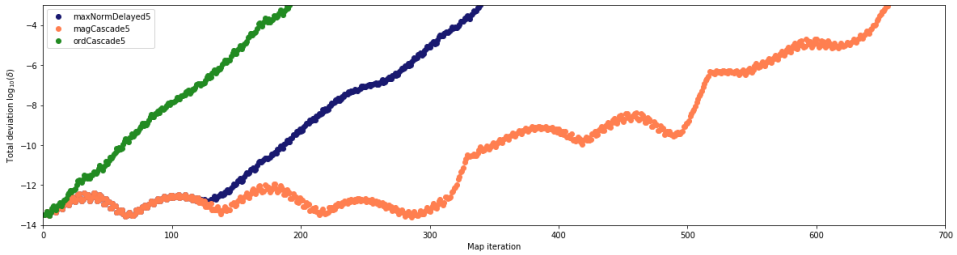
FIGURE 44. Competitive ratio of delayed maximum norm, magnitude cascade and ordered cascade strategies for *first-order islands* example in Cremona map with $N = 1280$.

Looking at the 80 symbols (Figure 43 (c)), ordered cascade strategy is the most useful.

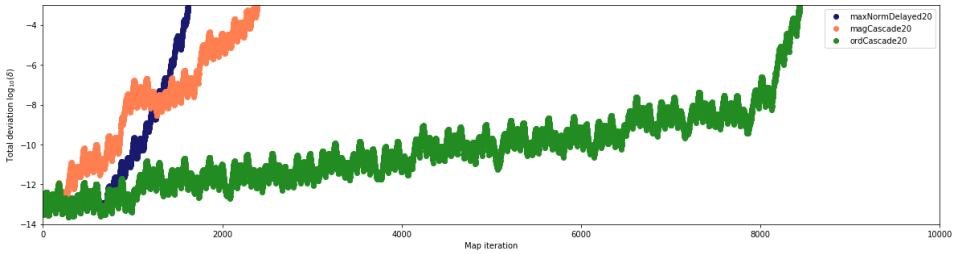
As we know from other experiments, the delayed maximum norm strategy depends a lot on the budget. But now, even with 320 symbols, this strategy was better than the ordered cascade strategy, as we can see in Figure 43 (d). Using 1280 symbols on the budget brings these two strategies very close to the full strategy orbit, which is a signal that we achieved interesting results without expending resources on the full infinity budget.

In this problem, the competitive ratio can be seen in Figure 44, and in that we can see better the behavior than in Figure 43 (e). Because of the division with the full strategy, we can get rid of some oscillatory behavior. With 11000 map iterations, the competitive ratio for the delayed maximum norm strategy is 0.09 (which is small compared with the ordered cascade equal to 0.23). With the competitive ratio of 0.09, we have a zonotope with the perimeter only 23% greater than the full strategy.

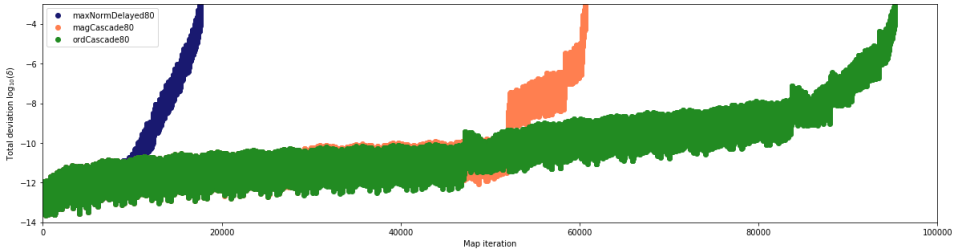
C.2. **Second-order islands.** Figure 45 shows how each strategy performs in this problem until the map iteration diverges. The problem is more stable than the first-order islands, where we have only five islands, and we can construct again orbits with more than 100000 map iterations. With a lot of islands, we expected to see a faster blow up of the condensation strategies, but what we have obtained is a more stable iteration for full strategy. As we can see in Figure 45 (e), the strategies are very cloose to the full, and we can see no difference between then in the graph.



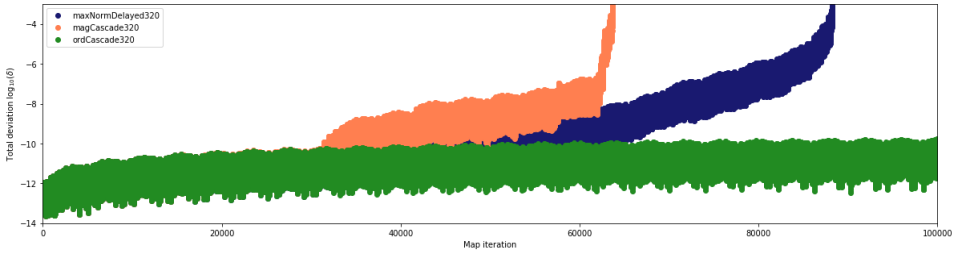
(a) N = 5



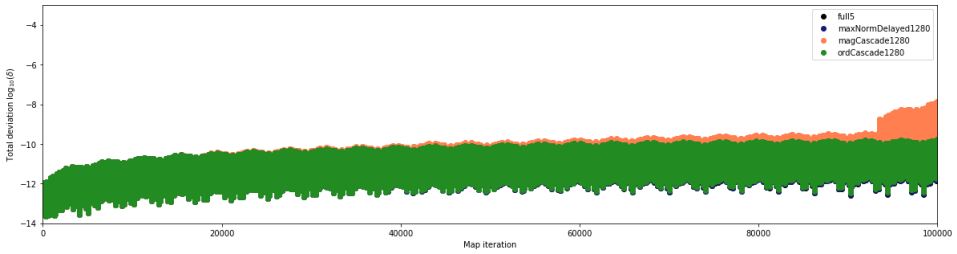
(b) N = 20



(c) N = 80



(d) N = 320



(e) N = 1280 with full strategy

FIGURE 45. Performance of delayed maximum norm, magnitude cascade and ordered cascade strategies based on total deviation for *second-order islands* example in Cremona map.

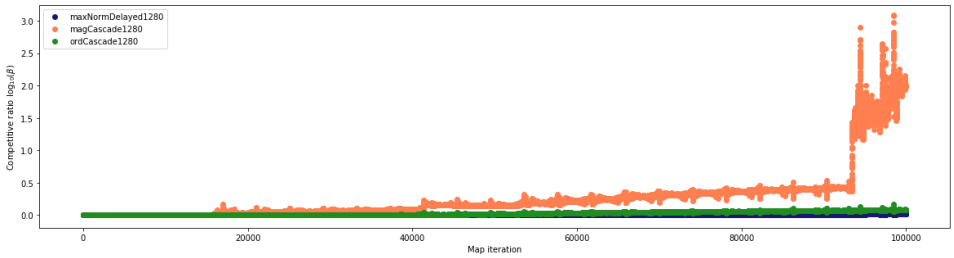


FIGURE 46. Competitive ratio of delayed maximum norm, magnitude cascade and ordered cascade strategies for *second-order islands* example in Cremona map with $N = 1280$.

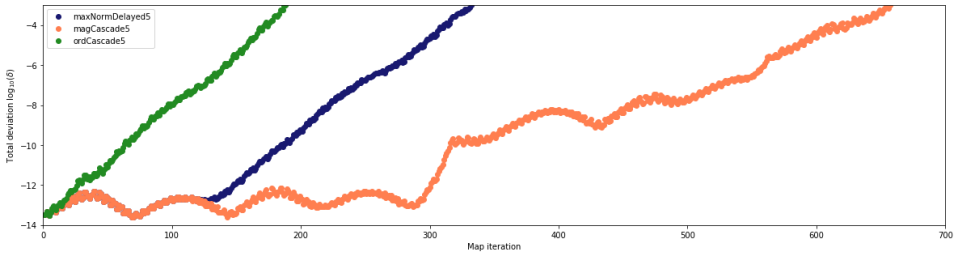
In Figure 45 (a), we can see that the magnitude cascade strategy has a great performance, even with only $N = 5$ symbols in budget. Now the ordered cascade is working better with only 20 symbols in the budget, and we can see a great behavior on Figure 45 (b). Surprisingly, we got more than 8000 iterations with this small budget. Looking at Figure 45 (c), the ordered cascade is doing a great job achieving a orbit with more than 90000 map iteration and only 80 symbols.

With a 320 symbols budget (Figure 45 (d)), we see that ordered cascade has the best performance, but with $N = 1280$ we do not have visibility of the best strategy.

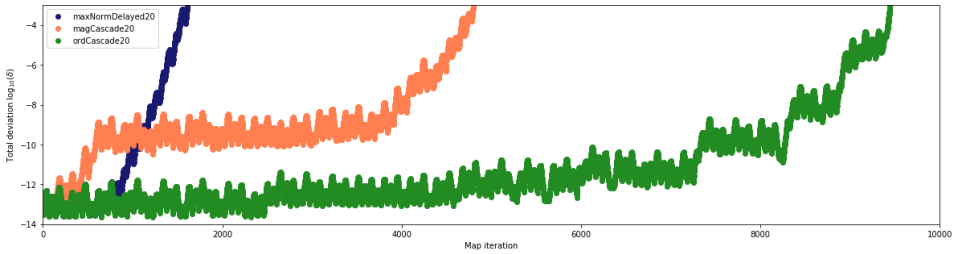
As we saw frequently in experiments, the delayed maximum norm depends a lot on the budget. But now, with $N = 320$ in Figure 45 (d), we can see the best performance of this strategy. We constructed an orbit with more than 80000 steps before the blow up. It is not so good as the ordered cascade, but the value of the delayed maximum norm is always with more than 1000 symbols on the budget.

Looking at the competitive ratio (Figure 46), we have a fierce dispute between the two strategies. However, the competitive ratio for the delayed maximum norm strategy is only 0.01, which is eight times smaller than the ordered cascade, showing again the value of the delayed maximum norm strategy with big budgets.

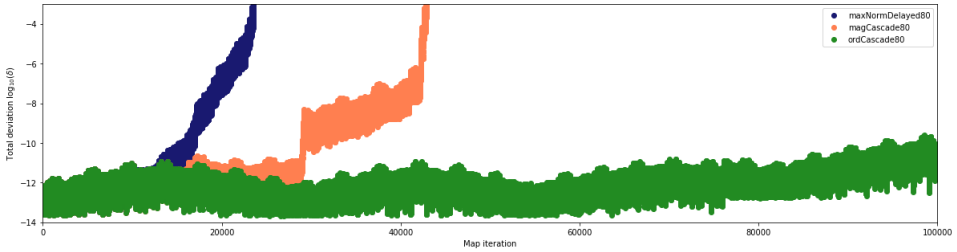
C.3. **Third-order islands.** Figure 47 shows a very similar behavior of the second order islands as we saw in Figure 45. We can see a very stable behavior on the full strategy, and it is not losing a relevant amount of precision. The delayed maximum norm and ordered cascade strategies are confounding with full graph on Figure 47 (e). Again, the ordered cascade strategy is very competitive. With a big



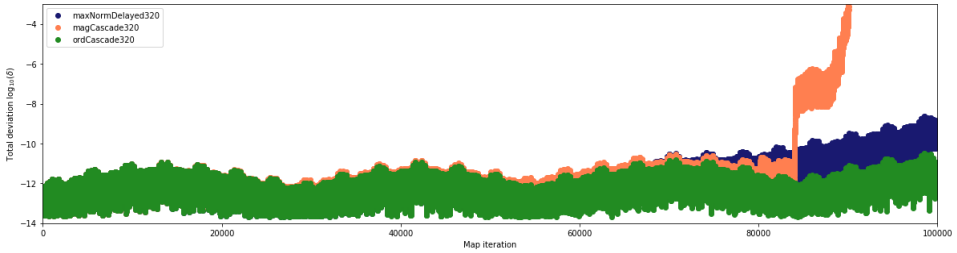
(a) N = 5



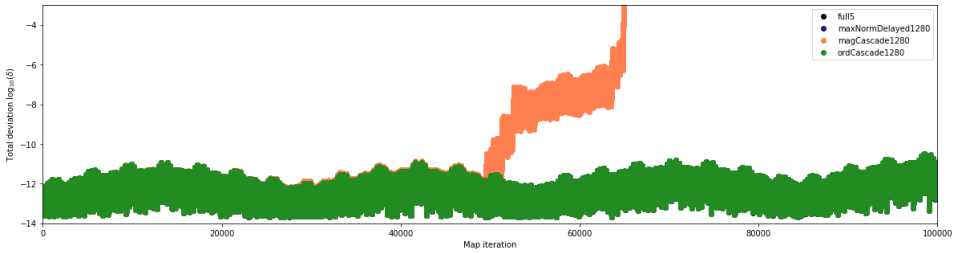
(b) N = 20



(c) N = 80

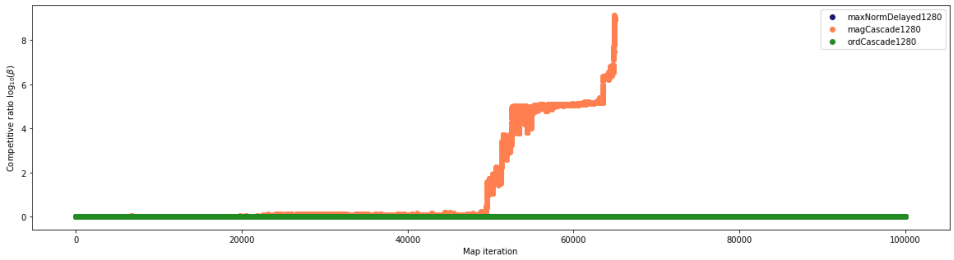


(d) N = 320

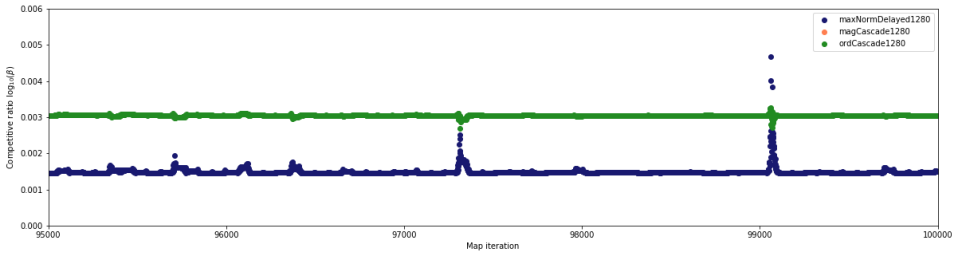


(e) N = 1280 with full strategy

FIGURE 47. Performance of delayed maximum norm, magnitude cascade and ordered cascade strategies based on total deviation for *third-order islands* example in Cremona map.



(a) N = 1280



(b) N = 1280 with zoom

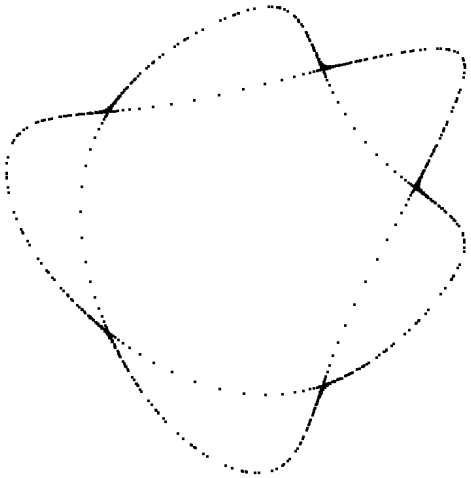
FIGURE 48. Competitive ratio of delayed maximum norm, magnitude cascade and ordered cascade strategies for *third-order islands* example in Cremona map with N = 1280.

budget, we can see how great the delayed maximum norm is, and now we achieve 100000 map iterations with zonotope total deviation of 10^{-11}.

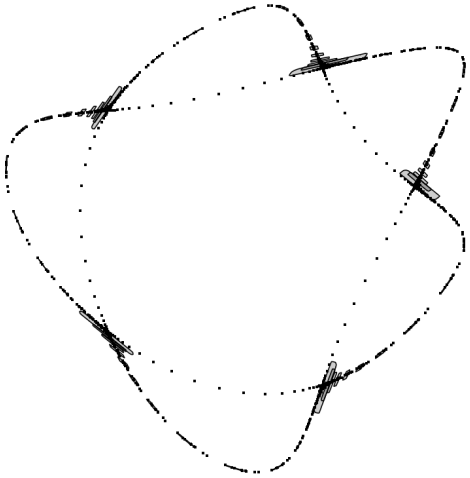
Looking at Figure 48 (a), the delayed maximum norm and ordered cascade strategies are very close on the competitive ratio. We need to zoom on Figure 48 (b) to see a small difference between the two strategies and how close they are to the full. The delayed maximum norm strategy passed 100000 map iterations (with a competitive ratio of 0.0014), which brings to us a zonotope with the perimeter almost identic to the full strategy, only 0.3% greater.

C.4. **Bounded scatter region.** Now, we will work with the scattered region that can be seen in Figure 49 (a). According to Hénon [23], this is interesting because there are invariant curves surrounding the five islands, so that the scattered set cannot escape to infinity: it is constrained to remain in a finite region of the plane, limited by an inner curve, an outer curve, and the islands.

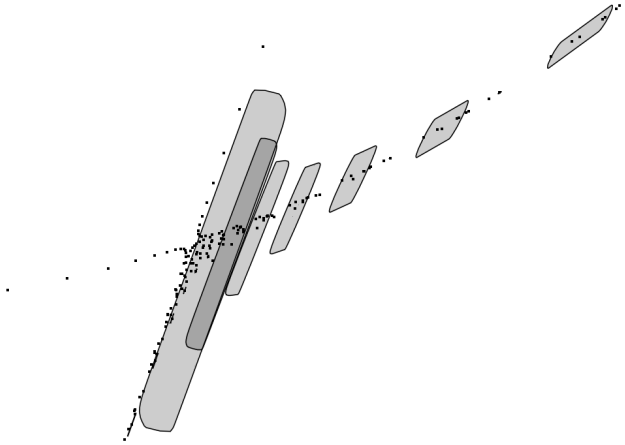
Observing how delayed maximum norm performs in Figure 49 (b), we can see a fast blow up in the orbit, with a little more than 1000 iterations. In the region next to the unstable fixed point, the



(a) Full strategy 900 map iterations

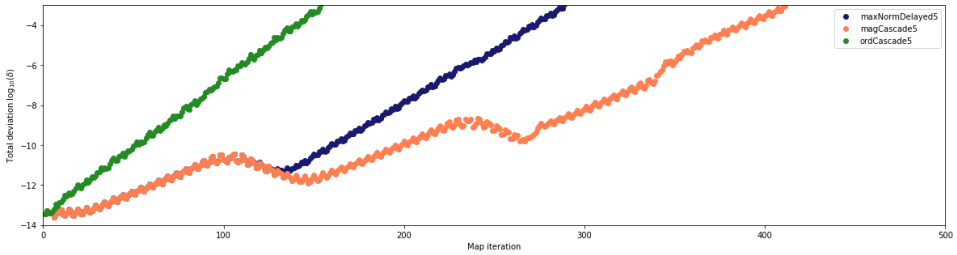


(b) Delayed maximum norm N = 320 and 1020 map iterations

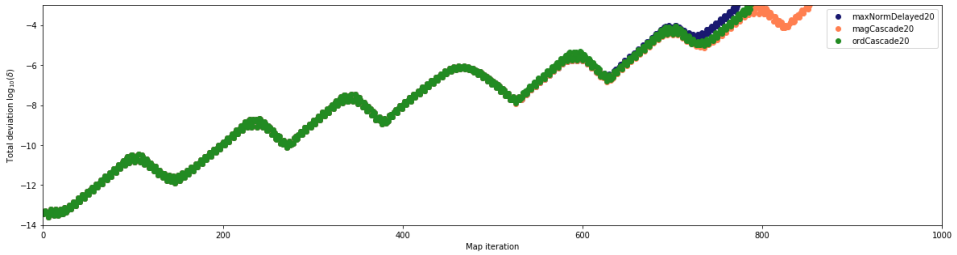


(c) Zoom in (b), next to bottom unstable fixed point

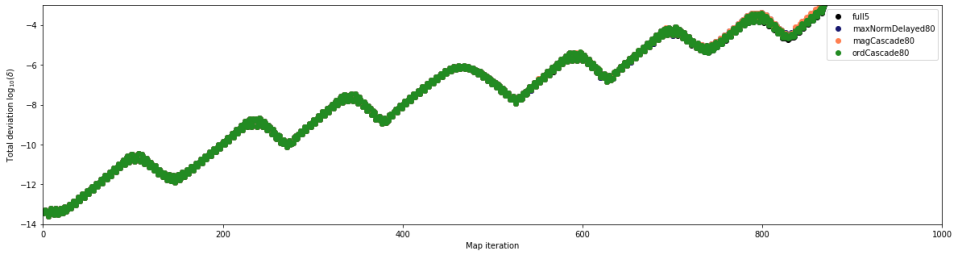
FIGURE 49. Reliable level curve for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.718, y_0 = 0.0$.



(a) N = 5



(b) N = 20



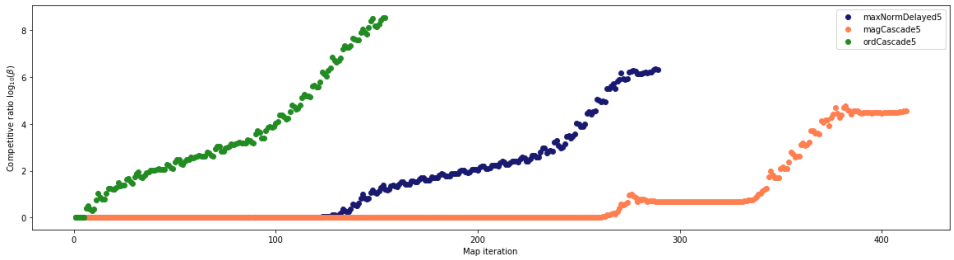
(c) N = 80 with full

FIGURE 50. Performance of delayed maximum norm, magnitude cascade and ordered cascade strategies based on total deviation for *bounded scatter region* example in Cremona map.

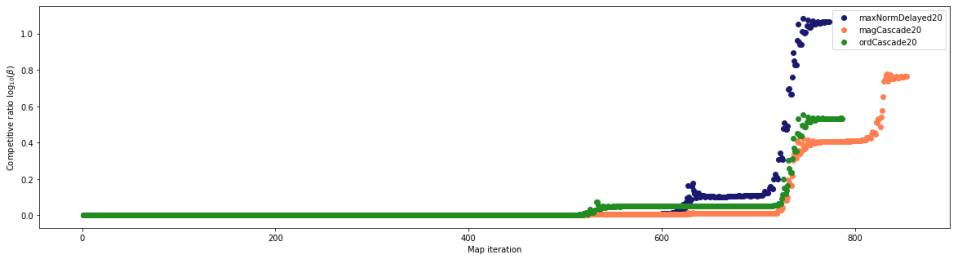
zonotope grows its total deviation very fast, if it touches the fixed point (Figure 49 (c)).

This scattered points set is chaotic, and by nature extremely divergent: any error, however small, in the position of a point is exponentially amplified by repeated application of the mapping, and soon results in completely different positions of individual points [23].

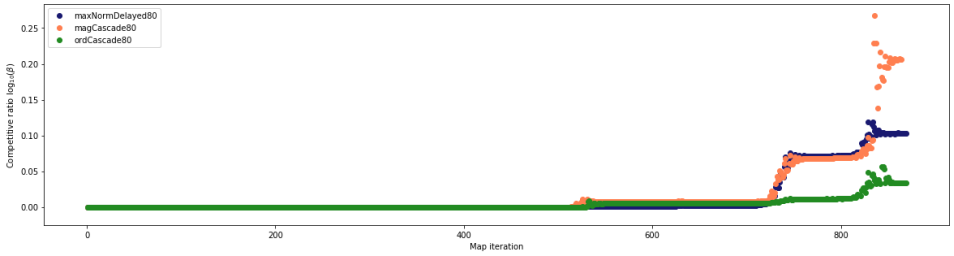
The first clear thing from this experiment is that full strategy blows up very fast. We need only 869 steps to a zonotope to get big enough and to touch the escape region of the map. What we can see from Figure 50 (c) is this behavior of fast-growing for the full strategy total deviation.



(a) N = 5



(a) N = 20



(a) N = 80

FIGURE 51. Competitive ratio of delayed maximum norm, magnitude cascade and ordered cascade strategies for *bounded scatter region* example in Cremona map.

In this experiment, we can see the strategies in Figure 50. We will not have a budget greater than 80 here, for it does not make sense looking at the full blowing-up with only 869 iterations. Even looking at only 80 symbols, in Figure Figure 50 (c), we can see no relevant difference in the strategies orbits, and we need to look at the competitive ratio to have more clarity on the behaviors.

For small budgets (Figure 50 (a) and (b)) we can see the value of the magnitude cascade strategy, generating long orbits (compared with full) with a restricted number of symbols to process.

From Figure 50 (c) it is hard to see anything about the strategies. Every strategy seems to perform well, and we need to use the competitive ratio to clarify that. Observing Figure 51 (a), with 5 symbols

FIGURE 52. 290 map iterations with magnitude cascade strategy for Cremona map with $\cos(\alpha) = 0.24$ and initial point $x_0 = 0.82, y_0 = 0.0$.

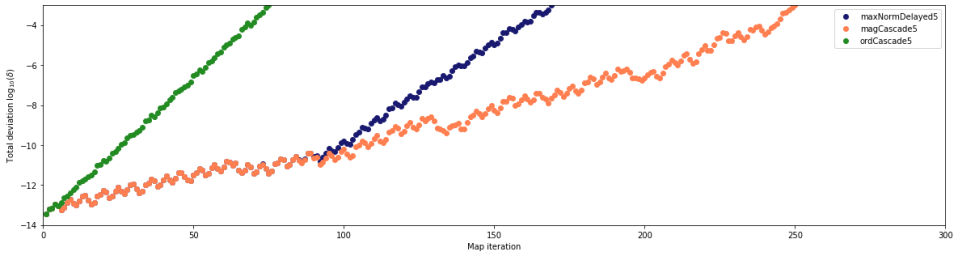
on the budget, we have magnitude cascade strategy sensibly better. On the other hand, with 20 symbols, this strategy continues better, but the other two get a better competitive ratio (Figure 51 (b)).

Looking in Figure 51 (c), we can see how next the competitive ratio is to the full strategy in this experiment. For visualization, full have the competitive ratio 0 by definition. We can see the better strategy, ordered cascade, that blows up with $\log_{10}(\beta) = 1.0011$, and a blow-up iteration equal to the full strategy. The delayed maximum norm strategy is a little worse blowing-up with $\log_{10}(\beta) = 1.0013$, and a blow-up iteration equal to the full strategy. We conclude here that the two strategies performed great in the problem and the blow-up was by touching an escape region, not the condensation on AA, as they diverged together with the full.

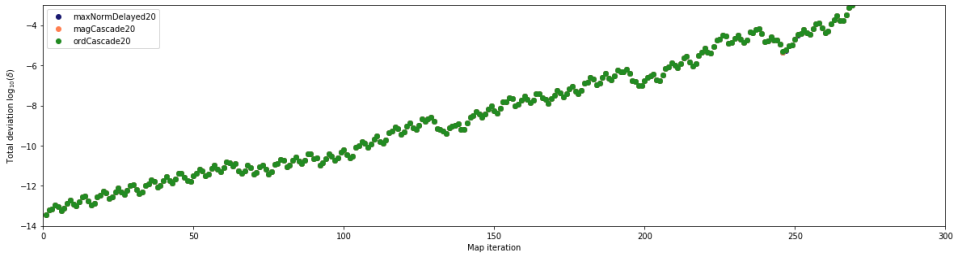
C.5. **Escaping region.** For the final experiment with this Cremona α-map, we shall explore an escaping region, as we can see in Figure 12. In this escaping orbit, it was proved by AA that this orbit escapes after 286 iterations. We can see the magnitude cascade strategy touching the escape region with 286 map iterations and the next iterations of the zonotope blowing up faster.

Now we can experiment with not much more than 286 steps in the mapping orbit, for the escaping orbit eventually touches a point strongly attracted by ∞. This behavior was seen even in full strategy and generates a lot of wrapping effect and a blow-up in the zonotope's total deviation.

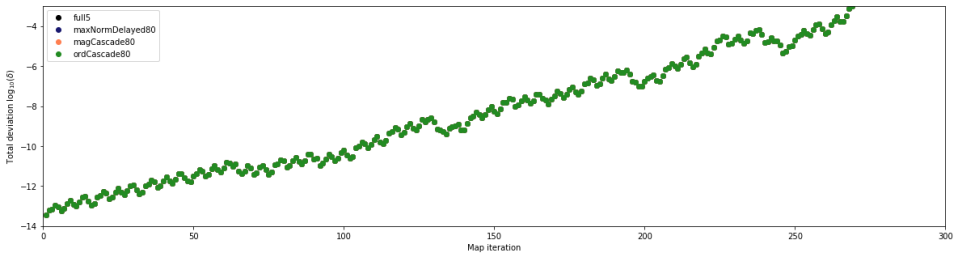
In Figure 53 (a) we can see a great behavior of magnitude cascade strategy, which has a great performance with only 5 symbols. If we need a faster algorithm and can work with only 5 symbols on the budget, we could use the magnitude cascade strategy, and get a good orbit with a small budget.



(a) N = 5



(b) N = 20



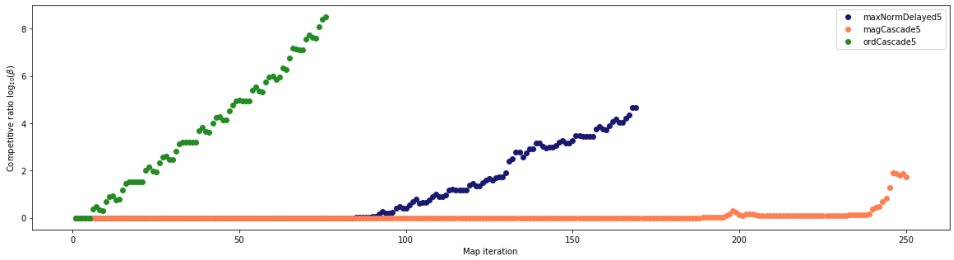
(c) N = 80 with full

FIGURE 53. Performance of delayed maximum norm, magnitude cascade and ordered cascade strategies based on total deviation for *escaping region* example in Cremona map.

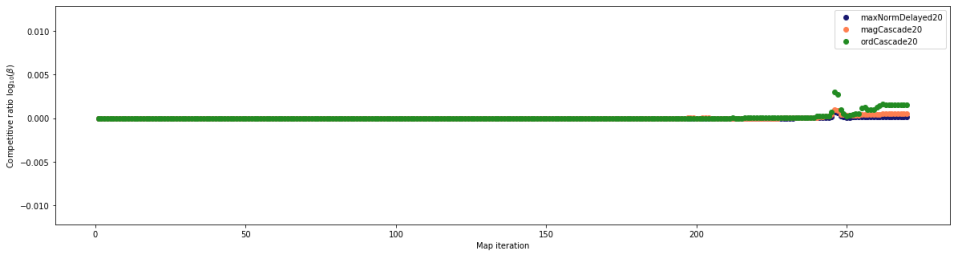
Figure 53 (b) and (c) do not show to us anything. Every strategy with more than 20 symbols can construct an orbit exactly equal to full before escaping. The only difference will be in the zonotope perimeter, which is best if looked in the competitive ratio analysis.

In Figure 54 (a), with 5 symbols of budget, it is expected to see magnitude cascade strategy great performance. In Figure 54 (b), we have the delayed maximum norm strategy a little better than the others.

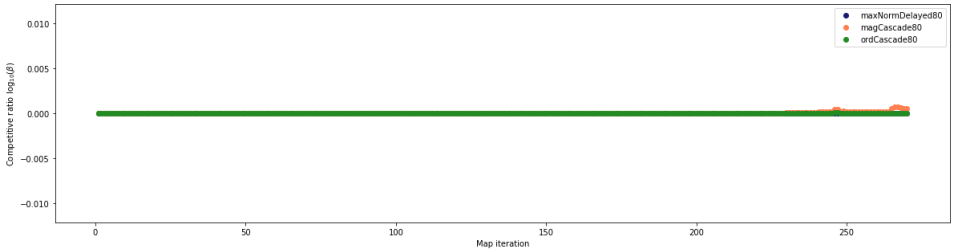
With 80 symbols in the budget, there is practically no difference between strategies, as we can see in Figure 54 (c). The last zoom in (Figure 54 (d)) show us that only ordered cascade ($\beta = 6.6 \times 10^{-7}$



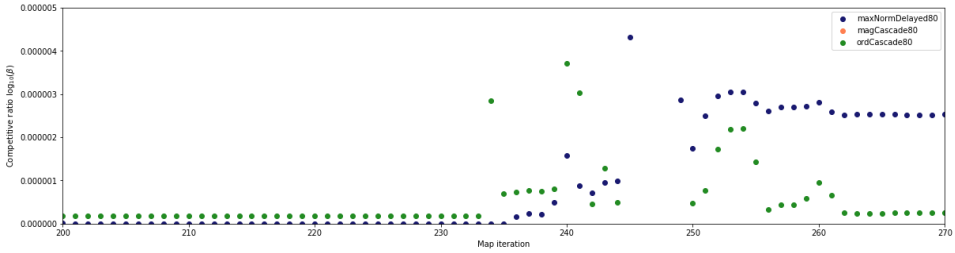
(a) N = 5



(a) N = 20



(a) N = 80



(a) N = 80 with zoom

FIGURE 54. Competitive ratio of delayed maximum norm, magnitude cascade and ordered cascade strategies for *escaping region* example in Cremona map.

in the iteration 260 zonotope) performs a little better than the other three, which is expected for ordered cascade with a medium budget.

Experiment	a	b	Period (initial value)		
			Orbit 1	Orbit 2	Orbit 3
e01	1.40	0.30	∞? ($x = 0.0, y = 0.0$)	none	none
e02	1.00	0.30	4 ($x = 0.0, y = 0.0$)	none	none
e03	1.04	0.30	8 ($x = 0.0, y = 0.0$)	none	none
e04	1.055	0.30	16 ($x = 0.0, y = 0.0$)	none	none
e05	1.057	0.30	32 ($x = 0.0, y = 0.0$)	none	none
e06	1.0579	0.30	64 ($x = 0.0, y = 0.0$)	none	none
e07	1.058	0.30	128 ($x = 0.0, y = 0.0$)	none	none
e08	1.3562	0.2586	7 ($x = 0.2643, y = 0.0263$)	18 ($x = -0.1492, y = 0.0692$)	36 ($x = 0.2956, y = -0.0907$)
e09	0.98	0.4415	8 ($x = 0.4302, y = 0.0458$)	12 ($x = 0.2643, y = -0.0282$)	20 ($x = -0.38, y = 0.0$)
e10	0.972	0.4575	12 ($x = 0.0809, y = 0.0673$)	16 ($x = -0.2253, y = -0.2097$)	∞? ($x = -0.38, y = 0.0$)
e11	0.97	0.466	8 ($x = -0.1414, y = -0.2195$)	∞? ($x = 0.5746, y = 0.7658$)	∞? ($x = -0.06453, y = 0.0153$)

TABLE 1. Experiments parameters, initial points and periods for Hénon map.

APPENDIX D. USE OF AFFINE ARITHMETIC IN ATTRACTOR ORBITS WITH HÉNON MAP

Experiment tables for the attractor orbits with Hénon map.

Experiment	Period	Proved period (proof iteration)										
		full	truncation	delayed truncation	maximum norm	delayed maximum norm	cascade	magnitude cascade	semi-adapt cascade	semi-adapt magnitude cascade	ordered cascade	ordered magnitude cascade
e01 $a = 1.40, b = 0.30$	∞?	none	none	none	none	none	none	none	none	none	none	none
e02 $a = 1.00, b = 0.30$	4	8(229)	none	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)
e03 $a = 1.04, b = 0.30$	8	8(149)	none	8(145)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)
e04 $a = 1.055, b = 0.30$	16	16(531)	none	16(279)	16(528)	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)
e05 $a = 1.057, b = 0.30$	32	32(891)	none	32(295)	32(508)	32(890)	32(891)	32(891)	32(891)	32(891)	32(891)	32(891)
e06 $a = 1.0579, b = 0.30$	64	none	none	none	none	64(192)	64(194)	64(185)	64(200)	64(193)	64(192)	64(193)
e07 $a = 1.058, b = 0.30$	128	none	none	none	none	128(823)	128(697)	128(699)	128(583)	128(647)	128(827)	128(934)
e08 $a = 1.3562, b = 0.2586$	7	7(169)	none	7(90)	7(116)	7(169)	7(141)	7(78)	7(169)	7(169)	7(169)	7(169)
	18	none	none	none	none	none	none	none	none	none	none	none
	36	none	none	none	none	none	none	none	none	none	none	none
e09 $a = 0.98, b = 0.4415$	8	none	none	8(415)	none	8(2015)	16(2031)	16(2031)	16(2031)	16(2031)	16(2031)	16(2031)
	12	24(431)	none	12(131)	12(1080)	12(466)	24(478)	24(431)	24(478)	24(431)	24(431)	24(431)
	20	none	none	none	none	20(1102)	40(2465)	40(2465)	40(2465)	40(2465)	40(2465)	40(2465)
e10 $a = 0.972, b = 0.4575$	12	12(73)	none	12(77)	12(73)	12(73)	12(72)	12(66)	12(79)	12(79)	12(93)	12(73)
	16	none	none	none	none	none	16(2486)	none	16(1727)	16(913)	none	16(1119)
	∞?	none	none	none	none	none	none	none	none	none	none	none
e11 $a = 0.97, b = 0.466$	8	none	none	8(415)	none	16(2104)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)
	∞?	none	none	none	none	none	none	none	none	none	none	none
	∞?	none	none	none	none	none	none	none	none	none	none	none

TABLE 2. Proof of Hénon map attractor periodic points using AA condensation strategies with rough initial interval precision equal to 10^{-7}.

Experiment	Period	Proved period (proof iteration)										
		full	truncation	delayed truncation	maximum norm	delayed maximum norm	cascade	magnitude cascade	semi-adapt cascade	semi-adapt magnitude cascade	ordered cascade	ordered magnitude cascade
e01 $a = 1.40, b = 0.30$	∞?	none	none	none	none	none	none	none	none	none	none	none
e02 $a = 1.00, b = 0.30$	4	8(229)	52(173)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)	8(229)
e03 $a = 1.04, b = 0.30$	8	8(149)	none	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)	8(149)
e04 $a = 1.055, b = 0.30$	16	16(531)	none	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)	16(531)
e05 $a = 1.057, b = 0.30$	32	32(891)	none	32(736)	32(891)	32(891)	32(891)	32(891)	32(891)	32(891)	32(891)	32(891)
e06 $a = 1.0579, b = 0.30$	64	none	none	none	none	64(1139)	64(1139)	64(1139)	64(1139)	64(1139)	64(1139)	64(1139)
e07 $a = 1.058, b = 0.30$	128	none	none	none	none	128(2867)	128(2867)	128(2867)	128(2867)	128(2867)	128(2867)	128(2867)
e08 $a = 1.3562, b = 0.2586$	7	7(134)	56(110)	7(134)	7(134)	7(134)	7(134)	7(134)	7(134)	7(134)	7(134)	7(134)
	18	36(461)	none	none	18(333)	36(461)	36(461)	36(461)	36(461)	36(461)	36(461)	36(461)
	36	none	none	none	none	none	none	none	none	none	none	none
e09 $a = 0.98, b = 0.4415$	8	none	none	8(991)	none	16(2031)	16(2031)	16(2031)	16(2031)	16(2031)	16(2031)	16(2031)
	12	24(431)	none	12(347)	12(1115)	24(431)	24(431)	24(431)	24(431)	24(431)	24(431)	24(431)
	20	none	none	none	none	40(2465)	40(2465)	40(2465)	40(2465)	40(2465)	40(2465)	40(2465)
e10 $a = 0.972, b = 0.4575$	12	36(255)	none	12(547)	36(255)	36(255)	36(255)	36(255)	36(255)	36(255)	36(255)	36(255)
	16	none	none	none	none	16(4986)	none	none	none	none	none	none
	∞?	none	none	none	none	none	none	none	none	none	none	none
e11 $a = 0.97, b = 0.466$	8	none	none	8(1027)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)	16(2130)
	∞?	none	none	none	none	none	none	none	none	none	none	none
	∞?	none	none	none	none	none	none	none	none	none	none	none

TABLE 3. Proof of Hénon map attractor periodic points using AA condensation strategies with refined initial interval precision equal to 10^{-14}.

Interval Methods for Fixed and Periodic Points: Development and Visualization

Journal of Universal Computer Science, vol. 26, no. 10 (2020), 1312-1330
submitted: 27/7/20, accepted: 27/10/20, appeared: 28/10/20

Interval Methods for Fixed and Periodic Points: Development and Visualization

José Eduardo de Almeida Ayres
(IMPA, Rio de Janeiro, Brazil
jeaayres@impa.br)

Luiz Henrique de Figueiredo
(IMPA, Rio de Janeiro, Brazil
lhf@impa.br)

Abstract: We describe the development of rigorous numerical methods based on interval analysis for finding all fixed points of a map and all attracting periodic points of a complex polynomial. We also discuss their performance with instructive visualizations.
Key Words: fixed points, interval analysis, computer-assisted proofs
Category: G.1.5, G.1.2

1 Introduction

Finding the fixed points of a function is important in many contexts. For instance, solving nonlinear equations is frequently cast as finding fixed points. Newton's method is the main example of this formulation. Fixed points and periodic points are also important in discrete dynamical systems, especially in complex dynamics, where periodic orbits play a key role [Branner 1989; Keen 1989].

In this paper, we describe the development of rigorous numerical methods for finding all fixed points of a map. The methods are based on interval analysis [Moore 1966, 1979; Moore et al. 2009; Tucker 2011]. There is a large literature on interval methods for solving nonlinear equations [Baker Kearfott 1996], but surprisingly very little that is specific to fixed points. We know only these papers: [Caprani and Madsen 1975, 1978; Rall 1982, 1987; Rihm 2001]. Although the two problems are mathematically equivalent, there are algorithmic opportunities to be exploited for computing fixed points. Our main contribution is a hybrid algorithm that switches to plain fixed-point iteration once it establishes the existence of an attracting fixed point using Banach's criterion before finding a small certified enclosure for the fixed point. After recalling the main definitions and facts about fixed points in §2, we briefly describe interval analysis in §3. Then in §4 we describe a series of interval algorithms for finding all fixed points of a map, culminating with our hybrid algorithm and its specialization for finding all attracting periodic points of a complex polynomial described in §5. Finally, in §6 we present results illustrating the performance of these algorithms, including novel, instructive visualizations of the progress of the algorithms.

2 Fixed points

Let $f\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}^d$ be a continuous function defined on a box Ω (that is, a product of compact intervals). A *fixed point* of f is a point $x^* \in \Omega$ such that $f(x^*) = x^*$. A fixed point x^* is *attracting* if $\|f(x) - x^*\| < \|x - x^*\|$ for all points $x \neq x^*$ in a neighborhood of x^*. A fixed point x^* is *repelling* if $\|f(x) - x^*\| > \|x - x^*\|$ for all points $x \neq x^*$ in a neighborhood of x^*. For differentiable functions f, a fixed point x^* is attracting iff $\|f'(x^*)\| < 1$ and repelling iff $\|f'(x^*)\| > 1$. When $\|f'(x^*)\| = 1$, the fixed point x^* is *indifferent* or *neutral*. Here, $\|f'(x^*)\|$ is the norm of the Jacobian matrix of f at x^*.

We shall use only the classical results on fixed points, which we discuss below. A wealth of results on fixed points can be found in the books [Agarwal et al. 2001] and [Berinde 2007], among many others.

The most popular numerical method for finding a fixed point of f is *fixed-point iteration* (also known as Picard iteration):

$$x_{n+1} = f(x_n), \quad x_0 \in \Omega$$

When this sequence converges, its limit is a fixed point of f, because f is continuous. Convergence typically depends on choosing the initial point x_0 sufficiently close to a fixed point of f, which may not be easy to find. Fixed-point iteration is mostly suitable for finding attracting fixed points. Repelling fixed points cannot be found using fixed-point iteration, unless it starts at the fixed point itself. In this sense, repelling fixed points cannot be directly observed. Indifferent fixed points can sometimes be found using fixed-point iteration, but not from all nearby initial points. We shall see that interval methods can find *all* fixed points: attracting, repelling, and indifferent.

Brouwer's fixed-point theorem guarantees the existence (but not uniqueness) of fixed points when $f\colon K \to K$ is a map on a convex compact set $K \subseteq \mathbf{R}^d$, such as a box. It is the staple existence theorem, due to its fairly general hypotheses and immediate generalizations ("convex" can be replaced by "homeomorphic to a ball"). Brouwer's fixed-point theorem is stated as an existence result, but there are constructive formulations based on Sperner's lemma that can be used algorithmically [Scarf 1967; Todd 1976].

Banach's fixed-point theorem guarantees the existence and uniqueness of fixed points when $f\colon K \to K$ is a contraction map on a compact set $K \subseteq \mathbf{R}^d$. In this case, f has a unique fixed point $x^* \in K$ and fixed-point iteration converges to x^* for every initial point $x_0 \in K$. This important theorem (stated in its full generality for complete metric spaces) is crucially used in many proofs and in many numerical methods. Although apparently a global result, it is typically used locally, near a fixed point, because f need not be a global contraction. Indeed, Newton's method for finding zeros of a function is an instance of fixed-point

iteration that converges provided the initial point is sufficiently near a fixed point (that is, a zero of the function). However, the global convergence properties of Newton's method are complicated, even for finding zeros of polynomials [von Haeseler and Peitgen 1988; Hubbard et al. 2001].

We shall see presently that the hypotheses of these classical theorems can be checked rigorously in a computer using interval analysis. A key point in this verification is finding an explicit box $X \subseteq \Omega$ such that $f(X) \subseteq X$. This guarantees the existence of fixed points of f in X by Brouwer's fixed-point theorem.

3 Interval analysis

Interval analysis is the main tool for rigorous numerical computation [Tucker 2011]. It is based on interval arithmetic, an extension of ordinary arithmetic operations and standard elementary functions to intervals [Moore 1966, 1979; Moore et al. 2009]. The basic fact in interval analysis is that for each function $f\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}$ expressed by a formula or an algorithm, there is a computable function F automatically built from the expression of f, called the *natural interval extension* of f, such that $F(X)$ is an interval that estimates the whole range of values taken by f on a box $X \subseteq \Omega$:

$$F(X) \supseteq f(X) = \{f(x) : x \in X\}$$

Finding the exact range $f(X)$ is a hard problem in general [Traylor and Kreinovich 1995]. Therefore, the inclusion $F(X) \supseteq f(X)$ is usually proper and interval estimates are usually overestimates. Nevertheless, the estimates $F(X)$ get better as X shrinks to a point in the sense that $F(\{x\}) = \{f(x)\}$ for every $x \in \Omega$. More precisely, we have at least linear convergence for interval estimates: $\operatorname{diam}(F(X)) \leq c \operatorname{diam}(X)$ for some $c > 0$ that depends only on f. Thus, as we shall see in §4, interval methods are typically divide-and-conquer methods that recursively explore the domain of f, getting better information about f as they refine the subdivision, and discarding boxes that cannot contain a solution. For instance, when finding the zeros of f in Ω, we can discard a box X whenever $0 \notin F(X)$. This is a computational proof that f has no zeros in X. However, because of overestimation, we cannot conclude that f has a zero in X when $0 \in F(X)$. In this case, we subdivide X and recursively test the pieces.

We extend interval estimates to functions $f\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}^m$ by combining interval estimates for each component of f. More precisely, if $f = (f_1, \ldots, f_m)$ and F_i is an interval extension of $f_i\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}$, then, for each box $X \subseteq \Omega$, the box $F(X) = F_1(X) \times \cdots \times F_m(X) \supseteq f(X)$ estimates the range of f on X.

Automatic differentiation [Moore 1966; Rall 1986; Moore et al. 2009; Tucker 2011] is the perfect companion for interval arithmetic and works in a similar fashion. It automatically converts an expression for f into an algorithm that

simultaneously computes the value of f and of all its partial derivatives. When fed intervals instead of numbers, this algorithm computes interval estimates for the value of f and of all its partial derivatives. This allows us to reason reliably about both the range of values of f and its regions of monotonicity.

Interval arithmetic and automatic differentiation allow us to check the hypotheses of the fixed-point theorems rigorously in a computer. The existence of fixed points in a box X guaranteed by Brouwer's theorem follows whenever $F(X) \subseteq X$ because then $f(X) \subseteq F(X)$ implies $f(X) \subseteq X$. The existence of a unique fixed point in a box X guaranteed by Banach's theorem follows whenever $F(X) \subseteq X$ and $\|F'(X)\| < 1$ because these imply that f is a contraction in X, thanks to the mean value inequality. Here, F' is an interval extension of the Jacobian matrix of f, which can be computed with automatic differentiation.

4 Finding fixed points

We shall now describe a series of four incrementally refined interval algorithms for finding *all* fixed points of f in Ω. The series culminates with our hybrid algorithm, Algorithm 4.

0. The starting point is Algorithm 0, the standard divide-and-conquer interval method that performs adaptive subdivision of Ω to find all zeros of a function $g\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}^d$ [Moore 1966].

Algorithm 0

```
procedure Explore(X)
  if 0 ∉ G(X) then
    discard X
  else if diam(X) < ε then
    accept X
  else
    SubExplore(X)
  end
end

procedure SubExplore(X)
  divide X into smaller subboxes
    X_1, ..., X_m
  for j = 1, ..., m do
    Explore(X_j)
  end
end
```

Given an interval extension G for g, we recursively explore Ω, discarding every box X that cannot possibly contain zeros of g, as proved by $0 \notin G(X)$. Otherwise, X *may* contain zeros of g. If X is small enough for our purposes, we accept X as containing a zero of g. Then the midpoint of X is an approximation for a zero of g within the given tolerance ε. Otherwise, we subdivide X into smaller subboxes and explore them as above to isolate the zeros of g. In low dimension (that is, $d \leq 3$), we typically split X at its center into 2^d subboxes, leading to quadtrees and octrees [Samet 1984]. In higher dimension, to avoid creating

a huge number of subboxes, we typically split X at its center across its longest side into two subboxes, leading to a bintree [Samet and Tamminen 1985]. This strategy is also frequently used in low dimension for boxes with high aspect ratio. More sophisticated subdivision strategies exist [Baker Kearfott 1996, §4.3]. As mentioned in §3, the basis for this method is that interval estimates get better as the boxes reduce in size, converging to the actual function value when the boxes shrink to a point.

To find the fixed points of f in Ω, we use this method with $g(x) = f(x) - x$ and its interval extension $G(X) = F(X) - X$. Then the zeros of g in Ω are exactly the fixed points of f in Ω. The union of the accepted boxes contains all fixed points of f in Ω.

1. Algorithm 0 can be cast in the context of fixed points by noting that all fixed points of f in X must lie in $X \cap F(X)$. Thus, we can discard X if $X \cap F(X) = \varnothing$, because then f has no fixed points in X. Algorithm 1 uses this formulation. The test $X \cap F(X) = \varnothing$ is equivalent to the test $0 \notin G(X)$ used in Algorithm 0, but it reads better in the context of fixed points. Brouwer's theorem implies that Algorithm 1 (and its subsequent refinements) can certify the existence of fixed points in X whenever $F(X) \subseteq X$ (code omitted). In this case, X contains at least one fixed point, but may contain more: Brouwer's theorem guarantees existence but not uniqueness. Algorithm 1 subdivides large boxes hoping to *isolate* fixed points within the tolerance ε.

Algorithm 1

```
procedure Explore(X)
  if X ∩ F(X) = ∅ then        [Brouwer certifies existence if F(X) ⊆ X]
    discard X
  else if diam(X) < ε then
    accept X
  else
    SubExplore(X)
  end
end
```

2. The formulation of Algorithm 1 in terms of $X \cap F(X)$ immediately motivates Algorithm 2, which recursively explores $X' = X \cap F(X)$, instead of X. No such reduction is available for Algorithm 0, which solves generic nonlinear equations. This tiny change brings a qualitative improvement: Algorithm 1 is *spatially adaptive* because its search is guided by the *location* of the fixed points of f. Algorithm 2 is also *analytically adaptive* because its search is also guided by the *nature* of the fixed points of f. Indeed, near an attracting fixed point $x^* \in X$, we typically have $F(X) \subset X$, strictly, and so $X' = F(X) \subset X$, strictly. The stronger

the attraction of x^*, the smaller X' is, and the faster Algorithm 2 converges to x^*. Conversely, near a repelling fixed point, we typically have $F(X) \supseteq X$, and so $X' = X$ signals that we need to subdivide X.

Algorithm 2

procedure *Explore*(X)
  $X' \leftarrow X \cap F(X)$
  if $X' = \varnothing$ **then**
    discard X
  else if $\operatorname{diam}(X') < \varepsilon$ **then**
    accept X'
  else
    SubExplore(X')
  end
end

Algorithm 3

procedure *Explore*(X)
  $X' \leftarrow X \cap F(X)$
  if $X' = \varnothing$ **then**
    discard X
  else if $\operatorname{diam}(X') < \varepsilon$ **then**
    accept X'
  else if $\operatorname{diam}(X') < \lambda \operatorname{diam}(X)$ **then**
    Explore(X')
  else
    SubExplore(X')
  end
end

Algorithm 2 can be seen as an *interval iteration* in the sense of [Rall 1982, 1987]:

$$X_{n+1} = X_n \cap F(X_n), \quad X_0 = \Omega$$

When $X_n = \varnothing$ for some n, the sequence *diverges* and there are no fixed points of f in Ω. Otherwise, (X_n) is a sequence of nested nonempty boxes and so *converges* to the box $X^* = \bigcap_{n=0}^{\infty} X_n \neq \varnothing$, by Cantor's intersection theorem. Moreover, X^* contains *all* fixed points of f in Ω. In the outward-rounded floating-point arithmetic used in interval arithmetic, the sequence converges in finite time, because there are finitely many intervals with floating-point numbers as extremes. Algorithm 2 goes beyond convergence in pure interval iteration in the sense of [Rall 1982, 1987] by subdividing large boxes to isolate fixed points.

3. Recursively exploring $X' = X \cap F(X)$ instead of X in Algorithm 2 is not always a clear advantage. It may happen that X' is not much smaller than X, because we are near a weakly attracting fixed point (or near a repelling fixed point, when $X' = X$). Algorithm 3 compares the diameters of X and X' to ensure that good linear convergence is preserved in these cases, while still taking advantage of strongly attracting fixed points, when X' is much smaller than X. We use $\lambda = \frac{1}{2}$ for comparing diameters, which corresponds to the reduction in quadtree subdivision. More sophisticated tests exist [Hansen and Walster 2003, §11.7]. Algorithm 3 outputs a certified enclosure for each fixed point of f in Ω, attracting, repelling, and indifferent. This is as far as one can go using just the continuity of f.

4. When f is differentiable, we use Algorithm 4, which is our final method: it outputs a small certified enclosure for each attracting fixed point of f in Ω. Algorithm 4 incorporates Banach's criterion for attracting fixed points, as mentioned in §3. As soon as we establish the existence of a unique attracting fixed point in a box, we switch to plain fixed-point Picard iteration, because it is faster than interval iteration. Upon convergence to an approximate fixed point $\hat{x}$, we find a small box X around $\hat{x}$ such that $F(X) \subseteq X$ using interval inflation. (We used $\eta = 10^{-15}$ and $h = 0.25$ in our experiments.) Finally, we refine this box using pure interval iteration to output a certified enclosure for the fixed point x^*.

Interval inflation was proposed by [Caprani and Madsen 1978], outside the context of adaptive subdivision interval methods. They argued that enlarging a box X around an approximate fixed point often increases the chances that $F(X) \subseteq X$. In more general forms, this strategy is known as ε-inflation [Mayer 1995; Rump 1998]; it also helps to avoid clusters of boxes around a single solution [Baker Kearfott 1996, §4.2], a nuisance in subdivision methods. Clustering may affect methods like Algorithm 3 that cannot prove uniqueness of solution in a box, but not Algorithm 4.

Algorithm 4

```
procedure Explore(X)
  X' ← X ∩ F(X)
  if X' = ∅ then
    discard X
  else if diam(X') < ε then
    accept X'
  else if F(X) ⊆ X and ‖F'(X)‖ < 1 then
    ExploreAttracting(X')        [Banach]
  else if diam(X') < λ diam(X) then
    Explore(X')
  else
    SubExplore(X')
  end
end
```

```
procedure ExploreAttracting(X)
  x̂ ← mid(X)
  repeat        [Picard]
    x̂ ← f(x̂)
  until convergence
  X ← [x̂ − η, x̂ + η]
  repeat        [inflation]
    X ← X + diam(X)[−h, h]
  until F(X) ⊆ X
  repeat        [interval iteration]
    X ← F(X)
  until convergence
  accept X
end
```

5 Finding attracting periodic points

A *periodic point* of $f\colon \Omega \subseteq \mathbf{R}^d \to \mathbf{R}^d$ is a point $x^* \in \Omega$ such that $f^n(x^*) = x^*$, where $f^n = f \circ \cdots \circ f$ (n times) is the n-th iterate of f. The *period* of a periodic point x^* is the smallest positive integer n such that $f^n(x^*) = x^*$. Periodic points of period n are thus fixed points of f^n. Conversely, the fixed points of f^n are the periodic points of f with period dividing n. Clearly, the periodic points of

period 1 are exactly the fixed points of f. The *orbit* of a point $x_0 \in \Omega$ is the set of the images of x_0 under the iterates of f, that is, $\{x_0, f(x_0), f^2(x_0), \dots\}$. Periodic points are exactly those that have finite orbits; the size of the orbit is the period. A *periodic orbit* is the orbit of a periodic point.

Periodic orbits are important in discrete dynamical systems, because they represent stationary states. Attracting periodic orbits are especially important because they represent the fate of nearby points. Periodic orbits play a key role in the dynamics of complex rational maps [Branner 1989; Keen 1989]. Typically, the basins of attraction of attracting periodic orbits divide the Riemann sphere into regions sharing a common boundary, the Julia set. The Julia set is the closure of the repelling periodic points. When f is a polynomial, the set of points having bounded orbits is called the filled Julia set; its boundary is the Julia set. In particular, all periodic points are in the filled Julia set.

We shall adapt Algorithm 4 to find all attracting periodic points of a complex polynomial $f\colon \mathbf{C} \to \mathbf{C}$. The periodic points of period n are found among the fixed points of f^n, that is, the roots of $f^n(z) = z$. This is a polynomial equation and so can in principle be solved numerically using one of several standard methods. However, if f has degree d, then the equation $f^n(z) = z$ has degree d^n and we do not want to find that many zeros only to choose the few that are attracting periodic points. Moreover, most standard methods need polynomial equations expressed in monomial form, and we would rather not expand $f^n(z)$ into monomial form.

Algorithm 5 is an adaptation of Algorithm 4 to find the fixed points of f^n, trying to discard its repelling fixed points and to detect its attracting fixed points, as soon as possible. Since the repelling fixed points are on the Julia set, we can discard boxes that are away from the Julia set, that is, outside a disk centered at the origin and containing the Julia set. Such a disk is called an *escape disk* because all orbits starting outside it escape to infinity. The radius of an escape disk can be explicitly computed from the coefficients of $f(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0$ [McClure 2019, §4.1]:

$$R = \max\left(2|a_n|, 2\frac{|a_{n-1}| + \cdots + |a_0|}{|a_n|}\right)$$

This formula generalizes the well-known formula $R = \max(2, |c|)$ for $f(z) = z^2 + c$.

Algorithm 5 computes interval estimates for $f^n(X)$ and $(f^n)'(X)$ iteratively, using the chain rule. As soon as an intermediate estimate $F^k(X)$ is outside the escape disk, we have proved that all orbits starting at X will escape to infinity. In this case, X cannot contain any attracting periodic points of f, because those are in the filled Julia set. This test is not strictly needed, but it helps to reduce the density of the quadtree. Without it, the large overestimation in the interval estimates for $f^n(X)$ due to the wrapping effect [Moore 1979] leads to needless

subdivisions. We can definitely discard a box X when $|(f^n)'(X)| \geq 1$, because then any periodic points of period n in X will be repelling or indifferent.

Algorithm 5

procedure *Explore*(X)
  $W, W' \leftarrow X, 1$
  for $k = 1$ to n **do**
    $W, W' \leftarrow F(W), F'(W)W'$
    if W is outside the escape disk **then**
      discard X
    end
  end
  $X' \leftarrow X \cap W$
  if $X' = \varnothing$ or $\|W'\| \geq 1$ **then**
    discard X
  else if $\operatorname{diam}(X') < \varepsilon$ **then**
    accept X'
  else if $W \subseteq X$ and $\|W'\| < 1$ **then**
    ExploreAttracting(X')
  else if $\operatorname{diam}(X') < \lambda \operatorname{diam}(X)$ **then**
    Explore(X')
  else
    SubExplore(X')
  end
end

6 Numerical experiments

We now present the results of some numerical experiments that illustrate the performance of the algorithms discussed above. We find periodic points of $f(z) = z^2 + c$ for several $c \in \mathbf{C}$ and several periods (see Table 1). The fourth column gives $\mu = |(f^n)'(x^*)|$, which measures the nature of the periodic point x^*: *super attracting* when $\mu = 0$, *strongly attracting* when $\mu < 1$ and is close to 0, and *weakly attracting* when $\mu < 1$ but is close to 1. The domain is $\Omega = [-2.01, 1.99] \times [-1.99, 2.01] \subseteq \mathbf{R}^2 \cong \mathbf{C}$. The tolerance used for termination is $\varepsilon = 10^{-12}$.

Pictures of the subdivision behavior of the algorithms can only show the early steps, due to limited spatial resolution. The interesting behavior happens in very small boxes. Therefore, we focus on how the algorithms find a single periodic point. The graphs shown below describe the main steps taken by an algorithm until it accepts a box containing a certified enclosure for a periodic point. The horizontal axis shows the sequence of steps. The vertical axis shows the decimal

Figure	period	x^*	μ	c
2a	3	0	0	$-0.122561166876654 + 0.744861766619744\,i$
2b	1	$-0.15 - 0.11\,i$	0.38	$-0.16 - 0.15\,i$
2c	2	0.17	0.80	-1.2
3a	8	-1.10	0	-1.3815474844320614695
3b	4	-1.29	0.18	1.3
3c	8	-1.35	0.91	-1.393
4a	5	-0.79	0	$-0.504340175446244 + 0.562765761452981\,i$
4b	1	$-0.15 - 0.11\,i$	0.38	$-0.16 - 0.15\,i$
4c	8	-0.19	0.91	-1.393
5a	8	-1.10	0	-1.3815474844320614695
5b	3	$-0.07 + 0.011\,i$	0.35	$-0.10 + 0.72\,i$
5c	2	-1.17	0.80	-1.2
6	1	$-0.15 - 0.11\,i$	0.38	$-0.16 - 0.15\,i$
7	3	$-0.0099 + 0.0052\,i$	0.058	$-0.12 + 0.74\,i$
8	4	0.38	0.18	-1.3

Table 1: Data for our numerical experiments.

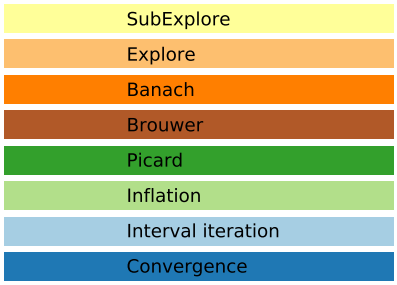
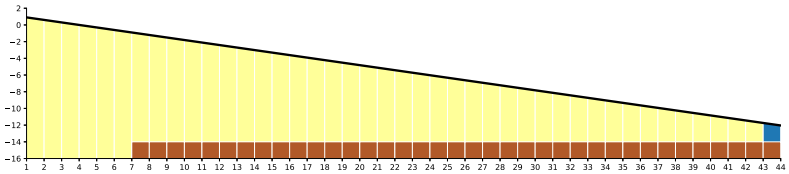


Figure 1: The meaning of the color bars in the graphs.

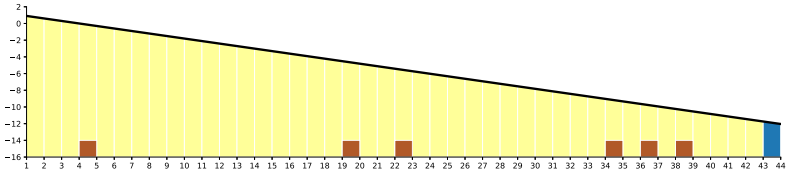
logarithm of the diameter of the current enclosure for the point; it indicates the number of correct decimal places in the current approximation. The bars are colored according to the legend in Figure 1. The certification of existence of fixed points via Brouwer's theorem is marked with a small brown box, when possible.

6.1 Individual performance

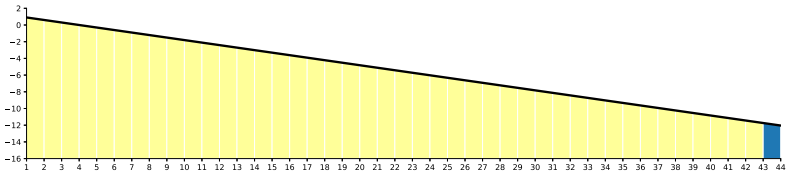
1. Figure 2a shows the steps taken by Algorithm 1 for finding a super attracting periodic point of period 3 for $c \approx -0.12 + 0.74\,i$. Note the constant decrease in diameter, due solely to subdivision; Algorithm 1 cannot exploit the nature of this point. Nevertheless, Algorithm 1 proves early on that there is a fixed point in a certain box using Brouwer's theorem. This certification persists until convergence. Figure 2b shows the steps taken by Algorithm 1 for finding a strongly attracting fixed point for $c = -0.16 - 0.15\,i$. Again, Algorithm 1 cannot exploit the nature of this point. Moreover, certification via Brouwer's theorem only happens for a



(a) super attracting point



(b) strongly attracting point

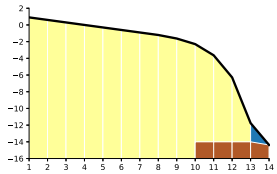


(c) weakly attracting point

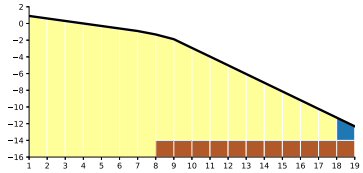
Figure 2: Performance of Algorithm 1.

few boxes. Figure 2c shows the steps taken by Algorithm 1 for finding a weakly attracting fixed point for $c = -1.2$. As a consequence, Algorithm 1 cannot certify any boxes via Brouwer's theorem. Figure 2 is typical of the behavior of Algorithm 1 for finding an attracting point. It subdivides steadily at a fixed rate but it cannot exploit the nature of the point to go any faster.

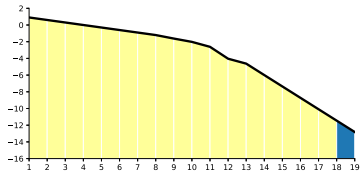
2. Figure 3a shows the steps taken by Algorithm 2 for finding a super attracting periodic point of period 8 for $c \approx -1.38$. Note the sharp decrease in diameter after step 10, when it also started certifying boxes using Brouwer's theorem. This reflects the nature of this point, because X' is much smaller than X. Figure 3b shows the steps taken by Algorithm 2 for finding a strongly attracting periodic point of period 4 for $c = -0.16 - 0.15\,i$. The nature of this point is reflected in the decrease in diameter after step 8, when it also started certifying boxes using Brouwer's theorem. The change of slope reflects the magnitude of μ. Figure 3c shows the steps taken by Algorithm 2 for finding a weakly attracting periodic point of period 8 for $c = -1.393$. As a consequence, Algorithm 2 cannot certify any boxes via Brouwer's theorem, but there is still a decrease in diameter after step 13. Again, the change of slope reflects the magnitude of μ. Figure 3 is typical



(a) super attracting point



(b) strongly attracting point



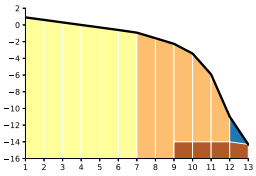
(c) weakly attracting point

Figure 3: Performance of Algorithm 2.

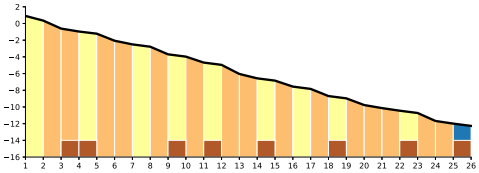
of the behavior of Algorithm 2 for finding an attracting point. It subdivides steadily at a rate that changes near the point to reflect the magnitude of μ.

3. Figure 4a shows the steps taken by Algorithm 3 for finding a super attracting periodic point of period 5 for $c \approx -0.50 + 0.56\,i$. Note that the switch to exploring X' after step 7 and the sharp decrease in diameter after step 9, when it also started certifying boxes using Brouwer's theorem. This reflects the nature of this point. Figure 4b shows the steps taken by Algorithm 3 for finding a strongly attracting fixed point for $c = -0.16 - 0.15\,i$. The nature of this point is reflected in the alternation of exploring and subdividing X', because μ is close to 0.5. Figure 4c shows the steps taken by Algorithm 3 for finding a weakly attracting periodic point of period 8 for $c = -1.393$. As a consequence, it needs to subdivide X' more often. Figure 4 is typical of the behavior of Algorithm 3 for finding an attracting point. It alternates between exploring and subdividing X'. The number of consecutive explore steps reflects the magnitude of μ.

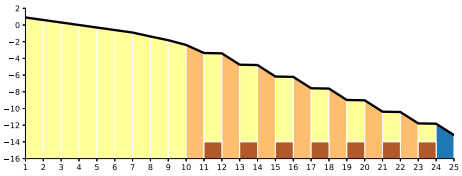
4. Figure 5a shows the steps taken by Algorithm 4 for finding a super attracting periodic point of period 8 for $c \approx -1.38$. Note that it certified existence and



(a) super attracting point



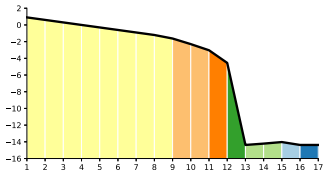
(b) strongly attracting point



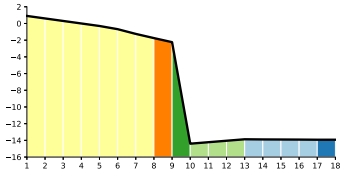
(c) weakly attracting point

Figure 4: Performance of Algorithm 3.

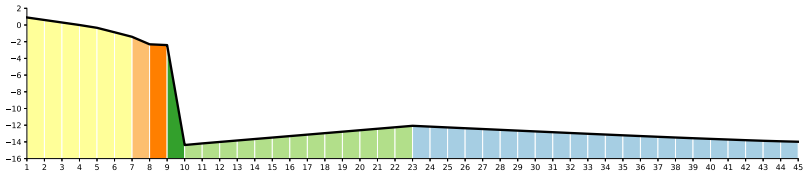
uniqueness using Banach's theorem at step 11 and so switched to Picard iteration at step 12. Inflation and refinement took just a couple of steps. This reflects the nature of this point. Figure 5b shows the steps taken by Algorithm 4 for finding a strongly attracting periodic point of period 3 for $c = -0.10 + 0.72\,i$. It switched to Picard iteration at step 9. Inflation and refinement took a few steps. This reflects the nature of this point. Figure 5c shows the steps taken by Algorithm 4 for finding a weakly attracting periodic point of period 2 for $c = -1.2$. As a consequence, inflation took several steps. Refinement did not succeed, but the output box was close to the tolerance. This reflects the nature of this point and the magnitude of $\mu = 0.8$. Figure 5 is typical of the behavior of Algorithm 4 for finding an attracting point. It subdivides until Banach's certification holds and then quickly finds a certified small box around the point using inflation. The number of steps in inflation and refinement depends on the nature of the point. In the hybrid approach of Algorithm 4, plain fixed-point iteration replaces many subdivisions steps in previous algorithms.



(a) super attracting point



(b) strongly attracting point



(c) weakly attracting point

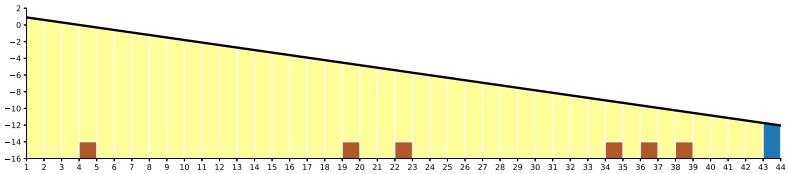
Figure 5: Performance of Algorithm 4.

6.2 Comparative performance

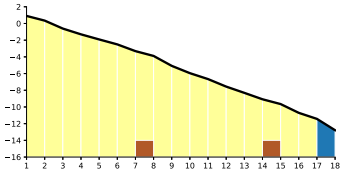
Figure 6 compares the performances of Algorithm 1 and Algorithm 2 for finding a strongly attracting fixed point for $c = -0.16 - 0.15\,i$. The main difference is the rate of diameter decrease, shown by the slope of the graph. As expected, near a strongly attracting point, X' is much smaller than X.

Figure 7 compares the performances of Algorithm 2 and Algorithm 3 for finding a strongly attracting periodic point of period 3 for $c = -0.12 + 0.74\,i$. Although Algorithm 3 takes slightly longer to find this point, it is overall faster. Algorithm 2 visits 1553 boxes to maximum depth 15 whereas Algorithm 3 visits 1389 boxes to maximum depth 18.

Figure 8 compares the performances of Algorithm 3 and Algorithm 4 for finding a strongly attracting periodic point of period 4 for $c = -1.3$. Algorithm 3 steadily approximates the point, and certifies existence using Brouwer's theorem from step 7 onward. Algorithm 4 fares much better. It certifies both existence and uniqueness using Banach's theorem at the same step as Algorithm 3 certifies existence only. From then on, it quickly finds a small certified enclosure for the point.

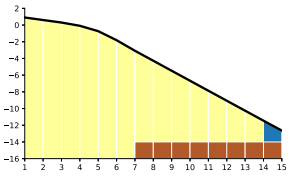


(a) Algorithm 1

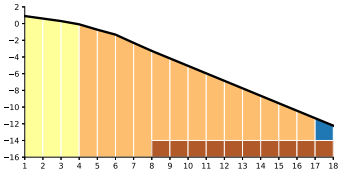


(b) Algorithm 2

Figure 6: Performance of Algorithm 1 vs. Algorithm 2.



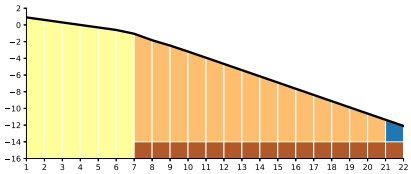
(a) Algorithm 2



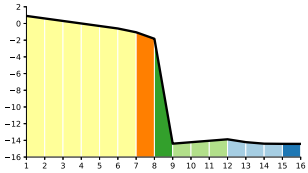
(b) Algorithm 3

Figure 7: Performance of Algorithm 2 vs. Algorithm 3.

Figure 9 compares the domain subdivisions performed by Algorithm 1 and Algorithm 2 for finding all fixed points for $c = -0.16 - 0.15\,i$. The maximum depth is 5, low on purpose for illustration. There is a strongly attracting fixed point on the left of the domain and a repelling fixed point on the right. This is reflected in the decomposition of Algorithm 2, which approaches the fixed points much faster than Algorithm 1 does, even at that low depth. The nature of the repelling fixed point is captured by the cluster of accepted boxes around it; such a cluster appears at all depths.



(a) Algorithm 3



(b) Algorithm 4

Figure 8: Performance of Algorithm 3 vs. Algorithm 4.

Figure 10 compares the domain subdivisions performed by Algorithm 4 and Algorithm 5 for finding all periodic points of period 6 for $c = -0.60 - 0.66\,i$. The maximum depth is 17. The subdivision performed by Algorithm 4 is everywhere dense, because of the repelling points on the Julia set, resulting in a large number of boxes accepted at that depth (1073 boxes corresponding to repelling points). The subdivision performed by Algorithm 5 is dense only near the Julia set, as expected, because it avoids boxes outside the escape disk. Both algorithms certified existence and uniqueness using Banach's theorem of all six attracting periodic points, shown in light blue. However, Algorithm 4 explored 234189 boxes while Algorithm 5 explored 20085 boxes, a gain of an order of magnitude (also reflected in the total execution times, which includes graphics output).

6.3 Execution times

We wrote prototype implementations of our algorithms in the scripting language Lua. Our programs were meant to support the qualitative analysis given above and were instrumented for that purpose. The programs were not meant to support quantitative analysis of execution time. Nevertheless, as a rough indication, Table 2 gives the execution times for the computations illustrated in the figures. Note that the experiments do not always solve the same instance or even the same problem: they were chosen to display different convergence behaviors. In particular, Algorithm 4 finds only attracting fixed points, whereas the previous algorithms find all fixed points. Algorithm 5 is specialized to complex polynomials and exploits the nature of complex dynamics.

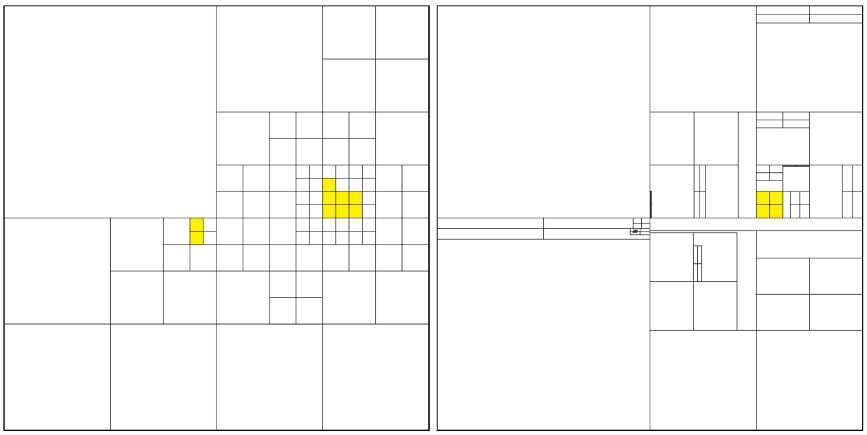


Figure 9: Decompositions of Algorithm 1 (left) and Algorithm 2 (right).

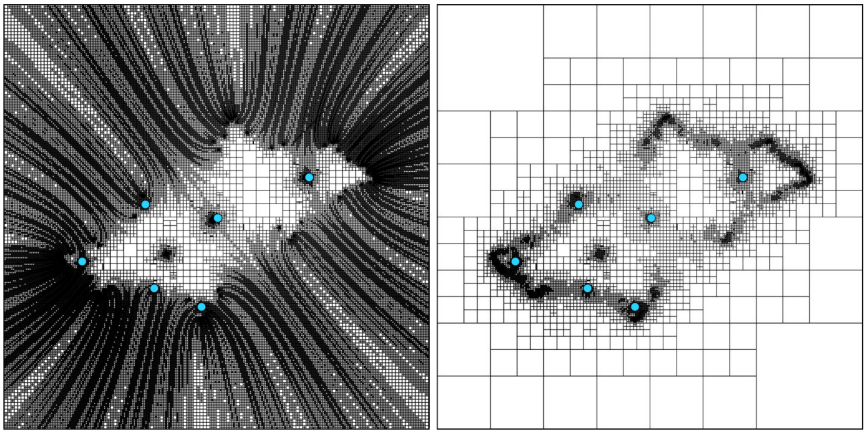


Figure 10: Decompositions of Algorithm 4 (left) and Algorithm 5 (right).

Figure	a	b	c
2	0.15567	0.01762	0.02860
3	19.46442	0.23579	18.64248
4	2.20107	2.20107	19.02098
5	19.42642	0.14639	0.01707
6	0.01762	0.01049	
7	0.13420	0.11114	
8	0.21073	0.23416	

Table 2: Execution times in seconds for our numerical experiments.

7 Conclusion

We proposed a hybrid interval algorithm that finds all attracting fixed points of a map. The algorithm is a novel combination of classical mathematical results and standard interval techniques. We also specialized this algorithm to find all attracting periodic points of a complex polynomial exploiting well-known facts of complex dynamics.

While the correctness of interval methods is assured, their efficiency depend on the quality of the interval estimates. Natural interval extensions provide first-order interval estimates. High-order estimates exist, but they are naturally more expensive to compute [Cornelius and Lohner 1984]. A natural direction for further work is to study the effect of high-order estimates on the overall efficiency of the algorithms presented here. Of special interest are second-order estimates, such as mean-value forms [Caprani and Madsen 1980] and affine arithmetic [de Figueiredo and Stolfi 2004], which may provide a good cost-benefit balance. Affine arithmetic will also probably help mitigate the wrapping effect.

Acknowledgements

The first author was partially supported by CNPq and FAPERJ doctoral scholarships. The second author is partially supported by a CNPq research grant. This research was done in the Visgraf Computer Graphics laboratory at IMPA. Visgraf is supported by the funding agencies FINEP, CNPq, and FAPERJ, and also by gifts from IBM Brasil, Microsoft, NVIDIA, and other companies.

References

[Agarwal et al. 2001] Agarwal, R. P., Meehan, M., and O'Regan, D. (2001). *Fixed point theory and applications*. Cambridge University Press.

[Baker Kearfott 1996] Baker Kearfott, R. (1996). *Rigorous global search: Continuous problems*. Kluwer.

[Berinde 2007] Berinde, V. (2007). *Iterative approximation of fixed points*, volume 1912 of *Lecture Notes in Mathematics*. Springer.

[Branner 1989] Branner, B. (1989). The Mandelbrot set. In [Devaney and Keen 1989], pages 75–105.

[Caprani and Madsen 1975] Caprani, O. and Madsen, K. (1975). Contraction mappings in interval analysis. *BIT*, 15(4):362–366.

[Caprani and Madsen 1978] Caprani, O. and Madsen, K. (1978). Iterative methods for interval inclusion of fixed points. *BIT*, 18(1):42–51.

[Caprani and Madsen 1980] Caprani, O. and Madsen, K. (1980). Mean value forms in interval analysis. *Computing*, 25(2):147–154.

[Cornelius and Lohner 1984] Cornelius, H. and Lohner, R. (1984). Computing the range of values of real functions with accuracy higher than second order. *Computing*, 33(3-4):331–347.

[de Figueiredo and Stolfi 2004] de Figueiredo, L. H. and Stolfi, J. (2004). Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1):147–158.

[Devaney and Keen 1989] Devaney, R. L. and Keen, L., editors (1989). *Chaos and fractals: The mathematics behind the computer graphics*. Proc. Symposia in Applied Mathematics 39. AMS.

[Hansen and Walster 2003] Hansen, E. and Walster, G. W. (2003). *Global optimization using interval analysis*. CRC Press.

[Hubbard et al. 2001] Hubbard, J., Schleicher, D., and Sutherland, S. (2001). How to find all roots of complex polynomials by Newton's method. *Inventiones Mathematicae*, 146(1):1–33.

[Keen 1989] Keen, L. (1989). Julia sets. In [Devaney and Keen 1989], pages 57–74.

[Mayer 1995] Mayer, G. (1995). Epsilon-inflation in verification algorithms. *Journal of Computational and Applied Mathematics*, 60(1-2):147–169.

[McClure 2019] McClure, M. (2019). *Basic complex dynamics: A computational approach*.

[Moore 1966] Moore, R. E. (1966). *Interval analysis*. Prentice-Hall.

[Moore 1979] Moore, R. E. (1979). *Methods and applications of interval analysis*. SIAM.

[Moore et al. 2009] Moore, R. E., Baker Kearfott, R., and Cloud, M. J. (2009). *Introduction to interval analysis*. SIAM.

[Rall 1982] Rall, L. B. (1982). A theory of interval iteration. *Proceedings of the American Mathematical Society*, 86(4):625–631.

[Rall 1986] Rall, L. B. (1986). The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282.

[Rall 1987] Rall, L. B. (1987). Interval methods for fixed-point problems. *Numerical Functional Analysis and Optimization*, 9(1-2):35–59.

[Rihm 2001] Rihm, R. (2001). Acceleration of iteration methods for interval fixed point problems. *Linear Algebra and its Applications*, 324(1-3):189–207.

[Rump 1998] Rump, S. M. (1998). A note on epsilon-inflation. *Reliable Computing*, 4(4):371–375.

[Samet 1984] Samet, H. (1984). The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260.

[Samet and Tamminen 1985] Samet, H. and Tamminen, M. (1985). Bintrees, CSG trees, and time. *Computer Graphics*, 19(3):121–130. (Proceedings of SIGGRAPH '85.).

[Scarf 1967] Scarf, H. (1967). The approximation of fixed points of a continuous mapping. *SIAM Journal on Applied Mathematics*, 15(5):1328–1343.

[Todd 1976] Todd, M. J. (1976). *The computation of fixed points and applications*, volume 124 of *Lecture Notes in Economics and Mathematical Systems*. Springer.

[Traylor and Kreinovich 1995] Traylor, B. and Kreinovich, V. (1995). A bright side of NP-hardness of interval computations: interval heuristics applied to NP-problems. *Reliable Computing*, 1(3):343–359.

[Tucker 2011] Tucker, W. (2011). *Validated numerics: A short introduction to rigorous computations*. Princeton University Press.

[von Haeseler and Peitgen 1988] von Haeseler, F. and Peitgen, H.-O. (1988). Newton's method and complex dynamical systems. *Acta Applicandae Mathematicae*, 13(1-2):3–58.