User Manual “Simulator”

User manual for using the "Simulator" program. The mathematical principles are described in the document "Dynamic systems".

Version 7.0 – 2025/03/01 - Dr. Hermann Biner

Contents

1. Introduction

The "Simulator" enables the simulation of simple dynamic systems. The mathematical principles for this and the concepts for implementation are described in the document "Dynamic systems". This document also contains suggestions for further developments or mathematical exercises.

The code of the program is published in GitHub and is available as open source. It is written in VB.NET and extensively commented. The development environment is the community version of Microsoft Visual Studio 2022, which is available free of charge and easy to install. At least version 17.9 is required, which is based on Microsoft Framework 8.0, which must also be downloaded and installed if it is not already available.

The current version of the "Simulator" is 7.0.0 published on 2025/03/01.

The GitHub link is as follows:

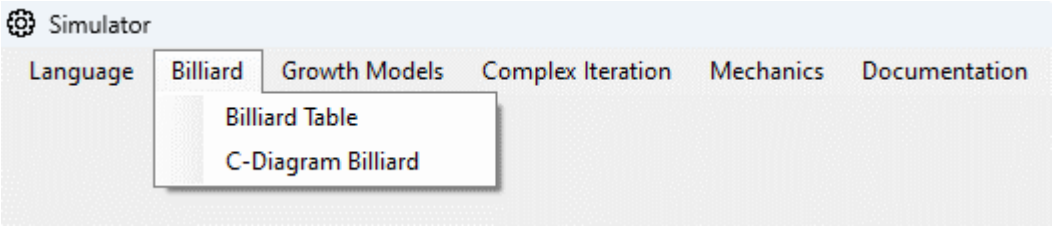
https://github.com/HermannBiner/Simulator

This manual explains how to use the "Simulator" and provides examples of its use.

2. Main Menu

The language can be selected under "Language":

Various forms of mathematical billiards can be examined in the "Billiards" menu.

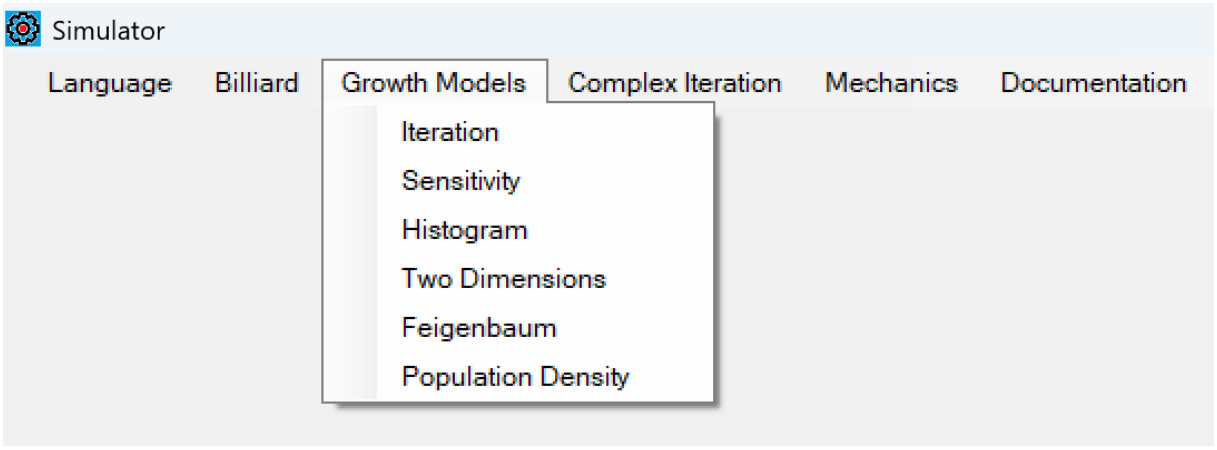


In particular, the "Billiard table" submenu offers the following options:

- Elliptical billiards
- Billiards in the stadium
- Oval billiards

The shape of the billiard table is defined everywhere by a parameter C. In the case of elliptical billiards, C is the axis ratio of the ellipse. Analogous to the Feigenbaum diagram in the case of quadratic functions, a C-diagram can be created here, which shows the behaviour of the iteration parameters as a function of C.

The "Growth models" section enables various experiments based on the iteration of real functions.

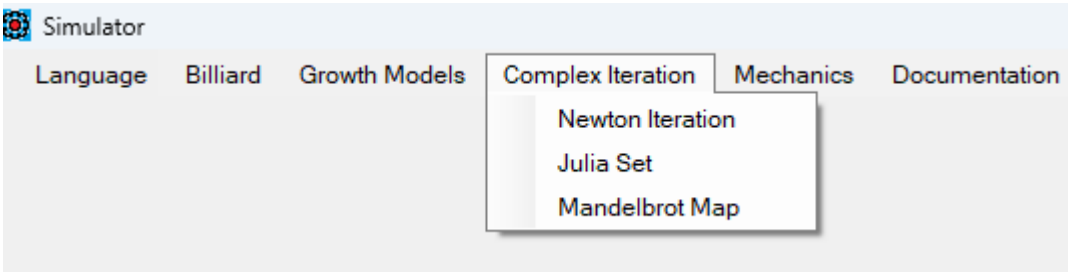


The following functions are implemented:

- Tent map
- Logistical growth
- Iteration of the parabola
- Mandelbrot iteration for real number (only for Feigenbaum, i.e. the investigation of the bifurcation in the Mandelbrot set for real c)

The submenus offer various functions, including the investigation of the chaotic case. They are described later in this document.

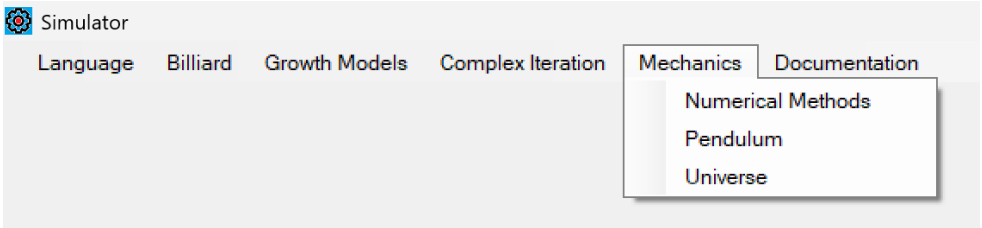
In the "Complex iteration" area, iterations in the complex plane are offered for investigation.



The following are available:

- Newton iteration (especially in the case of complex roots of unity)
- Generation of Julia sets and the corresponding Mandelbrot set for the quadratic function and the power function of degree n
- Mandelbrot set as a map for the associated July set

The "Mechanics" area offers various pendulum systems that can be examined.



Furthermore, various numerical methods for solving ordinary differential equations can be investigated using the example of the spring pendulum:

- Euler explicit

- Euler implicitly
- Midpoint rule
- Runge Kutta

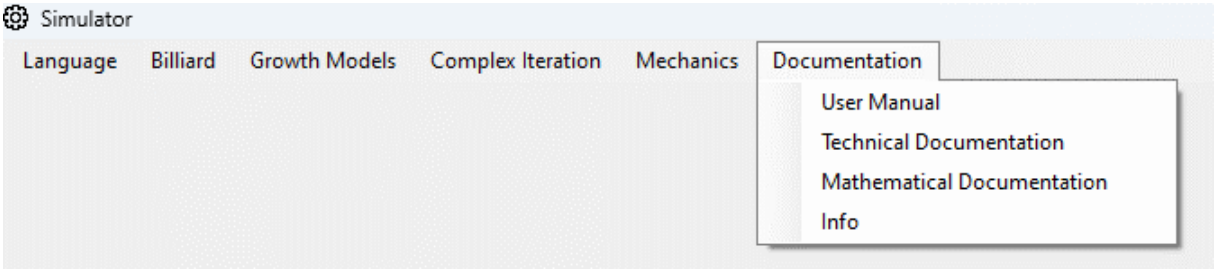
Various versions of coupled pendulums are available for examination in the "Pendulum" menu item:

- Double pendulum
- Oscillating spring pendulum
- Horizontal vibrating pendulum

The menu «universe» supports the simulation of

- The Newtonian universe
- Alternative universes

The "Documentation" menu item provides all the information required to use the program.



This includes a user manual, technical documentation and mathematical documentation. General information on the program is also available.

All documents are available in German or English, depending on the language selected.

3. Billiards Menu

3.1. Billiards

The "Simulator" makes it possible to examine different forms of billiards. The following types of billiards are implemented:

- Elliptical billiards
- Billiards in the stadium
- Oval billiards

In all cases, the shape of the billiard table is defined by a parameter $C > 0$.

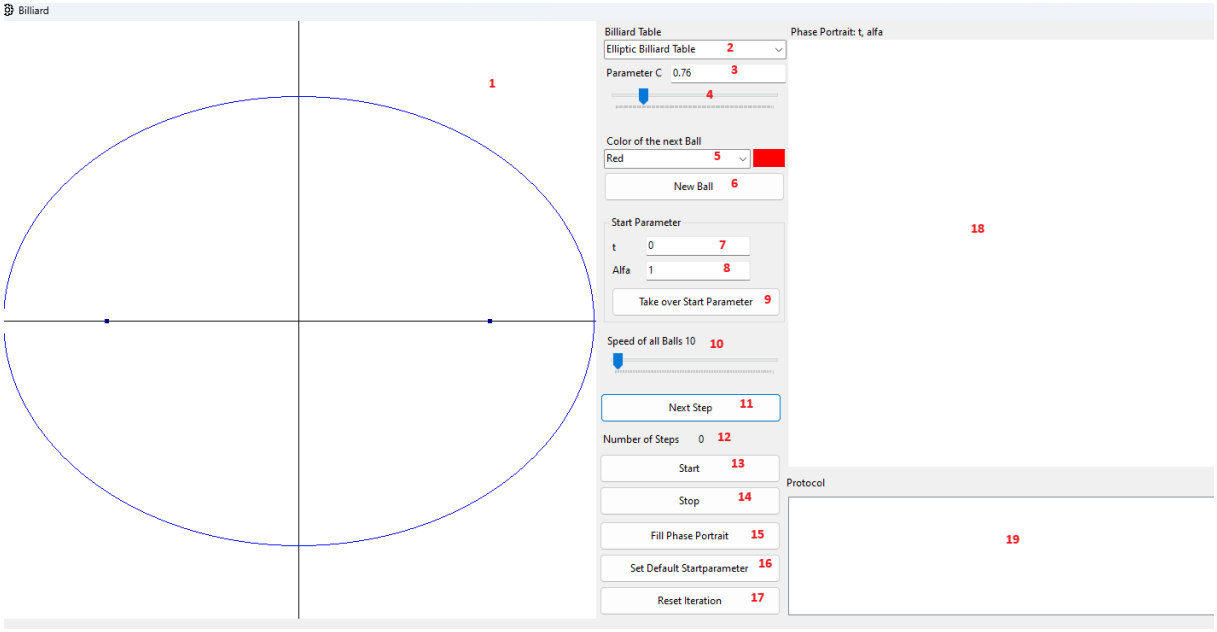
- C is the ratio of the minor axis to the major axis in elliptical billiards
- C is the ratio of circle diameter to rectangle width for billiards in the stadium
- C is the ratio of the circle radius (right part) to the main axis of the ellipse (left part)

As the code is public, the user can program other forms of billiards. All he has to do is implement an interface. See technical documentation or comments in the code.

It is assumed that the billiard ball moves without friction and that no energy is lost when it hits the edge of the billiard table. Any number of billiard balls can be placed on the table. However, they do not hit each other. They are used to visualize different orbits depending on the starting parameters.

The first starting parameter is always a point on the edge of the billiard table, which is described by a parameter *t* according to the mathematical documentation. The second parameter is the angle of reflection *α* (in radians) between the ball path and the tangent at the point of impact of the billiard table.

The procedure for examining all types of billiards is the same. The following window opens in the "Mechanics - Billiards" menu:



Window for examining the different billiards

1

This is the drawing area for the billiard table and the ball tracks.

2

The type of billiard table is selected here. See list at the beginning.

3

Here the shape of the billiard table can be defined by a parameter C according to the previous explanation. This ratio must be a positive number. For the value C = 1, a circular billiard table is obtained in the case of elliptical or oval billiards.

4

This scrollbar can be used to set the parameter C and thus the shape of the billiard table. The billiard table is then drawn directly.

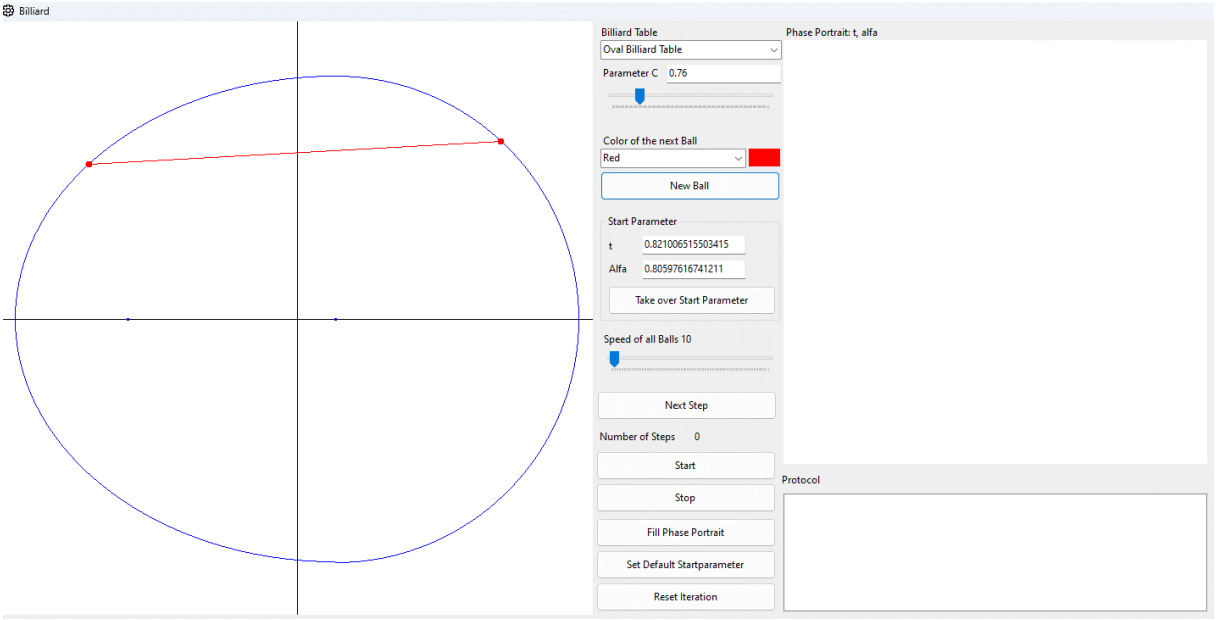
5

The ball is programmed in such a way that any number of instances (i.e. balls) can be generated. To differentiate between them, one of five colours can be specified for the next generated ball.

6

This button generates a new ball. This is placed by default, depending on the type of billiard table. The ball is then placed at its starting point by holding down the left mouse button. If the mouse

button is released, the starting point t is set. If the mouse button is pressed a second time, the direction of the first section of the lane can now be set. If the mouse is released again, the first path angle α is also set.



Oval billiard table with set ball including first lane section

If the ball and its first track section are placed manually in this way, the parameters t (position of the starting point on the edge) and α (first angle between the track section and the tangent to the billiard table) are displayed on the right in the "Start parameters" area. In certain cases, it is also useful to enter these parameters manually and then accept them as start parameters.

7, 8

The start parameters are displayed here t, α are displayed here. These values can also be set manually. This is necessary if you want to achieve certain trajectories using an approximation method. Once the start parameters have been defined manually, they can be transferred to the ball by pressing the

9

button. The ball is then placed accordingly. As the parameters t are calculated modulo 2π or modulo the circumference U of the billiard table and α modulo π, all real starting parameters are permitted. This means that $t \in [0, 2\pi[\ or\ t \in [0, U[$ and $\alpha \in]0, \pi[$.

10

Here you can control the speed of all balls for the next shots simultaneously.

11

Here, a single hit is performed by all the balls involved.

12

The number of steps is displayed here.

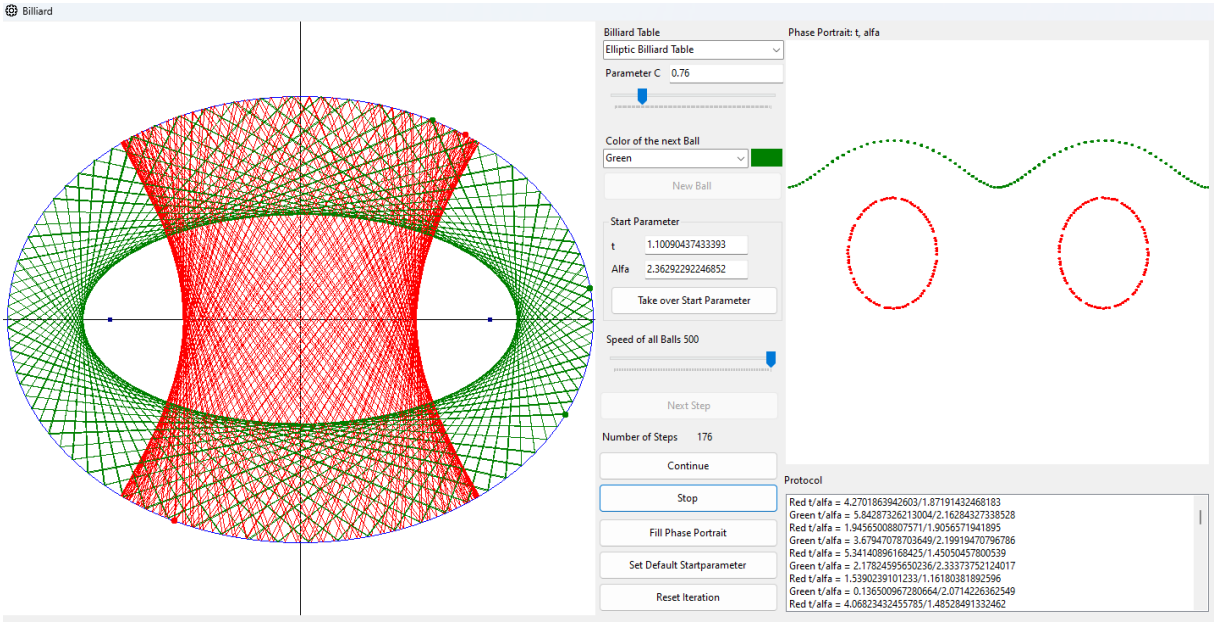
13

Here, all balls perform hits until the process is stopped by the stop button

14

is stopped.

Example



Elliptical billiards with two balls

In the example above, two balls have been placed. The red orbit runs between the focal points of the ellipse and the green orbit runs outside it. The mathematical documentation derives the caustics that occur.

15

Here, many billiard balls are created and distributed across the table so that the phase portrait can then be created by pressing "Start".

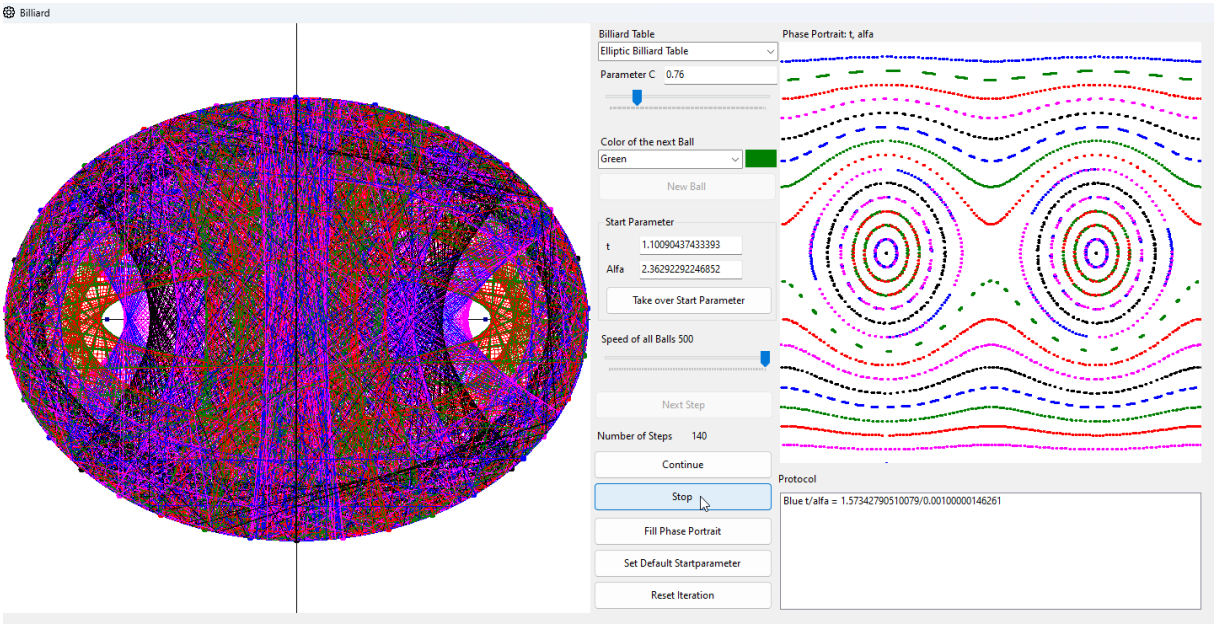


Image of the ball movements in phase space

16

This button resets the entire iteration and sets the start parameters t and Alfa to the default settings.

17

This button only resets the iteration, but the start parameters set by the user are retained.

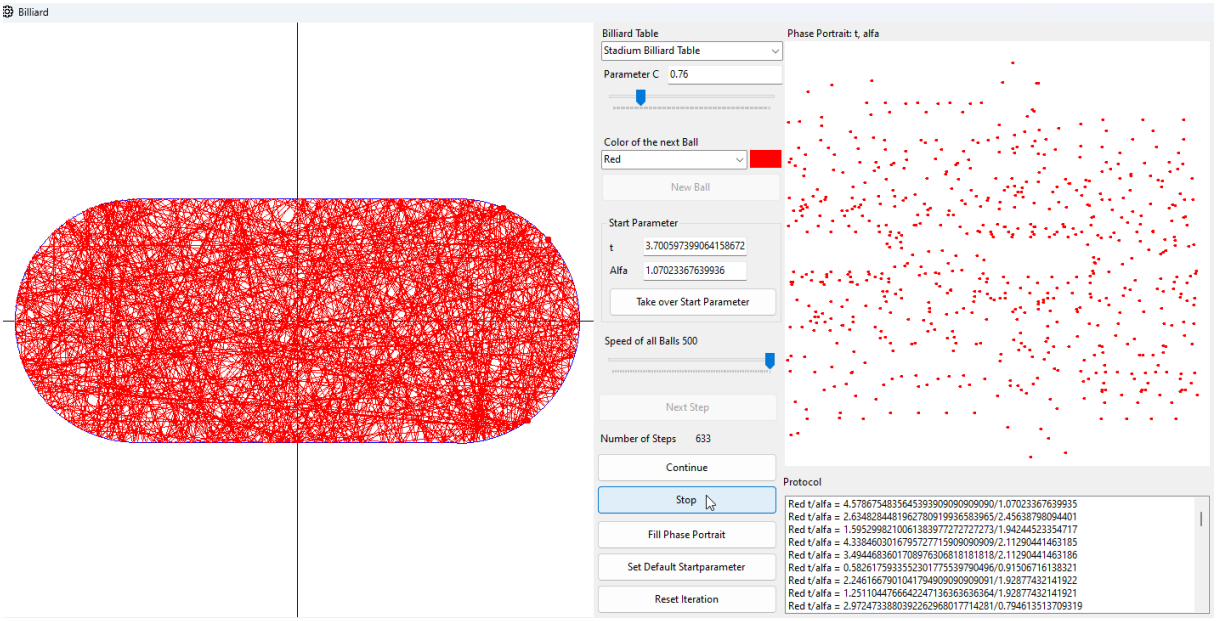
18

The "phase portrait" of the movement is sketched here: The parameter *t* is entered in the horizontal direction and the parameter α in the vertical direction. In the example above, you can see that in the green case *t* runs through all possible values, while α is restricted to a certain range. In the red case, both *t* and α are restricted.

19

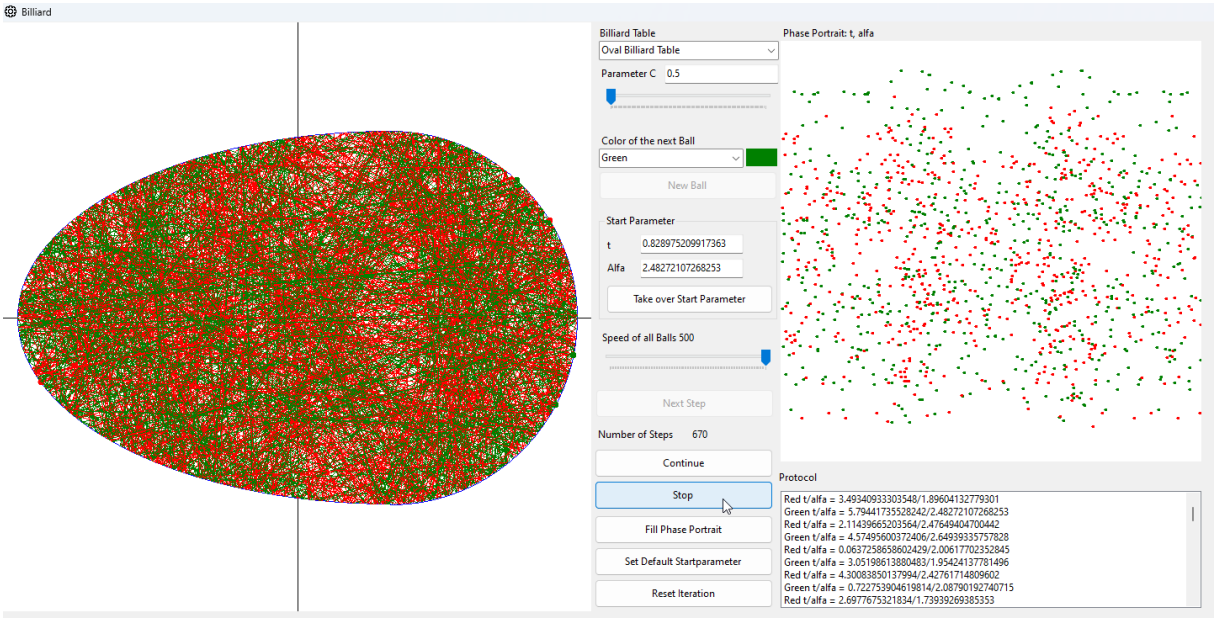
The parameter pairs are listed here (t_n, α_n) are listed here, differentiated according to the ball color.

Examples



A ball in the stadium-shaped billiards

No structure appears in the phase portrait on the right. The parameter pairs appear to be randomly scattered.

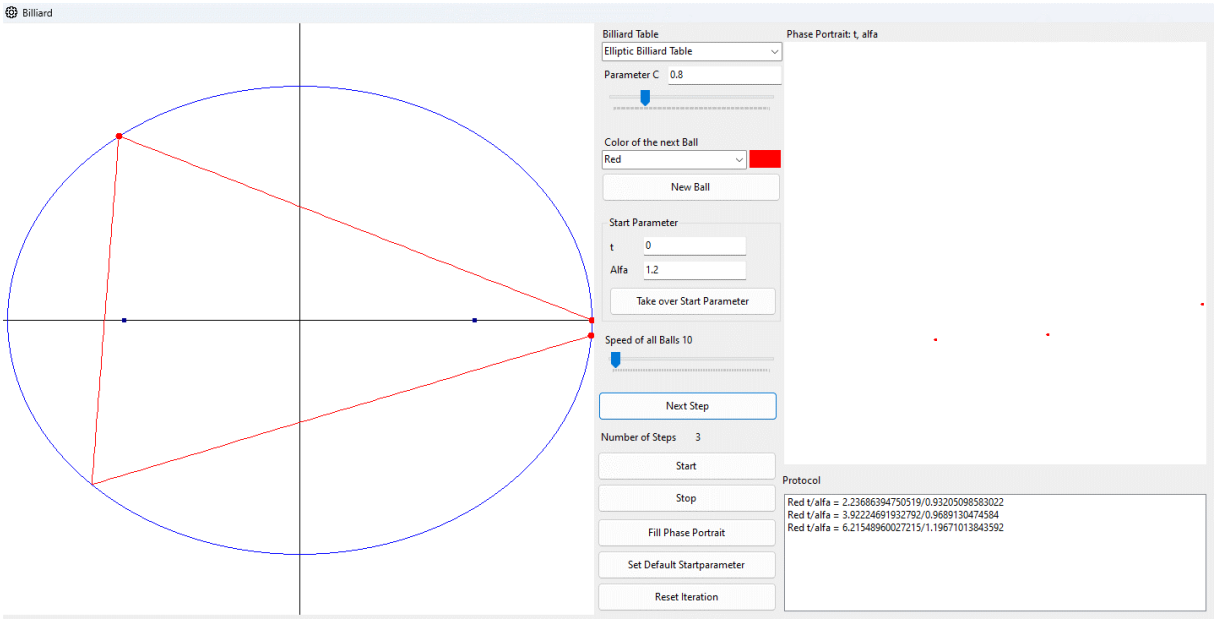


Two balls in an oval billiard

Here too, no structure appears in the phase portrait.

Another example shows the possibility of determining cycles approximately by interval nesting.

Suppose we are looking for a 3-period cycle in the case of elliptical billiards for the parameter $C = 0.8$. As the starting point for the cycle, we select the parameter $t = 0$, i.e. the starting point $(a, 0)$. Then, with a little experimentation, we see that the start angle $\alpha = 1.2$ provides a first very rough approximation. We enter these values manually as start parameters, accept the start parameters (this places the ball) and carry out the first three impacts:



Search for the 3-cycle: The value $\alpha = 1.2$ is slightly too small

You can still see this visually here. However, it is better to check the log on the right and you can see that $t = 6.215489$ … is not quite 2π.

As you can also see $\alpha_2 = 1.25$ is slightly too large. We enter these values manually in the parameter area and let the ball adopt the parameter values. We check the parameter *t* after every three impacts. If $t > 0$ is, then α was slightly too large. If *t* is just below 2π, then α was slightly too small. Depending on this, we correct this by halving the interval. This results in the following interval nesting for the required starting value α:

α_n	α_{n+1}
1.2	1.25
1.2	1.225
1.2125	1.225
1.21875	1.225
1.221875	1.225
1.221875	1.2234375
1.22265625	1.2234375

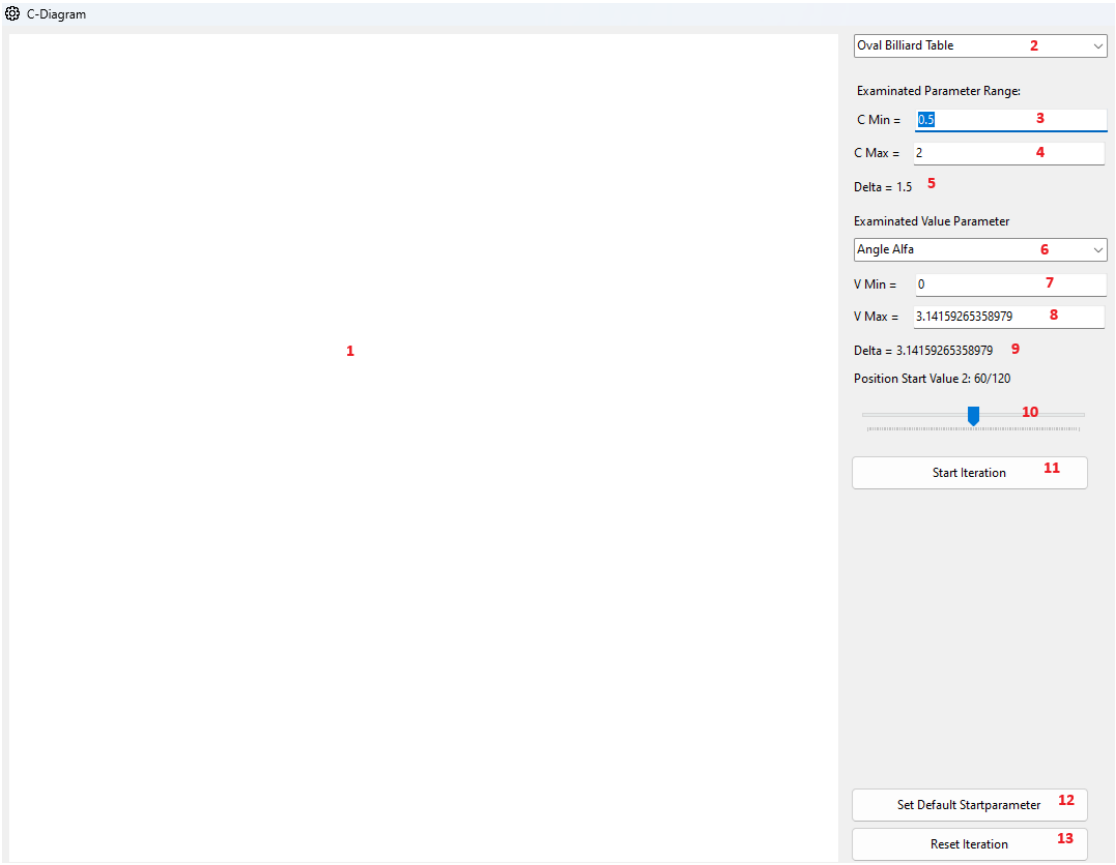
The required 3-cycle starts in the range $t = 0, \alpha \in [1.2226, 1.2235]$

The method converges very slowly. Based on continuity arguments, however, one can at least conclude the existence of a 3-cycle. See mathematical documentation.

3.2. C-diagram

In the case of quadratic functions, the Feigenbaum diagram shows the behaviour of the iteration as a function of the parameter a. In billiards, this role is played by the parameter C.

The following window opens to display the C diagram:



Window for examining the C diagram

1

The graphic is displayed here.

2

The shape of the billiard table is selected here.

3, 4, 5

Here you can specify which interval of C is examined. The following applies: $C \in [Cmin, Cmax]$. The width of the analyzed interval is displayed in 5.

6

The movement of the ball is described by two value parameters. The value parameter whose dependency on C is to be displayed, is selected here.

7, 8, 9

For the selected value parameter, the interval to be examined in more detail is specified here. The value parameter is then in the interval $[Vmin, Vmax]$. The width of this interval is displayed in 9.

10

The first value parameter for billiards describes the position of the impact point on the edge of the billiard table. The second value parameter describes the angle of reflection during the shot.

The value parameter that is *not* selected in 6, i.e. that is not examined more closely, is set to 1/3 of its value interval by default. For example, $t = {}^{2\pi}/_{3}$ in the case of the elliptical billiard.

For the value parameter to be examined, the starting position can vary between 1/12 and 11/12 of its value interval. For example, $\alpha = {}^{7\pi}/_{12}$ at the setting that is visible at 10 in the window above.

11

The iteration is started or stopped here.

12

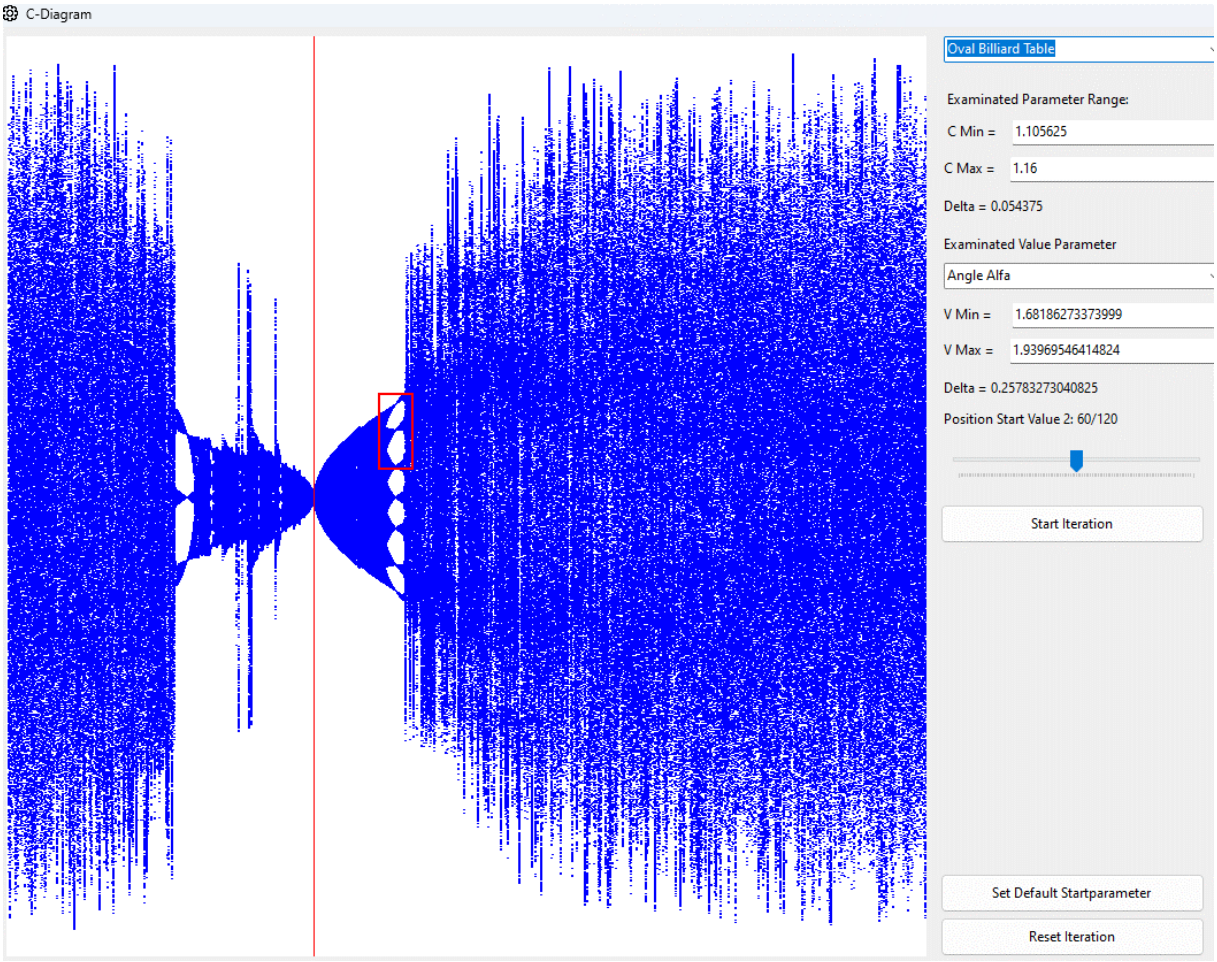
The iteration is reset here, and the parameter intervals are set to the default.

13

Only the iteration is reset here.

Selection of small sections in the diagram

We start the C-diagram for the oval billiard with the default parameters.



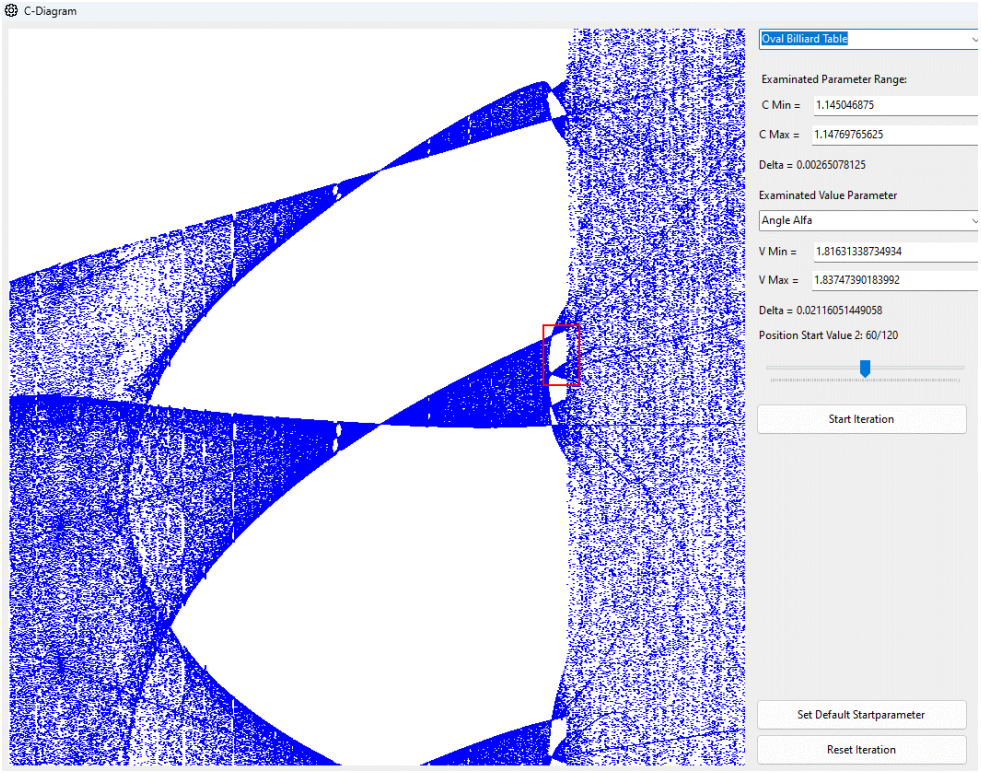
Examination of α in oval billiards with user selection - see red rectangle

Above, the oval billiard was examined for the parameter range $C \in [0.5, 2]$, namely the reflection angle α. The red vertical line marks the parameter value $C = 1$. For this value, the billiard table is a circle and α is constant, i.e. there is a point in the diagram.

The user now has the option of using a selection rectangle to select a section that they want to examine more closely. This is done by holding down the left mouse button. The parameter range in fields 3-5 on the right and the value range in fields 6-8 are automatically moved accordingly.

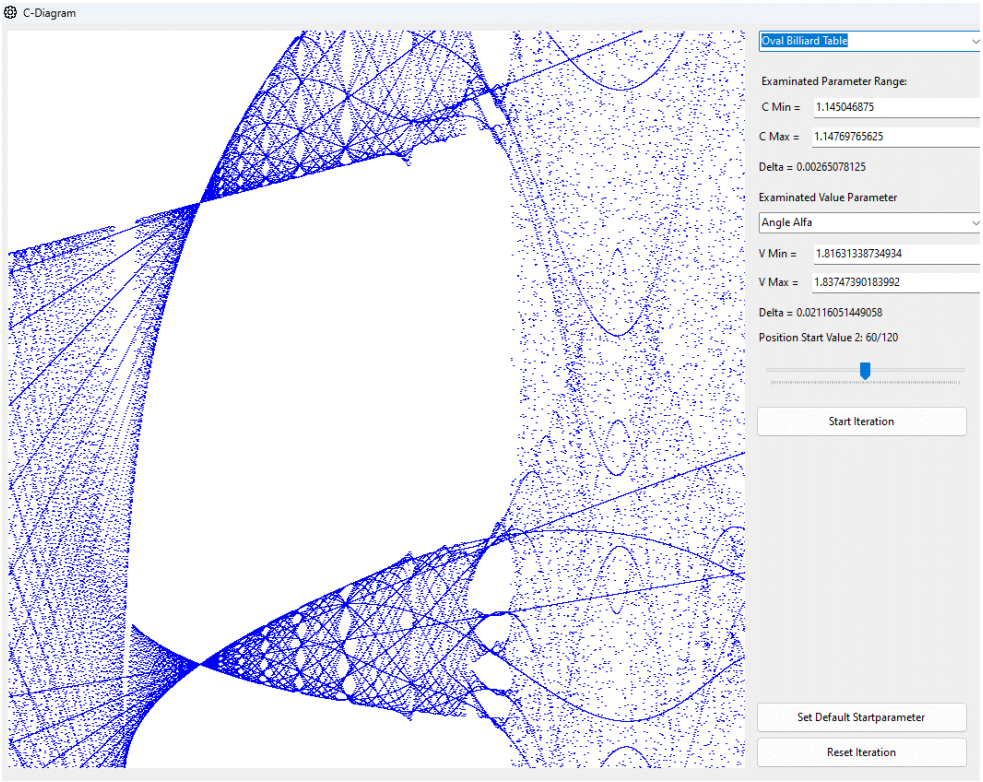
In the image above, the user has selected the section marked in red by holding down the left mouse button. The intervals $[Cmin, Cmax]$ and $[Vmin, Vmax]$ are traced on the right, as in the fig tree diagram.

If the iteration is now started again, the following picture is obtained:



Excerpt from the C-diagram for the oval billiard

Here, the user has selected another section for enlargement. If you start the iteration with this section, you get:



Another excerpt from the diagram

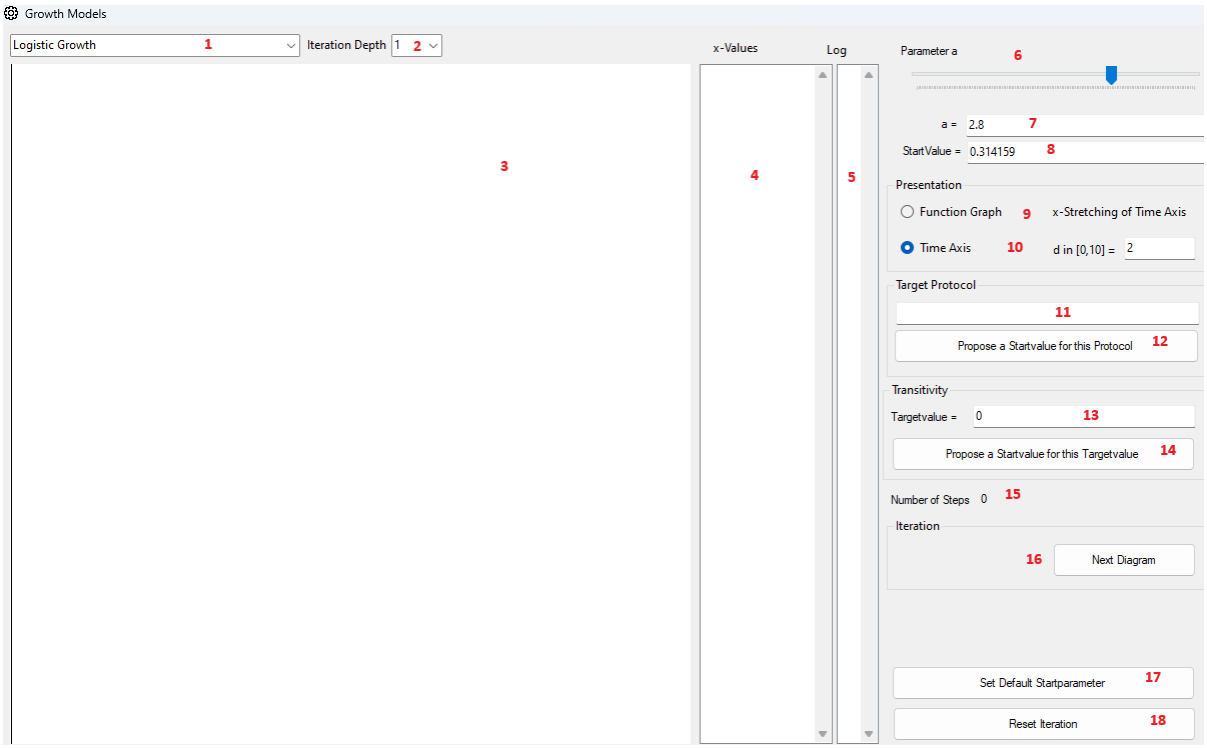
Remark for further experiments

In most cases, the examination of the reflection angle α is more meaningful than the parameter *t*, which determines the impact point. In the case of a circle or an orbit outside the focal points of the ellipse, the orbit of *t* is close to the permissible interval, unless a periodic orbit has just been found. In contrast, α is constant for the circle or oscillates in a subinterval of [0, π [for the ellipse. Before starting an investigation, you can also experiment with the position of the starting value to obtain interesting images. However, their in-depth mathematical interpretation is likely to be difficult.

4. Growth Models Menu

4.1. Iterations in the real interval

The following window is opened via the menu item "Growth Models - Iteration":



Window for examining iterations in the "Simulator"

1

The type of iteration is selected at the top left. The following are available:

- Tent map
- Logistical growth
- Iteration of the parabola

The iterations are defined as follows:

Tent map: $f\colon [0,1] \longrightarrow [0,1];\ f(x) = \begin{cases} ax, x \in [0,0.5[\\ a(1-x), x \in [0.5,1] \end{cases}, a \in]0,2]$

Logistical growth: $f\colon [0,1] \longrightarrow [0,1], x \longmapsto ax(1-x), a \in]0,4]$

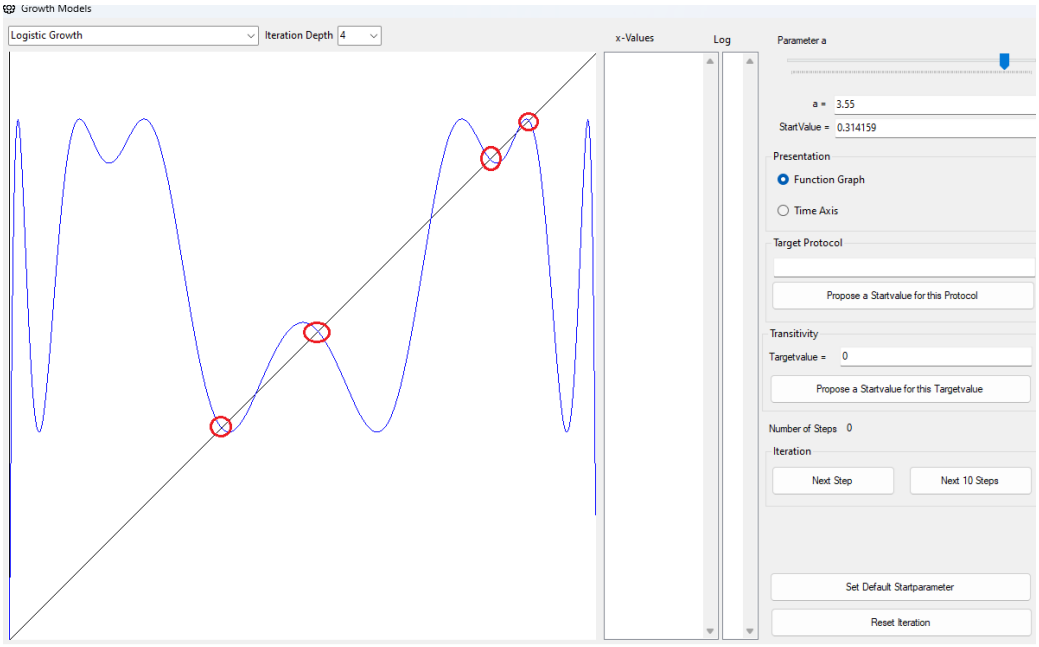
Iteration of the parabola: $f\colon [-1,1] \longrightarrow [-1,1], x \longmapsto 1 - ax^2, a \in]0,2]$

The user can program other types of iterations. All he has to do is implement an interface. See technical documentation or comments in the code.

2

The iteration depth is defined here. It determines how often the iteration function f is repeated per iteration step. e.g. for iteration depth = 1: $x_{n+1} = f(x_n)$. For an iteration depth = 5, the following applies: $x_{n+1} = f^5(x_n)$. This is not necessarily important for the iteration itself, but it is when higher order cycles are examined using the function graph.

For example, the logistic mapping for the parameter value $a = 3.55$ has an attractive 4-cycle.



Attractive 4-cycle

To make this visible, select "Logistic growth" as the iteration. As the parameter value $a = 3.55$ and an iteration depth of 4, i.e. we examine the quadruple iterated function f^4. If you draw its graph, you can see that it intersects the 45° straight line at four points where the multiplier (the slope of the tangent) is less than 1 and that this cycle is therefore attractive. You can also see four other intersections with the 45° straight line, which belong to a repulsive cycle, because the tangent gradient is greater than 1 here.

A detailed description can be found in the mathematical documentation.

3

This is the area for all graphical representations.

4

The sequence members (x_n) which are generated by the iteration are listed here.

5

The log is generated in this column for each iteration value x_n in the left-hand list. In the case of tent mapping and logistic growth, the log is $p(x_n)$ is defined as:

$$p(x_n) = \begin{cases} 0, x_n \in [0{,}0.5\,[\\ 1, x_n \in [0.5{,}1] \end{cases}$$

In the case of the iteration of the parabola, it is defined as:

$$p(x_n) = \begin{cases} 0, if\ x_n \in [-1{,}0] \\ 1, if\ x_n \in]0{,}1] \end{cases}$$

6

The parameter value a for the iteration function can be defined using a scrollbar. If the parameter value is changed, the graph of the iteration function is automatically drawn when the "Function graph" option is activa.

7

The parameter value a is displayed here or can be entered manually.

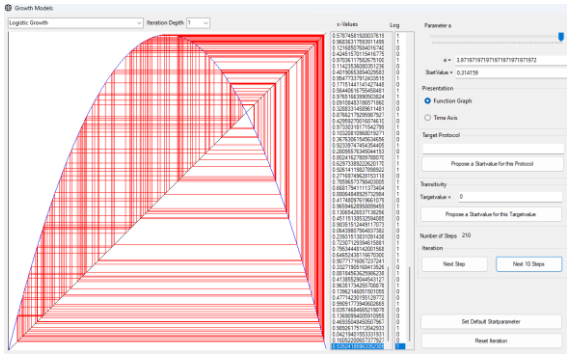
- Tent map: a must be in the range]0,2].
- Logistic growth: a must be in the range]0.4].
- Parabola: a must be in the range]0,2].

8

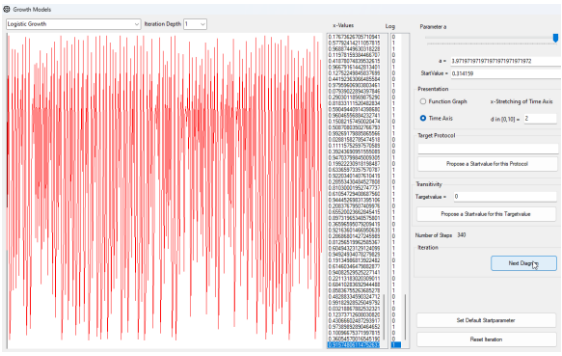
The start value is defined in this field. It can be specified manually or suggested by the program. For logistic growth and tent map, it must lie in the interval]0,1]. For the parabola, it lies in the interval [-1,1].

9, 10

There are two display options for examining the iteration:



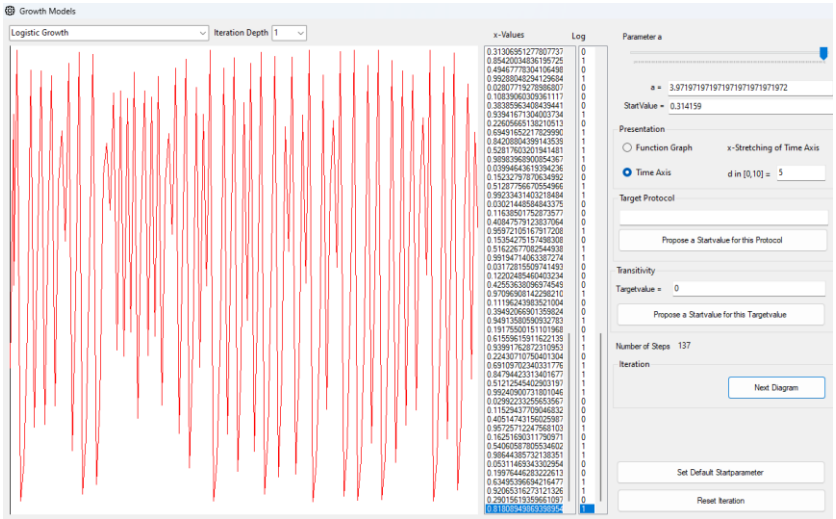
Representation in the function graph



The same iteration on the "time axis"

When displaying the function graph, you oscillate back and forth between the 45° straight line and the function graph. The "time axis" shows the number of iteration steps horizontally and the respective iteration value vertically.

If the "Time axis" representation is selected, it is possible that the iteration is no longer recognizable on the time axis because the jumps are too narrow. In this case, the time axis, which is displayed on the x-axis in the coordinate system, can be stretched. A value between 1 and 10 is possible.



Same iteration as above, but with a stretch of 5 instead of 2

11

In the chaotic case, any protocol consisting of "0" and "1" can be specified here. Due to the computing accuracy of the computer, protocols of length up to at least 50 are possible, depending on the type of iteration. The chaotic case exists for the following parameter values, among others:

- Tent map a = 2
- Logistic growth a = 4
- Iteration of the parabola a = 2

If the behaviour is not chaotic, a predefined protocol cannot be generated.

Example

We choose logistic growth with parameter a = 4, i.e. the chaotic case.

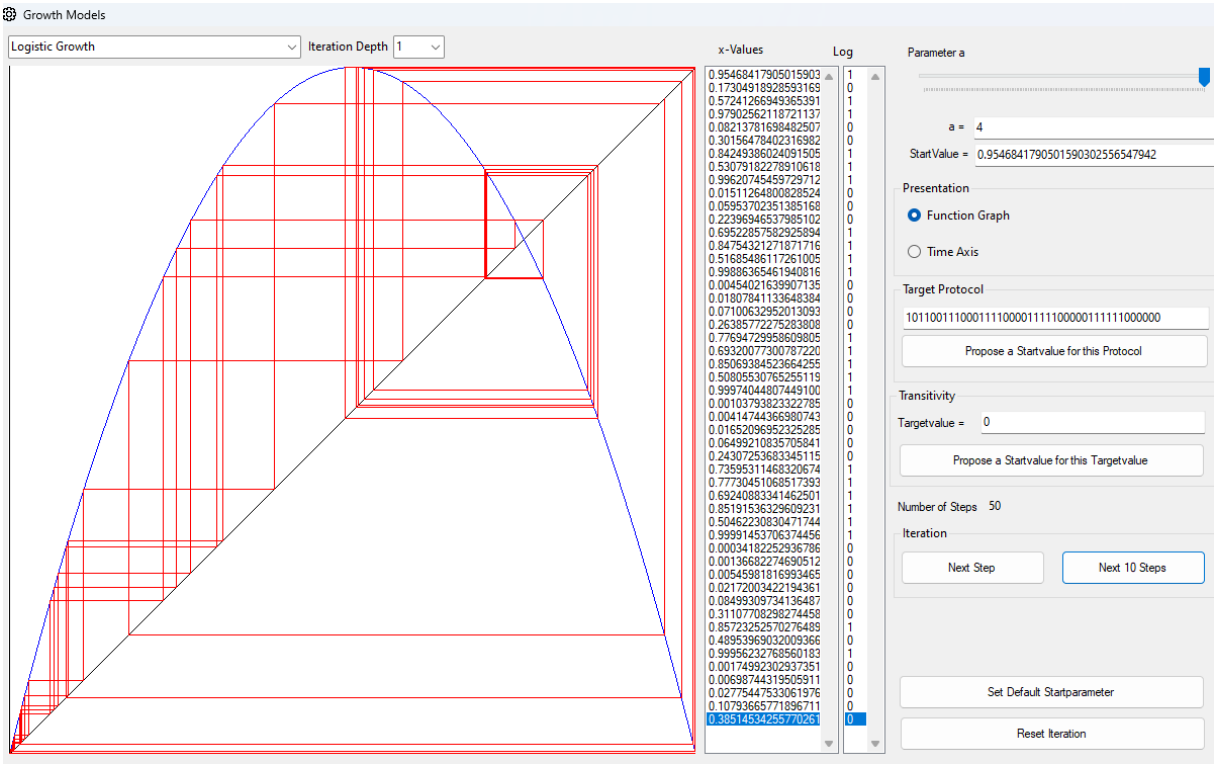
Now we enter a 42-digit 0-1 sequence as the target protocol:

101100111000111100001111100000111111000000

12

We then press the button: "Propose a start value for this protocol". The "Simulator" then suggests a start value: $x_1 = 0.954684179050159030255654794 2$.

Now press the "Next 10 steps" button a few times. In the log for the protocol, we can see that the specified protocol has actually been generated.



Generation of a predefined protocol

This means that this iteration is chaotic in the sense of a coin toss: For any given protocol, there is a start value that returns this protocol as the result.

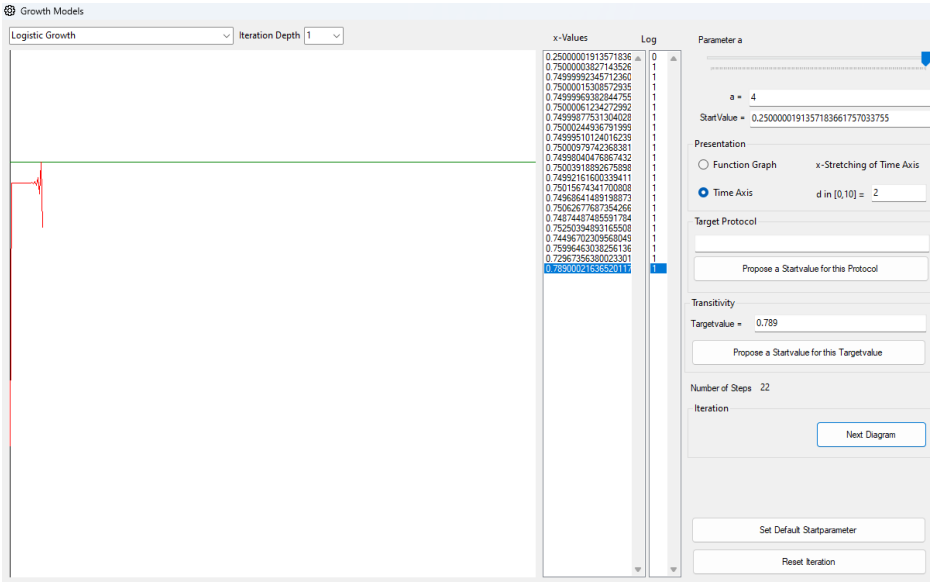
13

If the iteration is chaotic, the transitivity can also be examined. To do this, a start value is specified in field 8 and a target value in field 13. The theory then says that there is a slightly different start value in the chaotic case, so that the iteration comes arbitrarily close to the target value.

For example, we select 0.25 as the start value and enter 0.789 as the target value in field 13. Then press the

14

This calculates the slightly modified start value 0.250000019135718366175703375 5. We then select "Time axis" as the display and press the "Next diagram" button. The iteration then runs a few steps and stops when it gets close to the target value. This is the case here with the iteration value 0.789000216... - see below. The target value is market by the green horizontal line.



After a few steps you will be close to the target value.

15

The number of iteration steps is displayed here.

16

The "Next step" button is only visible when displayed as a function graph. It executes the next iteration step. In the "Function graph" display, the "Next 10 steps" button executes the next 10 iteration steps. In the "Time axis" display, the button is called "Next diagram" and as many iteration steps are executed until the entire width of the diagram is filled. If the button is pressed again, the iteration is continued, and a new diagram is created.

17

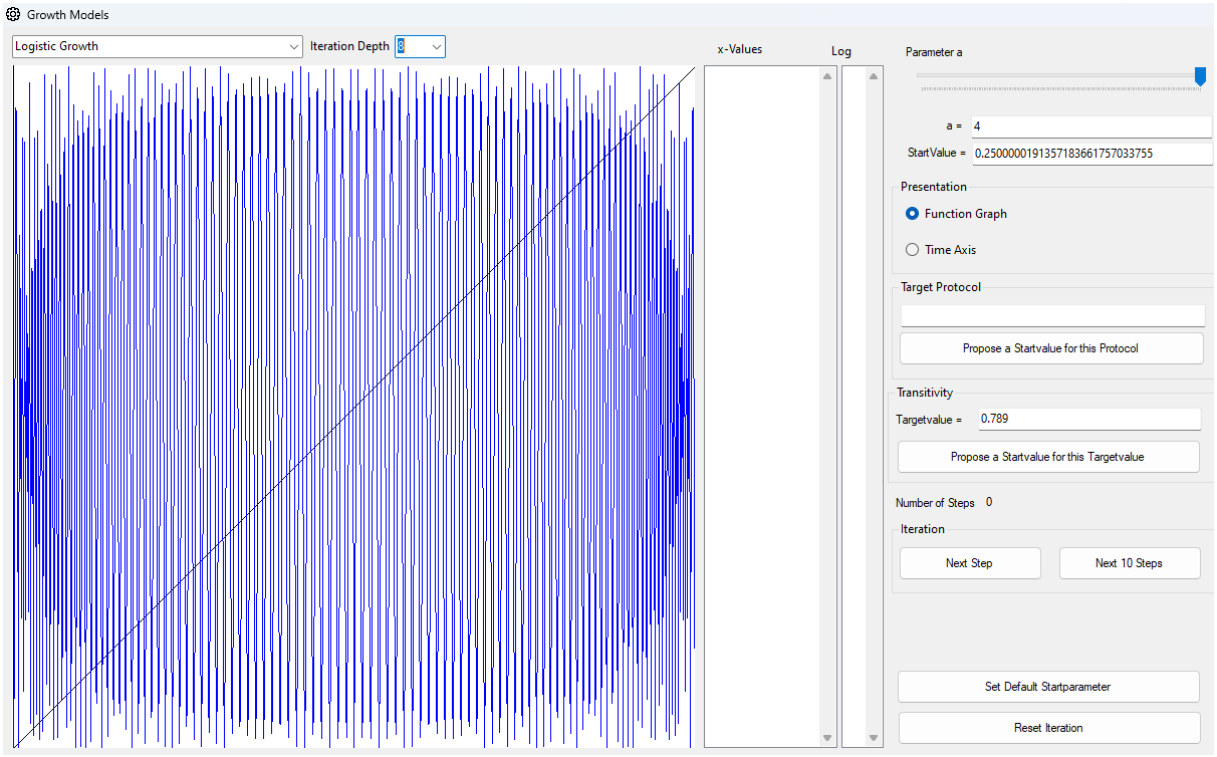
The iteration is reset here, and the start parameters are set to the default.

18

Only the iteration is reset here.

The mathematical documentation describes the chaotic case for $a = 4$. A dynamic system is chaotic in Devaney's sense if it has the properties of transitivity and sensitivity and if the set of repulsive cycles is dense in the iteration interval. We have shown transitivity here.

For the tightness of the repulsive cycles, you can look at the graph of the iterated functions. Cycles are intersections of their graphs with the 45° straight line. These are all repulsive and are closer together the higher the iteration period of the cycle is.

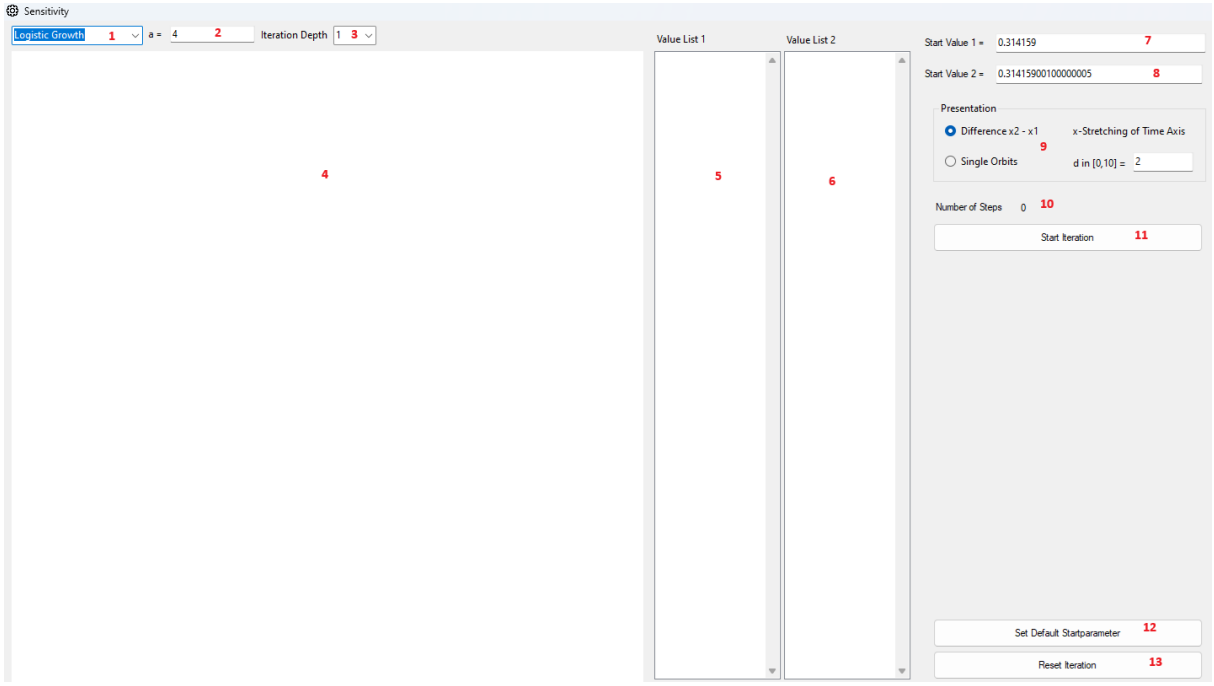


Graph of iterates f^8

The sensitivity can be shown in the following section.

4.2. Investigation of Sensitivity

The following window opens under the menu item "Growth Models - Sensitivity":



Window for investigating sensitivity with the "simulator"

1

The type of iteration is selected at the top left. The following are available:

- Tent map
- Logistical growth
- Parabola

2

The parameter value for each iteration can be entered here.

- Tent map: a must be in the range]0,2].
- Logistic growth: a must be in the range]0.4].
- Parabola: a must be in the range]0,2].

3

The iteration depth is defined here. It determines how often the iteration function *f* is repeated per iteration step.

4

This is the area for graphical representations.

5

The sequence members of the iteration are listed here, starting from the first start value.

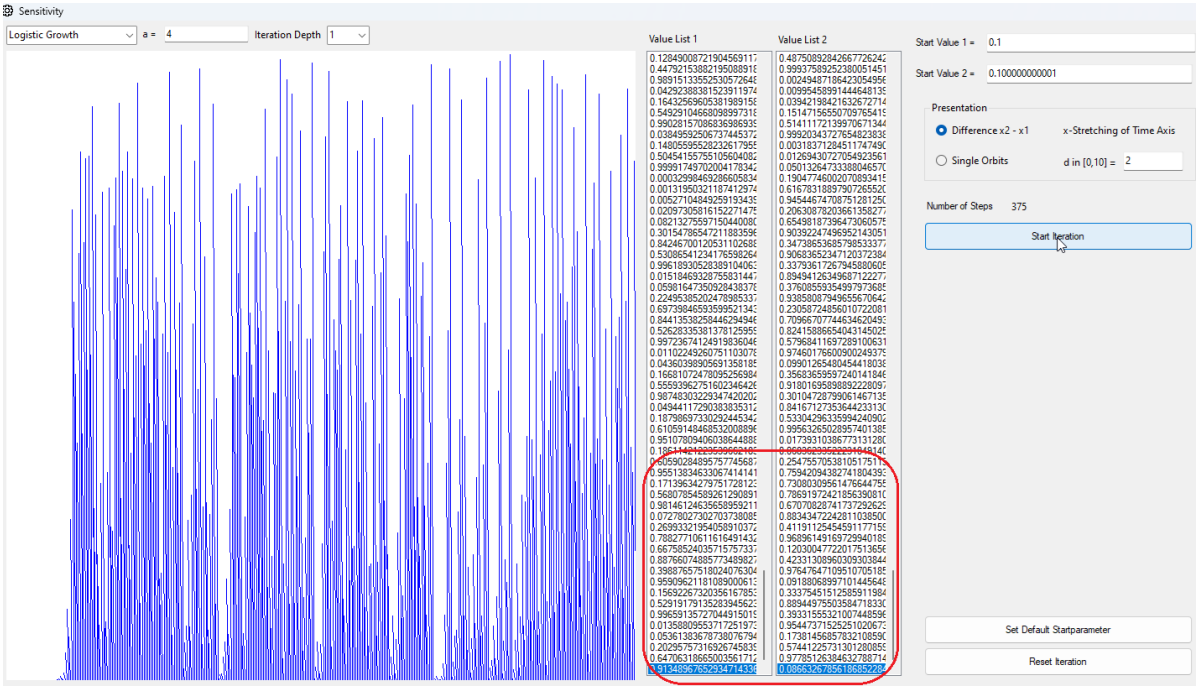
6

The sequence members of the iteration are listed here, starting from the second start value.

7, 8

Two start values can be entered here. These can be very close to each other. The aim is then to show that the generated sequence members diverge arbitrarily during iteration.

Example



Sensitivity: Two slightly different starting values. The difference between the orbits is displayed.

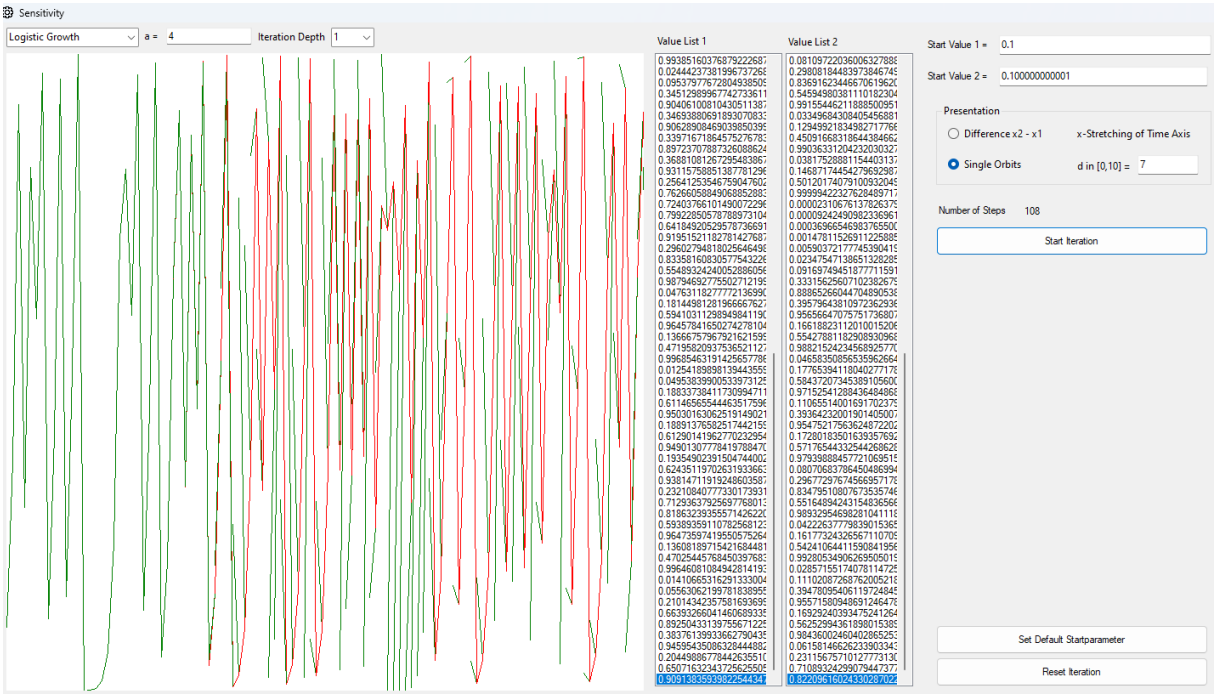
The two starting values above differ by 0.000000000001. As you can see above, the orbits are quite similar at the beginning, but then gradually start to diverge from the area marked in red. The graph shows the difference between the orbits. This is almost zero at the beginning, but then soon makes significant jumps.

10

Two representations are available for the graphical display of the iteration. Either the difference of the generated sequence elements is displayed starting from the initial value one or two. You can see that this difference can become very large after just a few steps within the iteration interval.

Alternatively, the two generated sequences can also be displayed separately.

In both cases, the display is on the time axis and can be stretched in the x-direction. The default is 2; a value between 1 and 10 is possible.



Sensitivity: The same starting values as before. The orbits are displayed individually

In both cases you can see that the orbits start out in the same direction but soon drift apart significantly. For better visibility, the elongation was set to 7.

9

Display of the number of iteration steps.

11

Pressing the "Start Iteration" button starts the iteration.

12

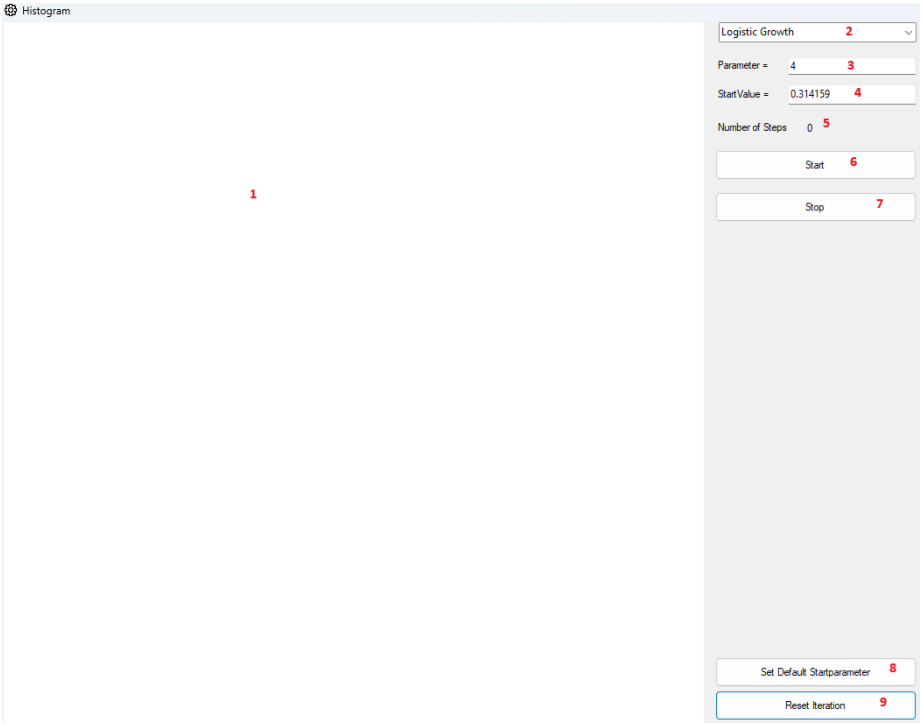
The iteration is reset here, and the start values are set to the default.

13

The iteration is reset here.

4.3. Histograms

In the chaotic case, the distribution of the iteration values can be examined in a histogram (this is not interesting in the non-chaotic case). The following window is opened in the menu item "Growth Models - Histogram":



Window for creating histograms

In this window, the value range is divided into small intervals. The width of such an interval corresponds approximately to the pixel size. The number of times an interval is "hit" by an x value during the iteration is then counted. The frequency distribution is then fairly even in the middle or in the tent map. With logistic growth or the parabola, the arcs on the far left and far right can be explained (see mathematical documentation).

1

The histogram is displayed here.

2

The iteration function is selected here. As always, the following are available:

- Tent map
- Logistical growth
- Parabola

3

The parameter of the iteration function is entered here. It only makes sense to examine the histogram in the chaotic case. This depends on the iteration function:

- Tent map a = 2
- Logistic growth a = 4
- Iteration of the parabola a = 2

4

You can enter a start value for the iteration here. If the start value is changed, an existing iteration is reset.

5

Display of the number of steps during the iteration.

6

The iteration is started here

7

And stopped here.

8

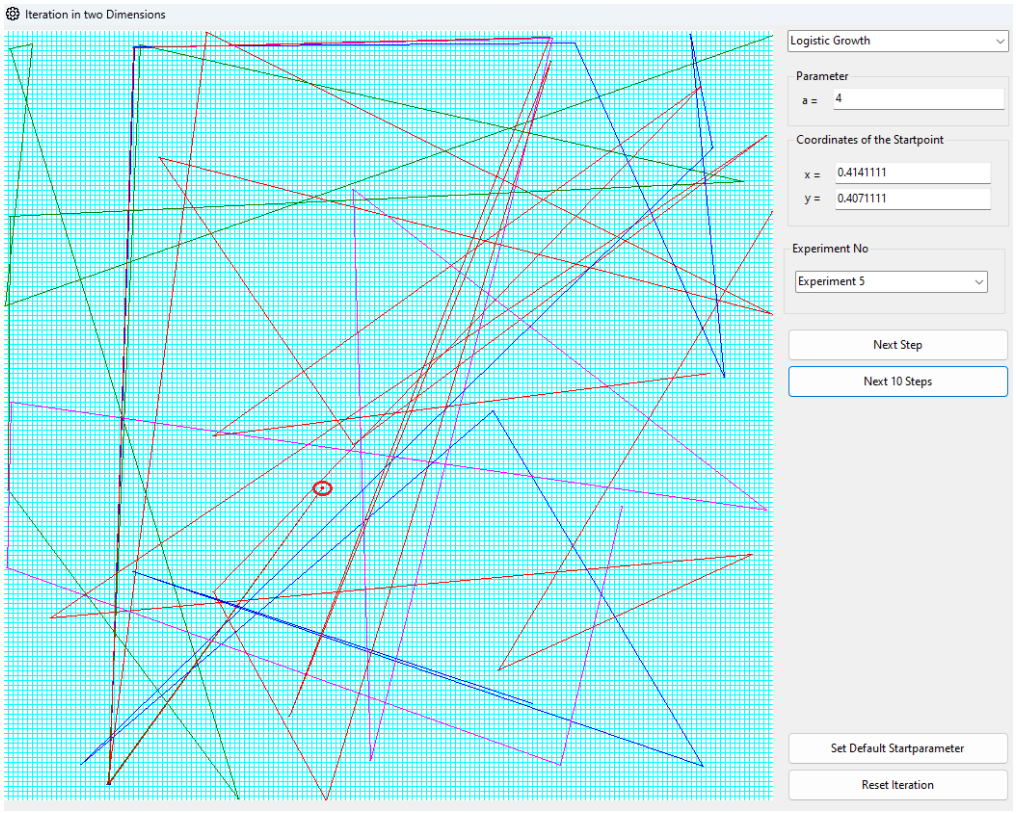
The iteration is reset here, and the start values are set to the default.

9

The iteration is reset, and the start values remain unchanged.

4.4. Two-dimensional Representation

The menu item "Growth Models - Two-dimensional representation" opens the following window:



Window for two-dimensional display

Before we explain the significance of this experiment, let us first explain the function of the individual fields in the window above.

1

The area for the graphical representation

2

The iteration function is selected here. As always, the following are available:

- Tent map
- Logistical growth
- Parabola

3

The parameter of the iteration function is entered here. The analysis only makes sense in the chaotic case. This depends on the iteration function:

- Tent map a = 2
- Logistic growth a = 4
- Iteration of the parabola a = 2

4 and **5**

Here, two start values x_1 and y_1 are entered. This provides a starting point $P(x_1, y_1)$. This is entered as a point on the left-hand side of the diagram.

6

The experimenter can now carry out up to five different experiments with different starting points. The corresponding orbits are shown in different colours in the diagram. In field 5, the experimenter can assign a number between 1 and 5 to each experiment.

We now assume that the experimenter has a limited measurement accuracy, which is represented in the diagram by the light blue grid. The grid size here is 5x5 pixels, which corresponds approximately to a mathematical grid of 0.00825 x 0.00825 units.

The experimenter has now started 5 experiments with the starting points in the diagram shown above: $P_1(0.414, 0.407), P_2(0.4141, 0.4071), P_3(0.41411, 0.40711),$

$P_4(0.414111, 0.407111), P_5(0.4141111, 0.4071111)$

However, all starting points are in the same measurement square of the experimenter. This means that the starting point is always the same for him. This is marked by a red circle in the diagram.

Now he runs each experiment. Due to the sensitivity, the orbits diverge after just a few steps. For the experimenter, it therefore looks as if the same starting point leads to completely different orbits during the iteration. The behaviour of the system appears to be random.

One of the available iteration functions is iterated, the same in both components x and y.

7

A single step of the iteration is performed and the new point in the diagram is connected to the previous one.

8

The next 10 steps of the iteration are carried out here and added to the diagram.

9

The iteration is reset here, and the start parameters are set to the default.

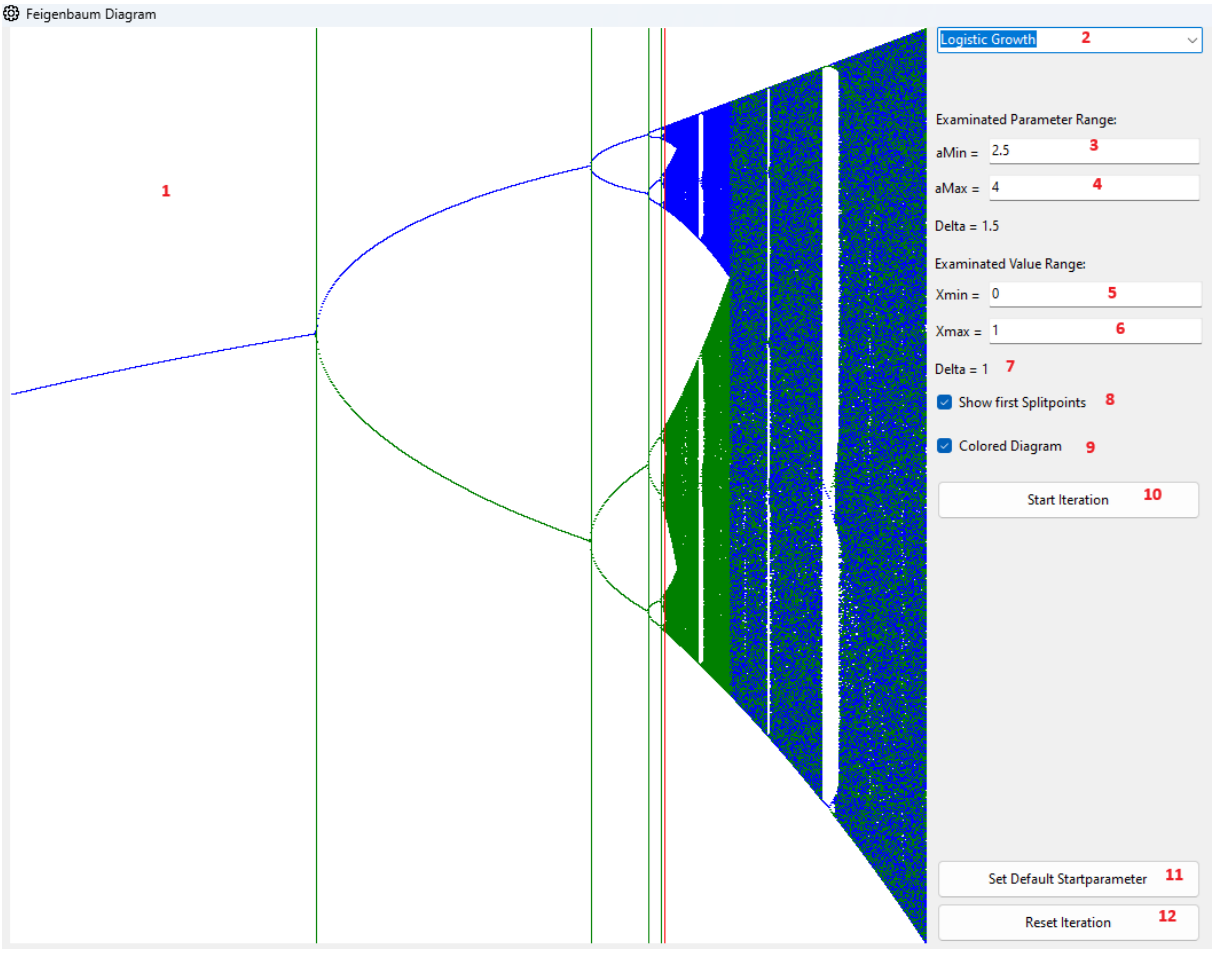
10

Only the iteration is reset here.

4.5. Feigenbaum Diagram

The simulator can be used to examine the well-known Feigenbaum diagram and period doubling. This is described in more detail in the mathematical documentation.

The corresponding window is displayed in the menu item "Growth Models - Feigenbaum".



Window for examining the Feigenbaum diagram

The diagram shows values of the parameter a in the horizontal direction, which grows linearly in this direction. For each parameter value, the iteration is first allowed to run for a while in the hope that it will have settled on an attractive fixed point or an attractive cycle, if there is one at all. The orbit is then entered into the diagram in a vertical direction.

In the above window, the analysis starts at a = 2.5. Up to the first split point (the first green vertical line), you have an attractive fixed point, and the iteration draws the fixed point to which the iteration converges for each a in this range. After the first split point, you have a 2-cycle up to the second split point, then a 4-cycle and so on until the behaviour becomes chaotic. This point is marked with a red vertical line. Further to the right, most cycles are no longer visible except in individual small windows.

Now first to the meaning of the individual fields.

1

The area for the graphical representation.

2

Here you can select the type of iteration. The following are available:

- Tent map
- Logistical growth
- Parabola
- Mandelbrot iteration for real numbers

3, 4

The parameter range to be examined for the parameter a is defined here. The minimum a is defined in 3 and the maximum a in 4. The parameter interval [aMin, aMax] is examined. The width of this range is displayed as the delta.

These values can be changed manually, but they must be within the permitted parameter range of the iteration:

- Tent mapping: a must be in the range]0,2]
- Logistic growth: a must be in the range]0.4]
- Parabola: a must be in the range]0,2]
- Mandelbrot: a must be in the range [-2, 0[

5, 6, 7

You may only want to examine a section of the value range. This section is defined here: The minimum value of x in 5 and the maximum value in 6. The range of values [Xmin, Xmax] is examined. The width of the examined value interval is shown in 7.

8

The vertical lines in the diagram show the first split points of the first period doubling from $a \approx 3.5$. This display can be shown or hidden.

9

The diagram can be displayed in one or two colours. This option can be selected here. If you would like to experiment with other colours, you can adapt the program code or the *SetColor* function in the code of the *FrmFeigenbaum* itself accordingly.

10

The iteration is started here.

11

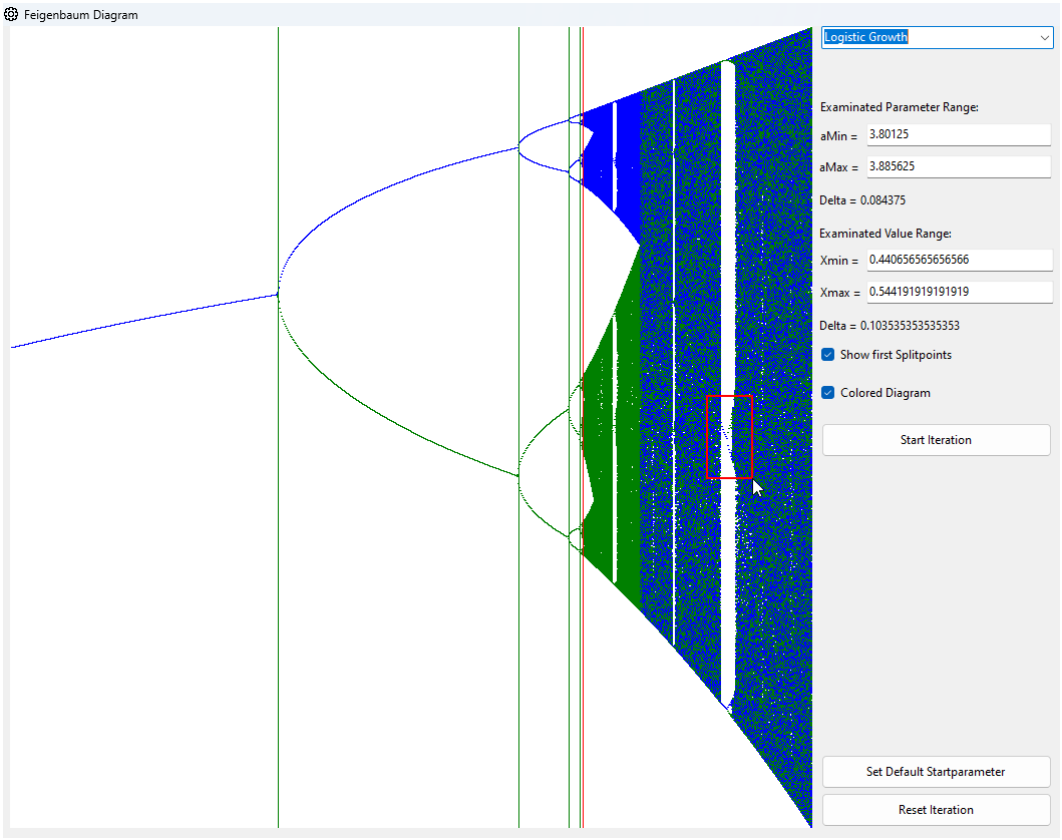
The iteration is reset, and the start parameters are set to the default.

12

The iteration is reset.

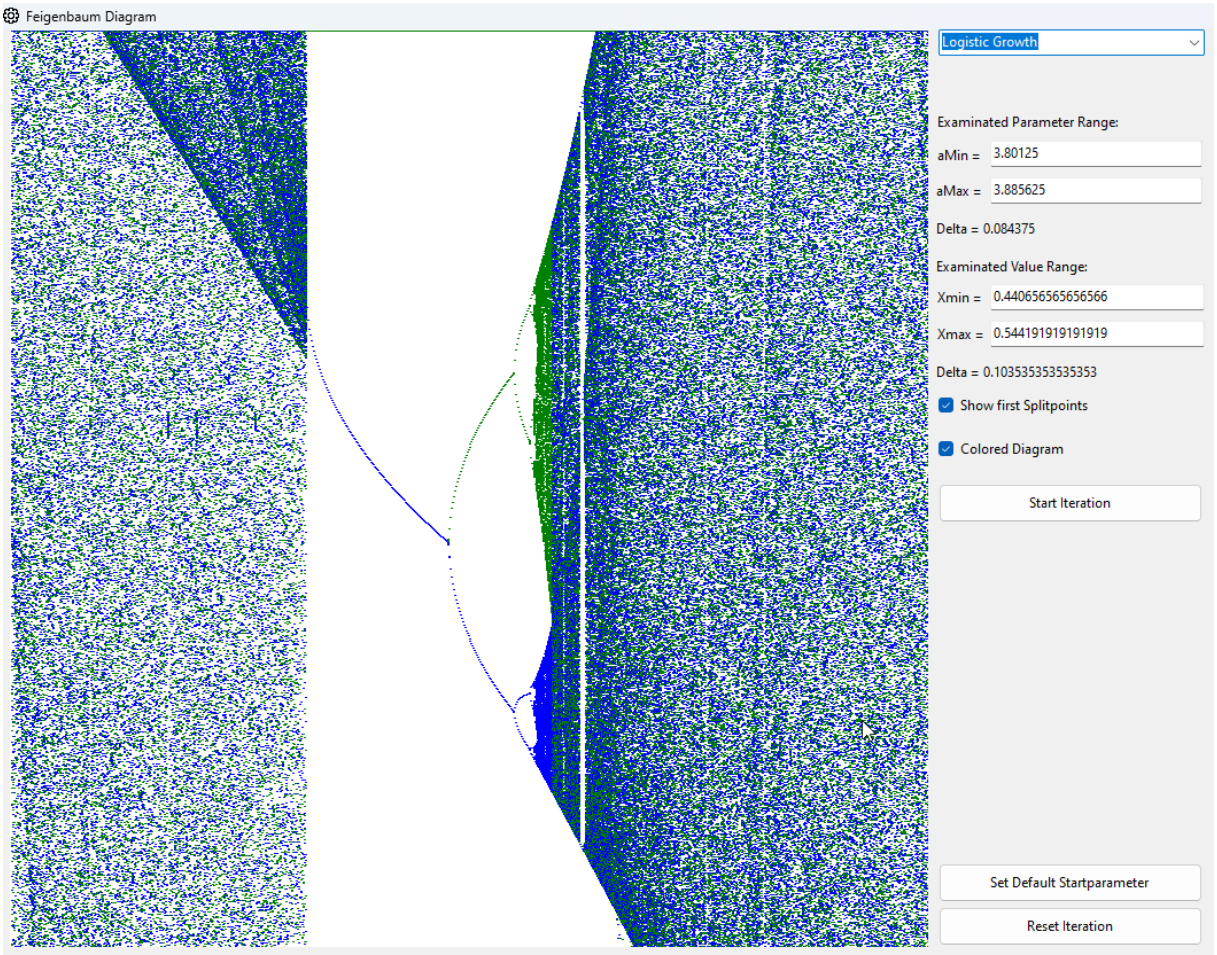
Selection of small sections in the diagram

The user now has the option of using a selection rectangle to select a section that they want to examine more closely. This is done by holding down the left mouse button. The parameter range in fields 2-4 on the right and the value range in fields 5-7 are automatically moved accordingly.



Selection of a small section in the diagram - see red rectangle

Above, the user would like to examine the surroundings of a 3-cyclic point in more detail. To do this, he has drawn a corresponding rectangle by holding down the left mouse button. On the right-hand side, the parameter range and value range have been adjusted accordingly. If the iteration is now started, this area is displayed.

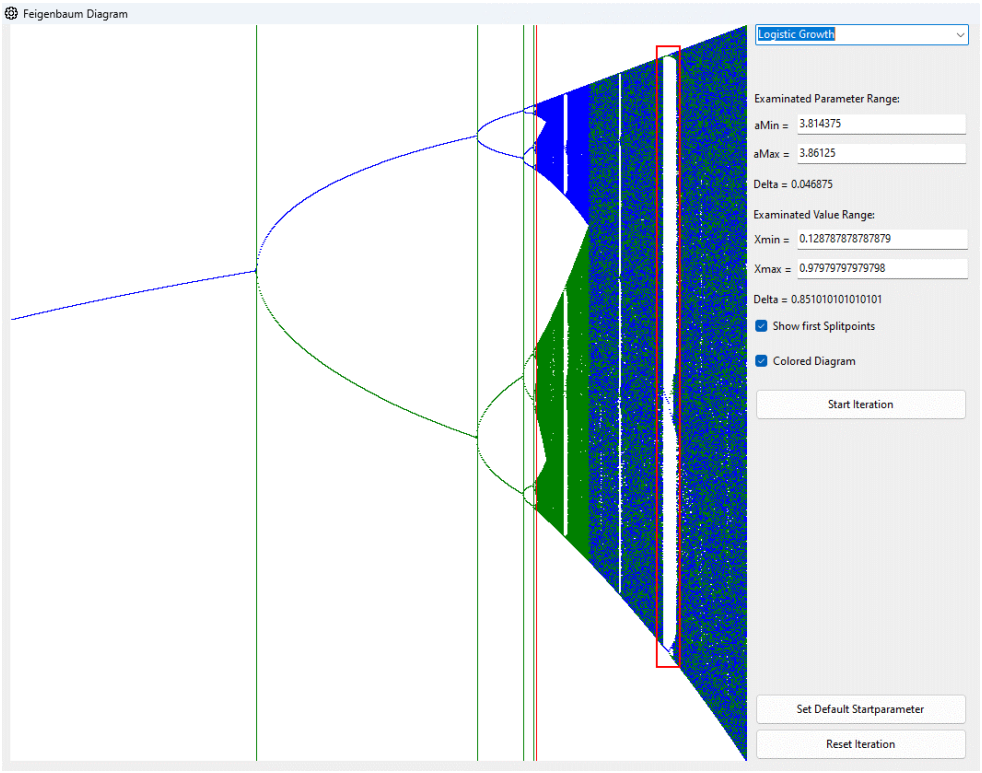


Display of the selected environment of the 3-cycle point

Here you can see how a 3-cycle emerges from the chaos. Due to the reduced value range, only one point of this cycle can be seen here. The diagram then returns to chaos by doubling the period again.

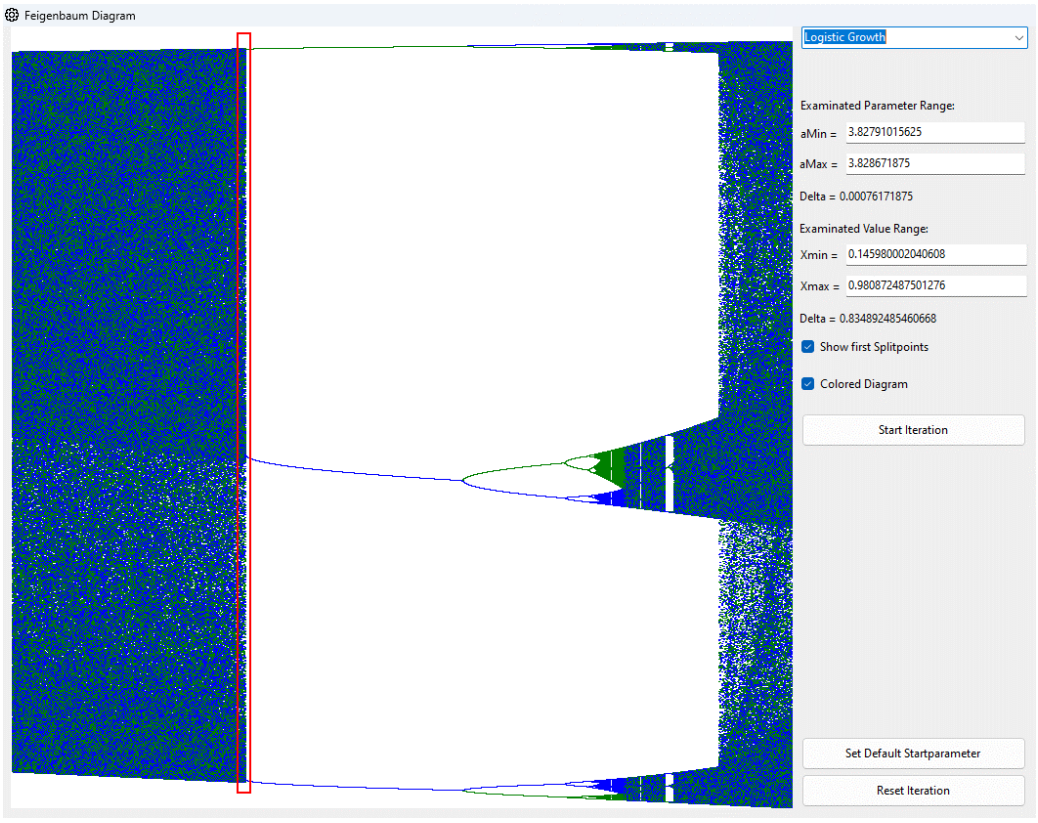
This selection procedure can also be used to examine for which parameter values of a certain cycles can be expected. By placing the selection rectangle, the corresponding intervals [aMin, aMax] are displayed on the right-hand side.

The following selection triangle can be used to limit the occurrence of the 3-cycle. You can read off here: $a \in [3.801, 3.885]$



Positioning the selection triangle to determine a parameter interval

If we now restart the iteration in this area of a, we can further narrow down the 3-cycle:

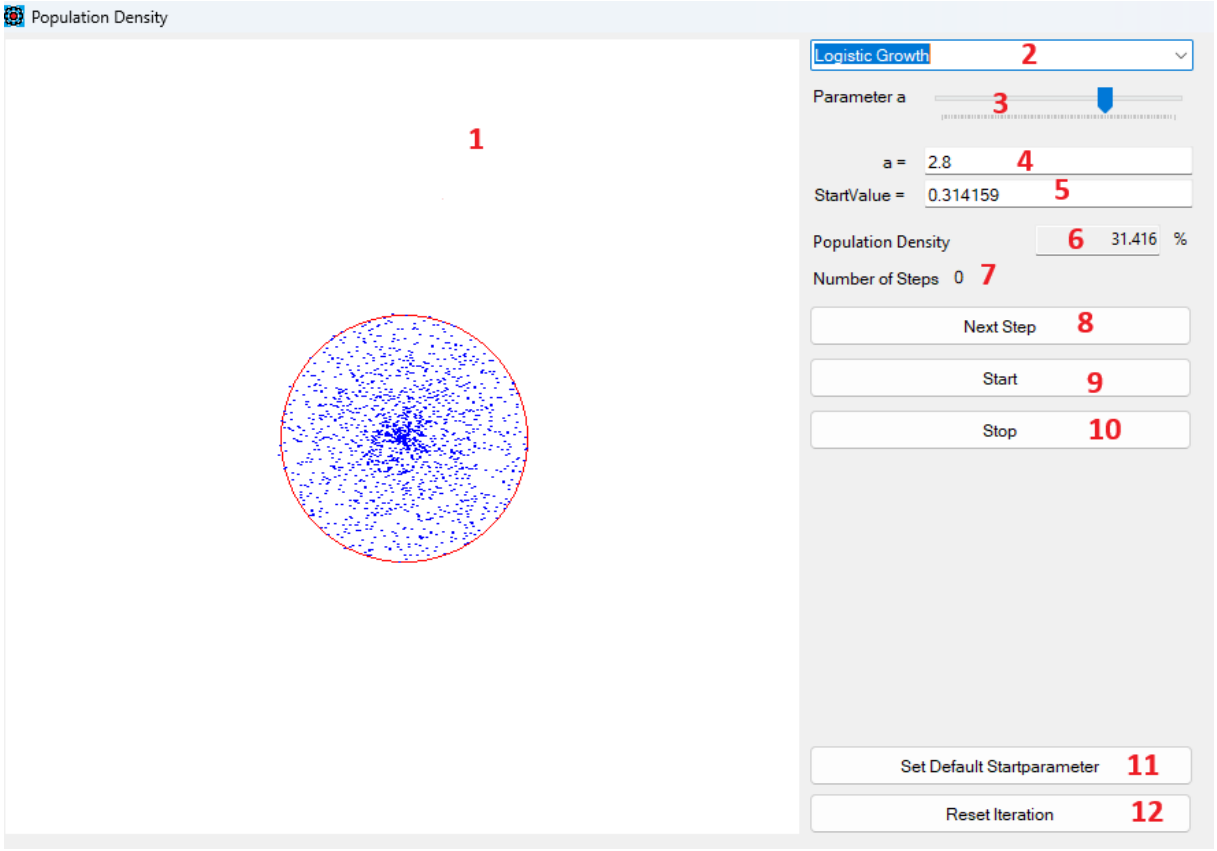


The 3-cycle occurs for $a \in [3.8279, 3.8286]$

4.6. Population density

This menu item is intended to illustrate the growth behaviour of a population in a circle diagram.

The corresponding window is opened in the "Growth models - Population density" menu.



Window for displaying the population density

Significance of the individual areas:

1

Representation of the circle diagram

2

Selection of the growth function

3

Shift register for changing the parameter a

4

Manual input of the parameter a

5

Start value

6

Display of population size in %

7

Number of steps in the iteration

8

Single iteration step

9

Start the iteration. It then runs until

10

Stop is pressed. The iteration can then be continued or reset.

11

Resets the iteration and also sets the default start values for the user.

12

Resets the iteration. The start values remain unchanged.

5. Complex Iteration Menu

5.1. Newton Iteration

The Newton iteration method for approximating the zeros of real functions can be derived on an elementary basis in secondary school or at least explained visually. This method can be extended to the complex level. This is described in the mathematical documentation. It is less about effectively finding the zeros of complex polynomials and more about determining their "catchment area" or "basin". This means the following: Let us assume $\zeta \in \mathbb{C}$ is a zero of a complex polynomial $p(z)$. Let's give this zero the colour red. We then start the Newton iteration with a pixel point of a window shape or the corresponding starting point $z_0 \in \mathbb{C}$. If this point converges towards the zero during the iteration ζ, we also colour it red. z_0 then lies in the basin of ζ. If there are several zeros, we give each zero a colour and then colour the converging starting points with the respective colour. Points that show no convergence behaviour after a certain number of steps or that approach ∞ remain coloured black.

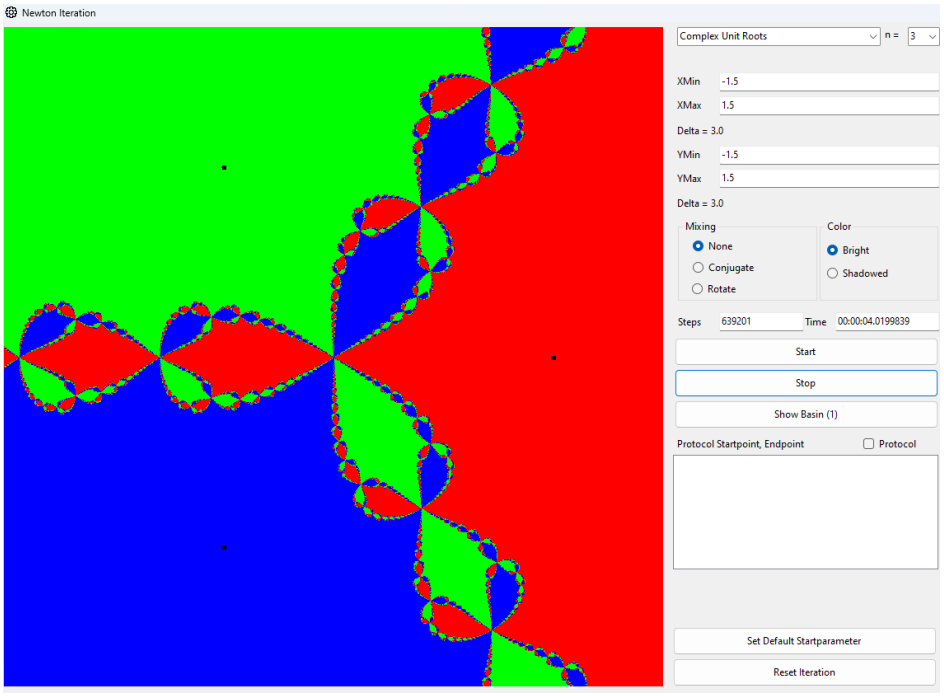
A point is considered convergent towards a zero point ζ if it comes sufficiently close to it. Since a zero point corresponds to a super-attractive fixed point of Newton's method, it can no longer "escape" its immediate basin.

For the zeros of the polynomial $p(z) = z^2 - 1$ all starting points z_0 with $Re(z_0) < 0$ strive towards the zero -1 and all starting points with $Re(z_0) > 0$ towards the zero +1. The edge separating the two basins is the y-axis. See the derivation in the mathematical documentation.

However, the basins and their boundary can look very complex for polynomials of degree > 2.

Among other things, the program can be used to examine the complex roots of unity, i.e. the zeros of the polynomial $p(z) = z^n - 1, n \in \mathbb{N}$ and their basins can be examined. In the program, n is a

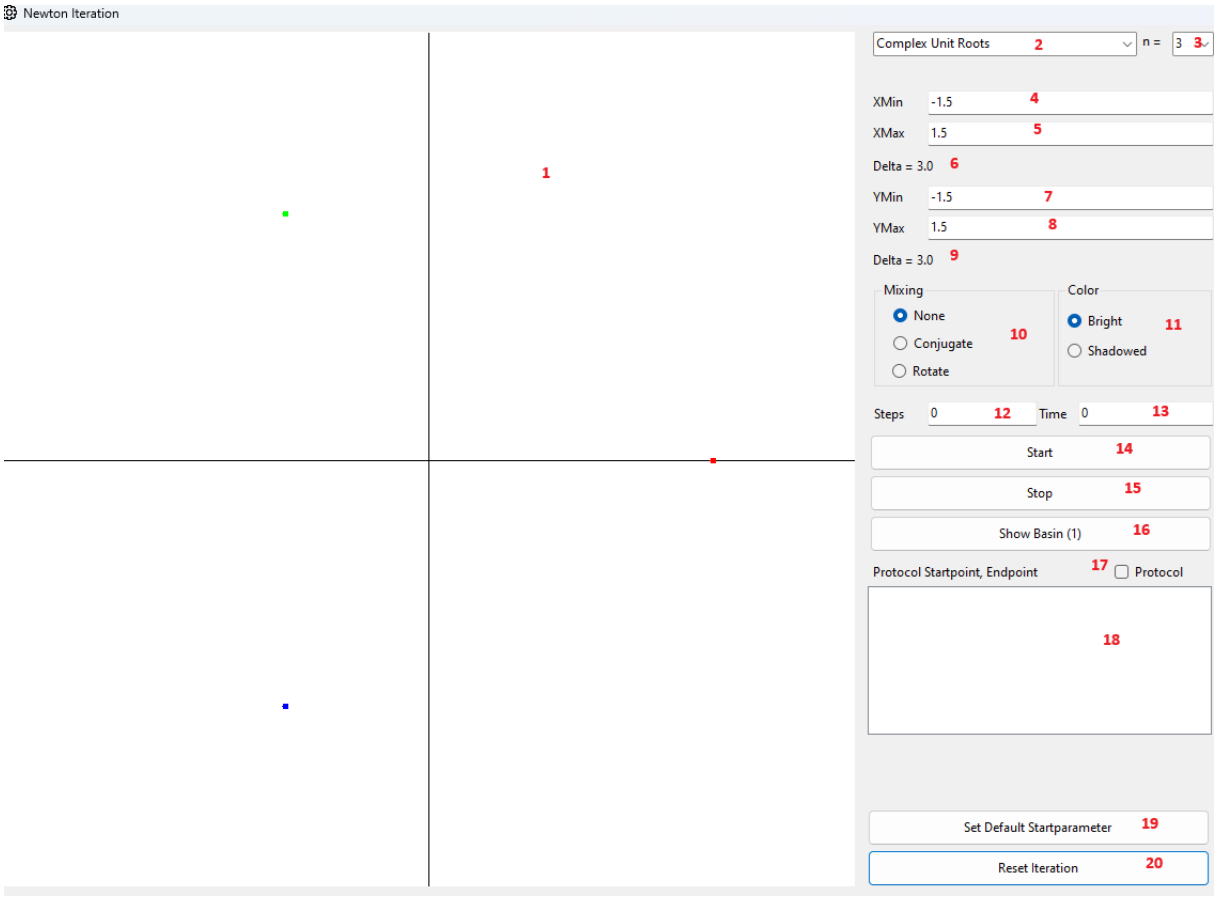
natural number between 2 and 12. The following figure shows the basins of the roots of unity in the case $n = 3$: $\zeta_1 = 1\ (red), \zeta_2 = 0.5 + \frac{\sqrt{3}}{2}i\ (green), \zeta_3 = 0.5 - \frac{\sqrt{3}}{2}i\ (blue)$.



Basins of unit roots of degree three

These basins are discussed in more detail in the mathematical documentation.

The following window opens in the "Complex Iteration - Newton Iteration" menu:



Window for investigations into the Newton method

1

Drawing area or section of the complex plane.

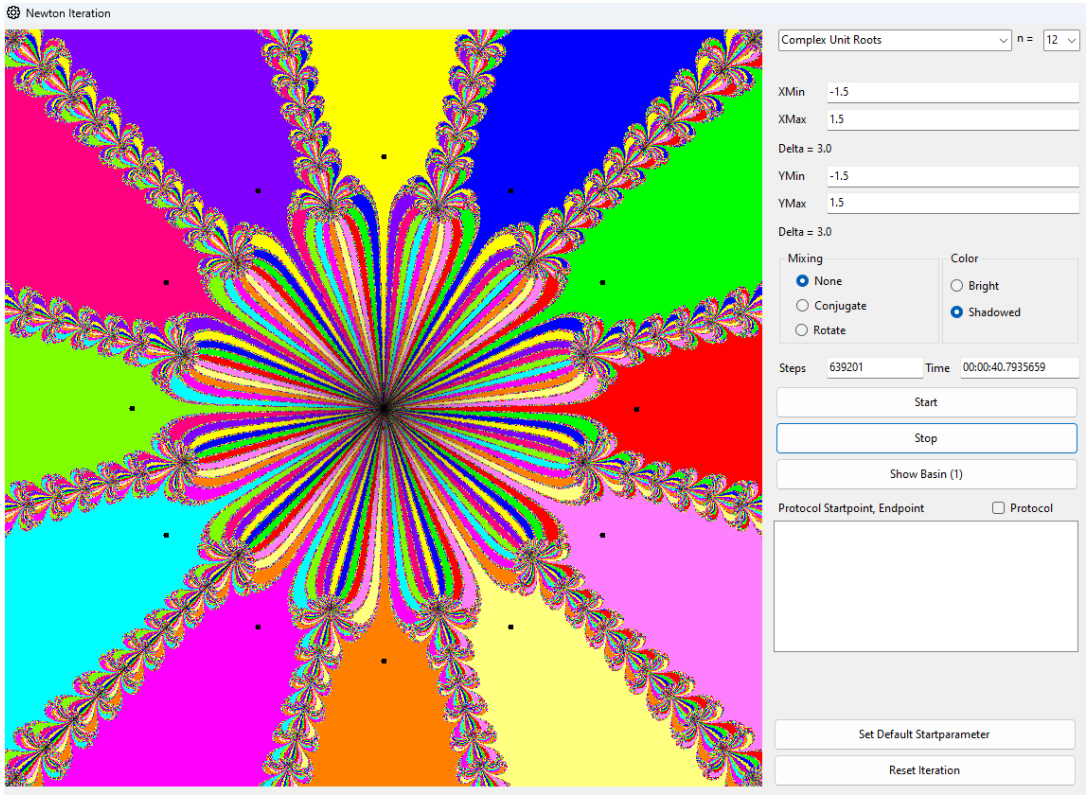
2

Select the polynomial whose zeros are to be analysed. You can choose from the following:

- Complex unit roots
- polynomial with three zeros, namely $p(z) = (z-1)(z+1)(z-c)$ where *c* can be defined by the user
- Complex roots of unity after inversion on the unit circle. This allows the investigation of the behaviour at infinity

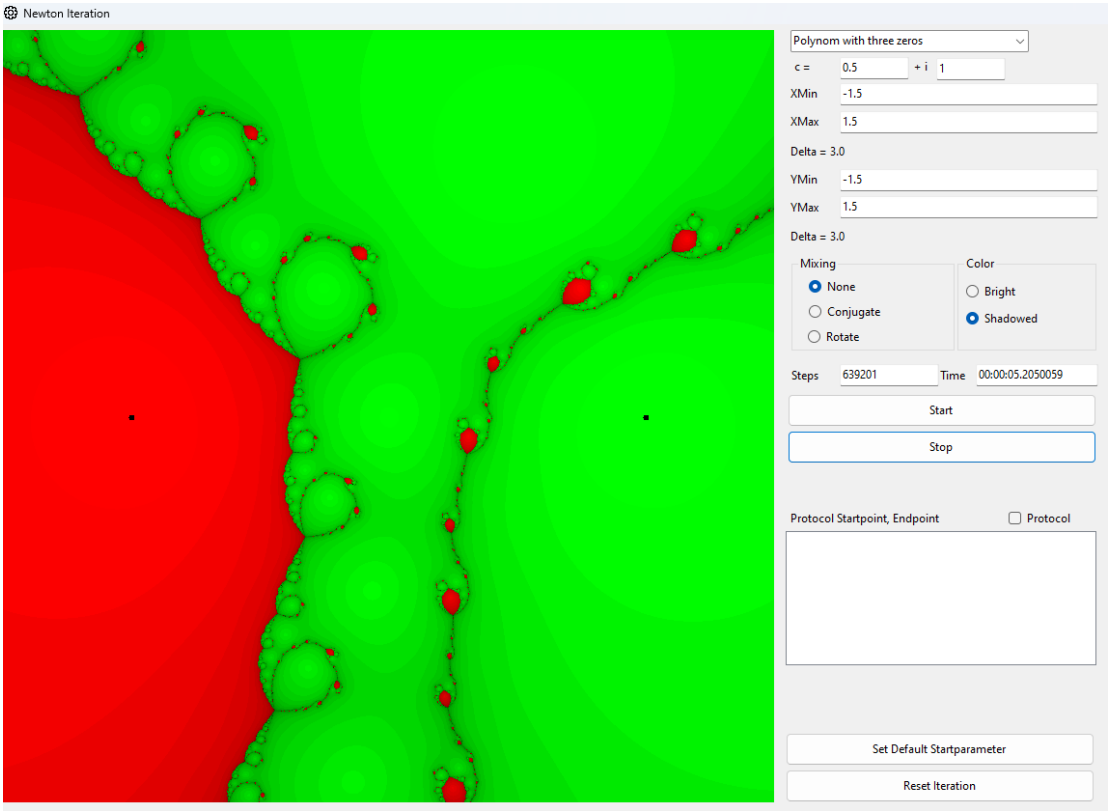
3

The degree can be specified here for the complex roots of unity or their inverses. Values of *n* between 3 and 12 are possible.



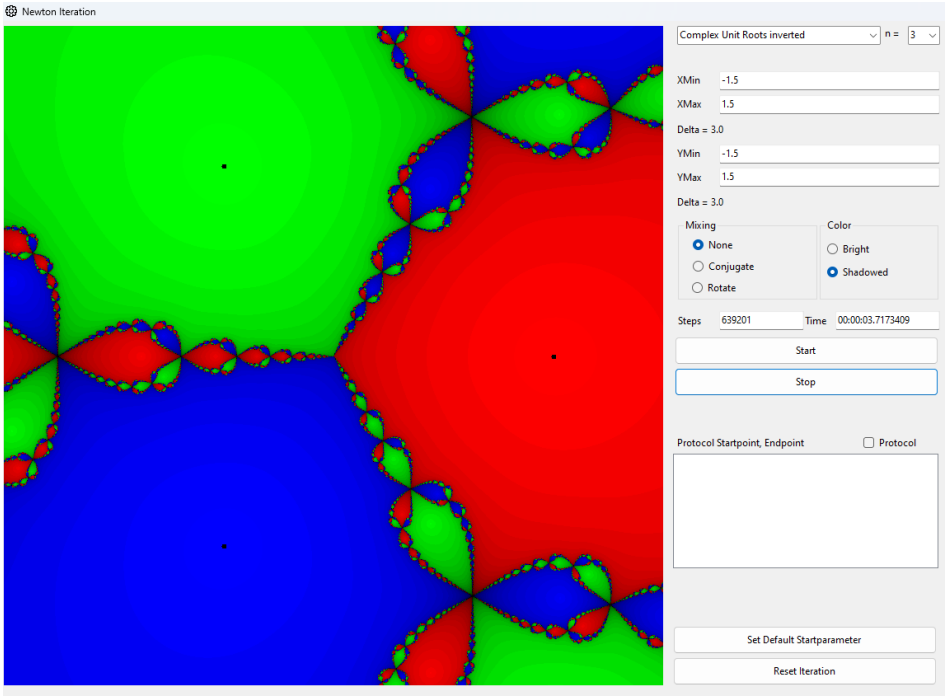
Basins of the twelfth unit roots in shaded representation

If the option "Polynomial with three zeros" is selected, a field appears below the selection box in which *c* can be entered:



Polynomial of degree three with the zeros $\pm 1, 0.5 + i$

The inverted roots of unity can be used to investigate how the Newton method works at infinity. For details, see the mathematical documentation.

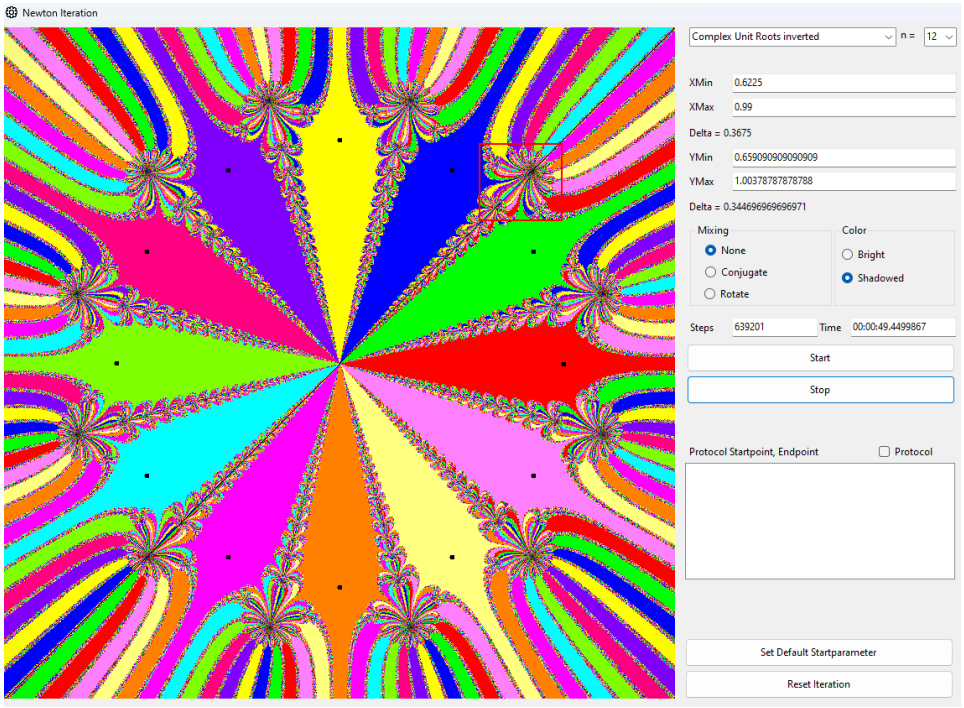


Inverted roots of unity of degree three

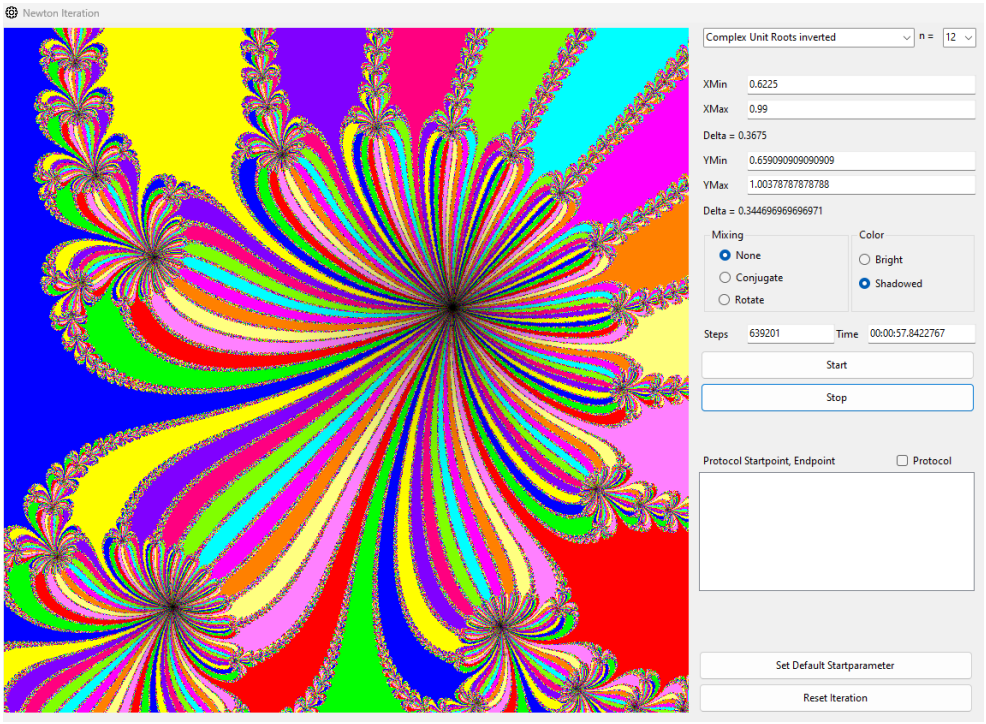
4, 5, 6, 7, 8, 9

If a section of the complex plane is selected with the mouse (by holding down the left mouse button), the coordinates of the section are displayed here, including its width and height.

The user can also enter a section manually here.



Inverted unit roots of degree 12 with a selected section



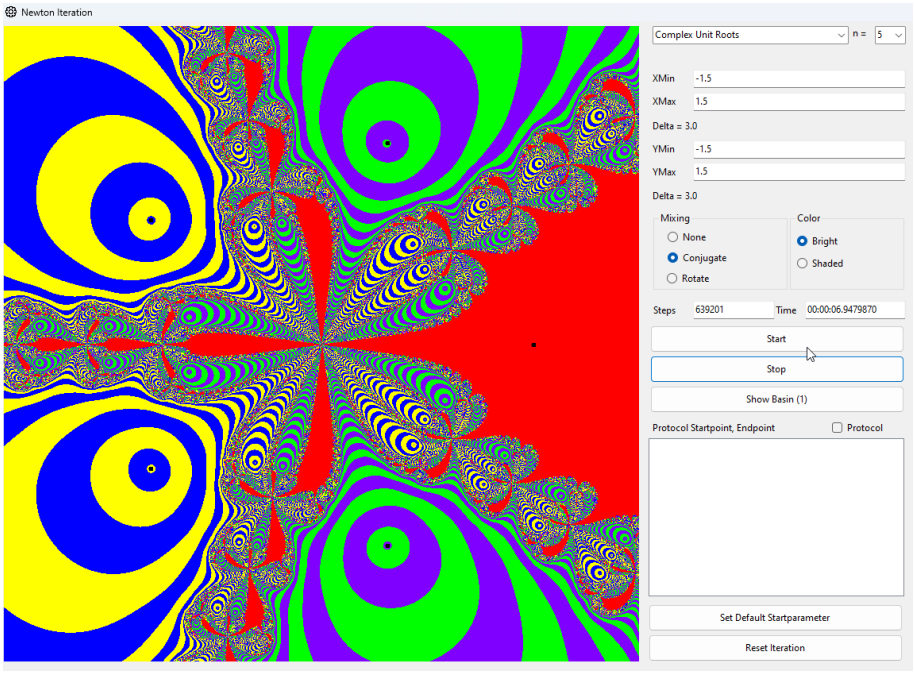
Iteration in the selected section, which is thus displayed enlarged

Since the image has fractal properties, the zoom can be continued infinitely within the calculation accuracy and similar images always appear

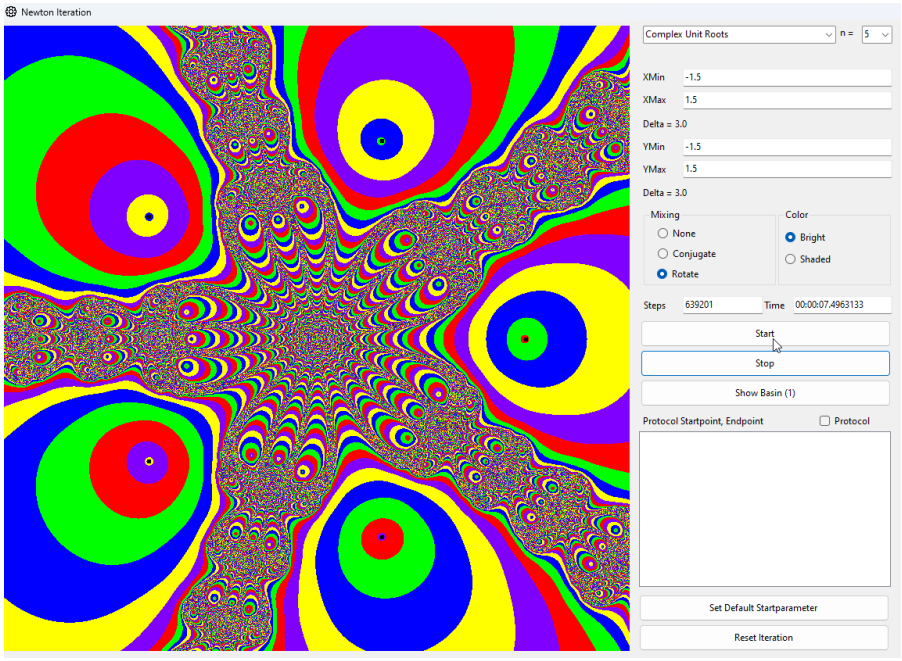
10

This is more about creating interesting images. The basins are "mixed". There is a choice:

- No mixture: you can see the original and real basins.
- Conjugation: Here, the iterated z_n is conjugated at each iteration step. As a result, basins that are symmetrical to the x-axis are "visited" alternately during the iteration and the starting point is then assigned the color of the basin where it is currently located at the end of the iteration. This makes it clear whether a point ends up at "its" zero point after an even or odd number of steps.
- Rotation: Same idea, but the iteration point is rotated at each iteration step and $2\pi/n$ rotated.



Representation of the fifth roots of unity including conjugation



Rotated representation of the roots of unity of degree five

For example, if you move from the red centre of the unit root $\zeta = 1$ clockwise through the basins, you can see that the colour red moves one step further away from the center with each additional rotation, until after five steps you end up back in the basin of $\zeta = 1$ again after five steps.

11

To make the "speed" of the convergence more visible, the colour of the basin can be subdivided here. The following colours are available:

- Light: The pool colour is uniformly light. This is the default if the "Rotation" or "Conjugation" option is selected at the same time.

- Shaded: Points that converge more slowly are given the colour of the pool to which they belong, but slightly darker than points that converge quickly. This is the default when the "No blend" option is selected.

12, 13

As the generation of the images is performance-critical, the number of iteration steps and the elapsed time for the iteration are displayed here.

14

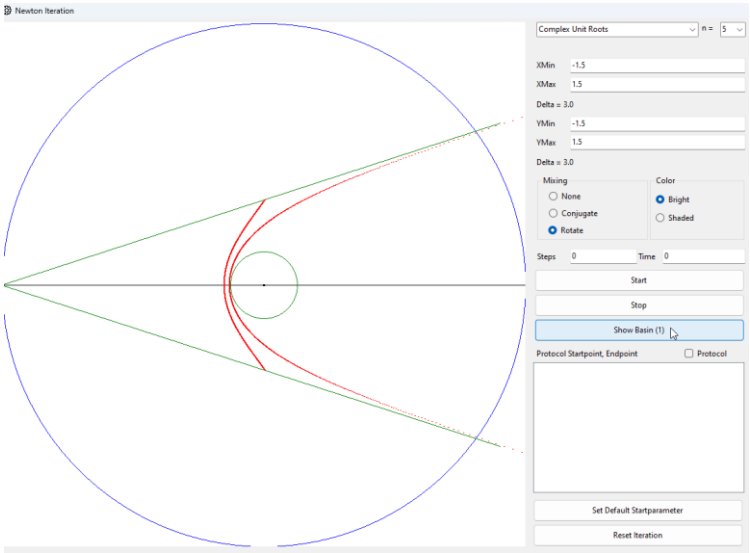
The iteration is started or continued here.

15

The iteration is stopped.

16

To decide when an iteration point lies in the immediate basin of a unit root, this basin must be determined. This key outlines the basin of 1. See mathematical documentation.



The immediate basin of 1

17

A log can be output by activating the checkbox. The display of the respective starting point and the last iterated point before it is considered convergent is implemented. However, the output of a log significantly slows down the iteration.

18

Display of the protocol.

19

The iteration is reset here and the selection windows are set to standard.

20

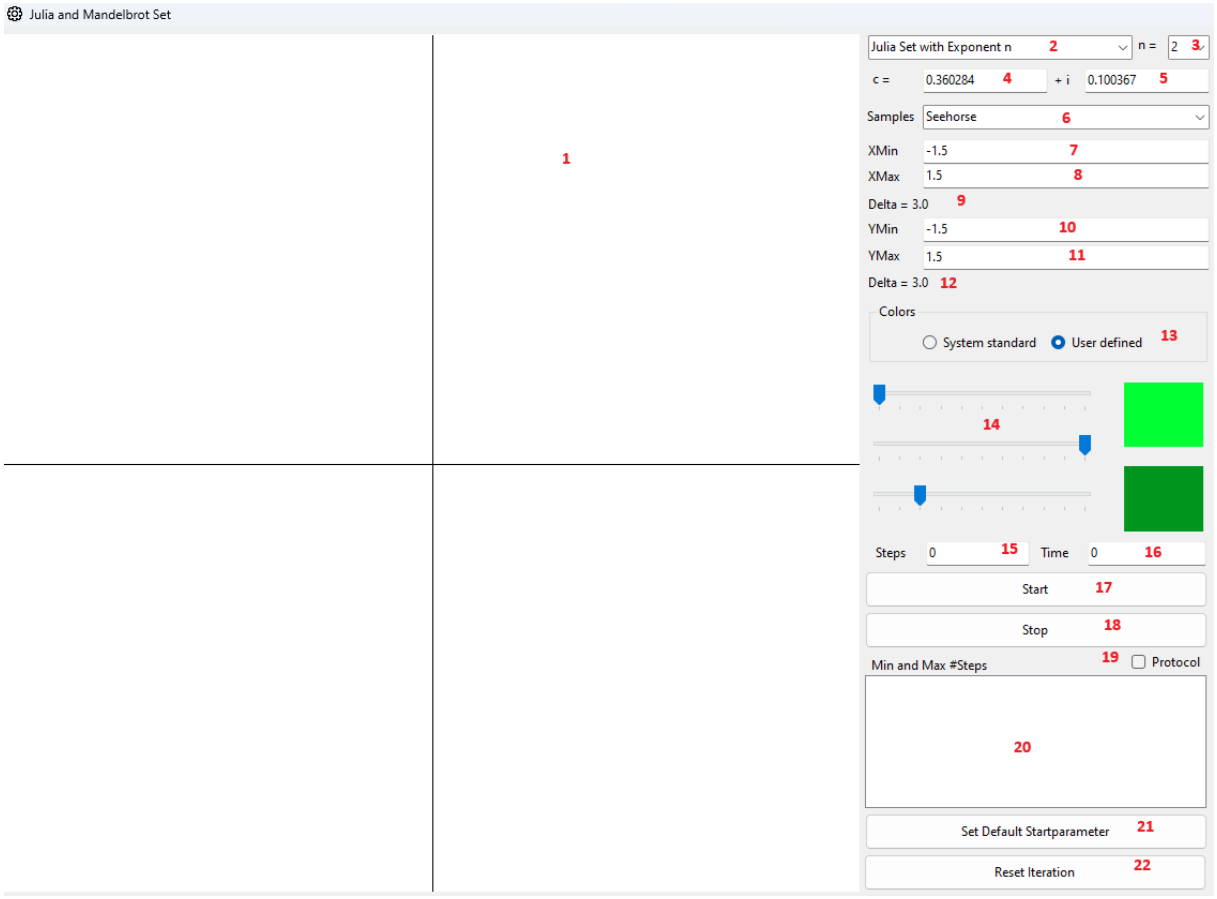
Only the iteration is reset here.

5.2. Julia Set and Mandelbrot Set

When iterating the function $f(z) = z^2$ all starting points z_0 with $|z_0| < 1$ tend towards 0. All starting points with $|z_0| > 1$ move towards infinity. The starting points on the unit circle remain on the unit circle. But the behaviour of the iteration on the unit circle is chaotic.

Now "build in" a small disturbance and replace the iteration with $f(z) = z^2 + c, c \in \mathbb{C}$. In doing so $c = a + ib; a, b \in \mathbb{R}$ can be entered by the user. The zeros of f are then the roots of $-c$. There are now starting points $z_0$which converge towards these zeros, i.e. where the generated sequence of points remains finite, and other starting points which tend towards infinity. The dividing line between the corresponding basins, which was initially a circle, is then increasingly deformed until it crumbles to dust This dividing line, which is also the edge of the basins, is a Julia set (see mathematical documentation). The "Simulator" now makes it possible to generate such Julia sets.

The following window can be opened in the "Complex Iteration - Julia Set" menu:



Window for generating the Julia and Mandelbrot set

1

Drawing area and section of the complex plane

2

Choice between Julia and Mandelbrot set with exponent n (iteration of $f(z) = z^n + c$). With the Julia set, one examines for each starting point $z_0 \in \mathbb{C}$whether it goes against ∞ during the iteration. If not, the starting point is coloured black. If yes, the more iteration steps it takes before it is clear that it is moving to infinity, the lighter the colour. Criteria: See mathematical documentation.

3

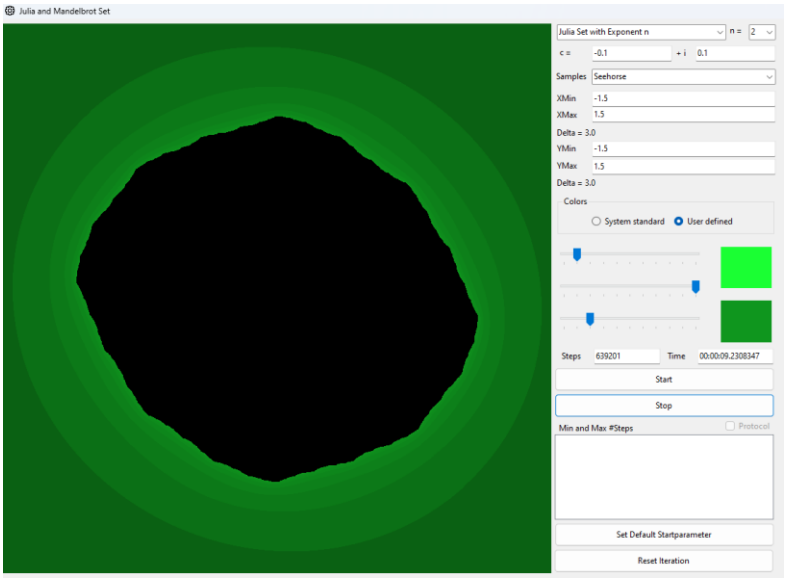
Choice of exponent $n \in \mathbb{N}$.

4, 5

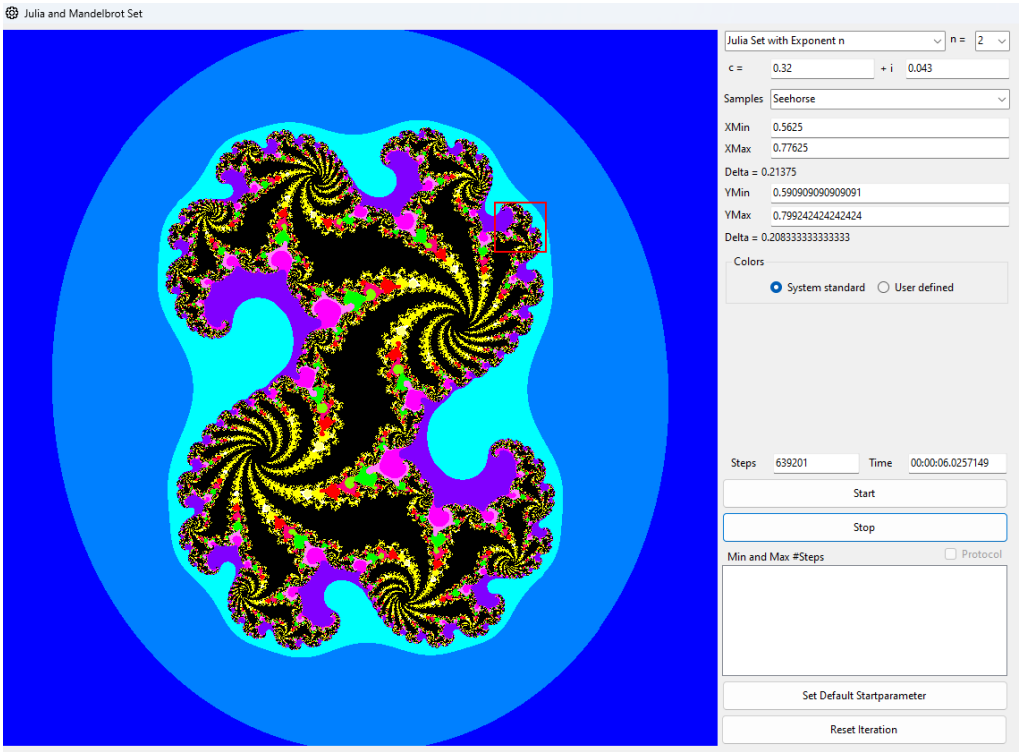
Entering the complex constants $c = a + ib$.

6

Some examples are predefined here. The corresponding constant c is set during selection.

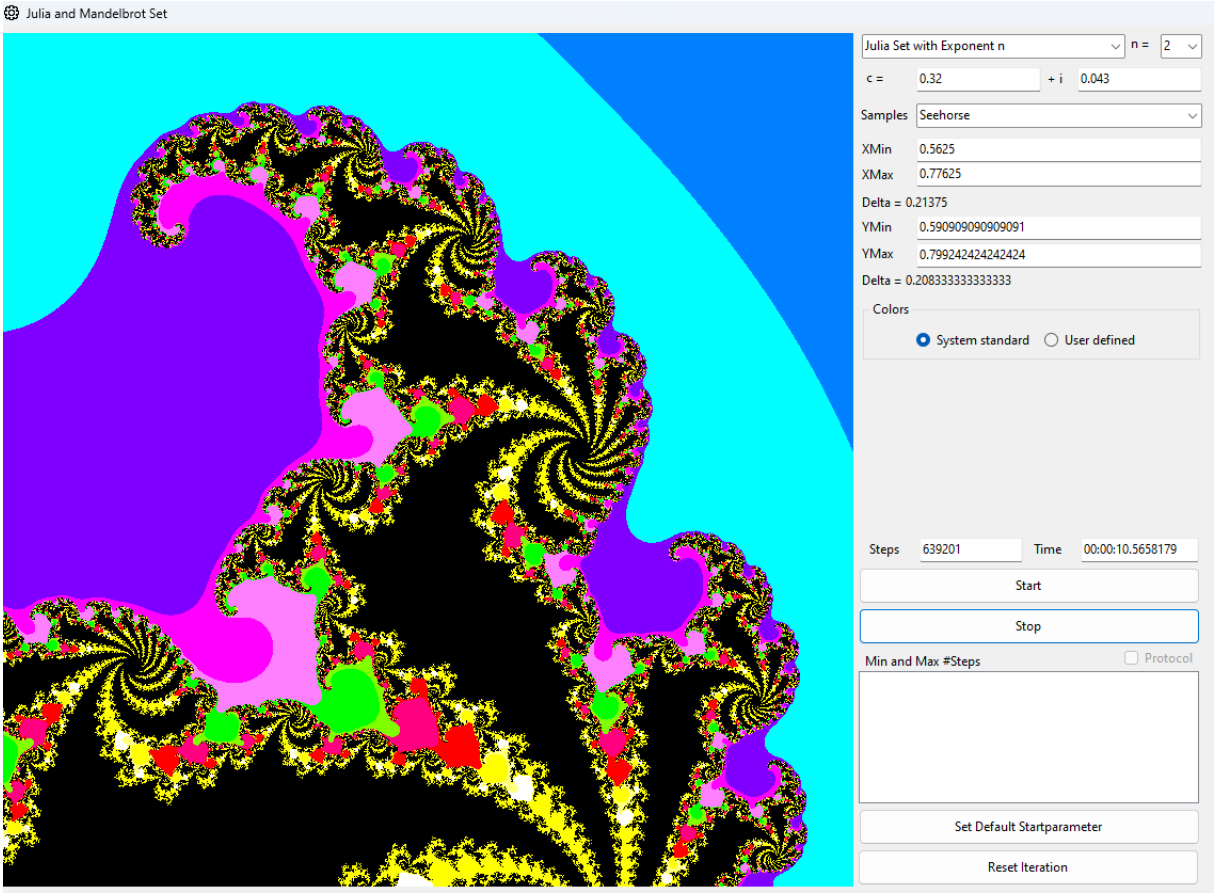


Julia set during a minor "malfunction" $c = -0.1 + 0.1i$. Opposite $c = 0$ the "unit circle" is a little shriveled. The colour is user-defined.

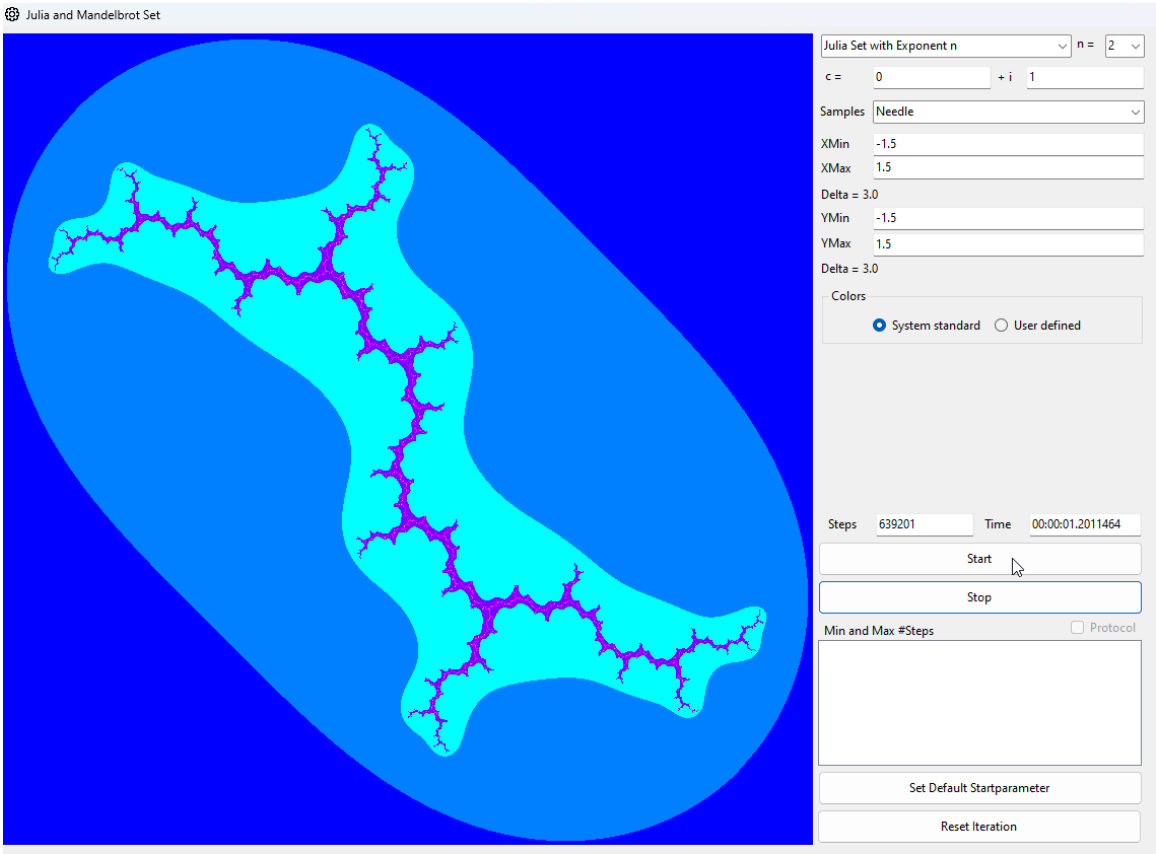


The "Seahorse" with $c = 0.32 + 0.043i$

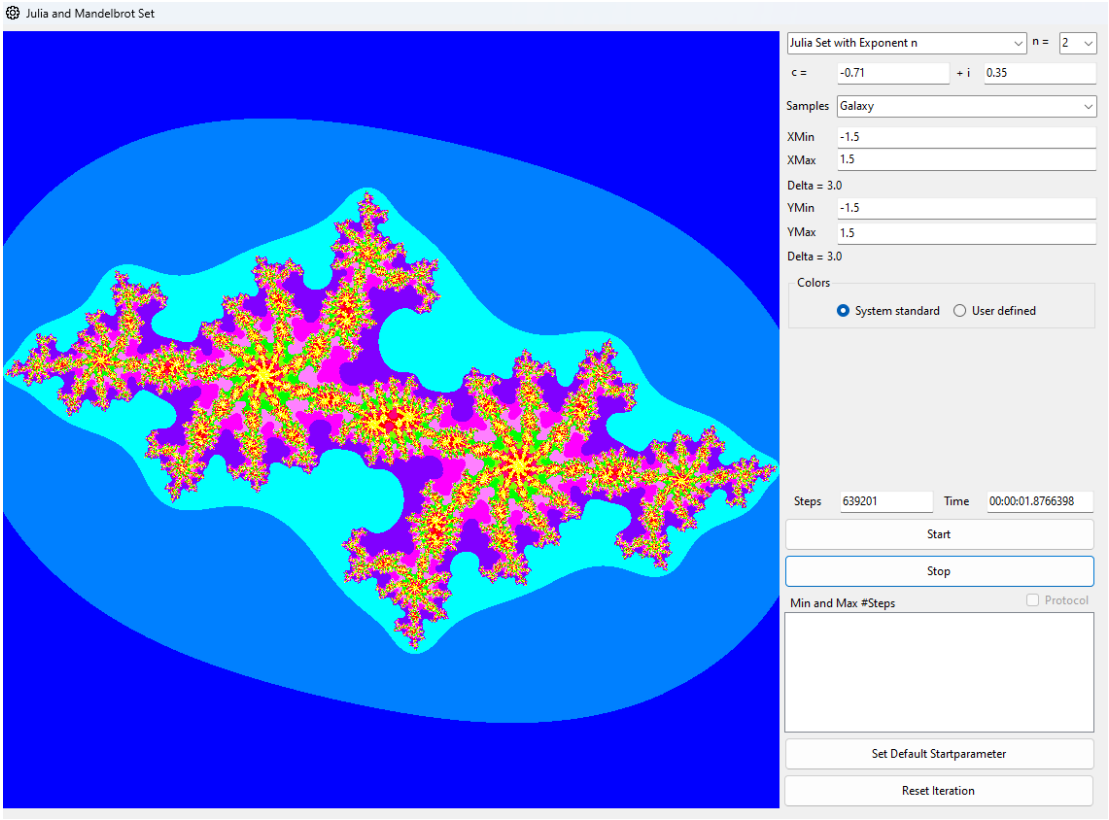
The "disruption" is greater here. The unit circle has turned into a fractal and is unrecognizable. The standard colours of the system were selected. Furthermore, a selection was made by holding down the mouse button, which is examined below.



The seahorse in the selected section.



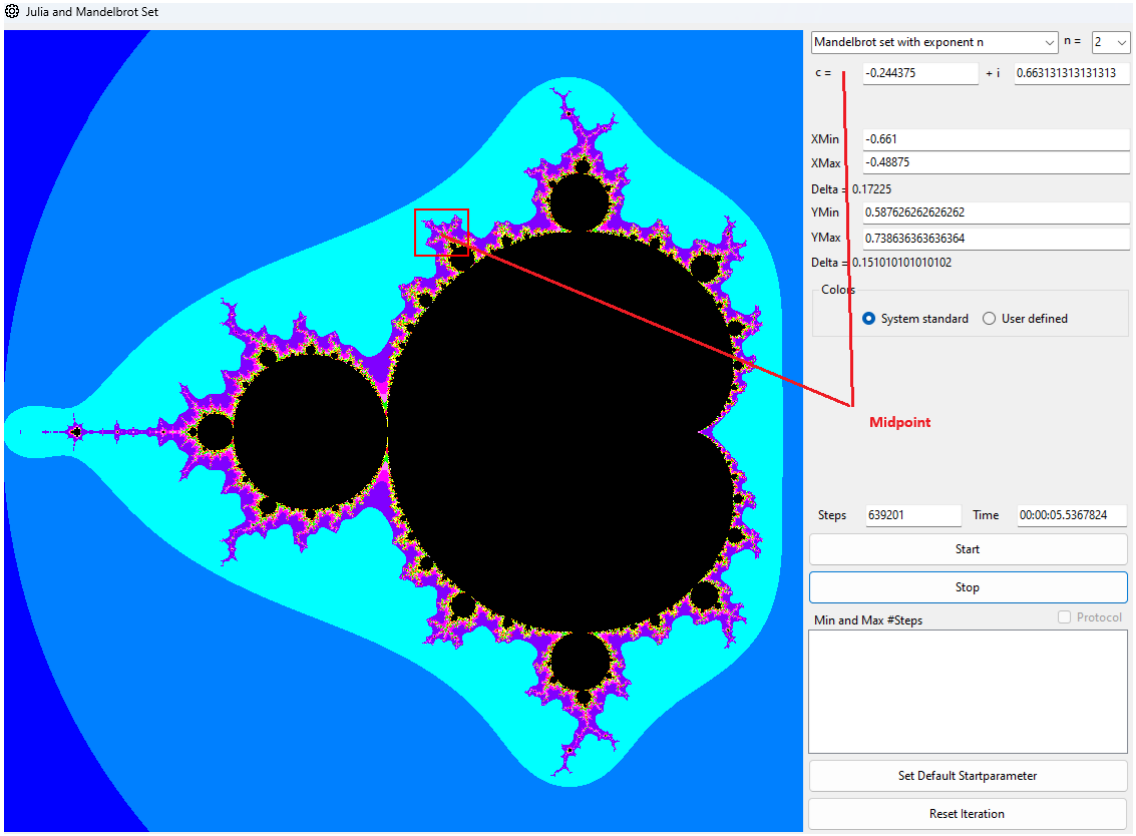
Julia set for the "Needle" with $c = i$. It is now just a thin, branching structure.



The Julia set for the "Galaxy" with $c = -0.71 + 0.35i$

How can we get an idea of the *c* for which the Julia set still contains inner points? The Mandelbrot set provides information about this. The fact is that every basin of an attractive fixed point or attractive cycle of the Julia set contains at least one critical point, i.e. a point with $f'(z) = 0$. In the case of $f(z) = z^n + c$ this is the zero point. So, if the zero point remains finite during iteration, then there is at least one basin of an attractive cycle and thus a Julia set with inner points. If the zero point tends to infinity, then there is no basin and no attractive fixed point. Then the Julia set decays to "dust".

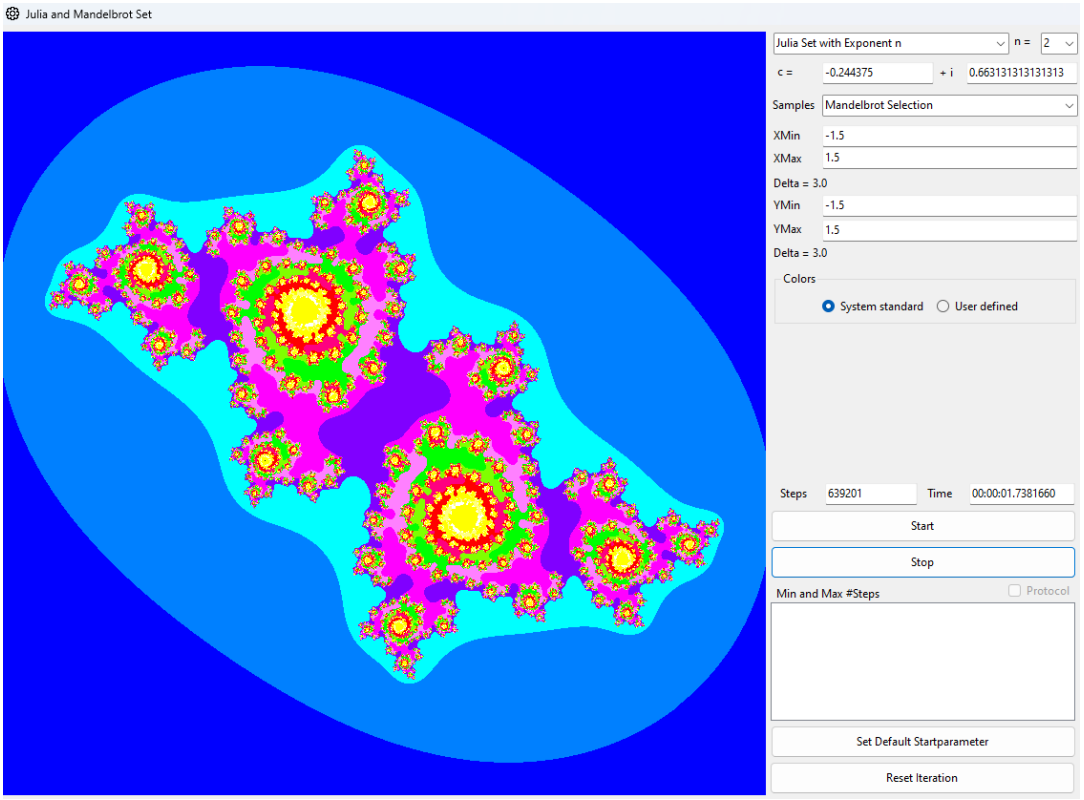
The Mandelbrot set is now the "map" of the $c \in \mathbb{C}$ and shows the behavior of the zero point during iteration. So, you choose a *c* as the parameter of the iteration, then choose the starting point $z_0 = 0$ and see whether it moves towards ∞ during the iteration. If not, colour the point black. If yes, colour it in a lighter colour if the divergence is slower. For more details and the criteria: see the mathematical documentation.



The Mandelbrot set

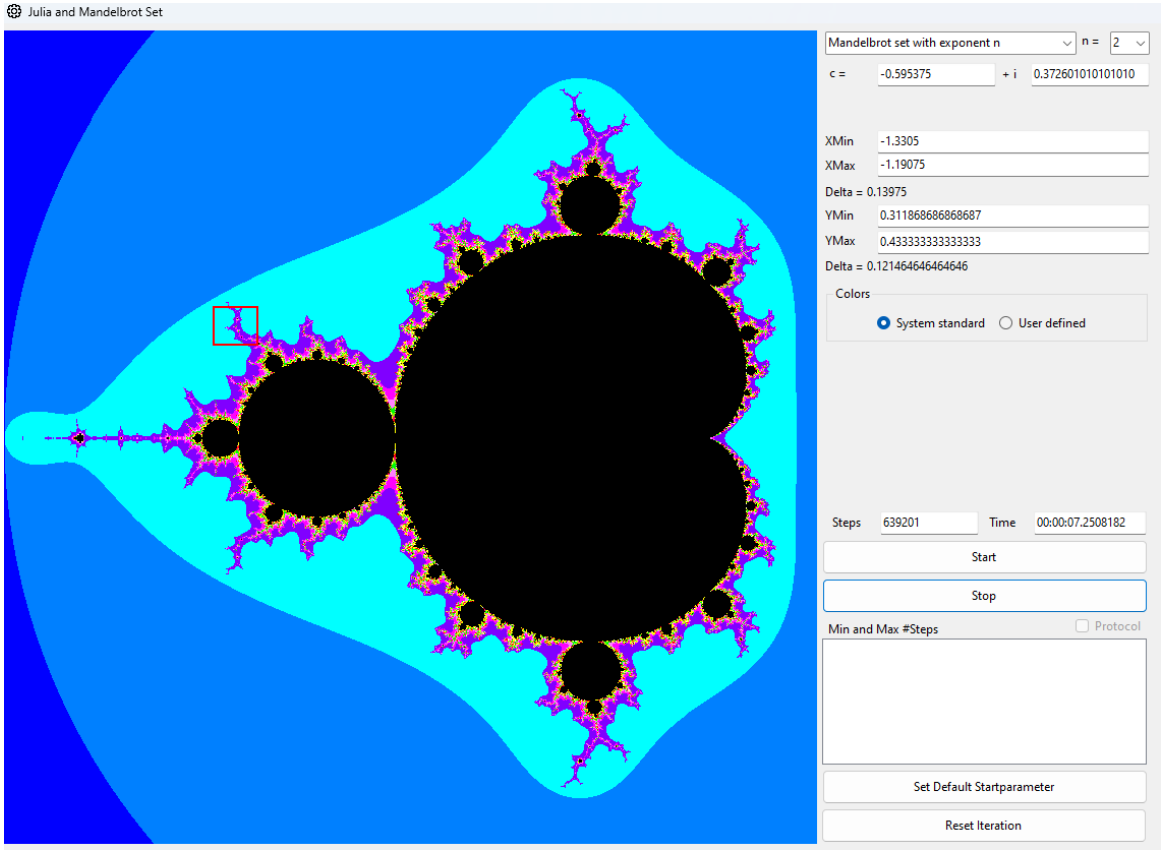
If you select a sub-area with the mouse button held down, the corresponding coordinates are displayed in the XMin, XMax, YMin, YMax fields. The value of *c* is then set according to the center of the selection.

If you now switch back to the Julia set via the combo box at the top right, the value of c is added to the list of examples as a "Mandelbrot selection". If you select this option, the Julia set associated with this *c* is generated:

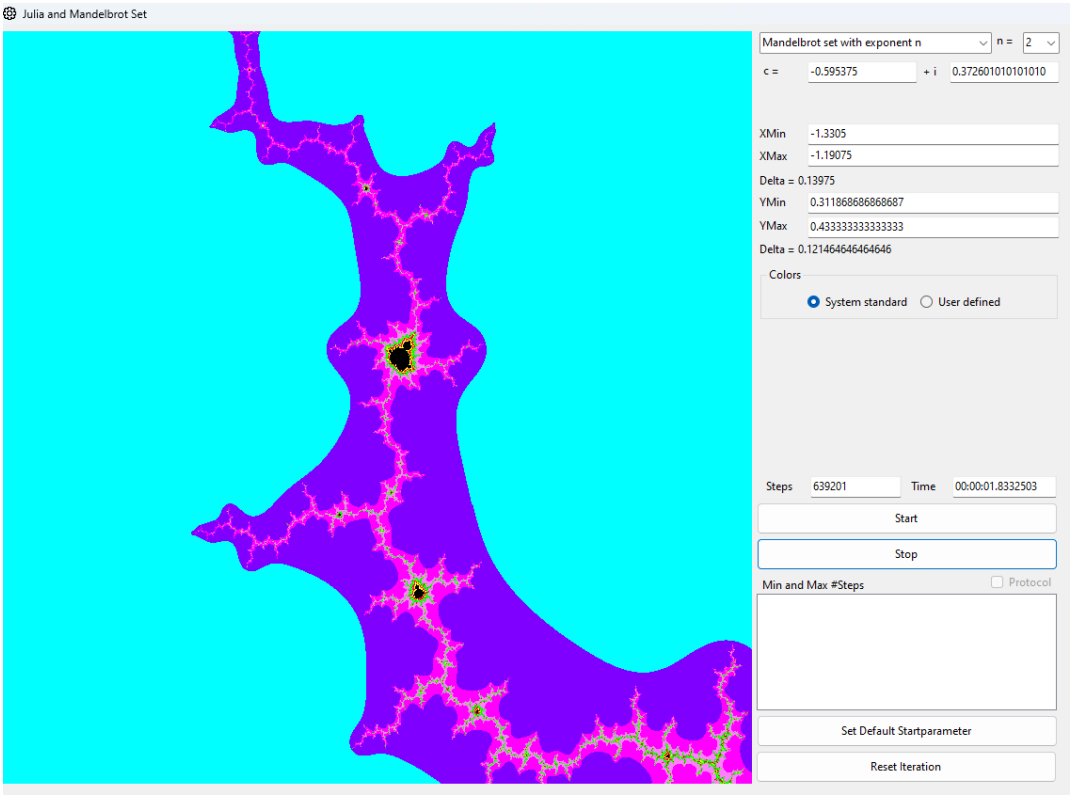


Julia set belonging to the previous c value

Like the Julia set, the Mandelbrot set also has fractal properties. This means that if you enlarge a section of the image, similar images appear again and again.

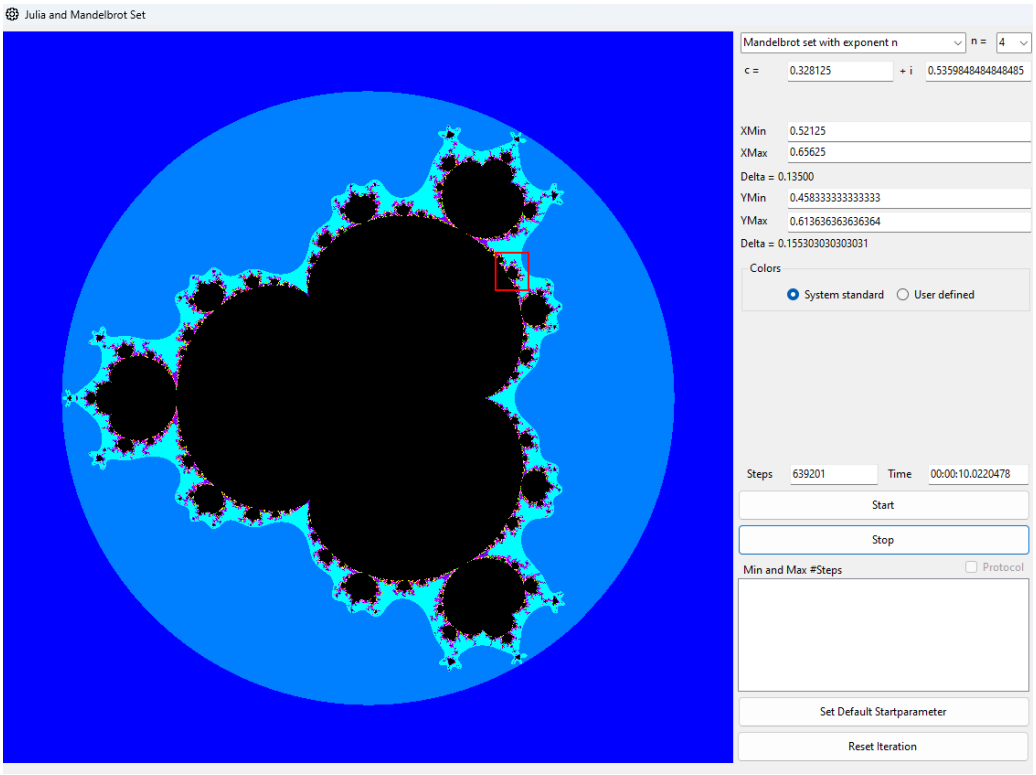


Once again, the Mandelbrot set with a selected small section.

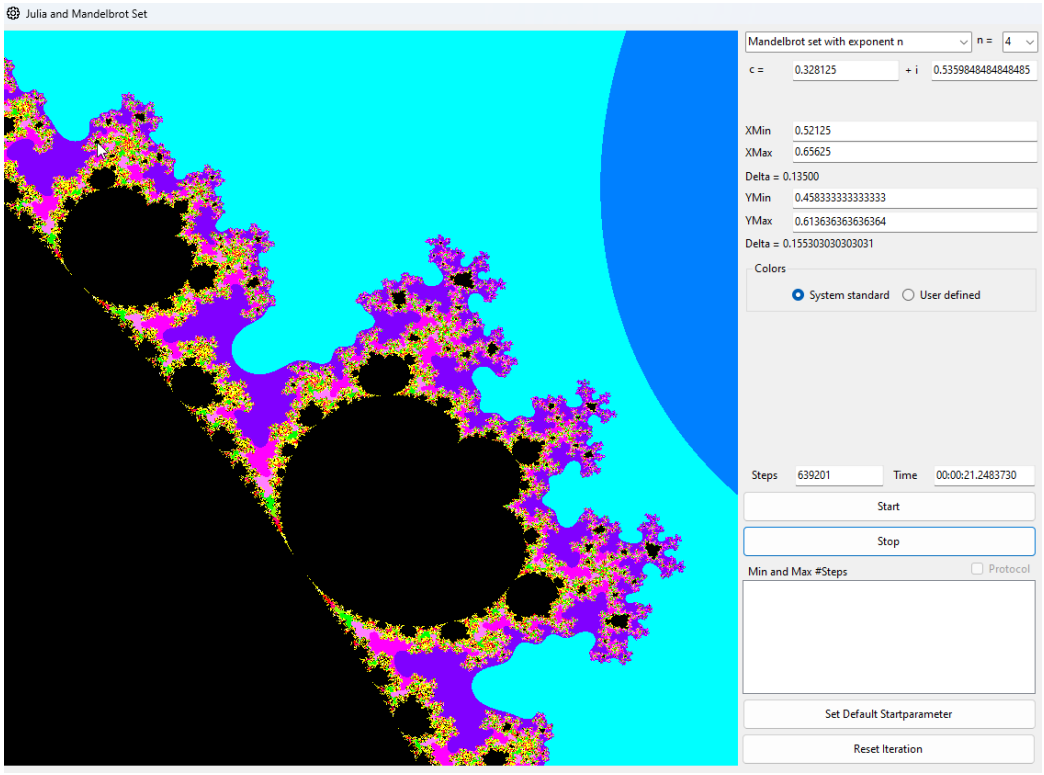


In this excerpt, specimens of the Mandelbrot set appears again

Concerning the choice of exponent: n is a natural number between 2 and 12. $n > 2$ do not provide any further fundamental insights. However, they can be used to generate interesting images. The rotational symmetry then increases according to the value of n. For the Julia set this is n-fold, for the Mandelbrot set (n-1)-fold.



Mandelbrot set for $n = 4$ and a selection made



The image of the selection made

7, 8, 9, 10, 11, 12

If a section is selected in the generated image with the mouse button pressed, its coordinates are displayed here. As the Mandelbrot set is a "map" for the respective Julia sets, the number *c* is also set as the centre of the section for a section from the Mandelbrot set. If you then switch to the "Julia set" option, this *c* is listed in the example list under "Mandelbrot selection" and you can create the corresponding Julia set.

13

Suitable colour levels were programmed as system defaults. The longer the starting point takes to converge to infinity, the lighter the colour. Starting points that remain at infinity during iteration are black.

The user can also define a colour themselves. Instead of the colour levels depending on the convergence speed, shades of the corresponding colour level are then displayed.

14

Controller for the composition of the user color from the primary colors red, green and blue.

15, 16

Display of the number of iteration steps and the elapsed time.

17

Start or continue the iteration.

18

Stop the iteration.

19

The output of a protocol can be requested by activating the checkbox. The respective minimum and maximum number of iteration steps are implemented. The generation of a protocol can reduce the iteration speed.

20

Presentation of the protocol.

21

The iteration is reset, and the user parameters are set to the default.

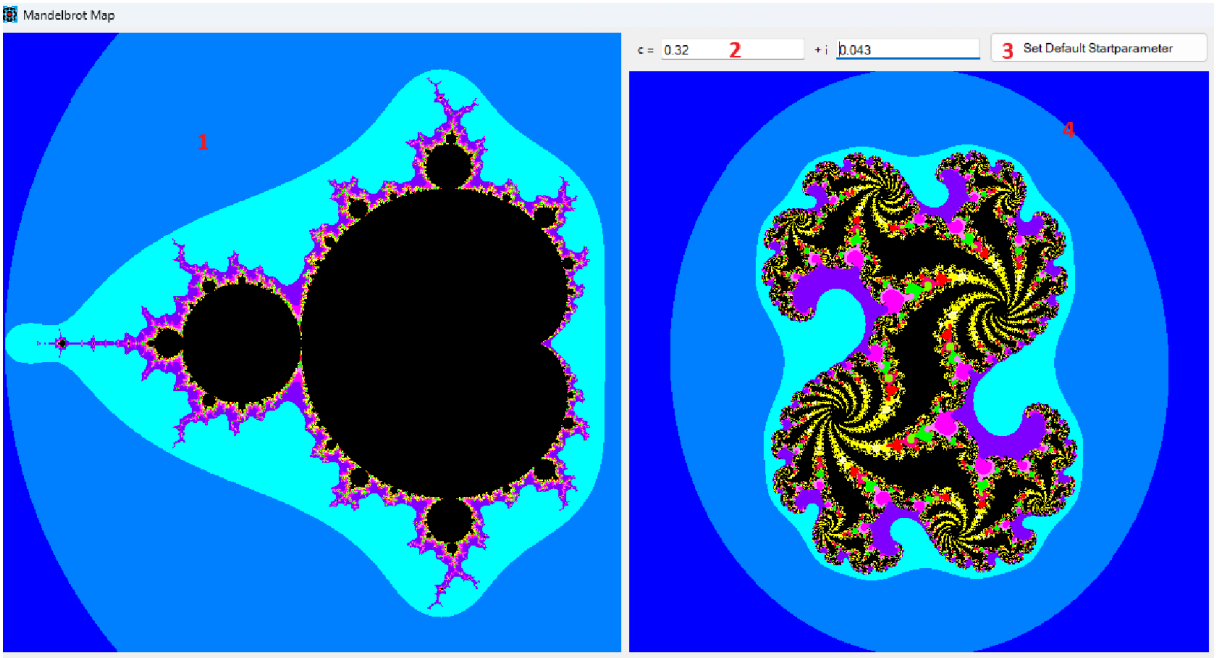
22

The iteration is reset.

5.3. Mandelbrot map

The menu "Complex Iteration - Mandelbrot Map" allows you to mark a point in the Mandelbrot set with the mouse so that you can see the corresponding Julia set directly on the right side.

The following window opens:



1

Representation of the Mandelbrot set. The mouse is positioned there with the left mouse button pressed and when the button is released, the corresponding Julia set is generated on the right

2

The complex constant c is displayed here, which belongs to the mouse position on the left-hand side.

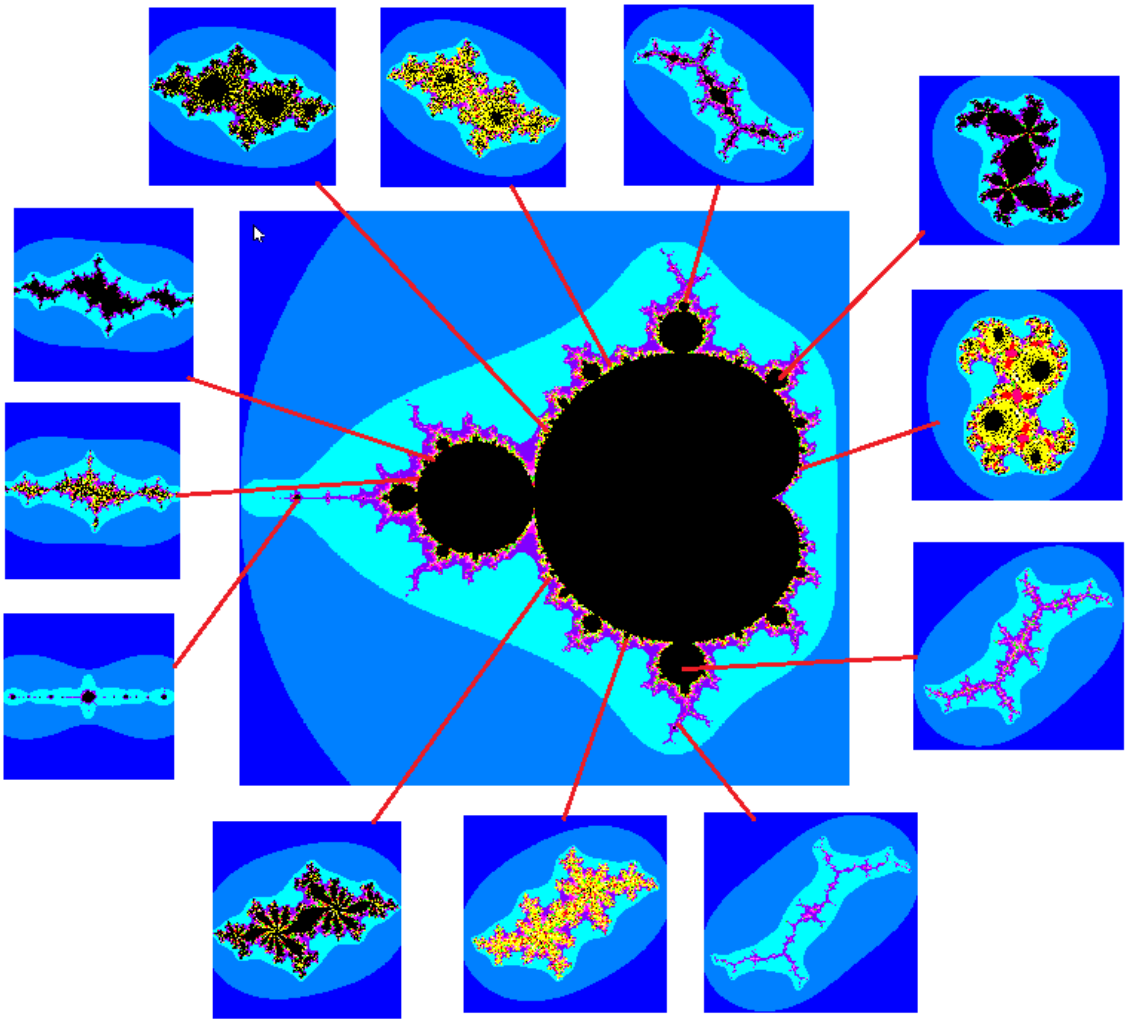
3

The default position of the mouse is set here (the image at the start of the window).

4

Display range for the Julia set.

Here are some examples:



6. Menu Mechanics

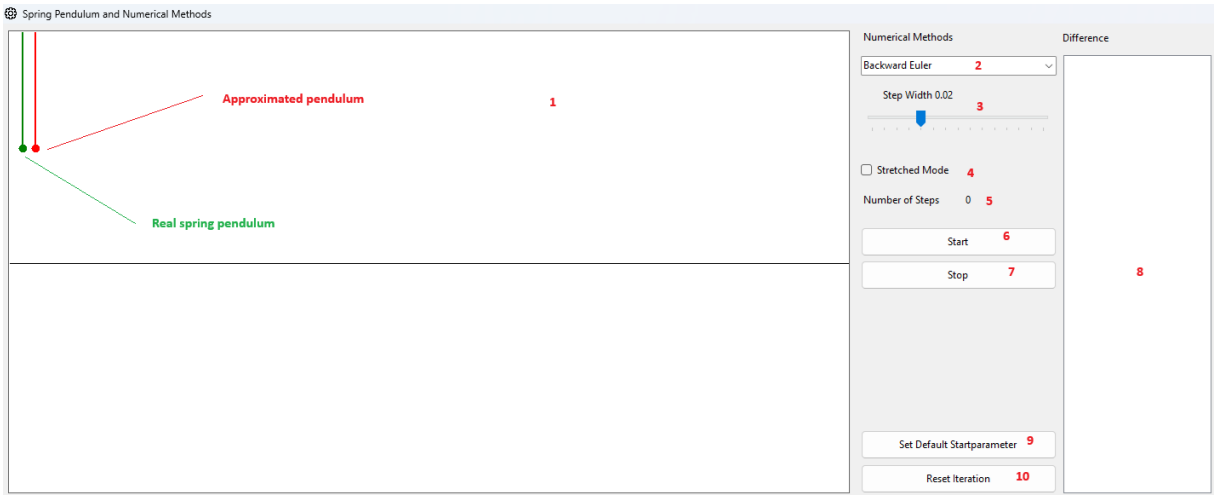
6.1. Numerical methods

The basis for these experiments is the classical spring pendulum, which oscillates here in a cosine oscillation. Its oscillation can now be compared with spring pendulums that do not oscillate according to cosine, but whose differential equations are approximated by numerical methods. These are simple methods which are elementarily accessible (except perhaps the Runge Kutta method). The following are available:

- Euler Implicit
- Euler Explicit
- Implicit midpoint rule
- Runge Kutta procedure level 4

These methods are described in detail in the mathematical documentation.

The following form is opened via the menu:



Window for investigating the numerical methods

The oscillations of the pendulums are displayed in the left-hand area. Green: The "real" spring pendulum with the closed solution of its differential equation, which oscillates here in a cosine oscillation. Red: The spring pendulum with a numerical solution of the corresponding differential equation.

The pendulums are in the zero position. If you start the simulation here, the pendulums will not swing. They must be moved to the starting position by holding down the left mouse button.

Horizontal: You can see the time axis and the zero position of the pendulum. In the y-direction, the pendulum position is in the interval [-1, 1].

1

Area for the graphical representation

2

The numerical method for the red pendulum is selected here. As a control, the red pendulum can also be selected as a "real spring pendulum".

3

The real, green pendulum increments the time variable by 0.1 in each step. The red pendulum can subdivide this interval further according to the step size selected in 2. If this is 0.02, for example, as in the picture above, then the red pendulum performs 5 iteration steps with a step size of 0.02 before its position is displayed in the picture. The green pendulum is then displayed at the same time with an increment of 0.1. This means that the pendulums are synchronized.

4

If you activate the "stretched display" option, then both pendulums swing with the same step size for the time parameter, e.g. both pendulums with a step size of 0.02. This results in a stretched display for both pendulums and you can see the deviations between the green and red pendulum better.

5

The number of steps of the green pendulum on the time axis is displayed here.

6

The pendulums are started here. If the oscillation is interrupted, it can be continued with this button.

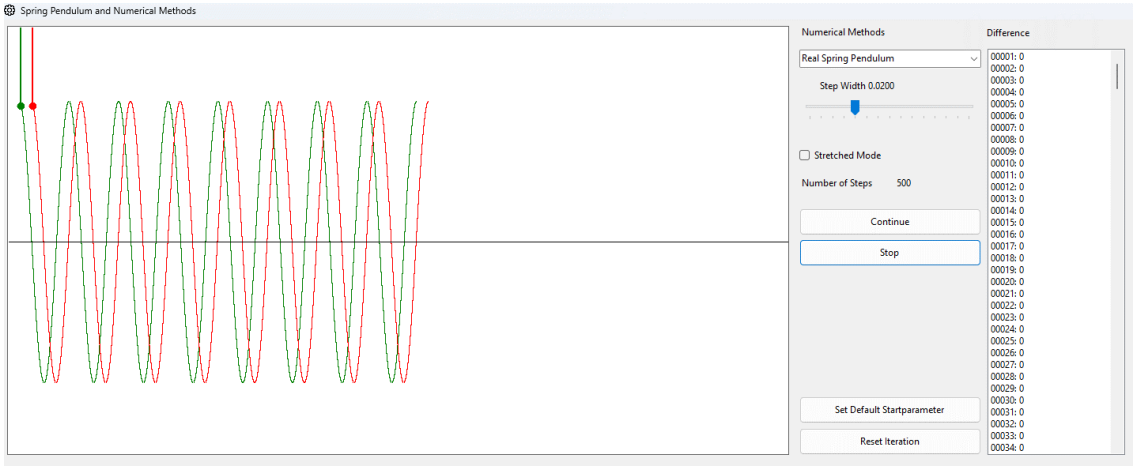
7

The oscillation of the pendulum is interrupted here.

8

The difference between the y-positions of the green and red pendulum is recorded here.

Here are some examples.



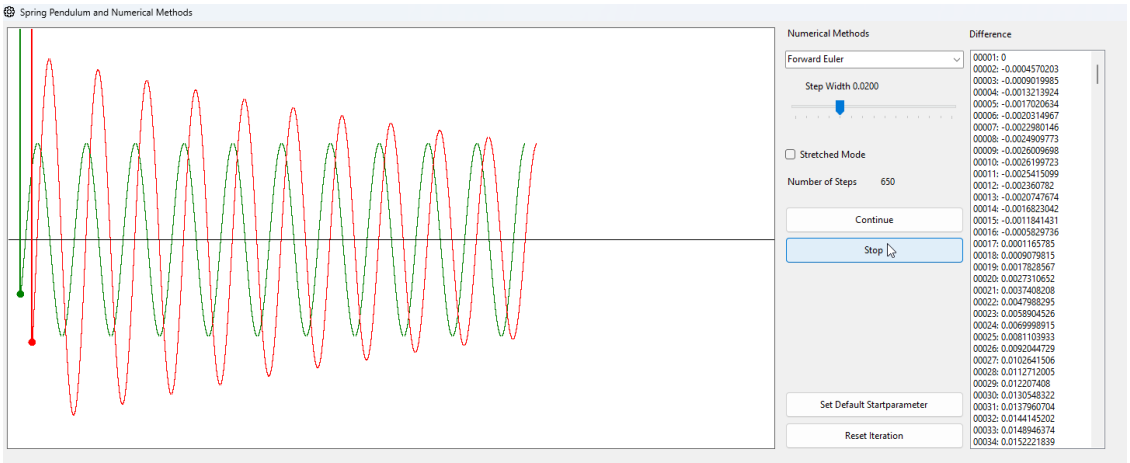
The red pendulum also swings like a real pendulum. The difference on the right is zero.

9

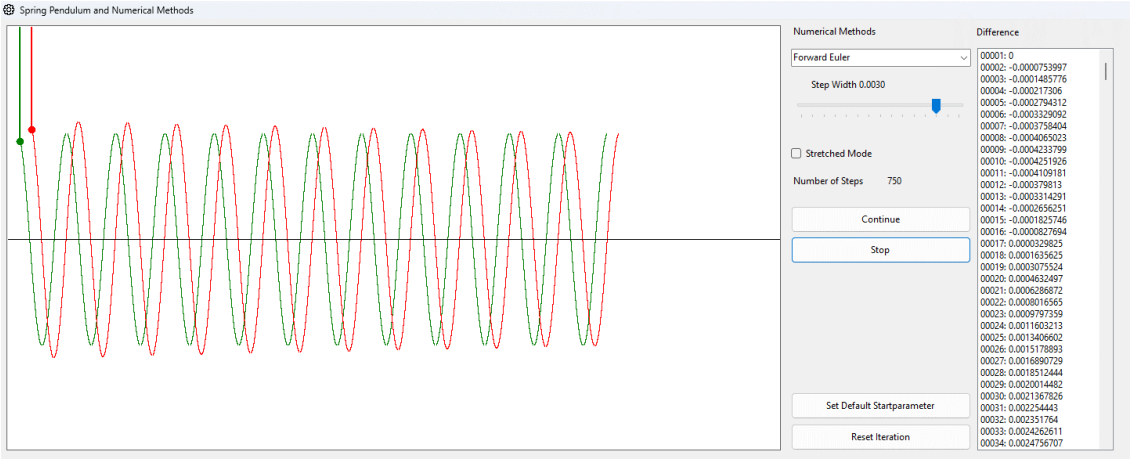
The iteration is reset here, and the pendulums are moved to the standard starting position.

10

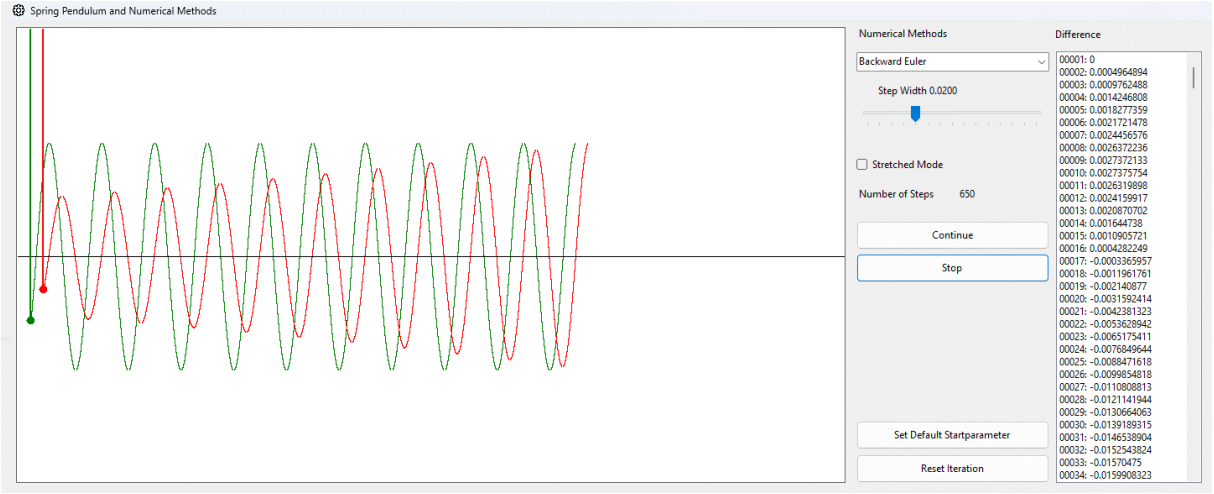
Only the iteration is reset here.



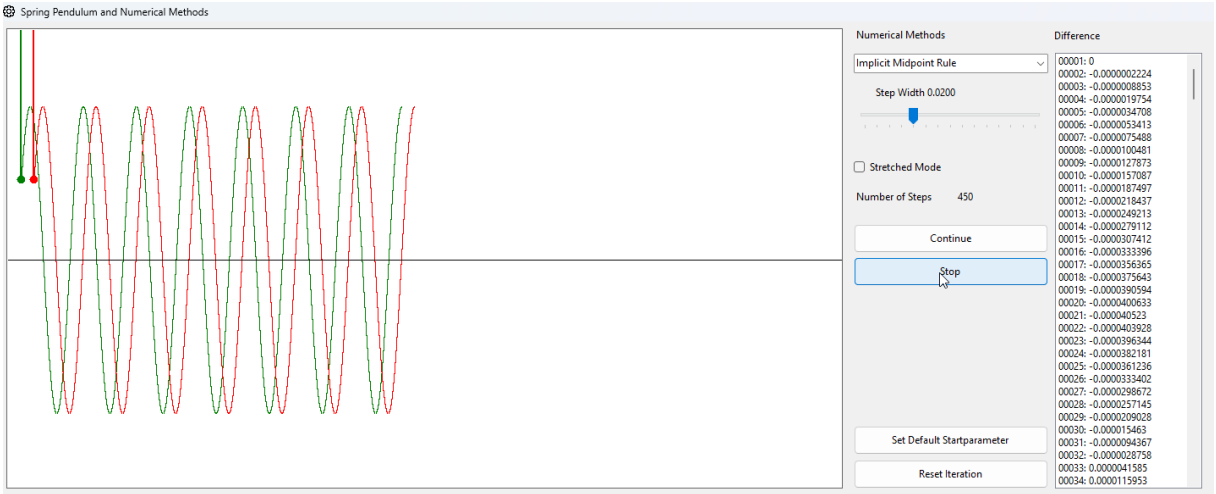
The red pendulum is approximated by the "Forward Euler" method (see math. doc.)



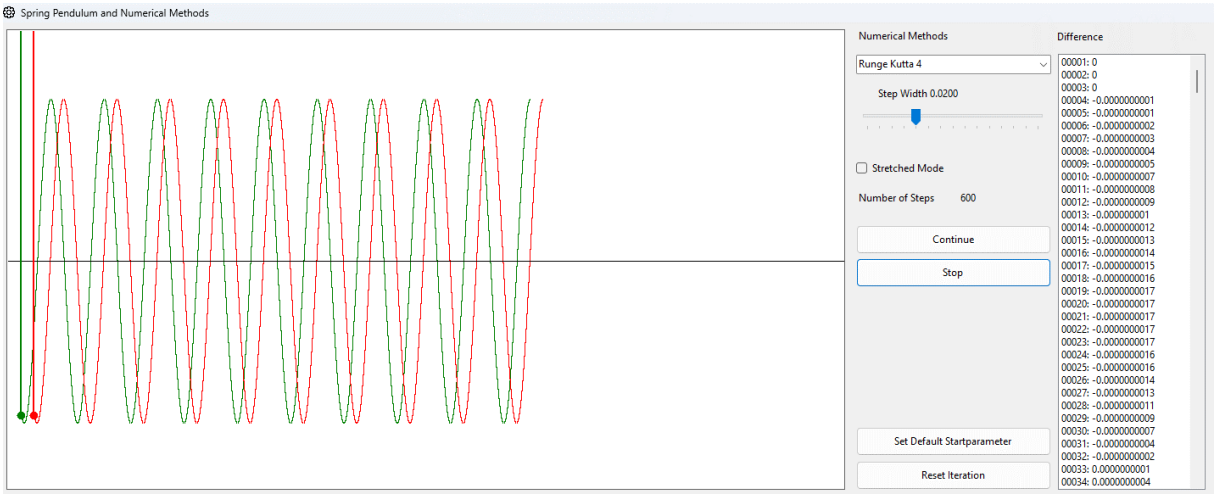
The same situation, but with a smaller increment for the red pendulum:
It takes longer for the tracks to diverge from each other



Here the red pendulum swings approximately using the "Backward Euler" method



The "implicit midpoint rule" provides a good approximation for the red pendulum.
However, you can see on the right that the difference between the positions is not zero



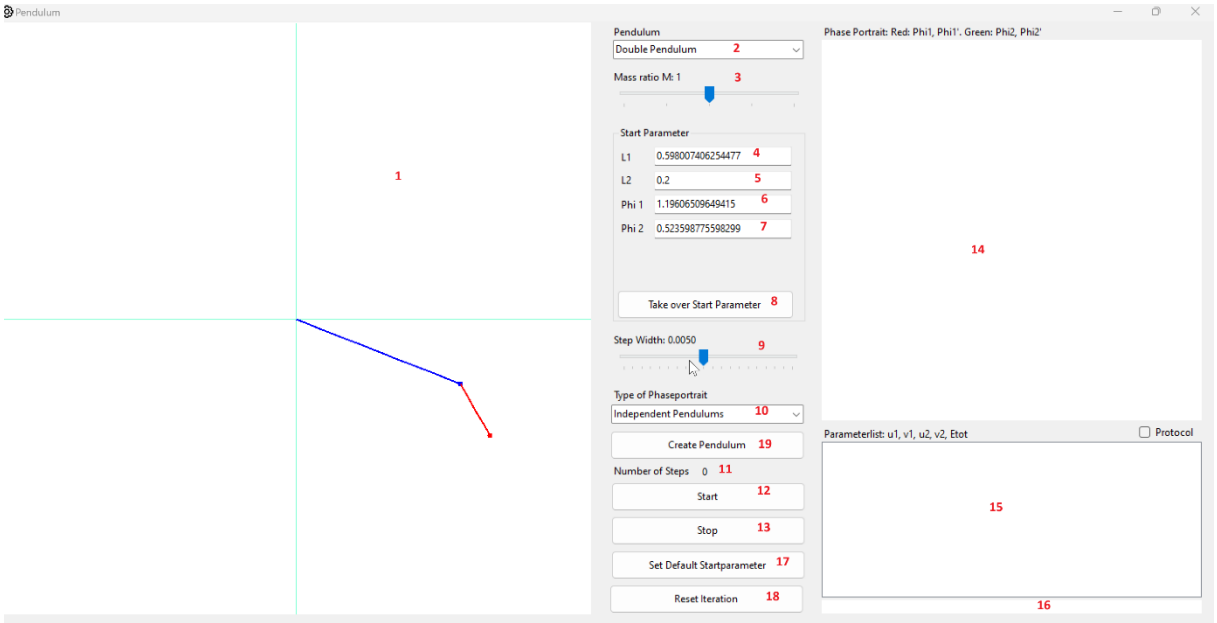
The "Runge Kutta" 4th order method performs best.
See the small differences on the right.

6.2. Pendulum

The following coupled pendulums are offered for examination here:

- Double pendulum
- Oscillating spring pendulum
- Horizontal vibrating pendulum

The "Mechanics - Pendulum" menu opens the following window:



Window for examining the different pendulums - here the double pendulum

1

The movement of the pendulum is shown here.

2

Selection of the pendulum.

3

Depending on the pendulum, a further parameter can be set here.

4, 5, 6, 7

The specific start parameters are displayed here or can be entered. Depending on the pendulum, this can be up to 6 different parameters.

8

This button can be used to transfer entered start parameters to the pendulum.

9

Step size of the Runge Kutta method.

10

Selection of the type of phase portrait. There is a choice:

- Both pendulums independent of each other
- Torus or cylinder (depending on the pendulum)
- Poincaré cut

11

Display of the number of steps in the iteration.

12

Start of the iteration.

13

Stop the iteration.

14

Display of the phase portrait.

15

Log of the relevant parameters and display of the current energy relative to the total energy at the start. The log can be activated by the protocol checkbox.

16

These pendulums react very sensitively to small changes in the start value and therefore also to small errors in the iteration. The approximated movement is certainly no longer realistic after a few steps. In order to at least partially control this, the total energy is shown in this bar. This should remain constant. See below.

17

The iteration is reset here, and the start parameters are set to default.

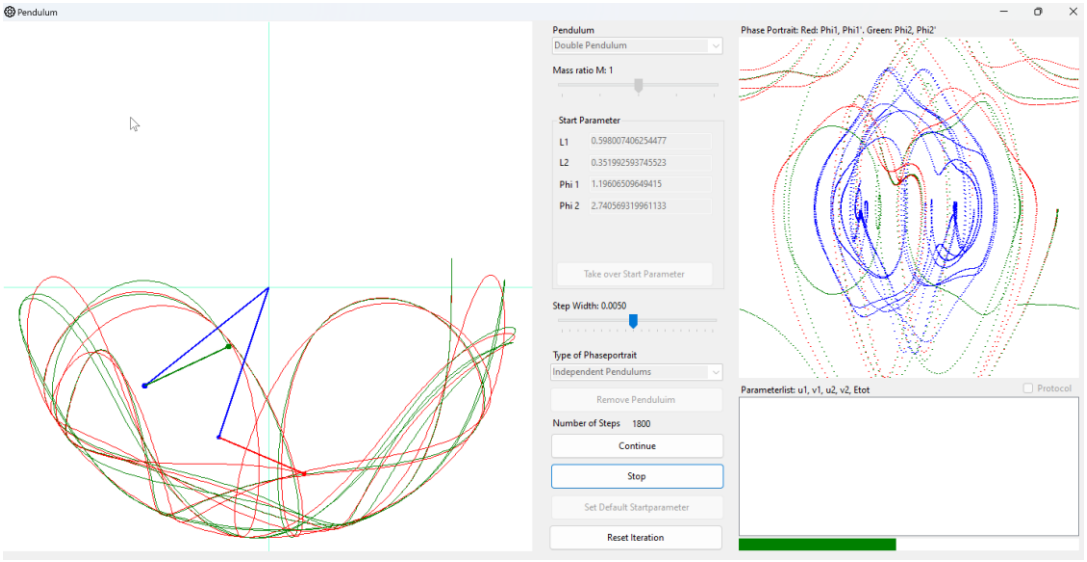
18

Only the iteration is reset here.

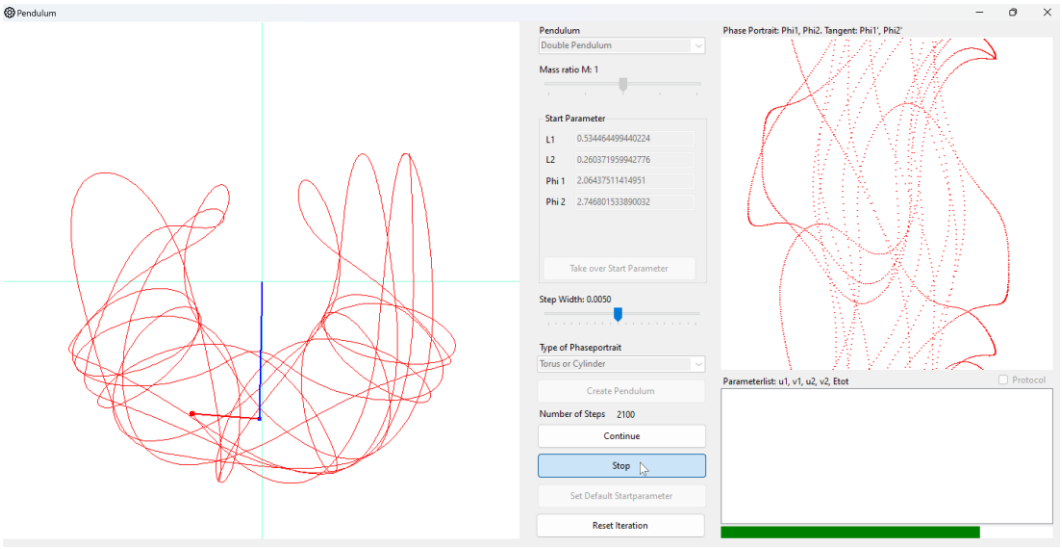
19

A "shadow pendulum" can be added here, the starting position of which deviates minimally from the main pendulum. Depending on the strength of the oscillation, the sensitivity then becomes visible.

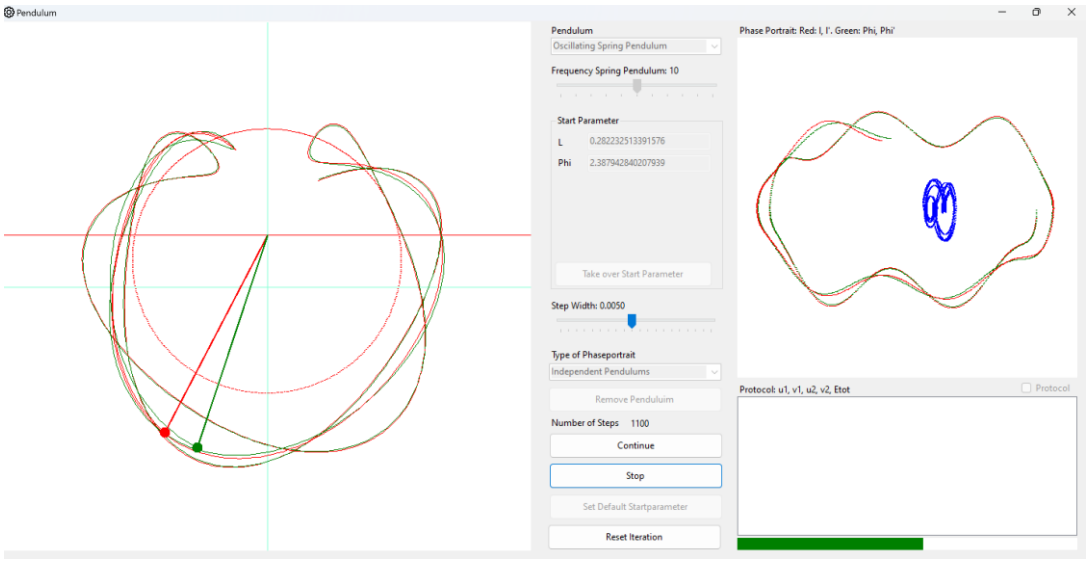
The Runge Kutta method (and also better methods) only show the "real" movement of the coupled pendulum over a few steps. Subsequently, the pendulum reacts too sensitively to the inaccuracies of the iteration. For this reason, the current energy is displayed if it deviates from the total energy at the start by more than ±10%. If the deviation is below this value, the corresponding bar of the energy display appears green. If the current energy is too low, the bar appears purple. If it is too high, it appears red.



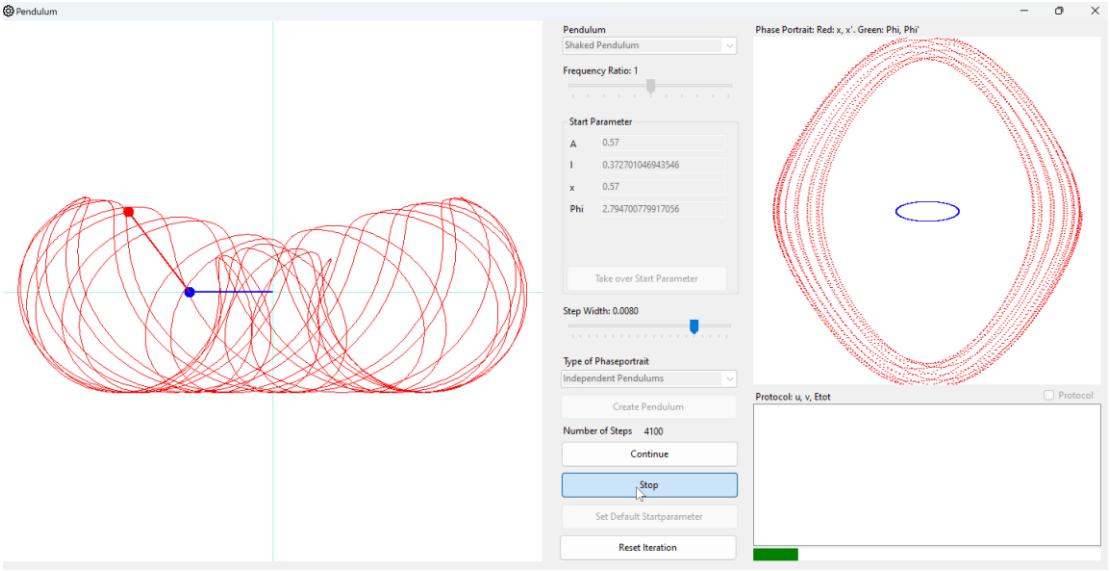
Movement of the double pendulum including its “Shadow”



Representation of the movement on the torus



Movement of the oscillating spring pendulum including its “Shadow”



Movement of the shaked pendulum

6.1. Universes

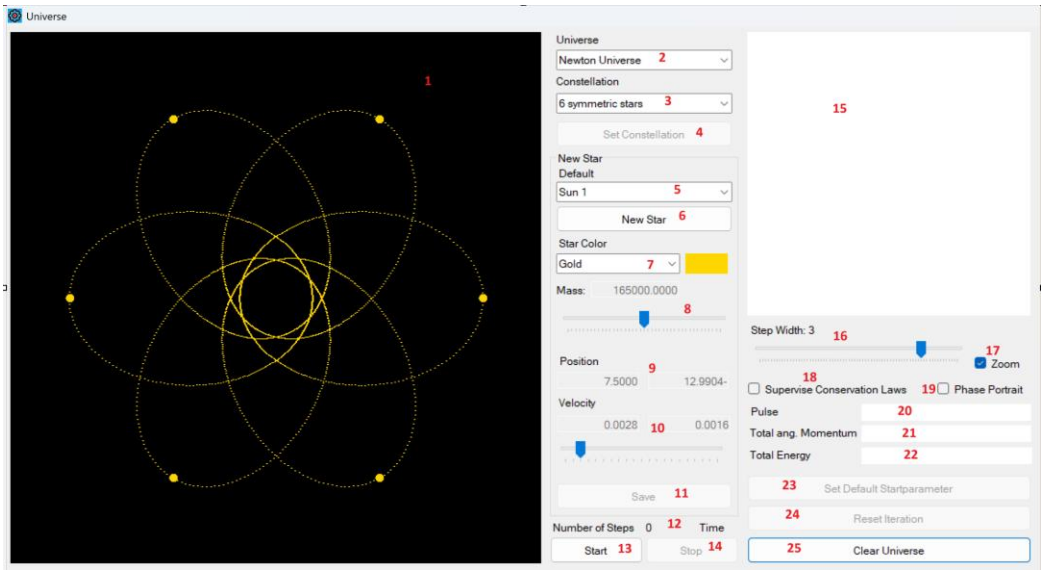
The simulator enables the simulation of star constellations in different universes. A universe is defined by the law of motion that applies in it and by the ranges in which the parameters of the stars move. We use the terms star and planet synonymously depending on the context.

In Newton's universe, the well-known Newtonian laws of force apply, and the gravitational force is determined by the gravitational constant. This universe is suitable for simulations in our solar system.

Alternatively, a "normalized universe" is also implemented. Newton's laws of force also apply here, but the star parameters are normalized so that the mass of the star is a few kilograms and the gravitational constant = 1. In this universe, stable periodic orbits of three stars can be investigated.

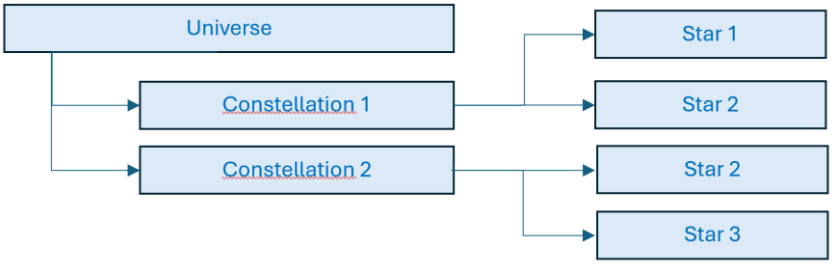
Further universes can be implemented in the future. However, the program expects that the laws of conservation apply and that the laws of motion are independent of the position of the coordinate system. This means that only the gravitational force can be manipulated, and in such a way that it can be derived from a potential field.

The "Mechanics - Universe" menu opens the following window:



Window for simulations in different universes

The structure of a universe is outlined in preparation for operating this window.



Structure of a universe

A universe contains predefined constellations of stars. One such constellation in Newton's universe is our planetary system, for example. Another constellation is the inner planets. This means that a planet can belong to different constellations at the same time, as in the case of Venus above.

When using the "Simulator", you first select the universe in which you want to work. It is the dynamic system and empty at the start. You then select a constellation. This can be transferred to the universe as a whole or you can also transfer individual stars from the constellation. These are listed for each constellation in a "Default" combo box. If you add individual stars, you can change their parameters (mass, position and speed) and then save the star in the universe.

1

Diagram showing the universe and the stars placed in it. The image above shows six symmetrically arranged suns.

2

Selection of the universe.

3

Selection of constellations. The predefined constellations belonging to the universe are listed in the combo box.

4

Button for accepting an entire constellation. After the transfer, the button is deactivated. This means that a constellation can only be transferred to the universe once.

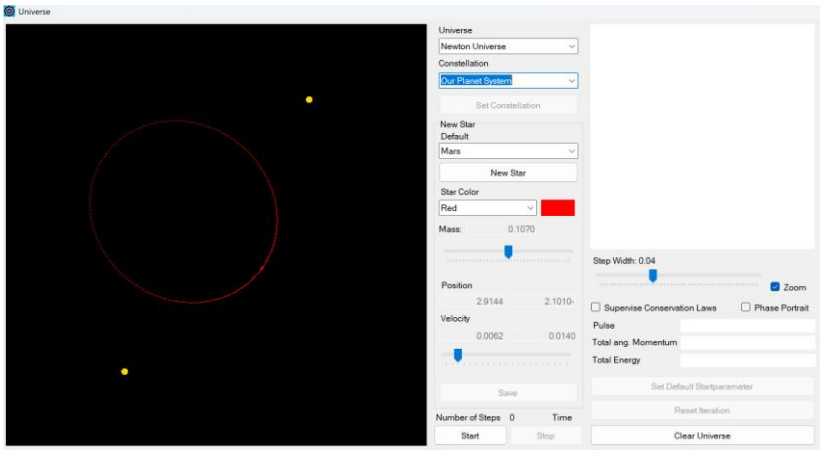
When a star is placed (either manually or as part of an entire constellation), its orbit is outlined relative to the common centre of gravity of all the other stars. This will not be the actual orbit of the star. However, it serves as a reference point for the user when placing a star. The same also applies to the stars of a constellation when it is adopted. The image above shows the corresponding orbits of the suns around the common centre of gravity.

5

Area **5 - 11** contains the elements that are used to place and manipulate a new star. The available stars (or planets) of the selected constellation are listed in combo box **5.**

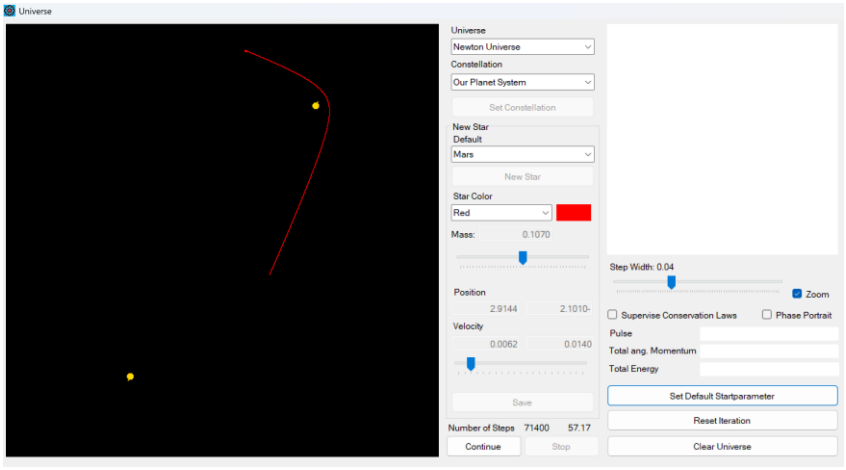
6

Once you have selected a star in combo box **5**, you can add it to the universe using the "New star" button. The "New star" button then becomes "Cancel" if you want to cancel the action. When you accept a new star, its orbit is outlined relative to the centre of gravity of all other existing stars.



Two suns and one Mars

Above, two suns have been placed in the universe. These have a starting speed of zero, i.e. no orbit is shown. Mars was then placed. Its orbit around the common centre of gravity of the system is shown.



The actual orbit of Mars then looks different. In addition, the suns are moving towards each other.

7

Here you can change the star colour. This makes it easier to distinguish the stars in the universe.

8

When a star is transferred, it has a predefined mass as standard. This is displayed in area **8** and the shift register is in the middle. The user can now change the star mass using the shift register, relative to the predefined standard mass. Changes of around $\pm 40\%$ are possible. In the case of Newton's universe, the unit of mass is the Earth's mass. In the standardized universe, it is 1 kg.

9

The position of the star is displayed here. This can be changed by holding down the left mouse button. In Newton's universe, the unit of length is the astronomical unit, i.e. the distance between the earth and the sun. In the standardized universe, the unit is 1 metre.

10

This area is used to display and manipulate the speed. Each star has a predefined speed. This can be multiplied by a factor in [0, 2] using the shift register. However, its direction remains unchanged.

11

Once you have adjusted the parameters of the new star, it can be definitively added to the universe by clicking button **11.**

12

The number of iteration steps and the elapsed time are displayed here.

13

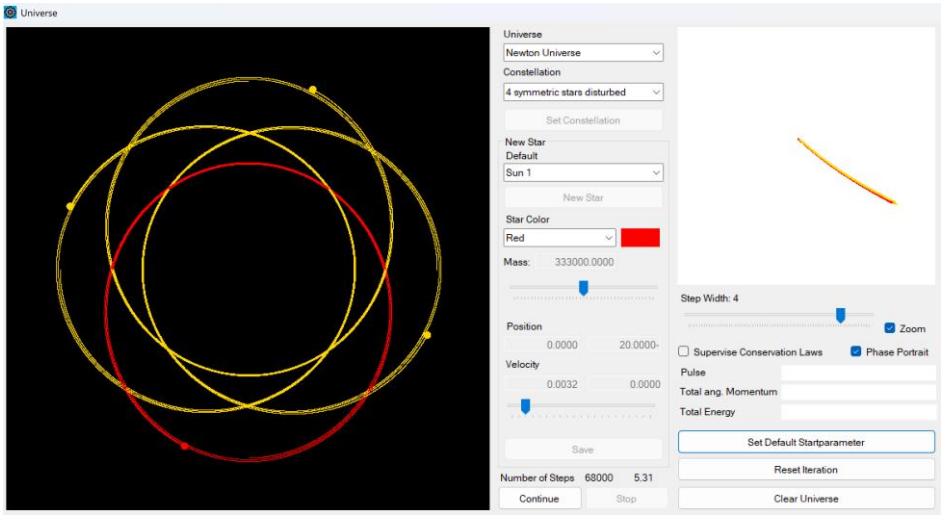
This is where the iteration is started or - if it was stopped after starting - continued.

14

The iteration is interrupted here. It is only stopped completely when the iteration is reset.

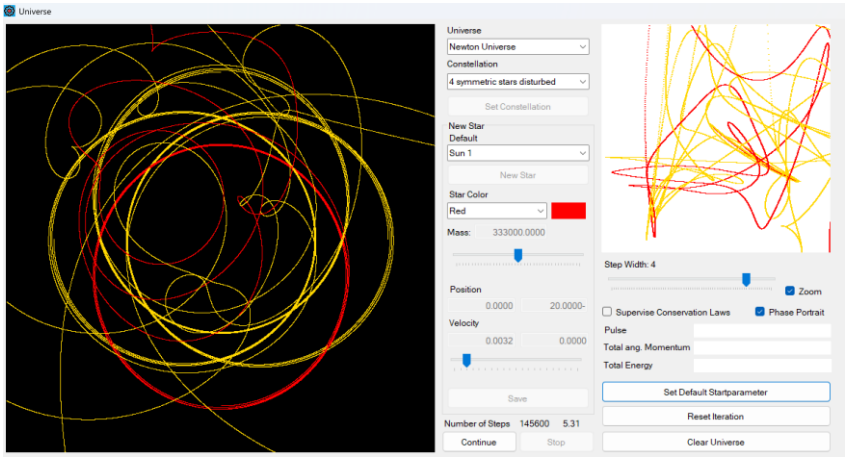
15

This area is used to display a phase portrait if option **19** is activated. The amount of the star's distance from the common center of gravity of all stars is displayed horizontally and the amount of its velocity is displayed vertically.



Four symmetrically arranged suns and their movement including phase portrait

The system is slightly disturbed: the distance of the red sun from the common centre of gravity is 0.00000001 units smaller than that of the others. While the movement of the suns appears stable at first, it becomes unbalanced after four orbits:



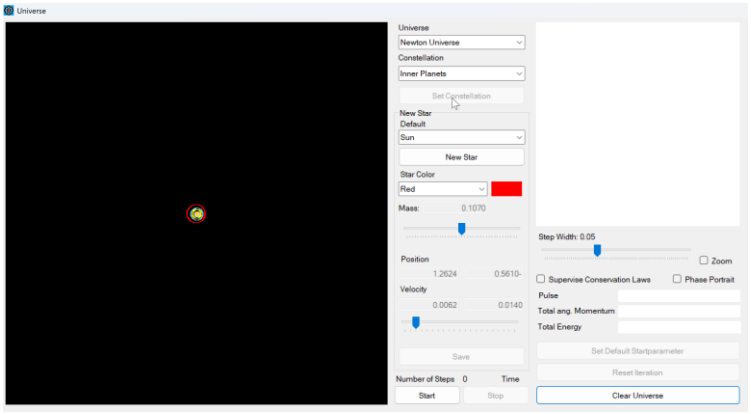
The system becomes unstable after four cycles

16

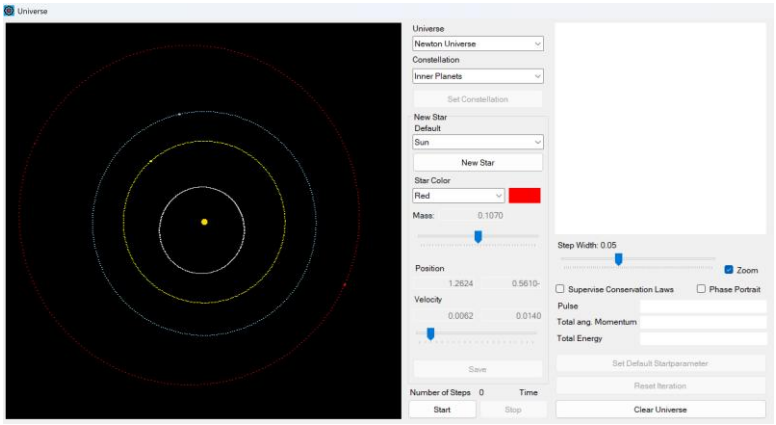
The step size of the iteration can be adjusted here. A small step size leads to a more precise but slower iteration.

17

The zoom is activated here. The common centre of gravity of all stars is always selected as the origin of the coordinate system. This is continuously recalculated, even if the star parameters are changed. As a result, the entire universe does not fly out of the user's field of vision at a constant speed during iteration. If the zoom is active, the display in the universe is also zoomed in such a way that the visibility of the stars is optimized.



Inner planets without zoom



Inner planets with zoom

18

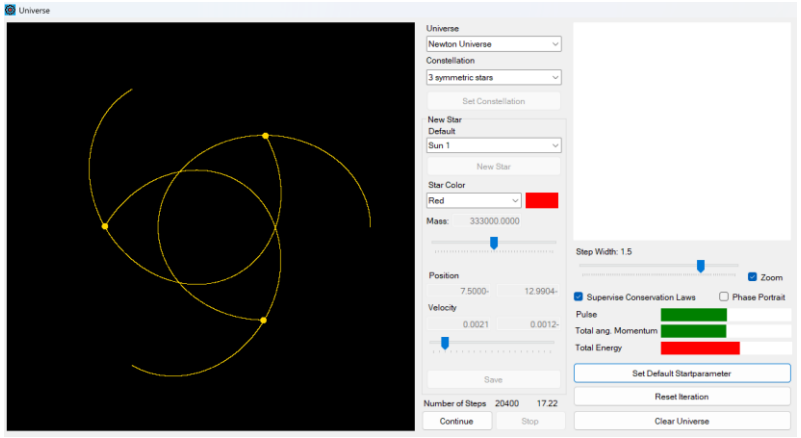
Monitoring of the maintenance records is activated here.

19

The phase portrait is activated. For performance reasons, this is not the case by default.

20 - 22

Monitoring of total impulse, total angular momentum and total energy of the system.



The orbit of three symmetrical stars around the common centre of gravity

Apparently, the total energy increases near the centre of gravity (red bar) and then decreases again to the green area. The reason for this effect has not yet been investigated in detail. Apparently, the loss of potential energy does not compensate for the increase in kinetic energy in the simulation.

23

The iteration is stopped, and the system is reset to the original data.

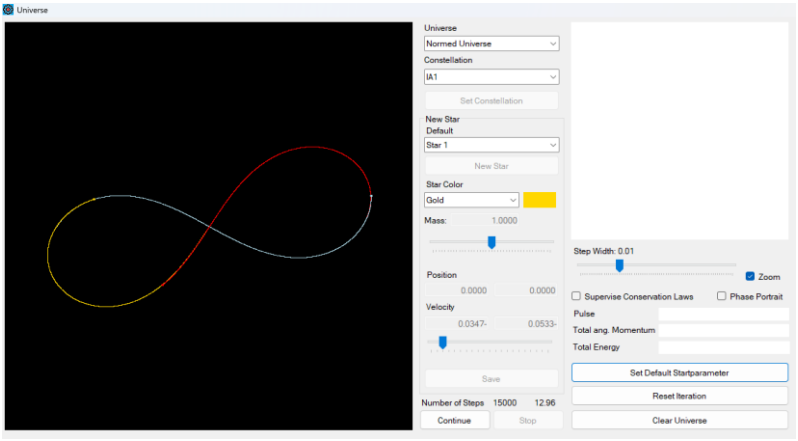
24

The iteration is stopped, and the system is reset to the data defined by the user.

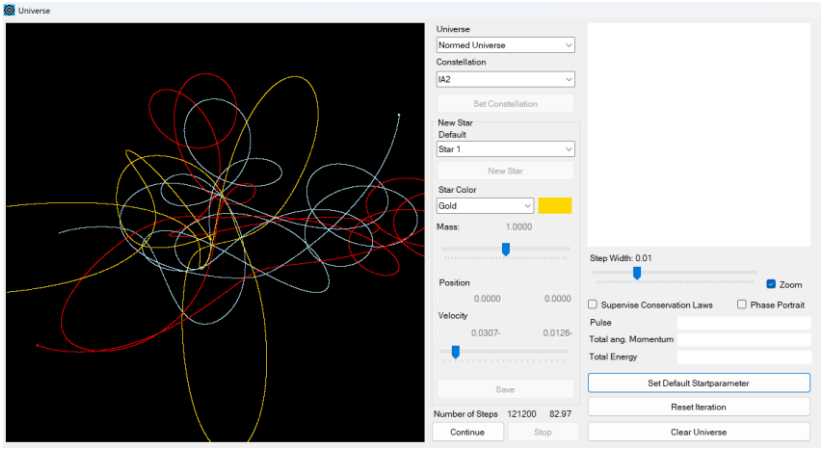
25

The universe is emptied.

The normalized universe should be used to investigate stable periodic orbits. Some corresponding constellations are available and documented in the mathematical documentation. However, as the Runge-Kutta method is relatively imprecise, this is only possible in exceptional cases and with low periods.



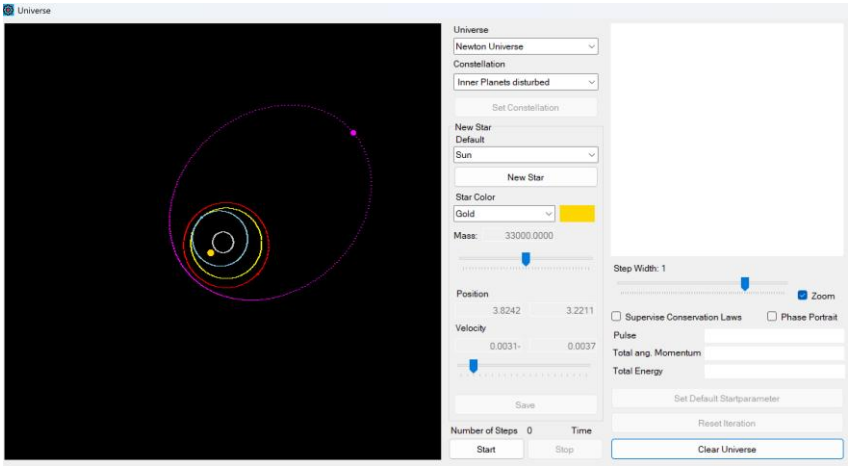
The periodically stable orbit IA1



The constellation IA2 should also be periodically stable. However, the Runge Kutta method is too imprecise.

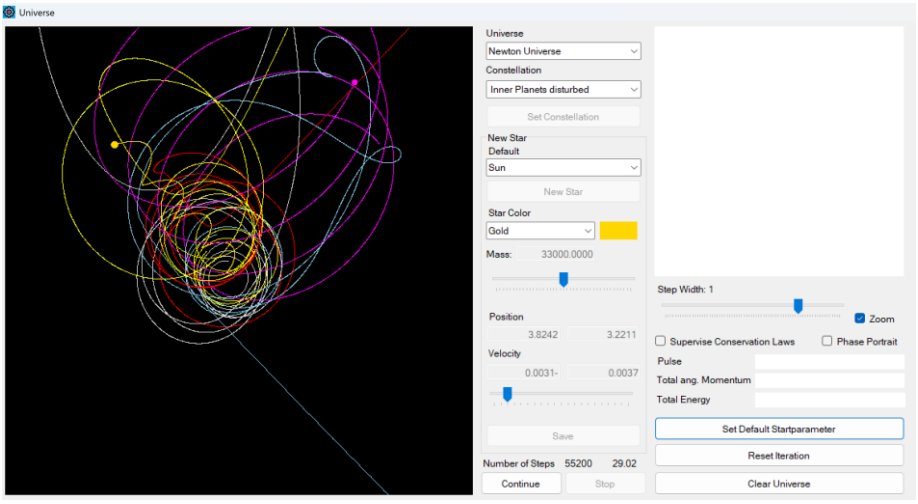
The fact that the Runge Kutta method is inaccurate combined with the fact that the N-body problem for$N > 2$ is generally chaotic and sensitive to perturbations should not be forgotten when studying stellar systems. What we see in the "simulator" is merely a numerical artifact of the real system.

Nevertheless, interesting effects can be investigated. Here is an example if Jupiter were heavier (about 1/10 of the mass of the sun) and came too close to the inner planets. These would then be "cleared out" of the solar system.



The constellation "Inner planets disturbed"

As always, the orbits are sketched which a planet would describe if the mass of the other planets and the sun were concentrated in its centre of gravity. You can also see above that this centre of gravity is slightly outside the sun. It is no longer at the centre of the orbits.



Jupiter has cleared out all the inner planets. You can see the orbit of the Earth (blue). In addition, (superheavy) Jupiter is shaking the sun.