Manual Simulator

User manual for the program "Simulator". Its mathematical background is described in the document "Dynamic Systems".

Version 5.0 -2024/05/01 – Ph.D. Hermann Biner

Inhalt

1. Introduction

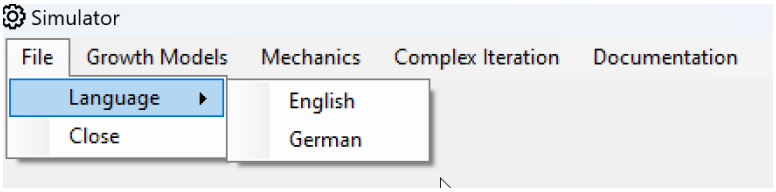
The “Simulator” supports the simulation of simple dynamic systems. The mathematical background and the concepts of the implementation are described in the document “Dynamic Systems”. There are proposals for further development of the “Simulator” and there are mathematical exercises.

The code of the “Simulator” is published on GitHub and available as open source. It is written in VB.NET and in detail commented. The development environment is the Community version of Microsoft Visual Studio 2022. It can be downloaded for free, and its installation is simple. The needed version is at least 17.9. The base for this version is the Microsoft .NET Framework 8.0.

This manual describes the use of the “Simulator” and presents examples for its application.

2. Main Menu

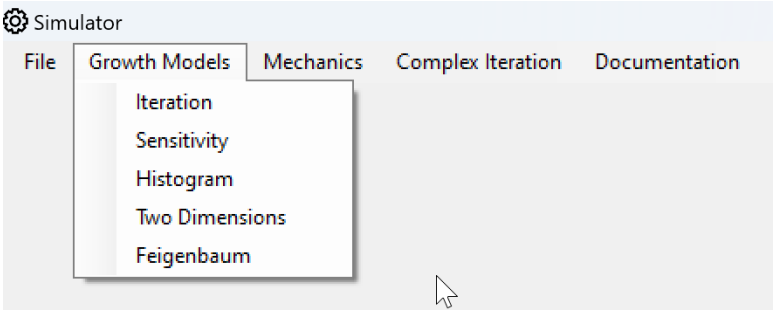
The menu “File” allows the selection of the language:



The section "Growth Models" supports different experiments, that are based on the iteration of unimodal functions.

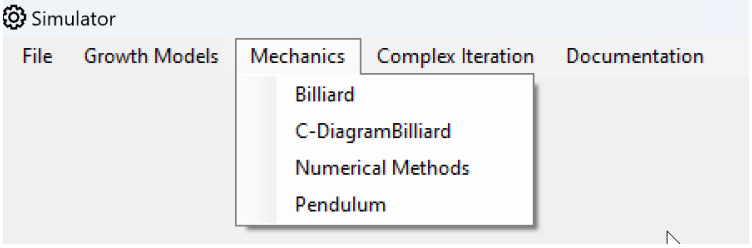
Implemented are:

- Tent map
- Logistic Growth
- Iteration of the Parabola



The submenus offer different functions, especially the examination of the chaotic case. They are described later in this document.

The section “Mechanics” contains different mechanical systems that can be analysed.



In this version 5.0 of the “Simulator” are different types of billiard implemented:

- Elliptic billiard
- Billard in the stadium
- Oval billiard

The form of the billiard table is everywhere defined by a parameter C. In case of the elliptic billiard, C is the ratio of the axis of the ellipse. As analogion to the Feigenbaum diagram in case of unimodal functions, one can create a C-diagram that shows the behaviour of the iteration parameters in dependence of C.

In addition, one can examinate different numerical methods to solve ordinary differential equations with the example of the spring pendulum:

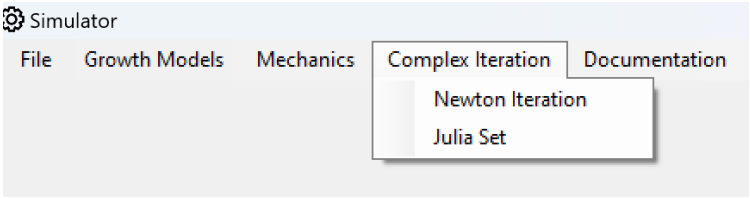
- Forward Euler
- Backward Euler
- Implicit midpoint rule
- Runge Kutta

The menu “pendulum” offers different coupled pendulums to be investigated:

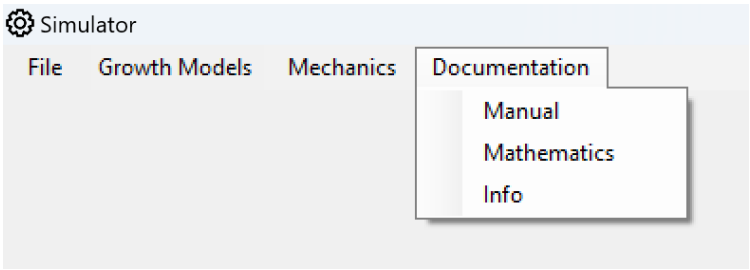
- Double pendulum
- Oscillating spring pendulum
- Horizontally shaked spring pendulum

The menu “Complex Iteration” offers the possibility of the simulation of iterations in the complex plane. Available are:

- Newton Iteration (especially in case of complex unit roots)
- Generation of Julia sets and the according Mandelbrot set



The menu “documentation” shows all information, that is needed to understand the use of the “Simulator”.

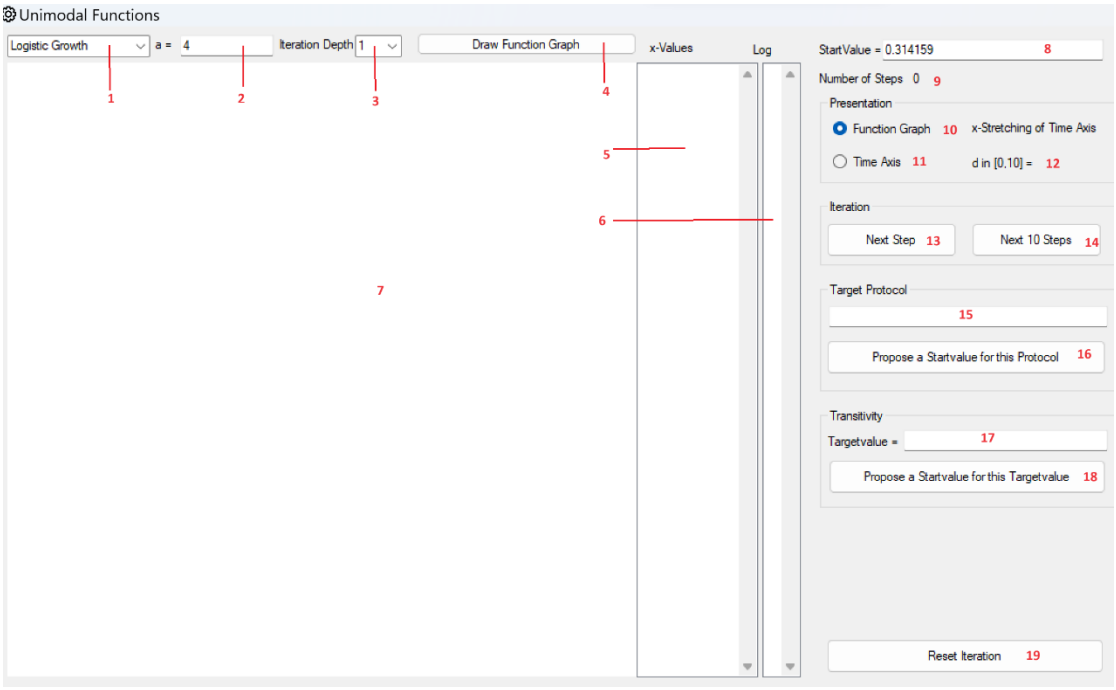


This manual is available in English or German, depending on the chosen language. The mathematical documentation, that is as well the background for the implementation, is available in German and English as well.

3. Growth Models (unimodal Functions)

3.1. Iteration in the real interval

The menu “Growth models – Iteration” opens the following window:



Window for the analysis of the iteration of unimodal functions

1

Here one chooses the type of iteration. Available are:

- Tent Map
- Logistic Growth
- Iteration of the Parabola

These iterations are defined as follows:

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

Logistic 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. To do so, he has just to implement an interface. See the mathematical documentation or the comments in the code.

2

Here, one enters the value of the parameter *a* for each iteration.

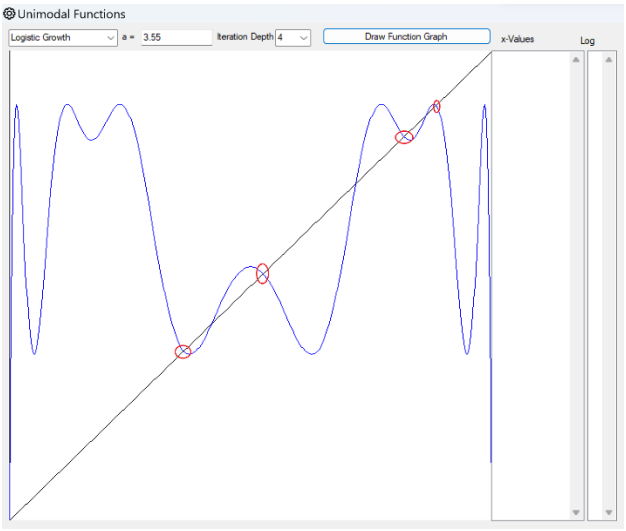
- Tent map: *a* is in the interval]0,2]
- Logistic growth: *a* is in the interval]0,4]
- Parabola: *a* is in the interval]0,2]

3

The depth of the iteration is defined here. It determines, how often the iteration function f is repeated in each iteration step. E.g., if the iteration depth is = 1, then it is: $x_{n+1} = f(x_n)$. If the iteration depth is = 5, then it is: $x_{n+1} = f^5(x_n)$. This is not important for the iteration itself, but it is useful when cycles of higher order are analysed with the help of the graph of the function. See next point.

4

The button "Draw Function Graph" draws the graph of the function with the chosen iteration depth. E.g., one sees for the parameter value a = 3.55 an attractive 4-cycle:



Attractive 4-cycle

To show this, one chose "logistic growth" as iteration, and the parameter value a = 3.55 with the iteration depth 4. That means, one analyses the four times iterated function f^4. If its graph is drawn, it appears that this graph intersects the 45° line at four points, where the elevation of the tangent is below 1 and that therefore the 4-cycle is attractive. One sees four more intersection points as well with the 45° line, that belong to a repulsive cycle, because the elevation of the tangent is above 1.

A detailed description is in the mathematical documentation.

5

The values x_n, produced by the iteration, are listed here.

6

This column shows the protocol of each value x_n in the left list. The protocol $p(x_n)$ is defined as follows:

In case of the tent map and the logistic growth:

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

In case of the parabola:

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

7

This is the sector to show all diagrams.

8

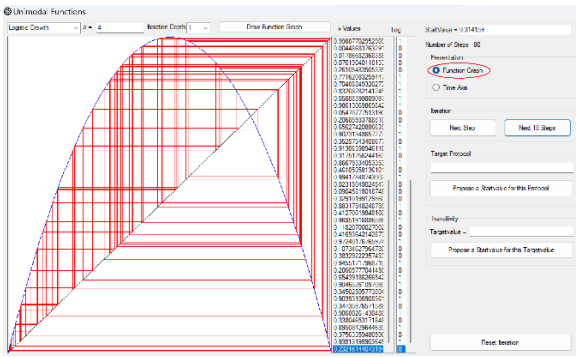
The start value of the iteration is defined in this text field. It is proposed by the program or can be set manually. In case of the tent map or the logistic growth, the start value must be in the interval]0, 1]. In case of the parabola in the interval [-1, 1].

9

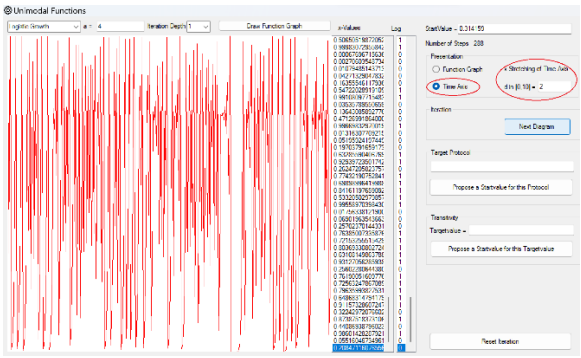
The number of iteration steps are displayed here.

10, 11

One has two options to present the result of an iteration:



Presentation by the function graph

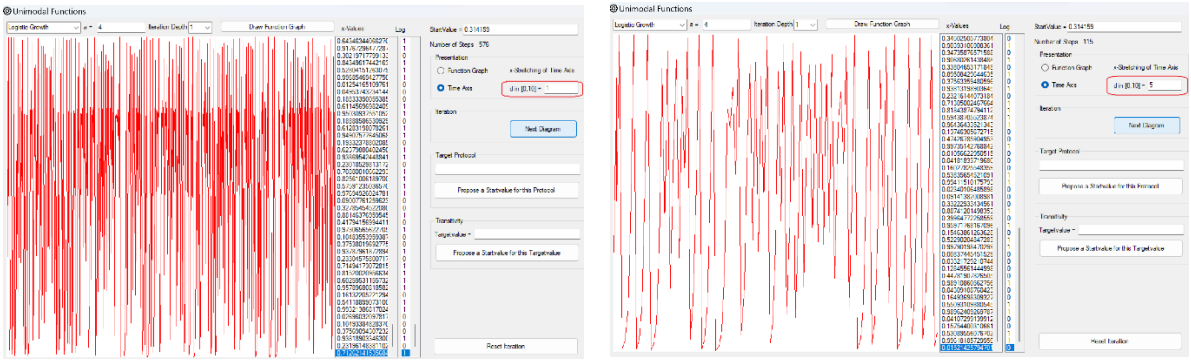


Presentation on the time axis

In the left case, one switches between the function graph and the 45° line. In the right case, the time axis shows in the horizontal direction and on the vertical direction, the actual value of the iteration is plotted.

12

In case of the presentation on the time axis, the iteration is probably not well visible, because it is too tight. Therefore, one can stretch the time axis by a parameter between 1 and 10.



The diagram on the left side has a stretch factor of 1, on the right side a stretch factor of 5

In case of the presentation by the function graph, the stretch factor is not relevant and therefore deactivated.

13

In case of the presentation by the function graph, one can iterate step by step with the button “Next Step”.

14

In case of the presentation by the function graph, the button “Next 10 Steps” iterates the next 10 steps. In case of the presentation on the time axis, this button is “Next Diagram” and starts so many steps of the iteration, until the diagram on the time axis is filled completely. If the button is pressed again, the iteration continues and creates the next diagram.

15

On can set an arbitrary protocol consisting of “0” and “1” in case of chaotic behaviour. Because of the limited accuracy of calculation of the computer, the length of the protocol can be maximal 50. The behaviour of the iteration is chaotic for the following parameter values:

- Tent map: a = 2
- Logistic growth: a = 4
- Parabola: a = 2

If the behaviour is not chaotic, an arbitrary protocol cannot be generated.

Example

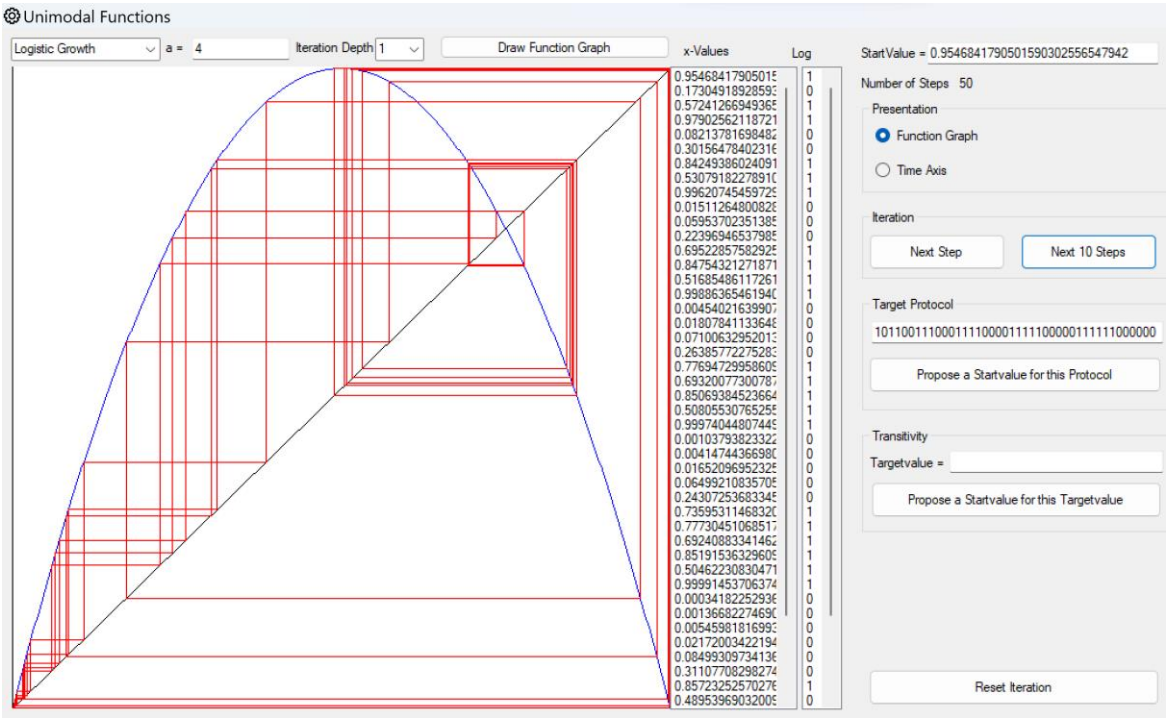
We choose the logistic growth with the parameter a = 4, that is the chaotic case.

We set as target protocol a 42-digit sequence of “0” and “1”:

101100111000111100001111100000111111000000

Now, we press the button “Propose a start value for this protocol”. The “Simulator” proposes the start value $x_1 = 0.954684179050159030255654794 2$.

By pressing the button “Next 10 Steps” the iteration is launched. In the column “log” we control that the desired protocol is effectively generated.



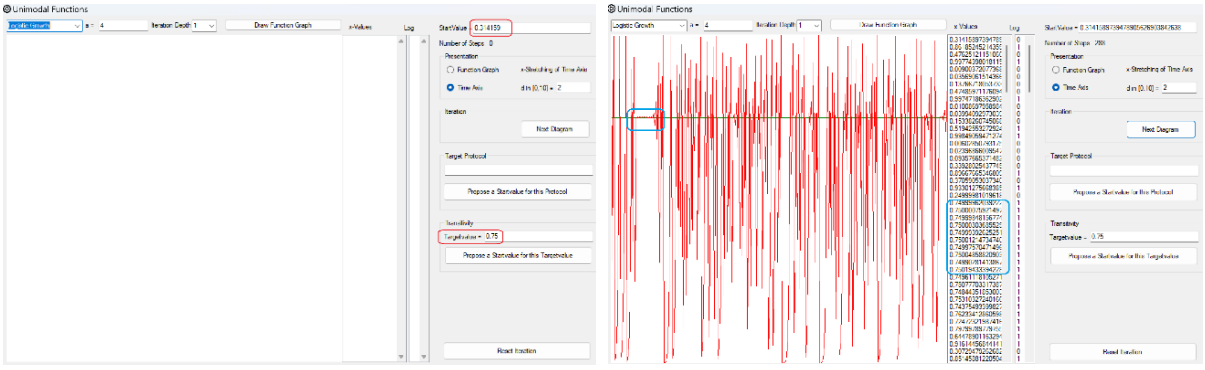
Generation of a desired protocol in the chaotic case

16

The button “Propose a start value for this protocol” starts the calculation of such a start value. This value is filled in field 8. Afterwards, one can start the iteration and control if the desired protocol is generated. This is only possible if the parameter a is chosen so that the behaviour is chaotic.

17

If the behaviour is chaotic, one can investigate the transitivity. For this, we choose a start value in field 8 and a target value in field 17. The theory tells us that there exists an alternative start value very near to the chosen one, so that the iteration sequence approaches the target value.



Investigation of transitivity

Example

See the picture above on the left side. The original start value in field 8 was 0.314159. The target value in field 17 is 0.75.

By pressing the button 18, the program proposes as alternative start value one near by the original one: 0.31415897394789056269038426 38. Afterwards, the iteration is started. In the diagram is the

target value marked by the green horizontal line. As we see, this target value is nearly hit after 24 iteration steps by the value 0.75000075…. See the right side of the picture and the blue marked sectors.

18

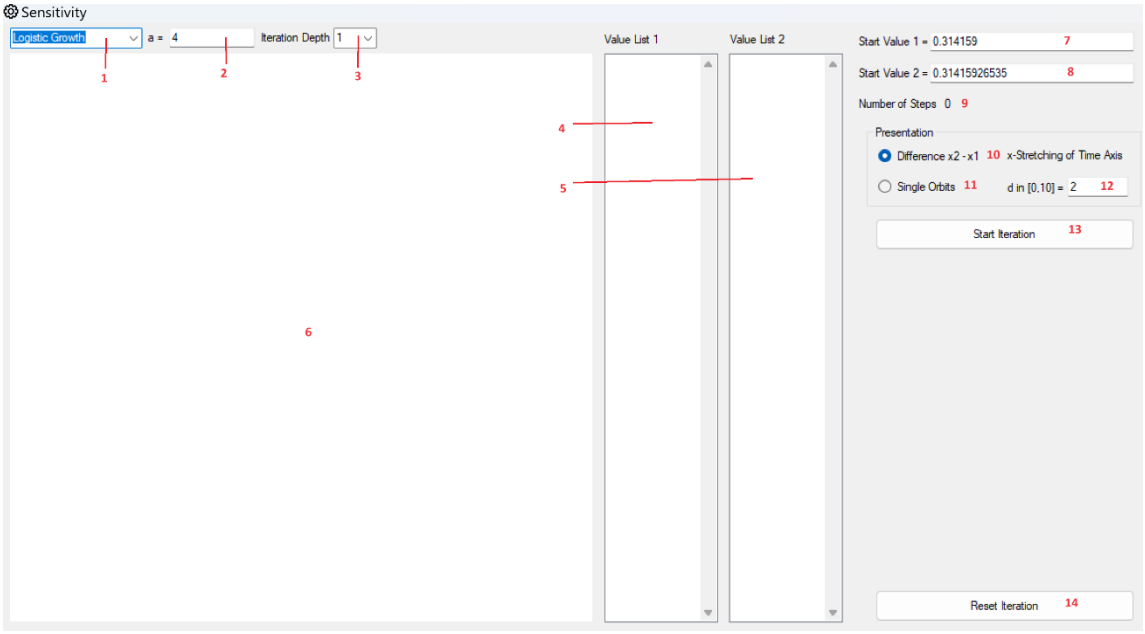
See the description above.

19

This button clears the diagram and resets the iteration.

3.2. Investigation of Sensitivity

The menu “Growth Models – Sensitivity” opens the following window:



Window to investigate sensitivity with the “Simulator”.

1

The type of iteration is chosen on the upper left corner. Available are:

- Tent map
- Logistic Growth
- Parabola

2

The parameter value a is set in this field.

- Tent map: a must be in the interval]0,2]
- Logistic Growth: a must be in the interval]0,4]
- Parabola: a must be in the interval]0,2]

3

The iteration depth is set here. It defines, how many times the iteration function *f* is repeated in each iteration step.

4

The generated values by the iteration are listed here, starting with start value 1.

5

The generated values by the iteration are listed here, starting with start value 2.

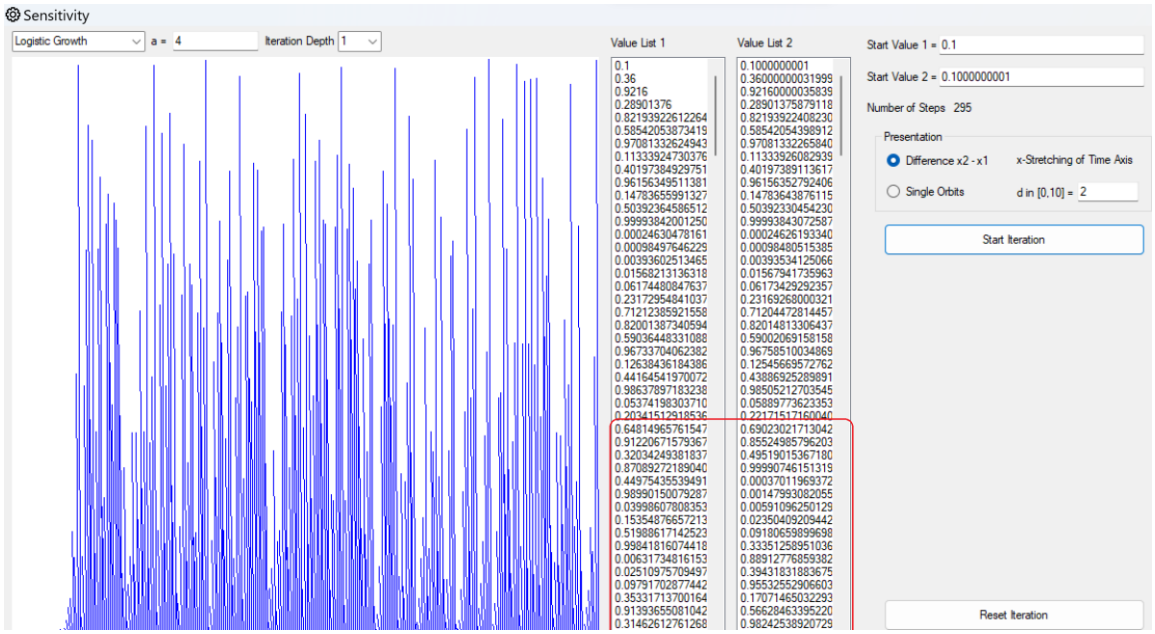
6

The sector for graphic diagrams.

7, 8

Two different start values are set in these fields. The distance between them can be very little. The goal is to show, that the generated values of the iteration can divergence arbitrarily.

Example



Two start values with a small difference

The two start values differ by 0.0000000001. In the diagram above, one can see that the generated orbits are quite similar in the beginning, but divergence later clearly in the red marked sector. The graphic on the left side shows the difference between the generated orbits in each step.

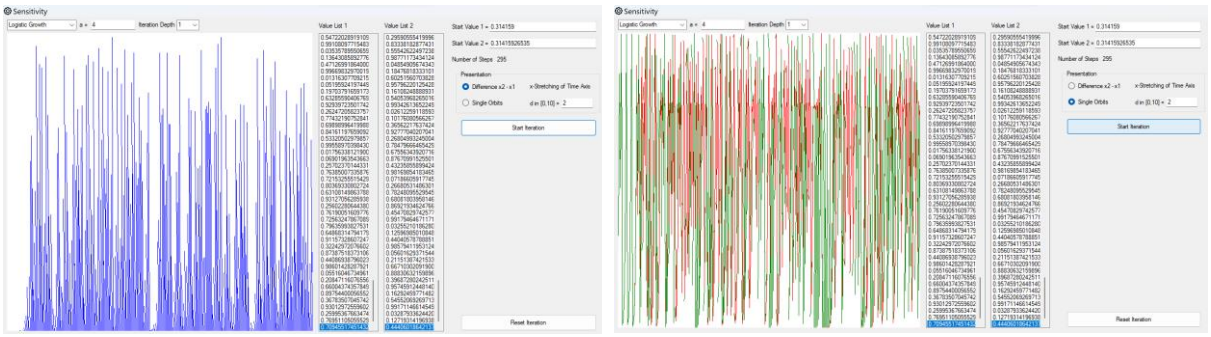
9

The number of iteration steps are shown here.

10, 11

There are two presentations for the graphical display of the iteration. On the left side, the difference between the orbits starting with start value 1 and start value 2 is shown. This difference can get arbitrarily big after some iteration steps.

On the right side, each orbit is displayed separately. In both cases, the presentation is on the time-axis and can be stretched.



Left: The display of the difference between the orbits. Right: Each orbit is shows separately

In both cases above one sees that the orbits are similar in the beginning and divergence clearly after a while. The stretch factor was set = 3.

12

In this field, the stretch factor is set. Standard is 2. Possible are factors in the interval [1, 10].

13

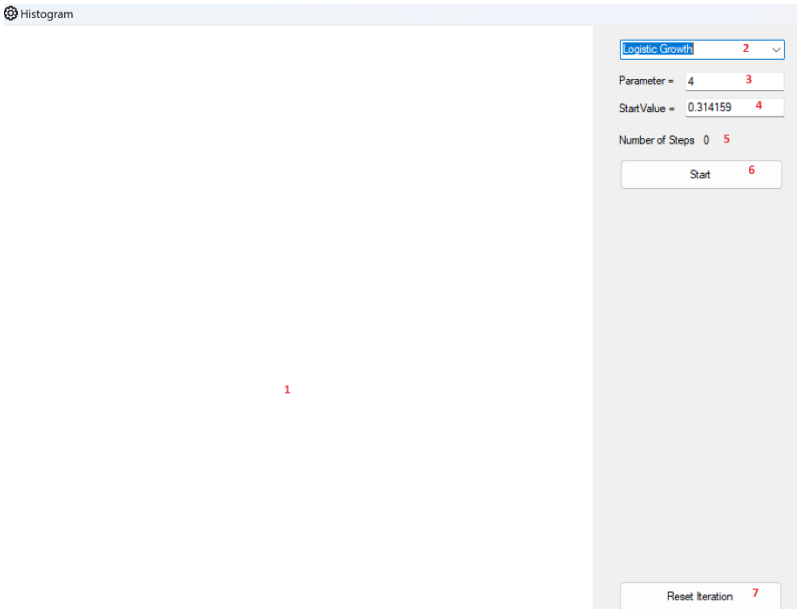
The button «Start Iteration» starts the iteration.

14

This button clears the diagram and resets the iteration.

3.3. Histograms

In the chaotic case, the distribution of the iteration values can be displayed in a histogram. (In the non-chaotic case, this is not interesting.) The menu point “Growth Models – Histogram” opens the following window:



Window to generate histograms

In this window, the value range is divided into small intervals. The width of such an interval corresponds about to the pixel size. The program counts how many times a small interval is hit by an iteration value during the iteration. In the middle or in case of the tent map, the distribution is quite evenly. In case of the logistic growth or the parabola, the borders of the histogram in the left and right side can be explained (see mathematical documentation).

1

Area to display the histogram.

2

The type of iteration is chosen here. Available are:

- Tent map
- Logistic Growth
- Parabola

3

The parameter value is set here. The histogram is only meaningful in the chaotic case. That is for the following parameter values:

- Tent map: a = 2
- Logistic growth: a = 4
- Parabola: a = 2

4

The start value for the iteration is set here.

5

The number of iteration steps is displayed here.

6

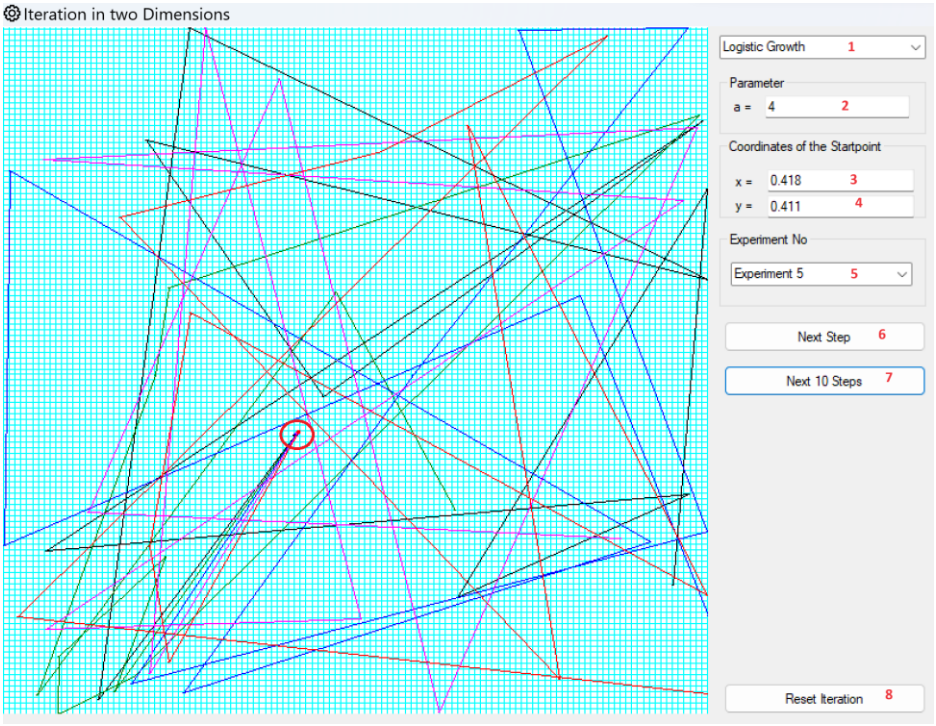
The button «Next 100 Steps» launches the next 100 iteration steps.

7

This button resets the iteration.

3.4. Presentation in two Dimensions

The menu “Growth Models – Two Dimensions” opens the following window:



Window for the two-dimensional presentation

The function of the different fields and buttons are explained below.

1

The type of iteration is chosen here. Available are:

- Tent map
- Logistic Growth
- Parabola

2

The parameter value is set here. The two-dimensional case is meaningful in the chaotic case. That is for the following parameter values:

- Tent map: a = 2
- Logistic growth: a = 4
- Parabola: a = 2

3 and **4**

Two start values x_1 and y_1 are entered in this fields. That gives a start point $P(x_1, y_1)$. It is plotted into the diagram on the left side.

5

The experimenter can start now up to 5 experiments with different start points. The different orbits are distinguished by different colours in the diagram. In field 5, the experimenter can allocate to each experiment a number between 1 to 5 in field 5.

Let's suppose that the experimenter has only a limited precision of measurements. This is shown in the diagram by the light blue raster. The size of the raster is 5x5 pixels that corresponds to a mathematical raster of 0.00825 x 0.00825 units.

The experimenter has started 5 experiments in the image above with the start points:

$$P_1(0.414, 0.407), P_2(0.415, 0.408), P_3(0.416, 0.409), P_4(0.417, 0.410), P_5(0.418, 0.411)$$

All start points belong to the same measure square of the experimenter. That means that all start points are equal for him. In the diagram, this start point is marked by an orange circle.

Now, the experimenter starts each experiment and launches the iteration with – for him – the same start point. But the orbit of the different start points divergence after a few steps of the iteration. For the experimenter it looks like the same start point leads to different orbits. The behaviour of the system seems random.

In the process, the chosen iteration is applied for both dimensions x and y.

6

Here, a single iteration step is done, and the new point is plotted in the diagram and connected with its predecessor.

7

The next 10 iteration steps are performed.

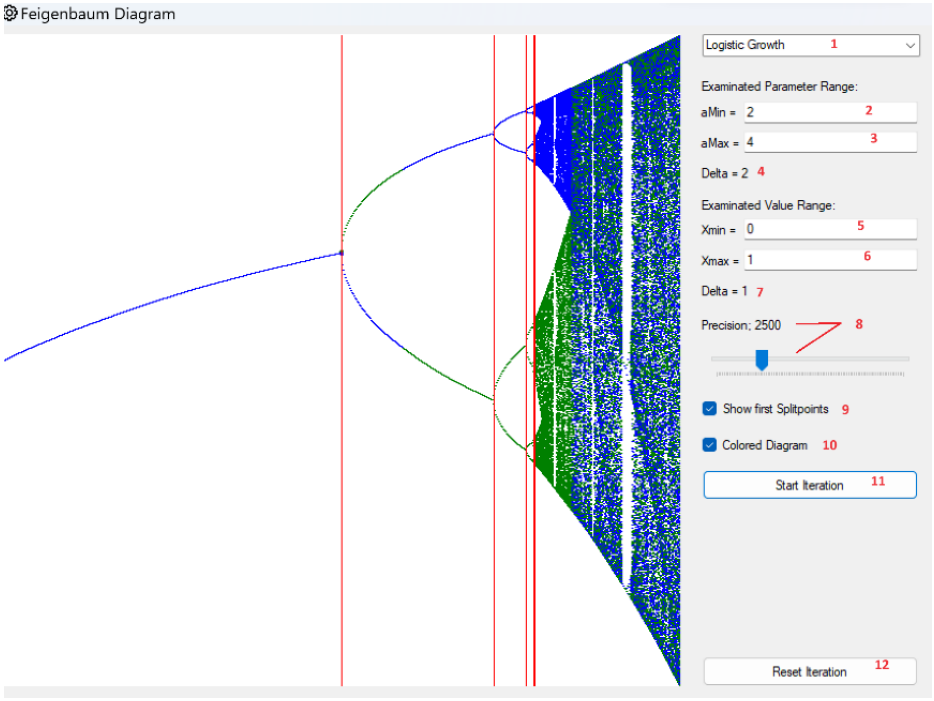
8

The iteration is reset.

3.5. Feigenbaum Diagram

The "Simulator" allows to examine the well-known Feigenbaum-diagram and the period doubling. This is described more detailed in the mathematical documentation.

The menu « Growth Models – Feigenbaum" opens the corresponding window.



Window for the examination of the Feigenbaum-diagram

The diagram shows in horizontal direction the values of the parameter a, that grows in this direction linearly. For each value of a, the iteration runs a while without plotting anything and supposing that the orbit convergences to an attractive fixpoint or cycle, if there is any at all. Afterwards, the orbit is plotted into the diagram in vertical direction.

In the window above the examination starts at a = 2. Till the first split point (the first vertical red line), the iteration has one fixpoint and it is plotted for each a by the program. Afterwards, this fixpoint splits in a 2-cycle, and then in a 4-cycle and so on until the behaviour gets chaotic. Further right, there are no cycles visible except in some little windows.

The function of the different fields is explained below.

1

The type of iteration is chosen here. Available are:

- Tent map
- Logistic Growth
- Parabola

2, 3, 4

The examined interval of the parameter a is defined here. In 2 the left and in 3, the right border of this interval is set. The examined area of a is then the interval [aMin, aMax]. In 4, the width of this interval is shown.

The user can set these values manually, but they must be in the allowed parameter interval of a that is:

- Tent map: a is in the interval]0,2]
- Logistic growth: a is in the interval]0,4]
- Parabola: a is in the interval]0,2]

5, 6, 7

If the user likes to see only a section of the value range, he can set it here: in 5 the minimal value and in 6 the maximal one. Displayed is then the value range (for the iteration values) [Xmin, Xmax]. The width of the value range is shown in 7.

8

Especially if the user zooms into small parts of the diagram, it can be necessary to increase the precision of the iteration. This precision specifies, how many iteration steps are done in the "dark mode", that is without plotting any values, before one hopes that now the cycles are ready for plotting (if there are any). The standard of the precision is 25'000 steps and that is enough mostly. But the precision can be augmented to 100'000 steps.

9

The vertical red lines in the diagram show the first split points of the first series of period doubling starting at about $a \approx 3.5$. This display can be activated or deactivated.

10

The diagram is displayed in one or two colors. The user can set this option here. If somebody likes more or other colors, he can adapt the program code of the function *SetColor* of the *FrmFeigenbaum*.

11

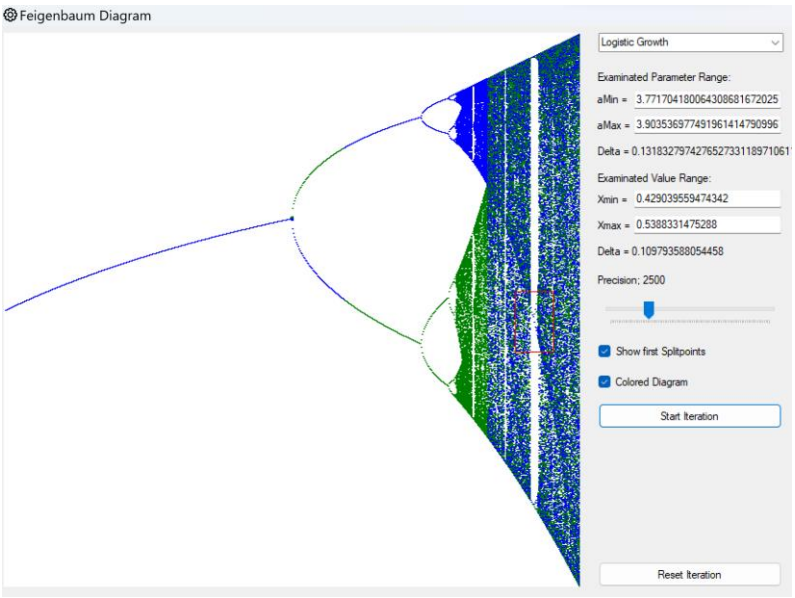
The iteration is started here …

12

.. and reset here.

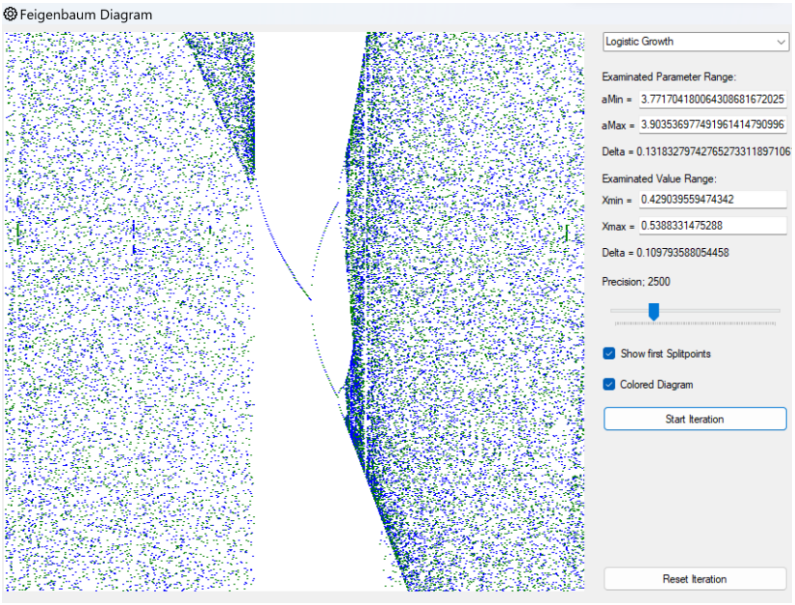
Selection of small sections in the diagram

The user can select the part of the diagram that should be examined deeper by a selection rectangle. This happens with a pressed left mouse button. Thereby the parameter fields 2-4 and the value fields 5-7 on the right side are adapted automatically.



Selection of a small section of the diagram

The user likes to examine the environment of a 3-cycle point deeper in the image above. He drew a rectangle with pressed left mouse button to mark this area. The parameter and value intervals were adapted on the right side to his selection. If the user starts the iteration now, the selected area is displayed:

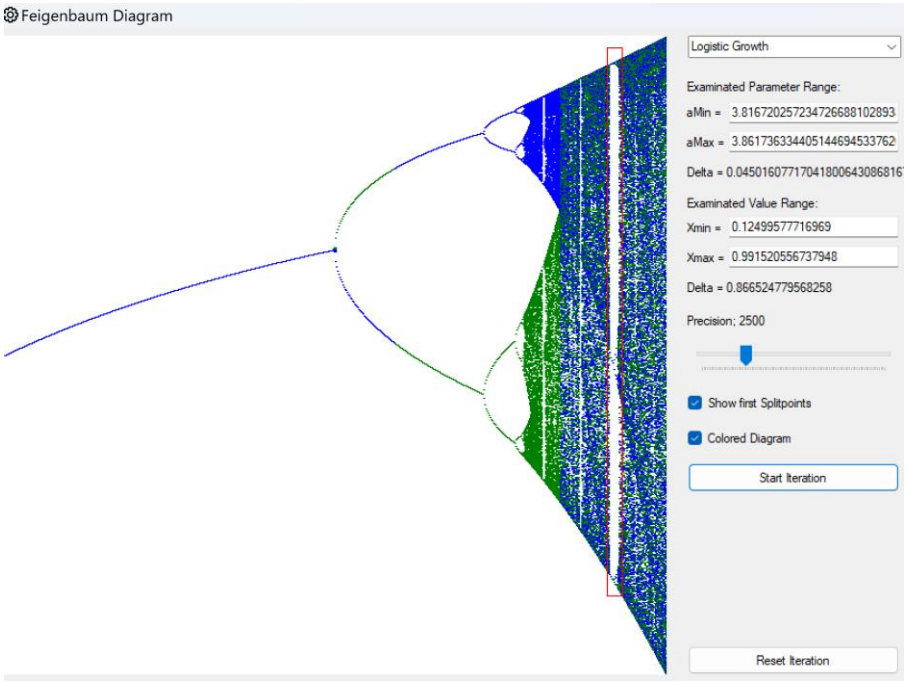


Display of the selected area of the 3-cyclic point

In the image above it is shown how the 3-cyclic point appears out of chaos. Because of the reduced value range, only one point of this cycle is visible. The diagram goes then again into chaos over a new period doubling.

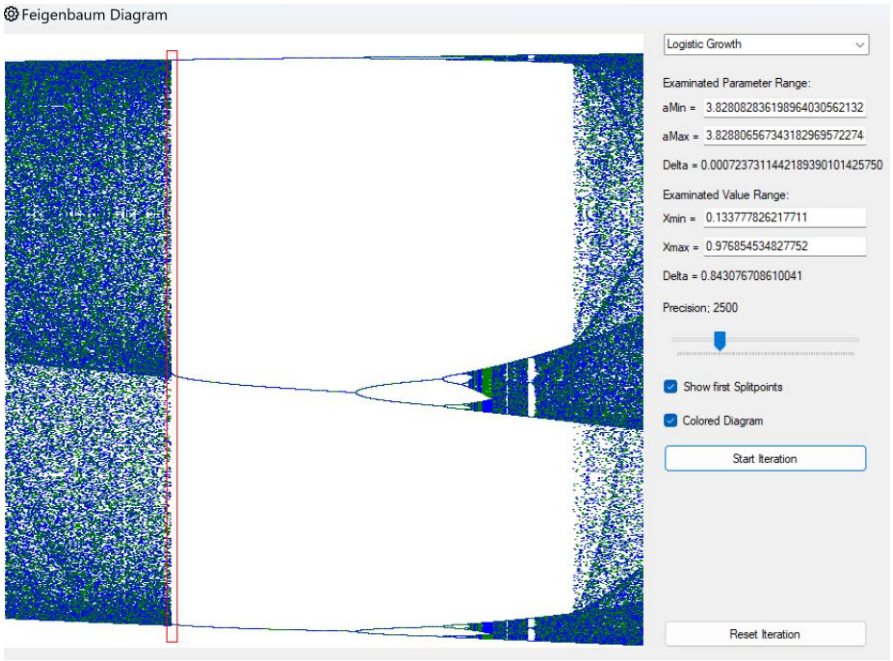
The user can also examine, for which parameter values of a certain cycles are expected. He can just place the selection rectangle on the interesting sector and then the corresponding values are shown on the right side in [aMin, aMax].

As example the selection rectangle in the image below shows the appearance of the 3-cycle. One sees that $a \in [3.816, 3.861]$.



Place the selection rectangle to examine a parameter interval.

If we restart the iteration in this range of a, we can locate the 3-cycle better:



The 3-cycle appears for $a \in [3.8280,3.8288]$

4. Mechanics

4.1. Billiard

The «Simulator» allows to examine different types of billiard. Implemented are:

- Elliptic billiard

- Billiard in the stadium
- Oval billiard

A parameter C > 0 defines in all cases the form of the billiard table. C is:

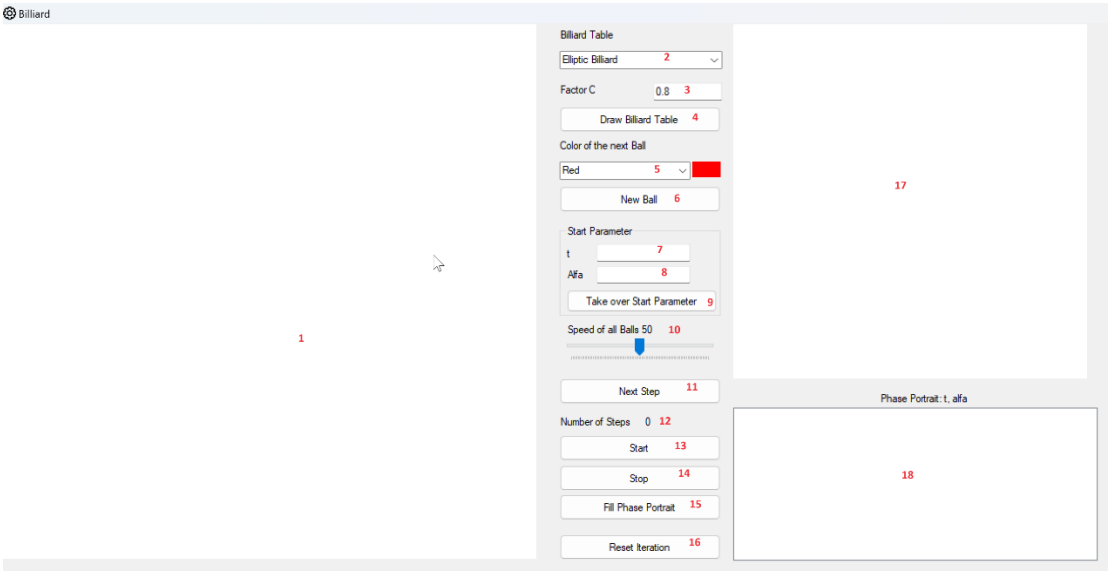
- The proportion of minor axis to major axis in case of the elliptic billiard
- The proportion of the circle diameter to the width of the rectangle in case of the stadium
- The proportion of the circle radius (on the right side) to the major axis of the ellipse (on the left side

The user can program other types of billiard because the code is public. He must only implement an appropriate interface. See the mathematical documentation or the comments in the code.

It is supposed that the billiard ball moves frictionlessly, and that no energy is lost when it hits the border of the billiard table. Furthermore, it is possible to place as many balls on the billiard table as one likes. They don't interact with each other. They are only useful to study different orbits depending on the start parameters.

The first start parameter *t* defines a point on the border of the billiard table according to the mathematical documentation. The second start parameter is the reflection angle α (in radian) between the orbit of the ball and the tangent in the hit point.

The procedure to investigate the billiard is the same for all types of billiard. The menu "Mechanics – Billiard» opens the following window:



Window to examine different types of billiard.

1

This is the area to display the billiard table and the orbit of the ball.

2

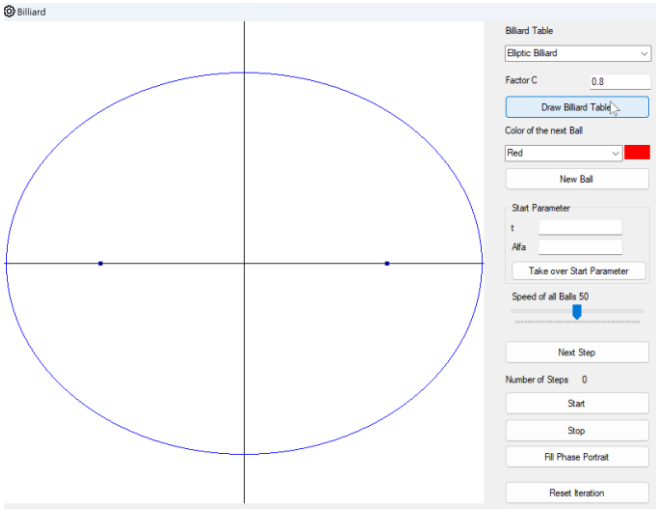
The type of billiard is chosen here. See the list on top.

3

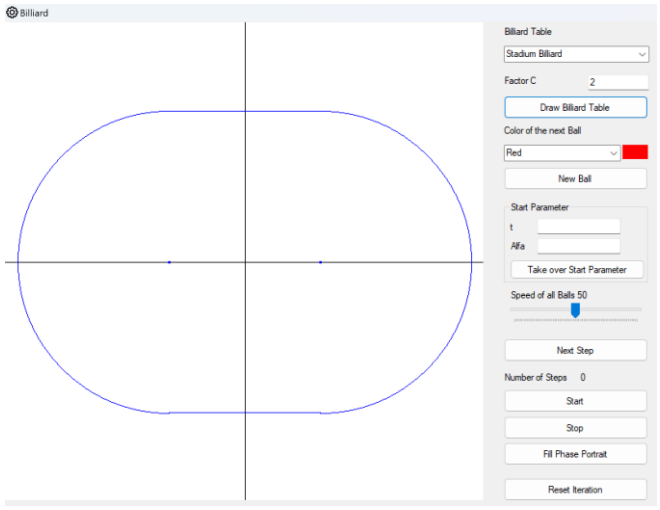
The form of the billiard table is defined hare by a parameter C according to the previous description. It is C > 0. In case of the elliptic or oval billiard, the billiard table is a circle for C = 1.

4

This button draws the billiard table. Two examples:



Elliptic billiard table



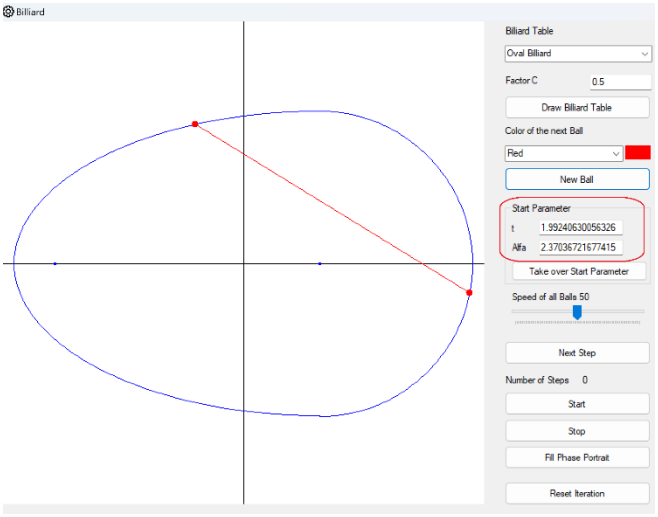
Billiard in the stadium

5

The ball is programmed in a way that arbitrary many instances of the ball can be generated. To distinguish them, one can choose a color for the next ball generated.

6

This button generates a new ball. It is placed on the border of the billiard table, depending on the type of billiard. Afterwards, the ball is placed manually by the mouse and pressed left mouse button. If one releases the mouse button, the start point of the ball on the border of the billiard table and thereby the parameter *t* is set. Now, the left mouse button is again pressed to define the direction of the first ball movement. If the mouse button is released, the first reflection angle α is set.



Oval billiard with set start position and start angle of a ball.

If the ball and the first section of its orbit is manually placed as described above, the start parameter t and the start angle α are displayed on the right side of the window. See the red marked area above. It can be useful to set these parameters manually in these fields on the right side. In that case, one can take over the parameters for the start of the ball.

7, 8

The start parameters are displayed in these fields. The values can also be set manually. This is useful if one tries to get certain orbits by an approximation procedure.

9

If the user has set the start parameters manually in the fields 7 and 8, these parameters can be taken as start position of the ball by this button. The ball is then placed accordingly. The parameter t is calculated modulo 2π or modulo the circumference L of the billiard table. The reflection angle α is calculated modulo π. Therefore, all real start parameters are allowed and after calculating modulo, it holds: $t \in [0, 2\pi[$ or $t \in [0, U[$ and $\alpha \in]0, \pi[$.

10

The speed of the billiard ball is regulated here. All balls have the same speed.

11

This button launches the next hit for all balls on the table.

12

The number of steps are displayed here.

13

This button launches all balls hitting until

14

This button is pressed.

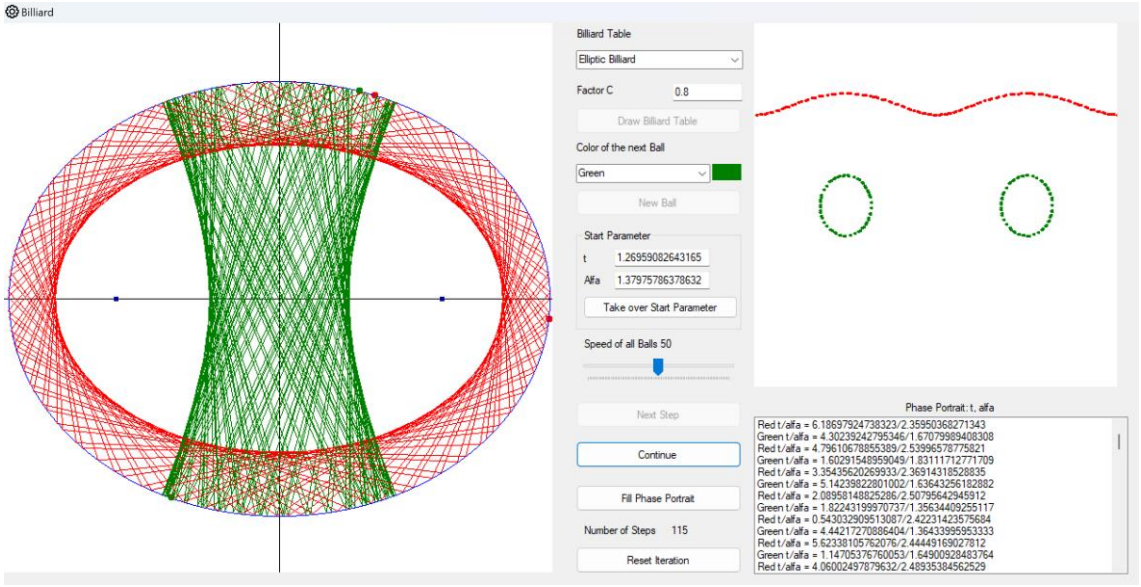
15

With this button, many balls are generated and spread over the table. With the “start” button, these balls are started and the phase portrait is filled.

16

This button resets the whole iteration and clears the diagram.

Example



Elliptic billiard with two balls

In the example above, two balls were placed. The red orbit runs between the ellipse focuses and the green orbit out of them. The caustics are deduced in the mathematical documentation.

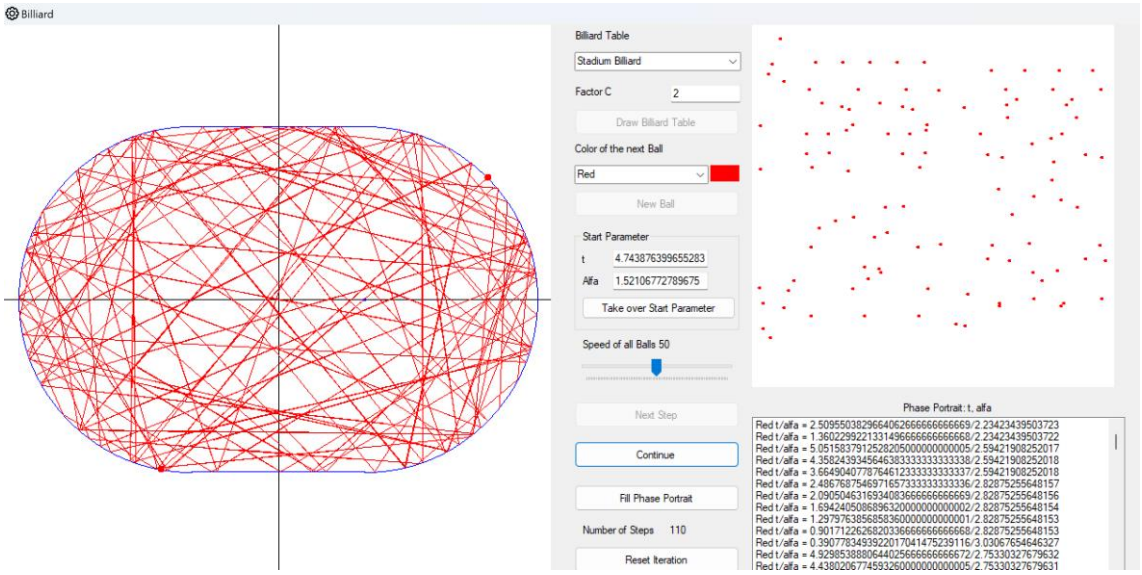
17

The “phase portrait” of the movement is plotted here. The parameter *t* is plotted in horizontal direction and the reflection angle α in vertical direction. In the example above, one sees that *t* passes through all possible values in the green case, while α is limited on a certain value range. In case of the red orbit, both parameters *t* and α are in a limited value range.

18

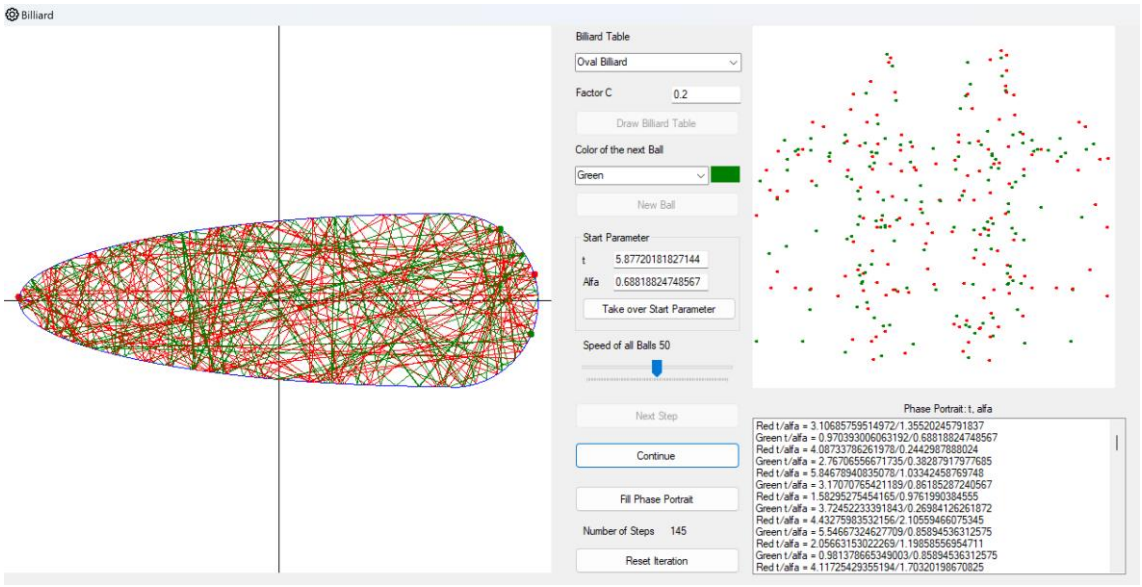
All parameter pairs (t_n, α_n) are listed here, distinguished by the ball color.

Examples



A ball on the stadium billiard table

There appears no structure in the phase portrait on the right side. The parameter pairs look strewed randomly.

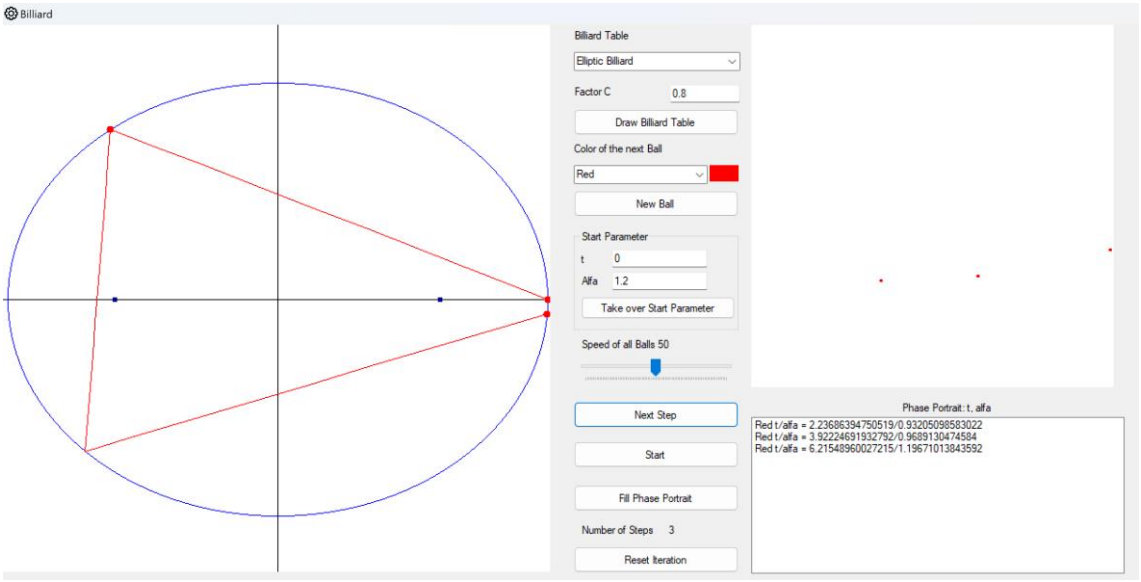


Two balls on the oval billiard table.

There is no structure in the phase portrait neither.

A last example shows the possibility of determining cycles approximately by nested intervals.

Supposed, we are looking for a 3-periodic cycle in case of the elliptic billiard. We choose as start point the parameter $t = 0$, that is the start point $(a, 0)$. With some tries we see that the start angle $\alpha_1 = 1.2$ delivers a first rough approximation. We enter these start parameters in fields 7 and 8, take them as start parameter for the ball by button 9 and thereby the ball is placed. Afterwards, we launch the first three hits of the ball:



The value $\alpha_1 = 1.2$ is a little too small.

This is visible optically. But it is better to control the protocol on the right side where it is obvious that $t = 6.215489\ldots$ is a bit smaller than 2π.

One realizes similarly, that $\alpha_2 = 1.25$ is a little too big. We enter these values in each case manually into the fields 7 and 8 and take over these values for the start position of the ball. Afterwards, we control the parameter *t* after three hits. If $t > 0$ then α was a little too big. If *t* is a little below 2π then α was a little too small. We divide in half the interval of the last α-values, and we replace one side of this interval by a better α-value. This gives us the following series of nested intervals:

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 wanted 3-periodic cycle starts at $t = 0, \alpha \in [1.2226, 1.2235]$

Of course, this method converges very slowly. But based on continuity arguments, one can conclude the existence of a 3-periodic cycle. See the mathematical documentation.

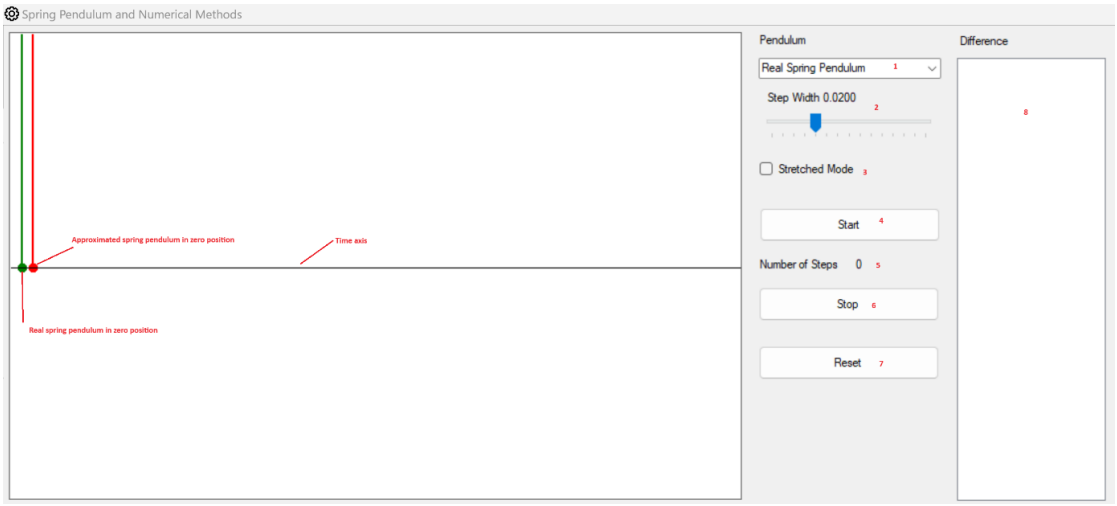
4.2. Numerical Methods

These experiments are based on the classic spring pendulum, that oscillates here with a Cosinus function. One can compare its oscillation with spring pendulums, that are not oscillating with a Cosinus function, but whose oscillations are approximated by numerical methods. These are simple methods that are accessible to an elementary mathematical view (except maybe the Runge Kutta method). Available are:

- Forward Euler
- Backward Euler
- Implicit Midpoint Rule
- Runge Kutta method of degree 4

All these methods are described in detail in the mathematical documentation.

To start, one opens the following window:



Window to examinate the spring pendulums

On the left side, the oscillations of the pendulums are shown. Green: The “real” spring pendulum that oscillates in a Cosinus function. Red: The spring pendulum whose oscillation is approximated by a numerical method.

The pendulums above are in a zero position. If one starts the iteration here, nothing happens. The pendulums must be set in a start position by pressed left mouse button.

The time axis and the zero level is shown horizontally. In vertical direction, the position of the pendulums is shown in an interval [-1, 1].

1

The numerical method of the red pendulum is chosen here. For control purposes, one can also chose the real spring pendulum.

2

The real, green pendulum increases the time variable in each step by 0.1. The red pendulum can divide this interval finer according to the step width chosen in 2. If this step width is e.g. 0.02 like in the image above, then the red pendulum iterates this step width 5 times, before its position is shown in the image. At the same time, the green pendulum is shown with its step width 0.1. This guarantees that the pendulums are synchronized.

3

If one activates “stretched mode”, both pendulums oscillate with the same step width for the time parameter, e.g. both with step width 0.02. This leads to a stretched presentation for both pendulums and one can see the differences between the green and the red pendulum better.

4

This button starts the iteration for both pendulums. If the oscillation is interrupted, it can be proceeded by this button.

5

The number of iteration steps on the time axis is shown.

6

The oscillation of the pendulums is interrupted.

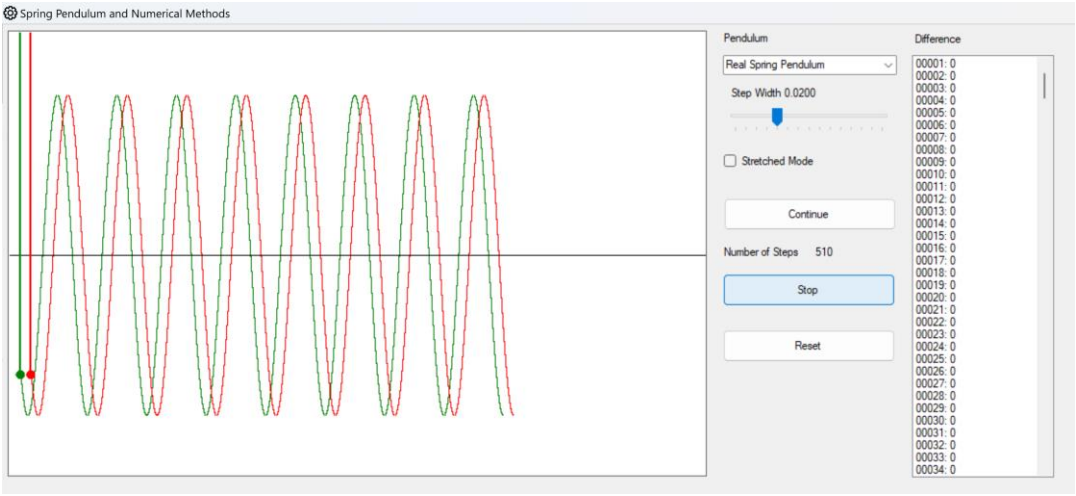
7

The iteration is reset, and the pendulums are again in the zero-start position.

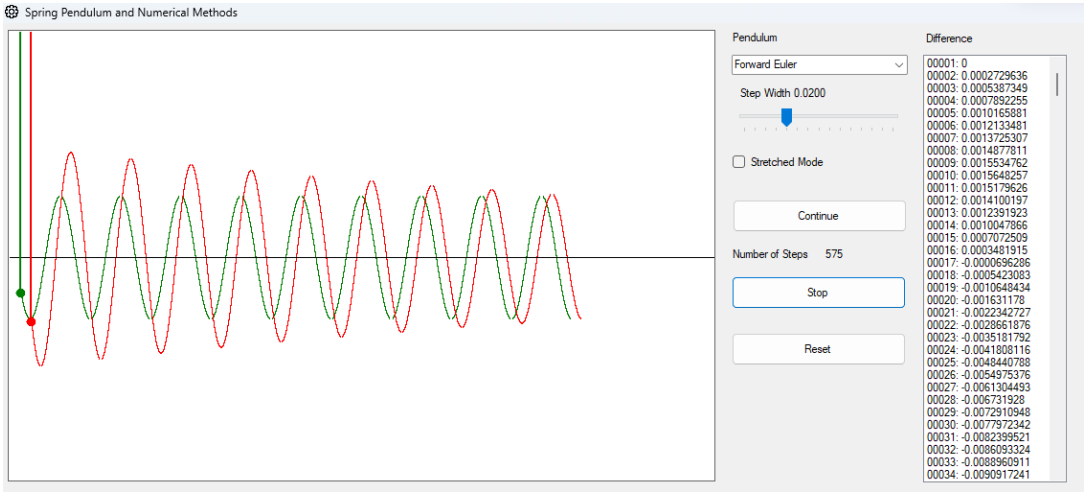
8

The difference of the y-position of the green and the red pendulum is protocolled here.

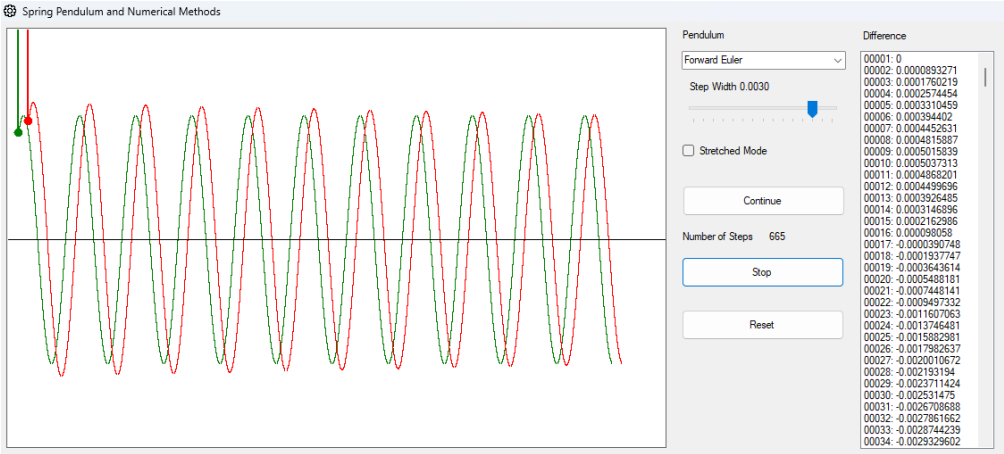
Some examples follow here:



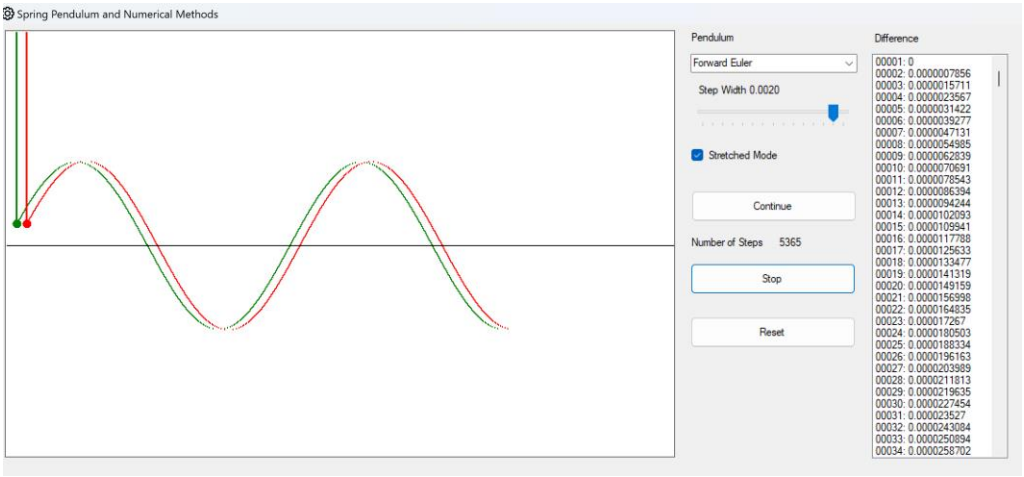
The red pendulum oscillates like a real pendulum too. The difference on the right side is zero.



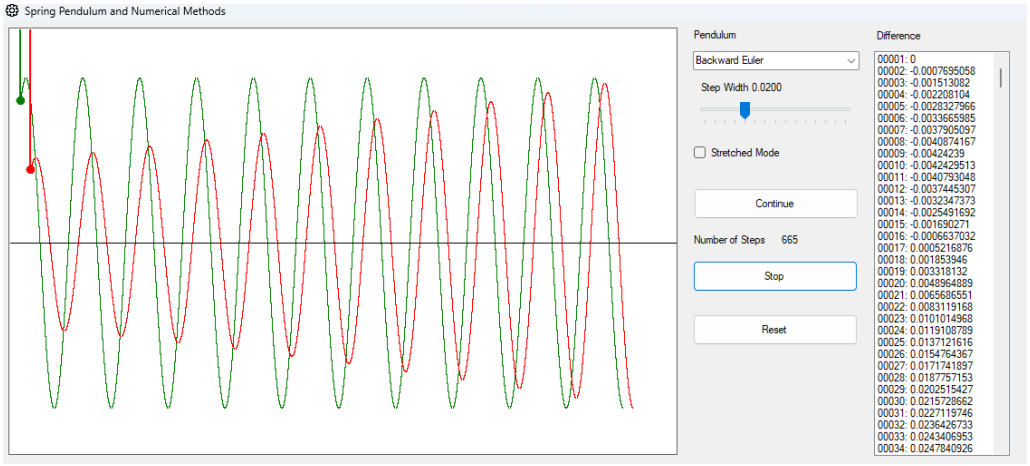
The red pendulum is approximated by the “Forward Euler” method.



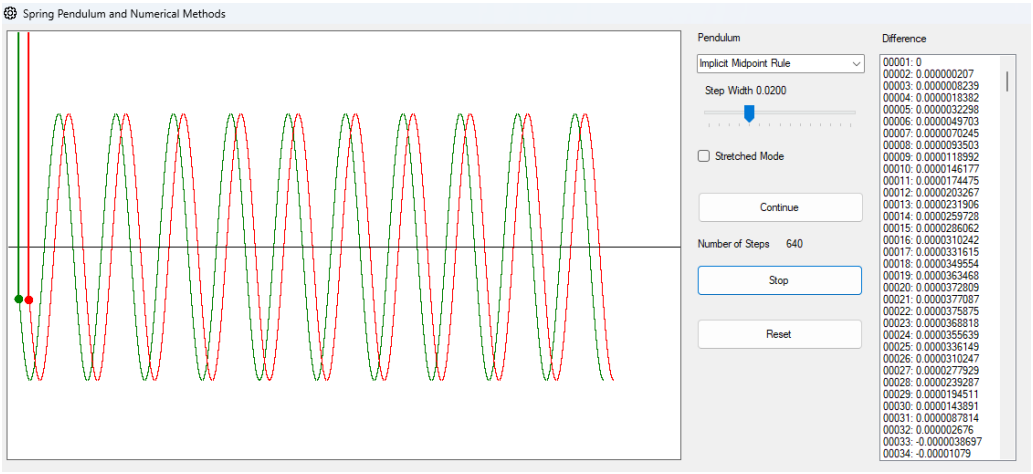
The same situation, but with a smaller step width for the red pendulum.
It takes more time until the difference between the pendulums is shown.



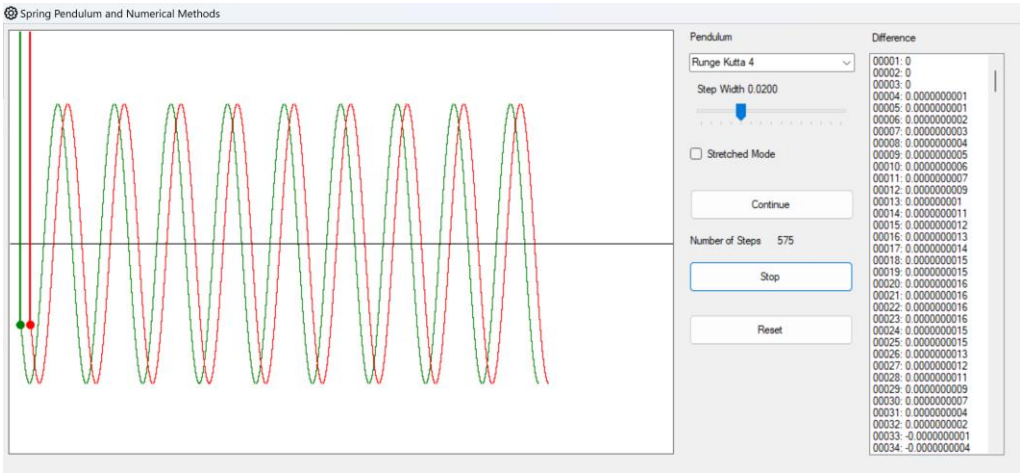
The same situation, but in stretched mode. Both pendulums have the step width 0.003.



The red pendulum is approximated by the “Backward Euler” method.



The “Implicit Midpoint Rule” method is a quite good approximation.
On the right side, one sees that the difference is not zero.

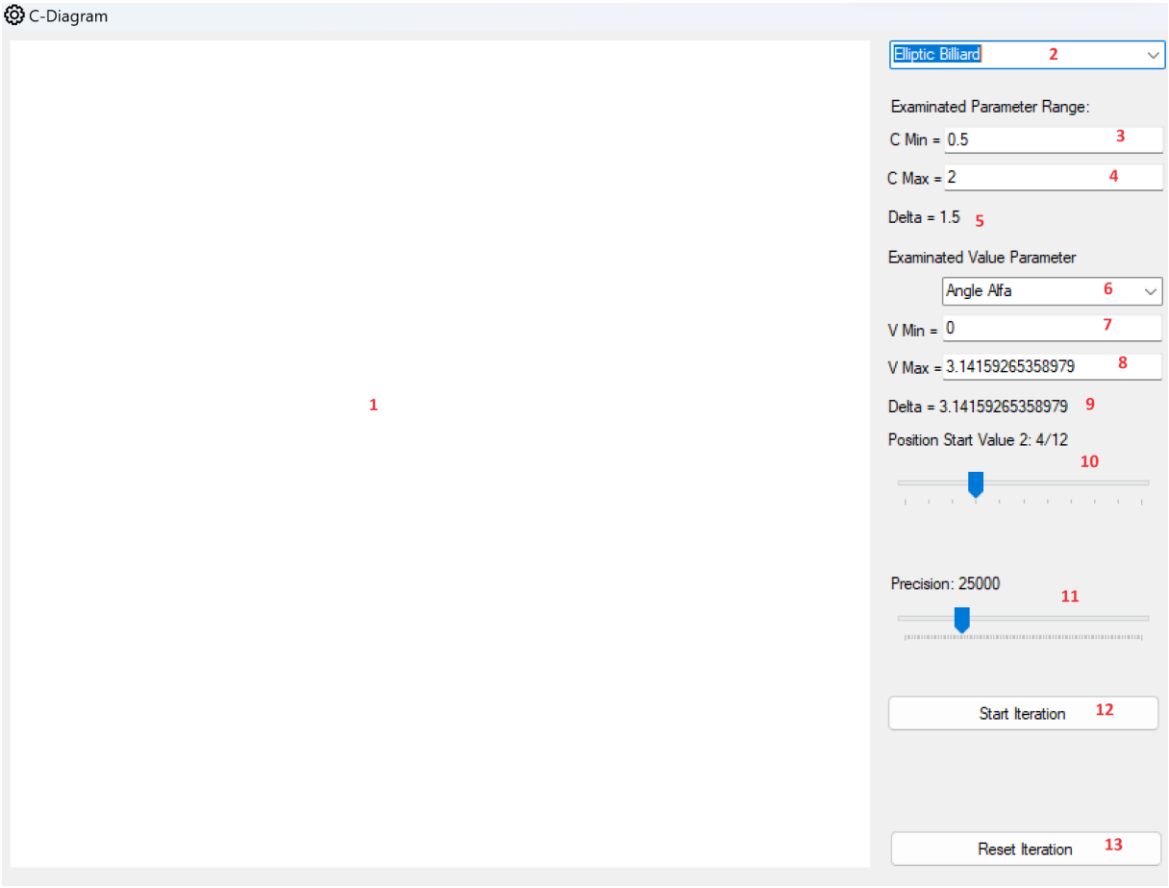


The best approximation is the “Runge Kutta” method of order 4.
See the small differences on the right side.

4.3. C-Diagram

The Feigenbaum-diagram shows in case of unimodal functions the behaviour of an iteration depending on the parameter a. In case of the billiard, the parameter C plays this role. We try to generate for the billiard a similar diagram like the Feigenbaum for unimodal functions. We call this diagram C-diagram.

The menu “Mechanics – C-Diagram” opens the following window:



Window to examine the C-diagram.

1

Area to display the C-diagram.

2

The type of billiard is chosen here.

3, 4, 5

The user determines here which interval of C is examined. It holds: $C \in [Cmin, Cmax]$. The width of this interval is shown in 5.

6

The movement of the ball is described by two parameters. We chose here which of them should by displayed in dependence of C.

7, 8, 9

We define here for the chosen value parameter, which interval should by investigated deeply. The value parameter is then in the interval $[Vmin, Vmax]$. The width of this interval is shown in 9.

10

The first value parameter t describes the position of the hit point on the border of the billiard table. The second value parameter α is the reflection angle at the hit point.

The value parameter, that is *not* chosen in field 6, is set by standard on 1/3 of its value interval. That is as example $t = {}^{2\pi}/_{3}$ in case of the elliptic billiard.

The start position of the other value parameter, that should be examined, can be set between 1/12 and 11/12 of its value interval. That is as example $\alpha = {}^{7\pi}/_{12}$ like shown in the image above in field 10.

11

The precision is set here like in case of the Feigenbaum-diagram.

12

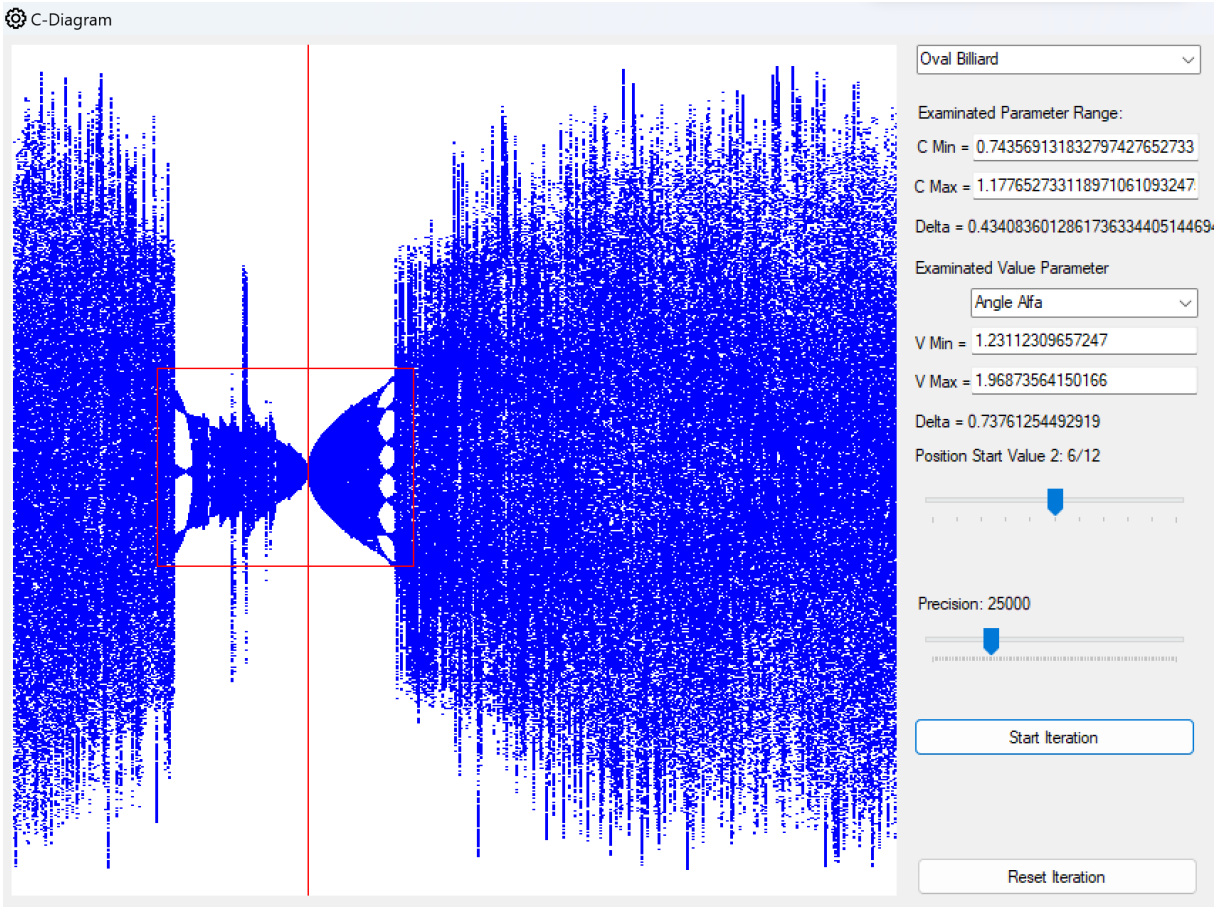
The iteration is started here …

13

… and reset here.

Selection of small sections in the diagram

Let's start the C-diagram for the oval billiard with its standard parameters.



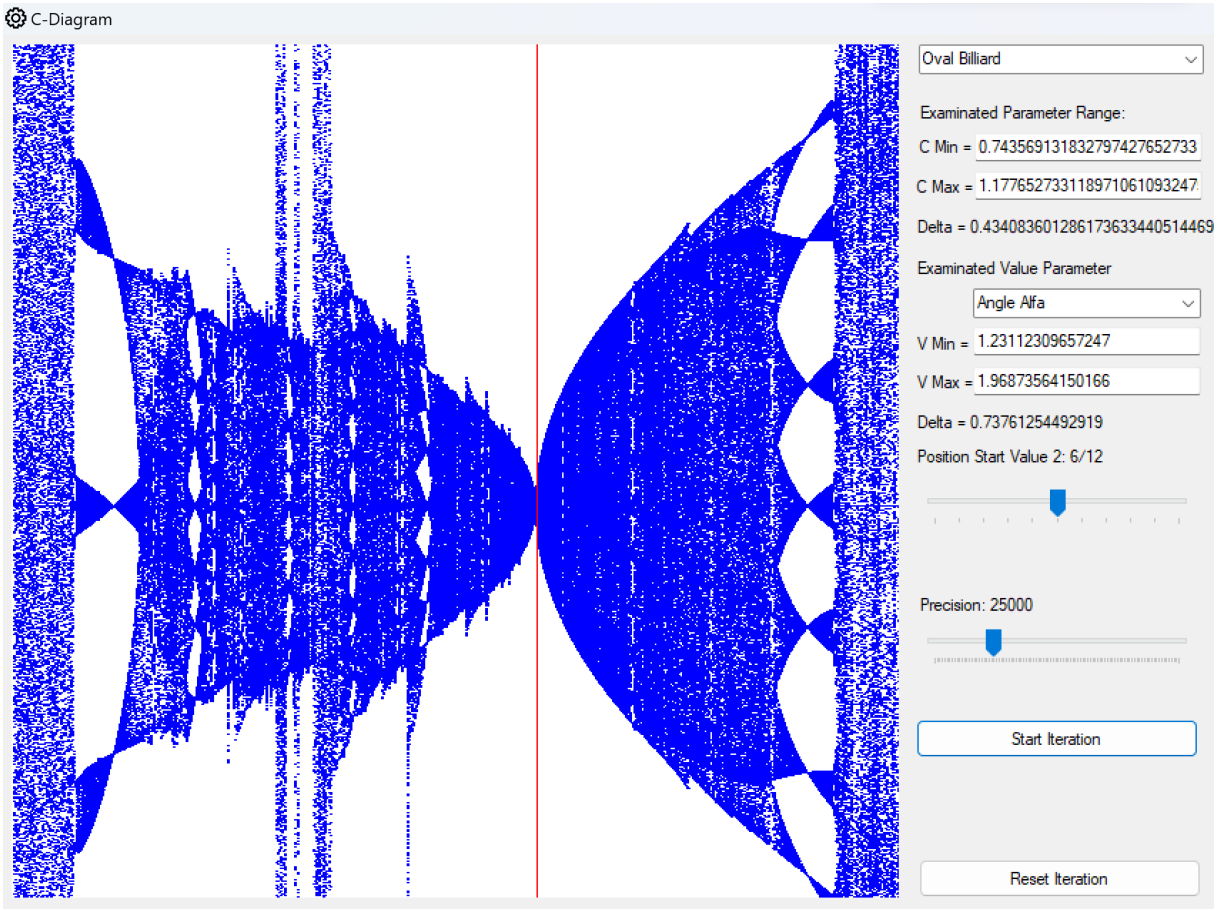
Examination of the value α in case of the oval billiard with a user selection

The reflection angle α was examined for the parameter range $C \in [0.5, 2]$ in case of the oval billiard above. The red vertical line marks the parameter value C = 1. For this value, the billiard table is a circle and the angle α constant. That is shown as a point in the diagram.

The user has now the possibility of choosing a part of the diagram by a selection rectangle like in case of the Feigenbaum-diagram. This happens with pressed left mouse button and moving the mouse. On the right side, the according values of the parameter range are shown in fields 2-4 and the according values of the value range in fields 5-7 automatically.

In the image above, the user selected with pressed left mouse button the red marked rectangle. On the right side, the according intervals $Cmin, Cmax]$ and $[Vmin, Vmax]$ are shown like in case of the Feigenbaum-diagram.

If one restarts the iteration, one gets the following image:



Selection of the C-diagram for the oval billiard.

Remark for further experiments

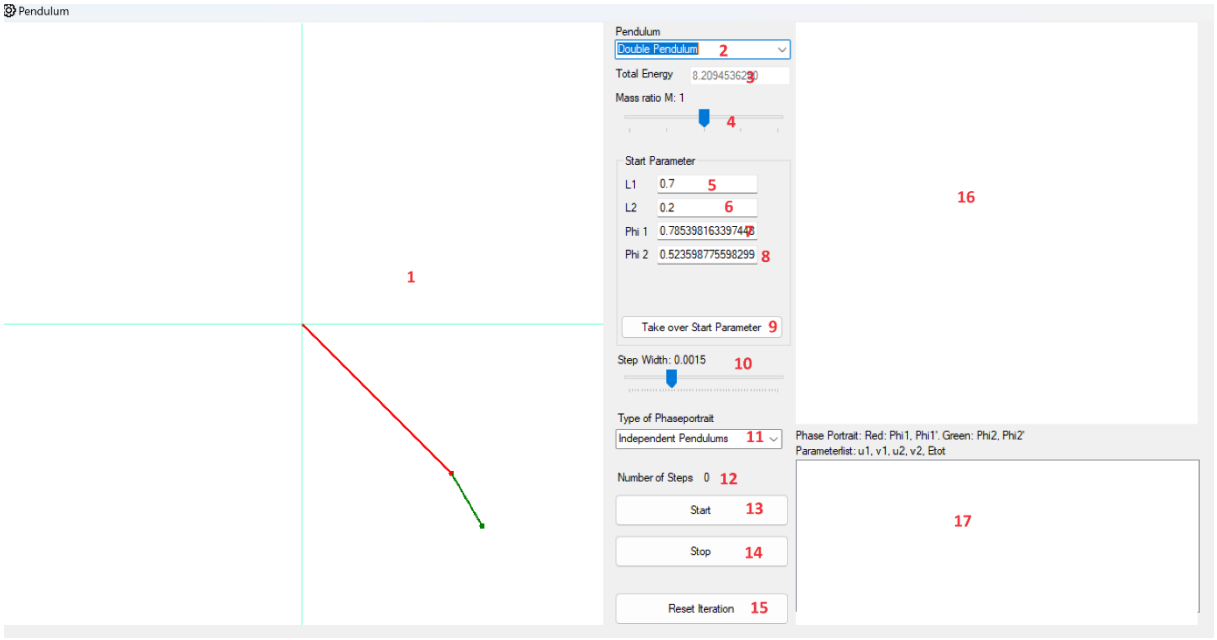
Usually, the investigation of the reflection angle α is more meaningful than the parameter *t* that defines the hit point. In case of a circle or an orbit out of the ellipse focuses, the orbit of *t* lies dense in the allowed interval, except one has a periodic orbit. On the contrary, α is constant in case of the circle or oscillates in a particular interval of [0, π [. It is recommended to experiment with the start value between 1/12 and 11/12 to get interesting images. The mathematical interpretation of these images is maybe more difficult.

4.4. Coupled pendulums

The following coupled pendulums are offered for examination here:

- Double pendulum
- Oscillating spring pendulum
- Horizontally shaked pendulum

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



Window for analysing coupled pendulums

1

The movement of the coupled pendulum is shown here.

2

Selection of the pendulum.

3

Display of the total energy at the start.

4

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

5 - 8

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

9

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

10

Step size of the Runge Kutta method.

11

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

12

Display of the number of steps in the iteration.

13

Start of the iteration.

14

Stop the iteration.

15

The iteration is reset here.

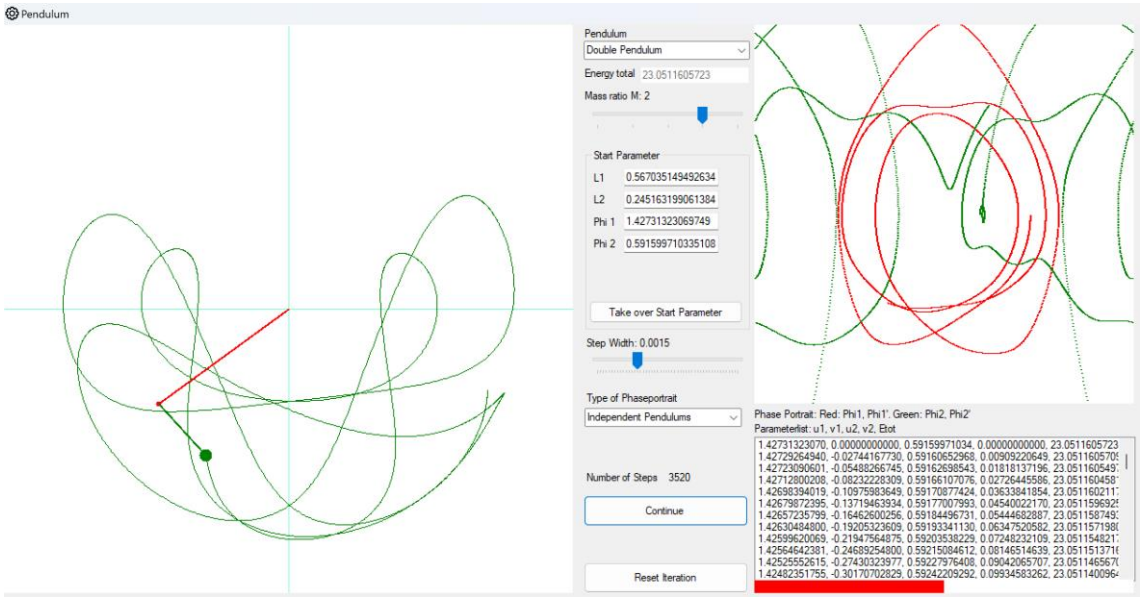
16

Display of the phase portrait.

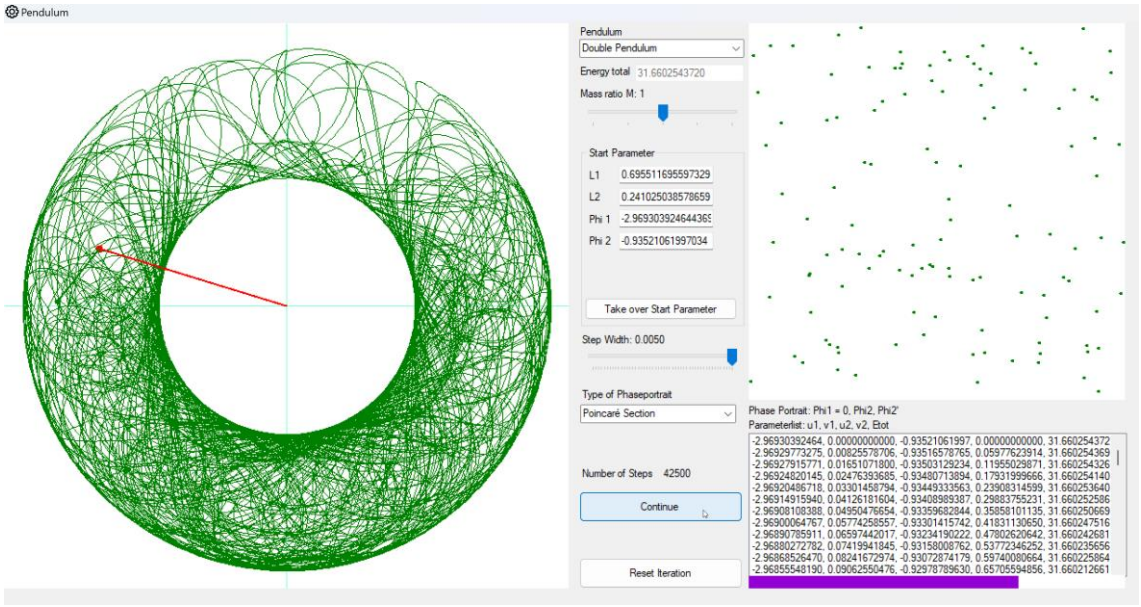
17

Log of the relevant parameters and display of the current energy relative to the total energy at the start.

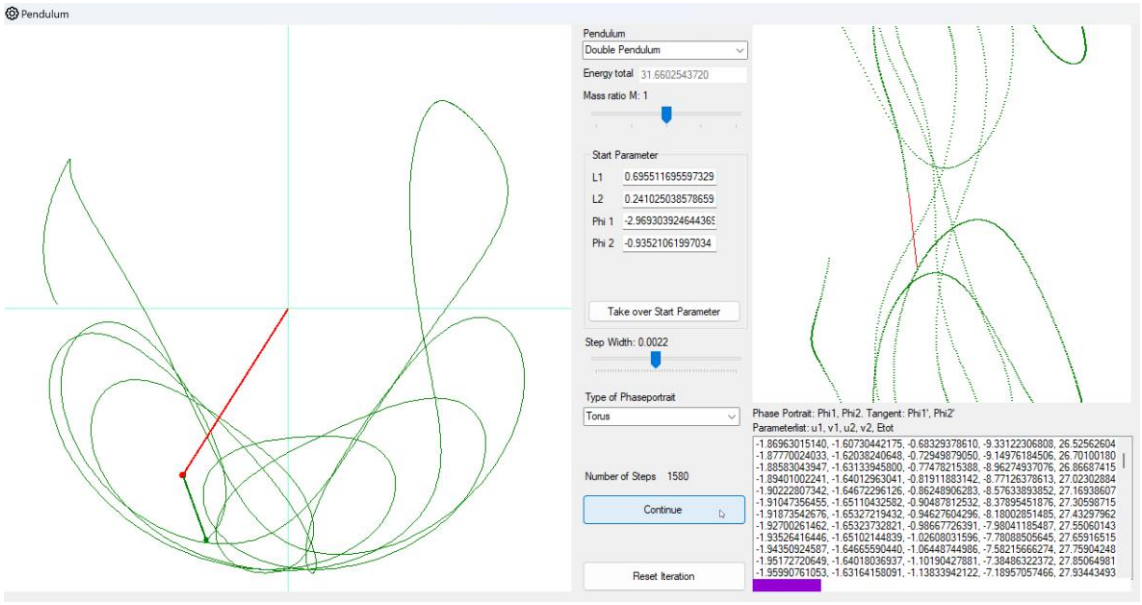
The Runge Kutta method (and also better methods) only shows the "real" movement of the coupled pendulum over a few steps. Subsequently, the pendulum reacts too sensitively to the inaccuracies of the iteration. For this reason, the current energy is displayed if it deviates from the total energy at the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



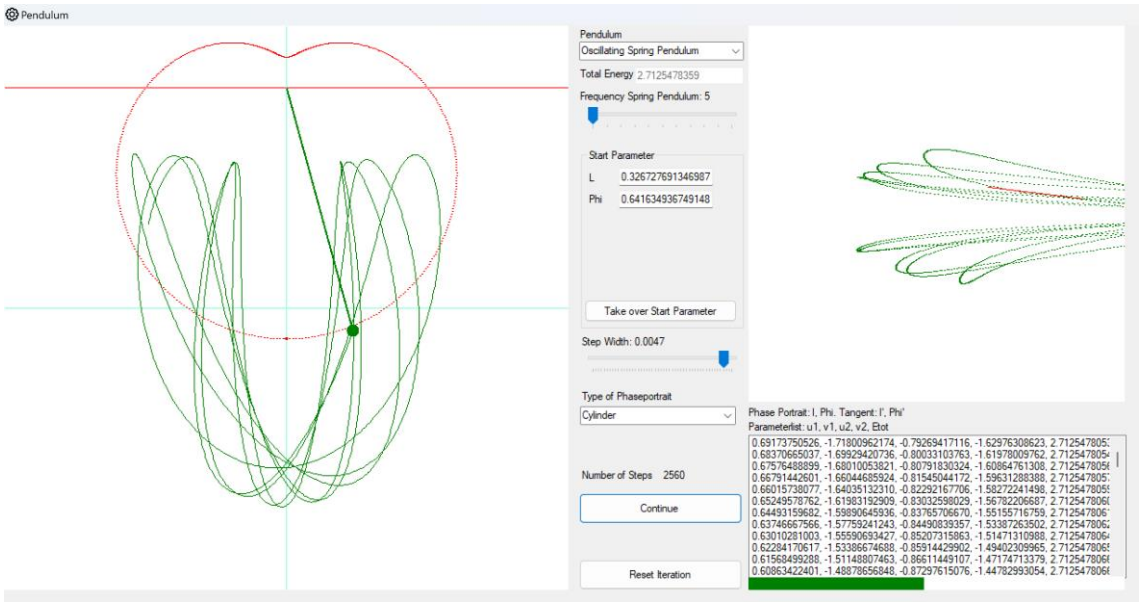
Movement of the double pendulum



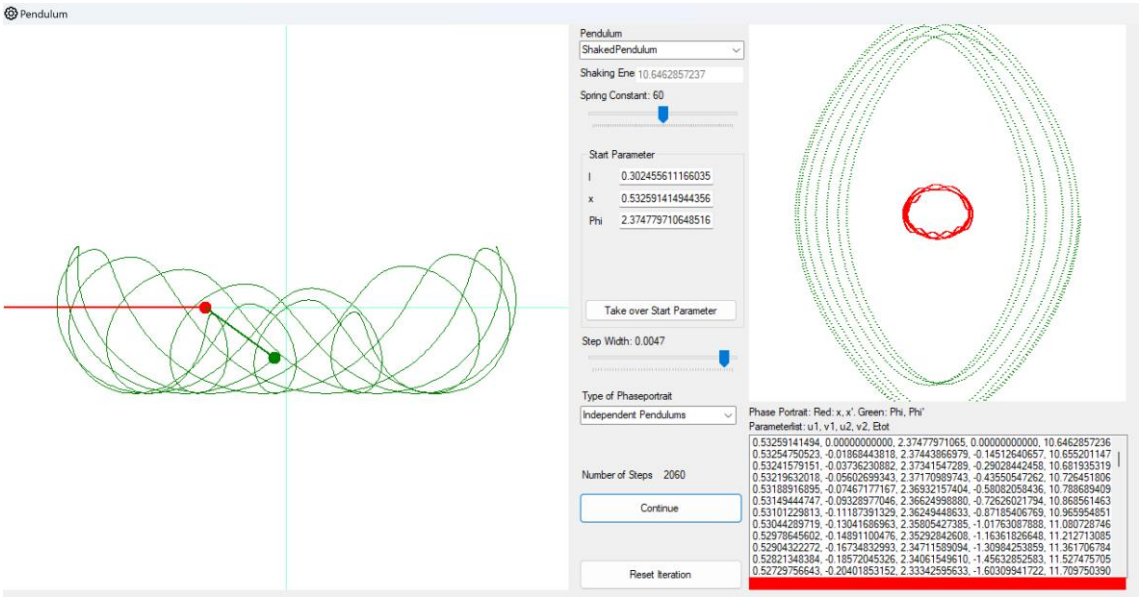
Poincaré cut on the double pendulum



Visualisation of the movement on the torus



Oscillating spring pendulum



Shaked pendulum

5. Complex Iteration

5.1. Newton Iteration

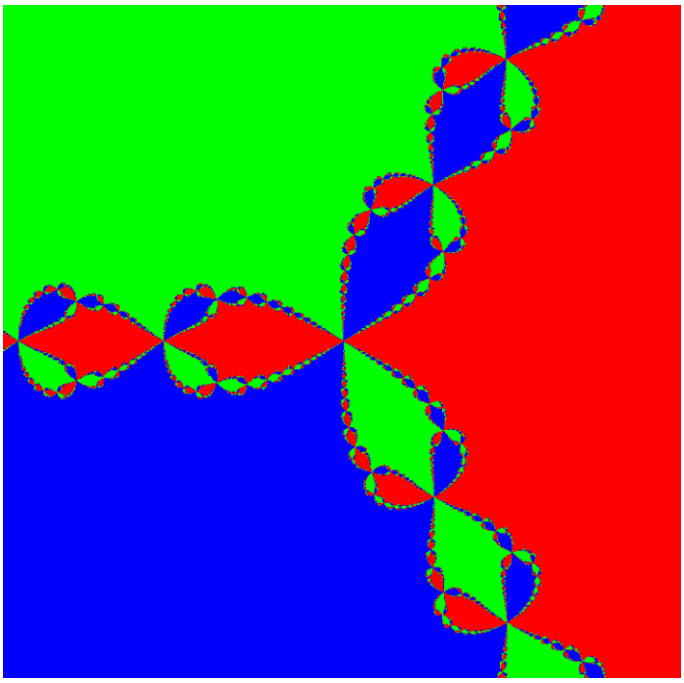
The Newton iteration method for approximating the zeros of real functions can be derived on an elementary basis in secondary school or at least explained graphically. This method can be extended to the complex plane. This is described in the mathematical documentation. It is less about effectively finding 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 ζ during the iteration, we also paint it red. z_0 then lies in the basin of ζ. If there are several zeros, we give each zero a colour and then paint the starting points with the respective colour of the zero that they converge to. Points that show no convergence behavior after a certain number of steps or that converge towards ∞ 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 superattractive 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.

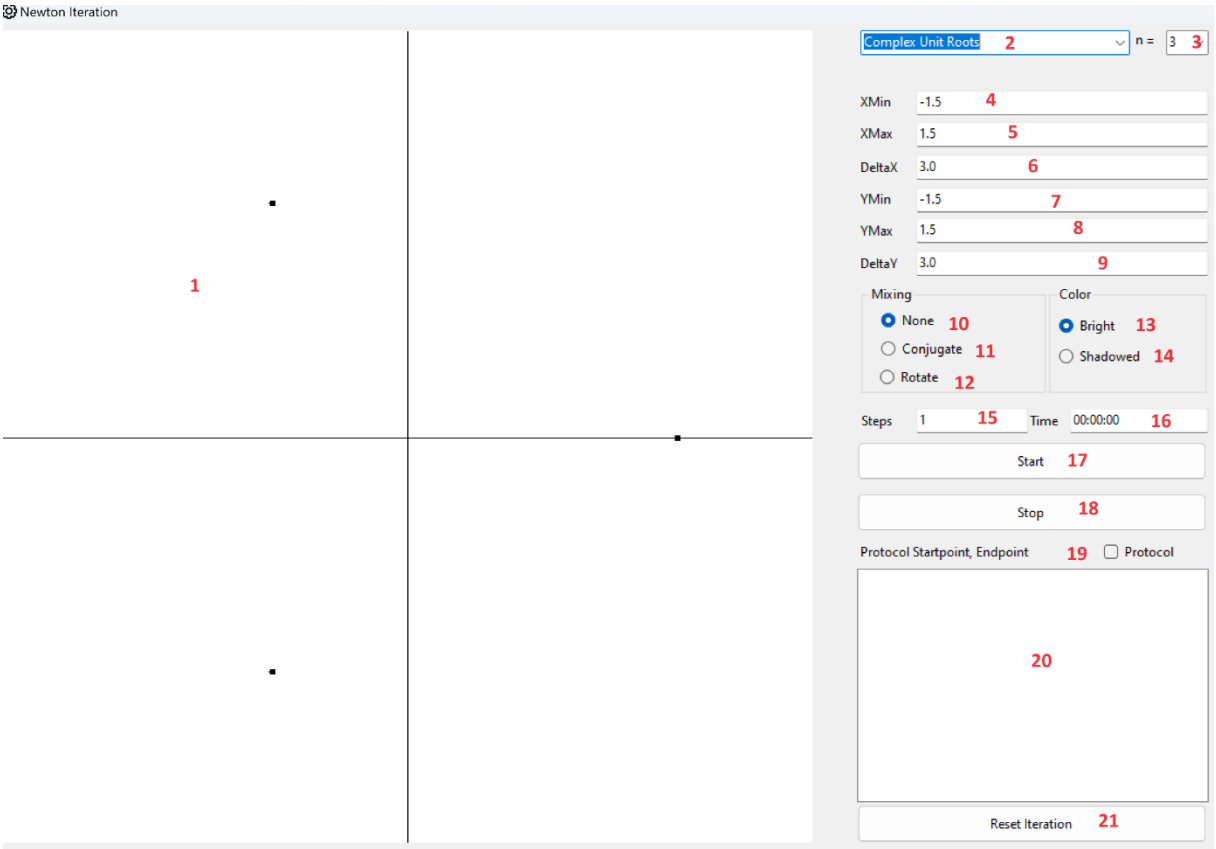
Among other things, the program can be used to examine the complex roots of unity, i.e. the zeros of the polynomial $p(z) = z^n - 1, n \in \mathbb{N}$ and their basins can be examined. In the program, n is a natural number between 2 and 12. The following figure shows the basins of the roots of unity 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

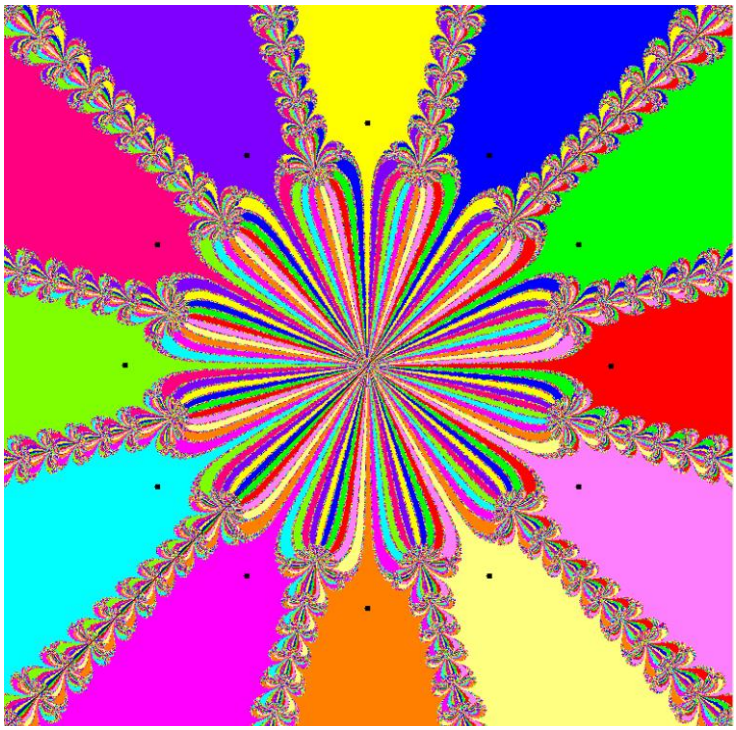
1

Drawing area or section of the complex plane.

2

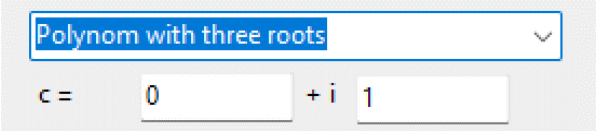
Select the polynomial whose zeros are to be analysed. You can choose from the following:

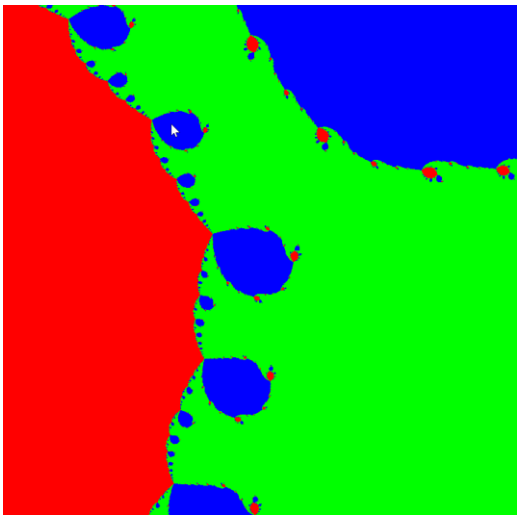
- Complex roots of unity where the degree can be selected between 2 and 12
- polynomial with three zeros, namely $p(z) = (z-1)(z+1)(z-c)$ where *c* can be defined by the user
- Complex roots of unity after inversion on the unit circle. This allows the investigation of the behaviour at infinity



Basins of the twelfth unit roots

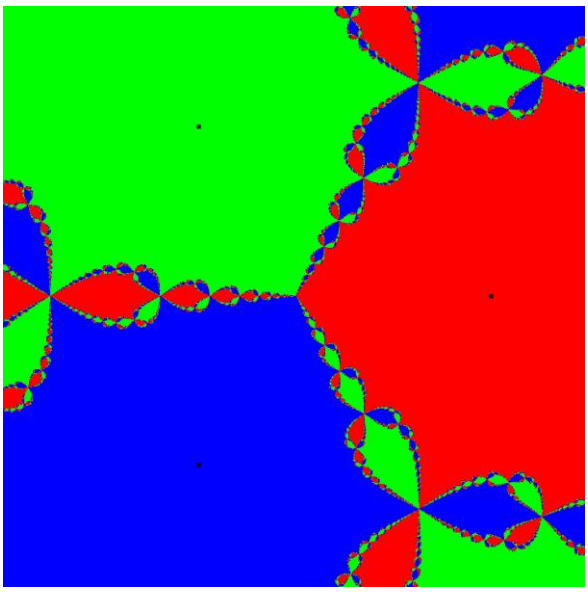
If the option "Polynomial with three zeros" is selected, a field appears below the selection box in which *c* can be entered:





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

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

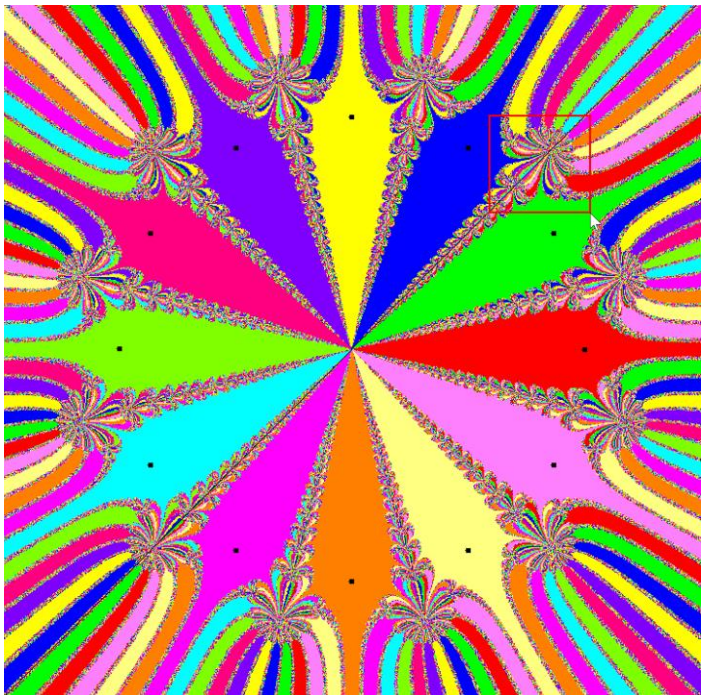
3

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

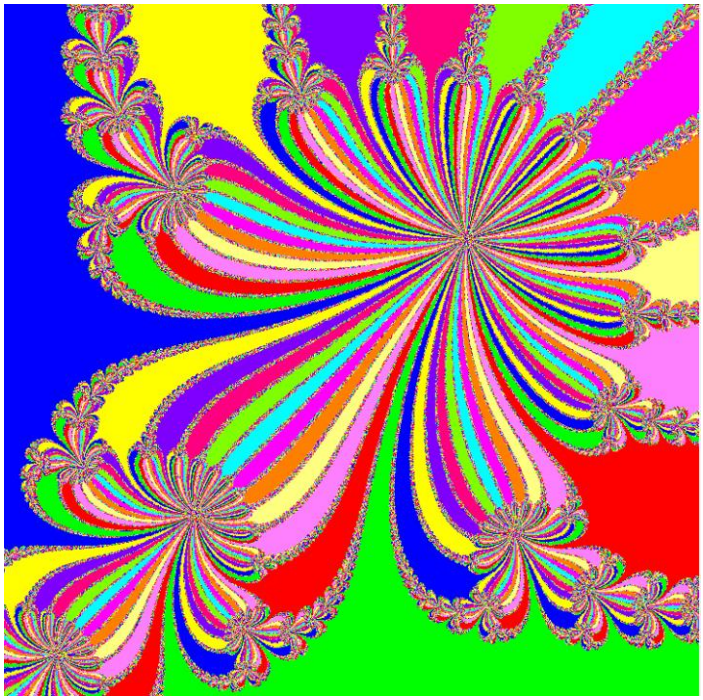
4, 5, 6, 7, 8, 9

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

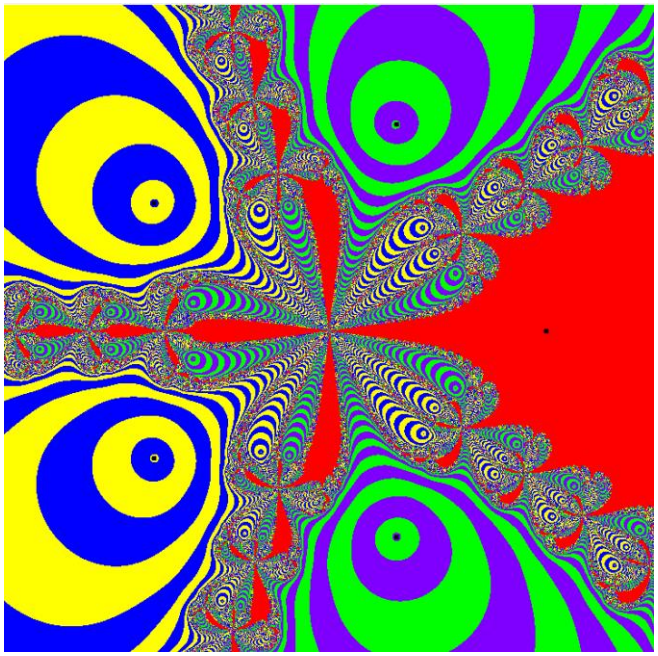
Since the image has fractal properties, the zoom can be continued infinitely within the calculation accuracy and similar images always appear

10, 11, 12

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

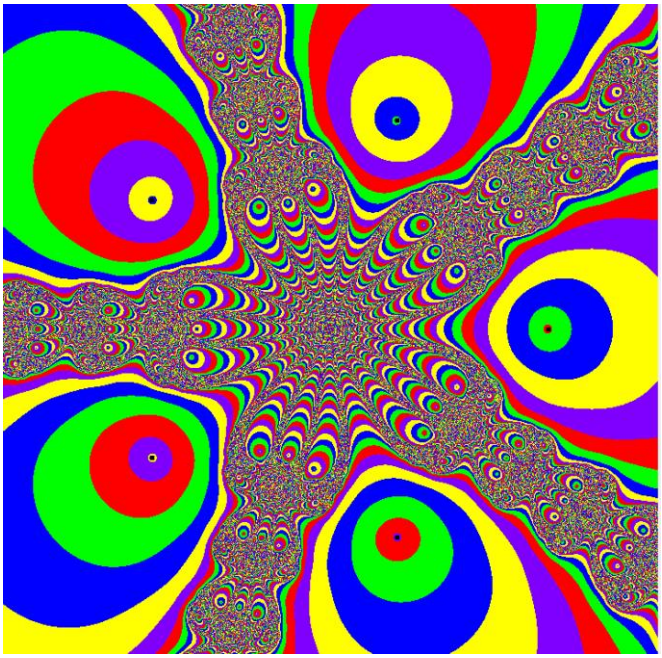
- No mixing: you can see the original and real basins
- Conjugation: Here, the iterated z_n is conjugated at each iteration step. As a result, basins that are symmetrical to the x-axis are "visited" alternately during the iteration and the

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.



Conjugate representation of unit roots of degree five

- Rotation: Same idea, but the iteration point is rotated at each iteration step by the angle $2\pi/n$.



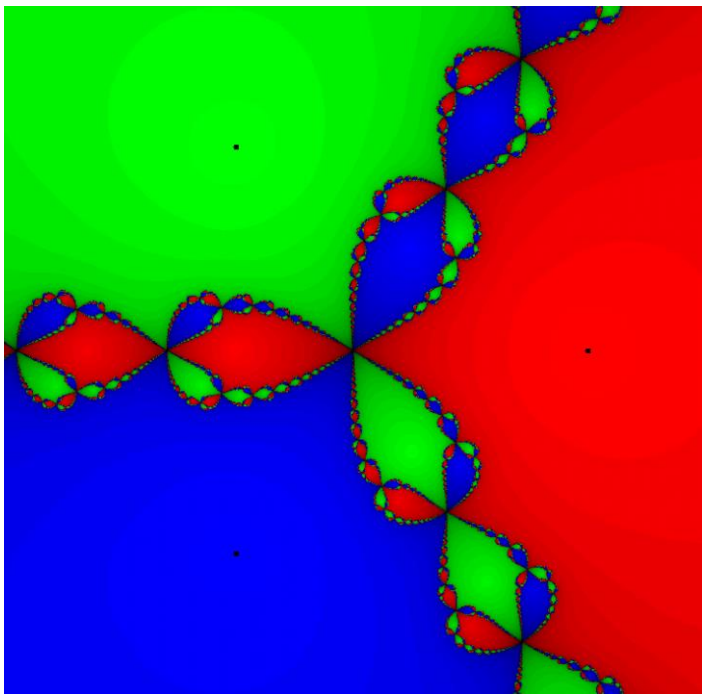
Unit roots of degree five including rotation

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 centre with each additional rotation, until after five steps you end up back in the basin of $\zeta = 1$ again after five steps.

13, 14

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.
- Shaded: Points that converge more slowly are given the colour of the basin to which they belong, but slightly darker than points that converge quickly.



Shaded variant of the unit roots of degree three

15, 16

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

17

The iteration is started or continued here.

18

The iteration is stopped.

19

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.

20

Display of the protocol.

21

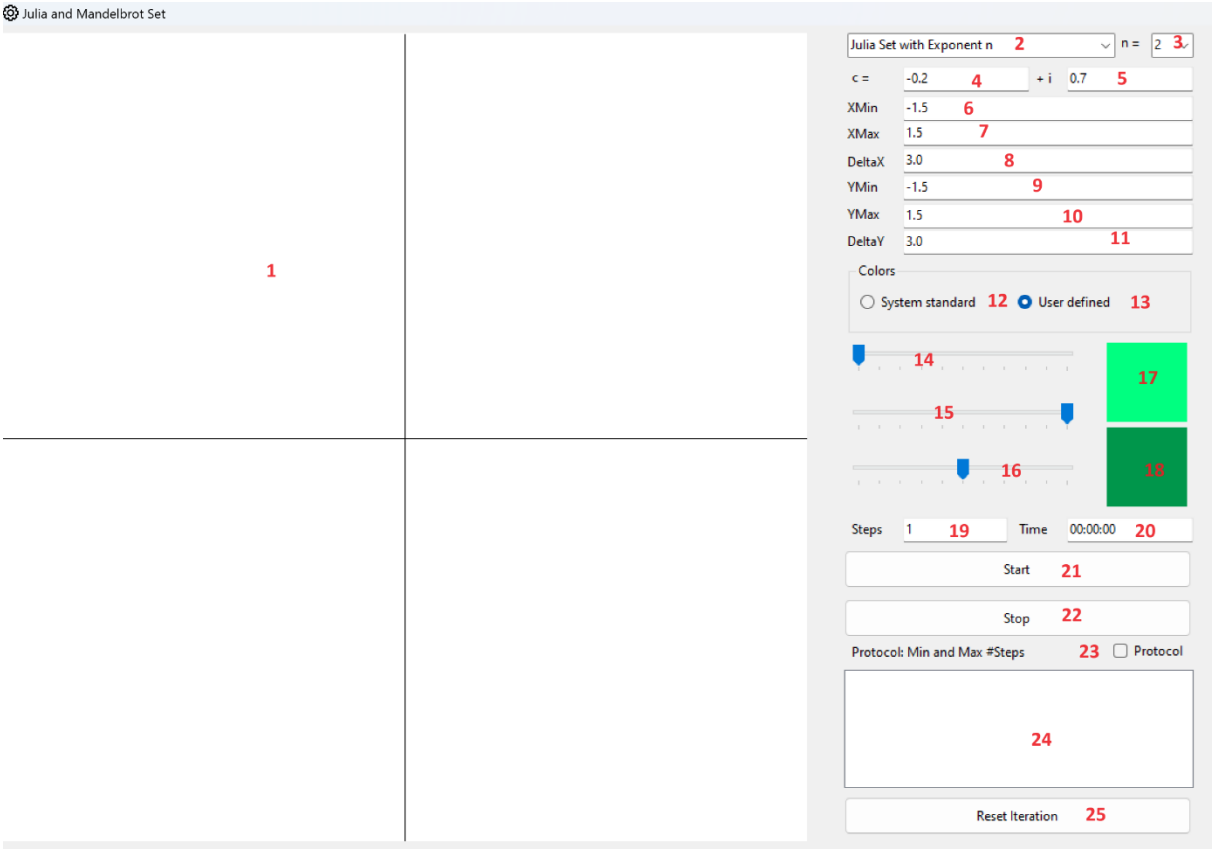
The entire iteration is reset here.

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 behavior 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 point sequence 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

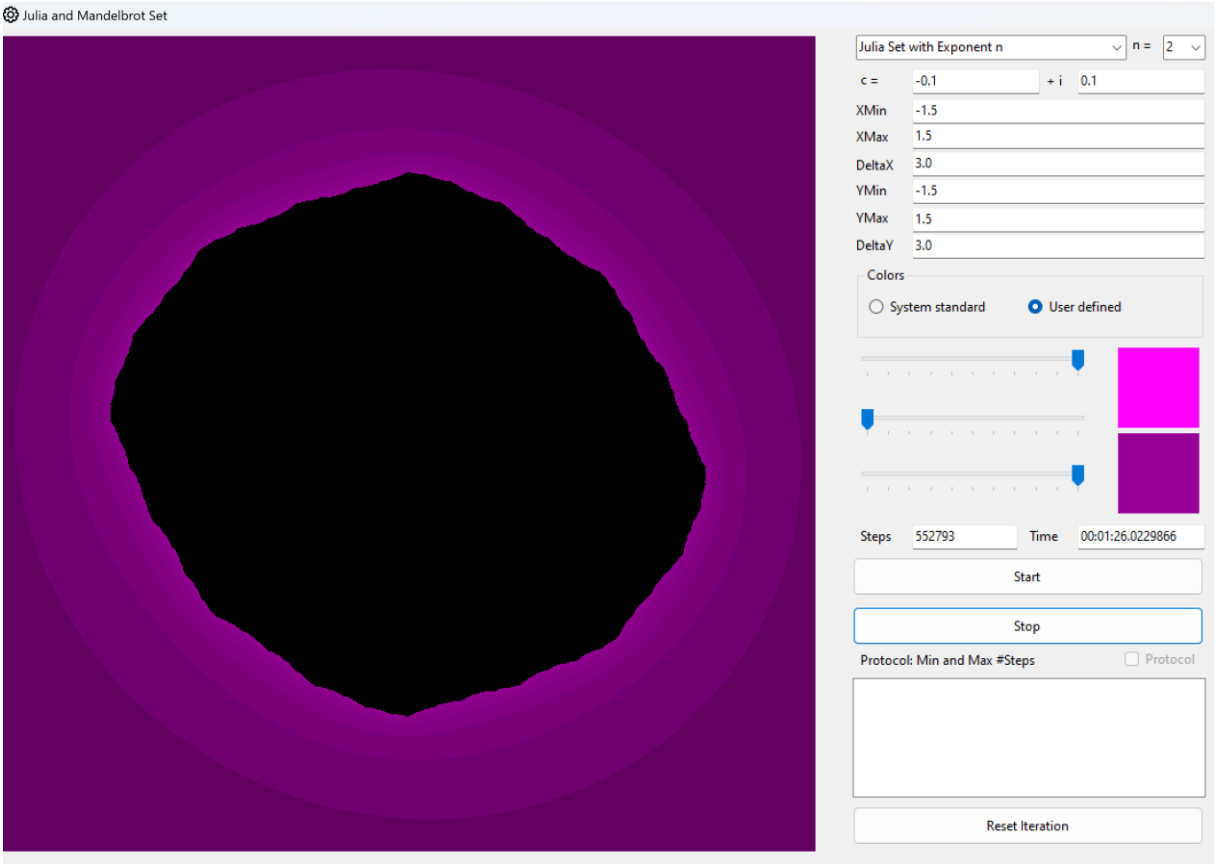
1

Drawing area and section of the complex plane

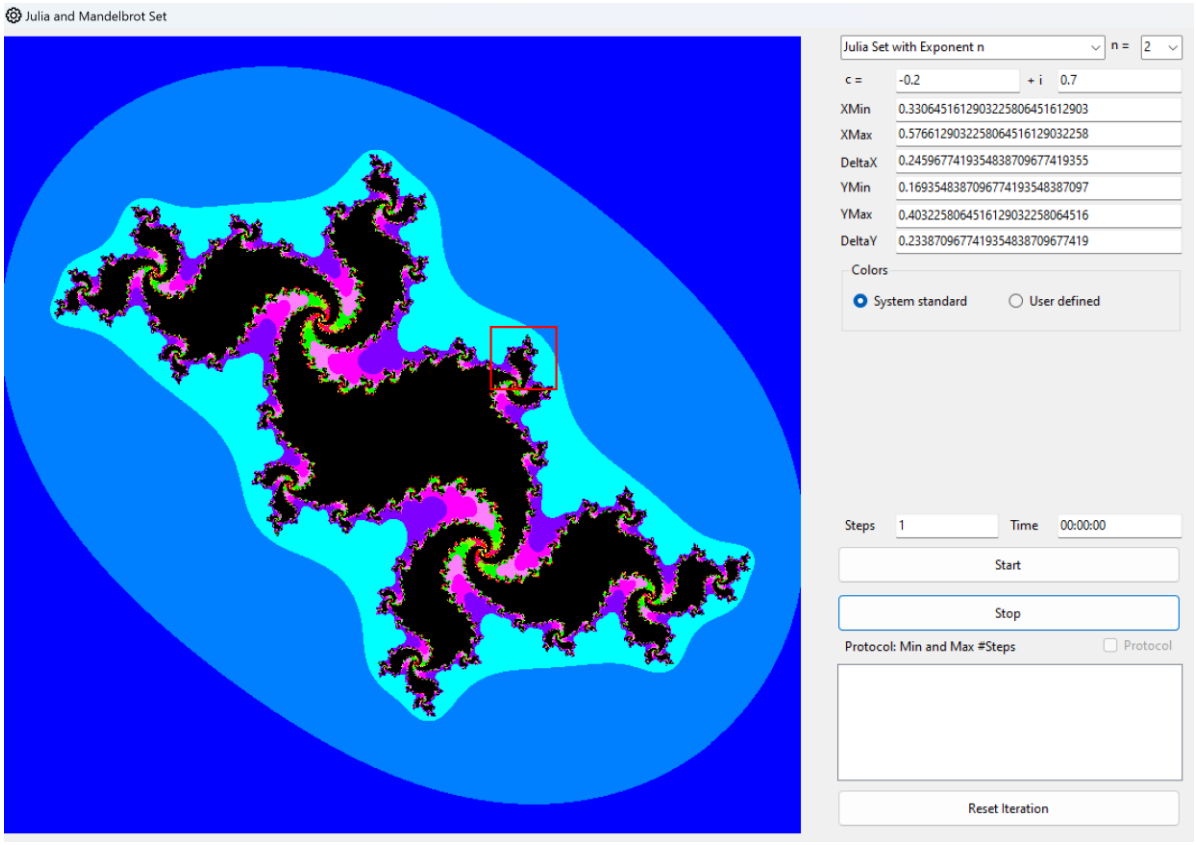
2

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 it is, the more iteration steps it takes, before it is clear

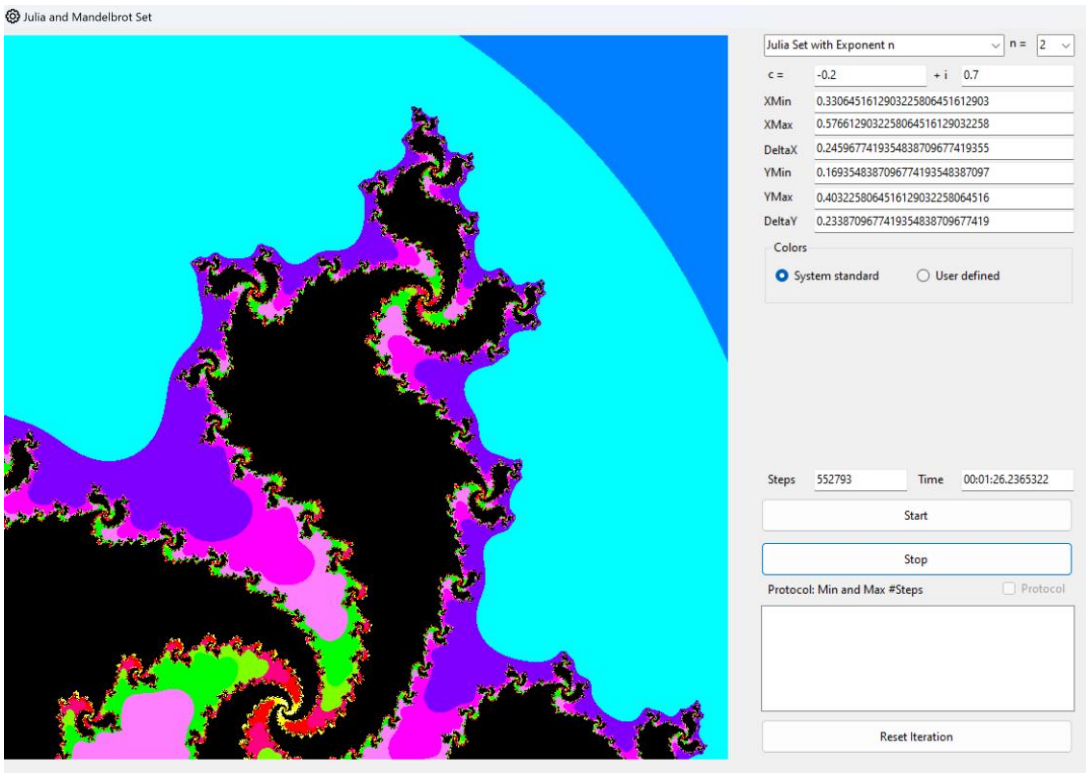
that it is moving in the opposite direction, the lighter the colour. ∞ strives. Criterias: See mathematical documentation.



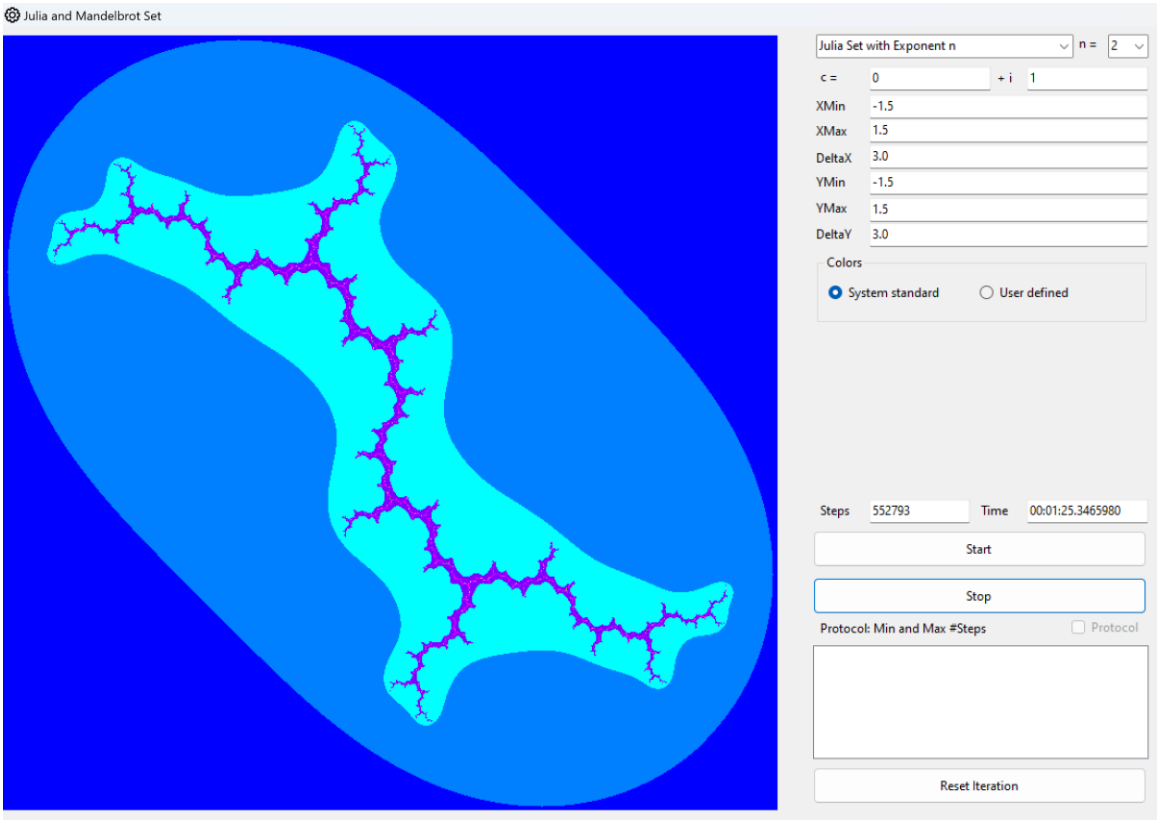
Julia Menge during a minor "disturbance" $c = -0.1 + 0.1i$. Opposite $c = 0$ the "unit circle" is a little shriveled. The color is user-defined.



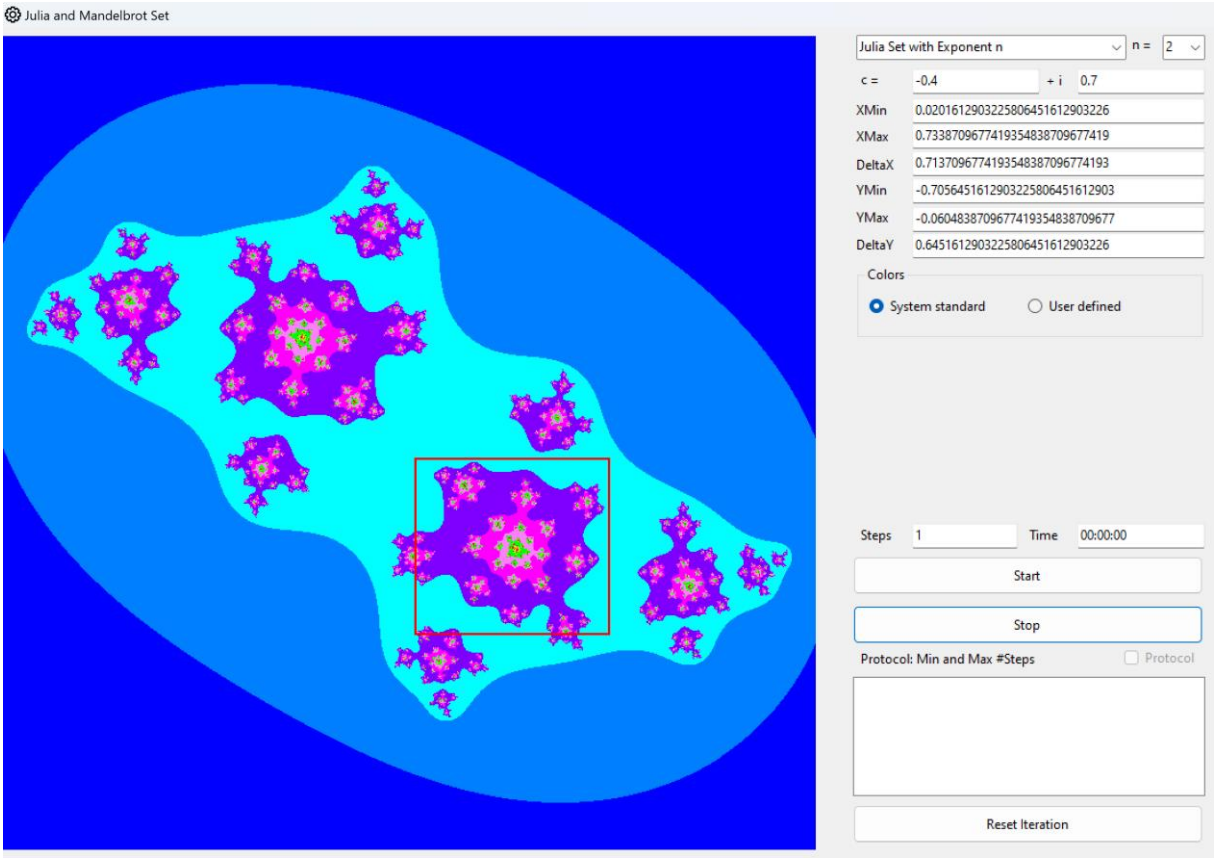
With $c = -0.2 + 0.7i$ the "disturbance" is greater. The unit circle has turned into a fractal and is unrecognizable. The standard colours of the system were used. Furthermore, a selection was made by holding down the mouse button, which is examined below.



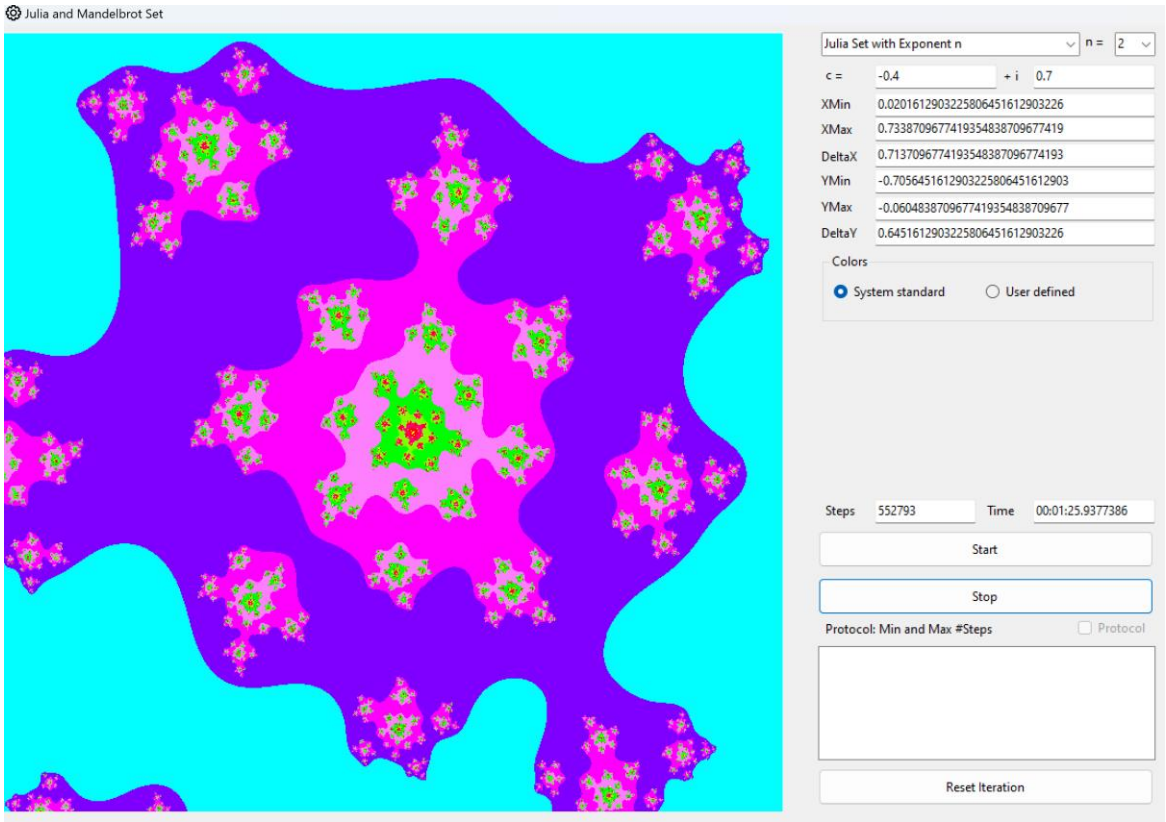
Julia set in the selected detail.



Julia set for $c = i$. It is now just a thin, ramified structure.



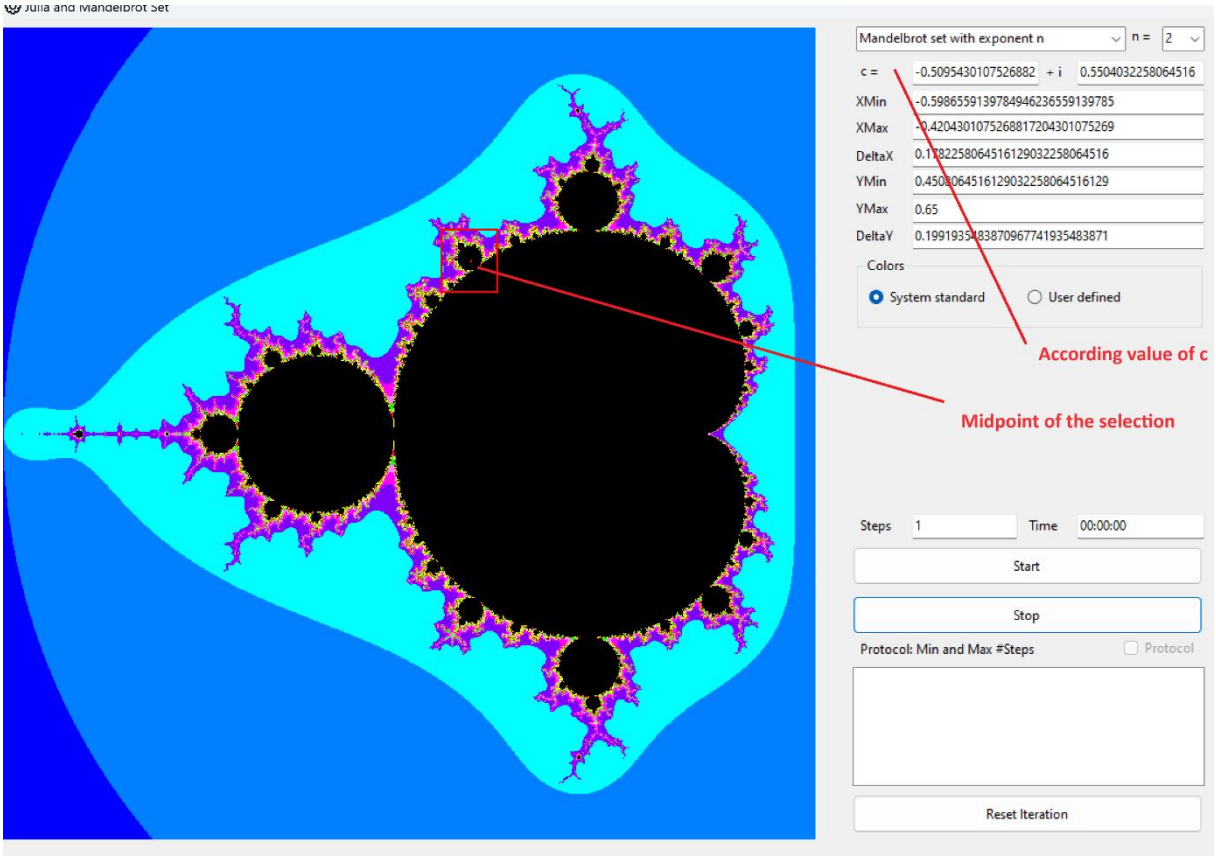
The Julia set for $c = -0.4 + 0.7i$ disintegrates into isolated clusters of points - into "dust".



Enlarged section of the previous image.

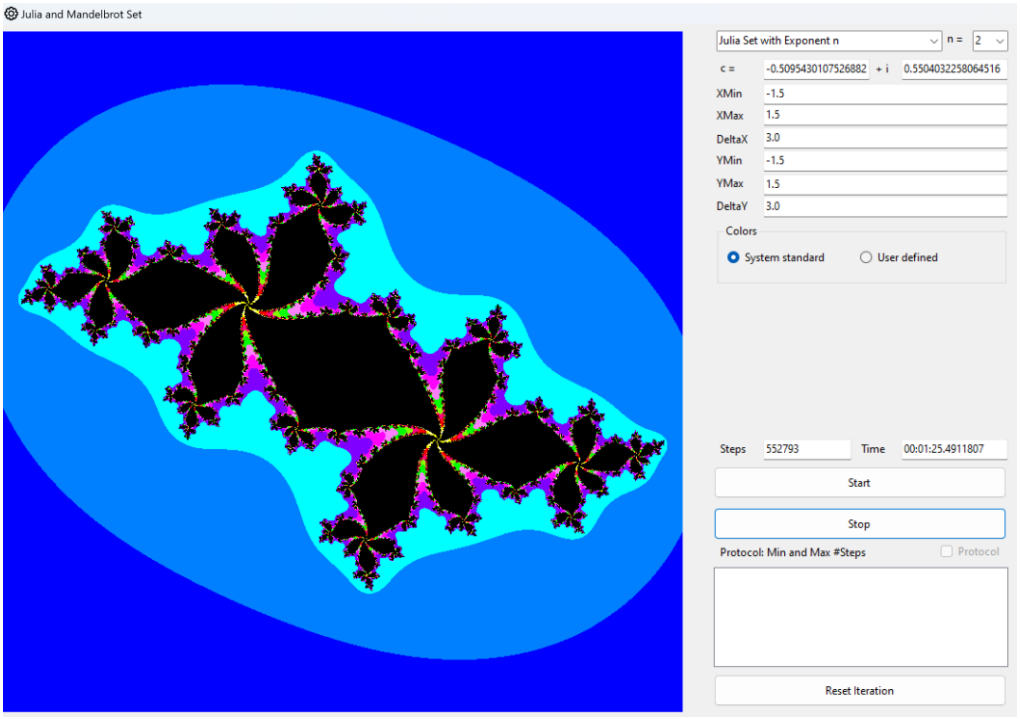
How can we get an idea of the *c* for which the Julia set still contains inner points? The Mandelbrot set provides information about this. The fact is that every basin of an attractive fixed point or attractive cycle of the Julia set contains at least one critical point, i.e. a point with $f'(z) = 0$. 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 $c \in \mathbb{C}$ and shows the behaviour 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 this 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.



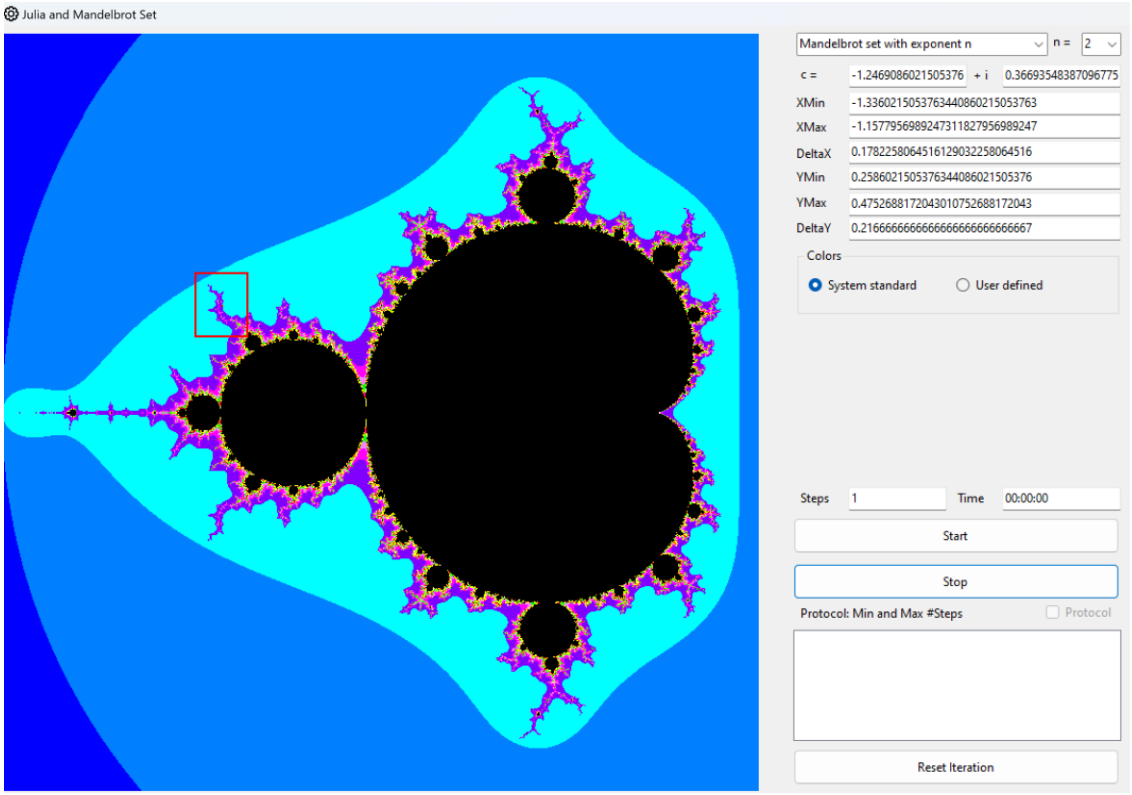
The Mandelbrot set above. If you select a sub-area with the mouse button held down, the corresponding coordinates are displayed in the XMin, XMax, YMin, YMax fields. The value of *c* is then set according to the centre of the selection.

If you now switch back to the Julia set via the combo box at the top right, the c value remains unchanged. The Jule set belonging to this *c* is then generated:

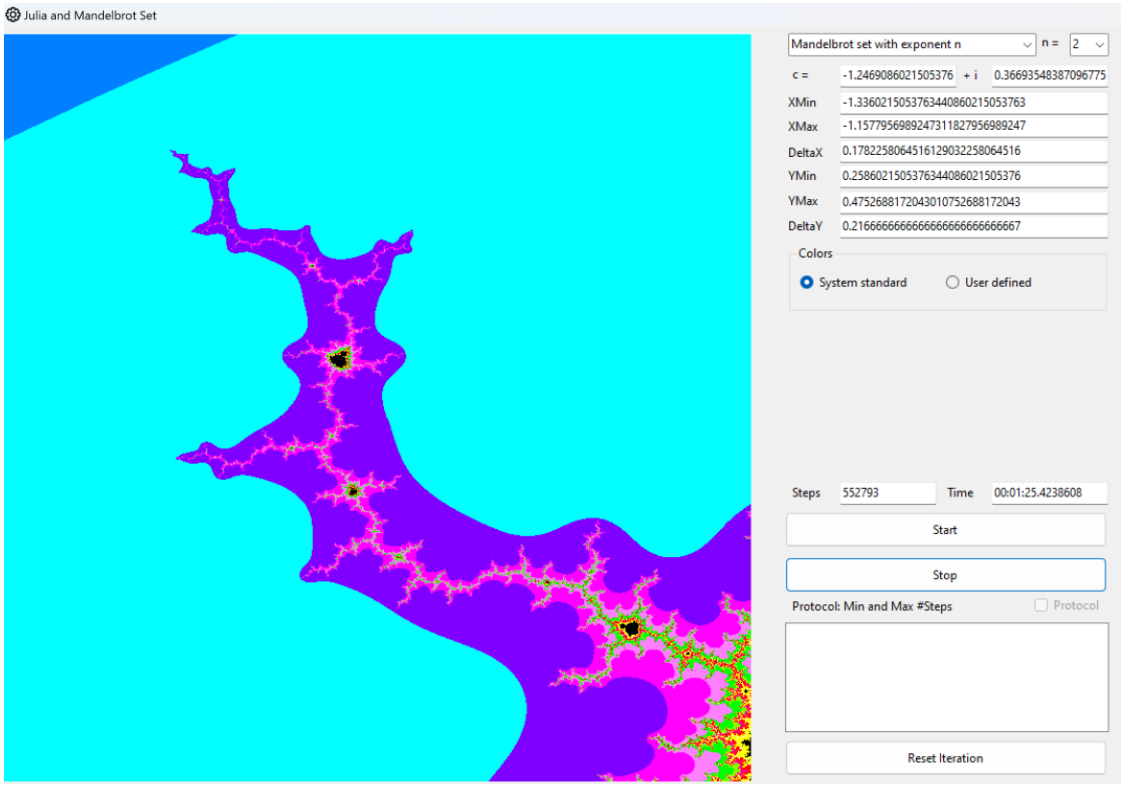


Julia set belonging to the previous c value

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.



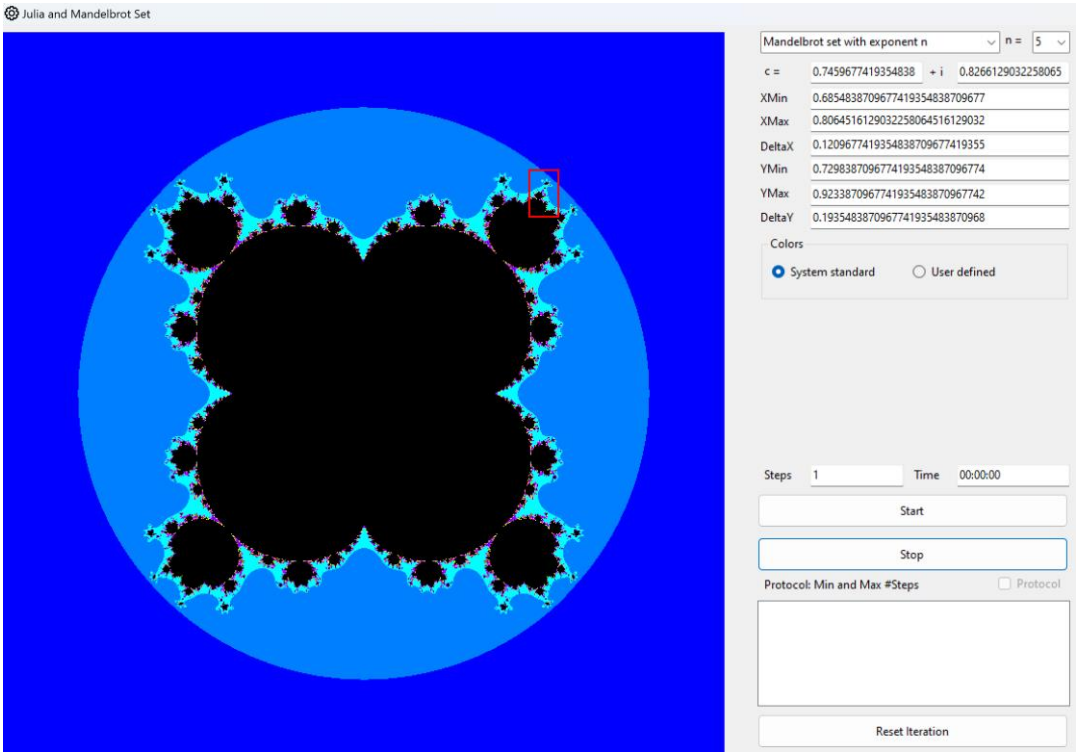
Once again the Mandelbrot set with a selected small section.



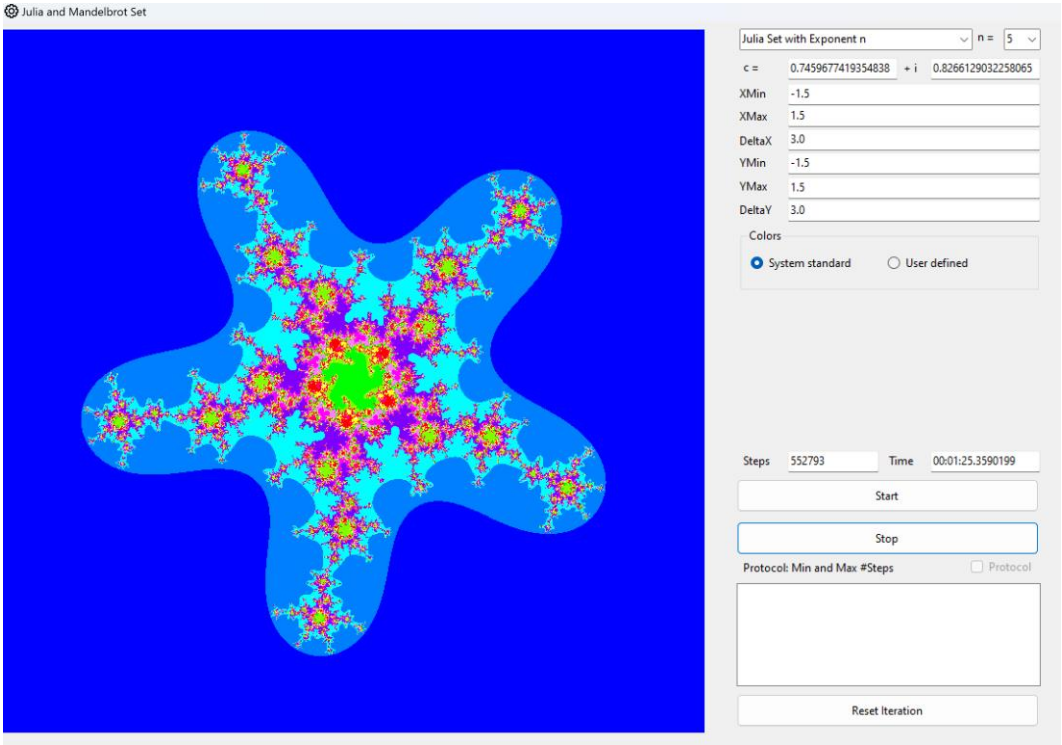
In this excerpt, specimens of the Mandelbrot set appear again

3

Choice of the exponent n. 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 $n = 5$ and a selection made, including the value of *c*.



Julia set for $n = 5$ with the previously selected value of *c*.

4,5

Specification of the "disturbance parameter" *c*. This can be set by holding down the mouse button and selecting it in the Mandelbrot set or entered manually.

6, 7, 8, 9, 10, 11

If a section of the generated image is selected by holding down the mouse button, 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* remains and you can generate the corresponding Julia set.

12

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.

13

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, 15, 16

Controller for the composition of the user colour from the basic colours red, green and blue.

17

Display of the user colour in a light tone.

18

Display of the user colour in a dark tone.

19, 20

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

21

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.

22

Presentation of the protocol.

23

Start or continue the iteration.

24

Stop the iteration.

25

The iteration is reset.