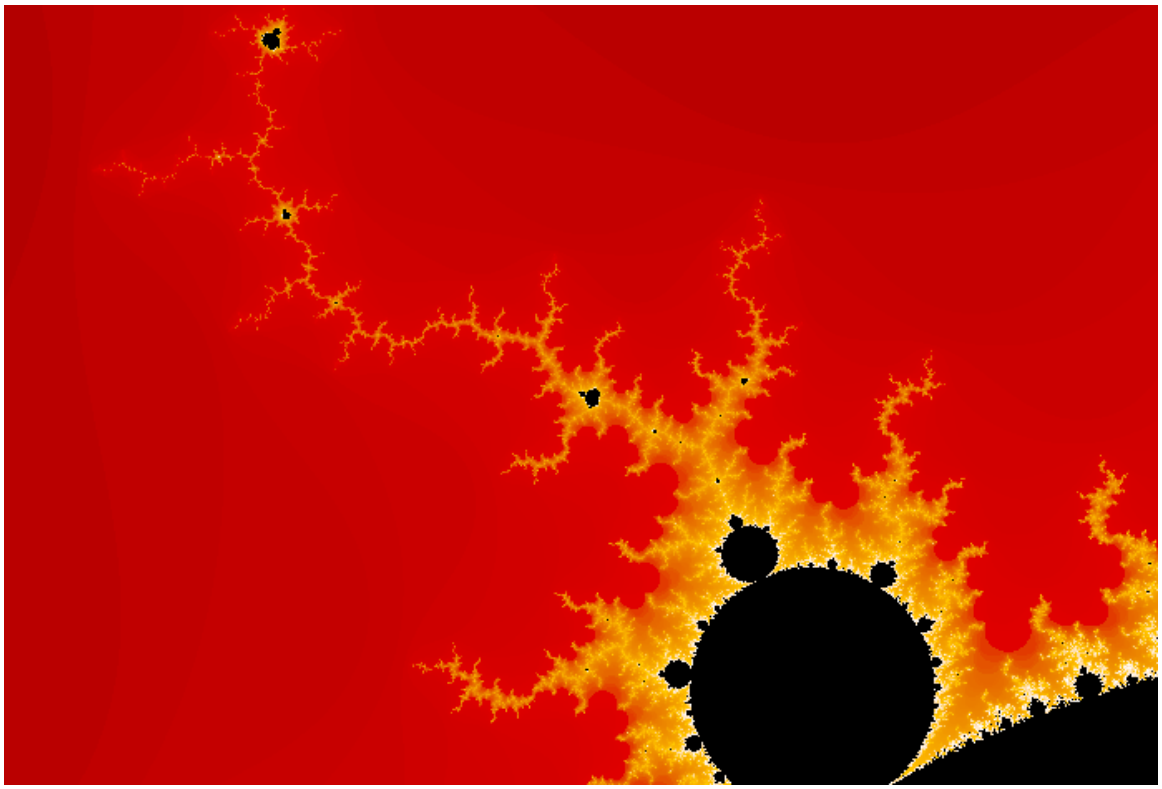


MathTools

This manual is valid for all versions 1.5.*

TigerGraphics

2025-05-10

Table of contents

Preface **6**
Licence . . . 6
Python Licence . . . 6

Introduction **7**

1 Installation **8**
1.1 Linux Installation . . . 8
1.1.1 Requirements . . . 8
1.2 Windows Installation . . . 9
1.3 MacOS Installation . . . 9
1.3.1 Requirements . . . 9
1.3.2 Download . . . 9

2 Start and Customizing **10**
2.1 The Main Widget . . . 10
2.2 Custom . . . 11
2.3 Help . . . 12

3 Fractals **13**
3.1 Calculate . . . 14
3.2 Parameter settings for the different fractals . . . 14
3.2.1 Highway Dragon . . . 14
3.2.2 Weierstrass Monster . . . 14
3.2.3 Sunflower . . . 14
3.2.4 PS print/Print . . . 14

4 Mandelbrot **15**
4.1 Mandelbrot's Set . . . 15
Live Julia . . . 17
4.2 Newton roots . . . 17
4.3 Collatz' Fractal . . . 17
4.4 Common widget functions . . . 17
4.4.1 Zooming . . . 17
4.4.2 Reset . . . 18
4.4.3 Calculate . . . 18
4.4.4 Reset . . . 18
4.4.5 Save & Print . . . 18

5 Slope **19**
5.1 One Dimensional differential equation . . . 19
5.2 Two Dimensional differential equation . . . 20
5.3 Parameters and Built in Math functions . . . 20
5.4 Linear Model . . . 20
5.5 Eigenrichtung . . . 20
5.6 Reload . . . 20
Python dependent functions . . . 20
5.7 Calc FP . . . 21
5.8 Make Linear . . . 21
5.9 Normalize . . . 21
5.10 Resolution . . . 22
5.11 Endtime . . . 22
5.12 Delay . . . 22
5.13 Solver Parameters . . . 22
5.13.1 Discrete . . . 22
5.14 Isoclines . . . 22
5.15 Scale . . . 23
5.16 ReSetScale . . . 23
5.17 Autoscale . . . 23
5.18 PosScale . . . 23
5.19 Trajectories . . . 23
5.20 Multi-Trajectories . . . 24
5.21 Check Buttons . . . 24
Time Plot . . . 24
Show Coor. . . . 24
Show Points . . . 24
AutoScale . . . 25
5.22 Load, Save & Edit . . . 25
5.23 PS print/Print . . . 25

6 Celluar Automata **26**
6.1 Life . . . 27
6.2 Universe according to Stephan Wolfram . . . 28
6.3 Disease . . . 29
6.4 Per Bak's sandpile model . . . 29
6.5 Circular Room . . . 30
6.6 Bug Spread . . . 30
6.7 Diffusion . . . 31
6.8 Advection . . . 31
6.9 Leopard . . . 32
Equations . . . 32
Parameters . . . 32
6.10 Common Settings . . . 33
6.10.1 Parameter Input . . . 33
6.10.2 Color Table . . . 33

6.10.3 Cells per row . . . 33
6.10.4 View of the world . . . 33
6.11 Common Functions . . . 33
6.11.1 Show . . . 33
6.11.2 Run . . . 34
6.11.3 Step . . . 34
6.11.4 Interrupt . . . 34
6.11.5 Clear . . . 34
6.11.6 Delay . . . 34
6.12 Setting the initial states . . . 34
6.12.1 Single Cells . . . 34
6.12.2 Randomfill . . . 34
6.13 Load & Save . . . 35
6.14 PS print/Print . . . 35

7 IFS - Iterated Function Systems 36
7.1 Calculate . . . 37
[Copy machine] switched off . . . 37
[Copy machine] switched on . . . 37
Reset . . . 37
Clear screen . . . 37
7.2 PS print/Print . . . 37
7.3 Set IFS . . . 38
7.4 Set Parameters . . . 38
7.4.1 Load & Save . . . 39

References 40

8 Parabola 41
8.1 Calculate . . . 41
8.2 Parabola of a single parameter r . . . 42
8.3 PS print/Print . . . 43
8.4 Miscellaneous . . . 43

9 Plotter 44
9.1 Plot . . . 45
9.2 Scale . . . 45
9.3 ReSetScale . . . 45
9.4 Autoscale . . . 45
9.5 PosScale . . . 45
9.6 Clear . . . 45
9.7 Coordinates . . . 45
9.8 Check Buttons . . . 46
Show Coor. . . . 46
AutoScale . . . 46
9.9 Load, Save & Edit . . . 46

9.10 PS print/Print 46
9.11 Grace print 46

Preface

This manual is newly editied in Quarto as a *book* project and completely written in *Markdown*.

The **MathTools** are free software under the following licence.

Licence

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see GPL.

Python Licence

We refer to the PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2 as published at Python.

Introduction

The **MathTools** are a set of educational tools for the visualization of several nice mathematical things.

The **MathTools** is built on the base of *Tcl/TK* version 8.6 and *GNU-C* version 7.5.
It is developed and tested on OpenSuSE Linux Version 15.6 and is ported to run also to Microsoft Windows and MacOS platforms, were 64-bit architectures are assumed.

1 Installation

The installation of **MathTools** requires some preparations that are different for the operation systems they should be use on.

1.1 Linux Installation

1.1.1 Requirements

Make sure that at least *gcc*, *make*, *tcl* and *tk* are installed on your system.
Furthermore make sure that **Python** at least in the version 3.10 with package **SymPy** is available on your system, either by system wide installation or locally for your user, which is supported by **Python**. Please read the according **Python** installation information.
Python is required for the linearisation, normalization of differential equation systems and for isocline computation in the Slope module.

1.1.1.1 Unpacking

Copy the file *mathtools.tgz* to the directory where the software should be installed.
Extract the data from the *.tgz*-file by
`tar -zxvf mathtools.tgz`
The archive is unpacked to the directory *MATHTOOLS*.

1.1.1.2 Compiling

The **MathTools** are running on 32-bit and 64-bit sytems. Thus, the binaries need to be built after unpacking. Go to the directory *MATHTOOLS/src* by f.e.
`cd MATHTOOLS/src`
and execute the script *install_all* from the command line
`./install_all`
All the Makefiles in the subdirectories will be executed.

1.1.1.3 Starting

Go to the directory *MATHTOOLS*.
The main script `mft.tcl` is executable from the command line and can be referenced by any application starter.

1.2 Windows Installation

Execute the file `mathtools-setup.exe` and follow the instructions.
The program can be started from the menu in the application group *TigerGraphics* or from the desktop if the option for a desktop icon has been selected during installation.
Also you should install **Python** from Python at least in the version 3.10.
Furthermore make sure that the package **SymPy** is installed with your **Python**. Please read the according **Python** installation information from Python.

1.3 MacOS Installation

1.3.1 Requirements

On your Mac you need to obtain the *Tcl/Tk* package from ActiveState according to your desired licence model.
Also you should install **Python** from Python at least in the version 3.10.
Furthermore make sure that the package **SymPy** is installed with your **Python**. Please read the according **Python** installation information from Python.

1.3.2 Download

Download the appropriate application for your architecture (*x86* or *arm*). Unpack the archive by double clicking on it and finally copy the application container *MFT.app* either to the systems application folder *Programs* or to any other desired location and simply start the application like any other one.

2 Start and Customizing

2.1 The Main Widget

The **MathTools** main widget serves to switch to the different groups of tools described in the following sections.

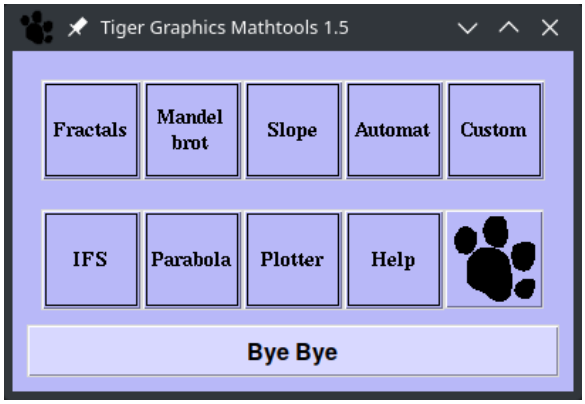


Figure 2.1: The main widget

2.2 Custom

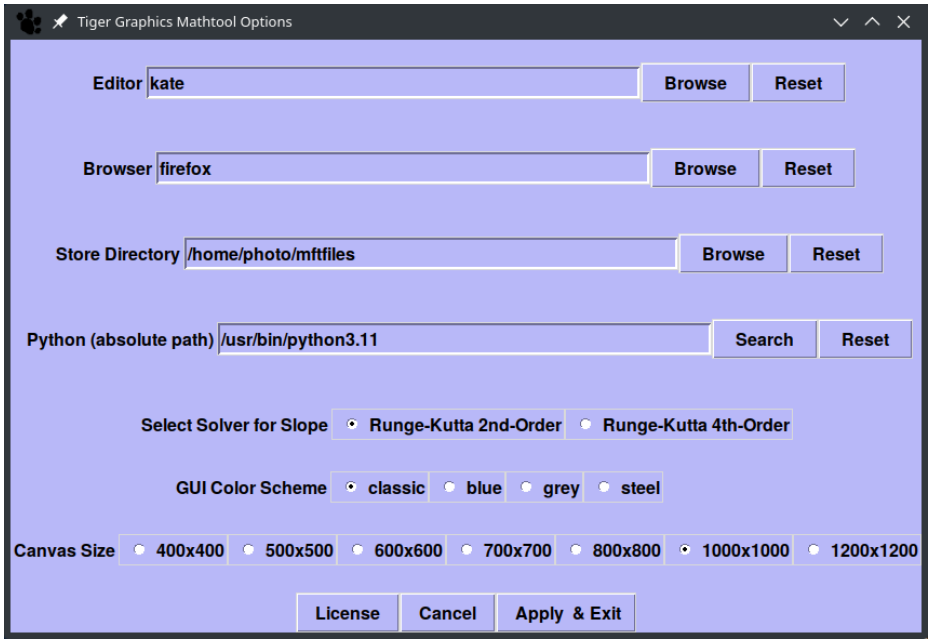


Figure 2.2: The custom widget

The default editor and the default browser for the **MathTools** Help can be set.
In the entry [Store Directory] a directory shall be selected according to your preferences where **MathTools** shall store plots, model files and from where the latter ones could be loaded.
Also the path to the **Python** exectuable must be set here. Please note that the absolute path to the **Python** executable is needed, because it may happend that different local version might be installed.
Please note that at least **Python** 3.10 is required for symbolic analyses in the Slope tool.

To calculate trajectories in the Slope tool one of two Runge-Kutta solvers can be seleted as the 'prime' solver.
Please consider the speed of your computer – Runge-Kutta 2nd-Order is faster.

Also select your preferred color scheme and the canvas size which defines the size of the graphics output.
Note: There are some defaults set.
Such it should not be necessary to customize the settings.
If an error message occurs while trying to edit, browse, store or load please select the executables and settings according to your system!

2.3 Help

Opens a new window with the selected browser (see Custom) and shows the *PDF* version of this manual.

3 Fractals

Several self similar fractals can be iterated:

- Highway dragon
- Koch Curve
- Cantor Set
- Weierstrass Monster
- Sierpinski triangle
- Pythagoras Tree
- Sunflower

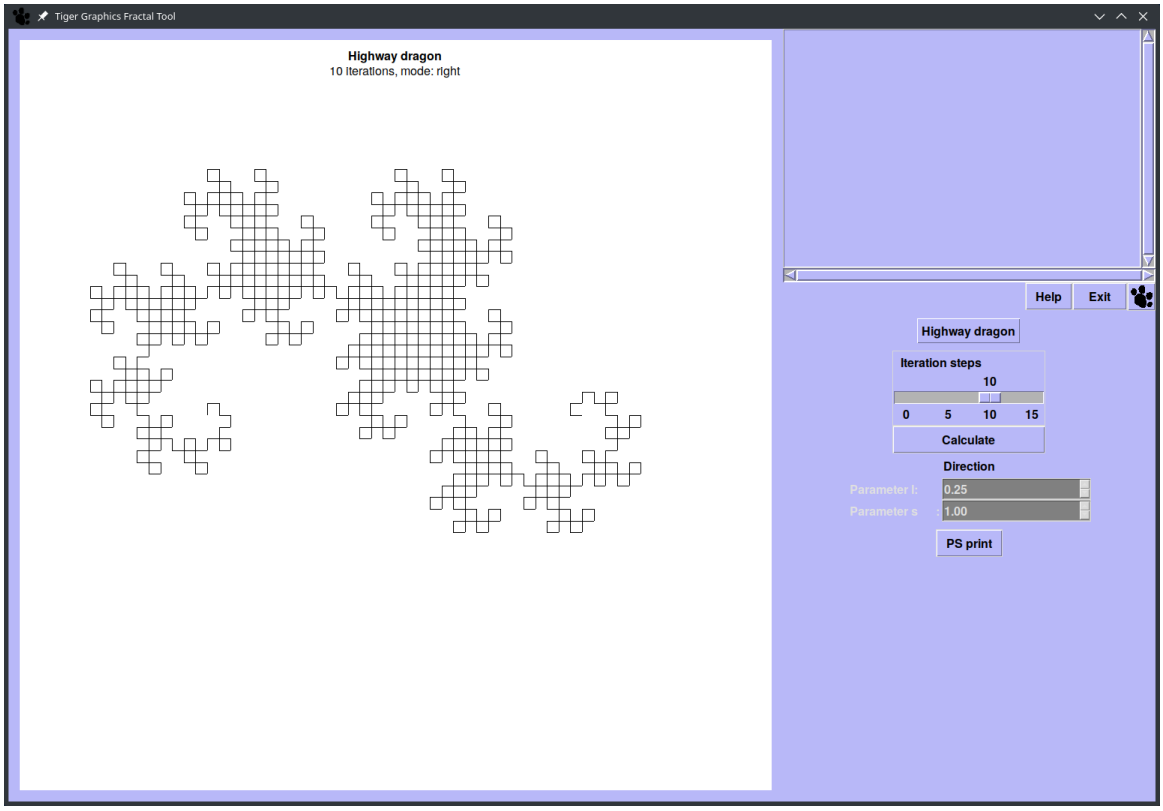


Figure 3.1: The Fractals Widget

The number of iteration steps can be set by the slider. The construction laws are self-explanatory if a low number of iteration steps is selected.

3.1 Calculate

The number of selected iterations is calculated and displayed.

3.2 Parameter settings for the different fractals

For the following fractals some parameters can be set to influence their generation.
If there are no parameter settings foreseen or necessary the buttons and input fields are deactivated.

3.2.1 Highway Dragon

The direction of the construction can be selected. The effect can be seen best if only a few iterations are selected.

3.2.2 Weierstrass Monster

The limit object of the Weierstrass Monster is a function which is continuous in $\mathbb{R}$, but not differentiable at any point in $\mathbb{R}$.
It is defined by

$$f(x) = \sum_{k=0}^{N} \lambda^{k(s-1)} \cdot sin(\lambda^{kt})$$

where N is the number of iterations. The values for λ and s can be selected.

Note: not every parameter combination leads to nice monsters.
The fractal dimension of the monster is estimated and displayed.
This is only a very rough estimation, because the calculation is done on the base of the Nth iteration by the curve length.

3.2.3 Sunflower

The configuration of seeds in a sunflower is simulated. Starting from a central point the radius of every step is increased by s while the angle is increased by φ given in degree.

3.2.4 PS print/Print

A Postscript output is generated and stored in *<Path to Store Directory>/plots*.
The PS-files are named automatically.

4 Mandelbrot

This tool collection includes three elements: the classical Mandelbrot set, the attractors of solving the complex equation $x^3 = -1$ by Newton's method and the so called Collatz fractal.

4.1 Mandelbrot's Set

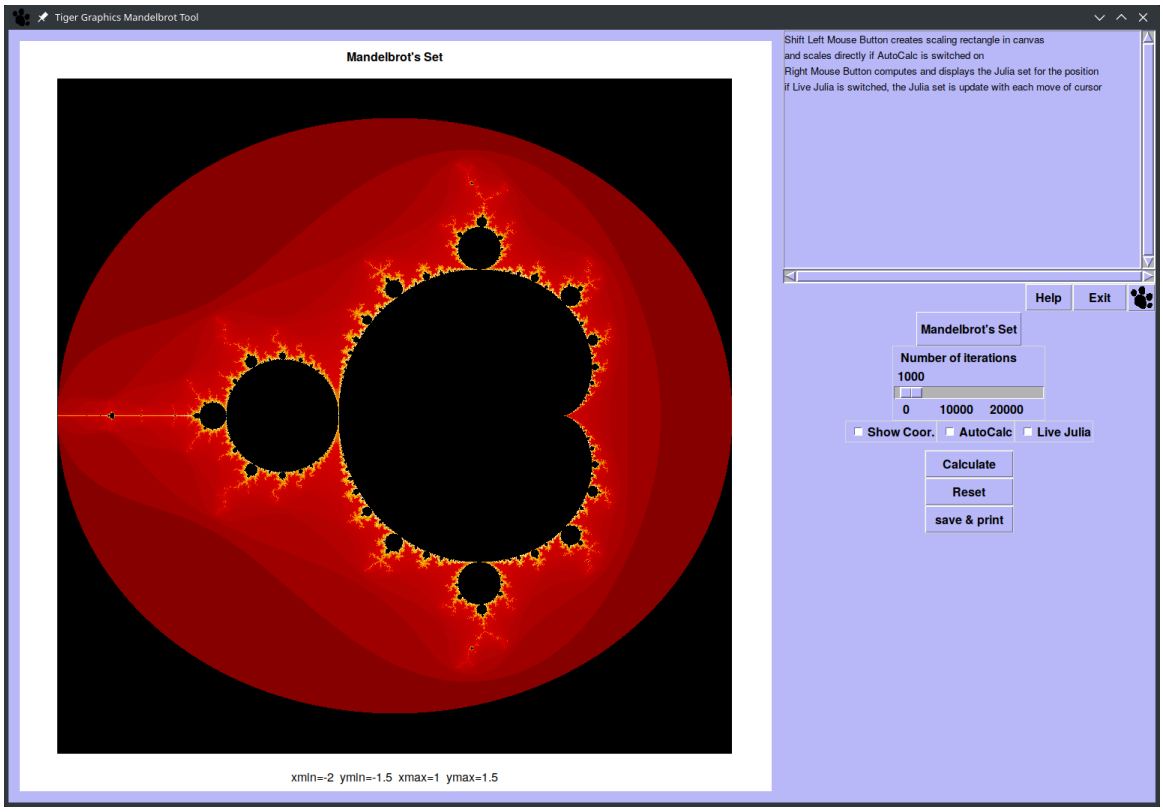


Figure 4.1: The Mandelbrot Widget

The Mandelbrot set is constructed and displayed. The Mandelbrot set is defined as the following subset of the complex plane:

$$M = \{c \in \mathbb{C} \mid (z_n)\ remains\ bounded,\ \ z_{n+1} = z_n^2 + c, z_0 = 0\}$$

Such, in the picture, the black object is the Mandelbrot set. The nice colors are determined by the speed of divergence.

The Julia set of a given point c is defined as the following set of points in the complex plane:

$$J_c = \{z \in \mathbb{C} \mid (z_n)\ remains\ bounded,\ \ z_{n+1} = z_n^2 + c, z_0 = z\}$$

The Julia set of a point is calculated if clicked with the right mouse button and additionally by selection the Live Julia checkbox (see below).
Clicking the right button at a point in the image initiates the computation of the Julia set with the selected numer of iterations and stops the Live Julia function.

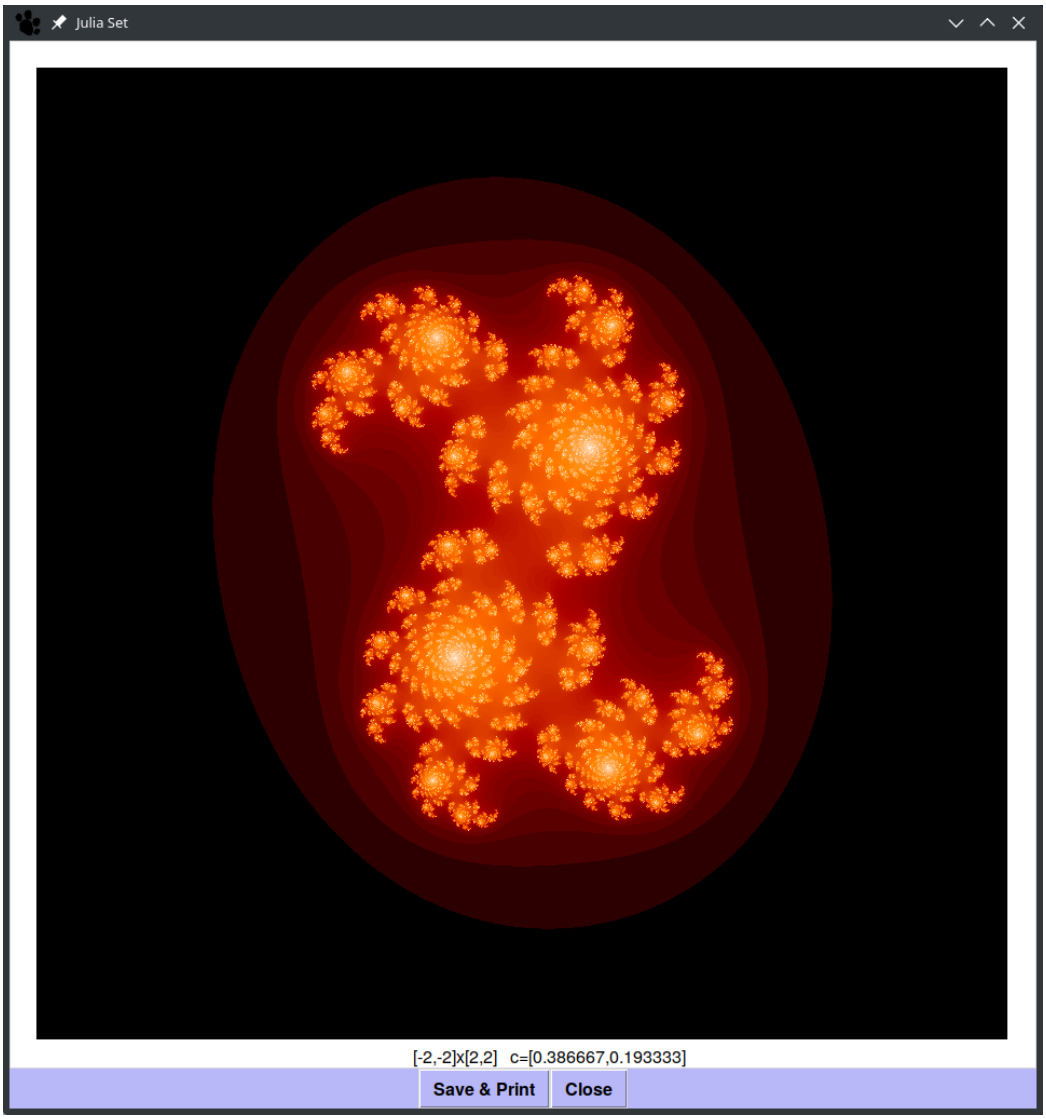


Figure 4.2: A Julia Set

Live Julia

Selecting this checkbox opens a new window. In this new window the Julia set of the Mandelbrot fractal is live updated upon cursor movements. To be fast enough, the Julia set is computed with a reduced number of iterations.

4.2 Newton roots

The tool allows also the visualization of the basins of the roots z_i of the complex equation $x^3 = -1$ calculated by Newton's method.

The three basins B_i are defined as the following sets of points:

$$B_i = \{x \in \mathbb{C} \mid x_n \to z_i,\ x_0 = x,\ x_{n+1} = x_n - \frac{f(x)}{f'x)},\ f(x) = x^3 + 1\}$$

4.3 Collatz' Fractal

The Collatz fractal in the complex plane is derived from the Collatz conjecture, a mathematical sequence defined by the function:

$$f(z) = \begin{cases} \frac{z}{2} & \text{if } z \text{ is even} \\ 3z + 1 & \text{if } z \text{ is odd} \end{cases}$$

or in complex formulation[1] :

$$f(z) = \frac{2.0 + 7.0 \cdot z - (2.0 + 5.0 \cdot z) \cdot \cos(\pi \cdot z)}{4.0}$$

Starting with a complex number z_0 , the Collatz fractal is generated by iteratively applying the function $f(z)$ to the result of the previous iteration.
The behavior of this iteration is observed in the complex plane.

4.4 Common widget functions

4.4.1 Zooming

Within the picture (the canvas area) a rectangle can be defined by pressing and holding the left mouse button and dragging the mouse while the button is pressed. Calculate starts the calculation of the new cut-out. The number of iterations must be set higher the more you go into detail.
Additionally, you can zoom in and out by using the mouse wheel.

[1]The derivation of this representation of the Collatz equation can be found at 3D-Meier

4.4.2 Reset

Pressing this button resets the canvas to the initial coordinates. The selected number if iterations is not effected.

4.4.3 Calculate

The part of the Mandelbrot Set is calculated depending on the cut-out. The number of iterations determines the accuracy.

4.4.4 Reset

Pressing this button resets the canvas to the initial coordinates. The selected number if iterations ist not effected.

4.4.5 Save & Print

A Postscript output and a *gif* output is generated and stored in *<Path to Store Directory>/plots*. The files are named automatically.

5 Slope

This tool allows the visualization of simple one or two dimensional differential equations.

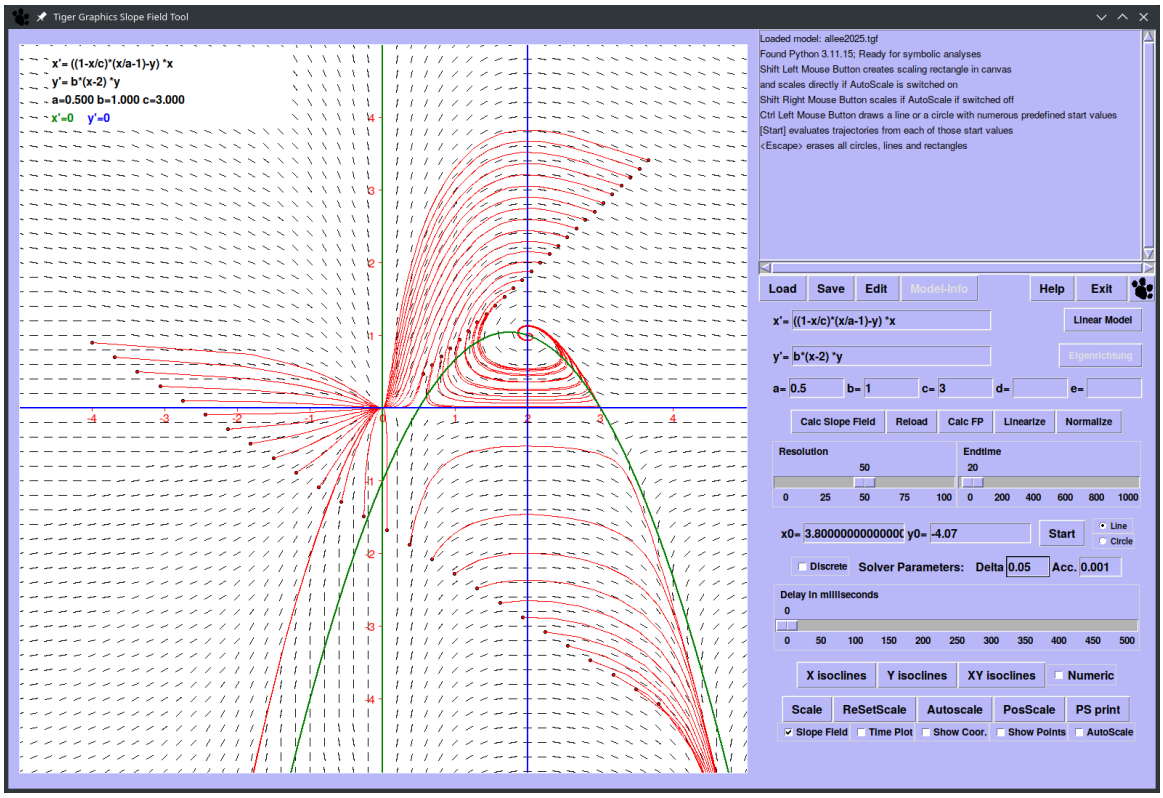


Figure 5.1: The Slope Widget

5.1 One Dimensional differential equation

Set $x' = 1$. This means that $x(t) = t(t_0 = 0)$. Set $y' =$ to the differential equation which should be considered.
[Calc Slope Field] computes and displays the slope field of the differential equation in the $t - y$ plane.

5.2 Two Dimensional differential equation

Set $x' =$ and $y' =$ to your differential equation system which should be considered.
[Calc Slope Field] shows the slope field of the system in the $x - y$ phase plane.

5.3 Parameters and Built in Math functions

Up to five parameters (a, b, c, d, e) can be used within the formulas.
The following math functions can be used within the formulas:

```
acos asin atan cos cosh exp log log10
pow sin sinh sqrt tan tanh abs
```

5.4 Linear Model

If the parameters a, b, c, d are set [linear] makes a linear system from these parameters:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

The Eigenvalues und Eigenvectors, the trace and determinant are displayed in the Info window.

5.5 Eigenrichtung

The directions of the Eigenvectors are plotted. This button is only active if we have non-complex Eigenvectors.

5.6 Reload

Pressing this button forces a reload of the original differential equation or differential equation system when needed, especially after the derived linear model has been computed and you want to return the non-linear.

Python dependent functions

The functions Calc FP, Make Linear and Normalize require **Python** for symbolic analyses.
If no suitable **Python** is found (see Custom (2.1)), those buttons will not appear.

5.7 Calc FP

The fixed points of the the defined system are computed and displayed in the Info window.

5.8 Make Linear

The linear model is derived and the fixed points are derived. If there are more than one fixed point, a selection window is opened to select one of the fixed points decribed by its value. After selection the according slope field is computed and displayed.
After the linear model has been derived, those input fields and function buttons which are meaningless for the derived linear model are disabled. They all are enabled again after reloading the original equations by Reload or loading a new model by [Load].

5.9 Normalize

Normalizing serves to compare models of different types by unifying the scales and shifing the fixed points to [1,1] in the [x,y]-plane. The process is as follows for generic model: Let the model be described by:

$$\begin{aligned} \dot{x} &= (g(x) - y \cdot h(x)) \cdot x \\ \dot{y} &= a \cdot (x \cdot h(x) - m) \cdot y \end{aligned}$$

For a fixed point (x^*, y^*) with $x^* > 0$ and $y^* > 0$
the normalized model is derived by the substitution $X = x/x^*$ and $Y = y/y^*$

$$\begin{aligned} \dot{X} &= \frac{dX}{dt} = \frac{1}{x^*} \cdot \frac{dx}{dt} = (g(X \cdot x^*) - Y \cdot y^* \cdot h(X \cdot x^*)) \cdot X \\ \dot{Y} &= \frac{dY}{dt} = \frac{1}{y^*} \cdot \frac{dy}{dt} = a \cdot (X \cdot x^* \cdot h(X \cdot x^*) - m) \cdot Y \end{aligned}$$

The formulas in the function entries are replaced by those for the normalized model where **X** stands for the term $(X \cdot x^*)$ and **Y** stands for the term $(Y \cdot y^*)$.
The fixed point coordinates are denoted by **Xf** and **Yf**.
If none or only one considerable fixed point is found, a message is given in the info window; in case of more than one considerable fixed points are found a selection window is opened to select one of the fixed points identified by its coordinates.
After selection of a fixed point the according slope field is computed and displayed.

If at least one considerable fixed point is found, the slope field of the normalized model is computed and displayed.
After the linear or a normalized model has been derived, those input fields and function buttons which are meaningless for the derived model are disabled.

They all are enabled again after reloading the original equations by Reload or loading a new model by [Load].

5.10 Resolution

The slider Resolution sets the resolution of the slope field. After setting a new value the slope field is refreshed.

5.11 Endtime

The slider Endtime sets the endtime for the evaluation of the trajectories.

5.12 Delay

The slider [Delay in milliseconds] set a delay for the solver when clicking the left or the middle mouse button. It slows down the evaluation of the trajectory for a better impression how it develops (see Section 5.19). This is very useful in combination with the selection of Show Points.

5.13 Solver Parameters

Here the parameters for the selected Runge-Kutta solver (see {-sec-custom}) are set **Delta** defines the time step and **Acc.** defines the accuracy (the local error) to reduce the internal time step if exceeded.

5.13.1 Discrete

If this checkbutton is selected Δt (the parameter **Delta**) will be set to **1** and the Euler method will always be used.

5.14 Isoclines

The isoclines of the system $(x_0 = 0, y_0 = 0)$ can be plotted. Either the single isoclines [X isoclines] or [Y isoclines] or both with [XY Isoclines]. If **Python** is avaiable the isoclines will be derived by symbolic analysis otherwise they will be determined numerically, which is less accurate.

5.15 Scale

The x and y ranges of the plotting area for the slope field can be set.
Applying the input values by [Apply] refreshes the slope field;
[Cancel] just closes the window and rejects the changes.

5.16 ReSetScale

This button resets the the range of the plotting area to the initial values.

5.17 Autoscale

The maximum values of x and y are evaluated and the plotting area is adapted to those values accordingly.

5.18 PosScale

The x and y range is shifted to the positive quadrant.

5.19 Trajectories

Trajectories to given initial values can be plotted in two ways:

1. Setting initial values for x and y in $x0 =$ and $y0 =$ and pressing [Start]
   or
2. Setting an initial value by a left mouse click within the slope field. The trajectory will be computed by the selected solver (Runge-Kutta as selected (see Section 2.2)) or Euler if Discrete is on.

3. Setting an initial value by a left mouse click within the slope field. The trajectory will be computed with the Euler method with a fixed time step of **1**.

It might be helpful to switch on Show coor. to see the coordinates of the mouse cursor.

The calculation is done by evaluating the differential equation with a second order method. By middle mouse click the evaluation is done with Euler's method.
The time step Δt for the evaluation is 0.01.
The simulation stops either at Endtime or can be stopped by a right mouse click in the slope field.

5.20 Multi-Trajectories

Although trajectories can be startet from any point in the slope field, it might be uncomfortable to inspect areas of interest by a lot of single clicks.
On the right hand side of the [Start] button you can select whether to draw a line or a circle for defining a set of inital values for trajectories.
By holding the $<CTRL>$ - key and pressing the left mouse button a line or circle can be drawn from the point of origin by dragging the mouse pointer while keeping $<CTRL>$ pressed. After releasing the mouse button the line resp. the circle will be transformed into a set of initial values displayed as a set of blue dots.
The number of dots is the half values of the Resolution parameter.
After pressing [Start] the evaluation of the trajectories for each of those initial points begins.
If a set of initial values is drawn but should not apply, it can be erased by pressing the $<Escape>$ - key on the keyboard.

5.21 Check Buttons

Time Plot

In the case of a two dimensional system the time information cannot be seen in the phase diagram. If Time Plot is switched on a second plot window is opened showing the temporal evolution of the trajectory.

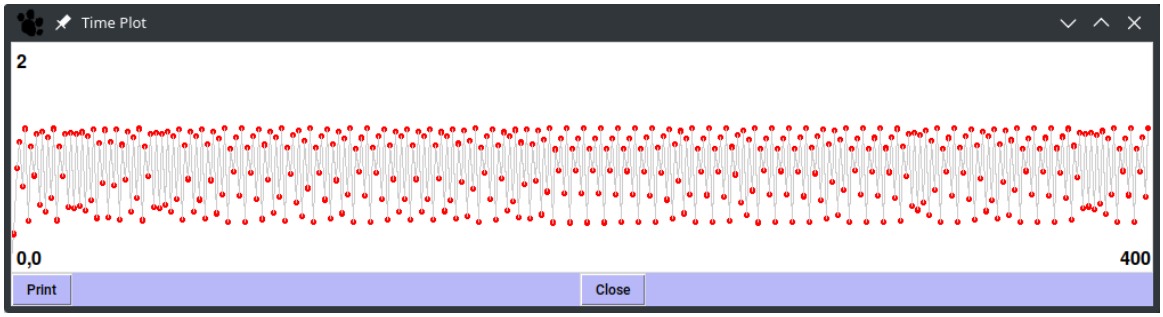


Figure 5.2: The TimePlot Window

Show Coor.

If Show Coor. is switched on the coordinates in the slope field are shown at the mouse pointer.

Show Points

If Show Points is switched on a dot is displayed in the slope field after each integration step of the trajectory. The distance between those dots depend on the value of Δt.

AutoScale

If AutoScale is switched on the definition of a scaling/zooming area by <*Shift*>-Left Mouse Button is directly applied.
Otherwise Shift-Right Mouse Button will initiate the scaling.
If a scaling rectangle is drawn but should not apply, it can be erased by pressing the <*Escape*> -key on the keyboard.

5.22 Load, Save & Edit

Equations, parameter values, coordinates, resolution and endtime can be saved in a file. These files get automatically the extension *.tgf* and are stored normally in <*Path to Mathtools*>/*mftfiles*.
After loading a file the equations, parameter values and the stored coordinate values, resolution and endtime are set.
The files can also be edited to set f.e. coordinate values not provided by default.
This editing functionality does not work under windows at the moment.
Please use your usual editor to edit the files manually.

5.23 PS print/Print

A Postscript output is generated and stored in <*Path to Store Directory*>/*plots*.
The PS-files are named automatically.

6 Celluar Automata

Cellular automata are discrete dynamical systems.
Their space is represented by a uniform grid, with each cell containing a value;
time advances in discrete steps and the rules of change of each cell is determined by the states of its closest neighbour cells.

Different celluar automata are implemented an can be simulated

- Conway's Life
- Wofram Universe
- Disease Spread
- Per Bak's sand pile
- Circular room
- Spreading of Bugs
- Diffusion
- two advections automats
- a diffusion based automat to generate leopard skins

The automat can be selected by the pull down menu. The active one is diplayed.

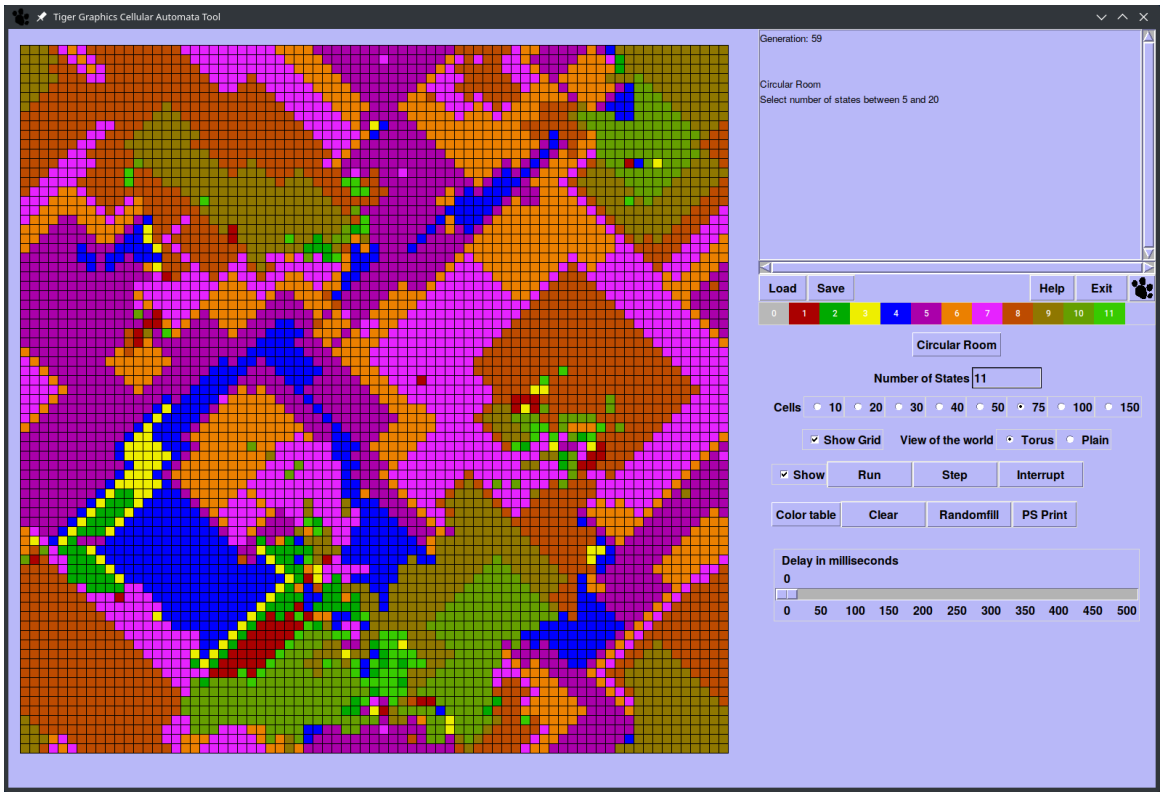


Figure 6.1: The Cellular Automata Widget

6.1 Life

Life was devised in 1970 by John Horton Conway.
The automat knows two states: dead (0, off) or alive (1, on).
The neighbourhood of a cell is defined by the eight surrounding cells.

The rules

- if, for a given cell, the number of neighbours is exactly two, the cell maintains its status quo into the next generation. If the cell is on, it stays on, if it is off, it stays off.

- if the number of on neighbours is exactly three, the cell will be on in the next generation. This is regardless of the cell's current state.

- if the number of on neighbours is smaller than or equal to 1 or greater than or equal to 4 the cell will be off in the next generation.

6.2 Universe according to Stephan Wolfram

To understand the principle, we will first look at Stephen Wolfram's one-dimensional model.
Stephen Wolfram (*1959) is a British physicist. He is the developer of Mathematica and the search engine Wolfram Alpha.
His book "A New Kind of Science (2002)" describes many applications of cellular automata.
A simple example is the following automaton: We divide the space into a number of cells (e.g. 150).
Each cell can assume the state dead or alive (0 or 1). Now you can start with a live cell in the centre.
The constellation in the next step results from the following **rules**

- A cell receives the new state 1 if there were one or two living cells in the block of three consisting of itself and its two neighbours.
- A cell receives the new state 0 if there were zero or 3 living cells in the block of three consisting of itself and its two neighbours.

If the last cell is reattached to the first (ring), the border cells also have 2 neighbours. You can then calculate more generations without getting edge effects.

The meaning of the headline is just the situation of the cell under regard (in the center of the triplet).
So **1 1 0** means: the left neighbor is alive (1), the cell itself is alive (1) and the right neigbor dead (0).

The rules are defined through the transistion of all eight triplets to their results.
In this case we see the rule **0 1 1 1 1 1 1 0**, meaning that **1 1 1** results in **0** and **1 0 1** results in **1**.
The complete transitions are:

1 1 1	1 1 0	1 0 1	1 0 0	0 1 1	0 1 0	0 0 1	0 0 0
↓	↓	↓	↓	↓	↓	↓	↓
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**

Finally the number of rules is the span that can be addressed by eight digits or bits which results in a binary number (represented by the bottom line of the above table). There are therefore 256 rule sets in total.
Wolfram used that principle and interpreted the eight-digit binary number as a decimal number. This allowed him to simply "name" all possible rules from 0 to 255.
The above rule is then number 126 (binary **0 1 1 1 1 1 1 0**).

6.3 Disease

This automat simulates the spread of a disease.
A cell can either be susceptiple (0), infected (1), ill (2) or immune (3).
The automat is not fully deterministic !!

The rules

- if a cell is susceptible it will be infected with a certain probability depending on the number of infected neighbours.
  The cell will be infected if the number of neighbours multiplied with a random number between 0 and 1 exceeds 0.9.

- infected cells become ill with a probability of 60%

- ill cells stay ill (50% probability), become immune (25% probability) or susceptible (25% probability)

- immune cell become susceptible with a probability of 5% to represent deaths and births

6.4 Per Bak's sandpile model

Per Bak's sandpile model is an example of self-organized criticality.
It is a cellular automat whose configuration is determined by the number of sandgrains on a cell.

The rules

- a grain of sand is added at a selected cell

- a cell with 4 or more sand grains becomes unstable and topples by distributing one grain of sand to each of it's four neighbours.
  This may cause unstable cells in the neighbourhood. An avalanche is born.

- if any avalanche dies out, another grain of sand is added to a selected cell.

There are two models implemented by selecting the number of neighbours in the input field and pressing return:

- 4 neighbours. A cell with 4 or more sand grains becomes unstable and topples by distributing one grain of sand to each of it's four (left, right, up and down) neighbours.

- the same model but with 8 neighbours. A cell with eight and more sand grains becomes unstable and topples by distributing one grain of sand to each of it's eight neighbours.

If you press Run a grain is added to a randomly selected cell.
If you select -4 or -8 in the parameter input field and then press Run the new grain will alway be put in the center of the field.

If you click with the left mouse button on a cell one grain is added to this cell.

6.5 Circular Room

The circular room is a nice automat to demonstrate self organization.
It is most impressive to start from a randomized initial state.

The rules

- The N states are circulary arranged: The state N is identified with state 0. Each cell has 4 neighbours.
- The state of a cell is increased by one if at least one neighbour has the state of the cell plus 1.

The number of states (N) can be input in the paramenter input field.

6.6 Bug Spread

This is an example of an automat combined with a differential equation model. The are two state variables, the number of bugs B within the part of the forest (cell) and the mean age A of the trees in this part of the forest. The differential equations for the bug growth, the aging of the forest and the bug spread are given by

$$\begin{aligned}\frac{\partial B}{\partial t} &= r \cdot A \cdot \left(1 - \frac{B}{K}\right) \cdot B - \mu \cdot \frac{B^2}{B^2 + M^2} + D_a \frac{\partial^2 B}{\partial x^2} \\ \frac{\partial A}{\partial t} &= \alpha \cdot (1 - A) - \beta \cdot \left(\frac{B}{K}\right)^3 \cdot A\end{aligned}$$

The diffusive spread is realized by an automat scheme with an additional wind direction from west to east.

$$\begin{aligned}\Delta B(i,j) &= -4\,\varepsilon_0 \cdot B(i,j) \cdot \Delta t \\ \Delta B(i-1,j) &= \varepsilon_0 \cdot (1 - \varepsilon_r) \cdot B(i,j) \cdot \Delta t \\ \Delta B(i+1,j) &= \varepsilon_0 \cdot (1 + \varepsilon_r) \cdot B(i,j) \cdot \Delta t \\ \Delta B(i,j-1) &= \varepsilon_0 \cdot B(i,j) \cdot \Delta t \\ \Delta B(i,j+1) &= \varepsilon_0 \cdot B(i,j) \cdot \Delta t\end{aligned}$$

(i, j) denotes the cell in the i-th row and j-th column. The differential equations are solved with Euler's scheme. The time step Δt for the calculations is 0.01.

Parameters

Parameter	Value	Unit	Meaning
r	54.75	1/y	growth in old forest
K	5000	bugs/cell	capacity of bugs
μ	36500	bugs/cell/y	predation by birds
M	500	bugs/cell	bug density controlling predation
α	0.15	1/y	aging factor of forest
β	1.7	1/y	damage of forest
ε_0	1.0	1/y	maximum part of bugs to diffuse
ε_r	0.9	-	parameter for direction (wind force)

The simulation shows the number of bugs per cell.
It takes several hundred generations until the dynamical state of the system appears.
Unfortunately this example is very slow due to the numerical solver of the differential equations.

6.7 Diffusion

Demonstration of diffusion.
The simulation should be started with a random filling or a small filled area (circle or square).
The color table is fixed to white&black. Filling can be done by a left mouse click.

6.8 Advection

Demonstration of advective transport.
The simulation should be started with a small filled area (circle or square).
The color table is fixed to white&black. Filling can be done by a left mouse click.

Two scenarios are implemented:

1. 1D advection (1 cell per time step): simulation of advection with numerical diffusion.
   The advection velocity is set by the parameter input in the input field in cells per time step in x direction.
   If **1** is input there will be numerical diffusion.

2. 2D advection (0.1 cell per time step): simulation of advection demonstrating the effect of numerical diffusion.
   The advection velocity is set to 0.1 cell per time step in north-east direction.

6.9 Leopard

The model describes a mechanism that can describe how the fur pattern of a leopard can develop through diffusion processes. The model considers two substances with different properties.

The **Activator**: Substance that produces the expression of a characteristic (pigment production) with the following characteristics:

- autocatalytic (self-reinforcing)
- generates inhibitor
- diffuses slowly (localized)

The **Inhibitor**: Substance that suppresses expression of the characteristic

- suppresses activator production
- diffuses quickly (long-range)

Equations

$$\frac{\partial a}{\partial t} = \frac{a^2}{b} - \mu_a \cdot a + \varepsilon_a + D_a \frac{\partial^2 a}{\partial x^2} \qquad Activator$$

$$\frac{\partial b}{\partial t} = a^2 - \mu_b \cdot b + \varepsilon_b + D_b \frac{\partial^2 b}{\partial x^2} \qquad Inhibitor$$

Parameters

Parameter	Meaning
$\mu_a = 1.0$	Decay rate of the Activator
$\mu_b = 1.0$	Decay rate of the Inhibitor
$\varepsilon_a = 0.01$	Basic production rate of the Activator
$\varepsilon_b = 0.01$	Basic production rate of the Inhibitor
$D_a = D_b/8.0$	Diffusion constant of the Activator (fixed ratio)
$D_b = selectable$	Diffusion constant of the Inhibitor

The parameter D_b can be input in the parameter input field.

6.10 Common Settings

6.10.1 Parameter Input

Directly below the selector for the automat a Parameter input field becomes available when parameters can be input for the active automat.
The parameter must be confirmed by pressing the return button which leads to a clearance of the graphics area and an update of the color table and color bar, if required.
The description follows the selected automat.

6.10.2 Color Table

The different automata look nicest with their default color table, but the colortable can be changed for some automata and is displayed in the color bar. The automata with discrete states are best visualized by the discrete color table. The number of colors displayed in the color bar correspond to the number of states of the active automat.
Alternative color tables can be seletcted for Circular Room, the Advection and the Diffusion automats.

6.10.3 Cells per row

The number of cells per row can be selected. The automat field is then built quadratic. The higher the number the slower the simulation runs.

6.10.4 View of the world

The world can either be assumed as closed (Torus) in this case the first row is neighbour of the last one and the first column is neighbour of the last column. If the case of a plain world boundary cells have less neighbours.

6.11 Common Functions

6.11.1 Show

If Show is switched on, every generation is shown, if switched off the picture is not actualized until Show is switched on again.
This is useful if only higher generation numbers are of interest because the simulation becomes quite faster without displaying.

6.11.2 Run

Run starts the simulation. The generation is displayed in the info window.
The simulation can be stopped by Interrupt .

6.11.3 Step

Step calculates the next generation

6.11.4 Interrupt

Interrupt stops the simulation.

6.11.5 Clear

Clear sets all cells to zero.

6.11.6 Delay

The slider [Delay in milliseconds] set a delay for the solver when clicking the left or the middle mouse button. It slows down the evaluation of the automat for a better impression how it develops.

6.12 Setting the initial states

6.12.1 Single Cells

The initial state of the automat can either be set by mouse within the automat field.
Depending on the number of states the following settings are possible

- left mouse click sets cell to 1
- <*Shift*>+ left mouse click sets the cell to 2
- <*ctrl*>+ left mouse click sets the cell to 3
- middle or right mouse click resets the cell to 0

6.12.2 Randomfill

Randomfill fills the field with random values depending on the number of possible states.

6.13 Load & Save

It is possible to save a configuration by [Save] in a file.
These files get automatically the extension *.lif* and are stored normally in *<Path to Mathtools>/mftfiles*.
The configuration can be loaded by [Load]

6.14 PS print/Print

A Postscript output is generated and stored in *<Path to Store Directory>*/plots.
The PS-files are named automatically.

7 IFS - Iterated Function Systems

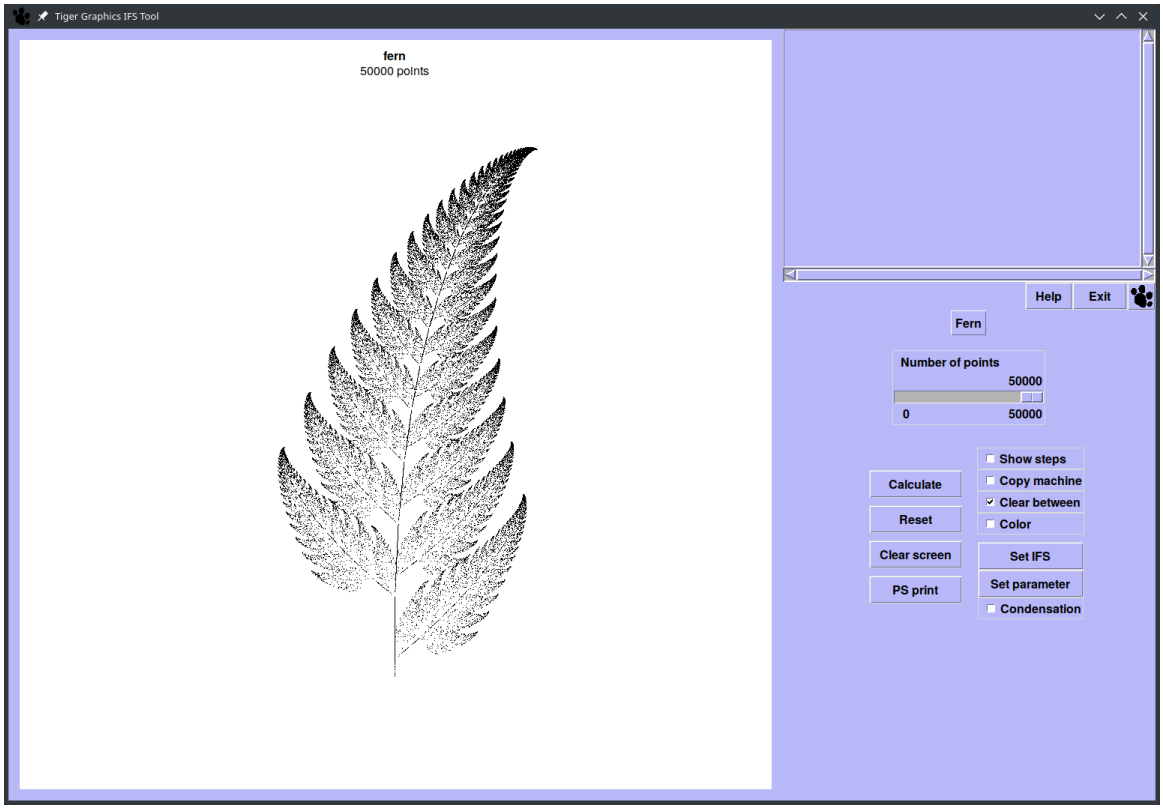


Figure 7.1: The IFS Widget

Iterative function systems are generators for self similar images.
Further information on IFS can be found in Barnsley. This tool allows to define up to 4 affine maps per image.
Predefined sets of maps are available for

- Fern leaf
- Highway dragon
- Maple leaf
- Simple tree
- Phytagoras tree
- Sierpinski's triangle

- Chess board[1]

7.1 Calculate

Here are two modes available to compute the iterated system:

[Copy machine] switched off

Starting from an initial point the chaos game is started.
The number of points to be drawn can be selected.
Due to the accuracy in the random function the algorithm runs into a cycle.
Such only a few number of points can be distinguished.

[Copy machine] switched on

A quadrangle is mapped by the given affine maps.
Pressing calculate gain maps the newly created quadrangles again and so on.
Due to the increasing number of quadrangles this process becomes slower and slower.
Such only 8 iterations are possible by the machine.

Reset resets this process.

If [Color] is selected each step is plotted in a different color.

If [Clear between] is selected only the quadrangles of the last step are printed.

Reset

Resets the iteration of the copy machine.

Clear screen

All printed stuff is deleted. The maps still exist.

7.2 PS print/Print

A Postscript output is generated and stored in *<Path to Store Directory>/plots*.
The PS-files are named automatically.

[1]The Chessboard is defined by 5 affine maps. The last one is treated as condensation!

7.3 Set IFS

A reference rectangle occurs in the picture window.
By left mouse click up to four affine maps can be defined.
The first click defines the image point of the upper left corner, the second of the lower left corner and the third the image point of the lower right corner.
The resulting quadrangle is plotted.
This process can be repeated three times.
The last map is additionally stored as fourth map. It is only evaluated if

- four different maps are defined
  or
- [Condensation] is switched on

7.4 Set Parameters

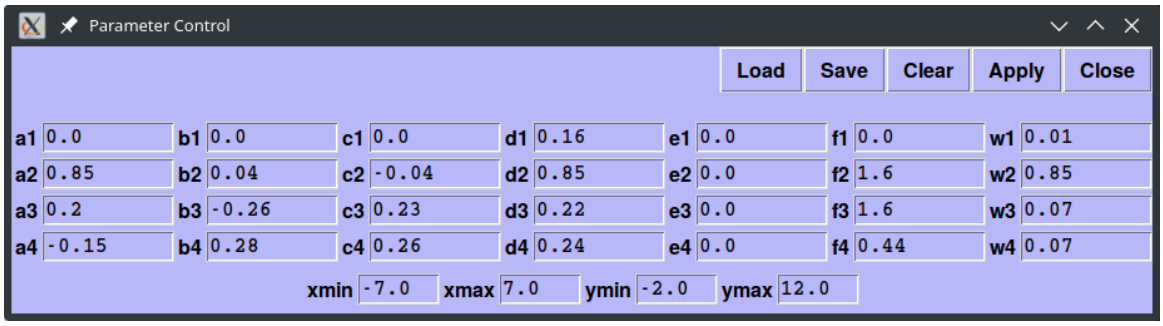


Figure 7.2: The IFS Widget

The parameters of the actual IFS are shown and can be modified by

- modifying them and then pressing [Apply]
  or
- selecting another predefined IFS
  or
- loading an IFS which has been saved earlier
  or
- defining a new IFS by Set IFS

In the parameter window each row represents one map.
The parameters are defined as follows:

$$f_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}, \qquad i = 1, .., 4$$

The values w_i in the last column give the probabilities for selecting this map by the random game.
If the sum of the first probabilities exceeds 1, the following maps are not evaluated.
If [Condensation] is active, the last map is evaluated as condensation independent of the probabilities.
The probability of the predefined maps are defined by the determinant of the matrix.
If an IFS is defined by Set IFS the probabilities are equally distributed.

7.4.1 Load & Save

The values can be saved in a file. The files holding IFS-data get the extension *.ifs* and are stored in *<Path to Mathtools>/mftfiles* by default.

References

BARNSLEY

BARNSLEY, Michael F.: Fractals everywhere. Morgan Kaufmann, 2000. – 534 S

8 Parabola

The iteration of parabola equation
$x_{n+1} = r \cdot x_n \cdot (1 - x_n)$
converges, shows cycles or chaos depending on the parameter r.

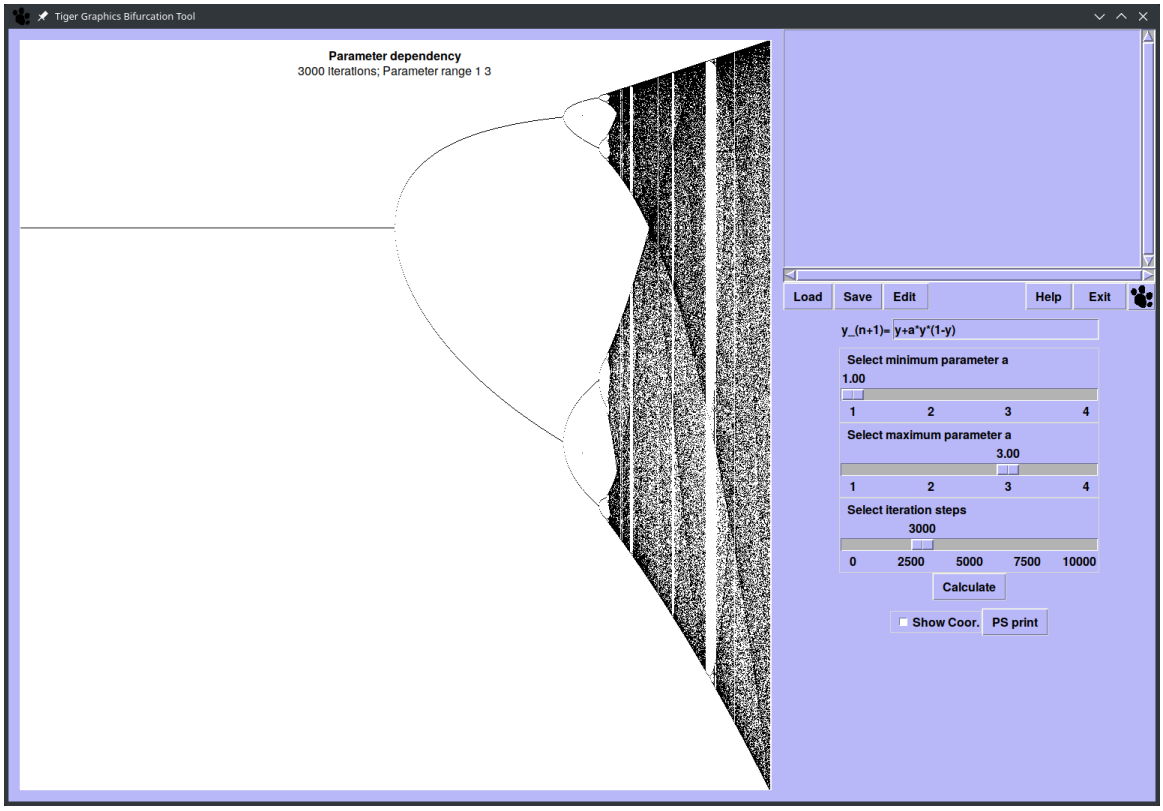


Figure 8.1: The Parabola Widget

8.1 Calculate

Shows the location of the limit points depending on the parameter r.
Clicking with the left mouse button into the picture shows the parabola of the selected value of r:
$f(x) = r \cdot x \cdot (1–x)$

8.2 Parabola of a single parameter r

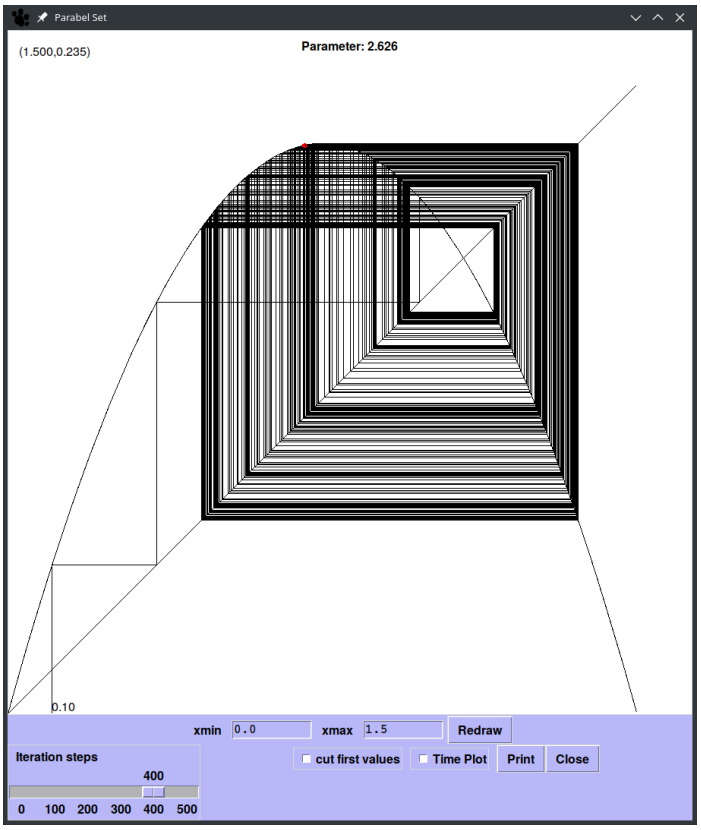
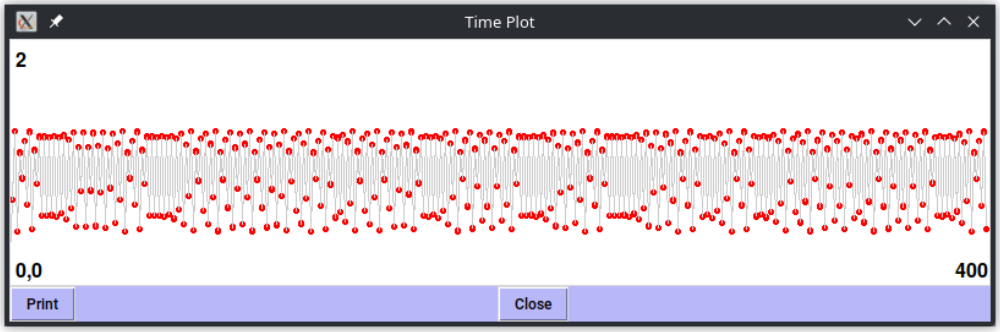


Figure 8.2: A single parabola with its iterations

A left mouse click starts the iteration within the parabola (initial value).
With middle mouse clicks the iteration can be continued step by step until the selected number of iterations is reached.
A right mouse click performs all iteration steps at once depending on the selected number of iterations.
If [cut first values] is switched on the initialization is cut off: only the limit point resp. cycle resp. chaos is shown.
If time plot is switched on the development of the iteration is shown in a separate window.



8.3 PS print/Print

A Postscript output is generated and stored in *<Path to Store Directory>/plots*.
The PS-files are named automatically.

8.4 Miscellaneous

The tool allows also the visualization of
$x_{n+1} = x_n + r \cdot x_n \cdot (1 - x_n)$
which is the discretization of the logistic differential equation
$\dot{x} = x \cdot (1 - x)$

9 Plotter

This tool allows the visualization of functions. In the **[1D]** mode up to three one dimensional functions of the form $f(x) = ...$ can be plotted.
The function is only plotted if the checkbutton on the left of the defintion is switched on.

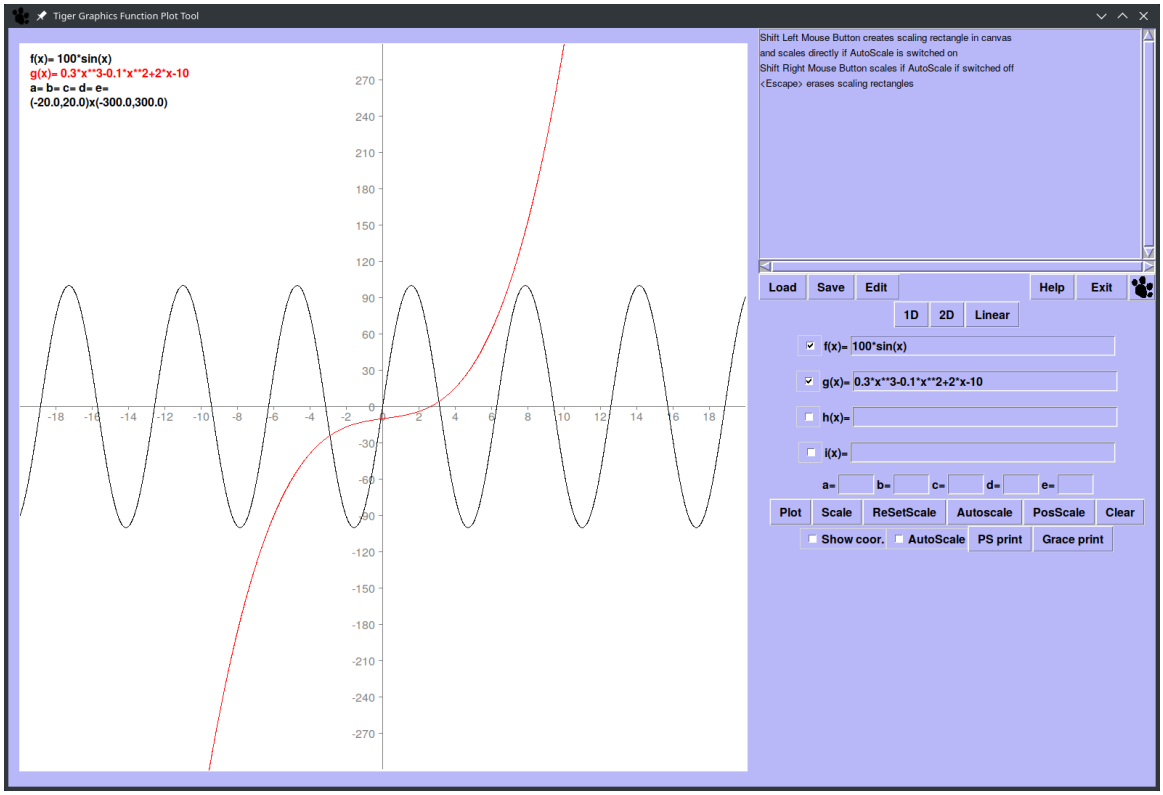


Figure 9.1: The Plotter Widget

In the **[2D]** mode a parametrized two dimensional function of the form
$x(t) = ...$
$y(t) = ...$
can pe plotted.

The following math functions can be used within the formulas:

```
acos asin atan cos cosh exp log log10
pow sin sinh sqrt tan tanh abs
```

The formulas can be defined with up to 5 parameters (a, b, c, d, e).

9.1 Plot

The defined function is plotted according to the settings of the independent variable x in [1D] mode or t in [2D] mode.
The axes can either be set by the sliders or set by Scale or AutoScale

9.2 Scale

The x and $y$ranges of the plotting area for the slope field can be set.
Applying the input values by [Apply] refreshes the slope field;
[Cancel] just closes the window and rejects the changes.

9.3 ReSetScale

This button resets the the range of the plotting area to the initial values.

9.4 Autoscale

The maximum values of x and y are evaluated and the plotting area is adapted to those values accordingly.

9.5 PosScale

The x and y range is shifted to the positive quadrant.

9.6 Clear

Clear clears the output window.

9.7 Coordinates

A left mouse click in the output windows shows a horizontal line within the output window at the actual y-coordinate and its value, a right mouse click shows a vertical line within the output window at the actual x-coordinate and its value.

9.8 Check Buttons

Show Coor.

If Show Coor. is switched on the coordinates in the slope field are shown at the mouse pointer.

AutoScale

If AutoScale is switched on the definition of a scaling/zooming area by *<Shift>*-Left Mouse Button is directly applied.
Otherwise Shift-Right Mouse Button will initiate the scaling.
If a scaling rectangle is drawn but should not apply, it can be erased by pressing the *<Escape>* -key on the keyboard.

9.9 Load, Save & Edit

Equations, parameter values and coordinates can be saved in a file.
These files get automatically the extension *.tgp* in [1D] mode resp. *.tgt* in [2D] mode and are stored normally in *<Path to Mathtools>/fracfiles*.
Before loading a file the mode ([1D] or [2D]) must be selected.
After loading the file the equations, parameter values and the stored coordinate values are set.
The files can also be edited to set f.e. coordinate values which not provided by default.
The editor must be set in Custom.

9.10 PS print/Print

A Postscript output is generated and stored in *<Path to Store Directory>/plots*.
The PS-files are named automatically.

9.11 Grace print

An **xmGrace** output is generated and stored in *<Path to Store Directory>/plots*.
The *xmgr*-files are named automatically.