Documentation for Basins of Attraction

Jasper Kosonen

May 4, 2021

Contents

1 Introduction

This document contains the documentation for the MATLAB App ”BasinsOfAttraction.mlapp”. It aims to further describe the functionalities of the app and how it is implemented. Therefore, this document alongside with documentation found in the MATLAB code provides all the necessary details for using the app correctly.

In Section 2, all necessary background for the program is presented briefly. The user’s guide can be found in Section 3. Implementation and restrictions of the app can be found in Sections 4 and 5, respectively.

2 Basins of Attraction

Newton’s method can be used to find the complex roots of a polynomial $p(z)$. The method iterates an initial point $z_0 \in \mathbb{C}$ under the expression

$$z_{n+1} = z_n - \frac{p(z_n)}{p'(z_n)}, \quad n \geq 0. \tag{1}$$

Some initial values do not converge to any roots of the polynomial whereas others do. The set of initial points $A \subset \mathbb{C}$ that converge to the root z^* while iterated under the expression (1) is called the *basin of attraction* of z^*.

The basins can be quite intricate and they are often fractals. The basins of attraction for the polynomial $z^5 - 1$ are shown in Figure 1.

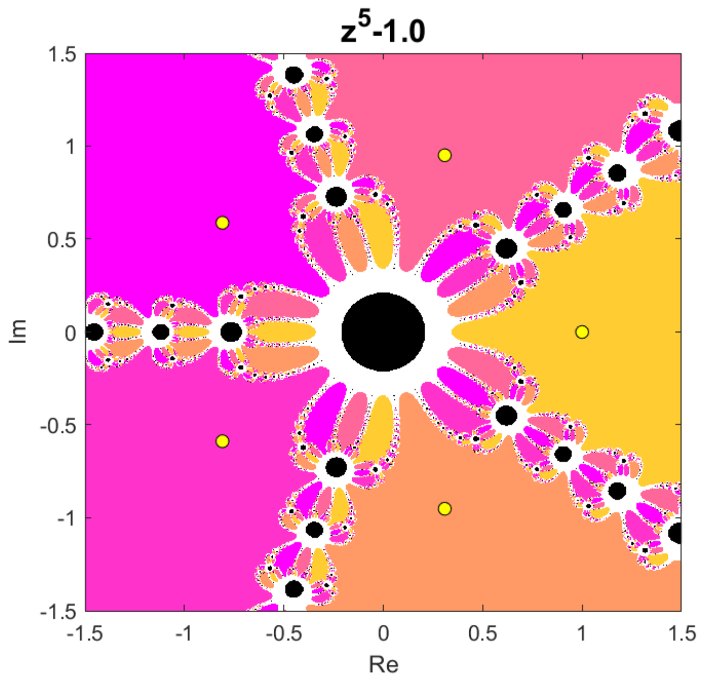


Figure 1: Basins of attraction for the polynomial $z^5 - 1$

In Figure 1 different colors indicate the convergence and divergence of different initial values. Initial values are colored black if they diverge when iterated under the expression (1). Although diverging, by definition, is simply the opposite of converging, ”to diverge” in this context implicitly means that the iterated values grow indefinitely. If an initial value does not converge nor diverge, it is labeled as indeterminate and colored white. This leaves only the converging initial values to be colored with any other colors.

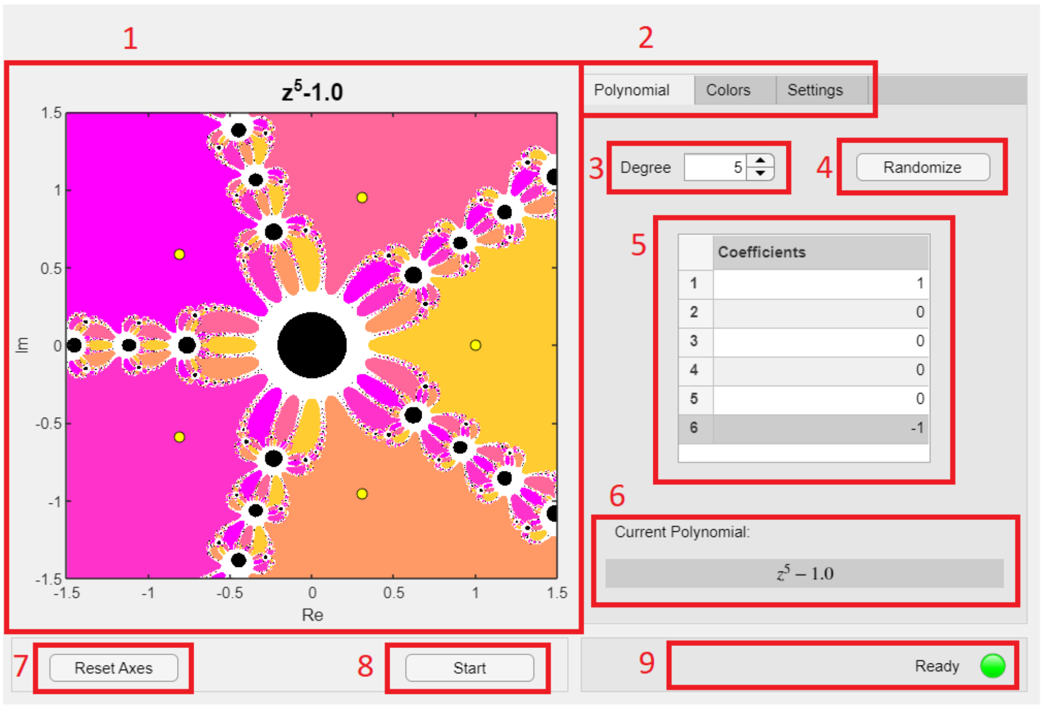
The reason this distinction between diverging and indeterminate initial values exists is due to the limitations of computing the iterations with real computers. There must be an upper limit for the number of iterations to be made. In this case, some values might take a longer time to converge or diverge. Therefore, they must be labeled as indeterminate if they have not converged nor diverged during the iteration process. This also implies that a tolerance for convergence is necessary. If the distance between an iterated value and a root of the polynomial is less than a fixed tolerance, the corresponding initial value is thought to converge to the root.

3 User's Guide

The user's guide provided in this Section helps to identify the different GUI components and their purpose. It also demonstrates how to calculate the basins of attraction for a given polynomial, how to change the colors and how the settings affect the iteration process. At the end, all the possible states and error states of the program are presented with guidance on how to deal with them.

3.1 App Layout

The main view of the program is presented in Figure 2. The key GUI components are enclosed in enumerated red rectangles. The corresponding explanations for the components' purpose and functionality are found below the figure.



Degree 5

	Coefficients
1	1
2	0
3	0
4	0
5	0
6	-1

Current Polynomial:

$z^5 - 1.0$

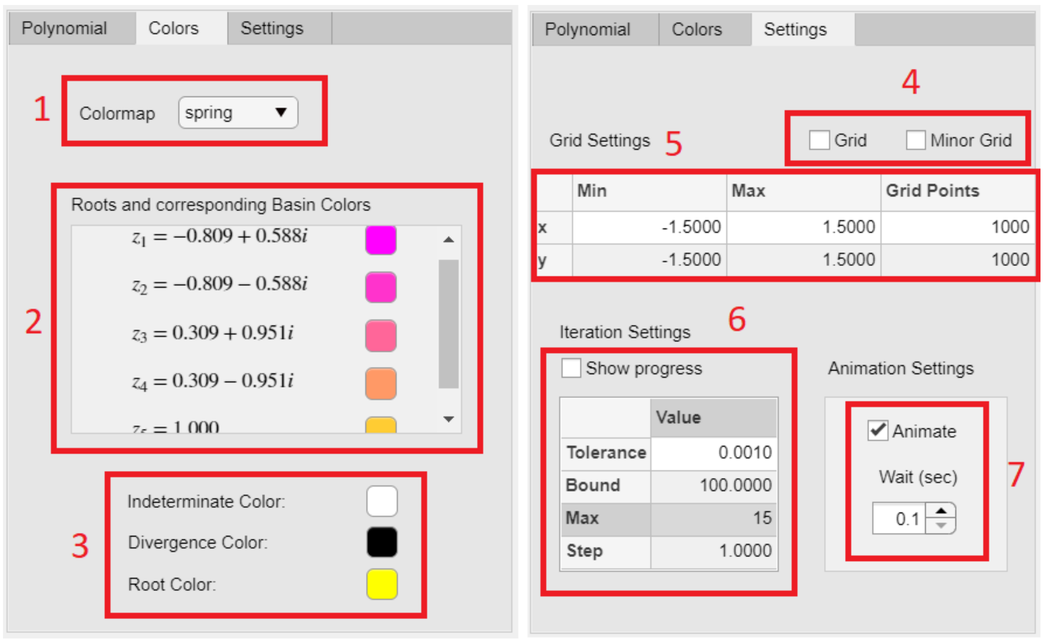
Ready ●

Figure 2: Main view of the program with key GUI components enumerated

1. Main window: Display the calculated basins of attraction for the given polynomial.
2. Group tabs: Avoid cluttering the GUI components.
3. Degree spinner: Set the degree of the polynomial.
4. Randomize button: Set the polynomial coefficients randomly.

5. Coefficient table: Store and display the current coefficients of the polynomial
6. Current polynomial: Display the polynomial interpreted from the given coefficients.
7. Reset Axes button: Clear the main window (1.) and set the axis limits.
8. Start/Abort button: Start/Abort the iteration process for calculating the basins of attraction.
9. State lamp: Display the current state of the program.

In addition to the main view, the user can also interact with the app on the Color and Settings tabs. The contents of these tabs are presented in Figure 3. Its key GUI components are also enclosed in enumerated red rectangles. Their description can be found below the figure.



Colormap: spring

Roots and corresponding Basin Colors

$z_1 = -0.809 + 0.588i$	🟪
$z_2 = -0.809 - 0.588i$	🟪
$z_3 = 0.309 + 0.951i$	🟥
$z_4 = 0.309 - 0.951i$	🟧
$z_5 = 1.000$	🟨

Indeterminate Color:	⬜
Divergence Color:	⬛
Root Color:	🟨

Grid Settings

☐ Grid ☐ Minor Grid

	Min	Max	Grid Points
x	-1.5000	1.5000	1000
y	-1.5000	1.5000	1000

Iteration Settings

☐ Show progress

	Value
Tolerance	0.0010
Bound	100.0000
Max	15
Step	1.0000

Animation Settings

✓ Animate

Wait (sec) 0.1

Figure 3: Color and Settings tabs with key GUI components enumerated

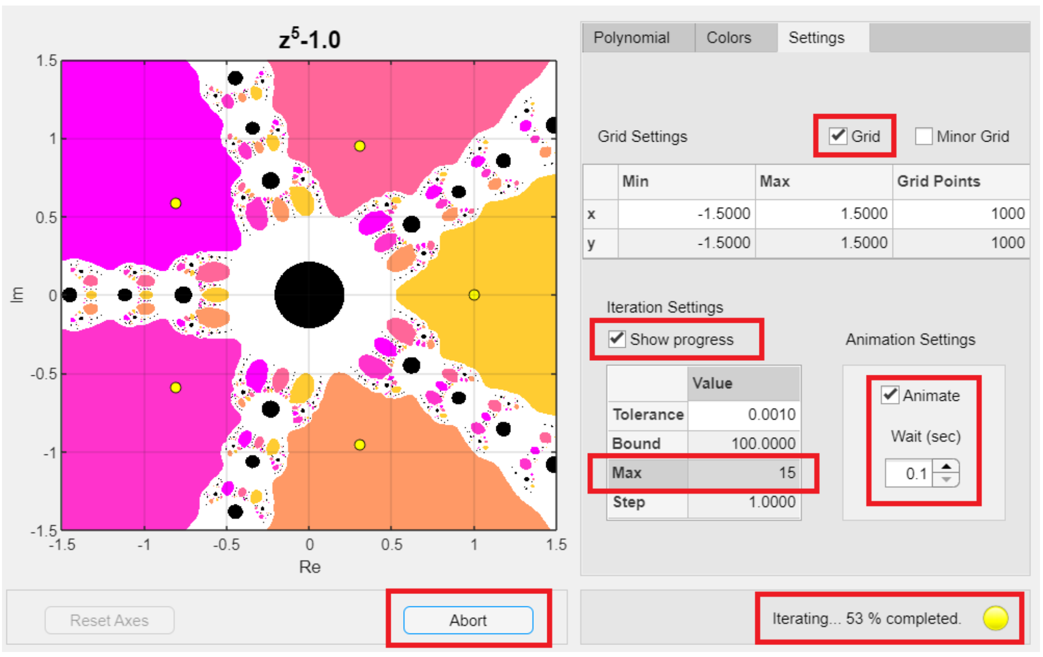
1. Colormap drop down menu: Select the current predefined colormap for basin colors.
2. Basin panel: Display the current roots and their corresponding basin color.
3. Reserved colors: Display the reserved colors.
4. Grid checkboxes: Draw or remove grid lines from main window. (See Figure 2.)
5. Grid settings: Set grid properties.
6. Iteration settings: Set iteration properties.
7. Animation settings: Set animation properties.

The app is designed to be as intuitive as possible. All GUI components serve a logical purpose and their functionalities are deducable from their names. Note that the opening view of the program is similar to Figure 1. Only exceptions are that the main window is empty and the default polynomial is set to $z^3 - 1$. This also changes the views presented in Figure 3. The default colormap is set to "parula" and the maximum number of iterations is 10.

3.2 Calculating the Basins of Attraction

The main purpose for the app is to calculate and visualize the basins of attraction for a given polynomial. First, the coefficient of the polynomial must be given in descending order. The degree dictates how many coefficients there are. As long as the state of the program is ready and the given polynomial is valid, the iteration can be started by pushing the Start button. Depending on the settings, the iterations are either animated or not and the progress can be shown. For further details, see Subsection 3.3.

Figure 4 illustrates the view after pressing the Start button. The state of the program is no longer ready as it is computing the iterations and displaying the results in the main window. Note that pressing the Start button changes its label to "Abort". By pressing it again, the user can abort the iteration process before completion. The progress is only shown if the corresponding checkbox is checked. Progress is simply calculated as the percentage of finished iterations in relation to the maximum number of iterations.



Grid Settings [✓ Grid] Minor Grid

	Min	Max	Grid Points
x	-1.5000	1.5000	1000
y	-1.5000	1.5000	1000

Iteration Settings

[✓ Show progress]

	Value
Tolerance	0.0010
Bound	100.0000
Max	15
Step	1.0000

Animation Settings

[✓ Animate]

Wait (sec) 0.1

Iterating... 53 % completed.

Figure 4: Animating the iterations and showing the progress

With different settings the basins displayed in the main window will look different. After the basins have been plotted, their colors can be modified in the Colors tab. The details of different settings can be seen in the following Subsection 3.3.

3.3 Settings

All polynomial related settings are located in the Polynomial tab. These settings only affect the polynomial in question. To change the plotting resolution, for example, the Settings tab comes in handy. Similarly, Colors tab takes care of color related changes. Although error states are presented more closely in Subsection 3.4, the demands for the data are also presented here. **Note that while the app is iterating no changes to the settings can be made.**

3.3.1 Grid

On the right in Figure 3 all grid related settings are located in the top part of the panel. The user can draw the Grid and Minor Grid lines to the main window by checking the corresponding checkboxes. The grid contains all the initial values for the iteration. It is essentially a large meshgrid of complex numbers the real and imaginary parts of which range with respect to the given min and max values. The number of grid points on each row determines how many linearly spaced points there are in the corresponding x (real) and y (imaginary) direction.

Note that naturally the number of grid points must be a positive whole number and the minimum value must be smaller than the maximum value. Error states are defined in Subsection 3.4. If the other instances of data are not integers, MATLAB will display the integer data with decimals. Despite the this inconvenience, the actual value itself must be an integer.

3.3.2 Iteration

Iteration settings can be found on right in Figure 3 and in the down left corner of the panel. The user can decide to show the iteration progress by checking the corresponding checkbox. Other iteration values include the tolerance for convergence, bound for divergence, maximum number of iterations and finally, the length of the iteration step.

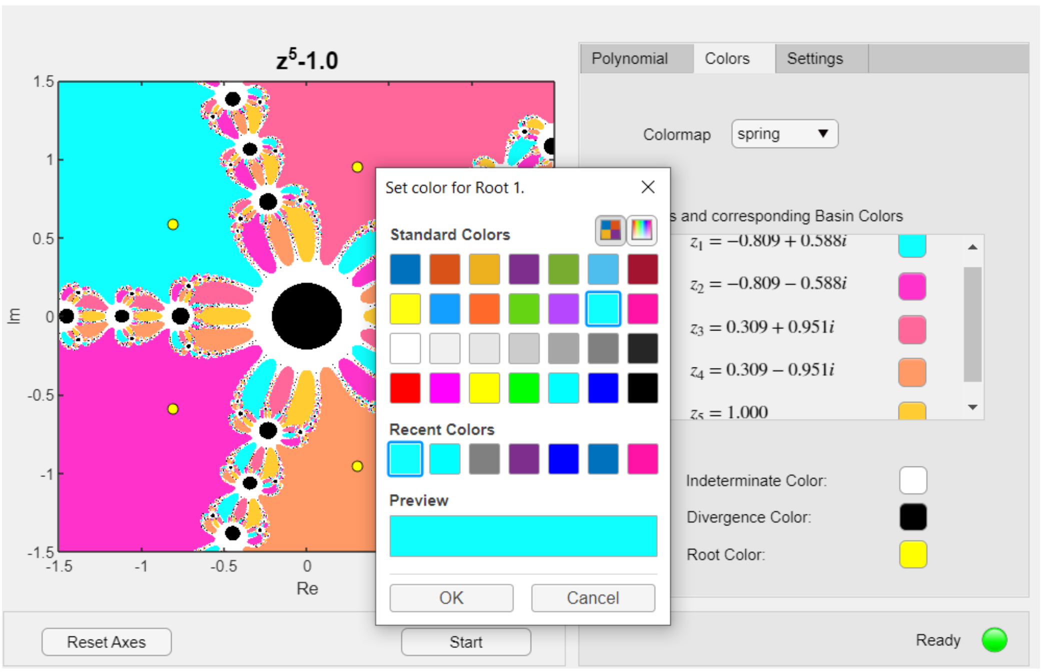
Similarly to the grid settings, the maximum number of iterations and the length of the iteration step must be positive whole numbers. Furthermore, the maximum number of iterations must be larger of the two. Also the tolerance for convergence cannot be larger than the bound for divergence for obvious reasons. No negative numbers are allowed.

3.3.3 Animation

Animation related settings can be on the right in Figure 3 and in the down right corner of the panel. The iteration process is animated if the animation checkbox is checked before the iterating has begun. Underneath the checkbox the user can change the animation speed. The Wait spinner is in seconds although the actual speed varies with each computer. As a rule of thumb, the larger the value, the longer each animation step takes.

3.3.4 Colors

The color settings can be seen on the left in Figure 3. The user can select a predefined colormap from the Colormap drop down menu. There is also an option to select the colors manually for each basin by pushing the corresponding color button on the right of the Basin panel. The color selection is done via uisetcolor and it is shown in Figure 5.



Colormap: spring

s and corresponding Basin Colors

$z_1 = -0.809 + 0.588i$

$z_2 = -0.809 - 0.588i$

$z_3 = 0.309 + 0.951i$

$z_4 = 0.309 - 0.951i$

$z_5 = 1.000$

Indeterminate Color:

Divergence Color:

Root Color:

Figure 5: Selecting the root color manually with uisetcolor

Note that the user cannot pick a color that is already in use. If the user closes the dialog without selecting a color, no changes are made. To change the colors manually, the current state of the program must be ready. By default, the colors for the basins are generated automatically with the chosen colormap. **All custom color changes must be done after selecting a predefined colormap.** Otherwise all custom changes are lost and the new colormap overwrites the basin colors. Four predefined colormaps are shown in Figure 6.

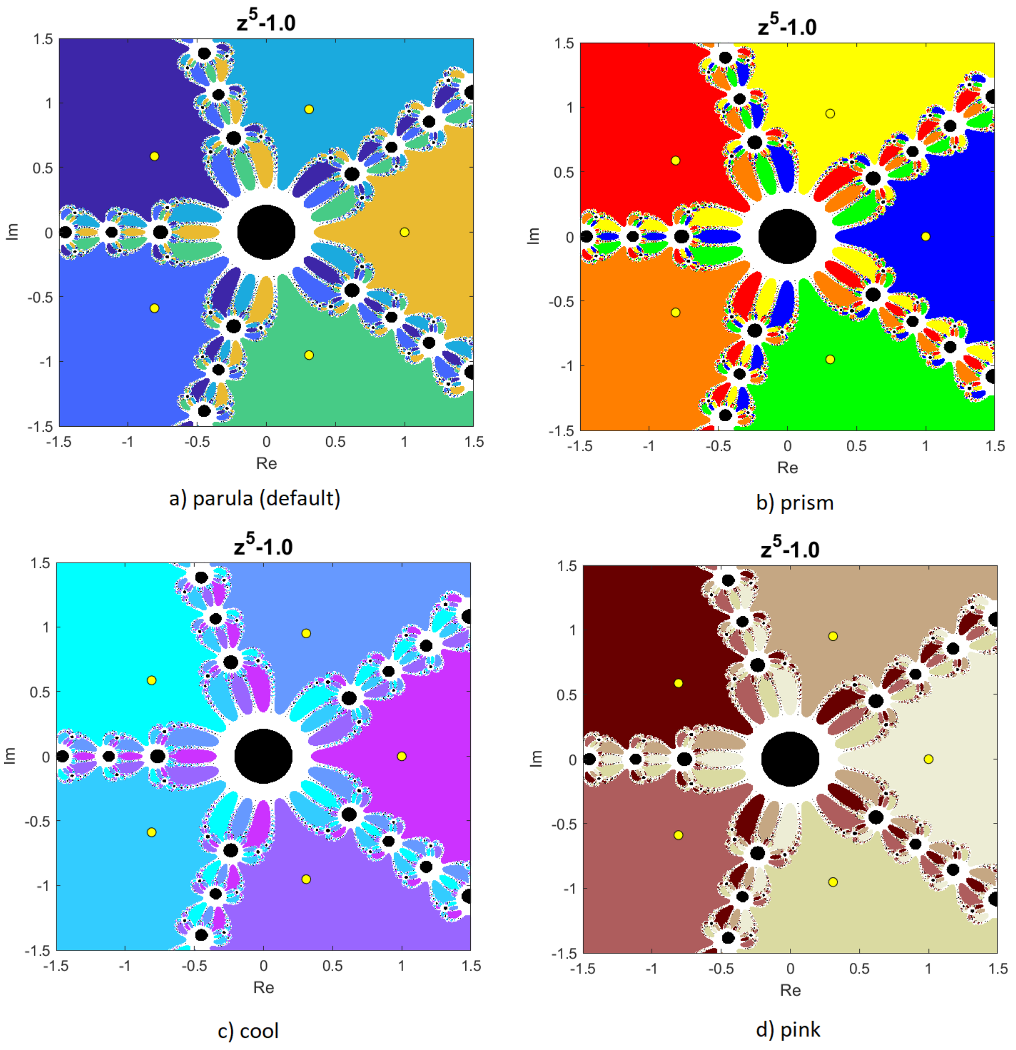


Figure 6: Four out of thirteen different predefined colormaps

The colormap of the main window can be changed after the iteration but only if the current polynomial is the same as the one plotted. Otherwise, the colormap can be taken into use by iterating and plotting a new image.

3.4 Program States

The current state of the program can be either ready, doing something or having an error. Instinctively, the error states are the most worrying and need to be taken care of. If the program is busy doing something, calculating the basins for example, the user simply has to wait until the program is ready. All possible states of the program are listed in Table 1. The table also includes a brief description of the state and how the user is informed of it.

Table 1: All possible states of the program and their descriptions

State	Description	Text displayed for the user
nanError	There are non-numerical data present.	Fix NaN data.
intError	Integer valued data is not an integer.	Fix NON-INTEGER data.
posError	Positive data is not actually positive.	Fix NON-POSITIVE data.
coeffError	The leading coefficient is zero.	Fix leading zero COEFFICIENT.
gridLimError	Grid limits are not possible.	Fix GRID LIMITS.
tolError	The tolerance is larger than the bound.	Fix TOLERANCE and BOUND.
iterBoundError	The iteration step is too long.	Fix iteration MAX and STEP.
colorError	The user selected color is not valid.	Fix basin COLORS.
init	The program is initializing.	Initializing...
iterate	The program is iterating the basins.	Iterating...
abort	The program is aborting the iterations.	Aborting...
tryAgain	The user must try again.	Try again!
ready	The program is ready for use.	Ready

The different states of the program are also conveyed to the user by setting the State lamp color accordingly. A green color is for when the program is ready, a yellow color is shown when the program is busy doing something, and finally a red color indicates that an error has occurred. The error texts should be self explanatory and they tell the user exactly what they need to do. The program also gives feedback instantly if it finds an error. The error state is active as long as all errors have been fixed.

4 Implementation

This Section aims to illustrate the decisions made when implementing the app. The detailed documentation for the functions can be found in the MATLAB code as H1-lines. In Figure 7, the initialization process is shown on a function call level. The arrows represent which functions call which functions to see internal dependencies.

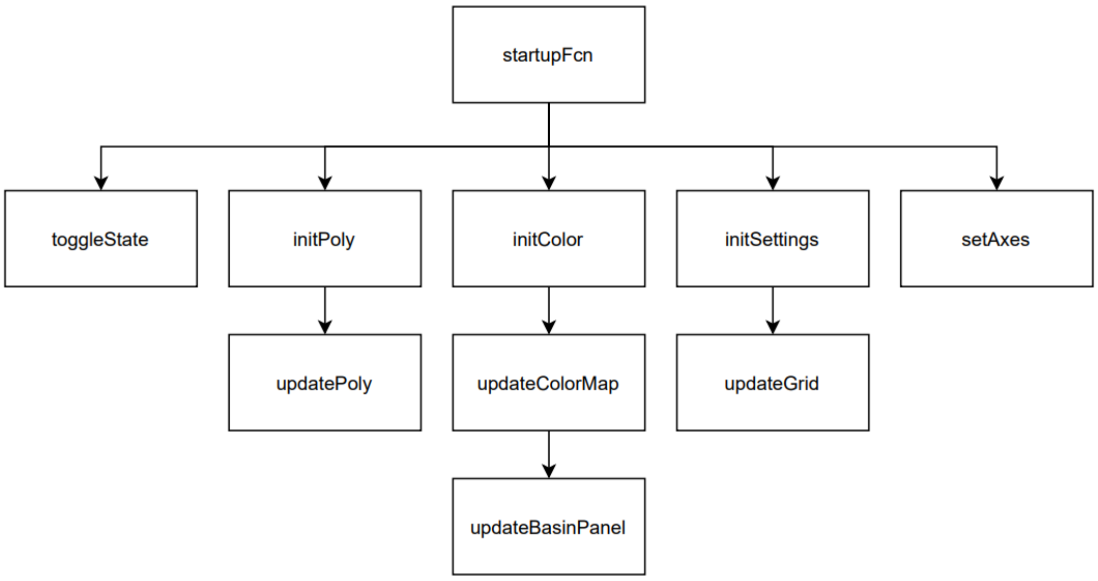


Figure 7: A schematic presentation of the initialization process

After the initialization, the app is ready for user inputs. The user can either change the default settings or start drawing the basins of attraction. If the user changes any data, the validity of the data is checked. Erroneous data prohibits the use of the app. To catch any user related errors, the data are checked after each modification via the general error handler ”checkForDataErrorsAndUpdate”. This process is shown in more detail in Figures 8 and 9.

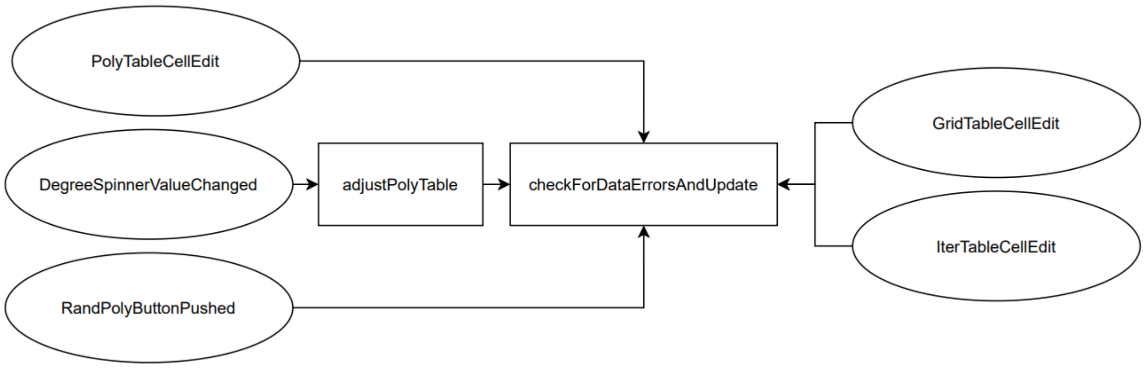


Figure 8: A schematic presentation of user related error checking

If the data are valid, they are updated to their corresponding properties. To maintain any previous unfixed errors, the data must be checked everywhere. This is done with the help of the function ”isValidData” which can catch non-numerical and non-integer data errors, for example. More details can be seen in Figure 9 and Table 1.

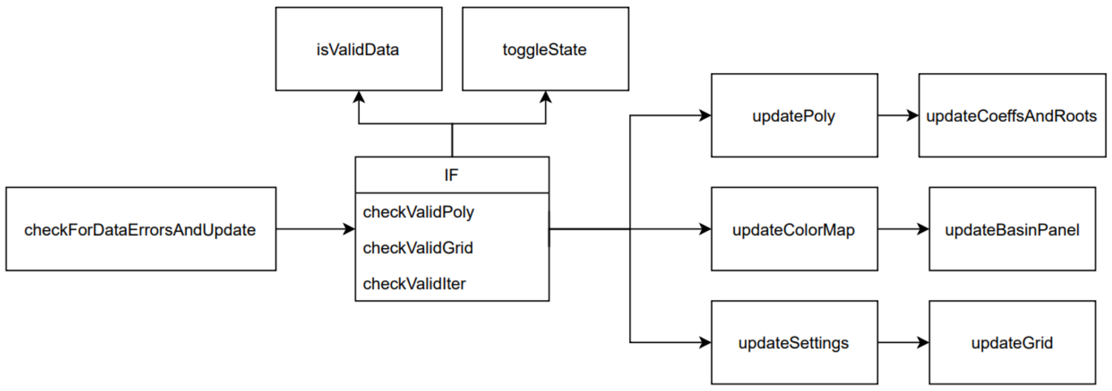


Figure 9: A schematic presentation of catching errors and updating

Note that no updates are made if there are any erroneous data. Also the app is not in its ready state until every error is fixed. The function ”toggleState” takes care of toggling the current state accordingly.

Finally, the rest of the GUI component callbacks and functions related to changing the basin colors are presented in Figure 10 alongside with other miscellaneous processes. The functions ”colorButton-Pushed” and ”uialertClosed” are self made callbacks. The former handles the color button pushing events for changing the basin colors and the latter is called after closing an uialert. The uialert pops up if the user selects an erroneous color.

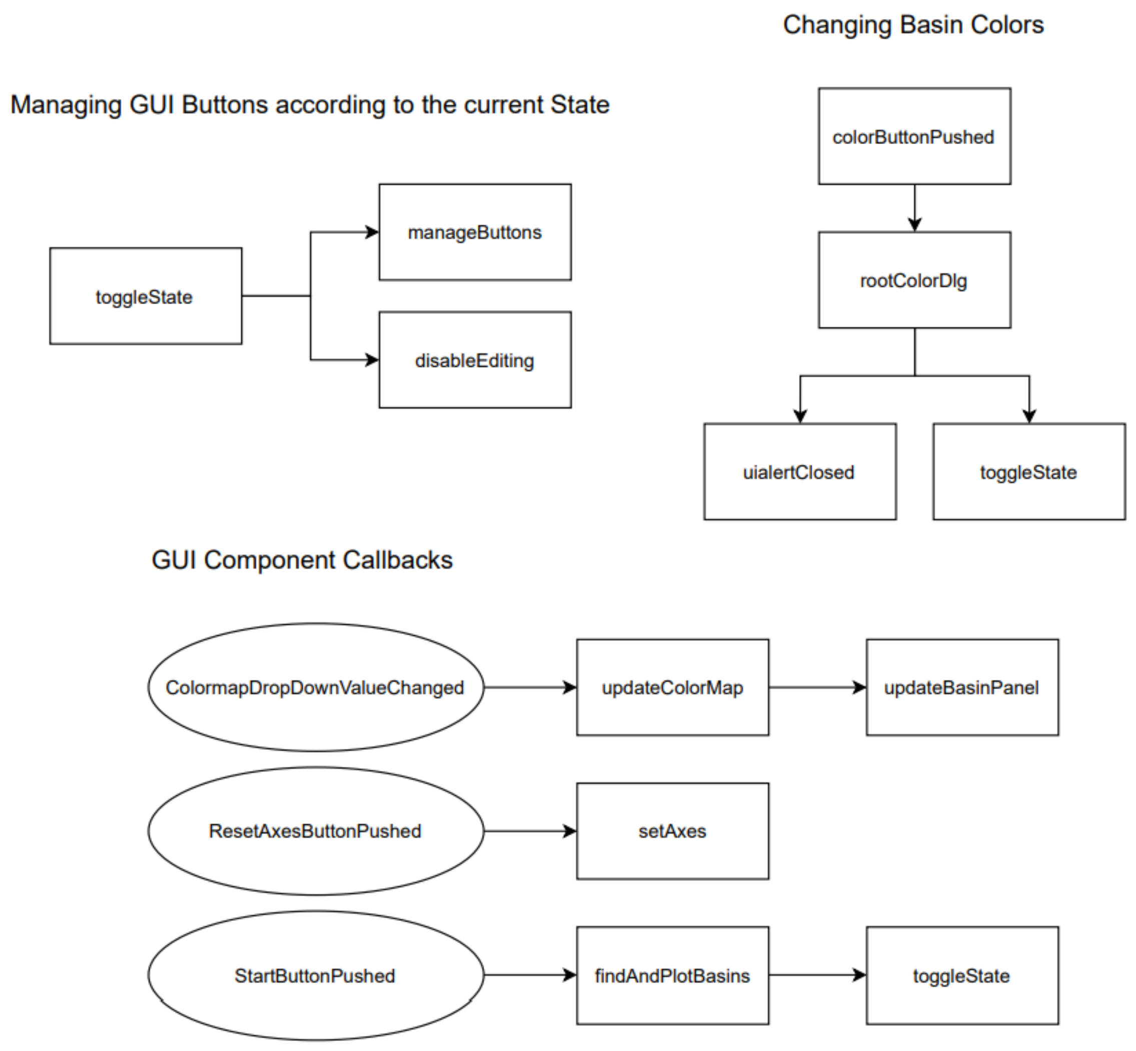


Figure 10: A schematic presentation of miscellaneous processes

Note that the function ”findAndPlotBasins” is responsible for calculating and visualizing the basins of attraction. Other functions take care of the interaction between the user and the app. Having a well-structured function hierarchy helps to pinpoint any abnormalities should they appear during program execution. The functions are also designed to execute logical tasks that should be deducible from their names.

5 Restrictions

In this final Section, the possible restrictions of the app are discussed. The app is coded with the version R2021a of MATLAB. This version has some features that earlier versions do not have. For example, the string interpreter is set to ”latex” to display any texts. The functionality of these and related features are not guaranteed to work properly with earlier versions.

One clear restriction is that symbolic coefficients of the polynomial are not allowed. The coefficients must be numeric values which implies that the user cannot give, for example, π as a coefficient. It must be first converted to numerical value. Also, any kind of parameterized coefficients will not work. The app takes care of checking for any erroneous coefficients.

When it comes to displaying the current polynomial, the labels and titles can only display the polynomial up to a certain length. Due to the LaTeX interpreter used, the labels do not display three dots if the label is too full. The title of the Main window does not have a limit but too long polynomials do not look as nice. Generally speaking, polynomials with larger degree than seven will start to have problems fitting if all of the coefficients are non-zero.

Finally, the degree of the polynomial must be at least one. Constant functions are not interesting as any initial value just stays the same as the value of the function in the iterations. The function does not even have zeros if the constant is non-zero. Otherwise, the function has a zero everywhere and all initial values converge to every zero. This is not desirable so this case is strictly forbidden.