Basins of Attraction of the Henon Map

In this exercise we will be computing the fate of orbits from an array of initial conditions in the Hénon map. At the end we will visualize the basins of attraction of the Hénon map by comparing the 'average divergence per orbit steps'. We will see that some orbits remain bounded and other orbits quickly escape to infinity. For some parameters, we shall be observing 'fractal basin boundaries'.

As usual we start by outlining the programming concepts and then describe the steps to achieve the desired result.

Programming concepts

Functional Programming in MATLAB 2: Mapping Boogaloo

Functional programming, as introduced a week earlier, is a new way to look at MATLAB scripts that trades off slightly more unreadable and obfuscated code for a major advantage; it namely directly ports to a parallelization on the GPU, which allows you to run code much faster. We previously introduced you to the `arrayfun` function as an example of a higher-order function. Higher-order functions are different from regular functions since they do not just take data (in MATLAB nearly always in the form of an array), but can also take functions as their input. What the `arrayfun` function does is apply the function passed to it on all elements of a passed array, such as in

```
arrayfun(@cos, [0, pi/2, pi, 3*pi/2]);
>> ans = [1 0 -1 0]
```

Cell array-based mapping and cellfun

As shown, the `arrayfun` function works on a container. But this container doesn't necessarily have to be an array. An identical higher-order function has been defined for cell arrays, namely `cellfun` (not to be confused with the portable digital communication device in your pocket, that is namely a cellphone).

The function is near-identical to the previously introduced `arrayfun`. It applies a function that takes a function `f` and a cell array `c`, and returns a vector `v`, of which `v(i) = f(c{i})`. This can be useful if your function takes arbitrary-length input.

Another use is a slight workaround, based on the fact that MATLAB has no simple way of applying a function column-wise unless directly built in, like in the `sum` function. To do this, we use the `num2cell` function on a matrix to transform it into a cell array, of which the `i`th element is equal to the `i`th column of the matrix. Subsequently, we use `cellfun` to apply the function to each cell array.

Meshgrids

When exploring a multidimensional state space or a parameter space, you often wish to compute a function over a grid in multiple dimensions. This can be done using embedded for loops. However, not only is this solution somewhat inelegant, MATLAB is not optimized for looping, but for vectorized operations. This results in a second method, which is far more efficient. It relies on using `meshgrid` to create a multidimensional grid over the parameter space or the state space.

Calling a meshgrid via

```
[x, y] = meshgrid(xs, ys);
```

results in two matrices containing the x and y matrix of coordinates for a 2D grid. The size of the grid, as well as the distance between two adjacent grid points, is passed into this function using the parameters `xs` and `ys` in the standard `begin:step:end` format. One can then pass the matrices 'x' and 'y' to a function that takes two coordinates as arguments.

To visualize this, try to imagine that you want to compute a function for coordinates in a 2D grid, with the function being computed for all combinations of x and y values between 1 and 5. This results in the following code:

```
[x y] = meshgrid([1:5], [1:5])
```

Before you go into the exercise, see if you can interpret the output of this function. Also, read the documentation!

Exercise: The Henon map and its stability

We will be producing code that displays the basins of attraction of the Hénon map.

This exercise is composed of four functions. Due to an oversight on our part, we did not ask you to write an `orbit` function that works in any dimensions. To correct this, we will implement an `nd_orbit` function. We then extend this function to `nd_orbit_after_transients` like done last time. We then compute the average orbit step for the Henon map starting at a specific initial value. We then apply a meshgrid to obtain the average orbit step for many values of x_0.

1. `nd_orbit.m`

Create a function that takes a map $f : \mathbb{R}^d \to \mathbb{R}^d$, a starting vector $x_0 \in \mathbb{R}^d$ and an integer number of iterations. Make it return the orbit as a $N_{it} \times d$ matrix. Please assure that you check for a dimensionality mismatch between the map and the initial position.

2. `nd_orbit_after_transients.m`

Implement the Henon map as an anonymous function. How do you make an anonymous function that takes a vector and returns a vector? Subsequently apply the function `nd_orbit` to get the orbit for the Henon map and remove the transients.

3. `average_orbit_step.m`

Obtain the orbit. Subsequently, for an orbit, obtain the average orbit step given by

$$s = \frac{1}{N-1} \sum_{i=1}^{N-1} \|x_{i+1} - x_i\|_2, \tag{1}$$

where $\|\cdot\|_2$ denotes the base-2 p-norm, and $x_i \in \mathbb{R}^2$, and return this value for s. A clever way to do this is by shifting operations. Don't forget to keep dimensionality in mind, since you are applying a norm function on columns!

Note that for many initial conditions the points escape to infinity, and thus a distance function will return a `NaN`. That's okay!

4. `average_orbit_on_grid.m`

Now, to explore the phase space, we want to get this s value for many different options of x_0. To do this, use the meshgrid and apply a function on all elements on the grid; this returns a matrix S where $s_{i,j}$ is the average orbit step for these indices of the meshgrid. For an example without fractal basin boundaries, take $a = 1.28, b = -0.3$. For an example with Fractal basin boundaries take $a = 1.4, b = -0.3$.

Subsequently, plot S as a heatmap. If the function `imagesc` does not work, please install the bioinformatics toolbox or talk to a TA. Like always, use figure handles to allow the unittests to work. Also return the matrix S in order to let the unittests do their job.

Further Explorations:

- Extend function 4 to compute the Lyapunov exponent
- Plot the equilibria of the Hénon map
- Calculate the stability of the equilibria