Project 1: 1D Maps (part I)

Learning goals

- Students learn the essential vocabulary of dynamical systems.
    - state, state variable, parameter, orbit (or trajectory), fixed point (equilibrium), attractor.
- Students code a function that iteratively computes an iterated map.
- Students understand how to define and call an anonymous function (lambda function) in matlab.
- Students use their fuction to study the dynamics of the logistic map.
- Students visualize the orbits of a 1D dynamical system via the cobweb plot.

1D Maps: one dimensional discrete dynamical dystems

Definitions

Our entry point for the vast field of dynamical systems is the 1D discrete dynamical system. Such systems are often used to model the temporal evolution of processes that are measured periodically (also called, stroboscopic), such as daily counting cells in a petri-dish. We will begin our project by producing **iterated maps**, focusing on the analysis of **orbits,** i.e., sequences of states emerging from the repeated application of a **mapping function**, which takes a given **state** into the future, one iteration at a time. We will gauge the behaviors of orbits **qualitatively**, through geometrical analysis via the **cobweb plot**.

A **map** is a function whose range and domain are the same. For example, F is a map that takes a state x from the real numbers R on to the next state :

$F : x \rightarrow x | x \in R$

The orbit is the sequence of states from the repeated application of the map. For example, if a certain system has a x(1) as the first state, the application of F to x(1) results in the next iterate: x(2) = F(x(1)), x(3) = F(x(2)) and so forth, x(t+1) = F(x(t)). Notice that the map is a **recursive** function!

We conveniently represent the kth iterate of the function F by F^k. The sequence of states in the orbit O as the sequence of $x(t)$, as {x(1), x(2), x(3), ..., x(it)}.

Iterate (k)	Orbit	Recursion	Also
0	x(1)	x(1)	$F^0(x_0)$
1	x(2)	F(x(1))	$F^1(x_0)$
2	x(3)	F(F(x(1)))	$F^2(x_0)$
3	x(4)	F(F(F(x(1))))	$F^3(x_0)$
4	x(5)	F(F(F(F(x(1)))))	$F^4(x_0)$

We will be studying the long term consequences of orbits for different maps. For that we first will create code that computes orbits for arbitrary 1D systems.

Computing orbits for arbitrary 1D systems

Here we will write a MATLAB function that iteratively applies an arbitrary function (e.g., the logistic map) to an **initial condition** (x_0), for a certain number of iterations (it). For this task we will introduce the concept of **anonymous functions** in MATLAB.

Anonymous functions

Anonymous functions are an abstraction borrowed from **functional programming** languages such as LISP and Mathematica (closely related to 'lambda functions' in python, LISP). There are many uses for anonymous functions. You can use an anonymous function to:

1. define a simple function without needing to create a file
2. pass functions as arguments to other functions
3. compute a function on entire arrays or multiple arrays
4. Simplify function calls for functions with many arguments
5. use MATLAB's GPU computing (if you have nVidia graphic cards and CUDA installed)

As an example, this is an anonymous function `deg2rad` that converts degrees into radians:

```
>> deg2rad = @(x)  pi * x / 180
```

To which MATLAB answers:

```
    deg2rad =
  function_handle with value:
    @(x)pi*x/180
```

This function can be called as such:

```
>> deg2rad(360)
ans =
    6.2832
```

The argument can be vectors or matrices as well:

```
>> deg2rad([360 270 180 90 0])
ans =
    6.2832    4.7124    3.1416    1.5708         0
```

In this case, standard matlab rules for vector and matrix handling apply (elementwise operations, for example).

1. CODE: Write a function that computes orbits for an arbitrary map

Let's call the function `orbit`. The arguments for this function are:

```
function output = orbit(fnct, x0, it)
% Where:
%- fnct: a 1D map defined as an anonymous function
%- x0: an initial condition
%- it: the number of iterations to compute forward
```

One example function call could be:

```
flip_half = @(x) -1/2*x % define a function that flips the sign and halfs the
value

orbit(flip_half, rand, 10)
```

Question: What is the long term behavior of this orbit?

Cobweb plot

Now we will build a facility that helps us to **analyze the long term behavior of the orbit for a variety of initial states** via a graphical display. For reasons that will become evident later, we call this the **'cobweb plot'**. The cobweb plot is also a 'first return map', where we plot the next state as a function of the previous state. That is, the axis of the cobweb plot are x(t) and x(t+1) and *every point in the plot is defined by its past and present state* (thus, every state along the orbit appears twice).

2. CODE: Write a function that displays a cobweb plot from an orbit

```
function fighandle = cobweb(fnct, orbit, varargin)
% fnct :     function handle to the mapping function
% orbit:     the sequence of states in an orbit as outputted via the function
'orbit'
% varargin:  familiarize yourself with varargin via matlab help
% fighandle: handle to the figure created

%sample usage for varargin
if not(isempty(varargin))
```

```
    if ishandle(varargin{1})
    % if we received a figure handle, use that figure
    fighandle = varargin{1};
    else
    % create new figure
    fighandle = figure;
    end
end
```

Requirements:

- The function takes as input the output of the 'orbit' calculation and plots 2 axes
    - **First axis:**
        - must display the identity function f(x) = x along with the cobweb.
        - must display the mapping function `fnct`.
        - plot the cobweb from the orbit (specified below).
            - The first point of the cobweb is the initial condition orbit$(x(1), x(2))$, thus assuming 0 as the previous state (since there's no state before the initial condition).
            - For every new state we plot two line segments.
                - `line([x x], [x fct(x)]); % cobwebline_iteration`
                - `line([x fct(x) ], [fct(x) fct(x)]); % cobwebline_identity`
    - **Second axis:**
        - plot the **time series** of the orbit. That is, on the x-axis are the iterations of the function (1,2,3,4,5,6) and on the y-axis the values of the function at f(x(1)), f(x(2)) and so forth.
- `varargin` can be used to pass a variety of parameters. We will use it as a toggle that decides whether we will clear the plot or we use the same plot as earlier.
- If we receive a figure handle as an argument, the function cobweb must be able to display the orbits 'additively'. That is, for any new orbit, **we add the orbit to the same axes as previously** and use a different color for each new orbit. `varargin` can be used for many other purposes, such as to clear the axis.
    - The function cobweb must take as input the handle to a figure and plot the additional orbits non-destructively.
    - We will use a nice coding practice here by using handles for figure, axes and data. Similar abstractions are used in python's mattplotlib . Using handles allow for superior organization of code, increased execution efficiency and much better memory management. Not to mention that it dispenses with the (ugly) use of `clc`, `hold on`.

Tutorial: Using Handles for figures, axes and other objects

```
  % Creating Handles is Cool and Clean:

  >> fig = figure %create figure and return the handle 'f'
  fig =
    Figure (1) with properties:

        Number: 1
          Name: ''
         Color: [0.9400 0.9400 0.9400]
      Position: [440 378 560 420]
         Units: 'pixels'
```

Calling figure always returns a handle, and now we can start to use it. One can for example change any **property** of the handle. Like this we can change the values of the fields:

```
  fig.Color = [1 1 1] % a nicer white background for our new figure
  fig.Name = 'a white canvas'
```

And similarly,

```
>> ax = axes %create the axes and return the handle 'a'
ax =
  Axes with properties:
             XLim: [0 1]
             YLim: [0 1]
           XScale: 'linear'
           YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1300 0.1100 0.7750 0.8150]
            Units: 'normalized'
```

Calling the plotting function 'line' also returns a handle, which has very neat consequences:

```
>> l = line([0 1], [0 1])
l =
  Line with properties:
              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [0 1]
              YData: [0 1]
              ZData: [1x0 double]
```

One now has access to all sorts of line properties, including the series data in the line `Xdata,YData` . This will become crucial when producing animated versions of plots and producing videos, as we will be doing in the future. But in addition, every time one calls 'line' in this graph, it is added to the current axis. For example:

```
% Create two lines in the same axes and see their handles with 'ax.Children'
>> l = line([0 1], [1 1]);
>> l = line([0 1], [1 0]);

>> ax.Children
ans =
  2x1 Line array:

  Line
  Line
```

Interpreting the Cobweb plot

The **cobweb** plot helps us visualize the orbits from a variety of **initial states**. (Note: the orbit in the cobweb plot is the discrete sequence of states from the iterated application of the map and **not** the segments connecting them).

3. Experiment: Examine the behavior of some functions for multiple random initial conditions.

Use the cobweb plot to study the long term behavior of the following maps, for any random initial condition x_0 (from the real numbers).

3.1 f(x) = -0.5x

3.2 f(x) = sin(x)

3.2 f(x) = cos(x)

Fixed Points (Equilibria)

Fixed points are points in **state space** where the application of the mapping function leads to the exact same state, or in other words, states in **equilibrium**.

$f(x^*) = x^*$; where x^* is the fixed point.

Orbits starting in the fixed point remain in the fixed point. **Graphically**, fixed points of one dimensional maps are the points where the **identity line intersects the mapping function**.

Fixed points come in different sorts, and can be categorized as a function of their **stability**. If orbits starting close to the fixed point return to the fixed point, the fixed point is **stable,** and in this case we find an attractor. In the converse case, where orbits starting close move further away from the fixed point, the fixed point is **unstable**, we have a **repellor**.

Clearly, **a 1D map can have any number of fixed points**, depending on the number of times the mapping function intersects the identity line.

To determine the category of fixed points (attracting or repelling) we calculate the **derivative of the map** at the fixed point. If the modulus of the derivative is larger than one, the fixed point is repelling, and attracting in the reverse case.

An attractor will draw closeby orbits asymptotically to itself. **The part of the orbit before the attractor is reached is called the transient**. As the attractor is never truly reached, **the duration of the transient is somewhat subjective**.

As the fixed point stability is a function of the local derivative, the **category of the fixed points can change** as the mapping function changes. This we call a **qualitative change** in the behavior of orbits. Consider for instance the following map:

$f(x) = a.x$

Where a is a **parameter**. Clearly, when $a < 1$ the fixed point at the origin is attracting, and in the converse case, $a > 1$, repelling.

In this map, we can see that when a = -1, something interesting happens, we obtain a flip-flop behavior. That is, we obtain a **periodic orbit** of period 2. That happens because applying the mapping function twice, takes any state into the same state:

$f(f(x)) = x, \forall x.$

The Logistic Map

This week and the next we will be discussing one of the simplest models of dynamical systems, which notwithstanding its simplicity, displays considerably complex behavior: **the logistic map**. It models the evolution of a population x that undergoes exponential growth with rate a and is bounded to a maximum population (also, carrying capacity) of 1 (representing the 'maximum'). The carrying capacity can be thought as the space available in a petridish, for example. Hence, for a certain population size x, the space available for growth is 1-x. These considerations results in a model that is a parabola:

$f(x) = ax(1 - x)$

Parameterized Dynamical Systems

We will now consider what happens to the dynamics of a system for **smooth changes** of the mapping function. In particular, we consider the different orbits of the **logistic map** for varying parameter a. **Each value of the parameter specifies a unique dynamical system**. To examine the long term behavior of orbits for different dynamical systems, we explored the **parameter space** of the logistic map.

4. Parameter Space Experiments for the Logistic Map:

Display orbits in the cobweb plot for a small selection of random initial conditions for the logistic map for varying the parameter a (for example, taking the parameter a in the interval [0-4]).

Questions:

- Is there anything common to all these orbits?
- Will the orbits always coverge to an equilibrium for any a?
- How many equilibria are there in the map? Does this change as a function of a?
- Can we find mathematical conditions to determine the point where the **qualitiative behavior** of the orbits change?
- Derive Formula: In the logistic map, what is the formula that returns the position of the equilibria as a function of the parameter a? Can you find the equilibria for 'any functions'?
- Can you see in the cobweb plot what makes an equilibrium stable or unstable?
- In the logistic map, at what value of a we see a change of behavior, from a fixed-point attractor to a periodic attractor?

Stability of Equilibria

Orbits starting with initial conditions close to equilibria (i.e., $x^* + \epsilon$) may display different behavior: some equilibria attract nearby orbits, some repell. These fixed-point equilibria are aptly denominated **attractors** and **repellors**. Can you figure out a condition that is able to distinguish between these different kinds of fixed-points?

We note that for some values of a (particularly those after 3, for example) we observed that orbits did not settle in fixed-points, rather assuming more interesting behavior, with time-series seemingly oscillating with multiple fixed points. Can we find a mathematical conditions for the point where the **qualitative behavior** of the function changes? Can you figure out a way to prove it?

Important Theorems

- Stability of Fixed Points (Ref [1] Theorem 1.5, pg. 10)
- Banach Fixed-Point Theorem: https://en.wikipedia.org/wiki/Banach_fixed-point_theorem

(Note that there exist many fixed point theorems, for instance, the Brower's fixed poind theorems)

References and Further Reading for This Week

[1] Chaos - Aligood and Yorke (Chap. 1.1, 1.2, 1.3 - next lesson 1.4, 1.5)

[2] Dynamics and Bifurcations - Hale and Koçak (Chap. 3.1 and 3.2)

[3] Non-linear oscillations - Holmes and Guckenheimer

[4] Non-linear Dynamics and Chaos - Strogatz (10.1 and 10.2, next lesson 10.3)