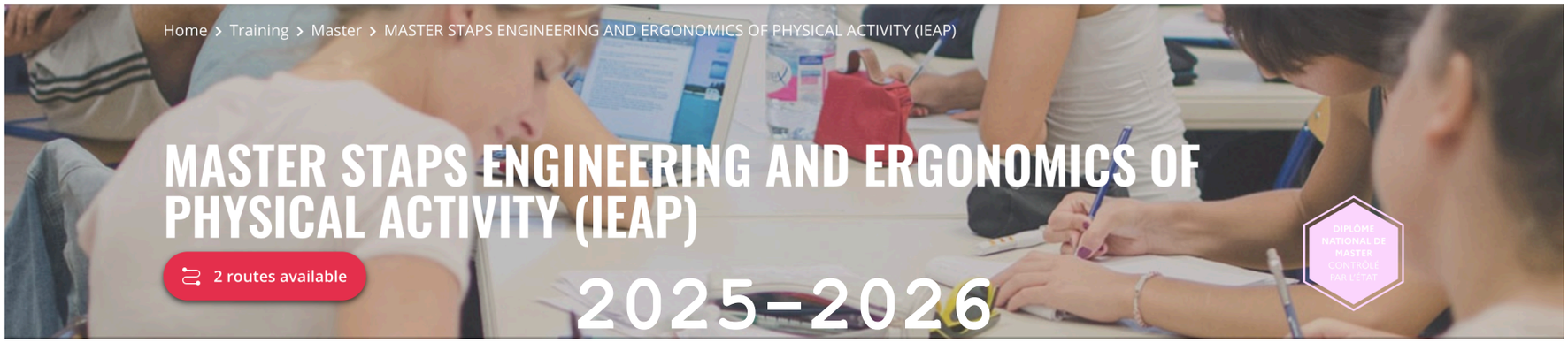
Modeling using Dynamical Systems

SAMID1MU

Mathieu Desroches

Inria



https://www-sop.inria.fr/members/Mathieu.Desroches/

https://team.inria.fr/mathneuro/

The program (see Moodle)

Master IEAP, SAMID1MU: **Modeling Using Dynamical Systems**
Mathieu Desroches, Inria Branch at the University of Montpellier[1]

Program of the course

Dynamical systems: The mathematics of change / Discrete systems [week 1]

- *Input/Output* relationships, *dynamical system*
- *State*, *State variable*, *State space*
- How to deal with time? *Discrete* vs. *continuous*
- Systems: *Linear* vs. *nonlinear*
- *Cobwebs* (discrete time) & *Euler*'s method (discretized time)
- **Examples:** basic maps (traffic light, walking/running, logistic, . . .)
- **Homework:** TBA

1D differential equations [week 2]

- What is a stationary solution or *equilibrium* point?
- What does it mean for this equilibrium to be *stable / unstable*?
- *Phase lines*
- **Examples:** population growth, RC electrical circuits, chemical reactions (autocatalysis), . . .
- **Homework:** TBA

2D differential equations [weeks 3]

- 1D systems cannot oscillate! Intermediate: *phase oscillator* (example: overdamped pendulum, Kuramoto model, quadratic-integrate-and-fire model)
- Notion of *nullclines* and *phase-plane analysis*
- Equilibria with complex eigenvalues: *damped* oscillations; linear systems classification with *tr.* and *det.*
- *Robust* oscillations without friction: harmonic oscillator, pendulum
- *Linearization* of a 2D linear system: *Jacobian* matrix, stability (*eigenvalues*, *eigenvectors*)
- Robust oscillations: *limit cycles*
- **Examples:** van der Pol / FitzHugh-Nagumo (from electrical circuit to neural membrane), population models (e.g., Lotka-Volterra), . . .
- **Homework:** TBA

Mechanisms of oscillations [weeks 4]

- *Feedback* loops
- Restorative vs regenerative forces
- Destabilization of equilibria or *bifurcations*
- Oscillations in *nature*
- **Examples:** genes/hormone regulation, neuronal spikes, muscle tremor, HKB model of human coordination
- **Homework:** TBA

[1] mathieu.desroches@inria.fr · https://www-sop.inria.fr/members/Mathieu.Desroches/

The program [week 1]

Dynamical systems: The mathematics of change / Discrete systems [week 1]

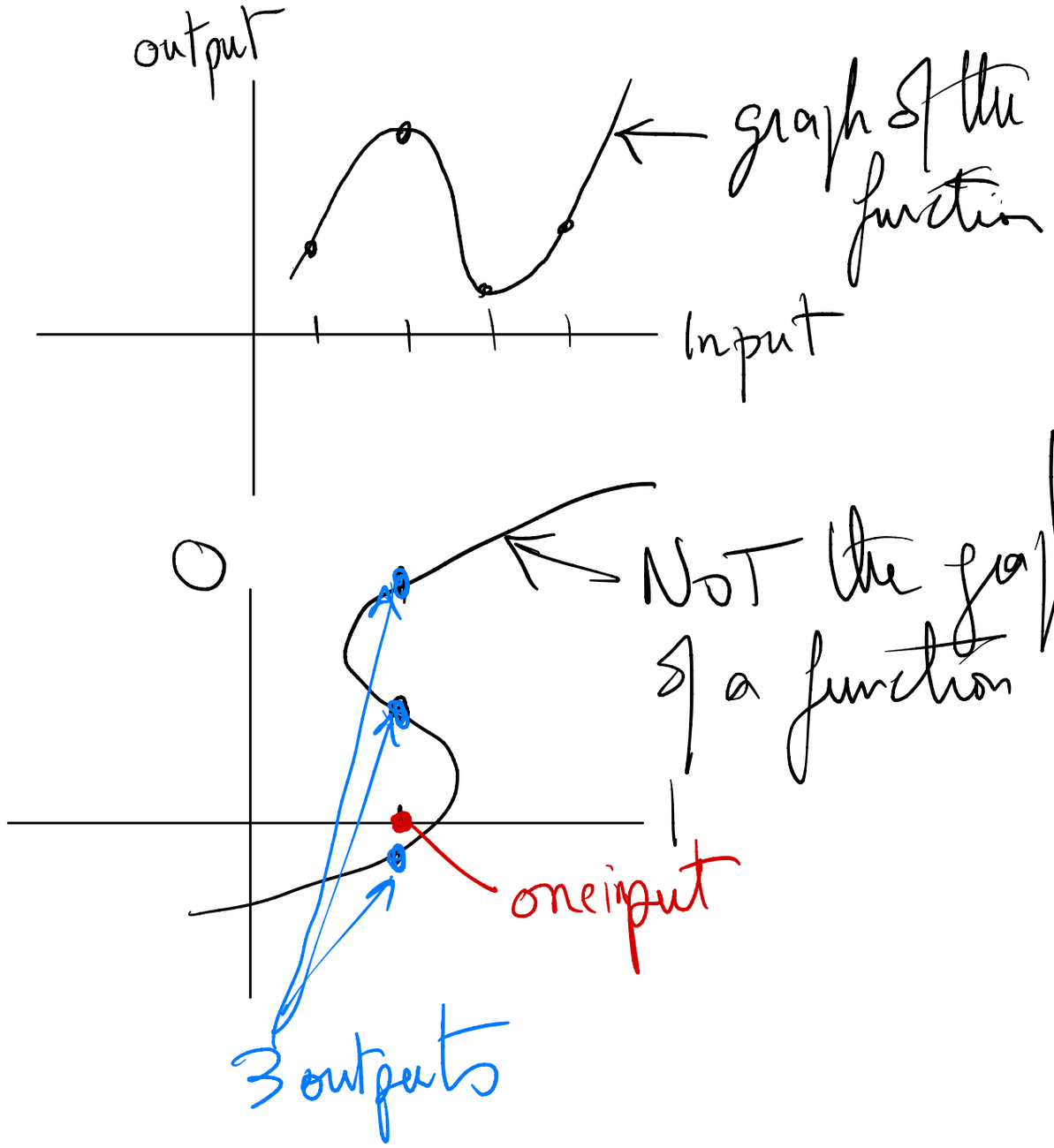
- *Input/Output* relationships, *dynamical system*
- *State*, *State variable*, *State space*
- How to deal with time? *Discrete* vs. *continuous*
- Systems: *Linear* vs. *nonlinear*
- *Cobwebs* (discrete time) & *Euler*'s method (discretized time)
- **Examples:** basic maps (traffic light, walking/running, logistic, . . .)
- **Homework:** TBA

Dynamical Systems

The mathematics of change

Functions

- **one** input → **one** output
- fundamental building block for describing relationships in systems

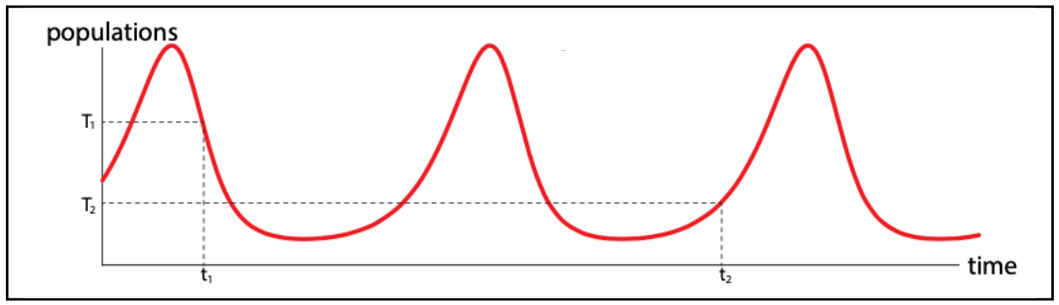


Functions

- **one** input → **one** output

- fundamental building block for describing relationships in systems

- prey population in a population-prey dynamical system; such a function of time is called a **time series**.



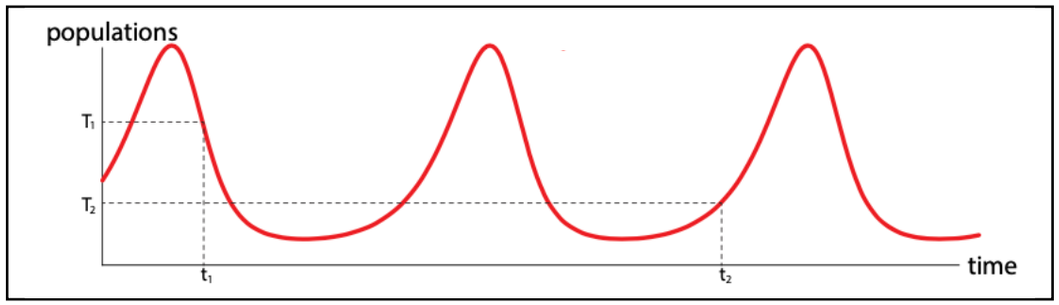
Functions

- **one** input → **one** output

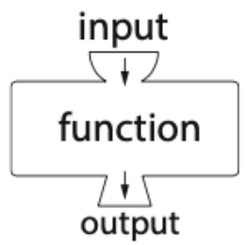
- fundamental building block for describing relationships in systems

- prey population in a population-prey dynamical system; such a function of time is called a **time series**.

f (time)



- functions as (predictable) machines



What is a dynamical system?

- a description of a phenomenon whose **state evolves in time** → ‘dynamical’

What is a dynamical system?

- a description of a phenomenon whose **state evolves in time** → ‘dynamical’

- described using appropriate **(state) variables** and **equations**

What is a dynamical system?

- a description of a phenomenon whose **state evolves in time** → ‘dynamical’

- described using appropriate **(state) variables** and **equations**

- *system vs. model*
  - **system:** often refers to the mathematical equations
  - **model:** can be broader (e.g., an “animal model” in biology)

x : real number

function: $f(x) = x + 2$

Variables, States, State Space

- **variables:** the quantities used to describe the system (model)
- **state:** the values of all variables at a given time; a **function of time** (time series)
- **state space:** the set of all conceivable states

Variables, States, State Space

- **variables:** the quantities used to describe the system (model)

- **state:** the values of all variables at a given time; a **function of time** (time series)

- **state space:** the set of all conceivable states

in this course

- state space is **continuous:** 1 (phase line) or 2 vars (phase plane)
- a state is a real number (1D) or a couple of real numbers (2D)

Variables, States, State Space

- **variables:** the quantities used to describe the system (model)

- **state:** the values of all variables at a given time; a **function of time** (time series)

- **state space:** the set of all conceivable states

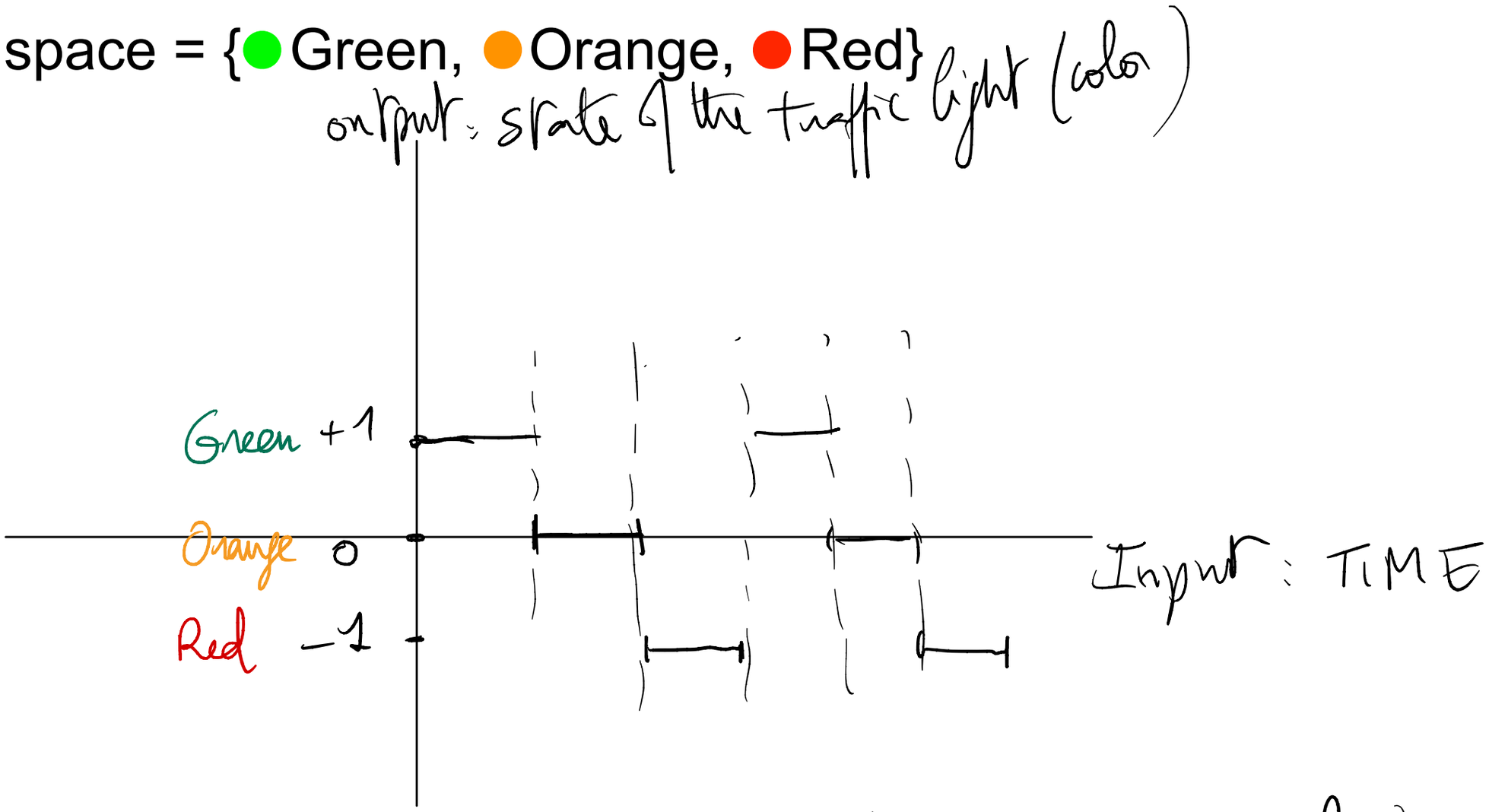
in this course

- state space is **continuous:** 1 (phase line) or 2 vars (phase plane)
- a state is a real number (1D) or a couple of real numbers (2D)

we will be *rarely* interested in the state of a system at a particular time, but rather we want to understand the system's *behavior*, that is, its changes from state to state, and why it exhibits one pattern of behavior (one can call it *activity* too) rather than another.

Examples of (discrete) state space

- **Traffic Light**
  - state space = {● Green, ● Orange, ● Red}



Describe: 1) it's periodic (cycle)
2) it oscillates

Examples of (discrete) state space

- **Traffic Light**
  - state space = {● Green, ● Orange, ● Red}
  - behaviour: ● → ● → ● → ● → ● → ● → …

Examples of (discrete) state space

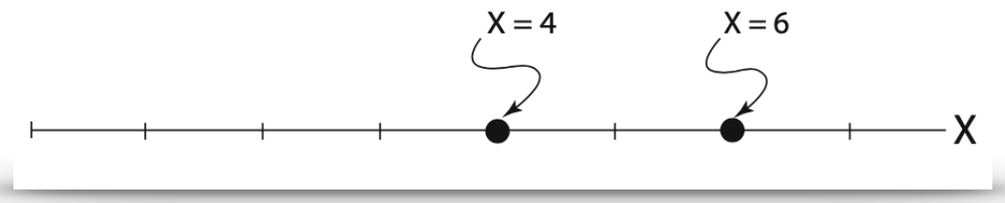
- **Traffic Light**
  - state space = {● Green, ● Orange, ● Red}
  - behaviour: ● → ● → ● → ● → ● → ● → …
- **Human movement**
  - state space = {Walking, Running}

Examples of (discrete) state space

- **Traffic Light**
  - state space = {● Green, ● Orange, ● Red}
  - behaviour: ● → ● → ● → ● → ● → ● → …
- **Human movement**
  - state space = {Walking, Running}
  - behaviour (Walking): equilibrium in the “Walking” state

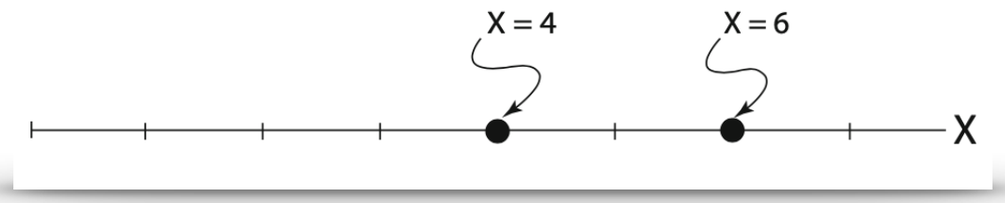
Visualising states: phase line & phase plane

- **1-variable system**
  - state is a point on a line (the **phase line**)

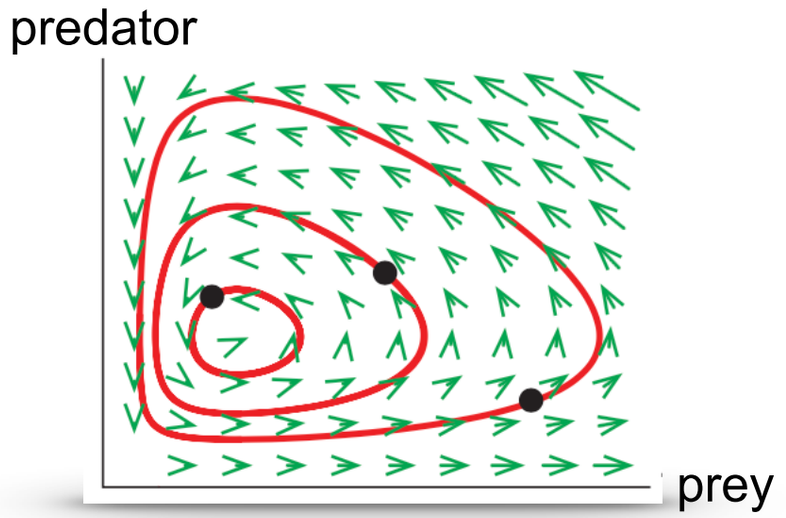


Visualising states: phase line & phase plane

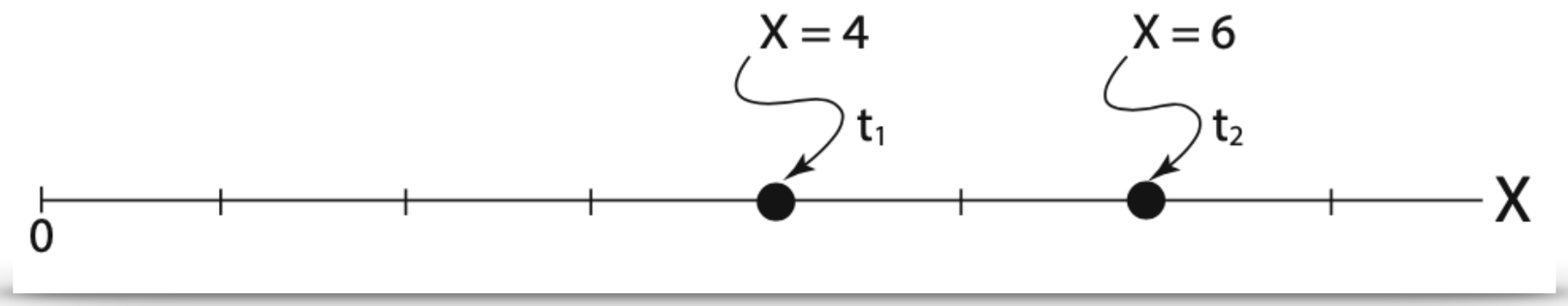
- **1-variable system**
  - state is a point on a line (the **phase line**)



- **2-variable system**
  - state is a point in a plane (the **phase plane)**



Change is movement in state space



- when a system changes, its state changes
- (fig. above) the system has changed from X=4 at time t_1 to X=6 at time t_2

- **Modeling change:** going beyond description to hypothesise the **causes** of this movement

- we study behaviour, not just states at fixed times

How to deal with time: discrete vs. continuous

- **Discrete Time**
  - time advances in distinct states: $t = 0, 1, 2, 3, \ldots$
  - data is collected at specific instants
  - dynamics is governed by **function iteration**: $X_{k+1} = f(X_k)$

$k=2$: $X_{k+1} = X_3$: state of X at time $t=3$

How to deal with time: discrete vs. continuous

- **Discrete Time**
  - time advances in distinct states: $t = 0, 1, 2, 3, \ldots$
  - data is collected at specific instants
  - dynamics is governed by **function iteration**: $X_{k+1} = f(X_k)$

- **Continuous Time**
  - time is a continuous variable t
  - dynamics is governed by **differential equations**

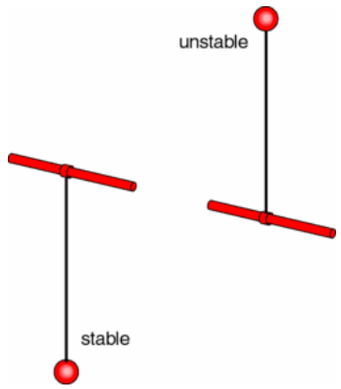
Discrete Systems

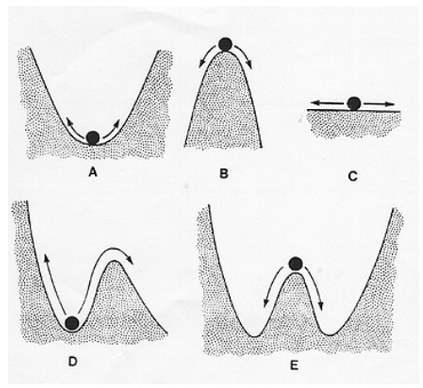
Linear vs. Nonlinear Systems

- **Linear System**
  - evolves by **proportionality**
  - discrete form: $X_{k+1} = RX_k$
  - **stability:**
    - stable if |R|<1
    - unstable if |R|>1
  - *examples: simple bank interest; population Growth*

Linear vs. Nonlinear Systems

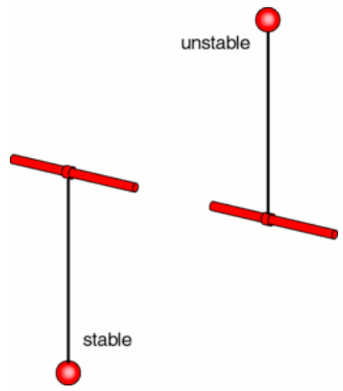
- **Linear System**
  - evolves by **proportionality**
  - discrete form: $X_{k+1} = RX_k$
  - **stability:**
    - stable if |R|<1
    - unstable if |R|>1
  - *<u>examples</u>: simple bank interest; population Growth*

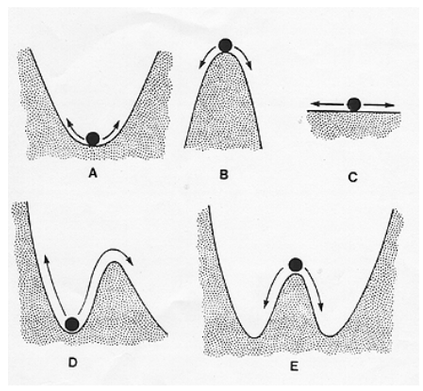




Linear vs. Nonlinear Systems

- **Linear System**
  - evolves by **proportionality**
  - discrete form: $X_{k+1} = RX_k$
  - **stability:**
    - stable if $|R|<1$
    - unstable if $|R|>1$
  - *examples: simple bank interest; population Growth*

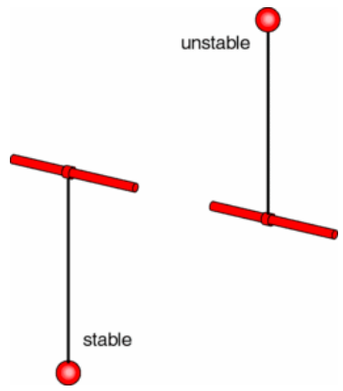


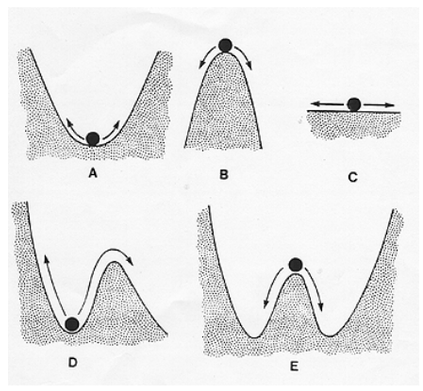


- **Nonlinear System**
  - the rule of change is *not* simply proportional
  - discrete form: $X_{k+1} = f(X_k)$ where f is a nonlinear function
  - *examples: discrete logistic map (better population model)*

Linear vs. Nonlinear Systems

- **Linear System**
  - evolves by **proportionality**
  - discrete form: $X_{k+1} = RX_k$
  - **stability:**
    - stable if $|R|<1$
    - unstable if $|R|>1$
  - *examples: simple bank interest; population Growth*





- **Nonlinear System**
  - the rule of change is *not* simply proportional
  - discrete form: $X_{k+1} = f(X_k)$ where f is a nonlinear function
  - *examples: discrete logistic map (better population model)*

$$f(X) = RX(1 - X/K)$$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$

start: $X = X_0$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$

start: $X = X_0$

iterate: $X_1 = f(X_0)$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$

start: $X = X_0$

iterate:
$$X_1 = f(X_0)$$
$$X_2 = f(X_1) \quad = f(f(X_0)) \qquad = f^2(X_0)$$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$

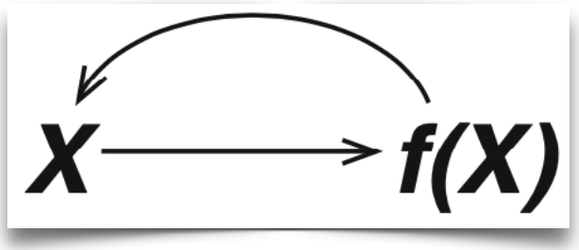
start: $X = X_0$

iterate:
$$\begin{aligned} X_1 &= f(X_0) \\ X_2 &= f(X_1) &&= f(f(X_0)) &&= f^2(X_0) \\ &\vdots &&\vdots &&\vdots \\ X_n &= f(X_{n-1}) &&= f(f(\ldots f(X_0))) &&= f^n(X_0) \end{aligned}$$

Simulation & Representation

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

system: $X_{n+1} = f(X_n)$



start: $X = X_0$

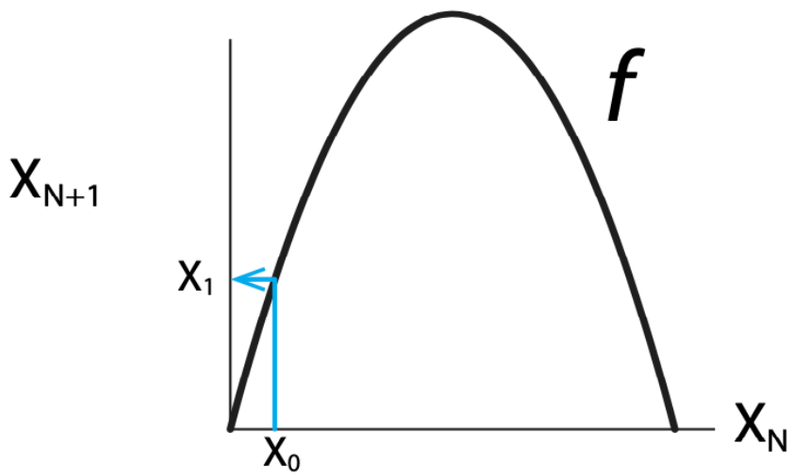
iterate:
$$\begin{aligned} X_1 &= f(X_0) \\ X_2 &= f(X_1) &&= f(f(X_0)) &&= f^2(X_0) \\ &\vdots &&\vdots &&\vdots \\ X_n &= f(X_{n-1}) &&= f(f(\ldots f(X_0))) &&= f^n(X_0) \end{aligned}$$

Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

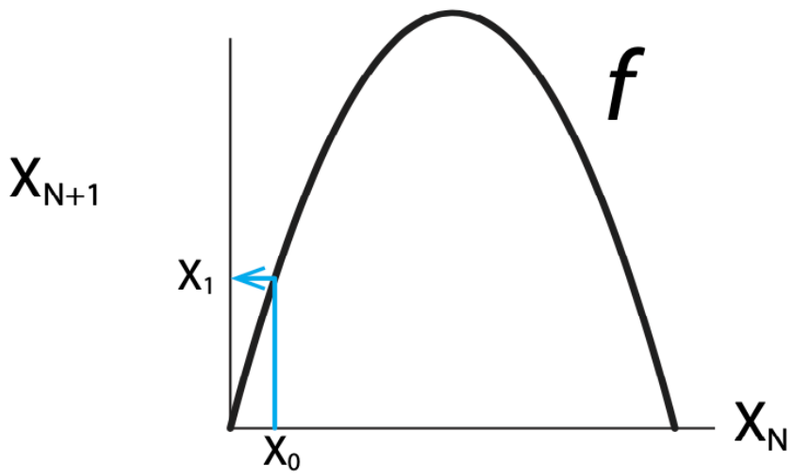
Simulation & Representation (…)

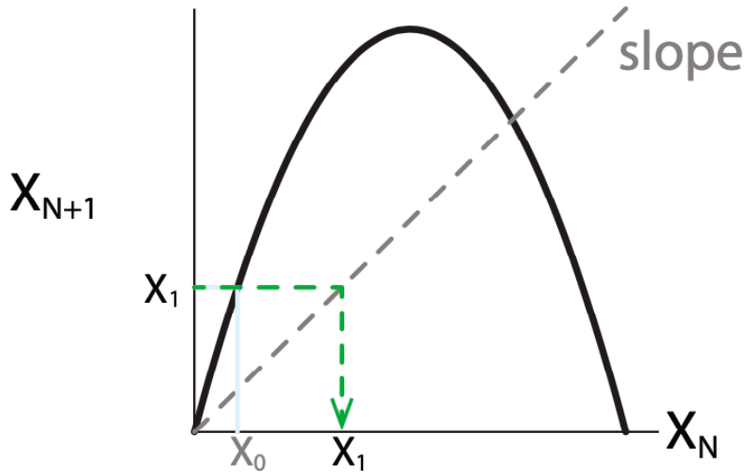
- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook



Simulation & Representation (…)

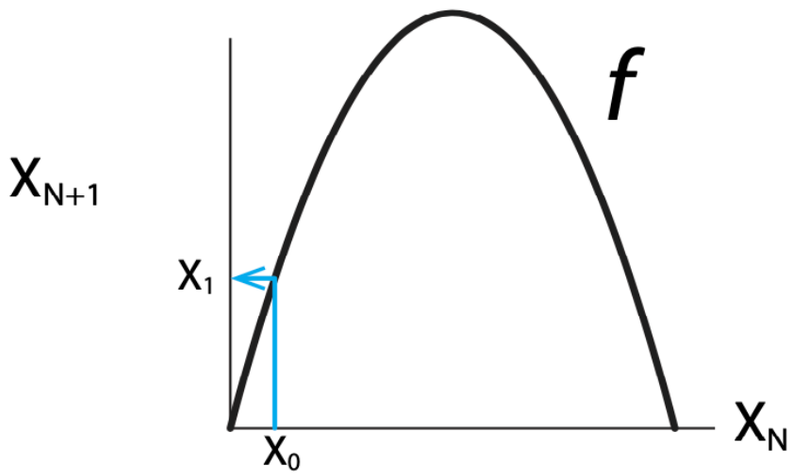
- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

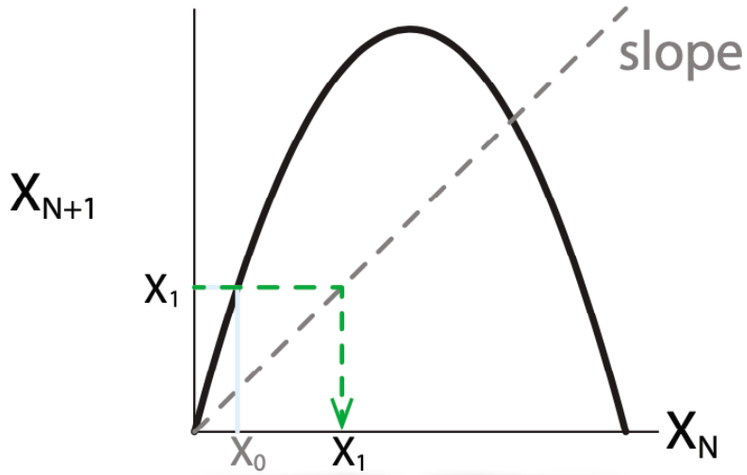




Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

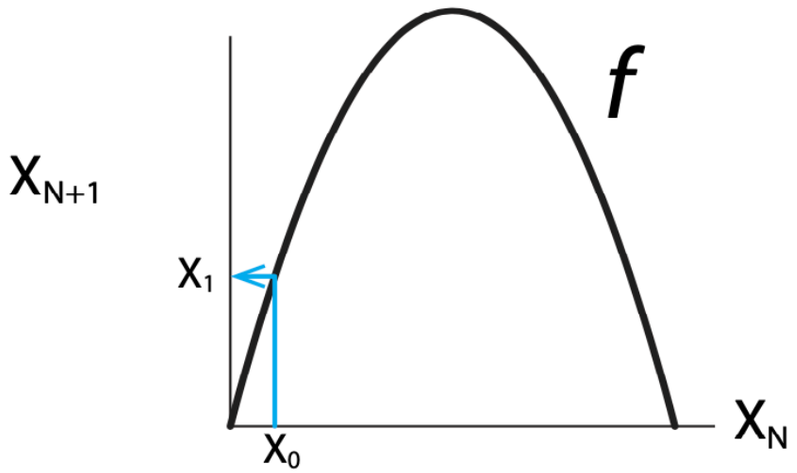


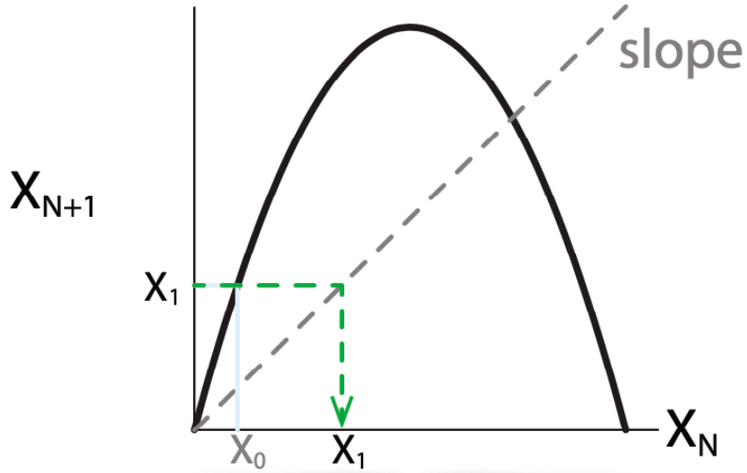


$$X_1 = f(X_0)$$

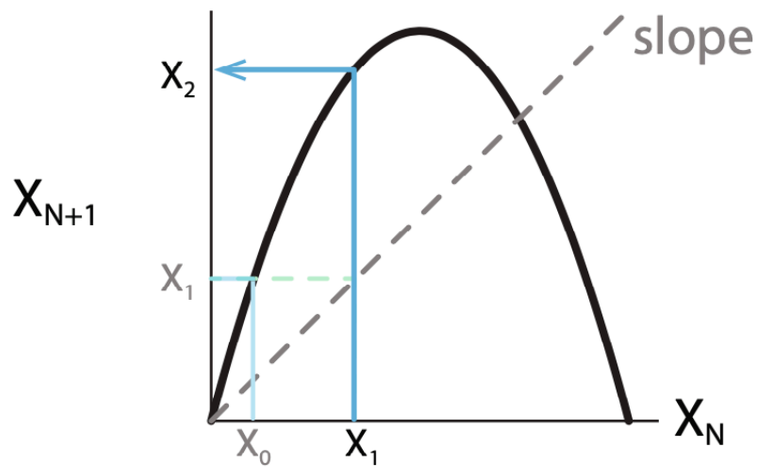
Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook



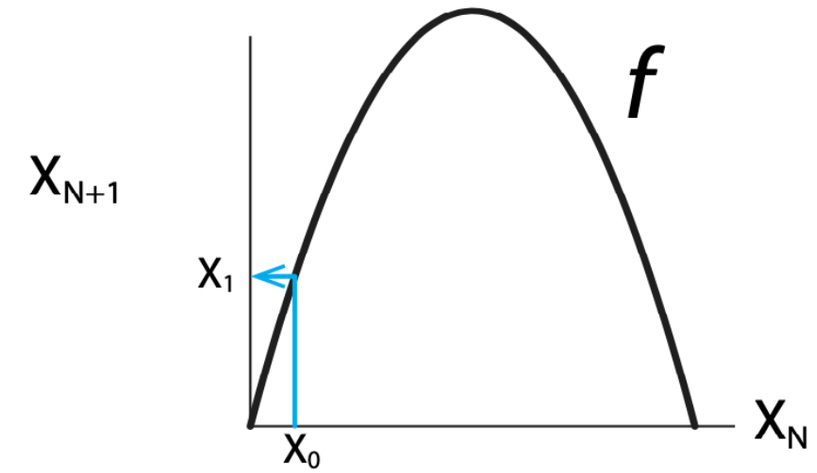


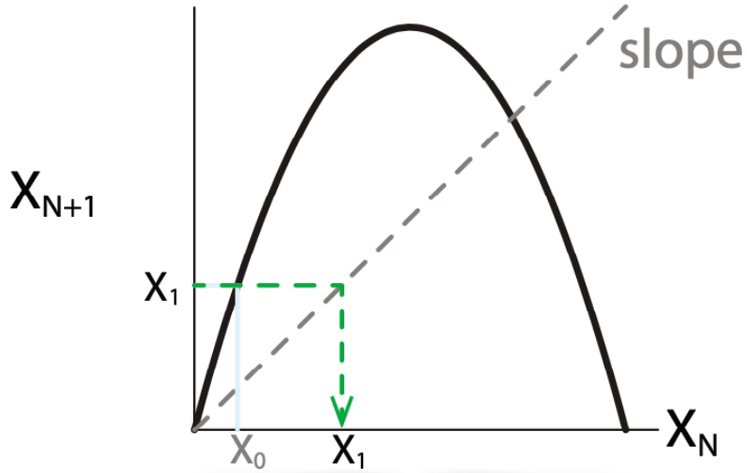
$$X_1 = f(X_0)$$



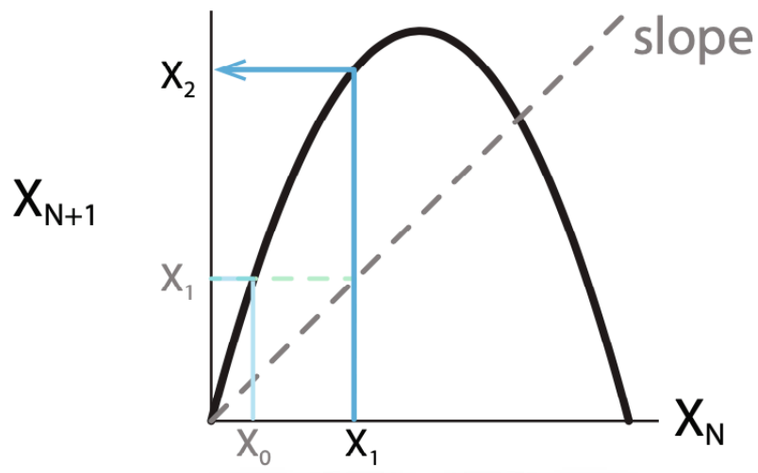
Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook





$X_1 = f(X_0)$



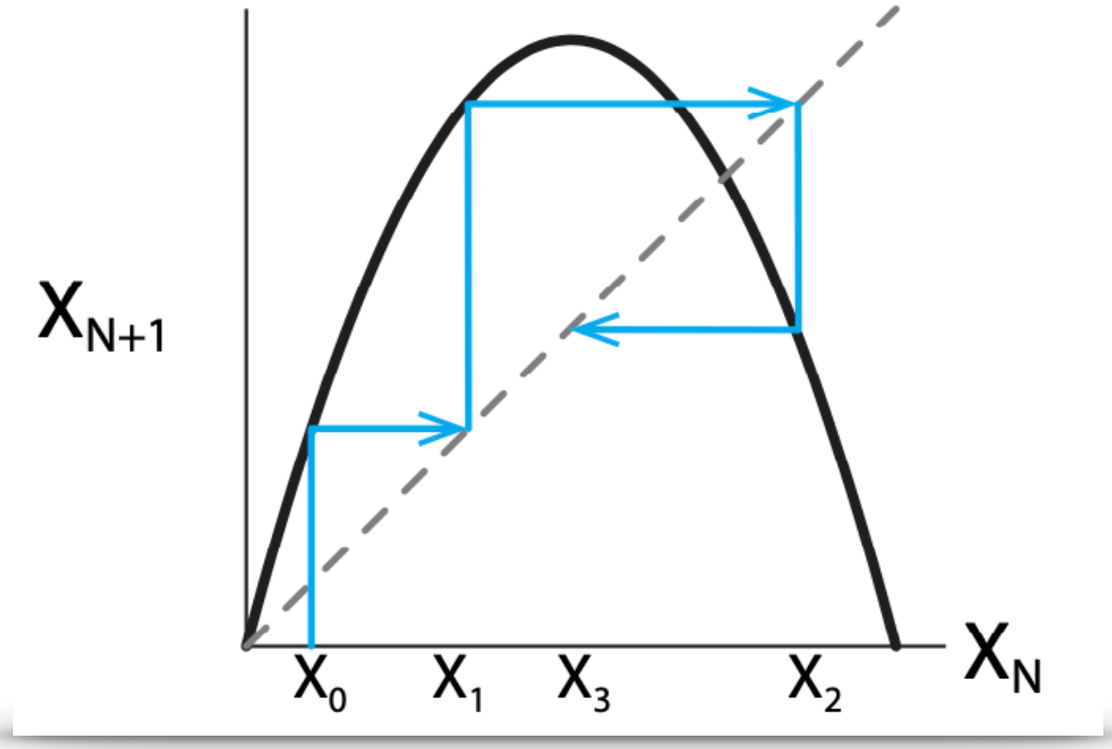
$X_2 = f(X_1)$

Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

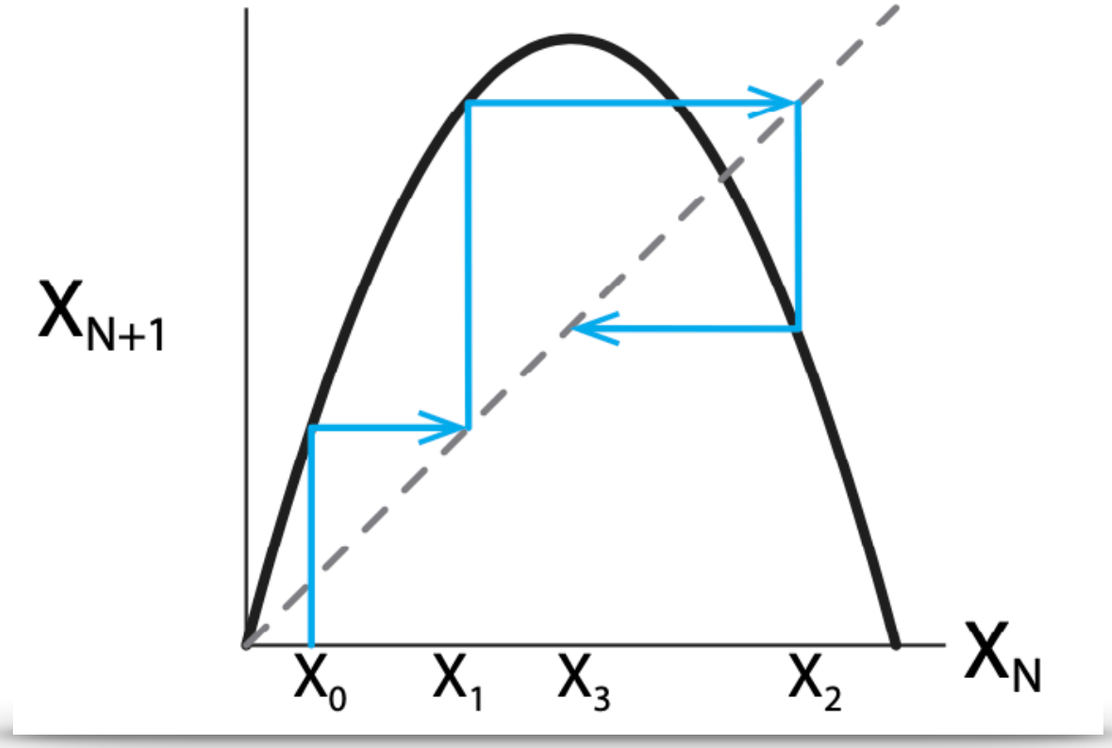


Simulation & Representation (…)

- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

Cobwebbing: reflecting alternately between the graph of *f* and the reference line $X_{N+1} = X_N$

Geometric realisation of the process of **iterating *f***

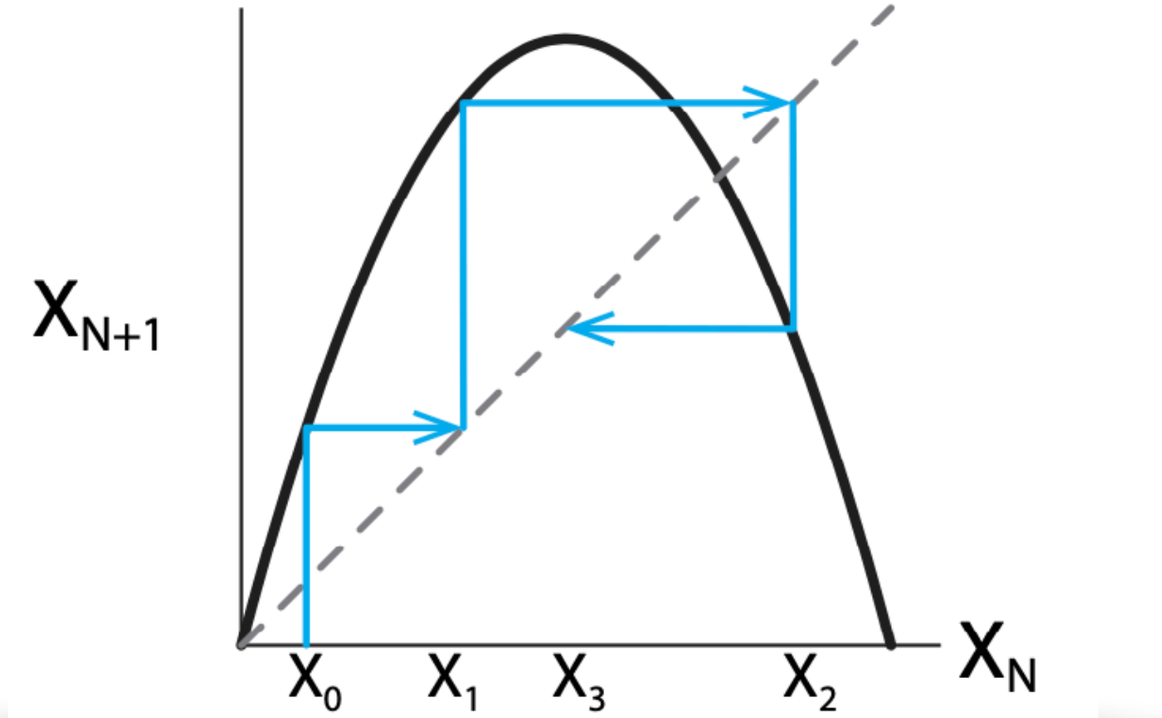


Simulation & Representation (…)

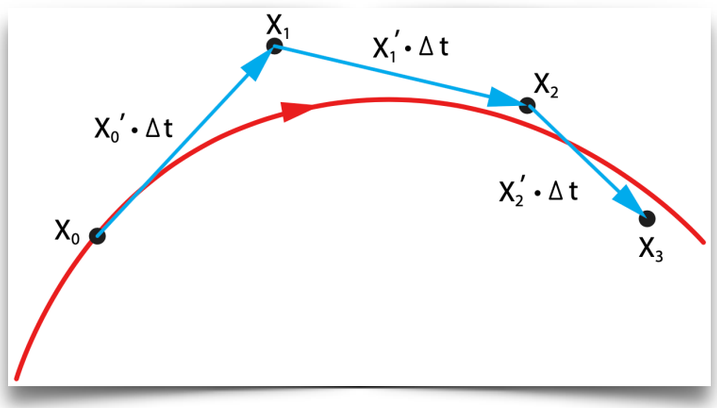
- **Cobweb plots:** a powerful tool to visualise the iteration of a function and find fixed points
  - *examples*: Geogebra Cobweb Demo ; `python` notebook

Cobwebbing: reflecting alternately between the graph of *f* and the reference line $X_{N+1} = X_N$

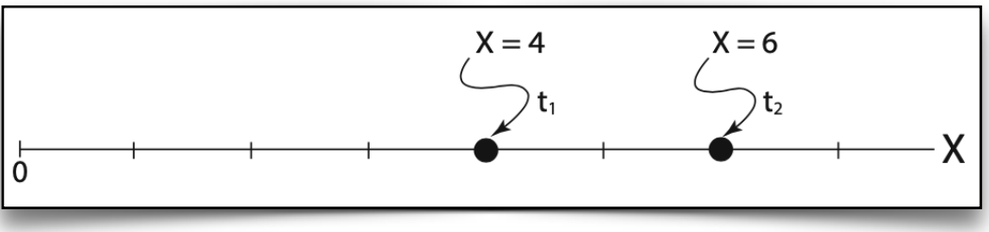
Geometric realisation of the process of **iterating *f***

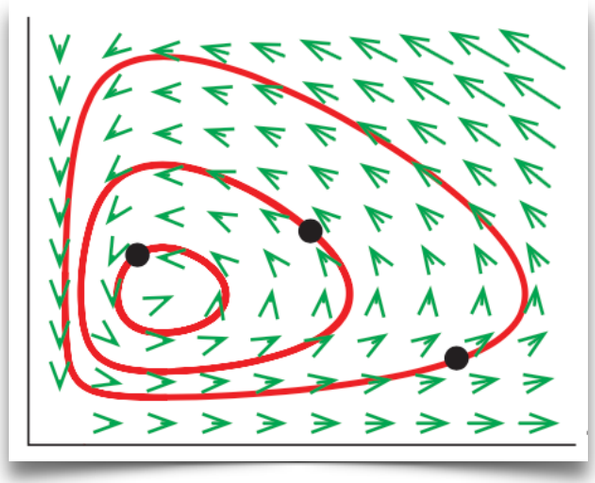


- **Discretisation**
  - discrete systems can also arise as approximation (**Euler scheme**) of continuous-time systems `[w2]`

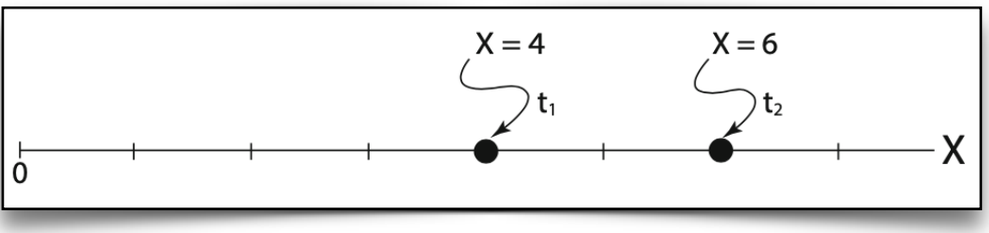


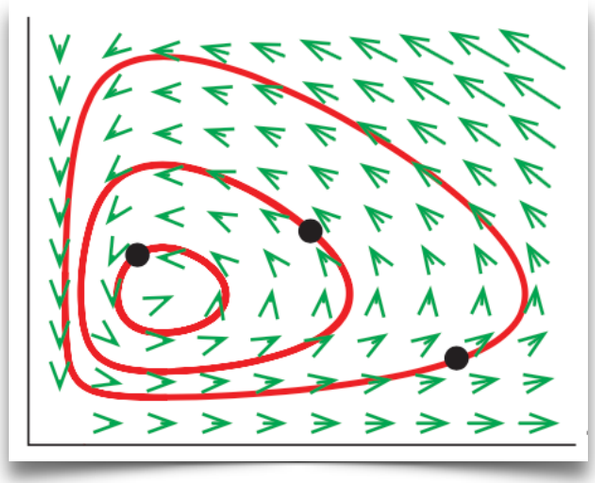
Graphical Recap

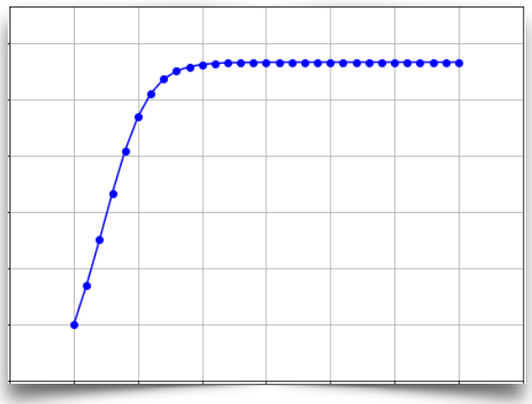


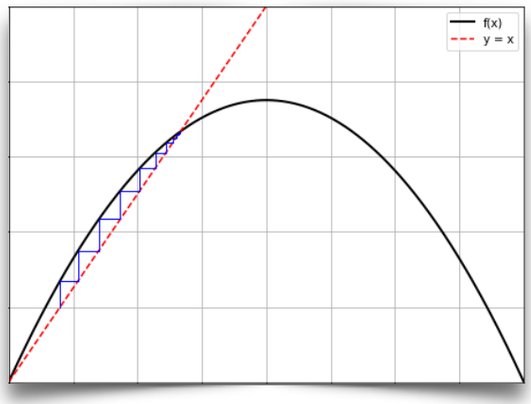


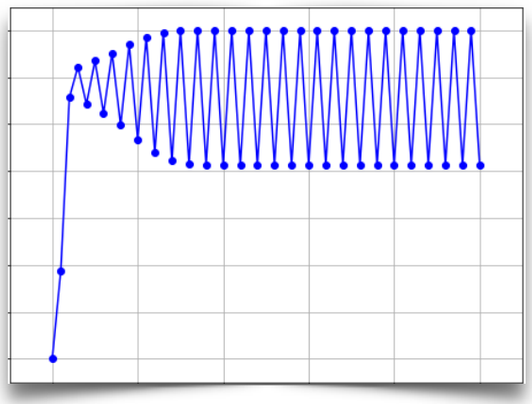
Graphical Recap

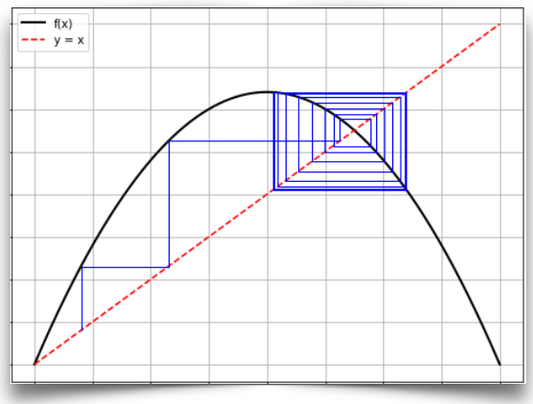




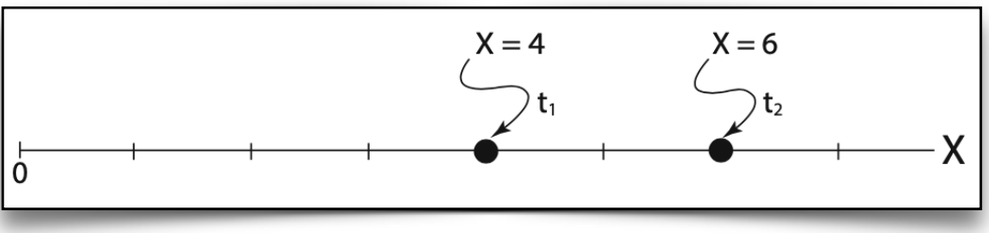


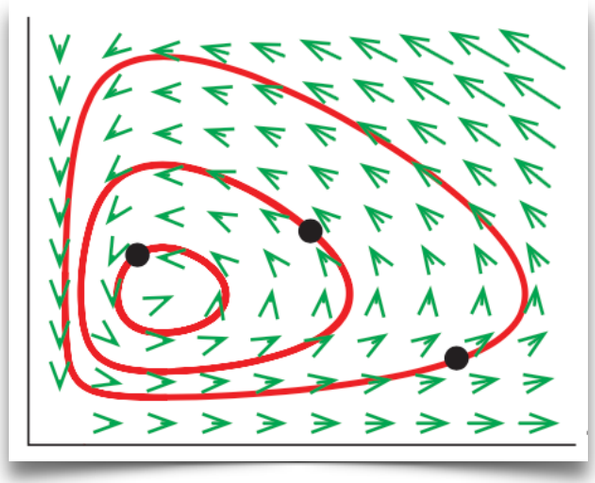


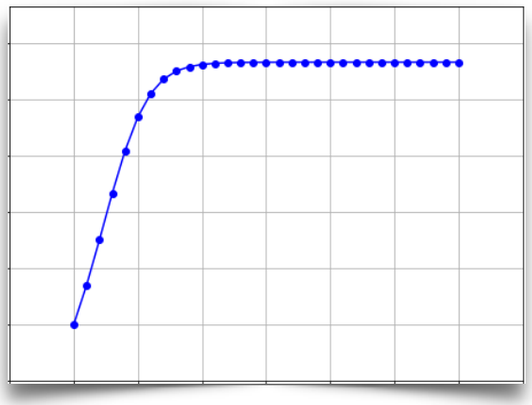


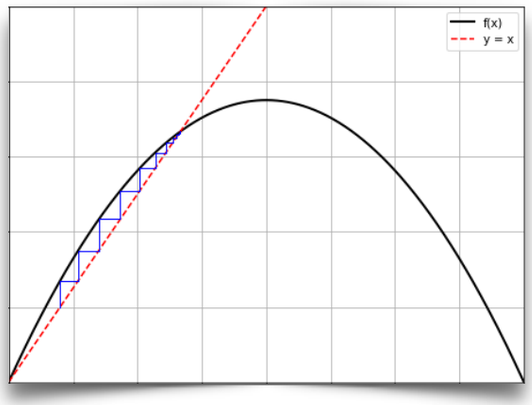


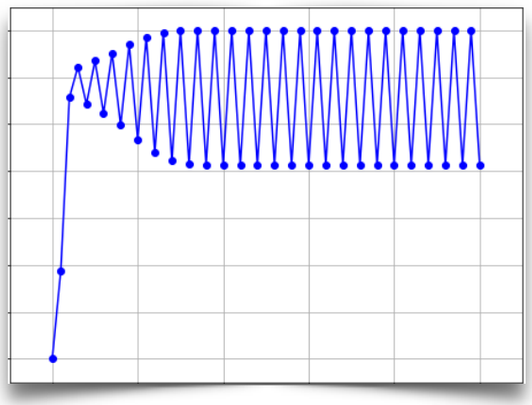
Graphical Recap

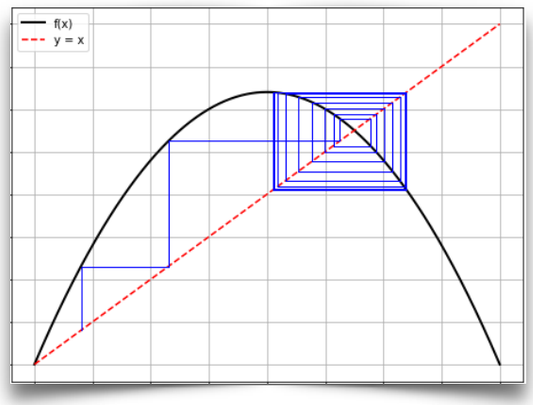


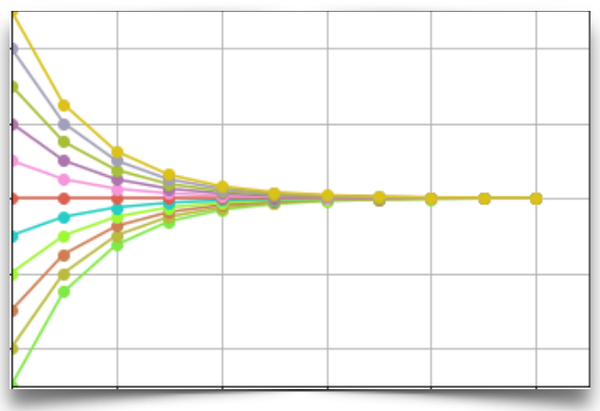


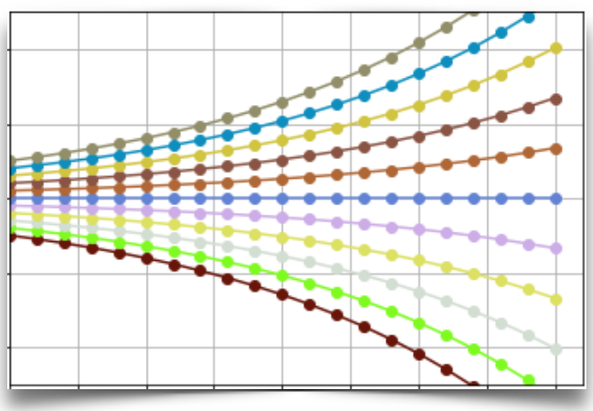












Homework for **next** week

d a y l i g h t

On to [week 2]

Homework #1: Daylight along the year across the globe [week 1 → week 2]

Four `.csv` files are available on Moodle, namely: `quito_sun_2023.csv`, `rome_sun_2023.csv`, `copenhagen_sun_2023.csv` and `tromso_sun_2023.csv`. They give all relevant pieces of information about daylight in these 4 cities for the year 2023.

1. Describe the content of these `.csv` files, that is, what each column represents.
2. Load **the first 3** of these files (corresponding to Quito, Rome and Copenhagen) onto a `python` notebook, for instance, using the package `pandas` (loaded at `pd`) and the associated command `pd.read_csv('city_sun_2023.csv')`. Now plot the column `daylight_h` for each of the three cities.
   - **2.a** Justify why you are effectively plotting the daylight in each city against time.
   - **2.b** What do you observe in terms of *dynamics*? In other words, how to characterise the evolution of daylight along the year in Quito, Rome and Copenhagen?
3. Use an example of discrete dynamical system seen in class in order to propose an elementary model reproducing these datasets of daylight records in 3 cities for 2023. What can you say about the influence of latitude on the dynamics of real system and of the model?
4. Now load the fourth `.csv` file, corresponding to the city of Tromsø, in Norway, and plot its `daylight_h` column.
   - **4.a** What do you observe compared to the other three cities? How
   - **4.b** How do you interpret it from the viewpoint of the underlying physical phenomenon?
5. Is the model you proposed to answer question **3.** capable of reproducing the behaviour corresponding to the city of Tromsø? Can you modify the model so as to capture this behaviour?