Lecture 2: Dynamical Systems and Iterated Maps

Modelling Complex Systems (Spring 2026)

Jordi-Lluís Figueras

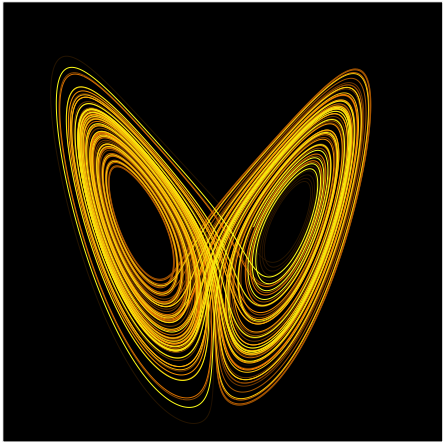
Matematiska Institutionen, Uppsala Universitet

March 26, 2026

Prelecture discussion

Have you seen the video from the website?

- Which models do you see in the video?
- Is a pendulum chaotic?



Source: Wikimedia Commons, `File:Lorenz attractor yb.svg` (Wikimol, Dschwen).

Lecture goals

- Introduce the language of dynamical systems.
- Distinguish continuous-time and discrete-time modeling.
- Study discrete dynamical systems through iterated maps.
- Explain how cobweb diagrams help visualize one-dimensional dynamics.
- Analyze linear maps and relate them to examples from complex systems.

What is a dynamical system?

A **dynamical system** consists of:

- a **state space** X,
- a notion of time,
- a rule that determines how the state evolves.

Typical questions are:

- What are the possible long-time behaviors?
- Which states are stable or unstable?
- How sensitive is the evolution to initial conditions?

Three quick examples

- **Pendulum:** the state contains position and velocity, and the rule is given by Newton's laws.
- **Population model:** the state is the population size, and the rule updates one generation to the next.
- **Opinion dynamics:** the state is a vector of opinions, and the rule averages with neighboring agents.

Discussion

For each example, try to identify the model:

- What is the **state**?
- What is the **time variable**?
- What is the **update rule** or law of evolution?

Why dynamical systems in complex systems?

Many complex systems are naturally modeled as evolving states:

- populations change from one generation to the next,
- opinions evolve after repeated interactions,
- epidemics spread through repeated contacts,
- traffic densities update over time,
- ecological and economic variables react to feedback.

Even when the rule is simple, repeated iteration can create rich collective behavior.

An intuitive viewpoint

In dynamical systems we want to answer the following question:

Core idea

If I know the state now, how do I predict the future?

In many complex systems:

- the state summarizes the relevant information,
- the rule encodes interactions and feedback,
- iteration reveals the cumulative effect of many small updates.

So a dynamical system is a mathematical way of turning *local update rules* into *global behavior over time*.

Continuous versus discrete time

Two common modeling choices are:

Continuous time

The state depends on $t \in \mathbb{R}$ and typically satisfies an ordinary differential equation,

$$\dot{x}(t) = F(x(t)).$$

Discrete time

The state is observed at times $n \in \mathbb{N}$ and satisfies an update rule,

$$x_{n+1} = f(x_n).$$

In this lecture we focus on the discrete case.

Examples of continuous and discrete modeling

Continuous time

- pendulum motion,
- temperature evolution,
- chemical concentrations.

Time flows continuously, so the model tracks the state at every instant.

Discrete time

- yearly population counts,
- daily epidemic updates,
- repeated opinion polls.

Time advances in steps, so the model updates from one observation time to the next.

Python demo: pendulum dynamics

Consider the pendulum equations

$$\dot{\theta} = \omega, \qquad \dot{\omega} = -\frac{g}{L}\sin(\theta) - b\omega.$$

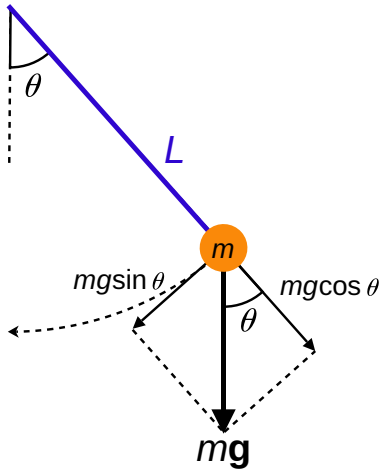
Here:

- θ is the angular displacement,
- ω is the angular velocity,
- g is the gravitational acceleration, L is the pendulum length, b is a damping coefficient.

Python script

The script `pendulum.py` shows:

- the phase portrait in (θ, ω),
- energy level curves,
- a numerical trajectory,
- the corresponding motion of the pendulum.



Source: Wikimedia Commons, `File:Simple pendulum generalized coordinates.svg` (Maschen).

Discrete dynamical systems

A **discrete dynamical system** is a map

$$f : X \to X,$$

together with the iteration rule

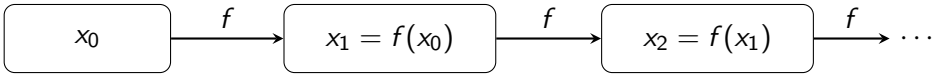
$$x_{n+1} = f(x_n), \qquad n = 0, 1, 2, \ldots$$

Starting from an initial condition x_0, we obtain the **orbit**

$$x_0,\ x_1 = f(x_0),\ x_2 = f(f(x_0)),\ \ldots$$

The n-th iterate is denoted by f^n, so $x_n = f^n(x_0)$.

How to think about a discrete system



Intuitively:

- we do not solve the whole future at once,
- we apply the same rule again and again,
- long-term behavior comes from accumulation of repeated updates.

A worked orbit

Consider the linear map

$$x_{n+1} = 0.8x_n + 1, \qquad x_0 = 2.$$

The first iterates are:

$$x_1 = 2.6, \qquad x_2 = 3.08, \qquad x_3 = 3.464.$$

Discussion

Can you describe how the forward orbit will look?

- Will the terms keep increasing?
- Will they approach a limiting value?
- Will they oscillate, or not?

Basic objects of study

For a discrete system $x_{n+1} = f(x_n)$ we study:

- **fixed points**: $f(x^*) = x^*$,
- **periodic points**: $f^p(x) = x$ for some $p \geq 2$,
- **invariant sets**: sets mapped into themselves by f,
- **stability**: whether nearby initial conditions remain close or converge,
- **basins of attraction**: initial conditions converging to the same attractor.

A collection of examples

Example	State	Update rule
Population model	population x_n	next generation depends on current size
Opinion dynamics	vector of opinions	repeated averaging with neighbors
Epidemic model	infected fraction x_n	next step balances contagion and recovery
Financial adjustment	price x_n	next price reacts to excess demand
Resource dynamics	stock x_n	growth and extraction at each period

The modeling question is always the same: choose a meaningful state and a meaningful update rule.

Iterated maps

An **iterated map** is simply the repeated application of the same function.

For a one-dimensional state variable,

$$x_{n+1} = f(x_n), \qquad x_n \in \mathbb{R}.$$

This framework is already rich enough to show:

- convergence to equilibria,
- oscillations,
- bistability,
- sensitive dependence on the parameter values,
- routes toward chaotic behavior.

Example: logistic-type population update

A classical nonlinear example is

$$x_{n+1} = r x_n (1 - x_n), \qquad 0 \le x_n \le 1.$$

Interpretation:

- x_n is a normalized population density,
- r is a reproduction parameter,
- the factor $(1 - x_n)$ models limited resources.

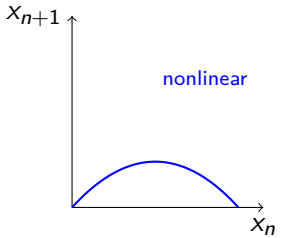
A simple rule can generate fixed points, oscillations, and more complicated behavior when the parameter changes.

Intuition behind the logistic map

$$x_{n+1} = r x_n (1 - x_n)$$

- If x_n is small, the population can grow.
- If x_n is close to 1, crowding suppresses growth.
- The update mixes *amplification* and *competition*.

This balance between positive and negative feedback is typical in complex systems.



Example: a coarse epidemic map

Consider the fraction $x_n \in [0,1]$ of infected individuals and a map of the form

$$x_{n+1} = x_n + \beta x_n(1 - x_n) - \gamma x_n.$$

Here:

- $\beta x_n(1 - x_n)$ represents new infections,
- γx_n represents recoveries,
- nonlinear feedback appears because contacts require both susceptible and infected individuals.

This is only a coarse discrete model, but it already illustrates threshold effects and equilibria.

Example: opinion averaging

If $x_n \in \mathbb{R}^d$ stores the opinions of d agents, a simple update is

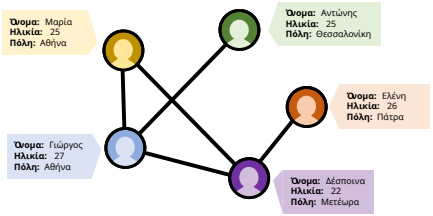
$$x_{n+1} = Ax_n,$$

where A is a matrix of interaction weights.

Questions:

- Do opinions converge to consensus?
- Do clusters persist?
- Which network structures accelerate convergence?

This is a linear example, but it is already relevant for social and networked systems.



Source: Wikimedia Commons, `File:Social network as graph.svg` (Dimitris131).

Cobweb diagrams for one-dimensional maps

For a map $x_{n+1} = f(x_n)$, a **cobweb diagram** compares:

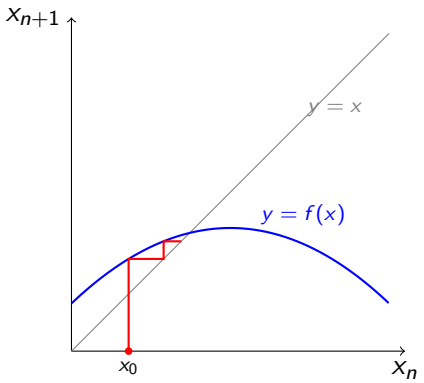
- the graph $y = f(x)$,
- the diagonal $y = x$.

Starting from x_0:

1. move vertically from $(x_0, 0)$ to the graph $y = f(x)$,
2. move horizontally to the diagonal $y = x$,
3. repeat.

This graphical construction encodes the successive iterates $x_1, x_2, x_3, \ldots$.

Cobweb sketch



The red path alternates between the graph and the diagonal, making the iteration visible step by step.

Python demo: logistic cobweb

For the logistic map

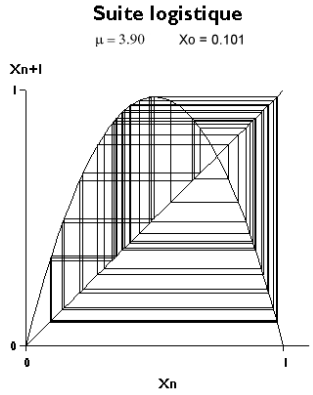
$$x_{n+1} = r x_n (1 - x_n),$$

the script `cobWeb.py` provides an animated cobweb diagram.

Python script

The script shows:

- the graph of $f(x) = rx(1-x)$,
- the diagonal $y = x$,
- the step-by-step cobweb construction,
- the forward orbit (x_n),
- the effect of changing the parameter $r \in [0, 4]$.



Source: Wikimedia Commons, `File:Suite logistique 390 101.png` (Jct, public domain).

What does a cobweb diagram reveal?

A cobweb plot gives immediate qualitative information:

- convergence to a fixed point,
- alternating approach to a fixed point,
- repulsion away from a fixed point,
- possible trapping regions,
- evidence of periodic behavior.

Stability

Intuition behind stability

Near a fixed point x^*, the map behaves approximately like a linear map:

$$x_{n+1} - x^* \approx f'(x^*)(x_n - x^*).$$

So the derivative tells us whether errors shrink or grow:

- if $|f'(x^*)| < 1$, deviations are contracted,
- if $|f'(x^*)| > 1$, deviations are amplified,
- if $f'(x^*) < 0$, the orbit tends to alternate from one side to the other.

This is why linear maps are so important even for nonlinear systems.

Fixed points and local stability in one dimension

Let x^* satisfy $f(x^*) = x^*$.

A useful local criterion is:

- if $|f'(x^*)| < 1$, then x^* is locally attracting,
- if $|f'(x^*)| > 1$, then x^* is locally repelling,
- if $|f'(x^*)| = 1$, then the linear test is inconclusive.

This criterion explains the visual behavior seen in many cobweb diagrams.

Three stability examples

Consider three maps with fixed point $x^* = 0$:

$$f_1(x) = 0.5x, \qquad f_2(x) = 1.2x, \qquad f_3(x) = -0.6x.$$

- For f_1, we have $|f_1'(0)| = 0.5 < 1$: trajectories are attracted monotonically to 0.
- For f_2, we have $|f_2'(0)| = 1.2 > 1$: trajectories move away from 0.
- For f_3, we have $|f_3'(0)| = 0.6 < 1$ and $f_3'(0) < 0$: trajectories are attracted to 0 while alternating sign.

One-dimensional linear maps

Consider

$$x_{n+1} = ax_n + b.$$

If $a \neq 1$, the fixed point is

$$x^* = \frac{b}{1-a}.$$

Writing $u_n = x_n - x^*$ gives

$$u_{n+1} = au_n,$$

so

$$u_n = a^n u_0.$$

Therefore the behavior is governed completely by the parameter a.

Classification of the linear map $x_{n+1} = ax_n + b$

- If $|a| < 1$, every orbit converges to the fixed point.
- If $0 < a < 1$, convergence is monotone.
- If $-1 < a < 0$, convergence is alternating.
- If $|a| > 1$, the fixed point is unstable.
- If $a = 1$, then $x_{n+1} = x_n + b$, so there is drift unless $b = 0$.
- If $a = -1$, then $x_{n+1} = -x_n + b$, which typically gives a period-two oscillation around the fixed point when it exists.

Examples of linear dynamics in complex systems

- **Consensus model:**

$$x_{n+1} = Ax_n$$

Here $x_n^{(i)}$ is the opinion of agent i at step n, and A_{ij} gives how much agent i weights agent j.

- **Age-structured population:**

$$x_{n+1} = Lx_n$$

Here $x_n^{(i)}$ is the number of individuals in age class i at year n, and L is a Leslie matrix containing fertility and survival rates.

Examples of linear dynamics in complex systems

- **Markov chain:**

$$p_{n+1} = P p_n$$

Here $p_n^{(i)}$ is the probability of being in state i at step n, and P_{ij} is the transition probability from state j to state i.

- **Linearization near equilibrium:**

$$u_{n+1} = A u_n$$

Here $u_n = x_n - x^*$ measures the deviation from a fixed point x^*, and A is the derivative matrix of the nonlinear model at x^*.

A visual summary of one-dimensional linear dynamics

Case $0 < a < 1$

- each step moves toward equilibrium,
- no sign change,
- monotone relaxation.

Case $-1 < a < 0$

- magnitude decreases,
- sign alternates,
- oscillatory relaxation.

$$u_{n+1} = au_n$$

$$u_{n+1} = au_n$$

In both cases $|a| < 1$ means memory of the initial condition is gradually lost.

Higher-dimensional linear maps

For $x_n \in \mathbb{R}^d$,

$$x_{n+1} = Ax_n + b.$$

After shifting by a fixed point, the dynamics becomes

$$u_{n+1} = Au_n.$$

Main principle:

- the eigenvalues of A control growth, decay, and oscillation,
- if all eigenvalues satisfy $|\lambda| < 1$, the fixed point is linearly attracting,
- eigenvalues with modulus larger than 1 create unstable directions.

Discussion: eigenvalues and eigenvectors

Discussion

For the map

$$u_{n+1} = Au_n,$$

what do you think eigenvalues and eigenvectors tell us about the dynamics?

Possible guiding questions:

- What happens if the initial condition points in an eigenvector direction?
- How does the modulus of an eigenvalue affect growth or decay?
- What kind of behavior might a negative or complex eigenvalue produce?
- Which directions should we expect to dominate after many iterations?

Take-away message

- Discrete dynamical systems are defined by repeated application of an update rule.
- Iterated maps provide a basic language for many complex systems.
- Cobweb diagrams give a geometric view of one-dimensional dynamics.
- Linear maps are completely analyzable and provide the local model near equilibria.
- Complex collective behavior often starts from simple repeated interactions.

Suggested computational experiments

1. Iterate $x_{n+1} = ax_n + b$ for different values of a.
2. Build a cobweb diagram for a nonlinear map such as the logistic map.
3. Compare two nearby initial conditions and track their separation.
4. Explore how changing a parameter modifies the qualitative long-time behavior.

Acknowledgment

These slides were prepared with the assistance of **OpenAI Codex / ChatGPT 5.4** as a drafting and editing tool.

All mathematical, pedagogical, and course-specific content remains under the responsibility of the lecturer.