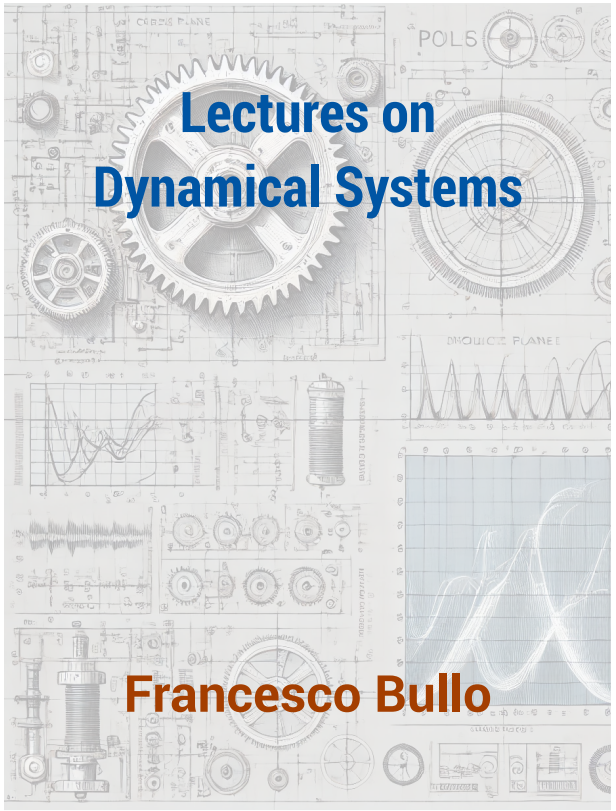
UC Santa Barbara, Department of Mechanical Engineering

ME103 Dynamical Systems. Chapter Slides.

Francesco Bullo
http://motion.me.ucsb.edu/ME103-Fall2025/syllabus.html



Contents

Chapter 1

Dynamical Systems: Definition and Scientific Examples

This chapter explores the foundational concepts and applications of dynamical systems, a mathematical framework essential for modeling time-dependent phenomena across various scientific and engineering disciplines. Dynamical systems are characterized by a set of rules or equations that describe how the state of a system evolves over time. The state is defined by a collection of variables, and the system's dimension is determined by the number of these variables. This chapter provides a comprehensive overview of dynamical systems, starting with basic definitions and examples, such as planetary motion and mechanical systems like the mass-spring-damper system, and extending to complex models in ecology and synthetic biology.

We begin with *one-dimensional models*, focusing on population dynamics through exponential and logistic growth equations. These models illustrate how populations change over time, considering factors like growth rate and carrying capacity. The chapter then progresses to *two-dimensional systems*, exemplified by the Lotka-Volterra predator-prey model, which requires numerical methods for analysis due to its complexity. *Phase portraits* are introduced as a tool for visualizing the trajectories of dynamical systems in both one and two dimensions, highlighting key features like *equilibrium points* and *stability*.

In summary, this chapter introduces the following concepts:

Systems

- first and second-order dynamical systems,
- linear and nonlinear systems,
- the distinction between a state variable and a parameter.

Phenomena

- an equilibrium point as a constant-state solution,
- trajectories that may diverge, converge to an equilibrium, or be periodic, and
- other complex phenomena such as chaos.

Analysis methods

- mathematical analysis to find closed-form solutions (when possible),
- numerical analysis using tools like Python to simulate and visualize solutions.

1.1 Definition and examples of dynamical systems

Dynamical systems are widespread in science and engineering and model a wide range of phenomena. These systems can represent planetary motion, the oscillation of a pendulum, the growth of a bacterial population, or fluctuations in stock market prices.

Figure 1.1: A long-exposure photograph showing star trails. The movement of stars across the night sky has long been a subject of scientific inquiry. This photo is a static visualization of the Earth's periodic rotation. Image sourced from https://unsplash.com.

In simple terms, a dynamical system provides a rule-based framework for describing how quantities change over time.

Definition 1.1. *A dynamical system is a mathematical concept that describes how variables of interest changes over time.*
A dynamical system is a set of rules or equations that dictates how a system's state evolves from one moment to the next.

The *state* of a dynamical system is a set of variables that provides a complete description of the system at a specific moment. For example:

(i) for a mechanical pendulum, the state could be its angle and angular velocity;

(ii) for an electric circuit, the state could be the voltage across a capacitor and the current through an inductor;

(iii) in fluid dynamics, the state might include pressure, temperature, velocity, and density at every point in the fluid.

The number of variables required to define the state is the *dimension of the system*.

Knowledge of the state at a given time, combined with the system's governing equations, allows for the prediction of the system's future behavior. The models we study generally share the following properties:

(i) *Dynamical systems are typically described by ordinary differential equations (ODEs)* or their variations, such as partial differential equations for spatially distributed variables or difference equations for discrete-time systems.

(ii) Given an initial condition and any external input, *there exists a unique solution* (also known as the system's evolution or trajectory).

(iii) The behavior of the dynamical system is typically affected by the values of one or more *parameters*.

Examples in mechanical engineering

Dynamical systems in mechanical engineering often exhibit rich and complex behavior. The following examples are widely studied due to their practical importance and interesting dynamics:

- The classic *mass-spring-damper system* models a wide range of structures and electro-mechanical systems. Mass-spring-damper systems exhibit underdamped or overdamped responses.
- The *double pendulum* is a classic example of a system with chaotic behavior. Even with two simple links, its motion can become highly unpredictable and sensitive to initial conditions.
- *Vibrating beams and plates*, such as those found in structures or mechanical components, exhibit complex dynamics, especially when subject to large deflections or nonlinear boundary conditions.
- *Rotating systems* like turbines or helicopter blades involve gyroscopic effects and can exhibit nonlinearities, whirling, and even chaotic vibrations under certain operating conditions.
- *Fluid-structure interactions*, such as aeroelastic flutter in aircraft wings or vortex-induced vibrations in bridges and pipelines, show rich dynamics due to the coupling between fluid flow and structural response.
- *Multi-link robotic arms* with flexible joints and components can exhibit rich nonlinear dynamics, especially when considering real-world effects like joint friction, flexible elements, and time-varying forces.
- *MEMS devices*, which operate on a micro-scale, often show nonlinear dynamics due to electrostatic forces, material nonlinearities, and damping effects, leading to intricate vibrational behaviors.

1.2 One-dimensional examples: exponential and logistic growth

We begin with two simple, intuitive examples describing the *dynamics of populations in ecology*. Both models are one-dimensional, meaning the system state is described by a single scalar variable. Both systems are modeled using differential equations.

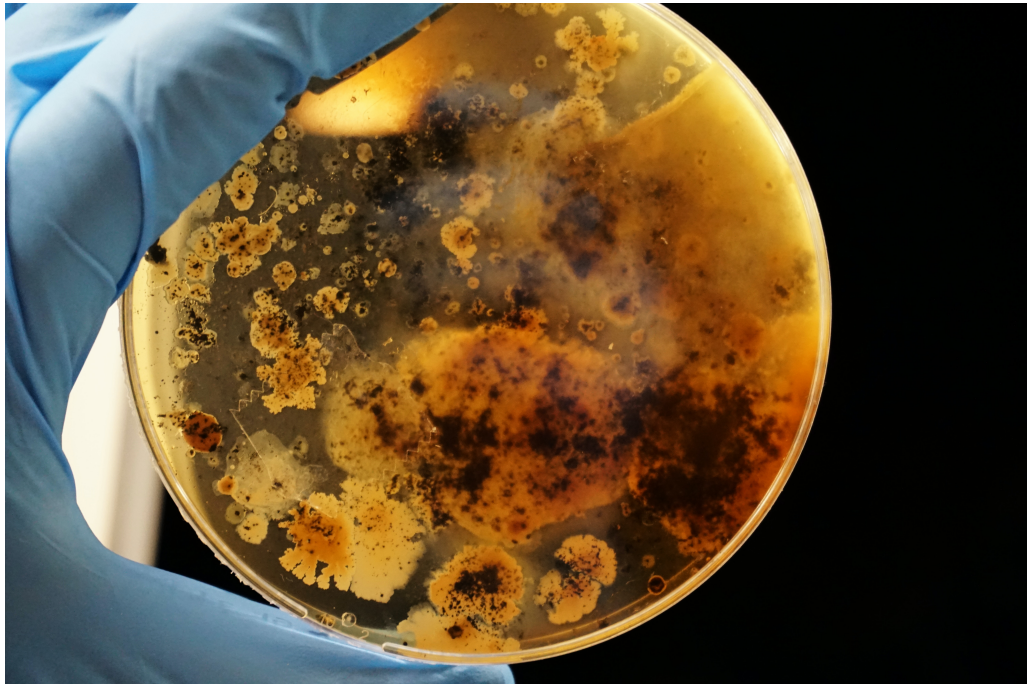


Figure 1.2: Bacterial colonies are shown growing on a plate, starting from just a few cells and expanding rapidly. The colonies grow at an exponential rate initially, but this growth slows as resources become limited. Image sourced from https://unsplash.com.

1.2.1 A first model: the linear growth/decay model

To develop a quantitative model, we make the following assumptions:

(i) we let t denote time, the independent variable;

(ii) we let $x(t)$ denote the *number of individuals*[1] in a population at time t;

(iii) we assume that the *birth rate* and *death rate* are proportional to the population size $x(t)$, that is,

$$\text{birth rate}(t) = r_{\text{birth}} x(t) \quad \text{and} \quad \text{death rate}(t) = r_{\text{death}} x(t),$$

where $r_{\text{birth}} > 0$ and $r_{\text{death}} > 0$ are the constant, positive, average *per-individual birth and death rates*, respectively;

(iv) we compute the net rate of change as $\text{birth rate}(t) - \text{death rate}(t) = (r_{\text{birth}} - r_{\text{death}})x(t)$ and define the *per-individual growth rate* as $r = r_{\text{birth}} - r_{\text{death}}$.

[1] Since the population size $x(t)$ is assumed to be sufficiently large, treating it as a continuous variable rather than an integer is an acceptable approximation.

In summary, the *linear growth/decay model* is

$$\dot{x}(t) = rx(t), \tag{1.1}$$

where the parameter r is *per-individual growth rate*.

- The model has one independent variable, t; one dependent variable, $x(t)$; and one parameter, r.
- The variable $x(t)$ is the *state of the system*, and r is a *parameter of the system*.
- Since the model involves a single independent variable and a first-order derivative of the dependent variable, it is a first-order ordinary differential equation.

Mathematical analysis of the linear growth/decay model

The model in equation (1.1) is *linear* because the right-hand side of the equation depends linearly on the state x. The solution to this linear model is:

$$x(t) = \mathrm{e}^{rt}\, x(0). \tag{1.2}$$

This model leads to three predictions about the long-term evolution of the population:

(i) if $r < 0$, the population will become extinct ($x(t) \to 0$ via exponential decay);

(ii) if $r = 0$, the population will remain constant at $x(0)$;

(iii) if $r > 0$, the population will experience exponential growth.

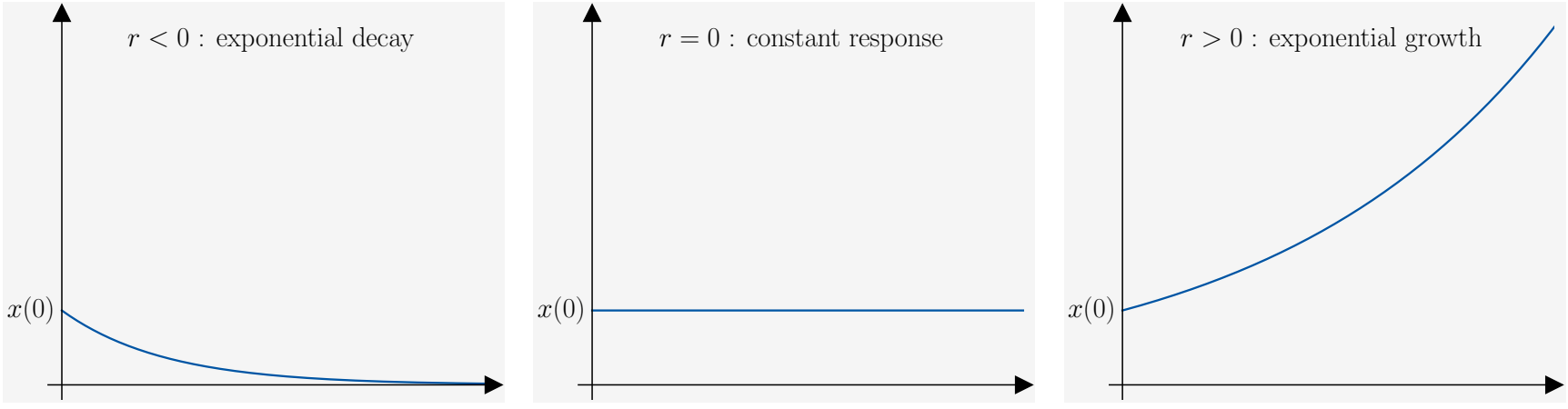


Figure 1.3: Solutions to the linear growth model in equation (1.2) for different values of the parameter r.

1.2.2 A second model: the logistic equation

Next, we consider a slightly more realistic population model. For $r > 0$, the exponential growth predicted by the linear model is unsustainable in an environment with finite resources. Therefore, a more accurate model should have a growth rate that depends on the population $x(t)$.

Specifically, we assume that the per-individual growth rate is a decreasing function of $x(t)$, reflecting that as the population grows, resources become scarcer. A simple way to model this *diminishing growth phenomenon* is to define the

$$\text{per-individual growth rate} = r\left(1 - \frac{x(t)}{K}\right), \tag{1.3}$$

where the parameter K is a positive constant that defines a *threshold behavior*:

- growth is positive for $x(t) < K$,
- growth is zero for $x(t) = K$, and
- growth turns negative, hence growth turns into decay, for $x(t) > K$.

The parameter K represents a critical population threshold that separates population increase from population decrease.

In summary, the *logistic growth model* is

$$\dot{x}(t) \;=\; r\Big(1-\frac{x(t)}{K}\Big)x(t) \;\;=\;\; \underbrace{rx(t)}_{\text{linear growth}} \;+\; \underbrace{\Big(-\frac{r}{K}\Big)x_i^2(t)}_{\text{logistic correction}}. \tag{1.4}$$

The model has two parameters:

- r is the *intrinsic growth rate* of the population: the rate at which the population would grow without constraints, and
- K is the *carrying capacity* of the environment: the maximum population that the environment can sustain over the long term.

Note: the logistic growth model (1.4) is a *nonlinear dynamical system*, whereas the growth/decay model (1.1) is a *linear dynamical system*.

Numerical analysis of the logistic growth model

We now perform a numerical analysis of the logistic equation through simulation.

```python
# Python libraries
import numpy as np; import matplotlib.pyplot as plt; from ...
    scipy.integrate import solve_ivp
plt.rcParams.update({"text.usetex": True, "font.family": ...
    "serif", "font.serif": ["Computer Modern Roman"] })

# Logistic equation: solve_ivp expects f(t, y, *args)
def logistic_equation(t, x, r, K):
    return r * x * (1 - x / K)

# Parameters
r = 0.1   # growth rate
K = 100   # carrying capacity

# Initial conditions and time range of integration
initial_conditions = [1, 5, 10, 25, 50, 75, 125, 150, 175]
T = 125.0          # maximum time
dt = 0.1           # time step
times = np.linspace(0.0, T, int(T/dt) + 1)
colors = ['#752d00', '#a43e00', '#d35000', '#ff6100', ...
    '#ff8800', '#ffaf00', '#ffcc00']

# Numerically solve the ODE and plot solution
plt.figure(figsize=(6, 2.7))
for i, x0 in enumerate(initial_conditions):
    sol = solve_ivp( logistic_equation, t_span=(0, T),
        y0=[x0], args=(r, K), t_eval=times,
        rtol=1e-8, atol=1e-10, method="RK45")
    plt.plot(sol.t, sol.y[0], label=f'$x(0)= {x0}$', ...
        color=colors[i % len(colors)])

# Labels, legend, axes
plt.xlabel(r"time $t$"); plt.ylabel(r"population size $x(t)$")
plt.legend(loc='lower right', fontsize=6); plt.grid(True);
plt.xlim(0, 120); plt.ylim(0, 180)

# Save to PDF
plt.savefig("logistic-equation.pdf", bbox_inches="tight")
```

Listing 1.1: Python script generating Figure 1.4. Available at: logistic-equation.py

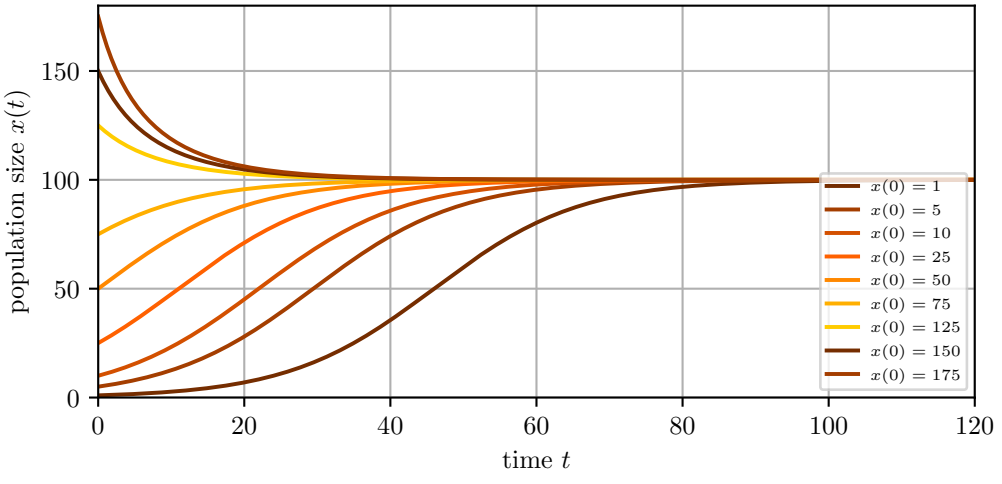


Figure 1.4: Numerical solutions from multiple initial conditions $x(0)$ to the logistic equation (1.4): $\dot{x} = r(1 - x/K)x$, for $r = 0.1$ and $K = 100$. Note:
(i) For each $x(0) > 0$, the solution $x(t)$ approaches the carrying capacity K as $t \to \infty$.
(ii) Each solution with $0 < x(0) < K$ is monotonically increasing, whereas each solution with $x(0) > K$ is monotonically decreasing.
(iii) For $x(0) = 0$, the solution is the equilibrium solution $x(t) = 0$. Negative initial conditions are not considered for population models.

In class assignment

What are the solutions for the initial conditions $x(0) = 0$ and $x(0) = K$?
Under what conditions do solutions have an inflection point?

Mathematical analysis of the logistic equation

Beside the numerical analysis (based upon numerical integration and visualization), a mathematical analysis provides further insight into the logistic equation $\dot{x}(t) = r(1 - x(t)/K)x(t)$.

Definition of an equilibrium point: For a dynamical system $\dot{x} = f(x)$:

(i) a point x^* is an *equilibrium point* if it is a solution to the *equilibrium equation* $f(x^*) = 0$,

(ii) if x^* is an equilibrium point, the constant function $x(t) = x^*$ is a solution to the dynamical system for all time,

(iii) an equilibrium point x^* is *stable* if trajectories starting near x^* converge to x^* as $t \to \infty$;

(iv) an equilibrium point x^* is *unstable* if trajectories starting near x^* move away from x^*.

Equilibrium points of the logistic equation: The logistic equation (1.4) has two equilibrium points, which are the solutions to $r(1 - x/K)x = 0$:

$$x_1^* = 0 \quad \text{and} \quad x_2^* = K. \tag{1.5}$$

The equilibrium $x_1^* = 0$ represents population extinction, while $x_2^* = K$ represents the population at the carrying capacity. Based on the numerical results in Figure 1.4, we can infer the stability of these points:

- $x_1^* = 0$ is an unstable equilibrium point.
- $x_2^* = K$ is a stable equilibrium point.

Remark 1.2. *As demonstrated in Exercise E1.1, the exact solution to the logistic equation is*

$$x(t) = \frac{K x_0 \, \mathrm{e}^{rt}}{K + x_0(\mathrm{e}^{rt} - 1)}, \tag{1.6}$$

where x_0 represents the population at initial time $t = 0$, that is, $x(0) = x_0$.

- *Equation (1.6) provides a closed-form solution for the logistic growth model (1.4).*
- *In general, finding a closed-form solution for a nonlinear dynamical system is not possible. Consequently, analysis typically relies on numerical simulations or qualitative techniques.*
- *This text will primarily focus on linear dynamical systems, which are analytically solvable (i.e., solutions can be derived analytically) and form the foundation of control theory.* •

The structure of a numerical simulation program

A typical Python script for simulating and visualizing an ODE performs the following steps:

(i) loads necessary software libraries (e.g., NumPy, SciPy, Matplotlib);

(ii) defines the function for the differential equation ($\dot{x} = f(x, t)$);

(iii) defines numerical values for all model parameters;

(iv) defines a set of initial conditions and a time integration range;

(v) computes numerical solutions to the ODE, for instance, using the `solve_ivp` routine from SciPy;

(vi) plots the state variables of the solution as a function of time;

(vii) saves the plot to a file.

It is important to remember that routines like `solve_ivp` compute a numerical *approximation* of the true solution. For high-precision applications, consult the documentation for solver options and error tolerances.

Programming notes

A brief comparison of Python and Matlab:

- Python is a general-purpose, open-source programming language. In contrast, Matlab is a proprietary high-level language and interactive environment designed specifically for numerical computation.
- Because Python is open-source, it is freely available on any operating system. The open-source community provides extensive resources, tutorials, and documentation.
- While Python's general-purpose nature may present a steeper learning curve than Matlab's specialized environment, modern large language models can generate initial script prototypes that users can then refine.
- Numerous powerful libraries are freely available for Python. For example, Matplotlib is a library for creating static, animated, and interactive visualizations. Other notable libraries include Pandas for data science, TensorFlow and PyTorch for machine learning, and the Robot Operating System (ROS) for robotics.
- These notes use the `streamplot` function in Python's Matplotlib library to draw phase portraits. Similar results can be achieved with the `streamline` function in Matlab. A streamplot provides a qualitative visualization of a vector field but does not guarantee that the plotted lines represent exact system trajectories.

1.3 Two-dimensional examples: periodic solutions

Figure 1.5: The Canadian lynx (*Lynx canadensis*) is a primary predator of the snowshoe hare (*Lepus americanus*). Historical records of animal pelts show that the populations of the two species oscillate periodically; see (Odum, 1959). Public domain image.

In this section, we examine the classic Lotka-Volterra predator-prey model. We let $x_1(t)$ denote the prey population and $x_2(t)$ denote the predator population. The *Lotka-Volterra predator-prey model* in the variables x_1 and x_2 is:

$$\begin{aligned} \dot{x}_1 &= \alpha x_1 - \beta x_1 x_2, \\ \dot{x}_2 &= -\delta x_2 + \gamma x_1 x_2, \end{aligned} \tag{1.7}$$

with four positive parameters:

- $\alpha > 0$ is the intrinsic growth rate of the prey,
- $\delta > 0$ is the intrinsic decay rate of the predator,
- $\beta > 0$ and $\gamma > 0$ characterize the inter-species interactions.

For general parameter values, no simple closed form solution is available. We therefore proceed with a numerical analysis.

Numerical analysis of the Lotka-Volterra predator-prey model

```python
# Python libraries
import numpy as np; import matplotlib.pyplot as plt;
from scipy.integrate import solve_ivp
plt.rcParams.update({"text.usetex": True, "font.family": "serif", ...
    "font.serif": ["Computer Modern Roman"], "font.size": 16})

# Define the Lotka-Volterra equations
alpha = 0.1; beta = 0.02; gamma = 0.01; Δ = 0.1
def lotka_volterra(t, z, alpha, beta, gamma, Δ):
    x, y = z
    dxdt = alpha * x - beta * x * y
    dydt = -Δ * y + gamma * x * y
    return [dxdt, dydt]

# Solve the equations:
initial_condition = (20, 5)
T = 200; dt = 0.1
times = np.linspace(0.0, T, int(T/dt) + 1)
sol = solve_ivp(lotka_volterra,
    t_span=(times[0], times[-1]),
    y0=initial_condition,
    args=(alpha, beta, gamma, Δ),
    t_eval=times,
    rtol=1e-8, atol=1e-10, method="RK45")

# Plot the solution as a function of time
plt.figure(figsize=(10, 5));
plt.plot(sol.t, sol.y[0], label='prey')
plt.plot(sol.t, sol.y[1], label='predator')
plt.xlabel(r"time $t$"); plt.ylabel("population size"); plt.legend()
plt.grid(True); plt.xlim(0, 200); plt.ylim(0, 22.25)
plt.savefig("lotka-volterra-predator-prey.pdf", bbox_inches='tight')

# Phase portrait
u, v = np.meshgrid(np.linspace(0, 25, 20), np.linspace(0, 15, 20))
du, dv = lotka_volterra(0, [u, v], alpha, beta, gamma, Δ)
magnitude = 2*np.sqrt(du**2 + dv**2)

plt.figure(figsize=(10, 5)); plt.xlim(0, 25); plt.ylim(0, 15)
plt.streamplot(u, v, du, dv, density=.95, color='#0085ff', ...
    linewidth=magnitude, arrowsize=2)

# Add the specific solution trajectory
plt.plot(sol.y[0], sol.y[1], color='#752d00', label='Trajectory from ...
    (20,5)')
plt.scatter(10, 5, color='black', s=50, zorder=5); plt.grid(True)
plt.xlabel('prey population $x_1(t)$')
plt.ylabel('predator population $x_2(t)$');
plt.savefig("lotka-volterra-predator-prey-phase.pdf", bbox_inches='tight')
```

Listing 1.2: Python script generating Figure 1.6. Available at lotka-volterra-predator-prey.py

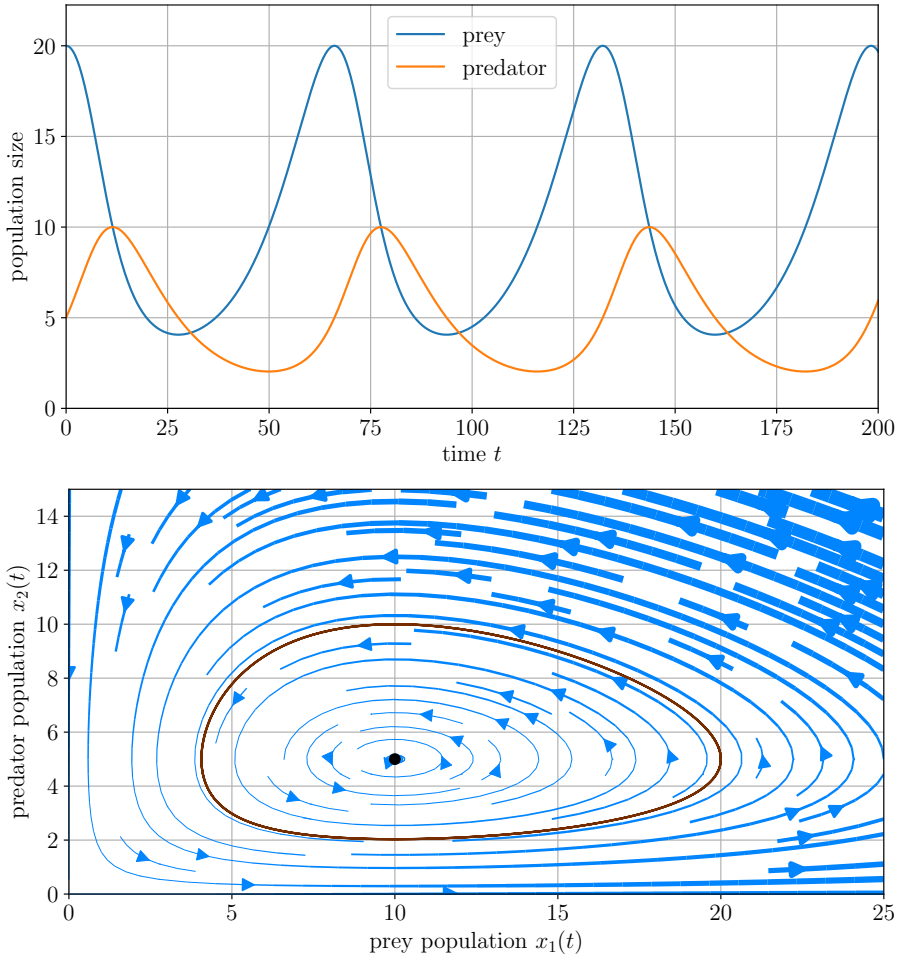


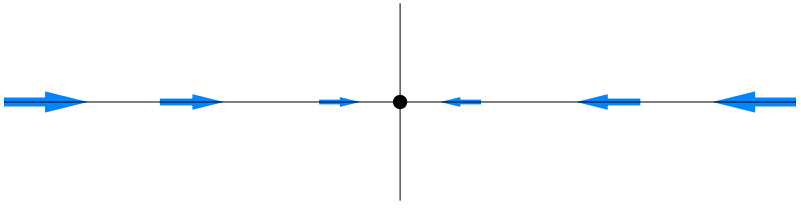
Figure 1.6: Solution and phase portrait for the Lotka-Volterra predator-prey dynamics (1.7).
The black line is the solution originating from initial condition $(20, 5)$.

1.4 Phase portraits

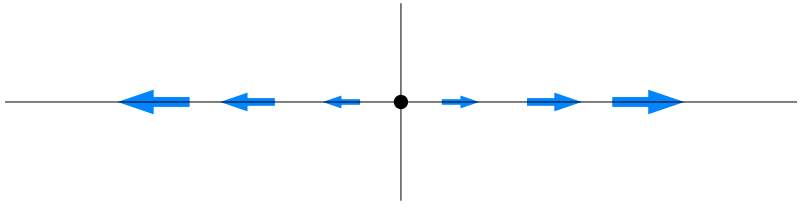
A one-dimensional ODE, $\dot{x} = f(x)$ with $x \in \mathbb{R}$,[2] defines a *vector field on a line*. This is also called a one-dimensional *phase portrait*.

- if $f(x) < 0$, then the solution starting at x moves to the left,
- if $f(x) = 0$, then the solution starting at x does not move, since the point x is an equilibrium, and
- if $f(x) > 0$, then the solution starting at x moves to the right.

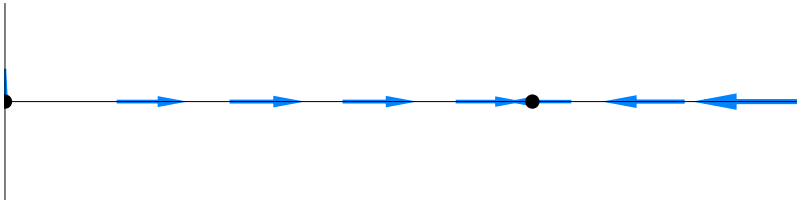
Figure 1.7 illustrates these vector fields for a few simple systems.



(a) Linear decay: $\dot{x} = -x$ (that is, $\dot{x} = rx$ with $r = -1$). Note that $x = 0$ is a stable equilibrium.



(b) Linear growth: $\dot{x} = x$ (that is, $\dot{x} = rx$ with $r = +1$). Note: $x = 0$ is an unstable equilibrium.



(c) Logistic equation: $\dot{x} = x - x^2$ (for $x > 0$, $r = 1$, $K = 1$.)

Figure 1.7: Phase portraits for simple one-dimensional ODEs.

[2] We use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{R}^d$ to denote the set of column vectors with d real entries.

Next, we consider a generic two-dimensional ODE:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2), \\ \dot{x}_2 &= f_2(x_1, x_2). \end{aligned} \tag{1.8}$$

A two-dimensional ODE defines a *vector field on the plane*. At each point (x_1, x_2), we can compute the velocity vector $(f_1(x_1, x_2), f_2(x_1, x_2$
A particle placed at (x_1, x_2) and moving according to this velocity vector traces a curve, which is a solution (or trajectory) of the ODE.

For clarity, remember these key ideas:

- An ODE is synonymous with a vector field.
- The plane is filled with trajectories, as a unique solution passes through each point.
- Trajectories do not intersect due to the uniqueness of solutions for the ODEs we consider.

The *phase portrait* (also known as the *trajectory diagram*) is a graphical representation of the trajectories of a two-dimensional dynamical system. It illustrates the paths that a particle obeying the ODE would follow. We illustrate how to draw the phase portrait in Figure 1.8.

A phase portrait allows for the recognition of a system's qualitative features and special solutions, such as:

(i) an equilibrium point (which may be stable, unstable, or marginally stable);

(ii) a closed orbit, which corresponds to a periodic trajectory;

(iii) a trajectory that diverges to infinity.

Generalizing from the plane, the *state space* is the multidimensional space where each axis corresponds to a state variable. Visualizing phase portraits in dimensions greater than two is challenging.

Example phase portraits

We provide some examples of phase portraits in Figure 1.9.

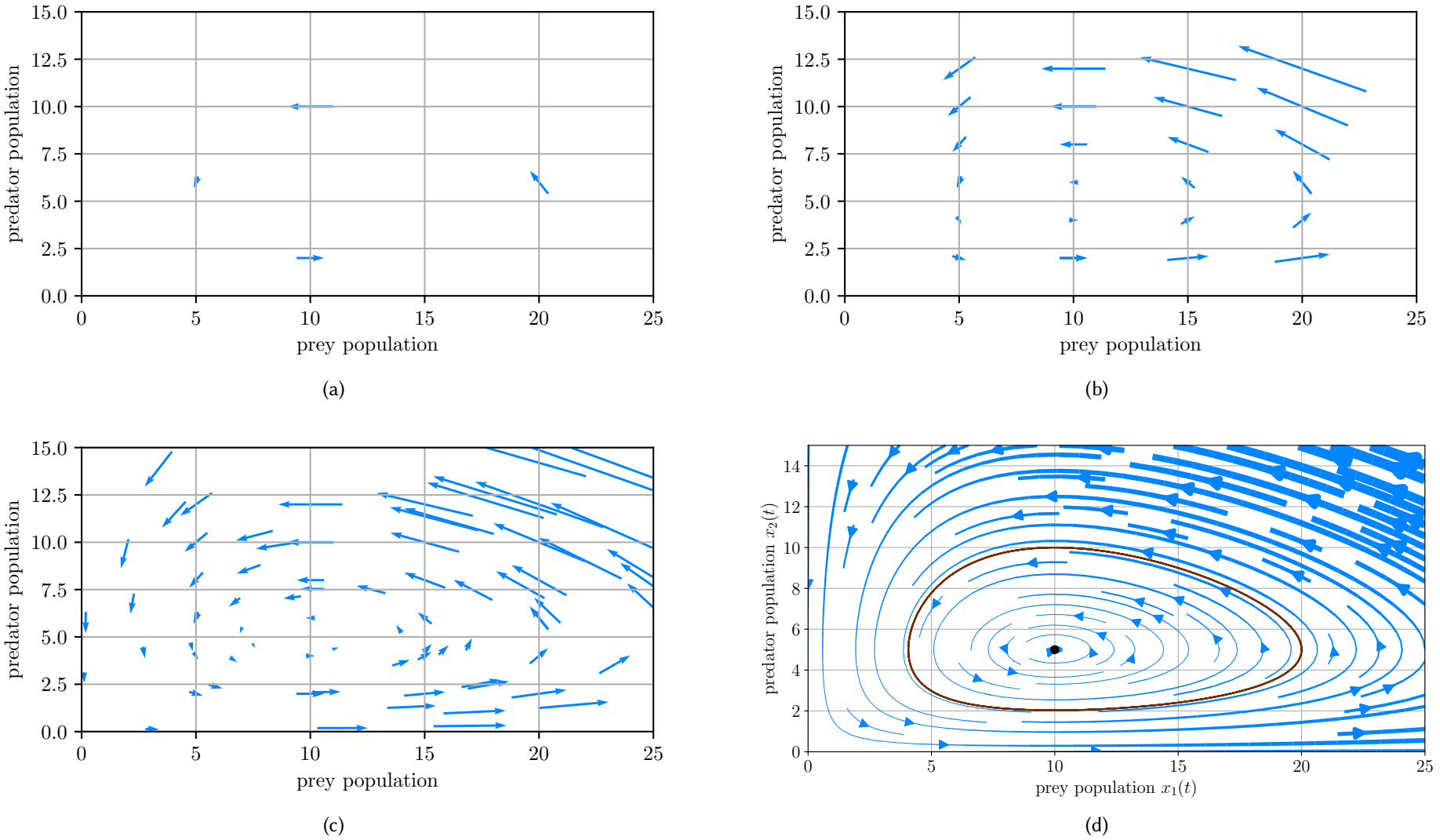


Figure 1.8: Constructing a phase portrait for the 2D Lotka-Volterra predator-prey system. The state space is the (x_1, x_2) plane. The final panel shows the phase portrait from Figure 1.6. Key questions for analysis include: How many equilibria exist? Are they stable? Do any trajectories diverge? Do any trajectories converge to an equilibrium?

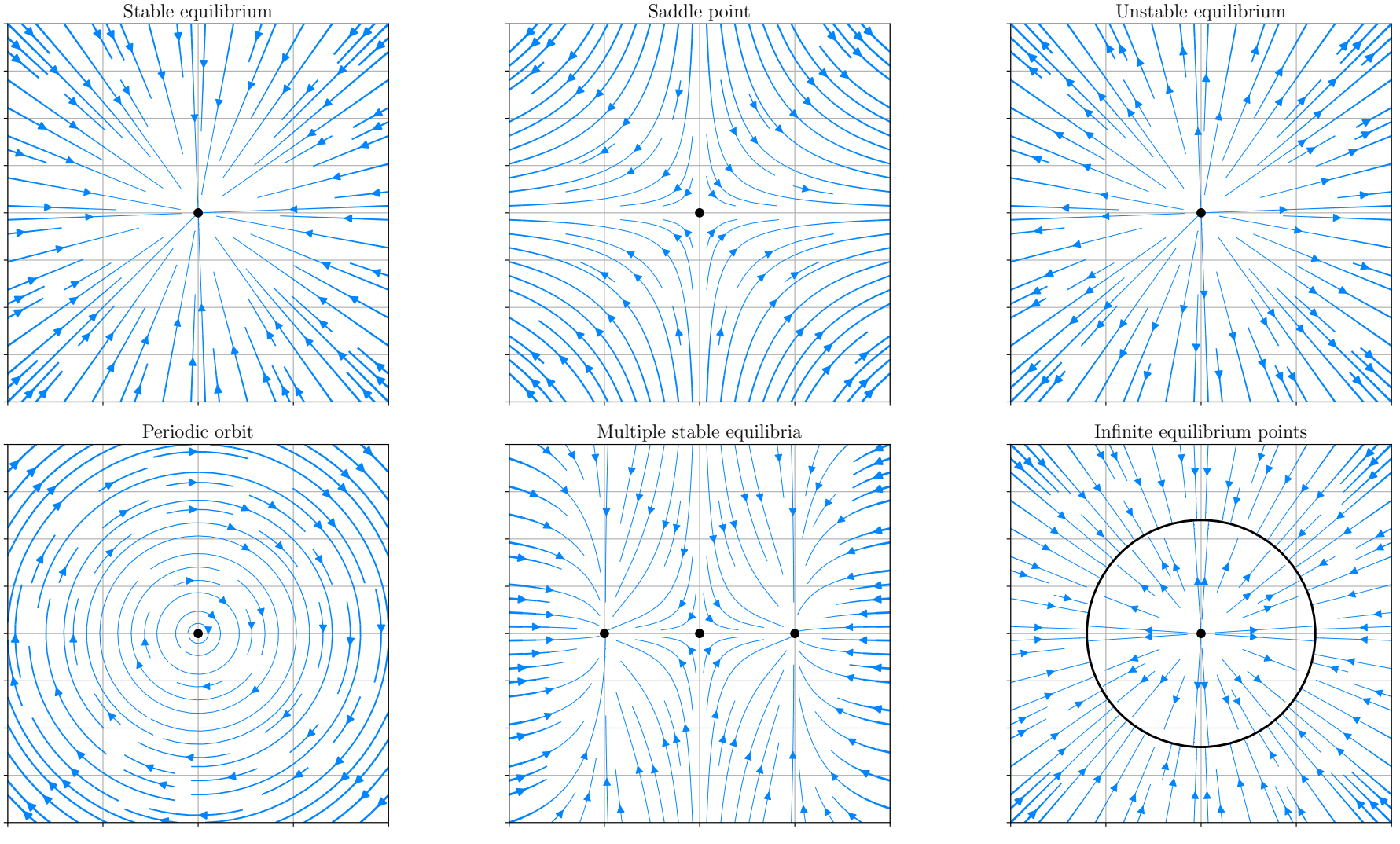


Figure 1.9: Examples of phase portraits.

First row: $\begin{cases} \dot{x} = -x \\ \dot{y} = -y \end{cases}$, $\begin{cases} \dot{x} = x \\ \dot{y} = -y \end{cases}$, and $\begin{cases} \dot{x} = x \\ \dot{y} = y \end{cases}$.

Second row: $\begin{cases} \dot{x} = y \\ \dot{y} = -x \end{cases}$, $\begin{cases} \dot{x} = x(1 - x^2) \\ \dot{y} = -y \end{cases}$, and $\begin{cases} \dot{x} = -x(1 - (x^2 + y^2)) \\ \dot{y} = -y(1 - (x^2 + y^2)) \end{cases}$.

1.5 Historical notes and further resources

The historical development of dynamical systems was not without controversy. For example, Galileo Galilei (1564–1642) faced severe consequences from the Catholic Church for advocating the heliocentric model, i.e., the once-revolutionary idea that the Earth revolves around the Sun. This view directly challenged the long-held geocentric beliefs of Aristotle (384–322 BC) and Ptolemy (c. AD 100–170), which had been adopted and defended by the Church. Galileo's support for the heliocentric theory contradicted centuries of philosophical and scientific tradition and sparked a broader conflict between emerging scientific thought and established authority.

For further reading, an excellent reference is the textbook (Strogatz, 2015) and, specifically, Chapters 2 "Flows on the line" and 6 "Phase plane". Numerous additional example systems can be found in classic textbooks on control such as (DiStefano et al., 1997; Ogata, 2003; Dorf and Bishop, 2011; Nise, 2019; Franklin et al., 2015), in (Åström and Murray, 2021, Chapter 3) and throughout (Strogatz, 2015). Extensive additional information is available on wikipedia, for example, see the page on the logistic equation.

An online application that draws phase portraits (more accurate than the Python's `streamplot`) is "Phase Plane - Nonlinear System with Nullclines" by Patrick Davix.

Interesting youtube videos (links are functional as of July 2025):

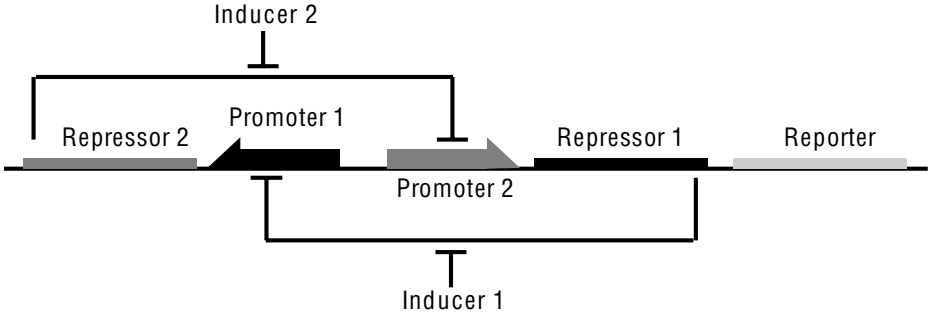
- Mechanical systems
  - simple harmonic motion and London's Millennium Bridge (9m 10s) and mathematics of harmonic motions (8m 9s)
  - damping and resonance (5m 3s) and safety in skyscrapers (tuned mass damper) (9m 36s)
  - the double pendulum and the chaos phenomenon (short)
  - resonance and the 1940 Tacoma bridge collapse
- Biology
  - simplest genetic circuit: the toggle switch (9m 49s)
  - a white blood cell chases bacteria in real life (29s)
- Ecology
  - schooling and collective intelligence in animals (with intervention by Dr. Ian Couzin, 8m 42s)
  - why do fish swim (3m 48s), and why do they swim in harmony (6m 6s)
- Coupled oscillators
  - synchronizing oscillators (short)
  - synchronization out of chaos (with intervention by Dr. Stephen Strogatz, 20m 57s)
- Traffic networks
  - traffic causes and control strategies (4m 29s)
- Last, but not least, an authoritative primer on systems of weights and measures:
  - Washington's dream (4m 50s)

1.6 Appendix: Two dimensional examples: bistability in synthetic biology

From Wikipedia: "Synthetic biology is a multidisciplinary field of science that focuses on living systems and organisms, and it applies *engineering principles* to develop new *biological parts, devices, and systems* or to redesign existing systems found in nature"

The field synthetic biology and systems biology studies *regulatory networks*, that is, groups of molecules (genes, proteins, etc) with varying concentration that interact and promote or repress each other. A *toggle switch* is a simple regulatory network that manifests *bistability*, that is, the ability to remain stably in one of two states and to be toggled between these two states using a stimulus.

(a) An electromechanical toggle switch



(b) A diagram illustrating a biological toggle switch is shown. While the specific chemicals and their interactions may not be immediately clear, a simplified mathematical model will be introduced below for clarity. This image is taken from the seminal work by (Gardner et al., 2000), which demonstrates the successful creation of a synthetic, bistable gene-regulatory network within *Escherichia coli* bacteria.

Figure 1.10: Synthetic biology holds the potential to design biological circuits in much the same way we design electrical circuits and electromechanical systems.

We present here a simple (possibly the simplest) dimensionless mathematical model of the toggle switch which exhibits bistability. Consider two genes with concentrations u and v, the *toggle switch dynamics* is

$$\begin{aligned} \dot{u} &= \frac{\alpha_1}{1+v^{n_1}} - \beta u, \\ \dot{v} &= \frac{\alpha_2}{1+u^{n_2}} - \beta v, \end{aligned} \tag{1.9}$$

where:

- the quantity $\frac{\alpha_1}{1+v^{n_1}}$ is the *production rate* of u, inhibited by v,
- the quantity $\frac{\alpha_2}{1+u^{n_2}}$ is the production rate of v, inhibited by u,
- the quantities n_1 and n_2 are *sensitivity coefficients* indicating the non-linearity of the repression,
- the quantities α_1 and α_2 are the *maximum production rates* of u and v, respectively, and
- the quantity β is the *degradation rate* common for both u and v (we choose it equal for simplicity).

Note: To describe the production rates, we used a *Hill function* $f(u) = \frac{\alpha}{1+u^n}$. This expression is derived from the theory of mass action kinetics in chemistry. The sensitivity parameter n plays a key role.

As we show in the next Figure 1.11, the system can exhibit bistability, meaning it can remain stable in a state where u is high and v is low, or vice versa. An external stimulus can be employed to switch between these states.

Numerical analysis of the toggle switch dynamics

```python
# Python libraries
import numpy as np; import matplotlib.pyplot as plt; from ...
    scipy.integrate import odeint;
plt.rcParams.update({"text.usetex": True, "font.family": "serif", ...
    "font.serif": ["Computer Modern Roman"], "font.size": 14 })

# Toggle switch dynamics
def toggle_switch(Y, t, alpha1, alpha2, n1, n2, beta):
    u, v = Y
    dudt = alpha1 / (1 + v**n1) - beta * u
    dvdt = alpha2 / (1 + u**n2) - beta * v
    return [dudt, dvdt]
# Parameters
alpha1, alpha2 = 1.0, 1.0  # production rates
n1 = 3.53; n2 = 3.55       # Hill coefficients
beta = 0.5                 # degradation rate

# Initial conditions: [u0, v0] and Time array
initial_conditions = [ [0.35, 1.1],  [.9, 0.5],  [1, 2], [2.5, 1] ]
t = np.linspace(0, 10, 1000); plt.figure(figsize=(10,4.6))

# Solve and plot for each initial condition
colors = ['b', 'g', 'r', 'k']
for idx, ic in enumerate(initial_conditions):
    Y = odeint(toggle_switch, ic, t, args=(alpha1, alpha2, n1, n2, beta))
    u, v = Y.T; plt.plot(t, u, color=colors[idx], label=f'$u_0={ic[0]}, ...
        v_0={ic[1]}$')
    plt.plot(t, v, '--', color=colors[idx], label=f'$u_0={ic[0]}$, ...
        $v_0={ic[1]}$')

# Annotate the plot and save to PDF (plt.xlabel('Time, t'))
# plt.title('Toggle Switch', fontsize=16)
plt.ylabel('$u (t)$ (solid) $v(t)$ (dashed)', ...
    fontsize=16);plt.legend(fontsize=10);
plt.xlabel('time $t$', fontsize=16);
plt.xlim(0, 10); plt.ylim(0, 2.25); plt.grid(True)
plt.savefig("toggle-switch.pdf", bbox_inches='tight')

# Phase portrait
u, v = np.meshgrid(np.linspace(0, 4, 20), np.linspace(0, 3, 20))
du, dv = toggle_switch([u, v], 0, alpha1, alpha2, n1, n2, beta)
magnitude = np.sqrt(du**2 + dv**2); plt.figure(figsize=(10,4.6));
plt.streamplot(u, v, du, dv, density=.75, color='#0085ff', ...
    linewidth=magnitude, broken_streamlines=False, arrowsize=1.5)
for idx, ic in enumerate(initial_conditions):
    Y = odeint(toggle_switch, ic, t, args=(alpha1, alpha2, n1, n2, beta))
    plt.plot(Y[:,0], Y[:,1], color=colors[idx], label=f'u0={ic[0]}, ...
        v0={ic[1]}')
eq_points = np.array([[1., 1.], [0.17, 2.], [2., 0.17]])
plt.scatter(eq_points[:, 0], eq_points[:, 1], color='black', s=50, zorder=5)
plt.xlabel('$u(t)$', fontsize=16); plt.ylabel('$v(t)$', fontsize=16); ...
    plt.xlim(0, 4); plt.ylim(0, 3)
plt.savefig("toggle-switch-phase.pdf", bbox_inches='tight')
```

Listing 1.3: Python script generating Figure 1.11. Available at toggle-switch.py

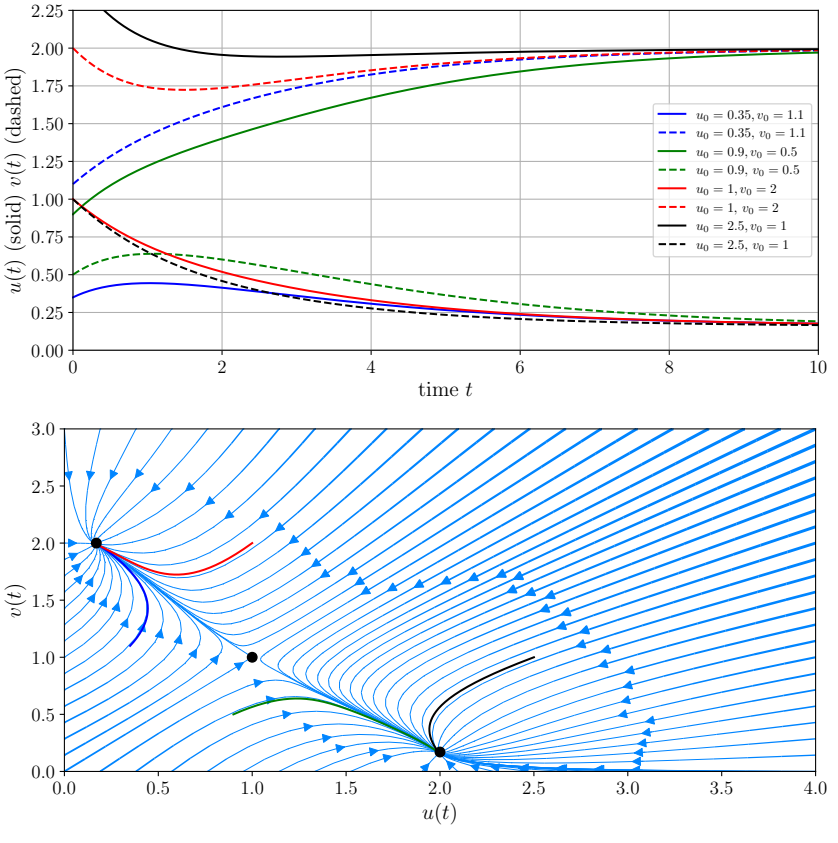


Figure 1.11: Solutions and phase portrait for the toggle switch dynamics (1.9). Top image: foud sample solutions ($u(t)$ solid and $v(t)$ dashed vs t). Bottom image: Phase portrait. Two trajectories (green and black) converge to the "(u, v) = on/off" equilibrium, the other two (red and blue) to the "(u, v) = off/on" equilibrium.

1.7 Exercises

E1.1 **Closed-form solution of the logistic equation.** Verify that

(i) the exponential function $x(t) = x_0\,\mathrm{e}^{rt}$ is the solution to the linear differential equation $\dot{x} = rx$ with initial condition $x(0) = x_0$;

(ii) the logistic function $x(t) = \dfrac{Kx_0\,\mathrm{e}^{rt}}{K + x_0(\mathrm{e}^{rt} - 1)}$ is the solution to the logistic differential equation

$$\dot{x} = rx\left(1 - \frac{x}{K}\right), \qquad x(0) = x_0.$$

Answer:

(i) The solution follows simply by taking the derivative of $x(t) = x_0\,\mathrm{e}^{rt}$ with respect to t:

$$\dot{x} = rx_0\,\mathrm{e}^{rt} = rx$$

An alternative method is via integration:

$$\frac{dx}{dt} = rx \quad \Longrightarrow \quad \frac{dx}{x} = rdt \quad \Longrightarrow \quad \int_0^t \frac{dx}{x} = rt \quad \Longrightarrow \quad \ln(x(t)) = \ln(x(0)) + rt \quad \Longrightarrow \quad x(t) = x(0)\,\mathrm{e}^{rt}.$$

(ii) To solve the logistic equation, we separate the variables and then integrate. Starting with:

$$\frac{dx}{x(1 - x/K)} = rdt.$$

We integrate both sides, with the right side integrated with respect to t and the left side with respect to x. The integral will yield an equation of the form:

$$-\ln\left(\left|\frac{K}{x} - 1\right|\right) = C - rt.$$

Solving for x and evaluating at $x(0) = x_0$, we compute

$$C = \ln\left|\frac{x_0}{x_0 - K}\right|.$$

Finally, the conclusion follows by substituting C into the solution for x yielding:

$$x(t) = \frac{K}{\frac{K - x_0}{x_0}\,\mathrm{e}^{rt} + 1} = \frac{Kx_0\,\mathrm{e}^{rt}}{K + x_0\,(\mathrm{e}^{rt} - 1)}$$

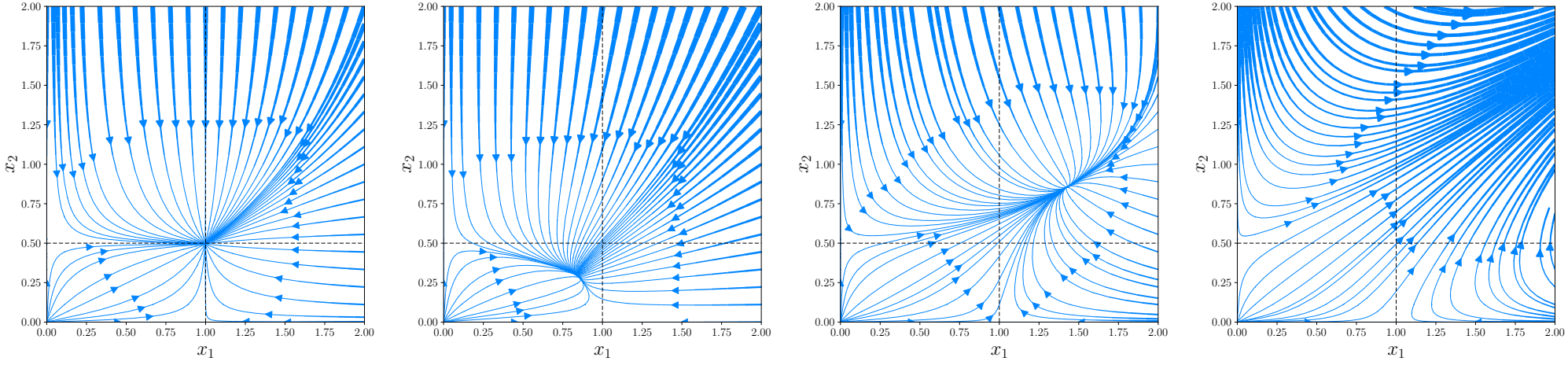
▼

E1.2 **Reasonable and unreasonable predictions of a Lotka-Volterra model.** Given an interaction parameter α taking one of the values $\{0, -0.5, 0.5, 2\}$, consider the following Lotka-Volterra system for two species (note this is not the same predator-prey system in Section 1.3):

$$\begin{aligned} \dot{x}_1 &= x_1 - x_1^2 + \alpha x_1 x_2 \\ \dot{x}_2 &= x_2 - 2x_2^2 + \alpha x_1 x_2 \end{aligned} \tag{1.10}$$

Note: when $\alpha = 0$ the two species do not interact with each other. When $\alpha > 0$, each species has a beneficial effect on the other; this is called *mutualism*. When $\alpha < 0$, each species has a detrimental effect on the other; this is called *competition*.

Note: *For clarity, x_1 and x_2 are normalized numbers, not absolute numbers. For example: the number of individuals of species #1 may be of $x_1 \cdot 10^6$.*



(a) No interaction: $\alpha = 0$ (b) Low competition: $\alpha = -0.5$ (c) Low mutualism: $\alpha = 0.5$ (d) High mutualism: $\alpha = 2$

(i) For the no-interaction case $\alpha = 0$, compute the carrying capacity for each of the two species in isolation and compute each equilibrium point of the two-species system.

(ii) For each case ($\alpha = 0$, $\alpha = -0.5$, $\alpha = 0.5$, $\alpha = 2$), how many equilibria exist?

(iii) For $\alpha \neq 0$, how does the interaction between species affect the location of the equilibria?

(iv) Is the behavior (location of equilibria and asymptotic value of solutions) predicted by the model (1.10) ecologically reasonable for each value of α?

Answer: First, observe that

$$\begin{aligned} \dot{x}_1 &= x_1 - x_1^2 + \alpha x_1 x_2 = x_1(1 - x_1 + \alpha x_2) \\ \dot{x}_2 &= x_2 - 2x_2^2 + \alpha x_1 x_2 = x_2(1 - 2x_2 + \alpha x_1) \end{aligned}$$

(i) Setting the above equations for $\dot{x}_1$ and $\dot{x}_2$ equal to 0 with $\alpha = 0$, we can see that the fixed points are $x_1 = 0, 1$ and $x_2 = 0, 0.5$. Thus, the carrying capacities are 1 and 0

(ii) Setting the above equations equal to zero, we can see that we have at least three equilibria in all four cases:

$$(0,0), (0,1/2), (1,0)\,.$$

Additionally, when $\alpha \neq \pm\sqrt{2}$, a fourth equilibrium also exists, which solves the system of equations

$$\begin{aligned} 0 &= 1 - x_1 + \alpha x_2, \\ 0 &= 1 - 2x_2 + \alpha x_1. \end{aligned}$$

A little linear algebra shows that this system has a unique solution at all four parameter values considered. The fourth equilibrium occurs at the point

$$\left(\frac{2+\alpha}{2-\alpha^2}, \frac{1+\alpha}{2-\alpha^2}\right)$$

(Note that in the case where $\alpha = \pm\sqrt{2}$, this system of equations has no solutions.) When $\alpha = 2$, the solution is $(-2, -3/2)$, which is not physically acceptable since the variables (x_1, x_2) are supposed to be positive or zero.

(iii) The first three equilibria are unaffected by the value of α. We can see that

As α increases from 0, the fourth equilibrium moves up and to the right until it escapes to (∞, ∞) at $\alpha = \sqrt{2}$.

As α decreases from 0, this point first moves down and to the left, and then moves down and to the right until it escapes to $(\infty, -\infty)$ at $\alpha = -\sqrt{2}$.

(iv) We at least expect both populations to be nonnegative and finite. We can see that the values for which this holds are $-1 < \alpha < \sqrt{2}$.

Thus cases (a), (b), and (c) are arguably reasonable, while case (d) certainly is not.

▼

E1.3 **A conserved quantity for the predator-prey system.** With the same notation as in Section 1.3, consider the Lotka-Volterra predator-prey model

$$\begin{aligned} \dot{x}_1 &= \alpha x_1 - \beta x_1 x_2, \\ \dot{x}_2 &= -\delta x_2 + \gamma x_1 x_2. \end{aligned} \tag{1.11}$$

Show that the following quantity is conserved along the solution of the dynamical system:

$$\mathcal{H}(x_1, x_2) = \gamma x_1 - \delta \ln(x_1) + \beta x_2 - \alpha \ln(x_2). \tag{1.12}$$

Hint: Note that the quantity $\mathcal{H}(x_1, x_2)$ along the solution of the dynamical system really is a function of time $\mathcal{H}(x_1(t), x_2(t))$, where $x_1(t)$ and $x_2(t)$ are the solutions the system.

Answer: To prove the conservation of $\mathcal{H}$ along the solutions, we need to understand that $\mathcal{H}(x_1, x_2)$ really is a function of time: $\mathcal{H}(x_1(t), x_2(t))$ and we need to show

$$0 = \frac{d\mathcal{H}}{dt} = \frac{\partial \mathcal{H}}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial \mathcal{H}}{\partial x_2}\frac{dx_2}{dt}. \tag{1.13}$$

We start by computing some partial derivatives:

$$\frac{\partial \mathcal{H}}{\partial x_1} = \gamma - \frac{\delta}{x_1}, \qquad \frac{\partial \mathcal{H}}{\partial x_2} = \beta - \frac{\alpha}{x_2},$$

so that, along the solutions of (1.11), we obtain:

$$\frac{d\mathcal{H}}{dt} = \left(\gamma - \frac{\delta}{x_1}\right)(\alpha x_1 - \beta x_1 x_2) + \left(\beta - \frac{\alpha}{x_2}\right)(-\delta x_2 + \gamma x_1 x_2). \tag{1.14}$$

We now claim that the two terms are equal and opposite. The first term satisfies:

$$\begin{aligned} \left(\gamma - \frac{\delta}{x_1}\right)(\alpha x_1 - \beta x_1 x_2) &= \gamma(\alpha x_1 - \beta x_1 x_2) - \frac{\delta}{x_1}(\alpha x_1 - \beta x_1 x_2) \\ &= \gamma\alpha x_1 - \gamma\beta x_1 x_2 - \delta\alpha + \delta\beta x_2. \end{aligned}$$

The second term satisfies:

$$\begin{aligned} \left(\beta - \frac{\alpha}{x_2}\right)(-\delta x_2 + \gamma x_1 x_2) &= \beta(-\delta x_2 + \gamma x_1 x_2) - \frac{\alpha}{x_2}(-\delta x_2 + \gamma x_1 x_2) \\ &= -\beta\delta x_2 + \beta\gamma x_1 x_2 + \alpha\delta - \alpha\gamma x_1. \end{aligned}$$

This concludes the proof that $\mathcal{H}(x_1(t), x_2(t))$ does not change with time. ▼

E1.4 **A delayed predator–prey model.** Let $x(t)$ denote the prey population (e.g., rabbits) and $y(t)$ denote the predator population (e.g., foxes) at time t. Consider the system

$$\dot{x}(t) = \alpha x(t) - \beta x(t) y(t), \tag{1.15}$$
$$\dot{y}(t) = -\delta y(t) + \gamma x(t-\tau) y(t), \tag{1.16}$$

where $\tau > 0$ is a fixed delay representing the gestation or maturation period of the predator population.

(i) Conceptually, why might the term $x(t-\tau)$ appear in the predator equation but not in the prey equation?

(ii) If $\tau = 0$, how does the system reduce to the classical Lotka–Volterra form?

Answer:

(i) The prey equation reflects that prey birth and death rates respond essentially instantaneously to predation pressure. However, predator reproduction depends on prey availability at an earlier time, since capturing prey today does not immediately translate into new predator births — gestation or maturation introduces a time lag.

(ii) If $\tau = 0$, then $x(t-\tau) = x(t)$ and the system becomes

$$\dot{x} = \alpha x - \beta x y, \quad \dot{y} = -\delta y + \gamma x y,$$

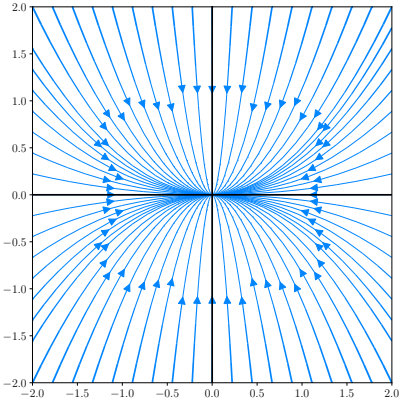
which is exactly the classical Lotka–Volterra predator–prey model.

▼

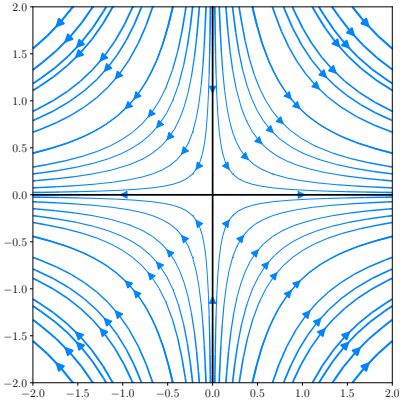
E1.5 **Matching phase portraits to dynamics.** Given a real number r taking six possible values in $\{-2, -1, -0.5, 0, 1, 5\}$, consider the linear dynamical system

$$\dot{x} = rx,$$
$$\dot{y} = -y.$$

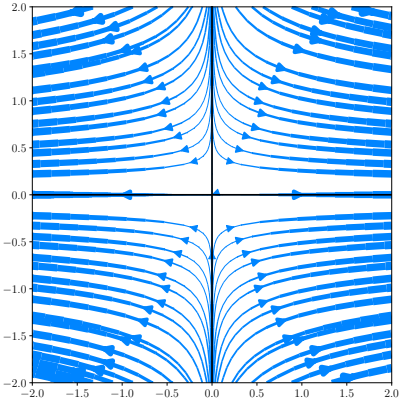
Match the six possible values of r to each of the six images below.



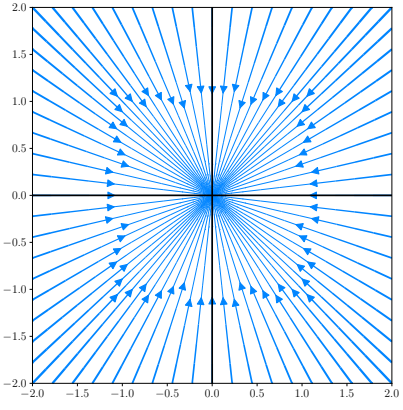
(a) $r =$ ☐



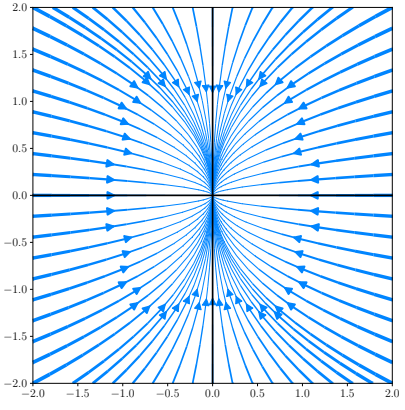
(b) $r =$ ☐



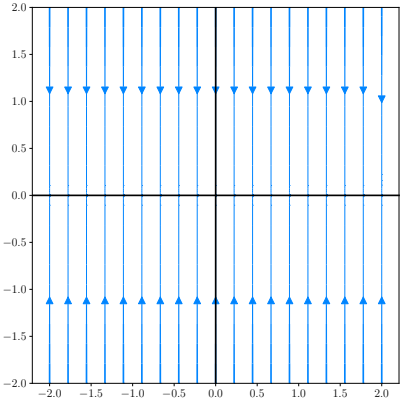
(c) $r =$ ☐



(d) $r =$ ☐



(e) $r =$ ☐



(f) $r =$ ☐

Answer: The answer is:

(a) $r = -0.5$, (b) $r = 1$, (c) $r = 5$, (d) $r = -1$, (e) $r = -2$, (f) $r = 0$

Here is the explanation: If $\dot{x} > 0$, the arrow points to the right and if $\dot{x} < 0$, the arrow points to the left. With the same concept, if $\dot{y} > 0$, the arrow points upward and if $\dot{y} < 0$, the arrow points downward (ref. Chapter 1, slide 23). In this case, the change rate $\dot{y}$ is not affected by r and we can see how the lines pointing towards the x-axis at $y = 0$, curve.

(a) When $r = -0.5, \dot{x} = -0.5x$ and $\dot{y} = -y$. There are two vectors, one points towards the x-axis at $y = 0$ and the other points towards the y-axis at $x = 0$. When the value of x is approaching 0, the change rate $\dot{x}$ is decreasing. Therefore, the lines pointing towards the x-axis at $y = 0$ is slightly curved because $\dot{x} < \dot{y}$.

(b) When $r = 1, \dot{x} = x$ and $\dot{y} = -y$. There are two vectors, one points towards the x-axis at $y = 0$ and the other points away from the y-axis at $x = 0$. Therefore, the lines curved and pointed out from the y-axis at $x = 0$. When the value of x is approaching 2 or -2, the change rate $\dot{x}$ is also increasing and the lines pointing to the x-axis at $y = 0$ are dramatically curved because $\dot{x} > \dot{y}$.

(c) When $r = 5, \dot{x} = 5x$ and $\dot{y} = -y$. There are two vectors, one points to the x-axis at $y = 0$ and the other points away from the y-axis at $x = 0$. Because the change rate $\dot{x}$ is much greater than $\dot{y}$ when x is increasing, it caused the lines to significantly point away from the y-axis at $x = 0$

(d) When $r = -1, \dot{x} = -x$ and $\dot{y} = -y$. There are two vectors, one points towards the x-axis at $y = 0$ and the other points towards y-axis at $x = 0$. The magnitude depends on the value of x and y. Adding two vectors together results in arrows pointing straight to $(0, 0)$ because $\dot{x} = \dot{y}$ everywhere.

(e) When $r = -2, \dot{x} = -2x$ and $\dot{y} = -y$. There are two vectors, one points towards the x-axis at $y = 0$ and the other points towards the y-axis at $x = 0$. When the value of x is approaching 0, the change rate $\dot{x}$ is decreasing. Therefore, the lines pointing to x-axis at $y = 0$ is slightly curved because $\dot{x} < \dot{y}$. However, with a larger x value, the change rate $\dot{x}$ is bigger and curved the lines in a greater magnitude because $\dot{x} > \dot{y}$.

(f) When $r = 0, \dot{x} = 0$ and $\dot{y} = -y$. We can see the straight lines point towards the x-axis at $y = 0$ when we apply $y = -2$ to 0 and $y = 2$ to 0.

▼

E1.6 **Discrete-time dynamical systems.** This exercise is meant to illustrate that dynamical systems exist also over the discrete-time domain and not only over continuous time. Instead of the time variable t (taking values in the non-negative real numbers), we let k be the discrete time variable, that is, a non-negative integer. An interesting example of discrete-time dynamical systems arises in the domain of finance, in the context of investments and accumulation of bank deposits.

A single initial deposit: Given a scalar a and an initial condition $x(0)$, consider the discrete-time equation $x(k+1) = ax(k)$, for all nonnegative integers k. Show that the solution is $x(k) = x(0)a^k$.

Note: *Assume you deposit an amount $x(0)$ at a bank today. Let the coefficient a satisfy $a = 1 + i$, where i is the yearly interest offered by the bank. Then $x(k)$ is the value in your bank account k years from now.*

An initial deposit, plus yearly deposits: Consider the discrete-time equation $x(k+1) = ax(k) + b$, where b is a constant input or forcing term. Show that the solution from initial condition $x(0)$ is

$$x(k) = \begin{cases} x(0) + kb, & \text{if } a = 1, \\ a^k x(0) + \dfrac{1-a^k}{1-a} b & \text{if } a \neq 1. \end{cases} \tag{1.17}$$

Hint: Recall the geometric series $\dfrac{1-a^k}{1-a} = 1 + a + a^2 + \cdots + a^{k-1}$.

Note: *This scenario assumes that, after depositing an amount $x(0)$ at year 0, you deposit the amount b each following year.*

Answer: We prove only equation (1.17), since the case where $b = 0$ is an immediate consequence. The case when $a = 1$ is trivial. We need to show that $x(k) = a^k x(0) + \frac{1-a^k}{1-a} b$ for $a \neq 1$, when $x(k+1) = ax(k) + b$. (The case $a = 1$ is elementary.) We proceed by induction. First, at $k = 0$, we verify

$$x(0) = a^k x(0) + \frac{1-a^k}{1-a} b \Big|_{k=0} = x(0) + \frac{0}{1-a} b = x(0). \tag{1.18}$$

Second, we assume the statement is true at k and we need to show that it holds at $k+1$. We compute

$$\begin{aligned} x(k+1) = ax(k) + b &= a\Big(a^k x(0) + \frac{1-a^k}{1-a} b\Big) + b \\ &= a^{k+1} x(0) + a\frac{1-a^k}{1-a} b + b \\ &= a^{k+1} x(0) + \Big(a\frac{1-a^k}{1-a} + 1\Big) b \\ &= a^{k+1} x(0) + \frac{a - a^{k+1} + 1 - a}{1-a} b = a^{k+1} x(0) + \frac{1-a^{k+1}}{1-a} b. \end{aligned}$$

This expression matches the form of the solution for $k+1$, so the inductive step holds. ▼

E1.7 **A discrete-time logistic equation.** Consider the difference equation

$$x_{k+1} = r\,x_k\,(1 - x_k) \tag{1.19}$$

where $0 < r \leq 4$ and the initial condition satisfies $x_0 \in [0,1] \subset \mathbb{R}$.

(i) Show by induction that $x_k \in [0,1]$ for all k.

(ii) Find all equilibrium points in $[0,1]$ of this difference equation.

(iii) Determine for which values of r each equilibrium is locally stable or unstable, using the stability test $\left|\frac{d}{dx}\big(rx(1-x)\big)\right| < 1$.

Answer:

(i) Base case: $x_0 \in [0,1]$ by assumption. Assume $x_k \in [0,1]$. Then $0 \leq x_k \leq 1$ implies $1 - x_k \geq 0$ and $x_k\,(1 - x_k) \leq \frac{1}{4}$. Hence

$$0 \leq x_{k+1} = r\,x_k\,(1 - x_k) \leq \tfrac{r}{4} \leq 1 \tag{1.20}$$

so $x_{k+1} \in [0,1]$, completing the induction.

(ii) Setting $x_{k+1} = x_k = x^*$ gives

$$x^* = r\,x^*\,(1 - x^*) \iff x^*\big(r\,(1 - x^*) - 1\big) = 0 \tag{1.21}$$

hence the equilibria are

$$x^* = 0 \quad \text{and} \quad x^* = 1 - \tfrac{1}{r} \tag{1.22}$$

where the latter exists only for $r > 1$ (otherwise $x^* < 0$).

(iii) Let $\phi(x) = r\,x\,(1 - x)$. Then

$$\phi'(x) = r\,(1 - 2x) \tag{1.23}$$

At $x^* = 0$, $\phi'(0) = r$, so by $|\phi'(0)| < 1$ the equilibrium $x^* = 0$ is stable if $0 < r < 1$ and unstable if $r > 1$. At $x^* = 1 - \frac{1}{r}$,

$$\phi'\big(1 - \tfrac{1}{r}\big) = r\,(1 - 2\,(1 - \tfrac{1}{r})) = -r + 2 \tag{1.24}$$

so $x^* = 1 - \frac{1}{r}$ is stable if $|-r + 2| < 1$, i.e. $1 < r < 3$, and unstable if $r > 3$.

▼

E1.8 **Finding typos in Python code.** Each of the following three programs contains precisely one programming mistake. Identify the mistake and the line in which it happens.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Logistic equation
def log_eq(x, t, r, K)
    return r * x * (1 - x / K)

# Parameters
r, K = 0.1, 100
initial_conditions = [1, 10, 50, 100]
times = np.linspace(0, 125, 1000)

# Solve and plot
plt.figure()
for x0 in initial_conditions:
    solution = odeint(log_eq, x0, ...
        times, args=(r, K))
    plt.plot(times, solution)

plt.xlabel("Time"), plt.ylabel("Population")
plt.legend(), plt.show()
```

Listing 1.4:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Logistic equation
def log_eq(x, t, r, K):
    return r * x * (1 - x / K)

# Parameters
r, K = 0.1, 100
initial_conditions = [1, 10, 50, 100]
times = np.linspace(0, 125, 1000)

# Solve and plot
plt.figure()
for x0 in initial_conditions:
    solution = odeint(log_eq, x0, ...
        times, args=(r, K))
    plt.plot(times, soluton)

plt.xlabel("Time"), plt.ylabel("Population")
plt.legend(), plt.show()
```

Listing 1.5:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Logistic equation
def log_eq(x, t, r, K):
    return r * x * (1 - x / K)

# Parameters
r, K = 0.1, 100
initial_conditions = [1, 10, 50, 100]
times = np.linspace(0, 125, 1000)

# Solve and plot
plt.figure()
for x0 in initial_conditions:
    solution = odeint(log_eq, times, ...
        args=(r, K))
    plt.plot(times, solution)

plt.xlabel("Time"), plt.ylabel("Population")
plt.legend(), plt.show()
```

Listing 1.6:

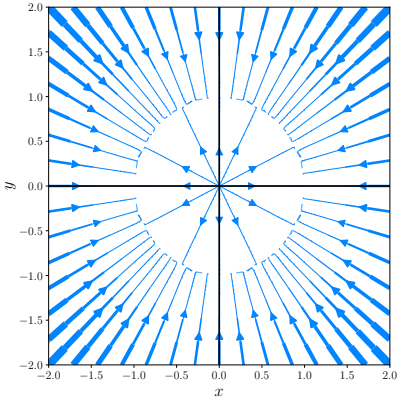
E1.9 **Matching phase portraits to dynamics.** Given a number ω taking values in $\{-1, 0, 1\}$, consider the nonlinear dynamical system

$$\dot{x} = \omega y + \big(1 - (x^2 + y^2)\big)x,$$
$$\dot{y} = -\omega x + \big(1 - (x^2 + y^2)\big)y.$$

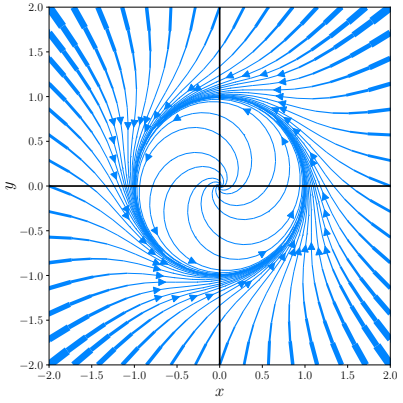
(i) Match the three possible values of ω to each of the three phase portraits below.

(ii) For each value of ω, calculate all the equilibrium points of the resulting system and their stability.

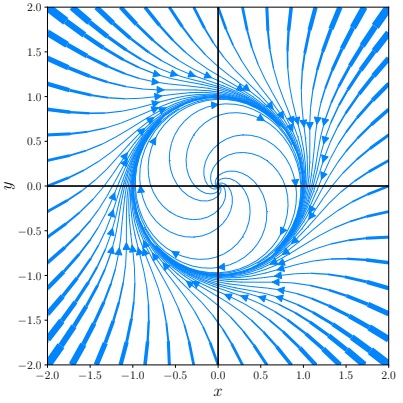
Hint: An equilibrium point x^* is *marginally stable* if nearby trajectories converge to x^* or remain near x^*.



(a) $\omega =$ ☐



(b) $\omega =$ ☐



(c) $\omega =$ ☐

E1.10 **Understanding phase portraits.** Here are some basic questions about phase portraits for 2-dimensional systems. Please provide a short 1-sentence response.

(i) why can there be only one solution curve passing through a given point in the phase portrait?

(ii) what do the arrows on the trajectories in a phase portrait represent?

(iii) why can solution curves never cross each other in a phase portrait?

(iv) what are equilibrium points, and how do you find them on a phase portrait?

(v) how can you tell if an equilibrium point is stable or unstable by looking at the phase portrait?

Note: *Please provide a short 1-sentence response.*

E1.11 **One-dimensional phase portrait.** Consider the one-dimensional system

$$\dot{x} = -\sin(x)$$

where x is in the interval from -2π to 4π included.

(i) Identify all equilibrium points within the given interval.

(ii) Plot the *one-dimensional* phase portrait of the system. Make sure to clearly plot each equilibrium point, and draw arrows to indicate the direction of the flow between equilibria.

(iii) Identify which equilibrium points are stable and which are unstable.

(iv) Consider the initial condition $x(0) = 3\pi/2$. To which equilibrium will the system go?

Bibliography

K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2 edition, 2021. URL http://www.cds.caltech.edu/~murray/books/AM08/pdf/fbs-public_24Jul2020.pdf.

J. J. DiStefano, , A. R. Stubberu, and I. J. Williams. *Schaum's Outline of Feedback and Control Systems*. McGraw-Hill, 2 edition, 1997.

R. H. Dorf and R. C. Bishop. *Modern Control Systems*. Prentice Hall, 2011. ISBN 0136024580.

G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, 4 edition, 2015.

T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–342, 2000. .

N. S. Nise. *Control Systems Engineering*. John Wiley & Sons, 2019. ISBN 1119590132.

E. P. Odum. *Fundamentals of Ecology*. Saunders Company, 1959.

K. Ogata. *Dynamical Systems*. Pearson, 4 edition, 2003. ISBN 0131424629.

S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, 2 edition, 2015. ISBN 9780813350844.