Ordinary Differential Equations

Lecture Notes

Francisco Richter

Università della Svizzera italiana
Faculty of Informatics

August 18, 2025

1 Lecture 2: Systems of First-Order Equations

1.0.1 Introduction to Systems

Many phenomena in science and engineering involve multiple interacting quantities that change simultaneously. A predator-prey ecosystem involves both predator and prey populations, a mechanical system has both position and velocity, and an electrical circuit may have multiple currents and voltages. Such situations naturally lead to systems of differential equations.

A system of first-order differential equations has the general form:

$$\frac{dx_1}{dt} = f_1(t, x_1, x_2, \ldots, x_n) \tag{1}$$

$$\frac{dx_2}{dt} = f_2(t, x_1, x_2, \ldots, x_n) \tag{2}$$

$$\vdots \tag{3}$$

$$\frac{dx_n}{dt} = f_n(t, x_1, x_2, \ldots, x_n) \tag{4}$$

This can be written compactly in vector notation as:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \tag{5}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the state vector and $\mathbf{f} = (f_1, f_2, \ldots, f_n)^T$ is the vector field.

The power of this formulation is that it reduces the study of systems to the study of vector fields in n-dimensional space. This geometric perspective provides deep insights into the behavior of complex dynamical systems.

1.0.2 Reduction of Higher-Order Equations

Any higher-order differential equation can be converted to a system of first-order equations. This reduction is fundamental because it allows us to study all differential equations using the unified framework of first-order systems.

Consider the second-order equation:

$$\frac{d^2y}{dt^2} = F\left(t, y, \frac{dy}{dt}\right) \tag{6}$$

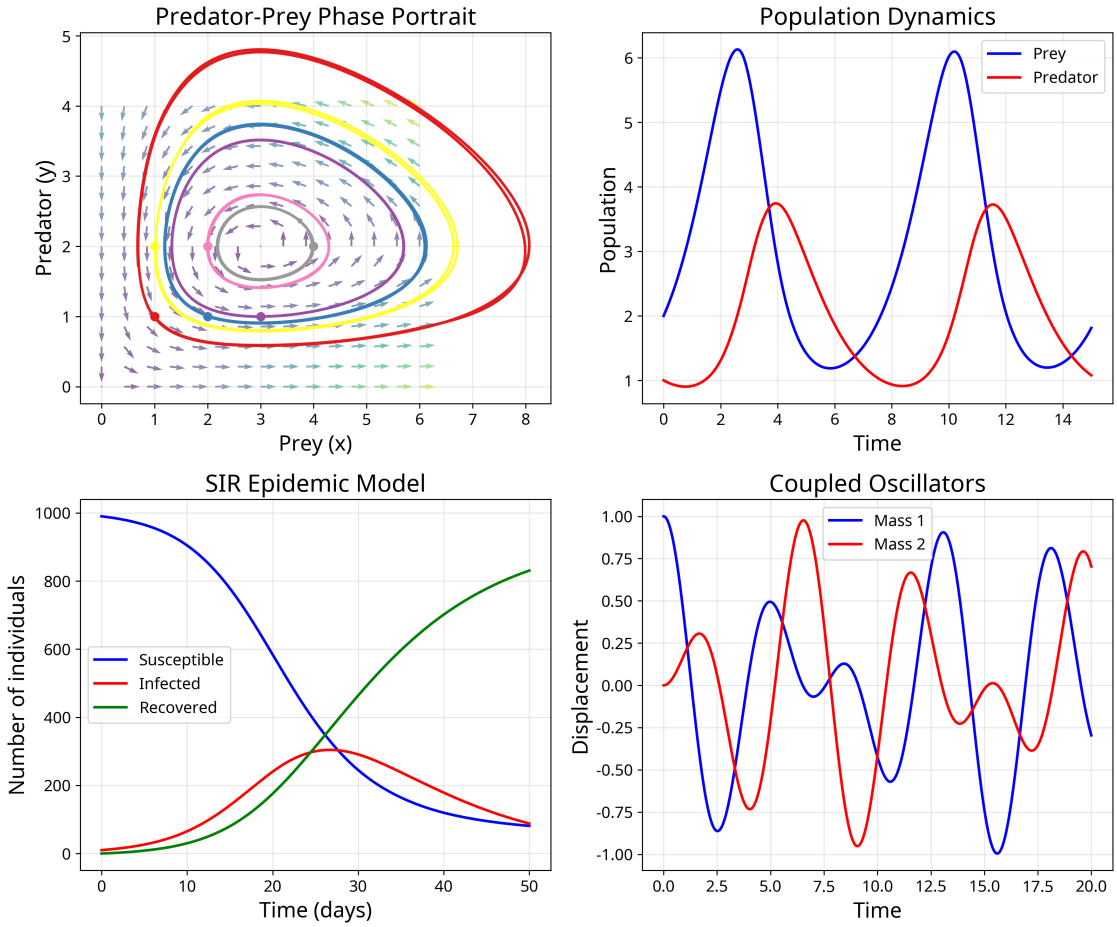

Figure 1: Examples of first-order systems: (top left) Predator-prey phase portrait showing closed orbits, (top right) Population dynamics time series, (bottom left) SIR epidemic model, (bottom right) Coupled oscillators demonstrating energy exchange.

We introduce new variables: $x_1 = y$ and $x_2 = \frac{dy}{dt}$. Then:

$$\frac{dx_1}{dt} = x_2 \tag{7}$$

$$\frac{dx_2}{dt} = F(t, x_1, x_2) \tag{8}$$

This technique extends to equations of any order, always producing a system of first-order equations with the same number of equations as the order of the original equation.

2.1: Harmonic Oscillator The equation for a harmonic oscillator is:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \tag{9}$$

Setting $x_1 = x$ and $x_2 = \frac{dx}{dt}$, we get:

$$\frac{dx_1}{dt} = x_2 \tag{10}$$

$$\frac{dx_2}{dt} = -\omega^2 x_1 \tag{11}$$

This system describes circular motion in the (x_1, x_2) phase plane with angular frequency ω.

1.0.3 Existence and Uniqueness for Systems

The existence and uniqueness theory for systems parallels that for scalar equations, but with vector-valued functions and matrix derivatives.

2.1: Existence and Uniqueness for Systems Consider the initial value problem:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \tag{12}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{13}$$

If $\mathbf{f}$ and the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ are continuous in a neighborhood of $(t_0, \mathbf{x}_0)$, then there exists a unique solution in some interval containing t_0.

The Jacobian matrix is:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{14}$$

This theorem guarantees that through each point in the domain, there passes exactly one solution trajectory. This means that trajectories in phase space cannot cross each other, a fundamental property that constrains the possible behavior of dynamical systems.

1.0.4 Phase Space and Trajectories

The phase space (or state space) of a system is the space of all possible states $\mathbf{x}$. For an n-dimensional system, the phase space is $\mathbb{R}^n$. Each point in phase space represents a complete specification of the system's state at a given time.

A trajectory (or orbit) is a curve in phase space that represents the evolution of the system over time. Starting from initial condition $\mathbf{x}_0$ at time t_0, the trajectory is the set of points $\{\mathbf{x}(t) : t \geq t_0\}$ where $\mathbf{x}(t)$ is the solution to the initial value problem.

The vector field $\mathbf{f}(t, \mathbf{x})$ assigns a velocity vector to each point in phase space. Trajectories are always tangent to the vector field, flowing along the directions indicated by the field.

1.0.5 Autonomous Systems

When the vector field does not depend explicitly on time, $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$, the system is autonomous. Autonomous systems have special properties:

1. **Time translation invariance**: If $\mathbf{x}(t)$ is a solution, then $\mathbf{x}(t + c)$ is also a solution for any constant c.

2. **Unique trajectories**: Each trajectory is determined entirely by its initial point, independent of the starting time.

3. **Phase portraits**: The collection of trajectories in phase space forms a phase portrait that completely characterizes the system's behavior.

For autonomous systems, we can think of the vector field as defining a flow in phase space, like the flow of a fluid. Trajectories are the streamlines of this flow.

1.0.6 Two-Dimensional Systems

Two-dimensional systems provide the richest setting for developing geometric intuition while remaining visualizable. The general autonomous system in the plane is:

$$\frac{dx}{dt} = f(x, y) \tag{15}$$

$$\frac{dy}{dt} = g(x, y) \tag{16}$$

1.0.7 Nullclines and Flow Analysis

Nullclines are curves where one component of the velocity vector vanishes: - **x-nullclines**: curves where $f(x, y) = 0$, so $\frac{dx}{dt} = 0$ - **y-nullclines**: curves where $g(x, y) = 0$, so $\frac{dy}{dt} = 0$

Nullclines divide the phase plane into regions where the flow has consistent direction. On x-nullclines, trajectories move purely vertically; on y-nullclines, they move purely horizontally.

Equilibrium points occur at intersections of x- and y-nullclines, where both $f(x, y) = 0$ and $g(x, y) = 0$.

2.2: Predator-Prey System The Lotka-Volterra predator-prey model is:

$$\frac{dx}{dt} = ax - bxy \tag{17}$$

$$\frac{dy}{dt} = -cy + dxy \tag{18}$$

where x is prey population, y is predator population, and $a, b, c, d > 0$.
The nullclines are: - x-nullclines: $x = 0$ and $y = a/b$ - y-nullclines: $y = 0$ and $x = c/d$
Equilibria occur at $(0, 0)$ and $(c/d, a/b)$.

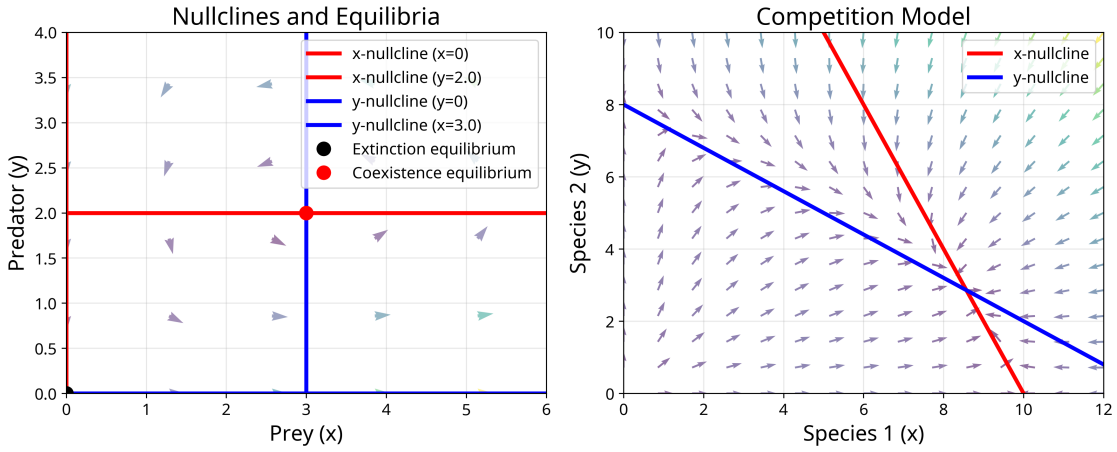

Figure 2: Nullclines and phase flow analysis: (left) Predator-prey system showing nullclines, equilibria, and direction field with closed orbits, (right) Competition model demonstrating different equilibrium types and flow patterns.

1.0.8 Direction Fields for Systems

The direction field for a two-dimensional system assigns a direction vector $(f(x, y), g(x, y))$ to each point (x, y) in the plane. This generalizes the direction field concept from scalar equations.

Constructing the direction field involves: 1. Choose a grid of points (x_i, y_j) 2. At each point, calculate the direction vector $(f(x_i, y_j), g(x_i, y_j))$ 3. Draw an arrow in this direction (usually normalized for visibility) 4. Trajectories flow along the direction field

The direction field provides immediate visual information about the system's behavior without solving the equations explicitly.

Computational Note: The file `lecture2.py` contains comprehensive tools for generating direction fields, plotting nullclines, and visualizing trajectories for two-dimensional systems.

1.0.9 Linear Systems

Linear systems form a special class where complete analytical solutions are possible. A linear system has the form:

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}(t)\mathbf{x} + \mathbf{b}(t) \tag{19}$$

where $\mathbf{A}(t)$ is an $n \times n$ matrix and $\mathbf{b}(t)$ is an n-dimensional vector.

When $\mathbf{b}(t) = \mathbf{0}$, the system is homogeneous:

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}(t)\mathbf{x} \tag{20}$$

1.0.10 Constant Coefficient Systems

When $\mathbf{A}$ is constant, the system becomes:

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} \tag{21}$$

The solution involves the matrix exponential:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 \tag{22}$$

where $e^{\mathbf{A}t} = \sum_{k=0}^{\infty} \frac{(\mathbf{A}t)^k}{k!}$ is the matrix exponential.

1.0.11 Eigenvalue Analysis

For constant coefficient linear systems, the behavior is determined by the eigenvalues and eigenvectors of matrix $\mathbf{A}$.

If $\mathbf{A}$ has eigenvalue λ with eigenvector $\mathbf{v}$, then $\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$ is a solution. This gives: - **Real eigenvalues**: Exponential growth ($\lambda > 0$) or decay ($\lambda < 0$) along eigenvector directions - **Complex eigenvalues**: Spiral motion with exponential growth or decay

2.3: Two-Dimensional Linear System Consider:

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} 1 & 2 \\ 3 & 2 \end{pmatrix} \mathbf{x} \tag{23}$$

The characteristic equation is:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = (1 - \lambda)(2 - \lambda) - 6 = \lambda^2 - 3\lambda - 4 = 0 \tag{24}$$

Eigenvalues: $\lambda_1 = 4$, $\lambda_2 = -1$

Eigenvectors: $\mathbf{v}_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$, $\mathbf{v}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

General solution:

$$\mathbf{x}(t) = c_1 e^{4t} \begin{pmatrix} 2 \\ 3 \end{pmatrix} + c_2 e^{-t} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{25}$$

1.0.12 Equilibria and Linearization

Equilibrium points (also called fixed points or critical points) are constant solutions where $\frac{d\mathbf{x}}{dt} = \mathbf{0}$.

For the autonomous system $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$, equilibria satisfy:

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{0} \tag{26}$$

1.0.13 Linear Stability Analysis

Near an equilibrium $\mathbf{x}^*$, we can approximate the nonlinear system by its linearization. Let $\mathbf{u} = \mathbf{x} - \mathbf{x}^*$ be the displacement from equilibrium. Then:

$$\frac{d\mathbf{u}}{dt} = \mathbf{J}(\mathbf{x}^*)\mathbf{u} + \text{higher order terms} \tag{27}$$

where $\mathbf{J}(\mathbf{x}^*) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*}$ is the Jacobian matrix evaluated at the equilibrium.

For small displacements, the linear approximation dominates:

$$\frac{d\mathbf{u}}{dt} \approx \mathbf{J}(\mathbf{x}^*)\mathbf{u} \tag{28}$$

The stability of the equilibrium is determined by the eigenvalues of the Jacobian: - **Stable**: All eigenvalues have negative real parts - **Unstable**: At least one eigenvalue has positive real part - **Neutrally stable**: All eigenvalues have non-positive real parts, with at least one having zero real part

2.4: Pendulum Equilibria The nonlinear pendulum equation is:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0 \tag{29}$$

Converting to a system with $x_1 = \theta$, $x_2 = \frac{d\theta}{dt}$:

$$\frac{dx_1}{dt} = x_2 \tag{30}$$

$$\frac{dx_2}{dt} = -\frac{g}{l}\sin x_1 \tag{31}$$

Equilibria occur at $(n\pi, 0)$ for integer n.

The Jacobian is:

$$\mathbf{J} = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l}\cos x_1 & 0 \end{pmatrix} \tag{32}$$

At $(0,0)$: $\mathbf{J} = \begin{pmatrix} 0 & 1 \\ -g/l & 0 \end{pmatrix}$ with eigenvalues $\pm i\sqrt{g/l}$ (center)

At $(\pi,0)$: $\mathbf{J} = \begin{pmatrix} 0 & 1 \\ g/l & 0 \end{pmatrix}$ with eigenvalues $\pm\sqrt{g/l}$ (saddle)

1.0.14 Applications and Examples

1.0.15 Coupled Oscillators

Two masses connected by springs provide a fundamental example of coupled dynamics:

$$m_1 \frac{d^2 x_1}{dt^2} = -k_1 x_1 - k_2(x_1 - x_2) \tag{33}$$

$$m_2 \frac{d^2 x_2}{dt^2} = -k_3 x_2 - k_2(x_2 - x_1) \tag{34}$$

Converting to first-order form with $\mathbf{x} = (x_1, \dot{x}_1, x_2, \dot{x}_2)^T$:

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1+k_2}{m_1} & 0 & \frac{k_2}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & 0 & -\frac{k_2+k_3}{m_2} & 0 \end{pmatrix} \mathbf{x} \tag{35}$$

The eigenvalues determine the normal mode frequencies of the coupled system.

1.0.16 Epidemiological Models

The SIR (Susceptible-Infected-Recovered) model for disease spread is:

$$\frac{dS}{dt} = -\beta SI \tag{36}$$

$$\frac{dI}{dt} = \beta SI - \gamma I \tag{37}$$

$$\frac{dR}{dt} = \gamma I \tag{38}$$

where S, I, and R are the numbers of susceptible, infected, and recovered individuals, respectively.

Since $S + I + R = N$ (constant total population), this reduces to a two-dimensional system in the (S, I) plane.

1.0.17 Chemical Reaction Networks

Consider the autocatalytic reaction scheme:

$$A + X \to 2X \tag{39}$$

$$X \to B \tag{40}$$

The rate equations are:

$$\frac{d[A]}{dt} = -k_1[A][X] \tag{41}$$

$$\frac{d[X]}{dt} = k_1[A][X] - k_2[X] \tag{42}$$

$$\frac{d[B]}{dt} = k_2[X] \tag{43}$$

If $[A]$ is held constant (large reservoir), this becomes a two-dimensional system that can exhibit bistability or oscillations depending on parameters.

1.0.18 Numerical Methods for Systems

Numerical methods for scalar equations extend naturally to systems. For the system $\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x})$, Euler's method becomes:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h \cdot \mathbf{f}(t_n, \mathbf{x}_n) \tag{44}$$

Similarly, the fourth-order Runge-Kutta method extends to:

$$\mathbf{k}_1 = h\mathbf{f}(t_n, \mathbf{x}_n) \tag{45}$$
$$\mathbf{k}_2 = h\mathbf{f}(t_n + h/2, \mathbf{x}_n + \mathbf{k}_1/2) \tag{46}$$
$$\mathbf{k}_3 = h\mathbf{f}(t_n + h/2, \mathbf{x}_n + \mathbf{k}_2/2) \tag{47}$$
$$\mathbf{k}_4 = h\mathbf{f}(t_n + h, \mathbf{x}_n + \mathbf{k}_3) \tag{48}$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{49}$$

1.0.19 Computational Considerations

When implementing numerical methods for systems, several factors become important:

1. **Computational cost**: Scales with system dimension 2. **Memory requirements**: Storage for multiple vectors 3. **Stability**: Numerical stability can depend on system properties 4. **Conservation laws**: Some systems conserve energy or other quantities

Computational Note: The file `lecture2.py` includes implementations of numerical methods for systems, phase portrait generation, and analysis tools for the examples discussed in this lecture.

This second lecture has extended our understanding from scalar equations to systems of differential equations, opening up the rich world of multidimensional dynamics:

Systems Framework: Any differential equation can be written as a first-order system, providing a unified approach to studying all differential equations. The vector field perspective gives geometric insight into solution behavior.

Phase Space Analysis: The phase space provides a complete description of system behavior. Trajectories, nullclines, and direction fields are powerful tools for understanding dynamics without explicit solutions.

Linear Systems: Constant coefficient linear systems can be solved completely using eigenvalue analysis. The eigenvalues and eigenvectors determine the qualitative behavior near equilibria.

Equilibria and Stability: Equilibrium points and their stability properties organize the global behavior of dynamical systems. Linearization provides local stability information that often determines global behavior.

Applications: Systems of differential equations model coupled phenomena across science and engineering. The examples of oscillators, epidemics, and chemical reactions illustrate the breadth of applications.

Numerical Methods: Computational methods extend naturally to systems, though computational cost and stability considerations become more important in higher dimensions.

The geometric perspective developed in this lecture provides the foundation for understanding more complex phenomena like bifurcations, chaos, and strange attractors that we will explore in later lectures. The interplay between local analysis (near equilibria) and global behavior (phase portraits) is a central theme in dynamical systems theory.

Computational Companion: All examples, phase portraits, and numerical methods discussed in this lecture are implemented in `lecture2.py`. The code includes interactive tools for exploring parameter dependence and visualizing high-dimensional projections.