

Dynamical Systems (ODEs)

A continuous-time dynamical system is defined by a system of n ODEs: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t; \theta)$, where $\mathbf{x} \in \mathbb{R}^n$.

- **Equilibrium Point** \mathbf{x}_e : A state where the system does not change, i.e., $\mathbf{f}(\mathbf{x}_e) = 0$.
- **Linear System**: A system of the form $\dot{\mathbf{x}} = A\mathbf{x}$.

Solving Linear Systems: $\dot{\mathbf{x}} = A\mathbf{x}$

1. Find eigenvalues λ_i by solving the characteristic equation: $\det(A - \lambda I) = 0$.
2. For each eigenvalue λ_i , find the corresponding eigenvector \mathbf{u}_i by solving $(A - \lambda_i I)\mathbf{u}_i = \mathbf{0}$.
3. The general solution is a linear combination of the "straight-line" solutions:

$$\mathbf{x}(t) = \sum_{i=1}^n c_i e^{\lambda_i t} \mathbf{u}_i$$

The constants c_i are determined by the initial conditions $\mathbf{x}(0)$.

Stability of Equilibria

Stability for a linear system’s equilibrium at the origin (or for a nonlinear system’s equilibrium \mathbf{x}_e based on its Jacobian J) is determined by the eigenvalues $\lambda = \alpha \pm i\beta$.

- **Asymptotically Stable**: All eigenvalues have real parts $\alpha < 0$. All nearby solutions converge to the equilibrium.
- **Stable**: All eigenvalues have $\alpha \leq 0$. No real parts are positive, and any with $\alpha = 0$ (purely imaginary) are simple. Nearby solutions stay nearby, but don’t necessarily converge (e.g., centers).
- **Unstable**: At least one eigenvalue has a real part $\alpha > 0$. Most nearby solutions will move away.

Eigenvalues	Type
λ_1, λ_2 real, distinct	
$\lambda_1, \lambda_2 < 0$	Stable Node (All paths head to origin)
$\lambda_1, \lambda_2 > 0$	Unstable Node (All paths leave origin)
$\lambda_1 \cdot \lambda_2 < 0$	Saddle (Unstable)
$\lambda_1 = \lambda_2$ real, repeated	Node (Stable if $\lambda < 0$, Unstable if $\lambda > 0$)
$\lambda = \alpha \pm i\beta$ (where $\beta \neq 0$)	
$\alpha < 0$	Stable Spiral (Spiral into the origin)
$\alpha > 0$	Unstable Spiral (Spiral away from the origin)
$\alpha = 0$	Center (Neutrally Stable, paths are closed orbits)

Nonlinear Systems: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$

- **Linearization**: To analyze the stability of an equilibrium point \mathbf{x}_e , we linearize the system using the **Jacobian matrix** $J = J_{\mathbf{f}}(\mathbf{x}_e)$, where $J_{ij} = \frac{\partial f_i}{\partial x_j}$.
- We then analyze the stability of the linear system $\dot{\mathbf{u}} = J\mathbf{u}$, where $\mathbf{u} = \mathbf{x} - \mathbf{x}_e$.
- **Hartman-Grobman Theorem**: The stability of the linearized system (based on J ’s eigenvalues) usually determines the stability of the nonlinear system, except for marginal cases (like $\alpha = 0$ centers).

Other 2D+ Concepts

- **Nullclines**: Curves in the phase space where one component of the vector field is zero (e.g., $\dot{x} = 0$ or $\dot{y} = 0$). Equilibrium points occur at the intersections of *all* nullclines.
- **Limit Cycle**: A stable, isolated, closed-orbit trajectory. Solutions nearby may spiral into or away from a limit cycle.
- **Poincaré-Bendixson Theorem**: In a 2D continuous system, the only possible long-term behaviors are approaching an equilibrium point, approaching a limit cycle, or diverging to infinity. This theorem implies that **chaos cannot occur in 2D continuous systems**.
- **Chaos**: Aperiodic, long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions. Associated with **Strange Attractors** (e.g., Lorenz Attractor) and requires **3 or more dimensions** for continuous systems.

Iterated Maps (Discrete-Time Systems)

A discrete-time system: $x_{n+1} = f(x_n)$. The sequence x_0, x_1, x_2, \dots is the **orbit**.

- **Fixed Point** x^* : A point that maps to itself, $f(x^*) = x^*$.
- **Stability of Fixed Points**: Determined by the multiplier $\lambda = f'(x^*)$.
 - $|\lambda| < 1$: **Stable** (attracting). Orbits starting near x^* converge to it.
 - $|\lambda| > 1$: **Unstable** (repelling). Orbits starting near x^* (but not exactly on it) move away.
 - $|\lambda| = 1$: **Marginal case** (e.g., $f'(x^*) = -1$ leads to a bifurcation).
- **Cobweb Plot**: A graphical method to visualize orbits. Draw a line from (x_n, x_n) to $(x_n, f(x_n))$, then horizontally to $(f(x_n), f(x_n)) = (x_{n+1}, x_{n+1})$, and repeat.

Logistic Map

The canonical example of a route to chaos: $x_{n+1} = rx_n(1 - x_n)$.

- As the parameter r increases (from 0 to 4), the system’s long-term behavior changes.
- It moves from a single stable fixed point, to a stable 2-cycle, then a 4-cycle, 8-cycle, etc. This is the **period-doubling bifurcation** route to chaos.
- Past a certain r , the system becomes chaotic, with regions of stability ("islands") interspersed.

Lyapunov Exponent

Measures the average exponential rate of divergence or convergence of nearby orbits, quantifying sensitivity to initial conditions.

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \ln |f'(x_k)|$$

- $\lambda > 0 \implies$ **Chaos**. Nearby orbits diverge exponentially.
- $\lambda < 0 \implies$ **Stable**. Nearby orbits converge.
- $\lambda = 0 \implies$ **Marginal**.

Cellular Automata (CA)

Discrete-time, discrete-space, discrete-state systems where cells update their state based on a **local rule** applied to their **neighborhood**.

- **Neighborhood**: Set of cells that influence a cell’s next state.

- *Von Neumann*: 4 neighbors (N, S, E, W).

- *Moore*: 8 neighbors (all adjacent cells).

- **Update Schedule**:

- *Synchronous*: All cells update simultaneously.

- *Asynchronous*: Cells update one at a time (e.g., in a random order).

- **Combinatorics**: For k states and a neighborhood of size $|N|$, the total number of possible rules is $k^{(|N|)}$.

- **Wolfram’s 1D CA**: $k = 2, |N| = 3$ (cell + left/right neighbors) $\implies 2^{(2^3)} = 256$ possible rules.

Wolfram’s 4 Classes of Behavior

1. **Class 1 (Stable)**: Evolves to a stable, homogeneous state (e.g., all black or all white).
2. **Class 2 (Periodic)**: Evolves to simple periodic structures (e.g., stable patterns or simple oscillators).
3. **Class 3 (Chaotic)**: Exhibits chaotic, aperiodic, fractal-like patterns (e.g., Rule 30).
4. **Class 4 (Complex)**: Exhibits complex, localized structures ("gliders") that move and interact. Can support computation (e.g., Rule 110 is Turing complete).

Conway’s Game of Life (2D CA)

A famous 2D CA with a Moore neighborhood and 2 states (live/dead).

- **Survival**: A live cell with exactly 2 or 3 live neighbors survives.
- **Death**: A live cell with < 2 neighbors (loneliness) or > 3 neighbors (overcrowding) dies.
- **Reproduction**: A dead cell with exactly 3 live neighbors becomes alive.

Networks

Networks (graphs) consist of nodes (vertices) and edges (links).

Key Properties of Real Networks

- **Small-World**: Low average path length h . The average distance between any two nodes is short, typically $h \sim O(\log N)$.
- **High Clustering**: High clustering coefficient C . A node’s neighbors are also likely to be neighbors of each other.
- **Scale-Free**: The degree distribution p_k (probability a node has k links) follows a power-law, $p_k \sim k^{-\gamma}$. This implies a few "hubs" with many links and many nodes with few links.

Network Models

Model	Scale-Free?	Small-World?	High C?
Erdos-Renyi (ER)	No	Yes	No
Watts-Strogatz (WS)	No	Yes	Yes
Barabasi-Albert (BA)	Yes	Yes	Kinda

- **Erdos-Renyi (ER):** A random graph. Start with N nodes, connect every pair with probability p . Degree distribution p_k is Poisson (peaked).
- **Watts-Strogatz (WS):** The "small-world" model. Start with a regular ring lattice (high C , high h), and "rewire" each edge with probability p . This quickly lowers h while retaining high C .
- **Barabasi-Albert (BA):** The "scale-free" model. Built via two mechanisms:
 1. **Growth:** Start with a small network, add one node at a time.
 2. **Preferential Attachment:** New nodes prefer to link to existing nodes that already have a high degree ("rich get richer").

Centrality Measures

Ways to quantify a node's "importance" in a network.

- **Degree:** Number of connections. Simple count of a node's "popularity".
- **Betweenness:** Fraction of all shortest paths in the network that pass through this node. Identifies "bridges" or "bottlenecks".
- **Closeness:** Inverse of the average shortest-path distance to all other nodes. Measures how "central" a node is or how fast it can reach everyone.
- **Eigenvector:** A node's importance is determined by the importance of its neighbors. A node is important if it's connected to other important nodes. Solved by finding the principal eigenvector of the adjacency matrix A : $A\mathbf{c} = \lambda\mathbf{c}$.
- **Alpha-Centrality:** A generalization that includes a node's intrinsic importance: $\mathbf{c} = \beta(I - \alpha A)^{-1}\mathbf{e}$.

Optimization

Gradient Descent (GD)

An iterative algorithm to find a local minimum of a function $f(\mathbf{w})$ by repeatedly moving in the direction of the negative gradient, $-\nabla f(\mathbf{w})$.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \nabla f(\mathbf{w}_k)$$

Where α is the **learning rate**.

- **Batch GD:** Calculate ∇f using the entire dataset. Very accurate gradient, but computationally slow for large datasets.
- **Stochastic GD (SGD):** Calculate ∇f using only *one* data point. Very fast updates, but the gradient is noisy, leading to a volatile convergence path.
- **Mini-batch SGD:** A compromise. Calculate ∇f using a small batch of data. Offers a balance between the stability of Batch GD and the speed of SGD.

Adam Optimizer: An advanced optimization algorithm popular for deep learning. It adaptively adjusts the learning rate for each parameter, combining the ideas of **Momentum** (using a moving average of the 1st moment/gradient) and **RMSprop** (using a moving average of the 2nd moment/squared gradient).

Local Search

An iterative improvement heuristic. Start with a candidate solution and repeatedly move to a "neighboring" solution if it is better.

- **Neighborhood:** The set of solutions accessible from the current solution by a small change.
- **Example: TSP Neighborhoods:**
 - **2-opt:** Remove two edges from the tour, and reconnect the four resulting endpoints in the only other possible way (which "uncrosses" the paths).
 - **3-opt:** Remove three edges, reconnect in a way that improves the tour.

Particle Swarm Optimization (PSO)

A population-based stochastic optimization algorithm inspired by social behavior (e.g., bird flocking). Used for black-box optimization.

- A "swarm" of particles (candidate solutions) "fly" through the search space.
- Each particle i has a position \mathbf{x}_i (its solution) and a velocity \mathbf{v}_i .
- Each particle remembers its **personal best** position found so far: $pbest_i$.
- The swarm tracks the **global best** position found by *any* particle: $gbest$.

Core Update Equations

$$\begin{aligned} \mathbf{v}_i(t+1) &= \underbrace{\omega \mathbf{v}_i(t)}_{\text{Inertia}} + \underbrace{c_1 r_1 (pbest_i - \mathbf{x}_i(t))}_{\text{Cognitive/Personal}} \\ &\quad + \underbrace{c_2 r_2 (gbest - \mathbf{x}_i(t))}_{\text{Social}} \\ \mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \end{aligned}$$

- ω : **Inertia weight.** Balances exploration (high ω) and exploitation (low ω).
- c_1, c_2 : **Acceleration coefficients.** Control the "pull" towards the personal best (c_1 , cognitive) and global best (c_2 , social).
- r_1, r_2 : Random numbers in $[0, 1]$ to add stochasticity.

Topologies

Defines how information (the $gbest$) is shared among particles.

- **gbest (Global Best):** All particles are connected. The $gbest$ is the best of the entire swarm. Converges very fast, but can get stuck in local optima.
- **lbest (Local Best):** Particles are in a smaller neighborhood (e.g., a ring). The $gbest$ in the equation is replaced with the $lbest$ (best in the neighborhood). Slower convergence, but more robust to local optima.

Multi-Robot Task Allocation (MRTA)

The problem of assigning a set of tasks T to a set of robots R to optimize a collective objective (e.g., minimize time, maximize utility).

MRTA Taxonomy

Problems are classified by:

- **ST/MT:** **Single-Task** / **Multi-Task** robots (robots can handle one vs. many tasks at a time).
- **SR/MR:** **Single-Robot** / **Multi-Robot** tasks (tasks require one vs. a team of robots).
- **IA/TA:** **Instantaneous Assignment** / **Time-Extended Assignment** (tasks are just assigned vs. tasks involve durations and travel, requiring scheduling/routing).

Mapping MRTA to Optimization Models

MRTA Type	Optimization
ST-SR-IA (One robot per task, one task per robot)	LAP (Linear Assignment Problem)
MT-SR-IA (Robots can take multiple tasks)	GAP (Generalized Assignment Problem)
ST-SR-TA (One robot per task, includes routing)	mTSP (Multi-Robot Traveling Salesman Problem)
MT-SR-TA (Robots take multiple tasks, includes routing)	VRP (Vehicle Routing Problem)

Set-Based Formulations (for MR tasks)

Used for coalition formation, where tasks require multiple robots (MR).

- **Set Covering (MT-MR-IA):** Find the minimum cost set of coalitions (subsets of robots) such that *every task is covered at least once*. Models MT-MR-IA, as robots can be in multiple coalitions.

$$\min \sum c_j x_j \quad \text{s.t.} \quad \sum a_{ij} x_j \geq 1$$

- **Set Packing (ST-MR-IA):** Find the maximum profit set of coalitions such that *each task is covered at most once*. Models ST-MR-IA, where task allocation must be exclusive.

$$\max \sum p_j x_j \quad \text{s.t.} \quad \sum a_{ij} x_j \leq 1$$

- **Set Partitioning:** Find the coalitions such that *every task is covered exactly once*. This is a very common base for routing problems.