Mapping: source → code entry

- **Butterfly effect** → Lorenz's work: 1963 "Deterministic Nonperiodic Flow"; 1972 talk that coined the phrasing. Code: `lorenz63(...)`. yorke.umd.edu
- **Competing Species** (competitive Lotka–Volterra) → early Lotka/Volterra lineage; modern ecology texts cover the logistic-competition form. Code: `competitive_lv(...)`. resources.saylor.org
- **Diff Eqn Sketchpad (app)** → "Introduction to Differential Equations with Geometer's Sketchpad." Code analogs: slope/phase via `vector_field_2d(...)`, `euler(...)`, `rk4(...)`. Reddit+1
- **Duffing** → Nonlinear oscillator introduced by Georg Duffing (1918). Code: `duffing(...)`. Bluffton University
- **Euler's Method** → Euler's 18th-century numerical scheme. Code: `euler(...)`. Bluffton University
- **Euler's Method for Systems** → same idea, vector-valued. Code: `euler(...)` on multi-dimensional `y`. Bluffton University
- **Graphing Solutions** → standard slope/solution-curve pedagogy. Code analogs: `vector_field_2d(...)` + RK4/Euler. ximera.osu.edu
- **HMS Glider** → interpreted as glider flight dynamics (lift/drag, flight-path angle). Code: `glider2d(GliderParams(...))`. grc.nasa.gov+1
- **HPG Solver** → Boston Univ. EML/ODE software lineage ("HPGSolver"). Code analogs: `rk4(...)` + our RHS models. uis.brage.unit.no
- **HPG System Solver** → BU EML "HPGSystemSolver" (systems). Code analogs: `rk4(...)` + `linear_system(A)` / any RHS. resources.saylor.org
- **Linear Phase Portraits** → MIT Mathlets (trace–determinant framing). Code: `classify_linear_2x2(A)` + `vector_field_2d(...)`. Taylor & Francis Online
- **Lorenz Equations** → Lorenz (1963). Code: `lorenz63(...)`.
- **Mass Spring** (SHO / damped) → canonical 2nd-order ODE. Code: `mass_spring_damper(...)`. Pauls Online Math Notes
- **Matrix Fields** → matrix-driven linear systems / vector fields; see MIT Mathlets. Code: `linear_system(A)` + `vector_field_2d(...)`. Taylor & Francis Online+1
- **Oscillating Chemical Reaction** → Oregonator model of the BZ reaction (Field–Körös–Noyes). Code: `oregonator(...)`. The Aperiodical
- **Pendulum Sensitive Dependence** → chaotic **double pendulum** demonstrations. Code: `double_pendulum(...)`. yorke.umd.edu+1

- **Pendulums** → simple (possibly driven/damped). Code: `pendulum_simple(...)`. (General reference covered by standard mechanics texts.)
- **Phase Lines** → 1-D qualitative analysis. Code helper: `phase_line_1d(fy, y_min, y_max)`.
- **Predator Prey** → Lotka–Volterra predator–prey. Code: `lotka_volterra_pred_prey(...)`. resources.saylor.org
- **RLC Circuits** → standard series RLC ODE. Code: `rlc_series(R,L,C,E)`. SpringerLink
- **TD Plane Animation** → trace–determinant (node/saddle/spiral). Code: `classify_linear_2x2(A)` (plug into your own animator). resources.saylor.org
- **TD Plane Quiz** → teaching asset built on TD plane. Code: same classifier above. Groningen Research Portal
- **Target Practice** → shooting to hit a boundary/target. Code: `shoot_for_target(...)` (secant-based “aim the IC” helper).
- **Vander Pol** → van der Pol’s relaxation oscillator (1920s). Code: `vanderpol(mu=...)`. Mathemanu

Notes on fidelity + simulation

- **Steppers**: Fixed-step RK4 is included for deterministic reproducibility; Euler is there for pedagogy. If you want adaptive control (Dormand–Prince), I can wire a tiny stepsize controller next.
- **Stiffness**: Oregonator and large-μ Van der Pol can get stiff; if you see step-size thrash, we'll switch to an A-stable routine.
- **Glider**: The provided point-mass model is intentionally minimal (CL(α) slope, parabolic drag); plug in your aeromodel to upscale fidelity. https://www.odu.edu/computer-science+1
- **TD plane**: `classify_linear_2x2(A)` returns trace, determinant, discriminant, and a qualitative type; couple it to `vector_field_2d(...)` for instant portraits.