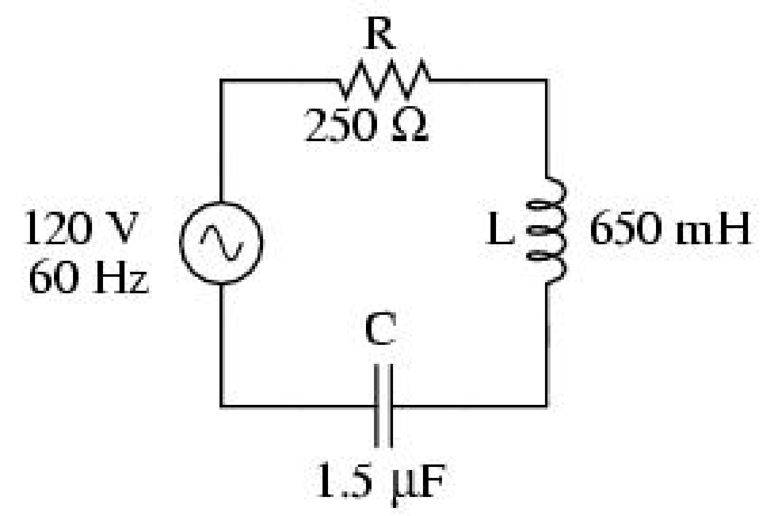
Ordinary Differential Equations

COS 323

Ordinary Differential Equations (ODEs)

- One independent variable; (PDEs have more)
- Differential equations are ubiquitous, the lingua franca of the sciences; many different fields are linked by having similar differential equations
  - electrical circuits
  - Newtonian mechanics
  - chemical reactions
  - population dynamics
  - economics… and so on, ad infinitum

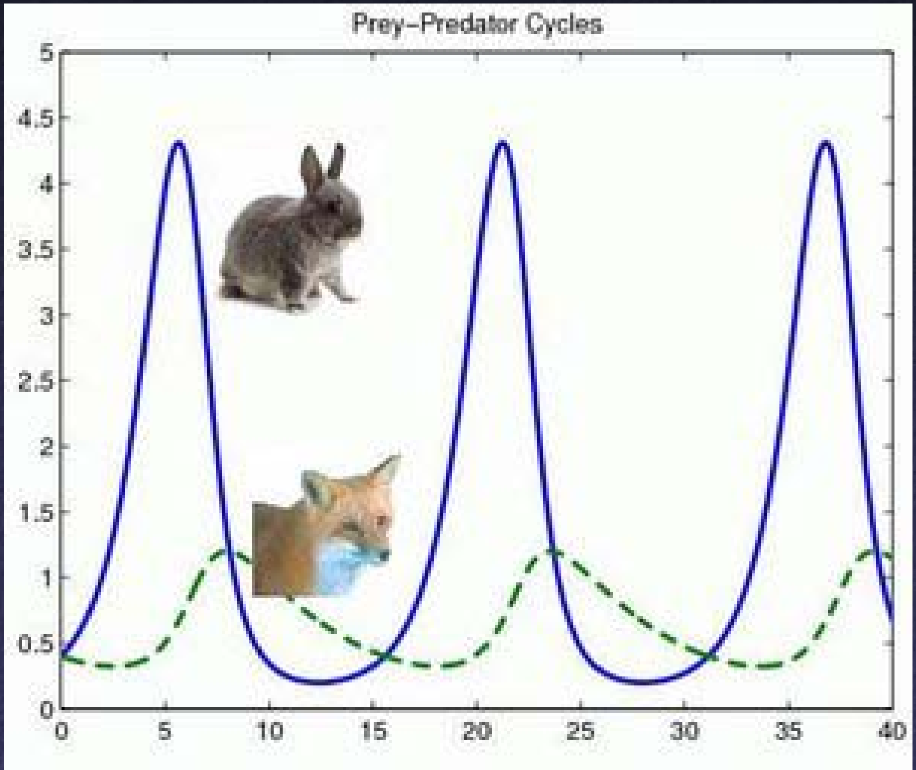
Example: RLC circuit



$$V = RI + L\frac{dI}{dt} + \frac{1}{C}\int I\,dt$$

$$\frac{d^2q}{dt^2} + \frac{R}{L}\frac{dq}{dt} + \frac{1}{LC}q = \frac{V}{L}$$

Example: Population Dynamics



- 1798 Malthusian catastrophe
- 1838 Verhulst, logistic growth
- Predator-prey systems, Volterra-Lotka

Population Dynamics

- Malthus:

$$\frac{dN}{dt} = rN \quad \rightarrow \quad N = N_0 e^{rt}$$

Yikes! Population explosion!

- Verhulst:

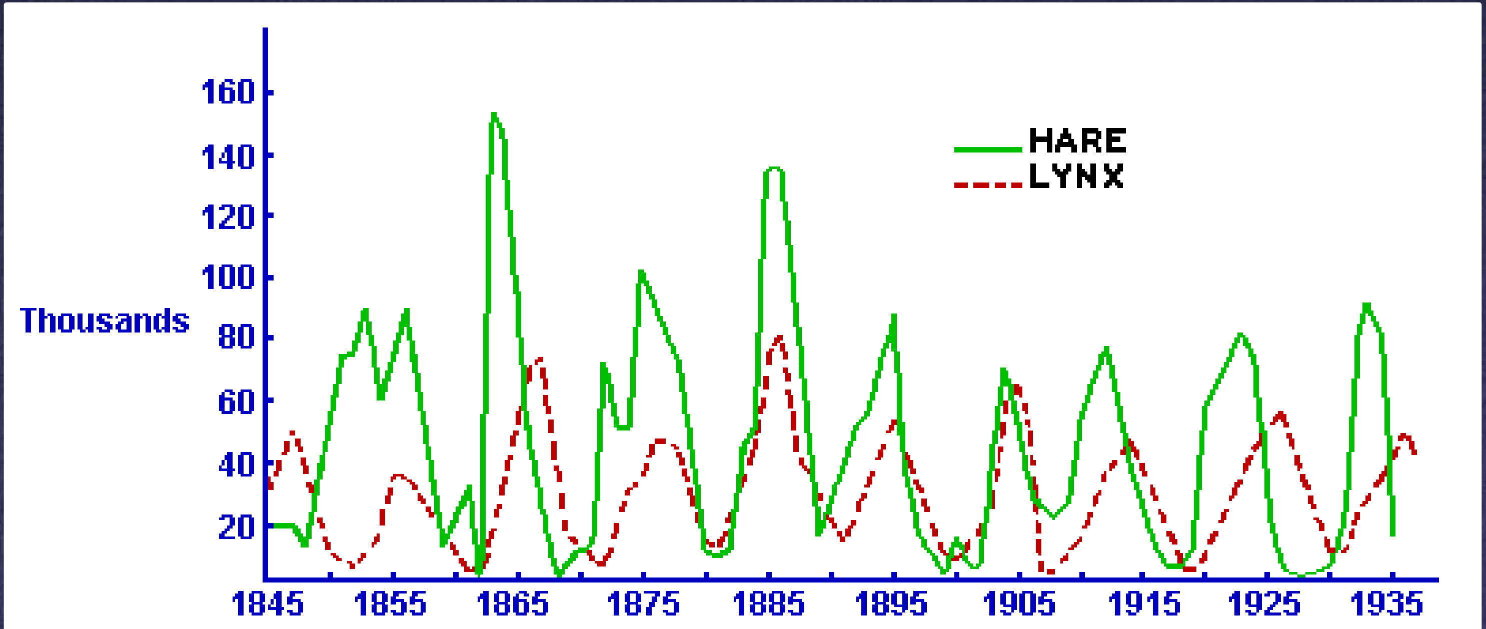
  Logistic growth

$$\frac{dN}{dt} = rN\left(1 - \tfrac{N}{K}\right) \quad \rightarrow \quad N = \frac{N_0 e^{rt}}{1 + \frac{N_0}{K}\left(e^{rt} - 1\right)}$$

Self-limiting

Predator-Prey Population Dynamics

Hudson Bay Company



Predator-Prey Population Dymanics

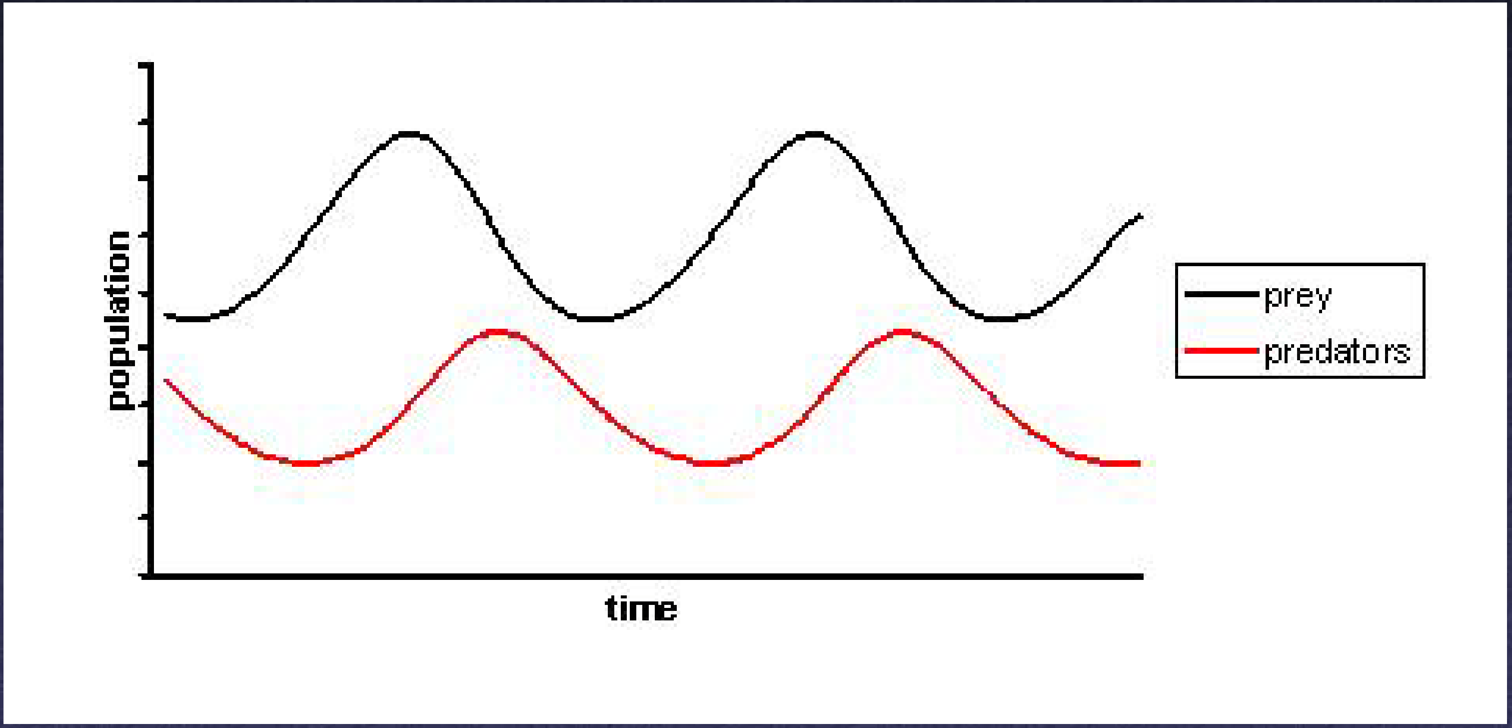
V .Volterra, commercial fishing in the Adriatic

x_1 = biomass of predators (sharks)

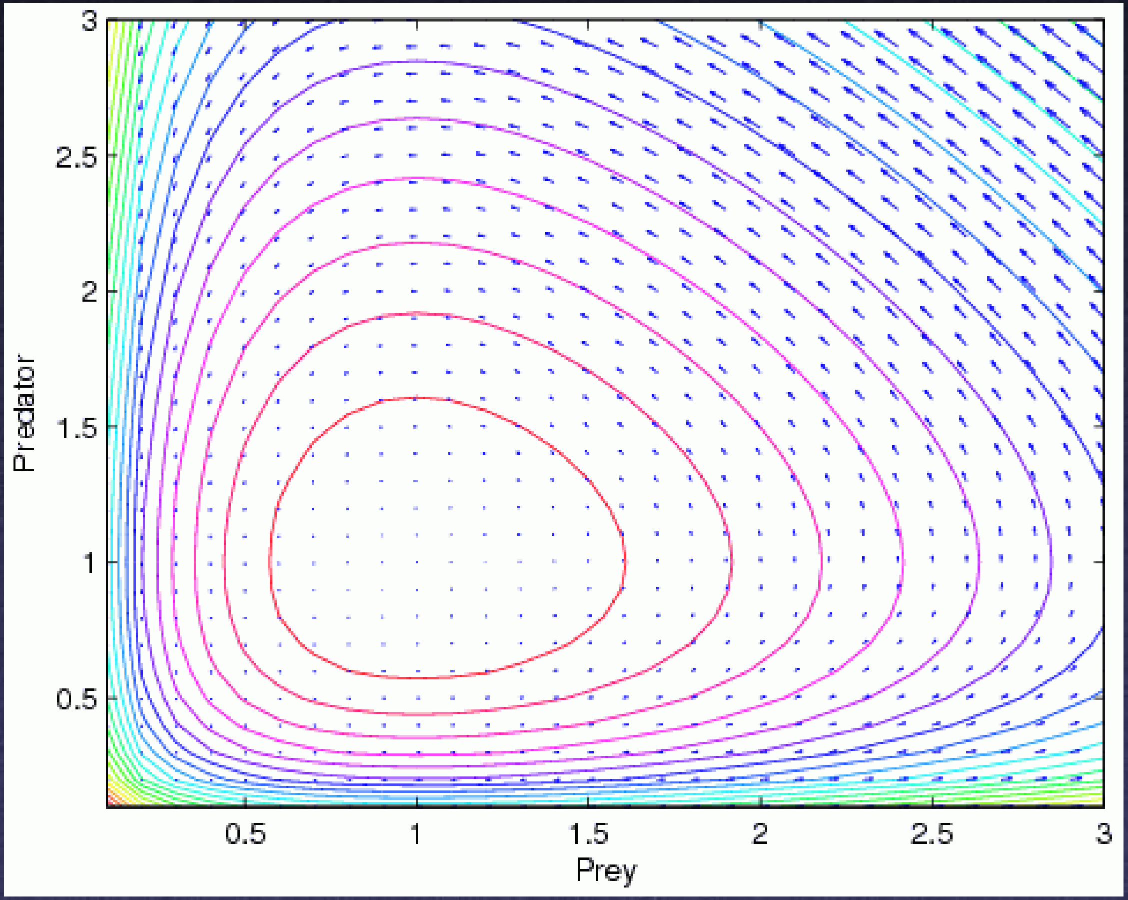
x_2 = biomass of prey (fish)

$$\frac{\dot{x}_1}{x_1} = b_{12}x_2 - a_1 \qquad \frac{\dot{x}_2}{x_2} = a_2 - b_{21}x_1$$

As Functions of Time



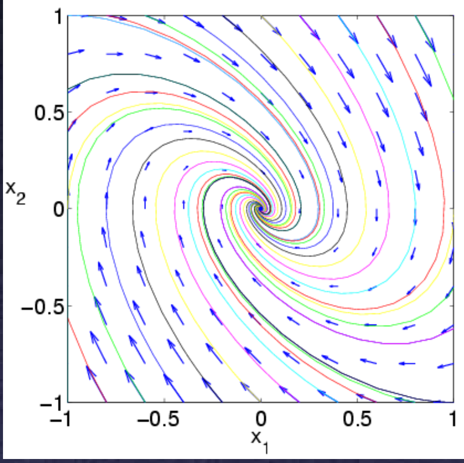
State-Space Diagram: The x_1-x_2 Plane



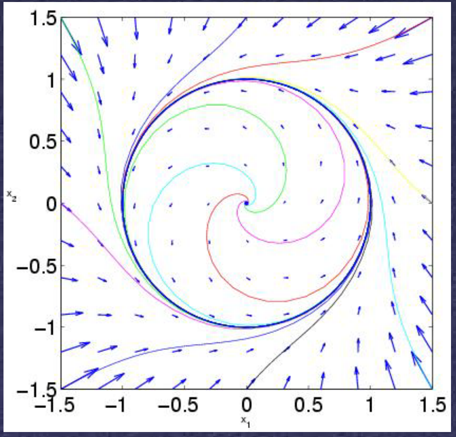
More Behaviors

Self-limiting term → stable focus

$$\frac{\dot{x}_1}{x_1} = b_{12}x_2 - a_1 \qquad \frac{\dot{x}_2}{x_2} = a_2 - b_{21}x_1 - c_{22}x_2$$



Delay → limit cycle



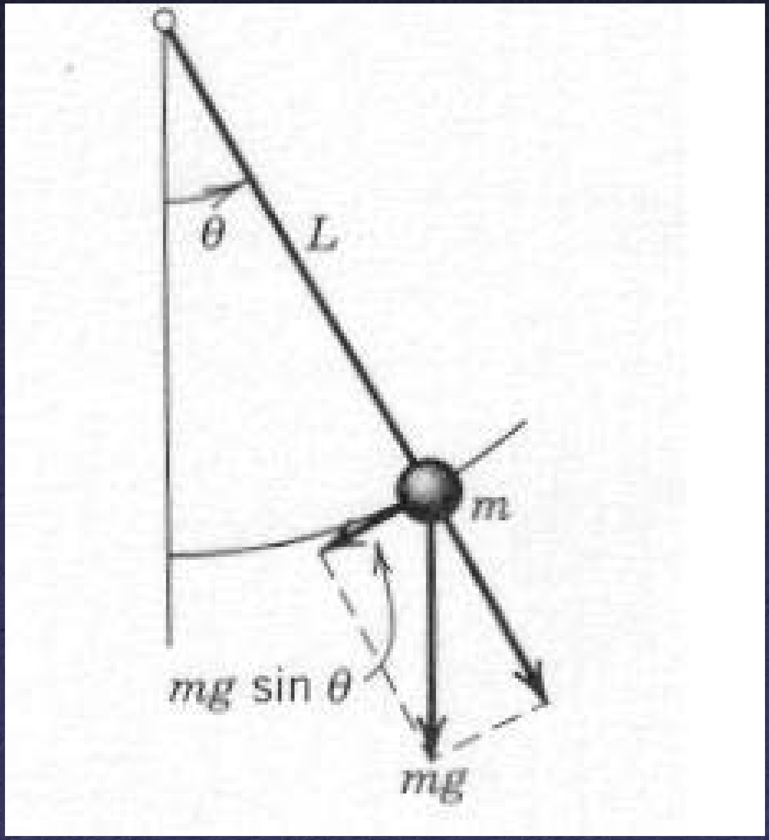
Putting Equations in State-Space Form

- Basic form: d**x**/dt = g(**x**), where **x** is vector-valued
  - Can introduce extra dimensions (variables) to eliminate higher-order derivatives, dependence of g on t
- Example: $\ddot{y} + \alpha\dot{y} + \beta y = f(t)$

$$\text{Introduce}: \begin{array}{ccc} x_1 & \sim & y \\ x_2 & \sim & \dot{y} \\ x_3 & \sim & t \end{array} \qquad \text{Then}: \begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x_3) - \alpha x_2 - \beta x_1 \\ \dot{x}_3 = 1 \end{array}$$

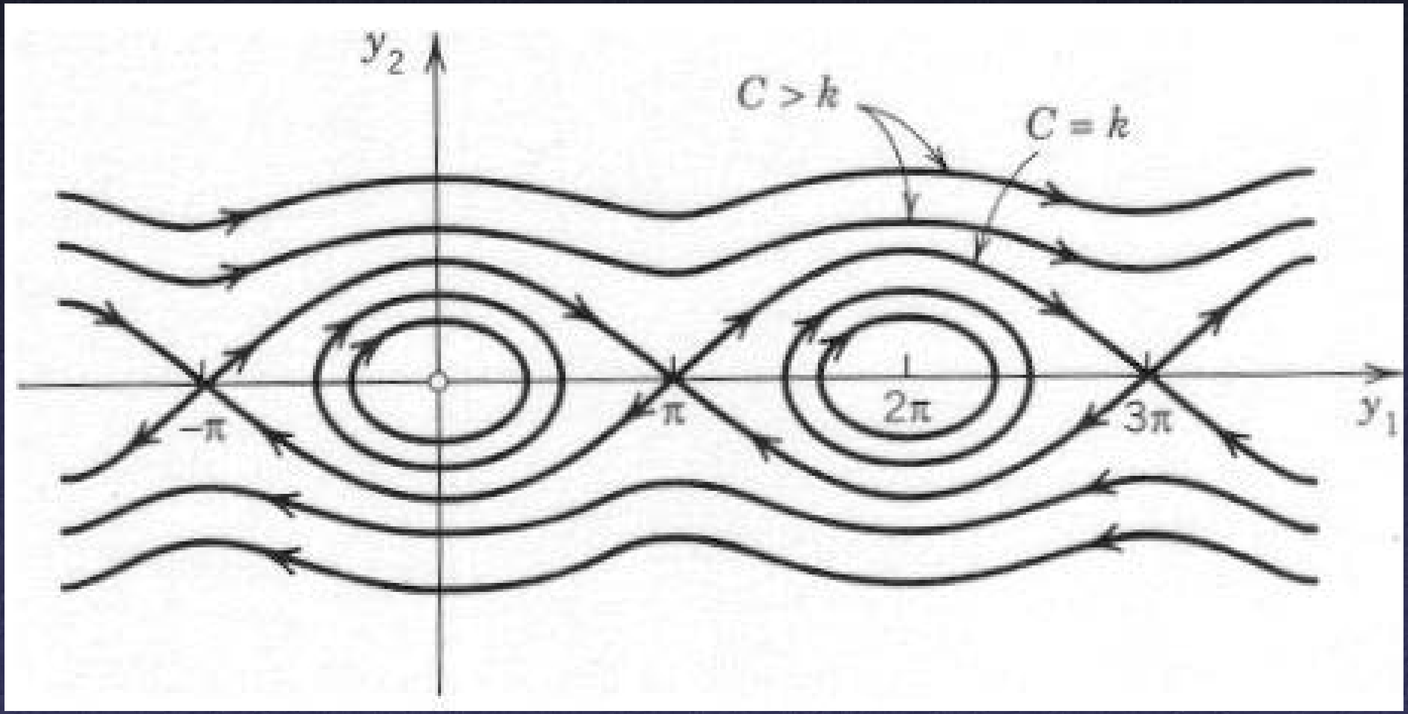
State Space

Traditional example: the (nonlinear) pendulum



$$\ddot{\theta} + (g / l)\sin\theta = 0$$

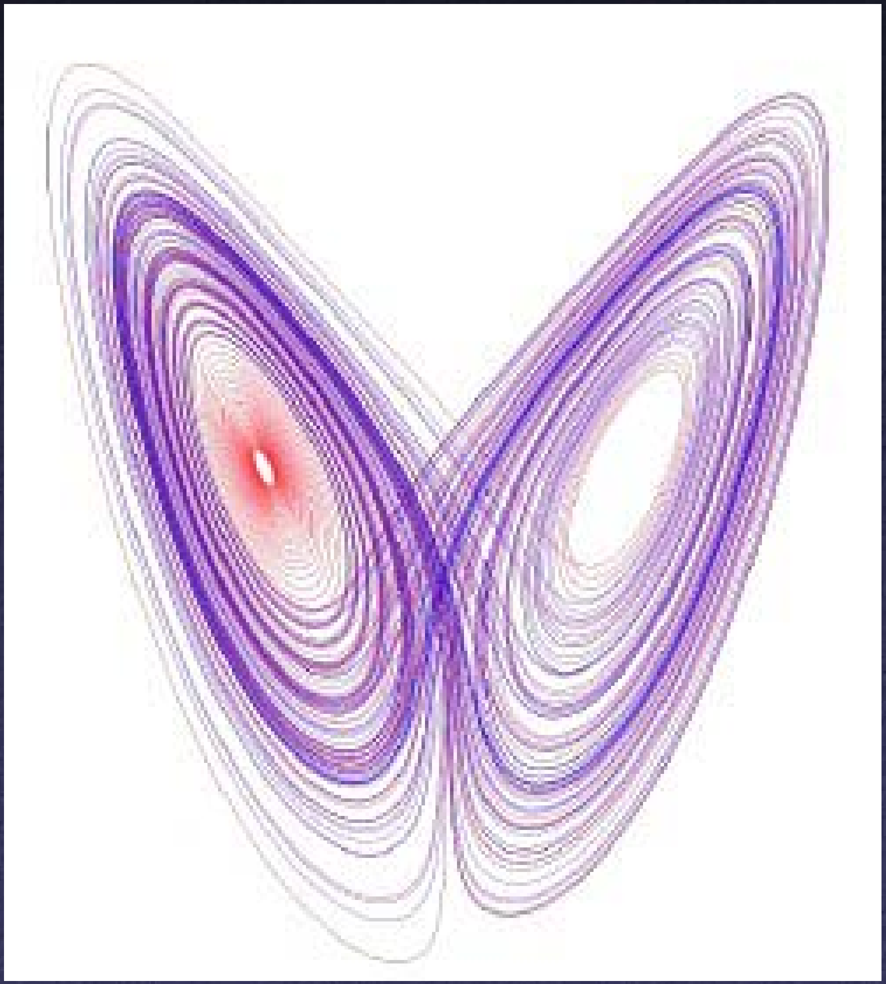
Pendulum in the Phase Plane



Varieties of Behavior

- Stable focus
- Periodic
- Limit cycle

Varieties of Behavior



- Stable focus
- Periodic
- Limit cycle
- Chaos

Numerical Evaluation of ODEs

- Today considering only Initial Value Problems (vs. Boundary Value Problems)
- Euler's method: simple-minded, basis of many others
- Runge-Kutta (usually 4th-order): faster convergence
- Richardson extrapolation: fast, robust, can add other tricks

Criteria for Evaluating

- Accuracy: use Taylor series, big-Oh, classical numerical analysis
- Efficiency: running time may be hard to predict, sometimes step size is adaptive
- Stability: some methods diverge on some problems

Forward (Explicit) Euler

$$\dot{x} = g(x)$$

$$x^{(k+1)} = x^{(k)} + g(x^{(k)})\Delta t \qquad (h = \Delta t)$$

- Local error = O(h^2)
- Global (accumulated) error = O(h)

- Limitation on step size: consider on $\dot{x} = -\lambda x$
  - Unstable for h > 1/λ

Towards Higher Order

- Midpoint method

$$a = hg(x^{(k)})$$
$$b = hg(x^{(k)} + a/2)$$
$$x^{(k+1)} = x^{(k)} + b + O(h^3)$$

- 4th-order Runge Kutta

$$a = hg(x^{(k)})$$
$$b = hg(x^{(k)} + a/2)$$
$$c = hg(x^{(k)} + b/2)$$
$$d = hg(x^{(k)} + c)$$
$$x^{(k+1)} = x^{(k)} + \tfrac{1}{6}(a + 2b + 2c + d)$$
$$+ O(h^5)$$

Extrapolation

- Richardson: compute for several values of h, combine to cancel error: higher-order method
  - As with integration, yields some "classical" algorithms: Euler + Richardson $\rightarrow$ Runge Kutta
- Burlisch-Stoer: fit function (polynomial or rational) to approximation as a function of h; extrapolate to $h=0$

Backward (Implicit) Euler

$$x^{(k+1)} = x^{(k)} + g(x^{(k+1)})h$$

- Local error still $O(h^2)$
- Stable for large step size! (At least on $\dot{x} = -\lambda x$)
- In general, requires nonlinear root finding
- Implicit and semi-implicit methods for higher orders

Accuracy and Stability

- Implicit methods important for "stiff" systems: explicit methods would need small h only for stability, not accuracy

$$\dot{x} = \begin{pmatrix} -1 & 0 \\ 0 & -100 \end{pmatrix} x, \text{ where } x_0 = \begin{pmatrix} 1 \\ 0.0001 \end{pmatrix}$$