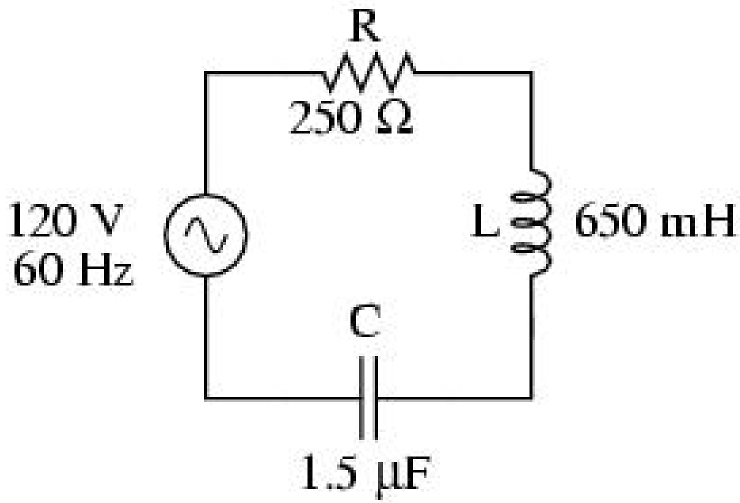
Ordinary Differential Equations
Part 1

COS 323

Ordinary Differential Equations (ODEs)

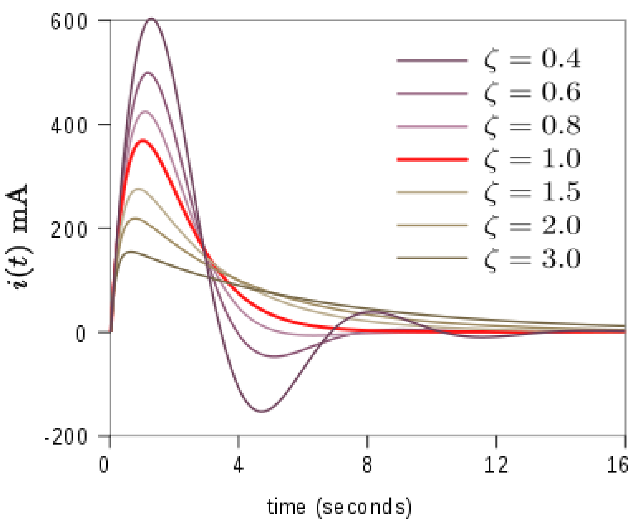
- Differential equations are ubiquitous: the lingua franca of the sciences. Many different fields are linked by having similar differential equations
  - electrical circuits
  - Newtonian mechanics
  - chemical reactions
  - population dynamics
  - economics… and so on, ad infinitum
- ODEs: 1 independent variable (PDEs have more)

ODE Example: RLC circuit

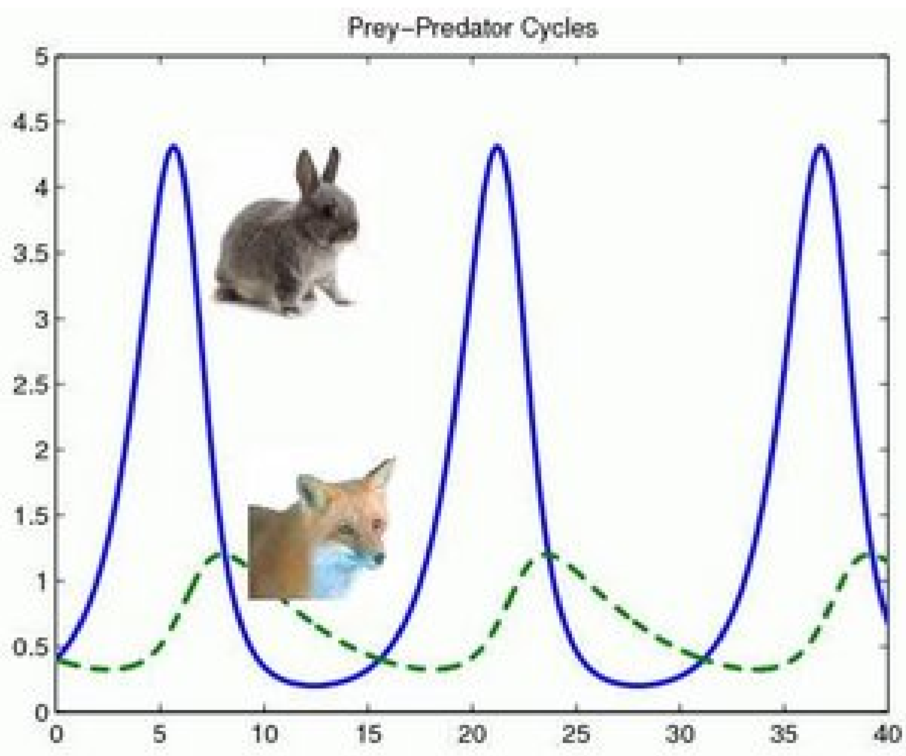


$$V = RI + L\frac{dI}{dt} + \frac{1}{C}\int I\,dt$$

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



ODE Example: Population Dynamics

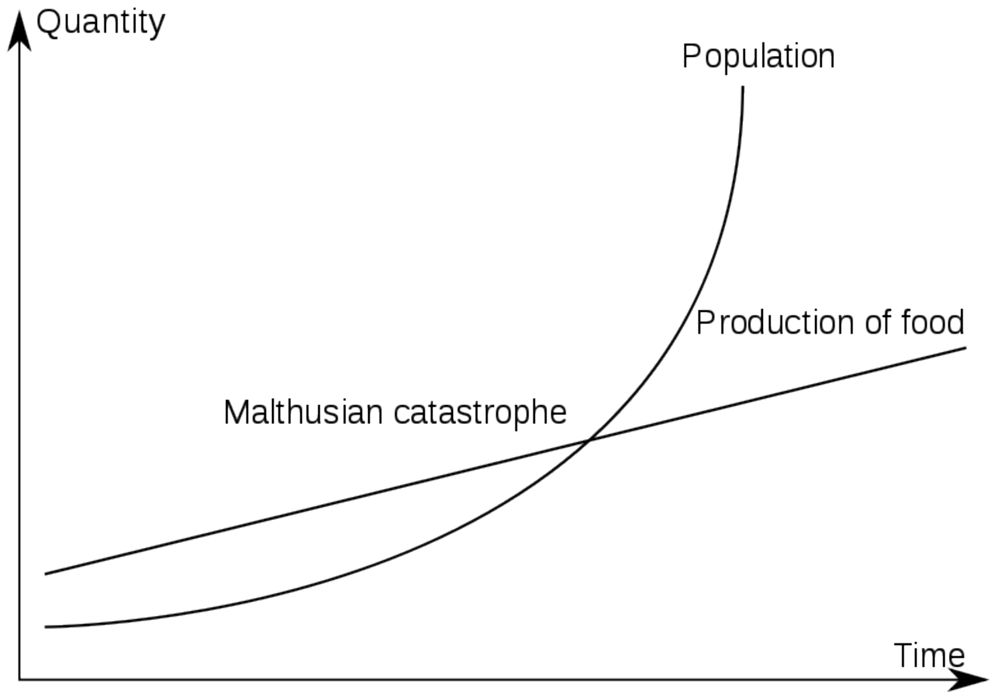


- 1798 Malthusian catastrophe
- 1838 Verhulst, logistic growth
- Predator-prey systems, Volterra-Lotka

Malthusian Population Dynamics

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

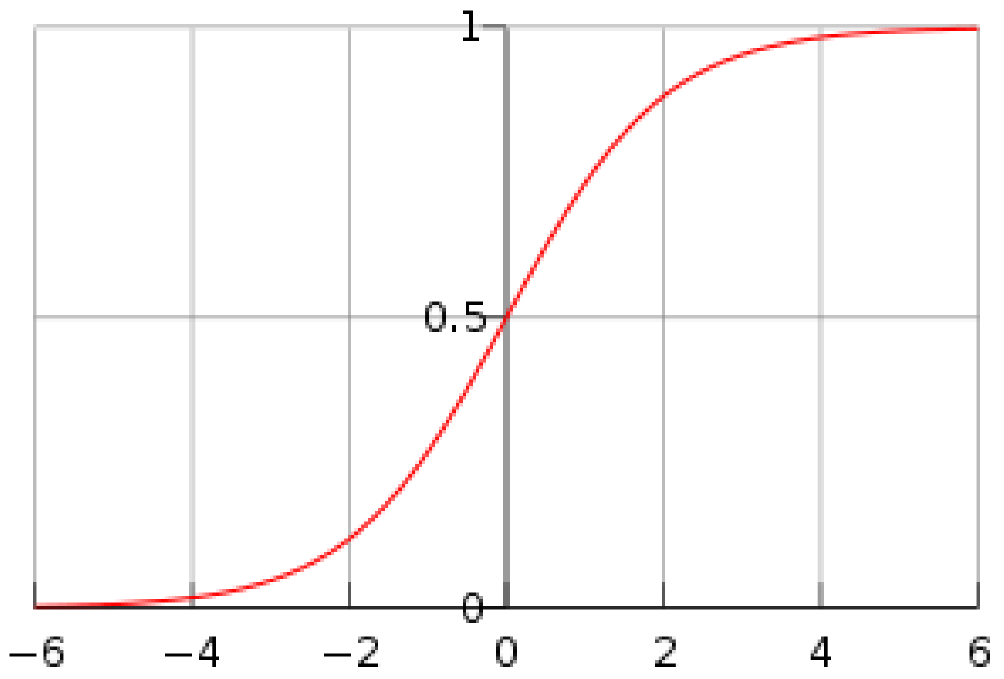
Yikes! Population explosion!



Verhulst: Logistic growth

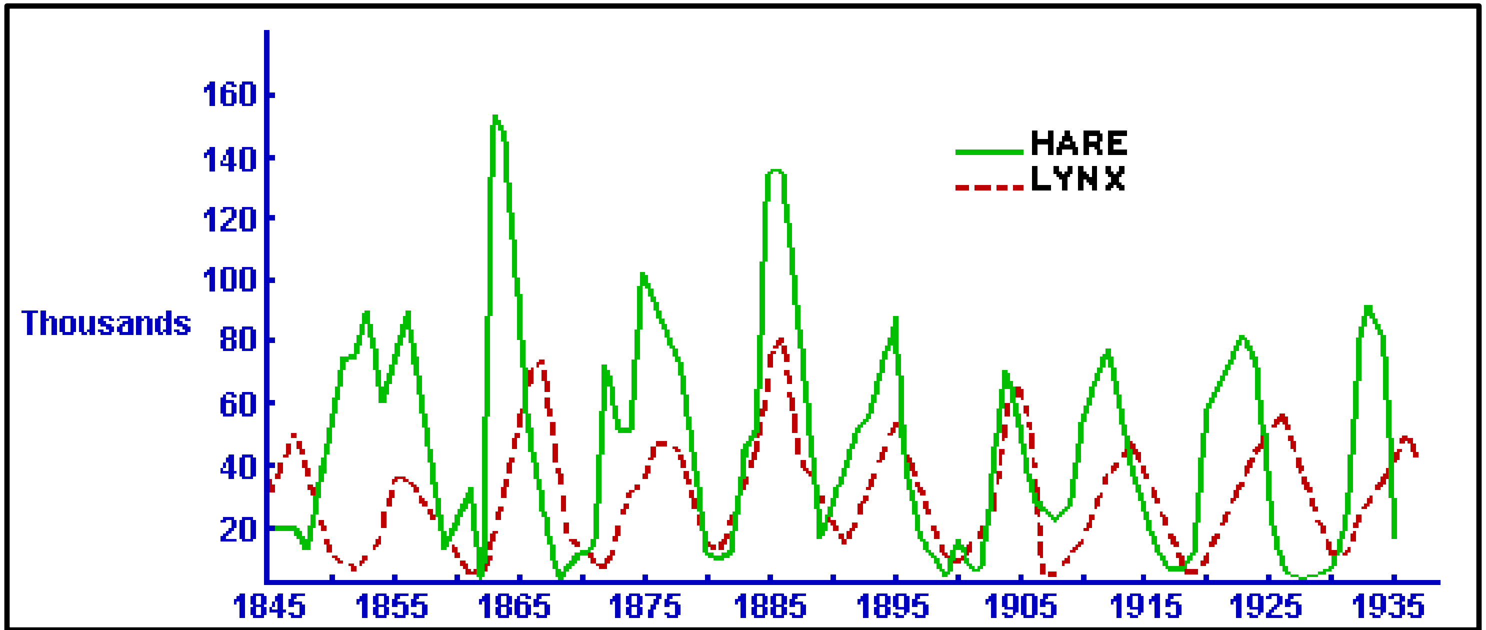
$$\frac{dN}{dt} = rN\left(1 - \tfrac{N}{K}\right) \quad \rightarrow \quad N = \frac{N_0 e^{rt}}{1 + \tfrac{N_0}{K}\left(e^{rt} - 1\right)}$$

Self-limiting



Predator-Prey Population Dynamics

Hudson Bay Company



Predator-Prey Population Dymanics

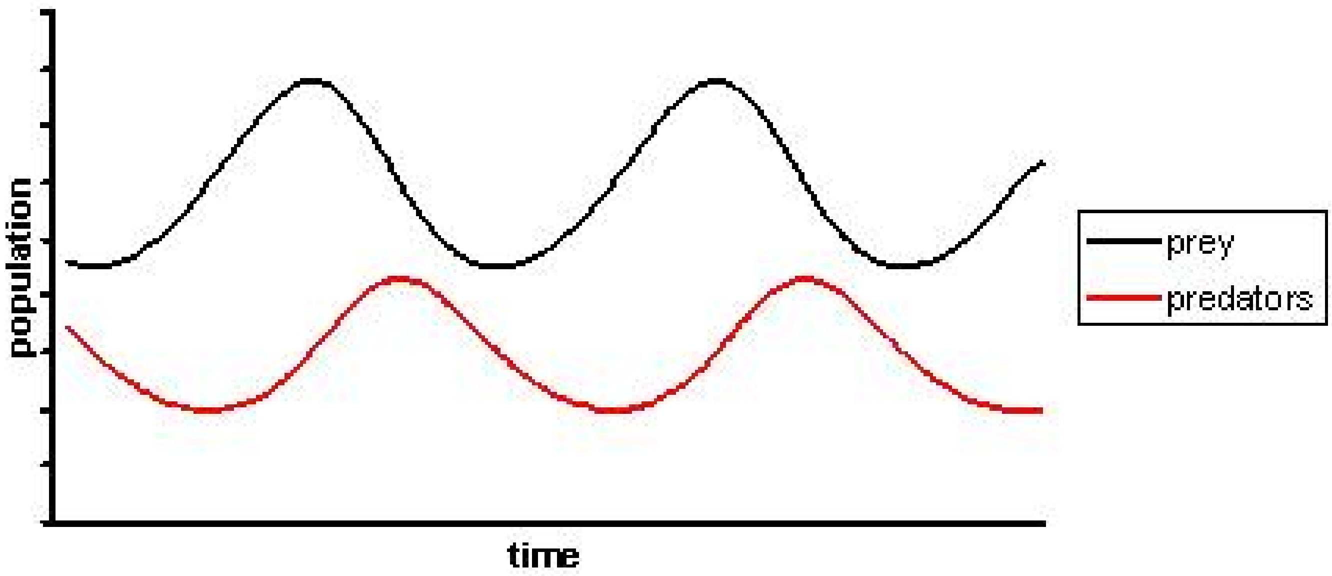
V .Volterra, commercial fishing in the Adriatic

x_1 = biomass of predators (sharks)

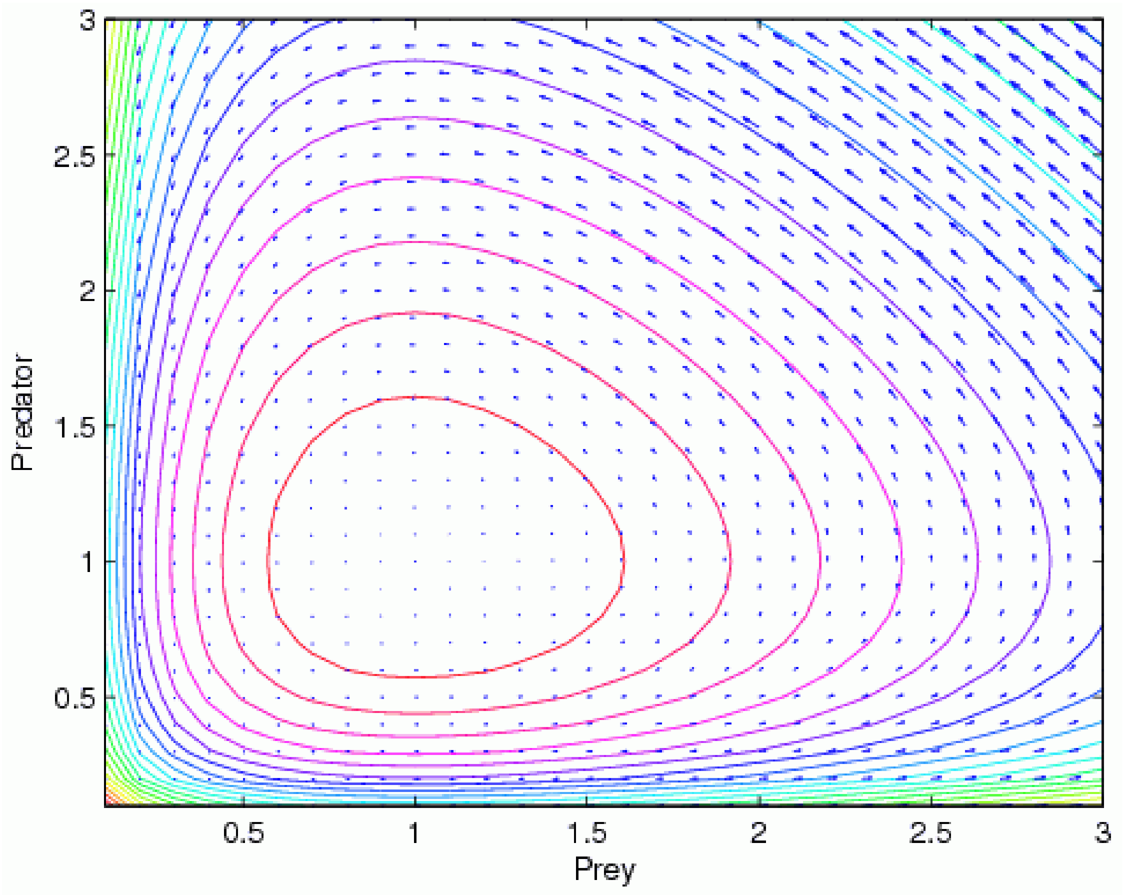
x_2 = biomass of prey (fish)

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

As Functions of Time

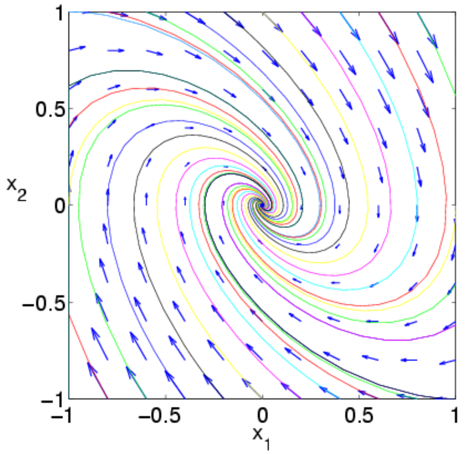


State-Space Diagram: The x_1-x_2 Plane



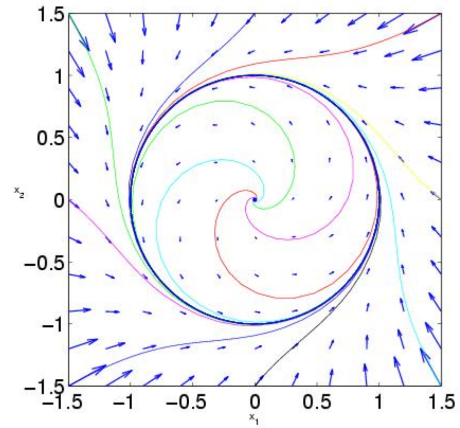
More Behaviors

Self-limiting term → stable focus



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

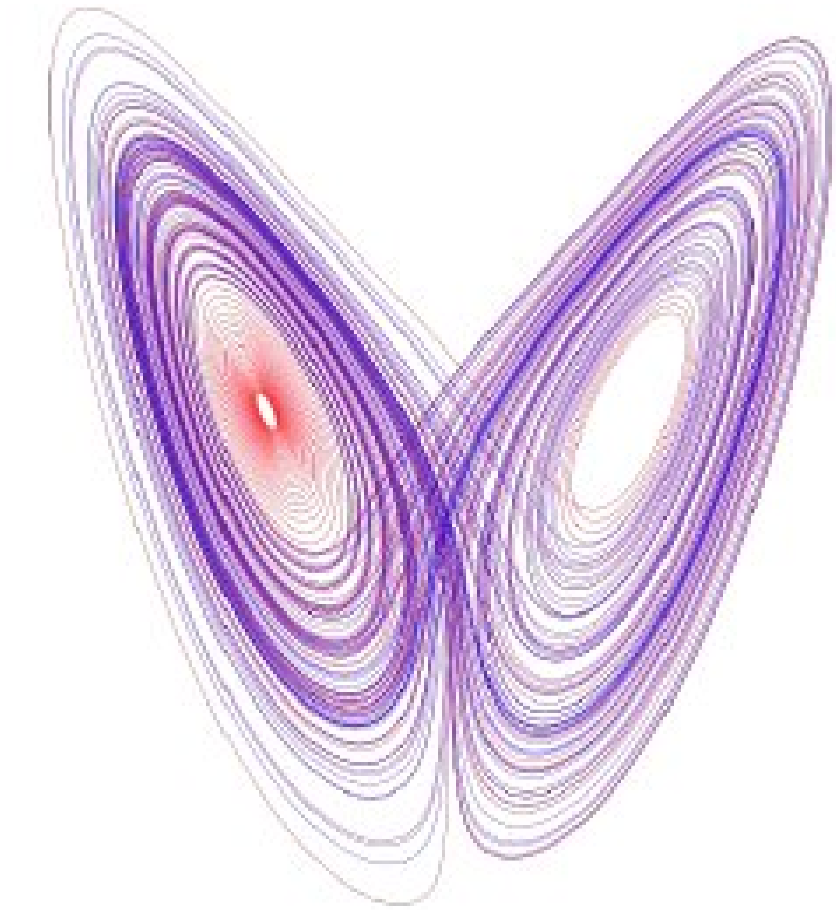
Delay → limit cycle



Varieties of Behavior

- Stable focus
- Periodic
- Limit cycle

Varieties of Behavior



- Stable focus
- Periodic
- Limit cycle
- Chaos

Terminology

- **Order**: highest order of derivative determines order of ODE

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

$$y'' = F / m$$

- **Explicit**: Can express k-th derivative in terms of lower orders

$$y^{(k)} = f(t, y, y', y'', \ldots, y^{(k-1)})$$

$$y'' = F / m$$

- **Implicit:** More general

$$f(t, y, y', y'', \ldots, y^{(k)}) = 0$$

Notational Conventions

- t is independent variable (scalar for ODEs)
- y is dependent variable
  - may be vector-valued
- focus exclusively here on explicit, first-order ODEs:

$$\mathbf{y}' = f(t, \mathbf{y}) \text{ where } f : \Re^{n+1} \to \Re^n$$

- Special case: f does not depend explicitly on t: **autonomous** ODE

$$\mathbf{y}' = f(\mathbf{y})$$

Transforming a higher-order ODE into a system of first-order ODEs

For k-th order ODE

$$y^{(k)}(t) = f(t, y, y', \ldots, y^{(k-1)})$$

define k new unknown functions

$$u_1(t) = y(t),\ u_2(t) = y'(t),\ \ldots,\ u_k(t) = y^{(k-1)}(t)$$

Then original ODE is equivalent to first-order system

$$\begin{bmatrix} u_1'(t) \\ u_2'(t) \\ \vdots \\ u_{k-1}'(t) \\ u_k'(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ u_3(t) \\ \vdots \\ u_k(t) \\ f(t, u_1, u_2, \ldots, u_k) \end{bmatrix}$$

Newton's second law as first-order system

$$y'' = F/m$$

Defining $u_1 = y$ and $u_2 = y'$ yields equivalent system of two first-order ODEs

$$\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} u_2 \\ F/m \end{bmatrix}$$

Solving ODEs

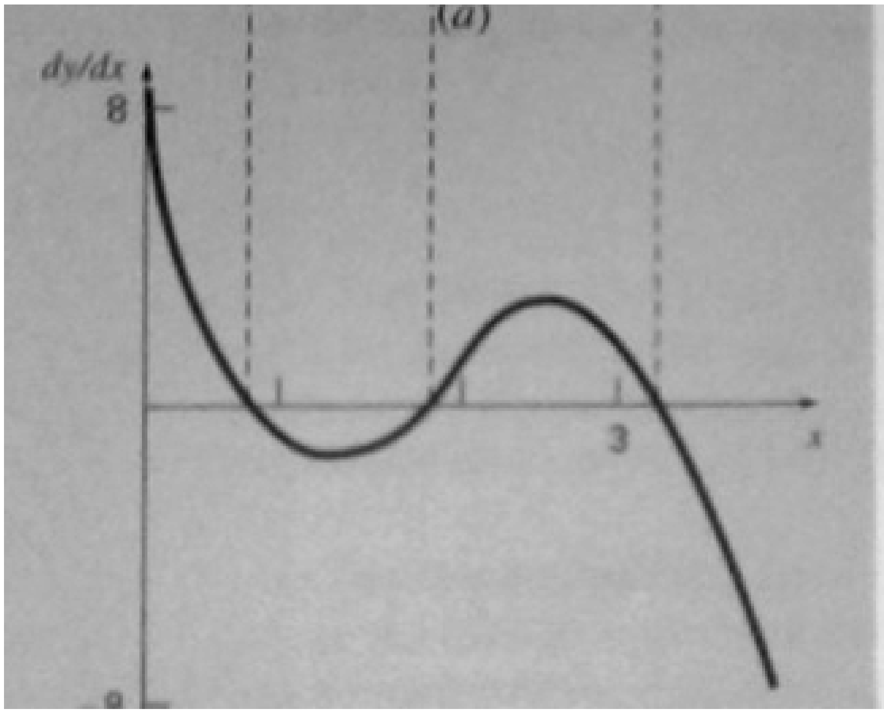
What does it mean to solve an ODE?

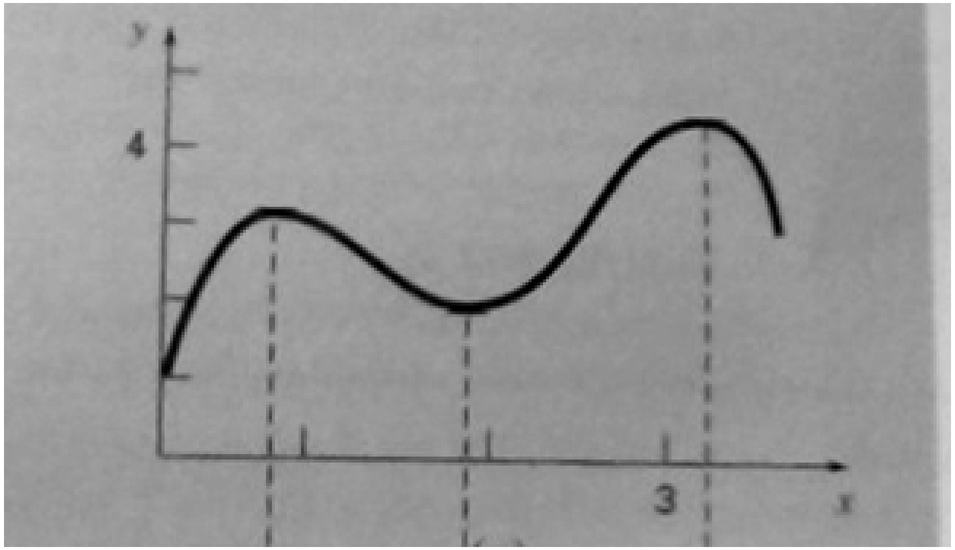
- **Analytically:**
  transform f(t, y, y′, y″… $y^{(k)}$)
  into equation of form y = …

  e.g., transform $\frac{dy}{dx} = -2x^3 - 12x^2 - 20x + 8.5$

  into $y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + C$
- **Numerically:**
  use f(t, y, y′, y″… $y^{(k)}$) to compute
  approximations of y for discrete values of t
  - **e.g.,** (y_1, t_1), (y_2, t_2), …(y_n, t_n)

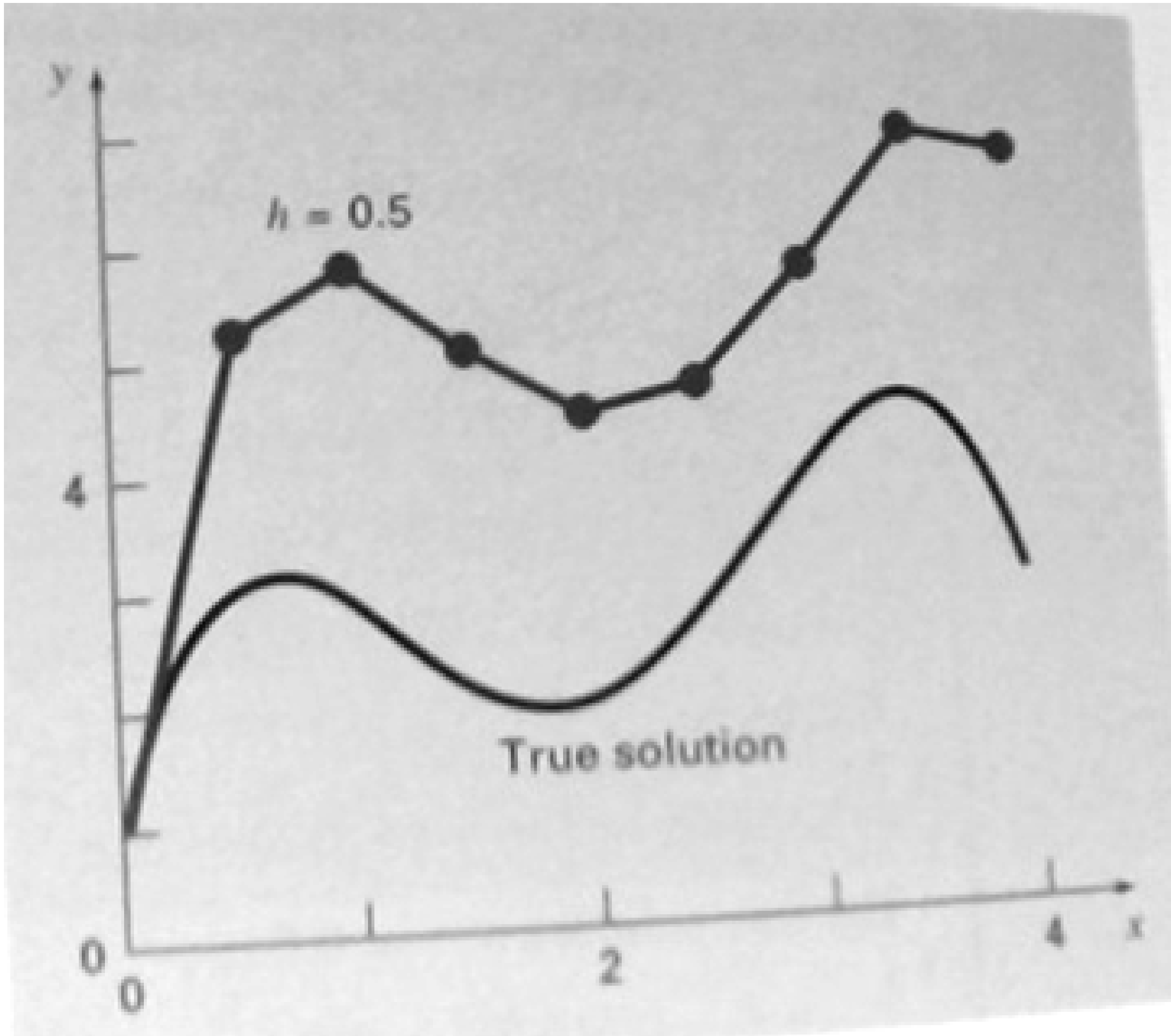
Analytically-derived solution





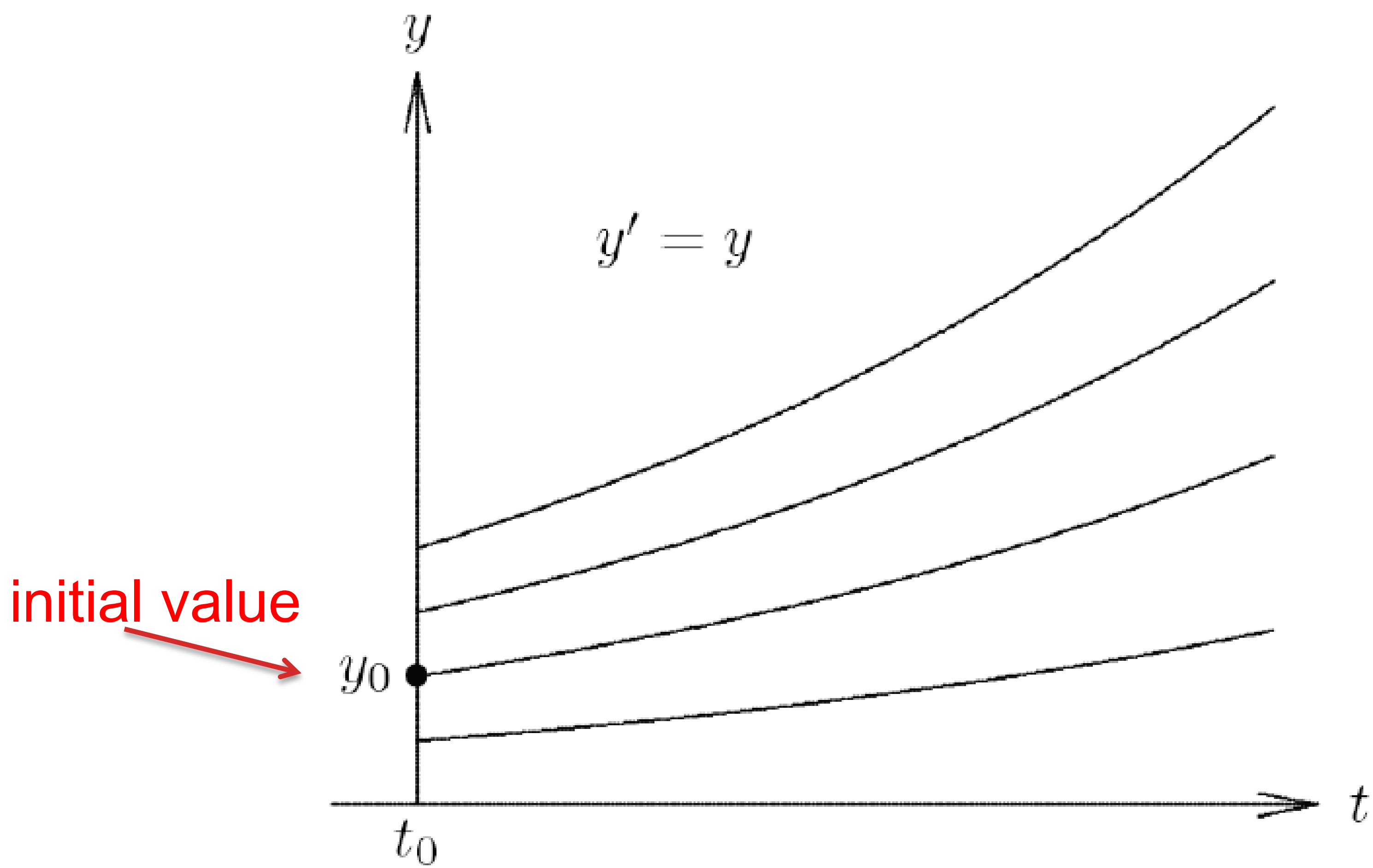
dy/dt → y

Numerically-derived Solution



ODEs have many solutions

Family of solutions for ODE $y' = y$



IVP vs BVP

- Today: Initial Value Problems
  - Complete state known at $t = t_0$
- As opposed to Boundary Value Problems
  - Parts of state known at multiple values of t

ODEs and integration

- If $y' = f(t, y)$ and $y(t_0) = y_0$, then

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds$$

- This directly useful only if *f* is independent of *y*, but helps us understand why there are so many parallels to numerical integration

Numerical Methods for ODEs

Need for numerical methods

- Linear ODEs are nice:

  $$a_n(t)\, y^{(n)} + \ldots a_1(t)\, y' + a_0(t)\, y = f(t)$$

- No analytical solutions for most nonlinear ODEs
- Can **sometimes** locally linearize non-linear ODEs; e.g., pendulum equation

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

  can be estimated as $\dfrac{d^2\theta}{dt} + \dfrac{g}{l}\theta = 0$

Numerical methods for ODEs

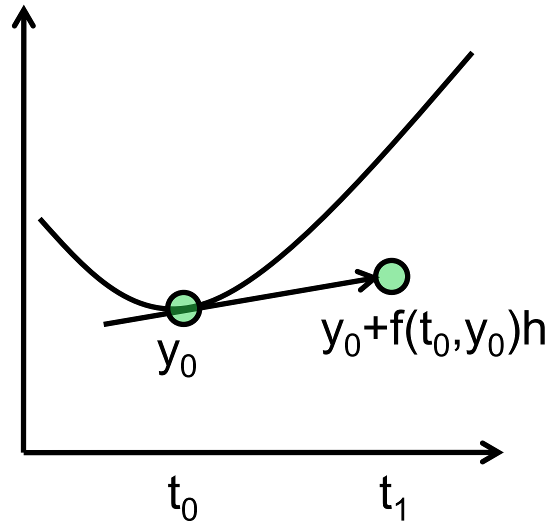
- Can't solve many (most) interesting problems analytically
- Numerical methods find y_k at a discrete set of t_k given $f(y, t)$ and y_0
- Important considerations:
  - Accuracy / error analysis
  - Efficiency: running time, number of steps
  - Stability: will estimate of $y(t_k)$ diverge from true value?

"Simplest possible" method

- Known: $$\frac{dy}{dt} = f(t,y)$$
  $$y = y_0 \text{ at t} = \text{t}_0$$
- What is y_1 at time $t_1 = t_0 + h$?

$$y_1 = y_0 + f(t_0, y_0)h$$

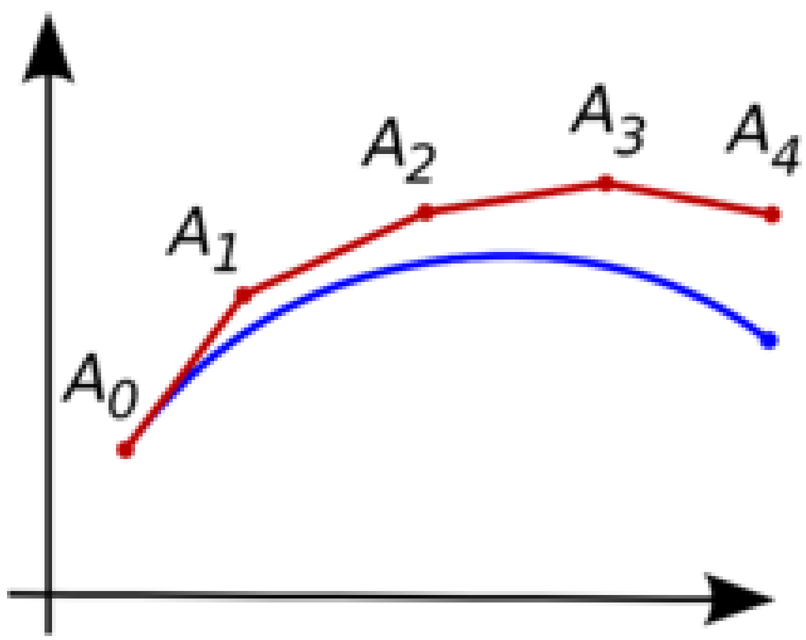
Euler's method



Forward (Explicit) Euler's method

- Can repeat for subsequent estimates:

$$y_{i+1} = y_i + f(t_i, y_i)h$$



Example

from Chapra & Canale

Solve $\dfrac{dy}{dt} = -2t^3 - 12t^2 - 20t + 8.5$

for $t = 1$ given $y = 1$ at $t = 0$, and for step size 0.5 :

Step 1 :

$y(0.5) = y(0) + f(0,1) * 0.5$

where $y(0,1) \;=\; -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$

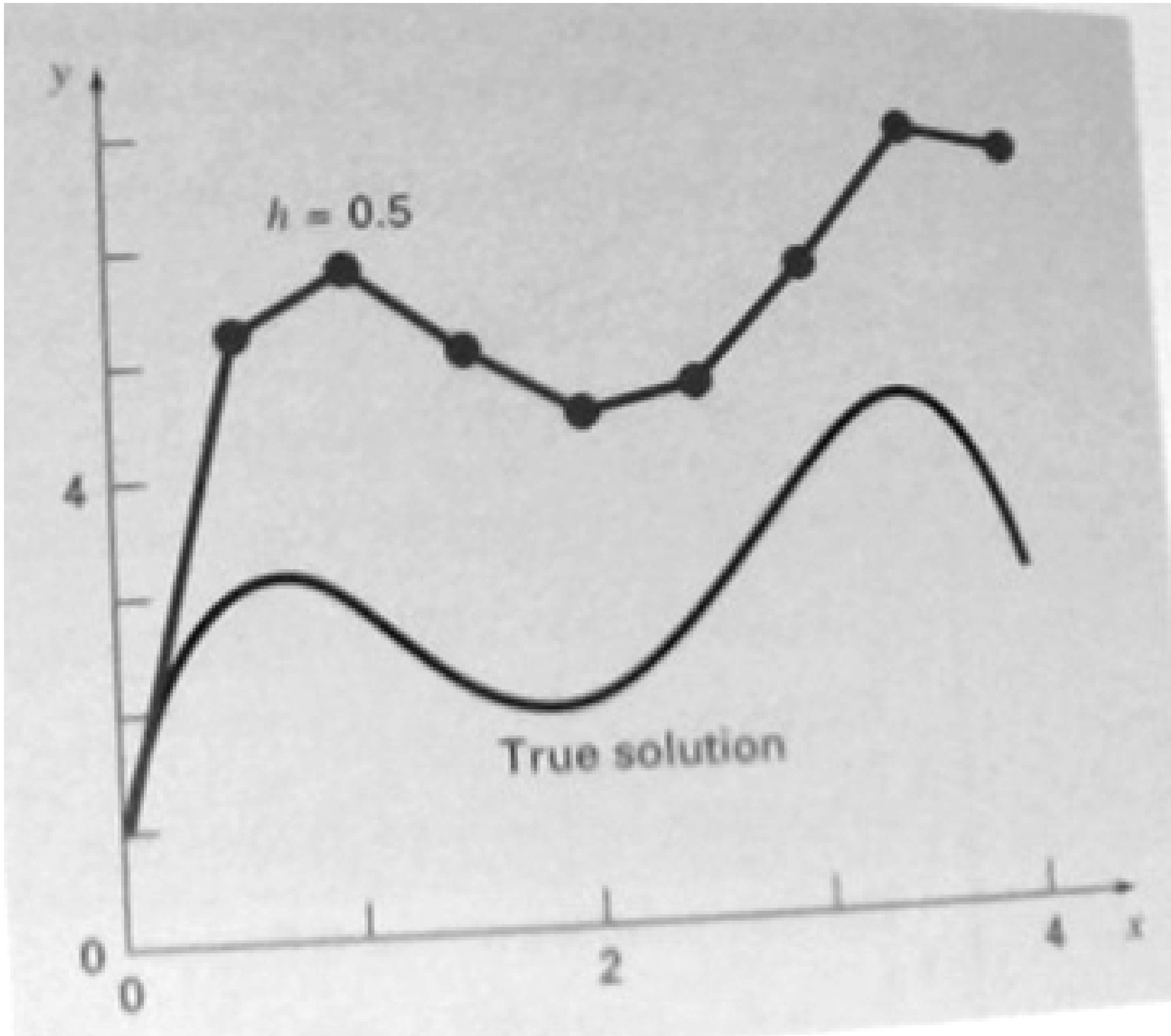
so $y(0.5) = 5.25$

Step 2 :

$y(1.0) = y(0.5) + f(0.5, 5.25) * 0.5$

$= 5.25 + [-2(0.5)^3 + 12(0.5)^2 - 20(0.5) + 8.5] * 0.5$

Sequence of Euler solutions



Error analysis of Euler's method

Derive y_{i+1} using Taylor series expansion around (t_i, y_i):

$$y_{i+1} = y_i + f(t_i, y_i)h + \frac{f'(t_i, y_i)h^2}{2!} + \ldots + \frac{f^{(n-1)}(t_i, y_i)h^n}{n!} + O(h^{n+1})$$

Euler's method uses first two terms of this, so we have

truncation error:

$$E_t = \frac{f'(t_i, y_i)h^2}{2!} + \ldots + \frac{f^{(n-1)}(t_i, y_i)h^n}{n!} + O(h^{n+1})$$

$$E = O(h^2)$$

This is **local error.**

Works perfectly if solution is linear: it's a **first-order method**

Local and Global Error

Global error: difference between computed solution and true solution $\boldsymbol{y}(t)$ passing through initial point $(t_0, \boldsymbol{y}_0)$

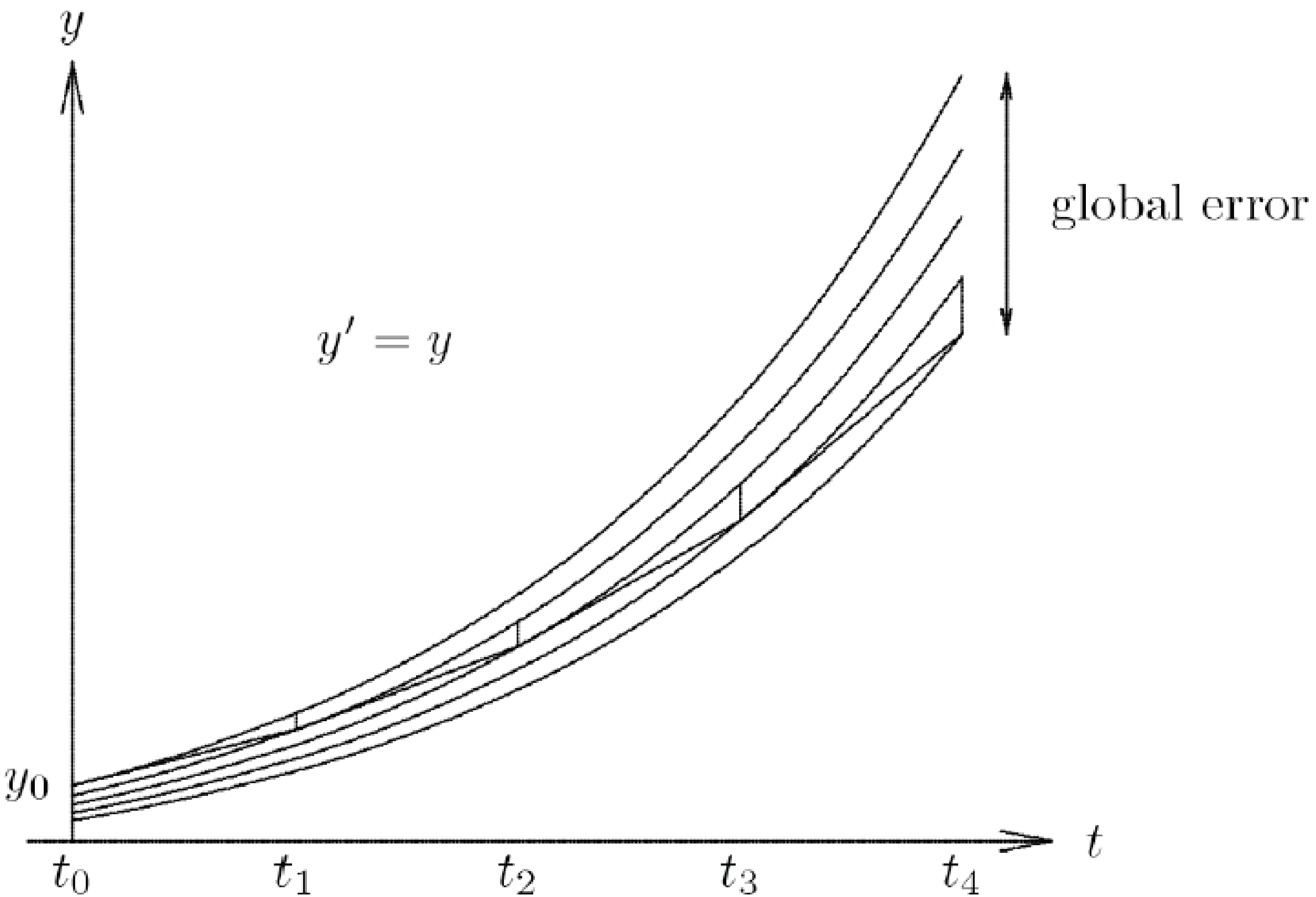
$$\boldsymbol{e}_k = \boldsymbol{y}_k - \boldsymbol{y}(t_k)$$

Local error: error made in one step of numerical method

$$\boldsymbol{\ell}_k = \boldsymbol{y}_k - \boldsymbol{u}_{k-1}(t_k)$$

where $\boldsymbol{u}_{k-1}(t)$ is true solution passing through previous point $(t_{k-1}, \boldsymbol{y}_{k-1})$

Local and Global error



Error analysis, in general

- Local error: concerned with **accuracy** at each step
  - Euler's method: $O(h^2)$
- Global error: concerned with stability over multiple steps
  - Euler's method: $O(h)$
- In general, for nth-order method:
  - Local error $O(h^{n+1})$, global error $O(h^n)$
- Stability is **not guaranteed**

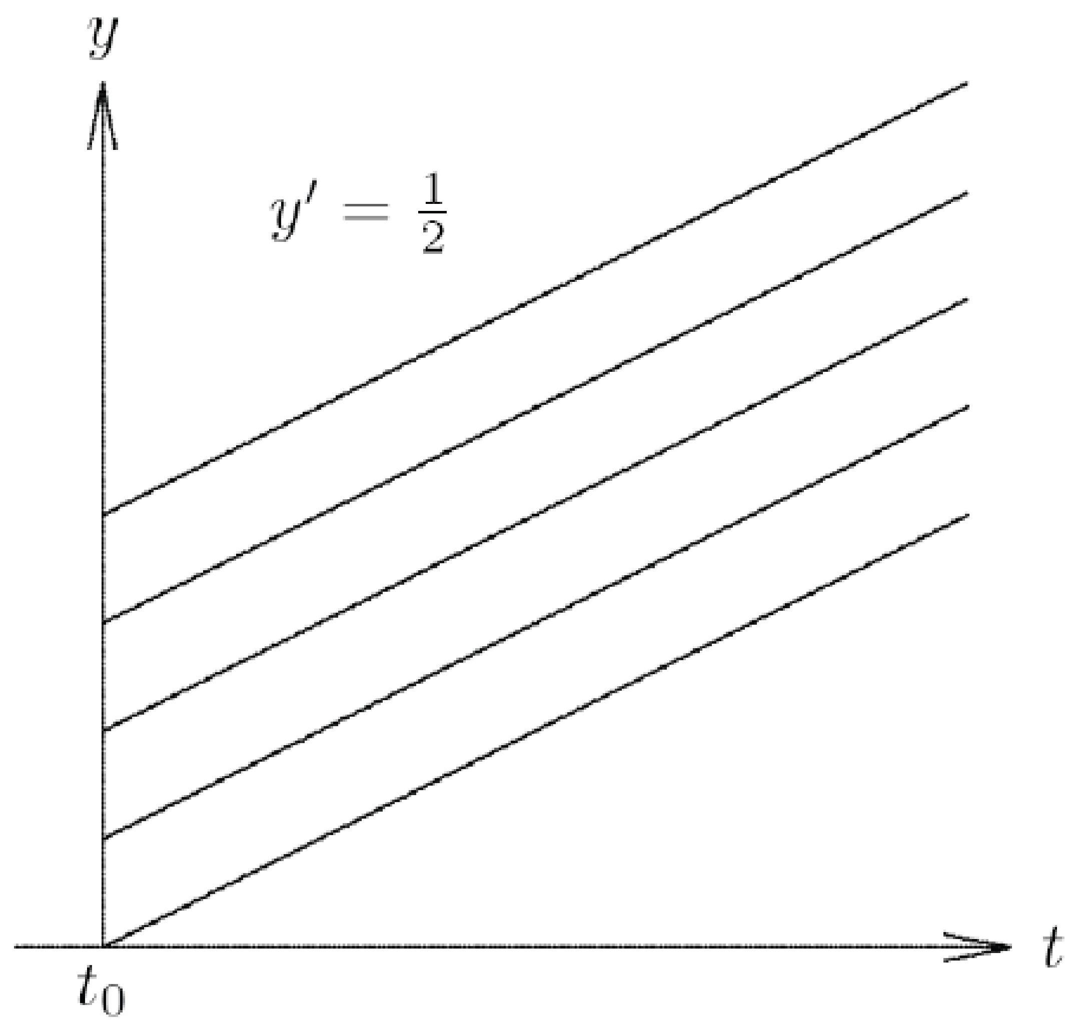
Stability of ODE

Solution of ODE is

- *Stable* if solutions resulting from perturbations of initial value remain close to original solution
- *Asymptotically stable* if solutions resulting from perturbations converge back to original solution
- *Unstable* if solutions resulting from perturbations diverge away from original solution without bound

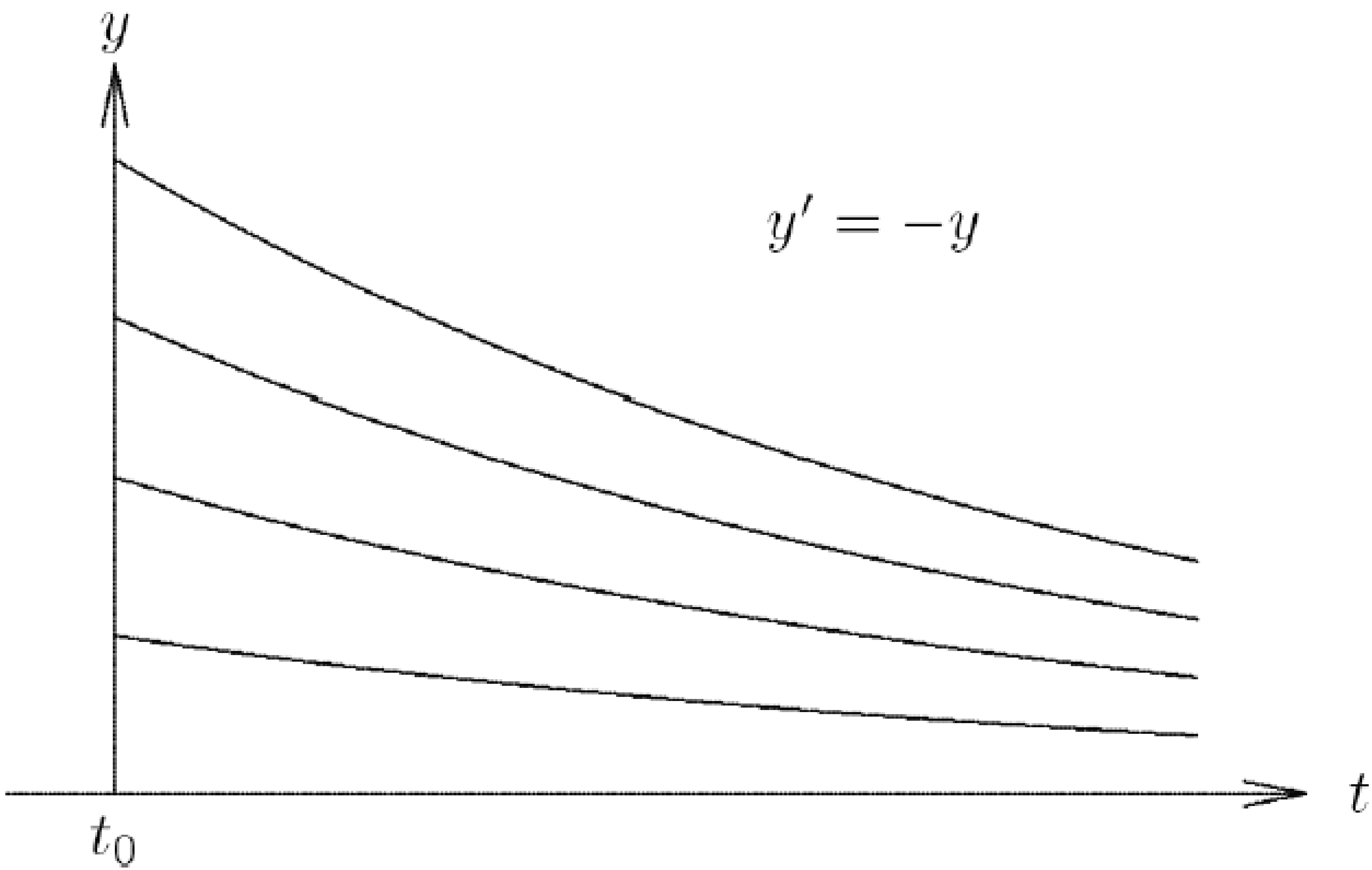
Stable

Family of solutions for ODE $y' = \frac{1}{2}$



Asymptotically Stable

Family of solutions for ODE $y' = -y$



Stability of **Method**

- Possible to have instability (divergence from true solution) even when solutions to ODE are stable
- Euler's method sensitive to choice of h:
  - Consider $dy/dt = -\lambda y$
  - Analytic solution is $y(t) = y_0 \, e^{-\lambda t}$
  - Forward Euler step is $y_{k+1} = y_k - \lambda y_k h = y_k (1 - \lambda h)$
  - Euler's method unstable if $h > 2/\lambda$

Other methods often have better stability.

Taylor Series Methods

- Euler's method can be derived from Taylor series expansion
- By retaining more terms in Taylor series, we can generate higher-order single-step methods
- For example, retaining one additional term in Taylor series

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + h\,\boldsymbol{y}'(t) + \frac{h^2}{2}\,\boldsymbol{y}''(t) + \frac{h^3}{6}\,\boldsymbol{y}'''(t) + \cdots$$

gives second-order method

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h_k\,\boldsymbol{y}'_k + \frac{h_k^2}{2}\,\boldsymbol{y}''_k$$

Why not use TS methods?

- Requires higher-level derivatives of y
- Ugly and hard to compute!
- More efficient higher-order methods exist

Runge-Kutta Methods

Runge-Kutta

- Family of techniques
- Achieves accuracy of Taylor Series without needing higher derivatives
- Accomplishes this by evaluating f several times between t_k and t_{k+1}

Runge-Kutta: General Form

$$y_{i+1} = y_i + \phi(t_i, y_i, h)h$$

where $\phi = a_1k_1 + a_2k_2 + \ldots + a_nk_n$

and

$$k_1 = f(t_i, y_i)$$

$$k_2 = f(t_i + p_1h, y_i + q_{11}k_1h)$$

$$k_3 = f(t_i + p_2h, y + q_{21}k_1h + q_{22}k_2h)$$

$$\vdots$$

$$k_n = f(t_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + \ldots + q_{n-1,n-1}k_{n-1}h)$$

Euler as R-K

- Let n = 1

$$y_{i+1} = y_i + \phi(t_i, y_i, h)h$$

where $\phi = a_1 k_1$

and

$$k_1 = f(t_i, y_i)$$

$$a_1 = 1$$

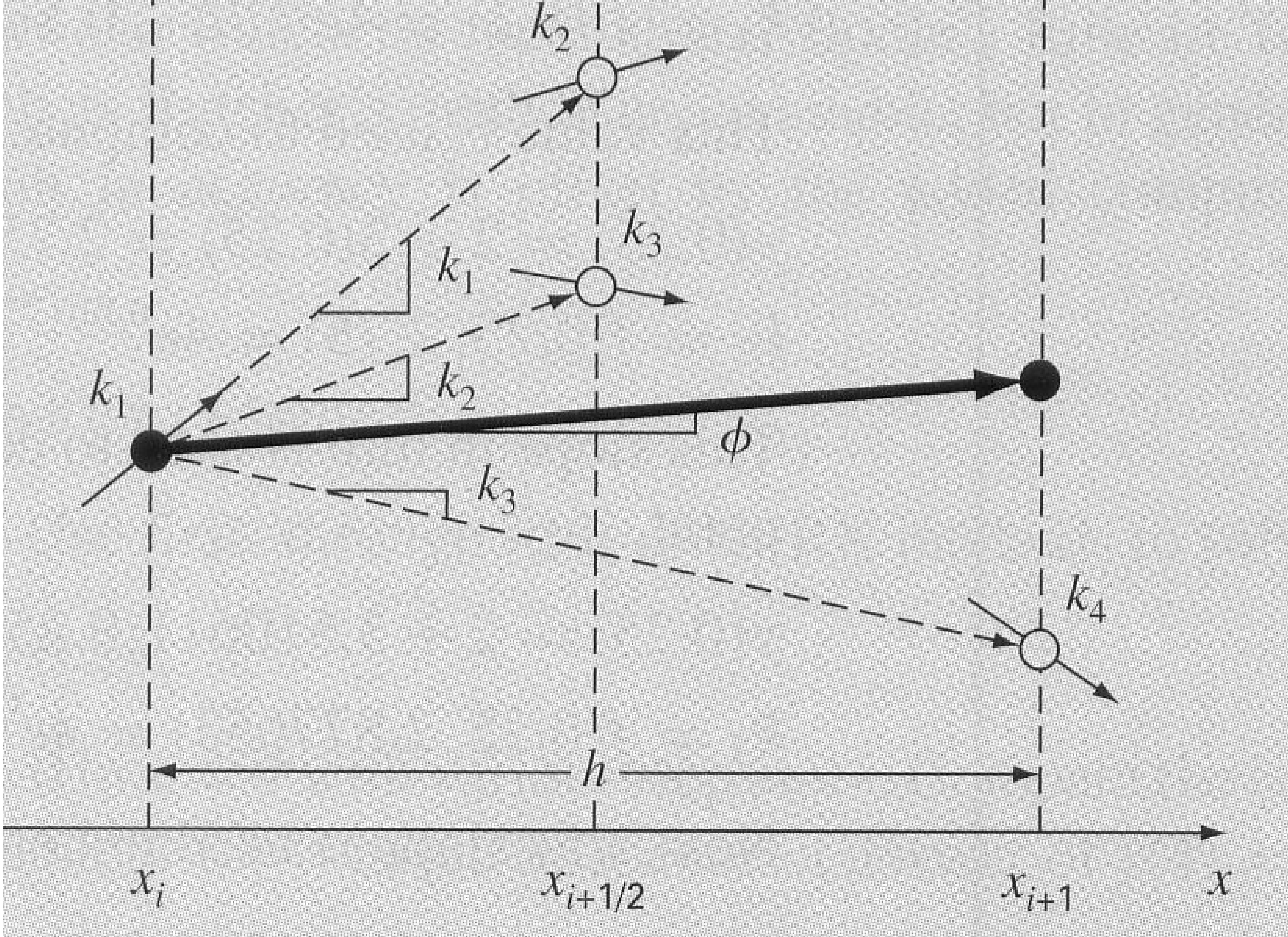
Higher-Order RK

- Midpoint method

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

- 4th-order Runge Kutta

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


k_2
k_3
k_1
k_1
k_2
ϕ
k_3
k_4
h
x_i
$x_{i+1/2}$
x_{i+1}
x

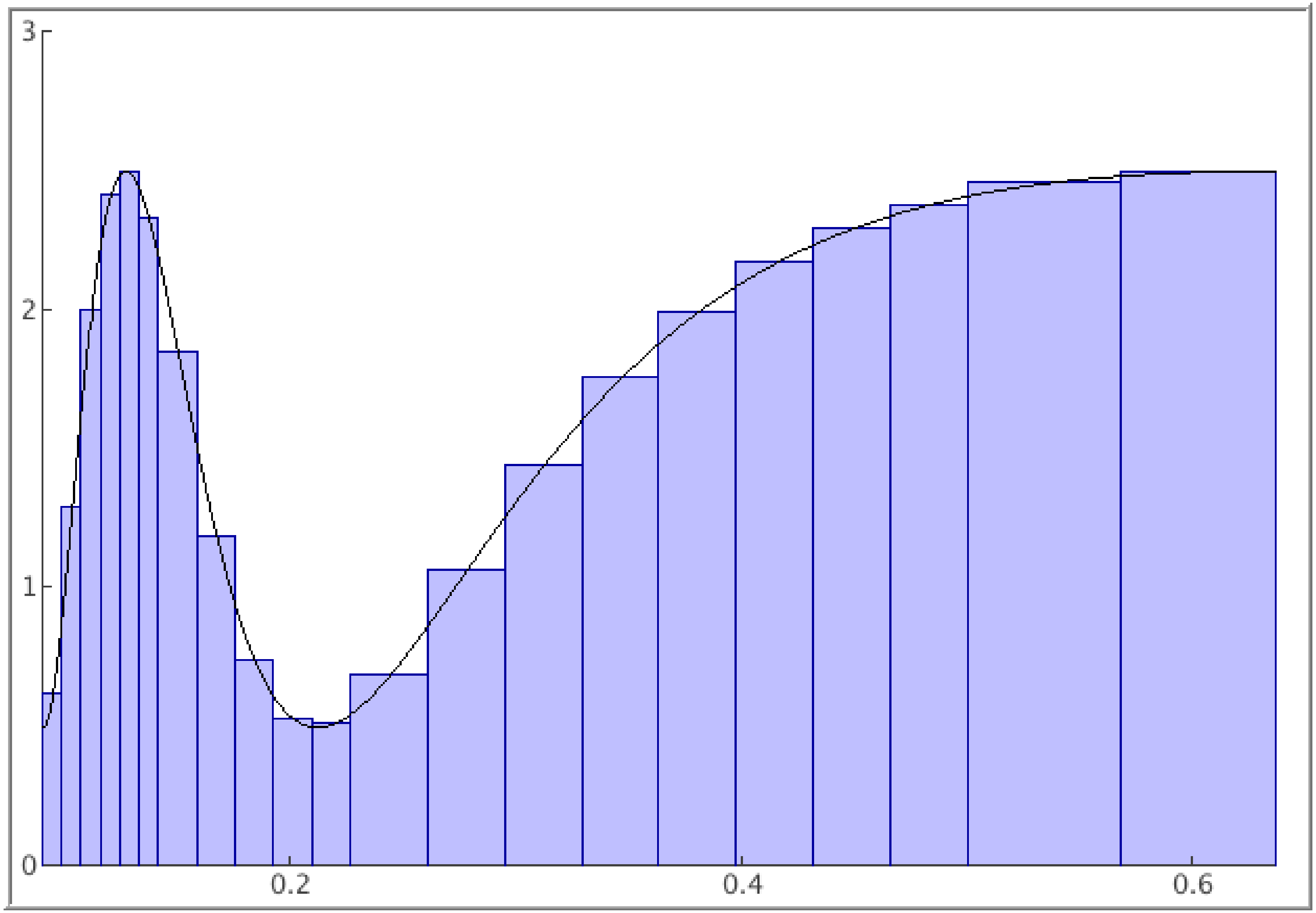
Usual Bag of Tricks: Extrapolation

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

Usual Bag of Tricks: Adaptive Solvers

- Change step size to get better coverage when function is chanigng quickly
- Determine appropriate step size by estimating error
  - Method 1: Halve the RK step size and compare results: Error = $y_2 - y_1$
  - Method 2: Compute RK predictions of different **order**

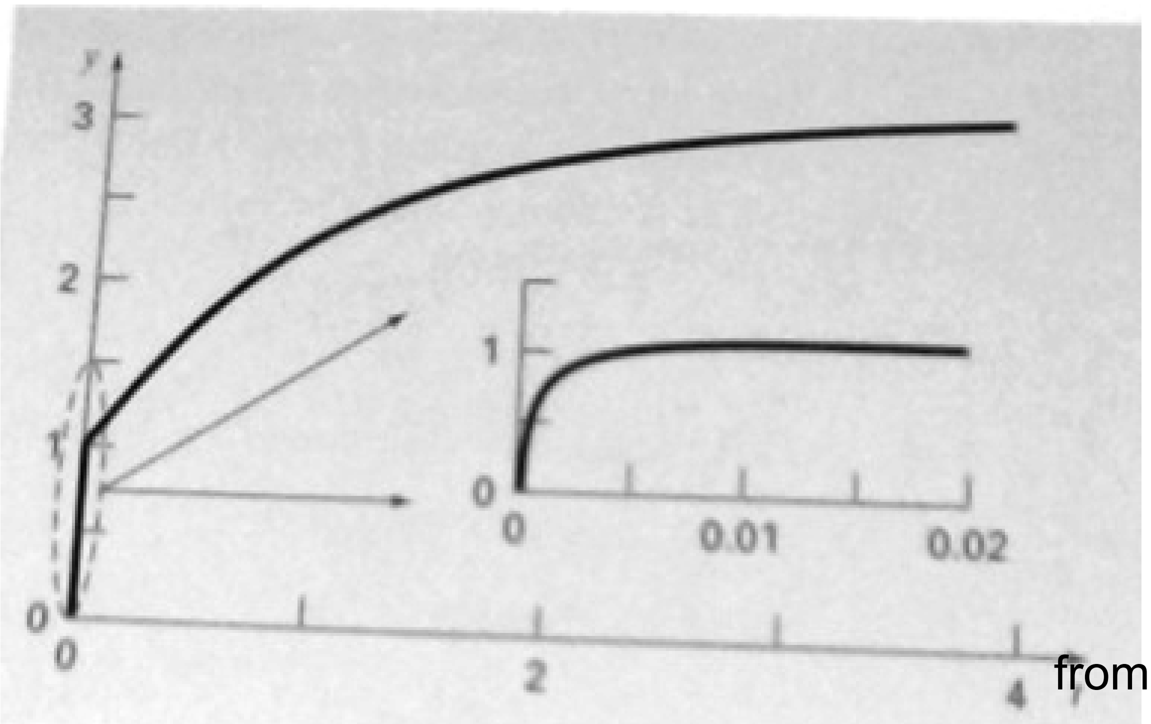
Adaptive Quadrature



Stiff ODEs and Implicit Methods

Stiff ODE

- May involve transients, rapidly oscillating components: rates of change much smaller than interval of study



from Chapra & Canale

Another Stiff ODE

- Consider scalar ODE

$$y' = -100y + 100t + 101$$

  with initial condition $y(0) = 1$

- General solution is $y(t) = 1 + t + ce^{-100t}$, and particular solution satisfying initial condition is $y(t) = 1 + t$ (i.e., $c = 0$)

- Since solution is linear, Euler's method is theoretically exact for this problem

- However, to illustrate effect of using finite precision arithmetic, let us perturb initial value slightly

- With step size $h = 0.1$, first few steps for given initial values are

t	0.0	0.1	0.2	0.3	0.4
exact sol.	1.00	1.10	1.20	1.30	1.40
Euler sol.	0.99	1.19	0.39	8.59	-64.2
Euler sol.	1.01	1.01	2.01	-5.99	67.0

- Computed solution is incredibly sensitive to initial value, as each tiny perturbation results in wildly different solution

- Any point deviating from desired particular solution, even by only small amount, lies on different solution, for which $c \neq 0$, and therefore rapid transient of general solution is present

See http://www.cse.illinois.edu/iem/ode/stiff/

Backward (Implicit) Euler

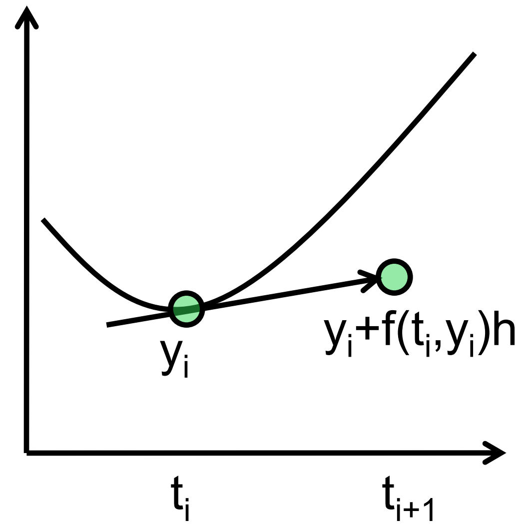
$$x^{(k+1)} = x^{(k)} + g(x^{(k+1)})\,h$$

- 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

Predictor-Corrector Methods

Heun's method

- Euler: Assumes derivative at t_{i-1} is a good estimate for whole interval



- Heun: average derivative at t_i, t_{i+1}

$$y_{i+1} = y_i + \frac{f(t_i, y_i) + f(t_{i+1}, y_{i+1})}{2} h$$

Heun's method

- Predict y_{i+1}, then use slope at y_{i+1} to correct the prediction
- Predictor:

$$y_{i+1}^{0} = y_i + f(t_i, y_i)h$$

- Corrector:

$$y_{i+1} = y_i + \frac{f(t_i, y_i) + f(t_{i+1}, y_{i+1}^{0})}{2} h$$

Heun: An iterative method!

- Corrector: $y_{i+1} = y_i + \dfrac{f(t_i, y_i) + f(t_i, y_{i+1}^0)}{2} h$
- Can apply corrector once (so it's a 2nd order RK) or iteratively
- Error estimate: $|E| = \left| \dfrac{y_{i+1}^j - y_{i+1}^{j-1}}{y_{i+1}^j} \right|$
  - guaranteed to converge to something, not necessarily 0
- Error might not decrease monotonically, but should decrease eventually for sufficiently small h

Heun: Example

Solve $\dfrac{dy}{dt} = 4e^{0.8t} - 0.5y$

for $t = 1$ given $y = 2$ at $t = 0$, and for step size 1:

Step 1, Predict:

$$y_1^0 = y_0 + f(t_0, y_0)h = 2 + 4e^0 - 0.5(2) = 3$$

Step 2, Correct:

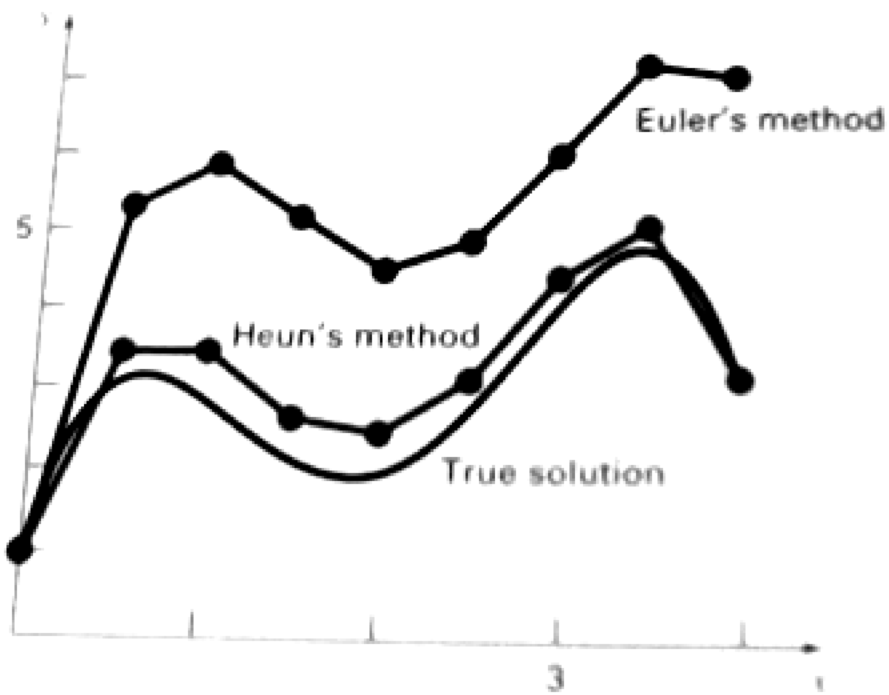
$$y_1^1 = y_0 + \frac{f(t_0, y_0) + f(t_1, y_1^0)}{2} h = 2 + \frac{3 + 6.402164}{2}(1) = 6.701082$$

Step 3, Correct again:

$$y_1^2 = y_0 + \frac{f(t_0, y_0) + f(t_1, y_1^1)}{2} h = 6.275811$$

Error of Heun's method

- Local: $O(h^3)$
- Global: $O(h^2)$ (i.e., it's a 2nd-order method)



Relationship between Heun and Trapezoid

- when dy/dt depends only on t:

$$dy/dt = f(t)$$

$$\int_{y_i}^{y_{i+1}} dy = \int_{t_i}^{t_{i+1}} f(t)dt$$

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t)dt$$

$$y_{i+1} \cong y_i + \frac{f(t_i) + f(t_{i+1})}{2}(t_{i+1} - t_i)$$