Ordinary Differential Equations
Part 2

COS 323

Last time

- Differential equations
- Numerical methods for solving ODE initial value problems
  - Euler's method
  - Heun's method
  - Runge-Kutta methods
  - Adaptive methods (e.g., adaptive R. K.)

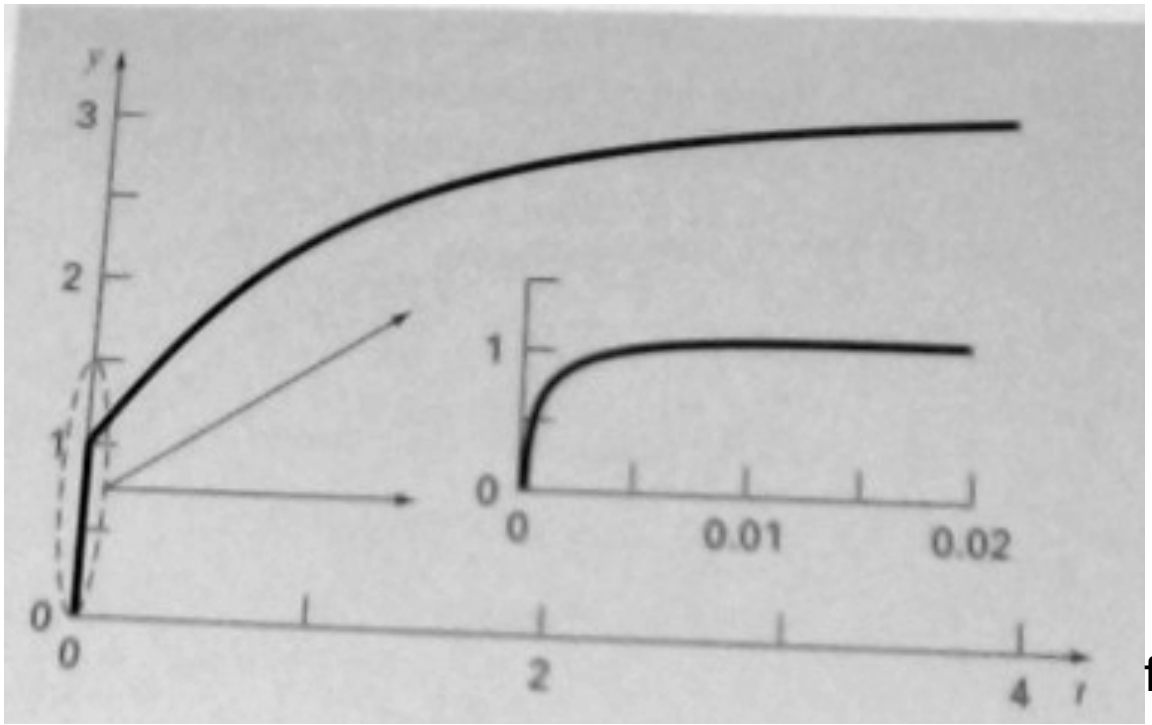
Today

- More methods for initial value problems
  - Stiff ODEs
  - Backward Euler
  - Multi-step methods
    - Adams methods
- Boundary value problems
  - Definition
  - Shooting method
  - Finite difference method
  - Collocation method

Stiff ODEs and Implicit Methods

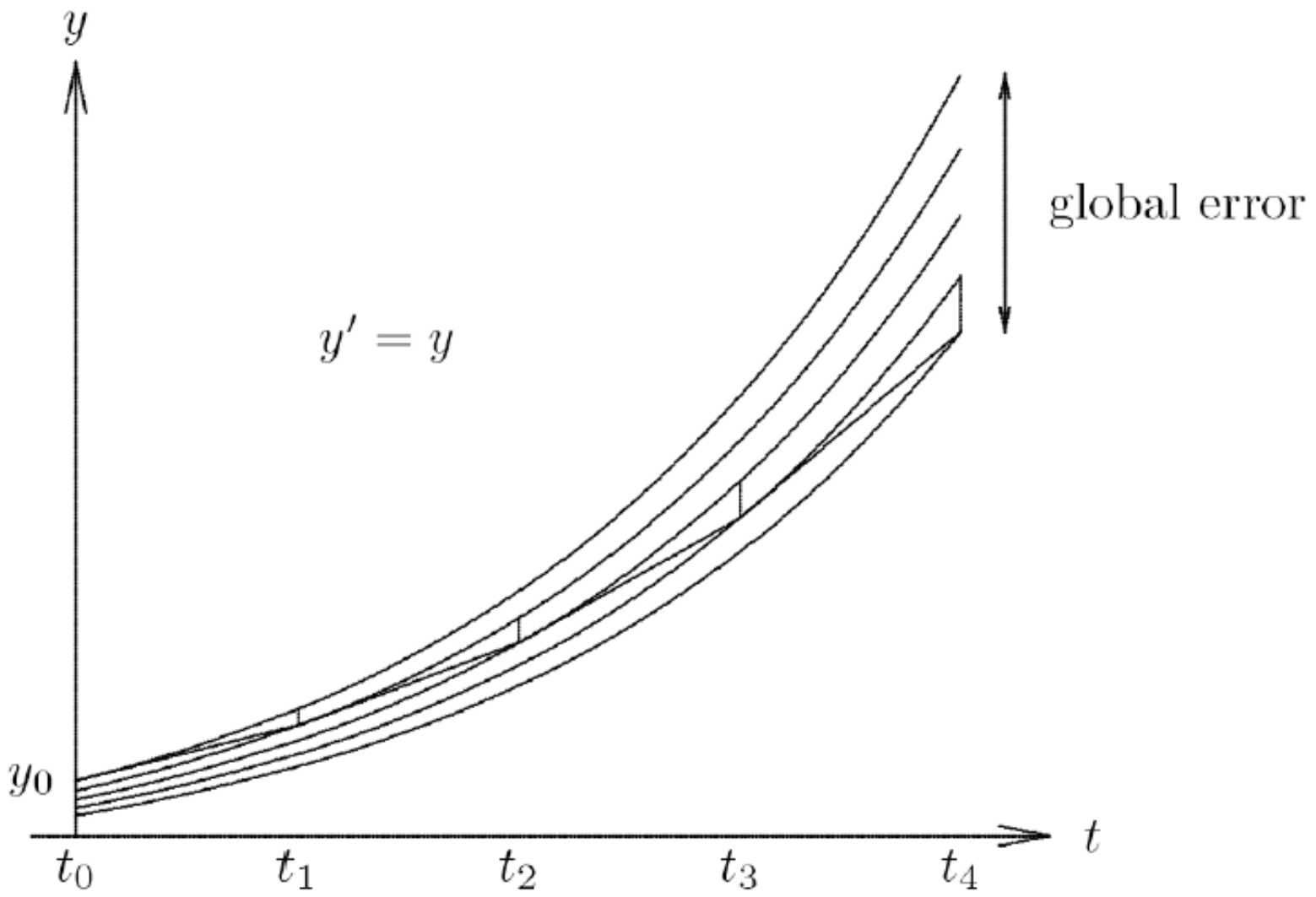
Stiff ODE

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

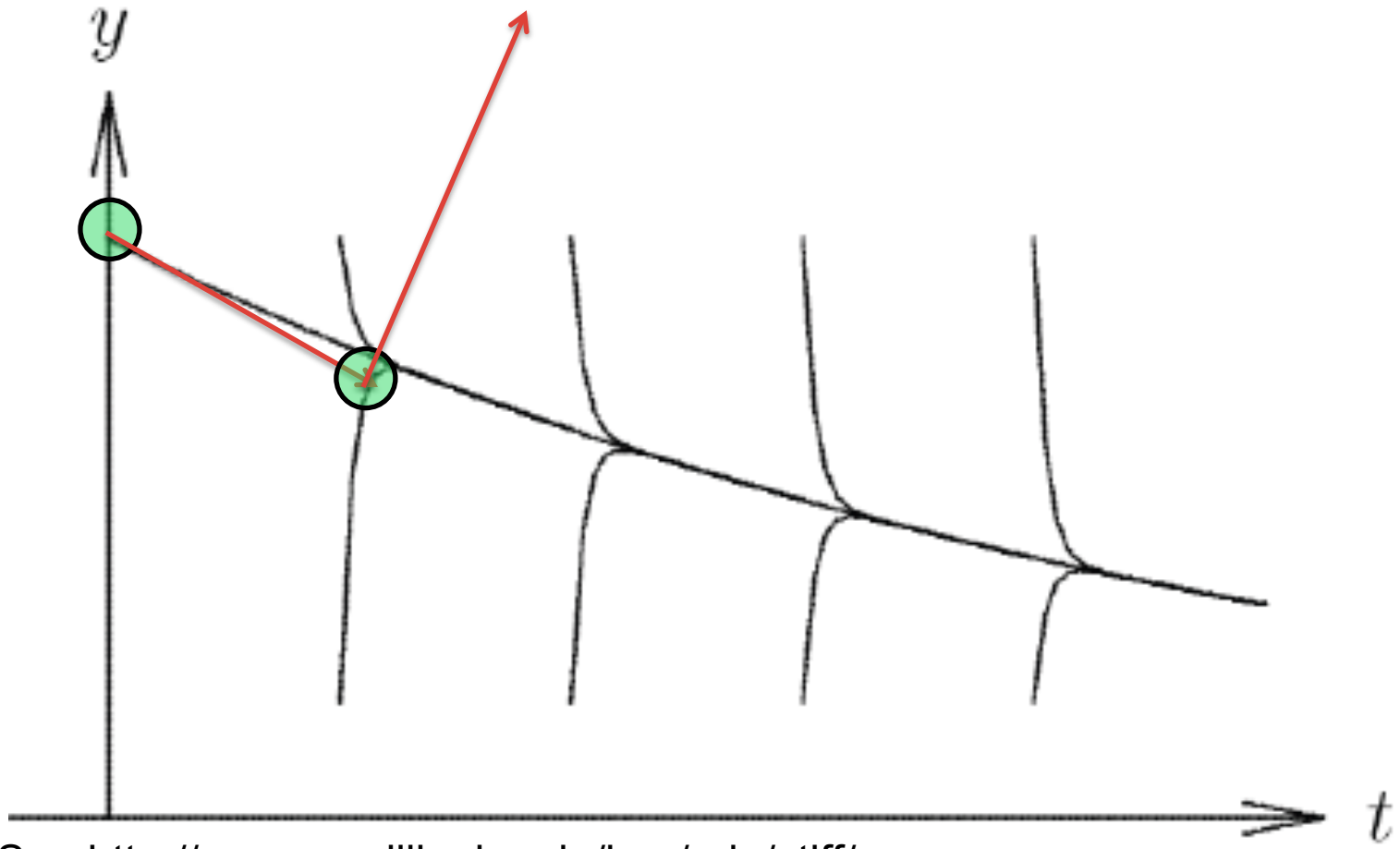


from Chapra & Canale

Non-stiff ODE



Stiff ODE



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

- 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/

Solving Stiff ODEs

- Adaptive Runge-Kutta?
  - Step size for stability may be VERY small, even when y not changing rapidly
- Implicit methods are often preferred method

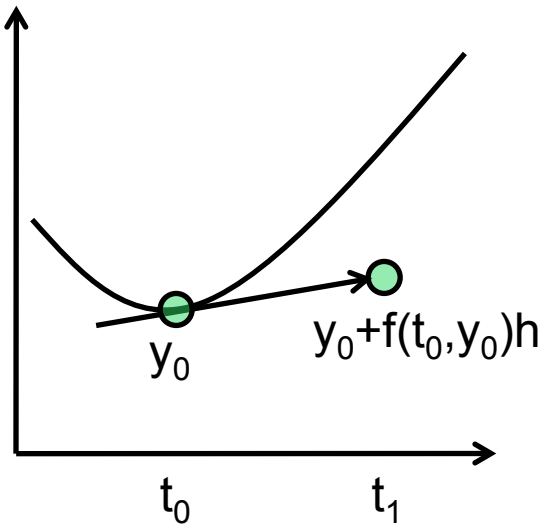
Euler's method

- Known: $\frac{dy}{dt} = f(t,y)$

  $y = y_0$ at $t = t_0$

- What is y_1 at time $t_1 = t_0 + h$?

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



Backward (Implicit) Euler's method

- Known: $\frac{dy}{dt} = f(t,y)$

  $y = y_0 \text{ at } t = t_0$

- What is y_1 at time $t_1=t_0 + h$?

$$y_1 = y_0 + f(t_1,y_1)h$$

- How to solve?
  - Fixed-point iteration or Newton-Raphson

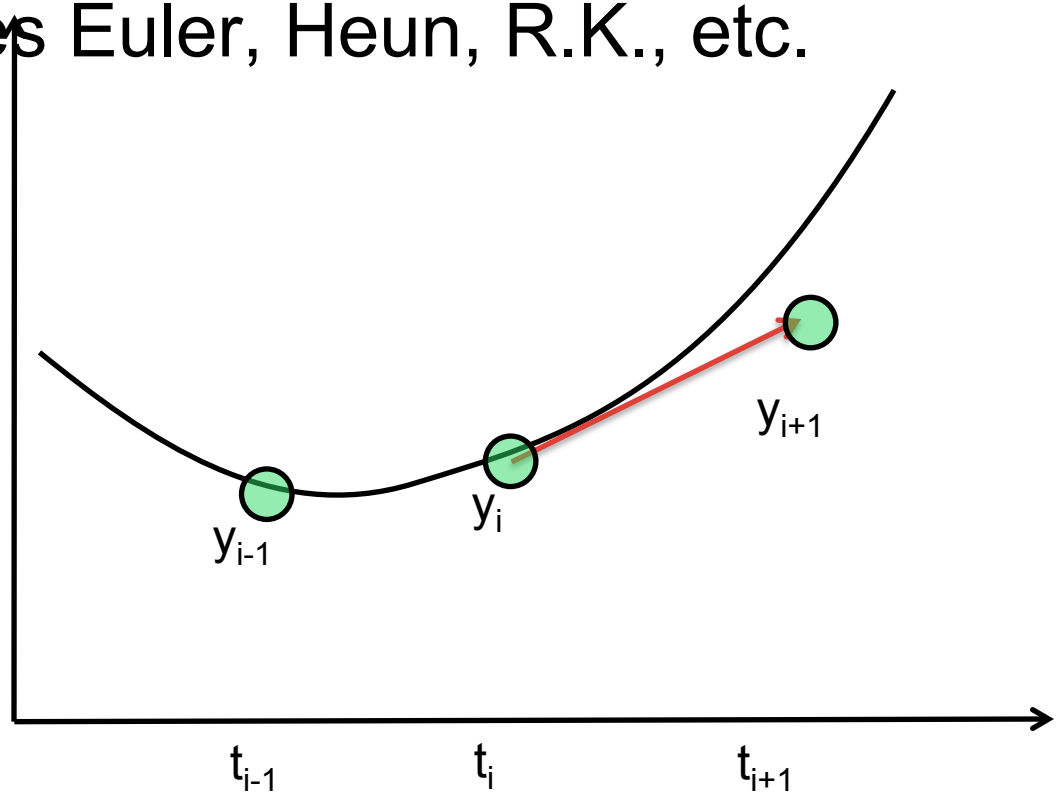
Implicit Methods

- Backward Euler
  - "Unconditionally stable" ☺
    - Unstable ODE still problematic
  - Can take larger step sizes
    - Can be more efficient, even given root-finding at each step!
  - Global error O(h) ☹
- Higher-order methods exist
  - e.g., second-order implicit trapezoid, implicit R.K. methods…

Multistep methods

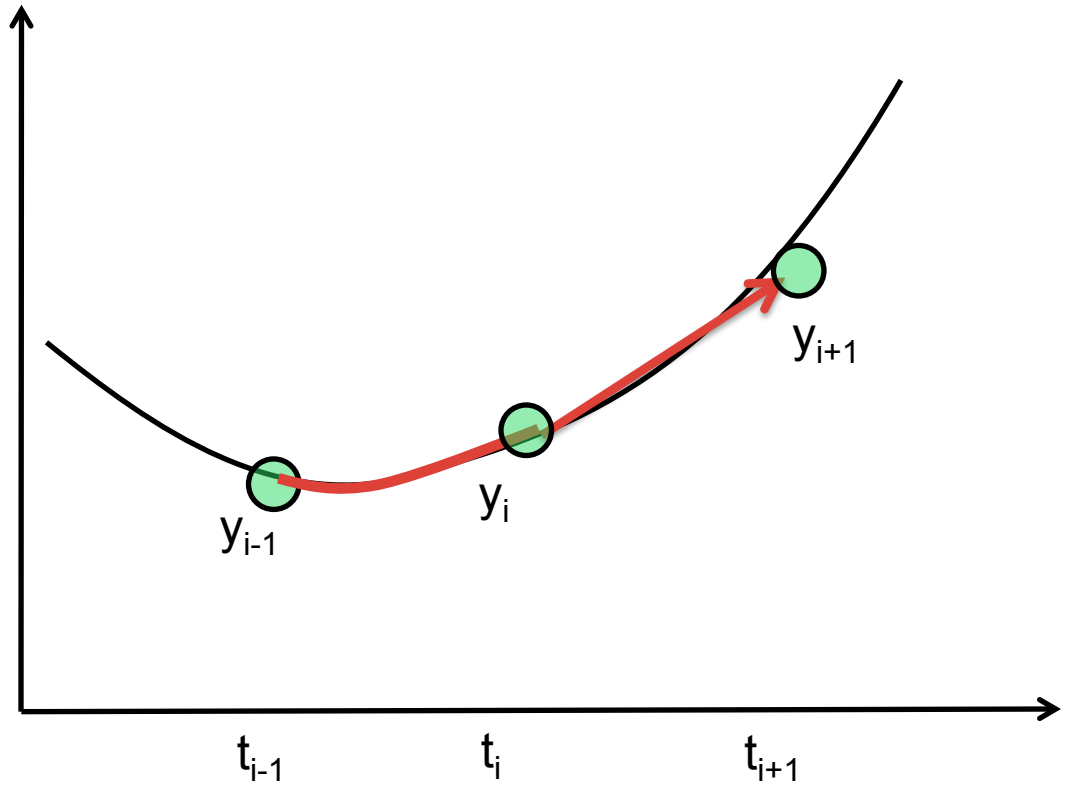
Single step methods

- Use information at a **single** t_i to predict y_{i+1} at t_{i+1}
- Includes Euler, Heun, R.K., etc.



Multistep methods

- Use information we've already learned from points at t_{i-1}, t_{i-2}, etc.



Heun (from last class)

- Predict y_i, then use slope at y_i to correct the prediction.
- Predictor: Euler's method

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

- Corrector: Trapezoidal rule

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

Non-self-starting Heun

- Use information from a previous point to improve prediction
- Old predictor:

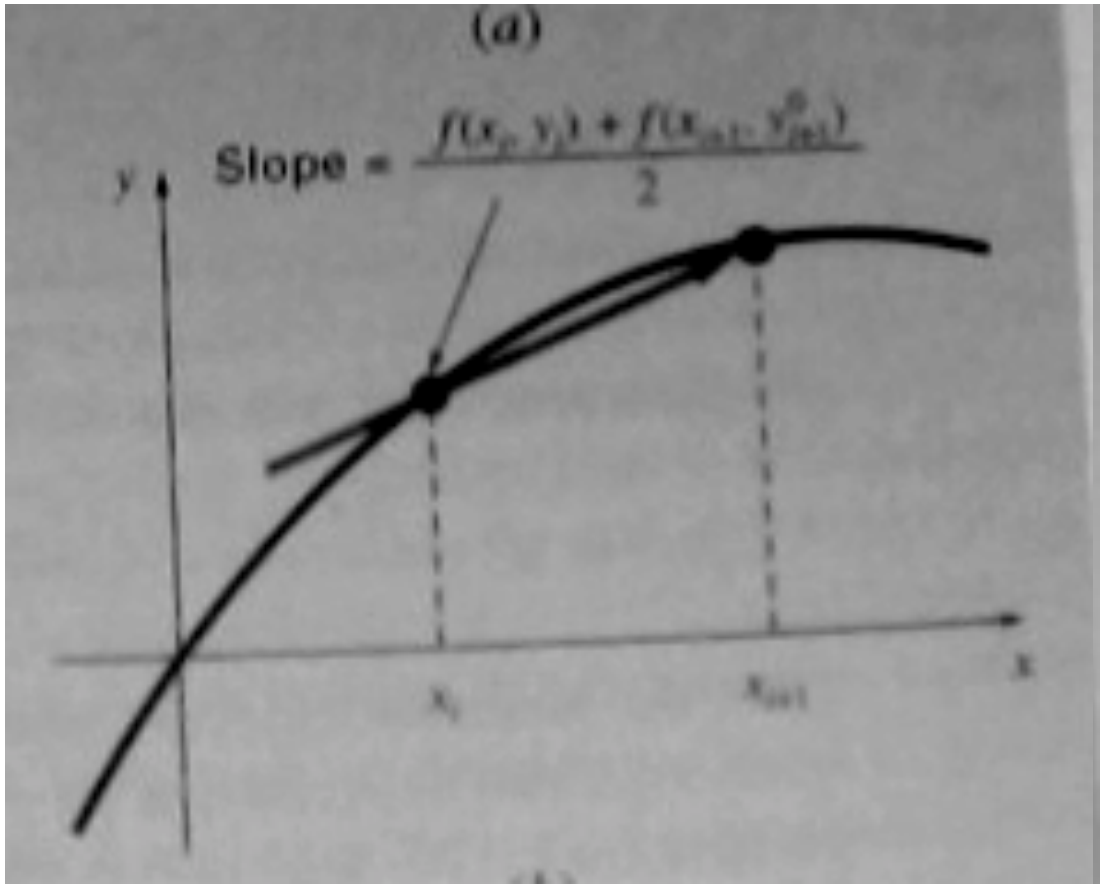
$$y_{i+1}^{0} = y_i + f(t_i, y_i)h$$

- New predictor:

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

- Improves accuracy of predictor from $O(h^2)$ to $O(h^3)$

Non-self-starting Heun



Predictor & Midpoint Rule

$$\int_{y_{i-1}}^{y_{i+1}} dy = \int_{t_{i-1}}^{t_{i+1}} f(t,y)dt$$

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

$$y_{i+1} \cong y_{i-1} + 2hf(t_i, y_i)$$

Corrector & Trapezoidal Rule

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

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

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

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

NSS Heun Error

- Use knowledge of error of midpoint and trapezoidal rules to determine $O(h^3)$ for both predictor and corrector
- Can relate error of predictor and corrector to get error estimate for a step:

$$E_c = -\frac{y_{i+1}^0 - y_{i+1}^{corrected}}{5}$$

- Use error to modify corrector and/or adaptively adjust step size

More multi-step methods

- Open formula used as predictor to obtain initial estimate, then closed formula used to correct the predictor
- Can use higher-order predictors and correctors…

Newton-Cotes formulas

- For **open** formulas:

$$y_{i+1} = y_{i-n} + \int_{t_{i-n}}^{t_{i+1}} f_n(t,y)dt$$

where $f_n(t,y)$ is an n-point interpolating polynomial (e.g., n=1 yields midpoint rule)

- **Closed**:

$$y_{i+1} = y_{i-n+1} + \int_{t_{i-n+1}}^{t_{i+1}} f_n(t,y)dt$$

n=2 yields Trapezoidal rule; n=3 yields Simpson's rule; etc.

Adams-Bashforth Formulas

- Open formulas
- Use **forward** Taylor series expansion around t_i and **backward** differences to approximate derivatives of f

$$y_{i+1} = y_i + f_i h + \frac{f_i'}{2} h^2 + \frac{f_i''}{6} h^3 + \ldots$$

$$\text{Approximate } f_i' \text{ as } f_i' = \frac{f_i - f_{i-1}}{h} + \frac{f_i''}{2} h^2 + O(h^2)$$

Adams-Moulton Formulas

- Closed formulas
- Use **backward** Taylor series expansion around t_i and **backward** differences to approximate derivatives of f

$$y_i = y_{i+1} - f_{i+1}h + \frac{f'_{i+1}}{2}h^2 + \frac{f''_{i+1}}{6}h^3 + \ldots$$

$$\text{Approximate } f'_{i+1} \text{ as } f'_{i+1} = \frac{f_{i+1} - f_i}{h} + \frac{f''_{i+1}}{2}h^2 + O(h^2)$$

About the Adams formulas

- Choice of truncation point for Taylor Series estimate yields formulas of different order
- Adams formula of order n incurs truncation error of $O(h^{n+1})$
- Can couple Adams-Bashforth predictor with Adams-Moulton corrector
- 4^{th}-order Adams Method: very popular

Comments on Multistep Methods

- Non-self-starting
- Changing step size is complicated
- Predictor and corrector yield good local error estimate
- Implicit** methods have greater stability than explicit methods, but require iterative correction
  **e.g., Adams-Moulton by itself is implicit
- If order >2, an implicit method is never unconditionally stable
- Especially effective for stiff equations

Review of IVP methods

- **Euler's**: 1-step, O(h), 1st-order R.K.
- **Heun**: 1-step, $O(h^2)$, 2nd-order R.K., iterative correction step
- **Second-order Ralston**: 1-step, $O(h^2)$, optimal 2nd-order R.K.
- **4th-order Runge-Kutta**: $O(h^4)$, very popular
- **Embedded** Runge-Kutta: $O(h^5)$, allows efficient step-size adjustment
- **Backward Euler**: O(h), stable, usually requires root-finding
- **Non-self-starting Heun**: $O(h^3)$, multi-step, iterative correction
- **Fourth-order Adams**: $O(h^5)$, multi-step, popular

Boundary-Value Problems

IVPs vs BVPs

- An nth-order ODE requires n auxiliary conditions
- Can transform nth-order ODE into system of n first-order ODEs
- Initial Value Problems
  - All n conditions specified at same value of dependent variable (e.g., t_0)
- Boundary Value Problems
  - Conditions known at different values of t (often extreme values)
  - **Can't** start with one value of t for which we know the solution!

Example BVP

- $y'' = F/m$
- IVP: $y=0$ at $t=0$, $y' = 0$ at $t=0$
- BVP: $y=0$ at $t=0$, $y=10$ at $t=100$
  - or $y = 0$ at $t = 0$, $y'=.3$ at $t=100$

Translating higher-order ODE into system of equations

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

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

Two-point BVP for second-order scalar ODE

$$u'' = f(t, u, u'), \qquad a < t < b$$

with BC

$$u(a) = \alpha, \qquad u(b) = \beta$$

is equivalent to first-order system of ODEs

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ f(t, y_1, y_2) \end{bmatrix}, \qquad a < t < b$$

with separated linear BC

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_1(a) \\ y_2(a) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_1(b) \\ y_2(b) \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$