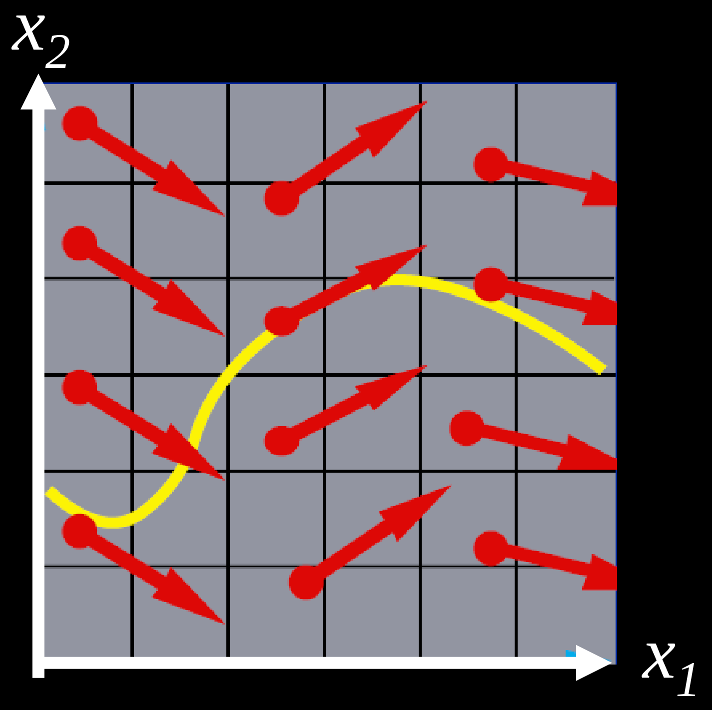
Differential Equation Basics (2)



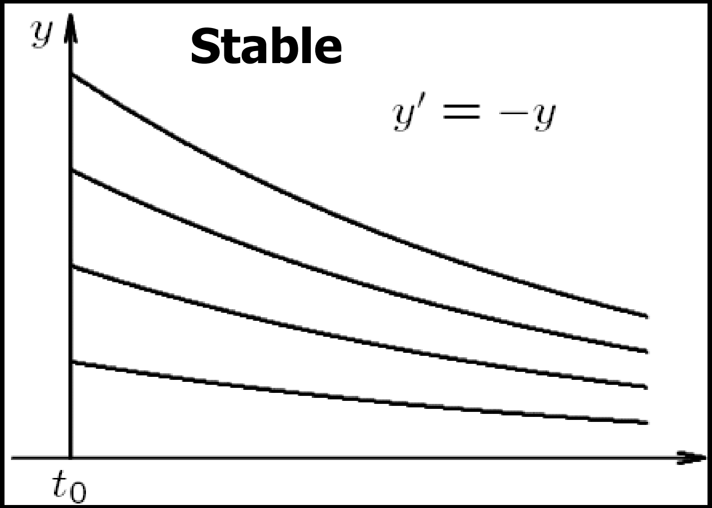
Outline

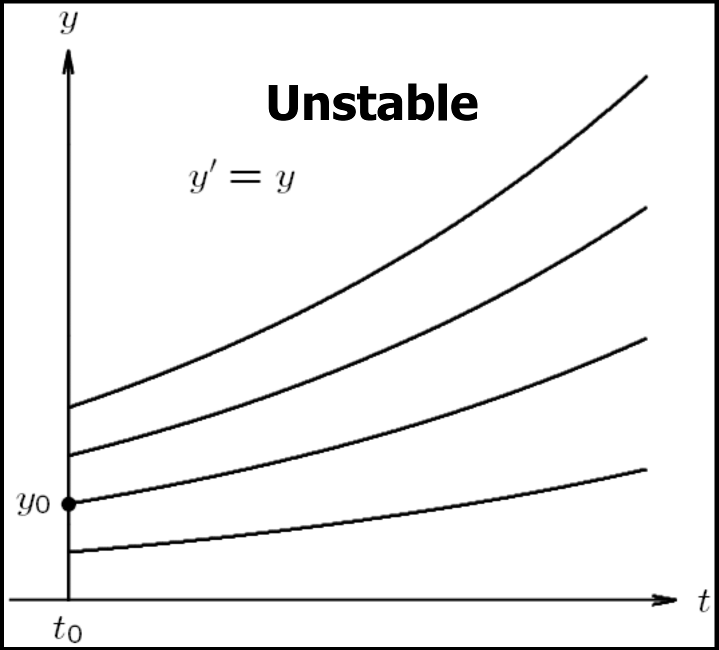
- Stability of ODE
- Propagation error
- Stiff system
- Implicit method

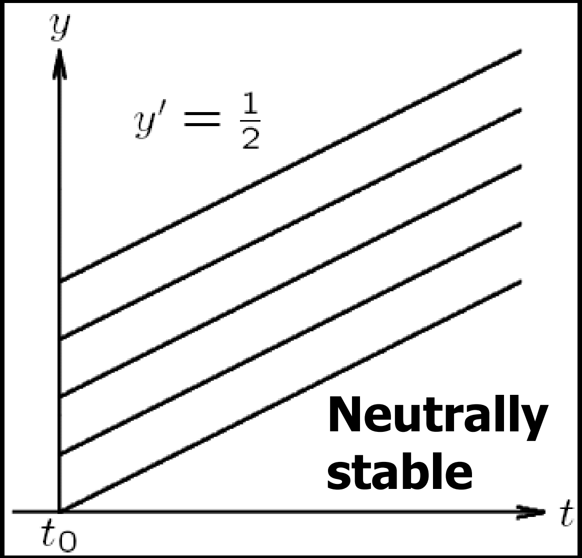
Stability of Analytic Solution of ODE

- Stable if solutions resulting from perturbations of initial value remain close to original solution
- Unstable if solutions resulting perturbations diverges away from original solution without bound
- Neutrally stable if solutions resulting from perturbations are neither stable or unstable

Example: Stability of ODE







Stability of Numerical Solution of ODE

- Numerical solution is stable if small perturbations do not cause resulting numerical solutions to diverge from each other without bound
- Divergence of numerical solutions could be caused by instability of analytical solution to ODE, but can also be due to numerical method itself, even when solutions to ODE is stable

Determining Stability and Accuracy

- Simple approach to determining stability and accuracy of numerical method is to apply it to scalar ODE $x' = Ax$, where A is (possibly complex) constant

- For a given numerical method, we can
  - Determine stability by characterizing growth of numerical solution
  - Determine accuracy by comparing exact and numerical solutions

Example: Euler's Method

- Applying Euler's method to $x' = Ax$ using fixed step size h, we have

$$x_{n+1} = x_n + hAx_n = (1 + Ah)x_n$$

$$x_n = (1 + Ah)^n x_0$$

- If Re(A) < 0, exact solution is stable
- Numerical solution is also stable if $|1 + Ah| < 1$
  - If A<0 and is real, we must have h<-2/A for Euler's method to be stable

Propagated Error

- Error made early in the process will also affect the late computations—the early error will be propagated

- Propagated error analysis is not easy

- We only analyze the propagated error for Euler's method here

Propagated Error—Euler's method

- Numerical solution by Euler's method

$$X_{n+1} = X_n + hf(t_n, X_n)$$

- Analytic solution using Taylor series

$$x_{n+1} = x_n + hf(t_n, x_n) + \frac{h^2}{2} x''(\xi_n), \qquad t_n < \xi_n < t_n + h$$

$$e_{n+1} = x_{n+1} - X_{n+1} = x_n - X_n + h[f(t_n, x_n) - f(t_n, X_n)] + \frac{h^2}{2} x''(\xi_n)$$

$$= e_n + h \frac{f(t_n, x_n) - f(t_n, X_n)}{x_n - X_n} (x_n - X_n) + \frac{h^2}{2} x''(\xi_n)$$

$$= e_n + hf_x(t_n, \eta_n) e_n + \frac{h^2}{2} x''(\xi_n), \qquad \eta_n \text{ between } x_n,\ X_n$$

Propagated Error—Euler's method

$$e_{n+1} = e_n + hf_x(t_n,\eta_n)e_n + \frac{h^2}{2}x''(\xi_n), \qquad \eta_n \text{ between } x_n,\ X_n$$

$$e_{n+1} \le (1+hK)e_n + \frac{h^2}{2}x''(\xi_n), \quad \text{where} \left| f_x(t_n,\eta_n) \right| \text{is bounded by } K$$

$$e_1 \le (1+hK)e_0 + \frac{h^2}{2}x''(\xi_0) \xrightarrow{e_0 = 0} e_1 \le \frac{h^2}{2}x''(\xi_0)$$

$$e_2 \le (1+hK)\frac{h^2}{2}x''(\xi_0) + \frac{h^2}{2}x''(\xi_1) = \frac{h^2}{2}[(1+hK)x''(\xi_0) + x''(\xi_1)]$$

$$e_n \le \frac{h^2}{2}[(1+hK)^{n-1}x''(\xi_0) + (1+hK)^{n-2}x''(\xi_1) + \ldots + x''(\xi_n)]$$

Propagated Error—Euler's method

- Global error is the sum of propagated error and local error

$$e_{n+1} \le (1+hK)e_n + \frac{h^2}{2}x''(\xi_n)$$

- Truncation error at each step is propagated to every later step with a growth factor (1+*hK*) each time!

$$e_n \le \frac{h^2}{2}[(1+hK)^{n-1}x''(\xi_0) + (1+hK)^{n-2}x''(\xi_1) + ... + x''(\xi_n)]$$

Propagated Error—Euler's method

- Error does not grow if $|1 + hK| < 1$

- This can be generated to higher-order case

$$\dot{\mathbf{x}} = f(\mathbf{x}, t)$$

Jacobian of f: $J_f(i, j) = \dfrac{\partial f_i}{\partial x_j}$

$$e_{n+1} \leq (I + hJ_f)e_n + (\text{local error at step } n+1)$$

- Error does not grow if the norm of all eigenvalues of (I+hJ_f) less than 1

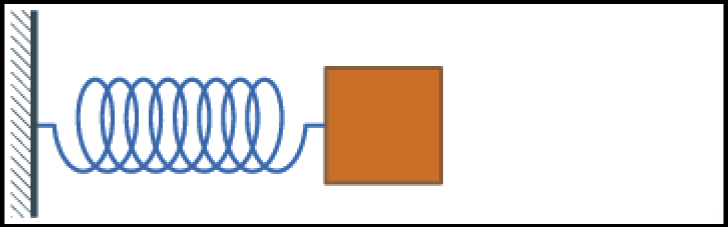
Stability of Numerical Methods for ODEs

In general, growth factor depends on

- Numerical method, which determines form of growth factor
- Step size *h*
- ODE, which determines Jacobian J_f

Stiff Equations

- Let's consider a simple ODE *x'=-kx* **Damper system**
- *k* can be viewed as a stiffness constant
- Some Physics: Spring-Damper System



Spring force $F_s = -k_s x$

Damping force: reduce the amplitude of oscillation

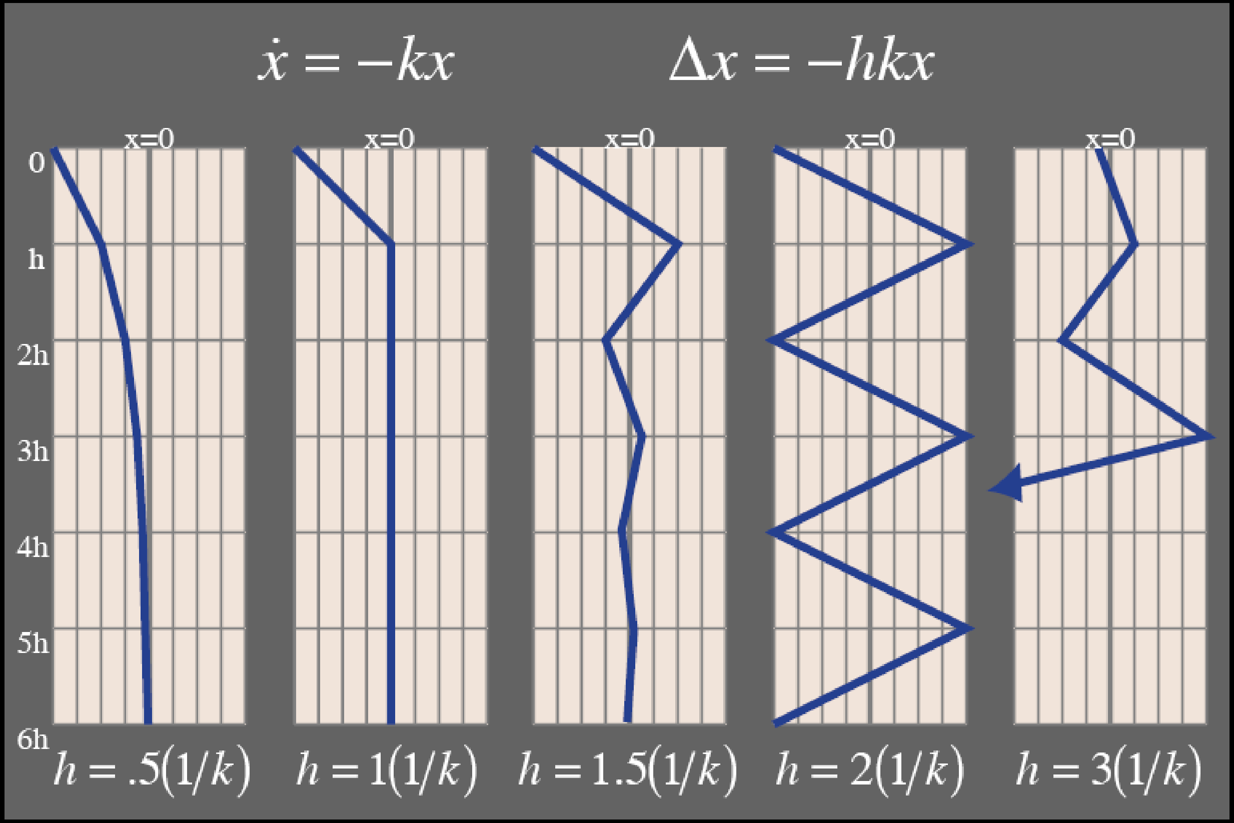
$$F_d = -cv$$

$$ma = -cv$$

$$\dot{v} = -\frac{c}{m}v$$

Step size of Euler's method is limited by k

- If $h > 2/k$, explode!
- For a big k, h needs to be very small!



$$x_{n+1} = x_n + hf(x_n, t_n)$$
$$= x_n + h(-kx_n)$$
$$= (1 - hk)x_n$$

For convergence:

$$r = |1 - hK| < 1$$

Stiff Equations

- In more complex system, step size is limited by the largest k. One stiff spring can screw it up for everyone else
- Systems that have some big k's mixed in are called stiff systems
- Remedy to stiff equations
  - Using small step size → very inefficient
  - Implicit methods

Implicit Method

- Euler's method is explicit in that it uses only information at time t_n to advance solution to time t_{n+1}

- Larger stability region can be achieved using information at time t_{n+1}, which makes method implicit

$y=f(t)$: y is an explicit function of t
$f(y,t)=0$: y is an implicit function of t

Backward Euler Method

- Explicit Euler Method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + hf(\mathbf{x}_n, t_n)$$

- Implicit (Backward) Euler Method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + hf(\mathbf{x}_{n+1}, t_{n+1})$$

- Example: *x'=-kx*

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h(-k\mathbf{x}_{n+1})$$

$$\mathbf{x}_{n+1} = \frac{1}{(1+hk)}\mathbf{x}_n$$

In this case,
stable for any $h > 0$

Implicit Euler Method: $\mathbf{x}_{n+1} = \mathbf{x}_n + hf(\mathbf{x}_{n+1}, t_{n+1})$

- We must evaluate f with $\mathbf{x}_{n+1}$ before we know its value
- Therefore, we need to solve algebraic equation to determine $\mathbf{x}_{n+1}$
  - Root finding
  - Approximate f by $\dfrac{\partial f}{\partial x}$

Example: solving a *k*th-order ODE

$$\frac{d^{(k)}y}{dt} = f(y^{(k-1)}, y^{(k-2)}, \ldots, y', y, t)$$

- Convert to a system of 1st-order equations

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t))$$

- Implicit method

Convert to a system of 1st-order ODE

- Given *k*-th order ODE

$$\frac{d^{(k)}y}{dt} = f(y^{(k-1)}, y^{(k-2)}, ..., y', y, t)$$

- Define

$$x_1(t) = y$$

$$x_2(t) = y'$$

$$x_3(t) = y''$$

$$x_k(t) = y^{(k-1)}$$

- Original ODE equivalent to first order system

$$\begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_{k-1} \\ x'_k \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ x_k \\ f(y^{(k-1)}, \cdots, y', y, t) \end{bmatrix}$$

Example: *k*th-order ODE

$$\frac{d}{dt}\mathbf{X}(t) = \dot{\mathbf{X}}(t) = f(\mathbf{X}(t))$$

$$\mathbf{X}_{n+1} = \mathbf{X}_n + hf(\mathbf{X}_{n+1})$$

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_k} \\ \frac{\partial f_2}{\partial x_1} & & \cdots & \frac{\partial f_2}{\partial x_k} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_k}{\partial x_1} & \frac{\partial f_k}{\partial x_2} & \cdots & \frac{\partial f_k}{\partial x_k} \end{bmatrix}$$

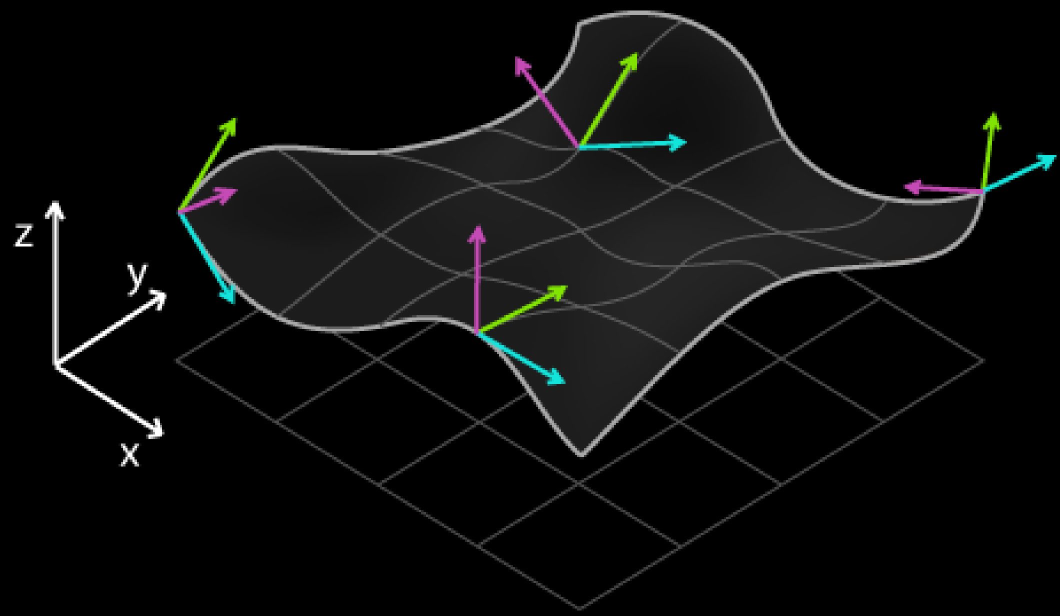
$$f(\mathbf{X}_{n+1}) \approx f(\mathbf{X}_n) + \mathbf{J_f}(\mathbf{X}_{n+1} - \mathbf{X}_n)$$

$$\mathbf{X}_{n+1} = \mathbf{X}_n + h(f(\mathbf{X}_n) + \mathbf{J_f}(\mathbf{X}_{n+1} - \mathbf{X}_n))$$

$$\mathbf{X}_{n+1} = \mathbf{X}_n + (\mathbf{I} - h\mathbf{J_f})^{-1} hf(\mathbf{X}_n)$$

Side Note: Jacobian

- F(u,v) = [x, y, z]
  - x(u,v) = u
  - y(u,v) = v
  - z(u,v) = f(u,v)



$$J^T(u,v) = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{\partial f}{\partial u} \\ 0 & 1 & \frac{\partial f}{\partial v} \end{bmatrix} = \begin{bmatrix} \mathbf{t_u} \\ \mathbf{t_v} \end{bmatrix}$$

$$\mathbf{t_u} \times \mathbf{t_v} = \mathbf{n} = \begin{bmatrix} -\frac{\partial f}{\partial u} & -\frac{\partial f}{\partial v} & 1 \end{bmatrix}$$