UNIVERSITY OF TEXAS ARLINGTON

PHYS 5319-001: Math Methods in Physics III

Numerical Solution of ODE

Instructor:	Dr. Qiming Zhang
Office:	CPB 336
Phone:	817-272-2020
Email:	zhang@uta.edu

Ordinary Differential Equations

Standard form:

$$\frac{d\vec{y}}{dt} = \vec{f}(t, \vec{y})$$

Or a set of coupled 1st order ODEs:

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, y_2, \dots, y_N)$$

$\vec{f}$: known
$\vec{y}$: unknown except initial value

e.g. SHO:

$$\boxed{\frac{d^2x}{dt^2} + \frac{k}{m}x = 0}$$

Define: $y^{(1)}(t) \equiv x(t);\ y^{(2)}(t) \equiv \frac{dx}{dt}$

$$\Rightarrow \quad \frac{dy^{(1)}(t)}{dt} = y^{(2)}(t)$$

$$\frac{dy^{(2)}(t)}{dt} = -\frac{k}{m}y^{(1)}(t)$$

ODEs with initial values

- Starting with 1st order ODE

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

with initial condition $y(0) = y_0$.

- The Taylor series expansion of ~~x~~ y around t gives

$$y(t + \Delta t) = y(t) + \frac{dy}{dt}\Delta t + \frac{1}{2}\frac{d^2 y}{dt^2}(\Delta t)^2 + \cdots \quad (1)$$

Assume a reasonably smooth function $y(t)$ and a small interval Δt, we can propagate $y(t)$ to $y(t + \Delta t)$ to any accuracy desired, as long as we know the derivatives of $y(t)$.

Numerical solution: Euler method

- When Δt is small, it is a good approximation by simply ignoring the second and higher orders of Δt in Eq. (1) as

$$y(t+\Delta t) \approx y(t) + \frac{dy}{dt}\Delta t = y(t) + f(t, y(t))\Delta t \quad (2)$$

- This is the First-order Approximation – Euler method.
- Or in general: $\vec{y}_{n+1} = \vec{y}_n + h\vec{f}(t_n, \vec{y}_n) + O(h^2)$

- Obtaining $y(t+\Delta t)$ from $y(t)$ with Eq. (2), we can estimate $y(t+2\Delta t)$ from $y(t+\Delta t)$ as

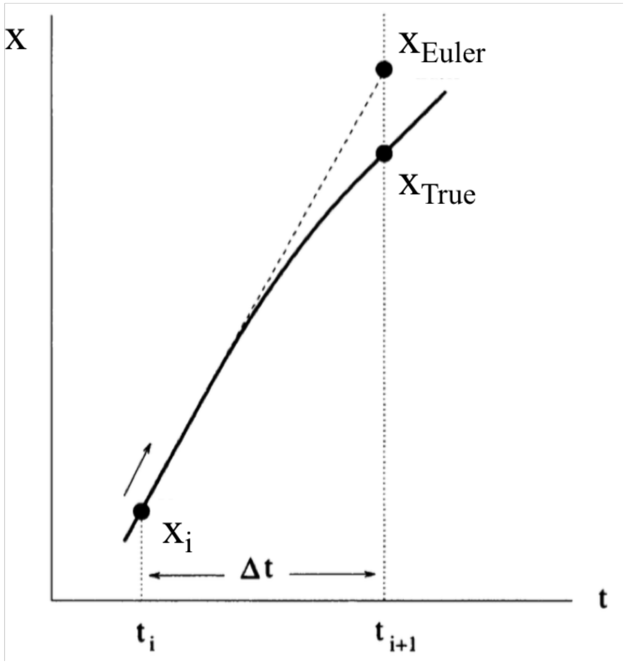
$$y(t+2\Delta t) = y(t+\Delta t) + f(t+\Delta t, y(t+\Delta t))\Delta t$$

- Iteratively, we can obtain $y(t+N\Delta t)$. $(N\Delta t \rightarrow the\ interested)$

- How to interpret Euler approximation geometrically?

$$x(t+\Delta t) \approx x(t) + \frac{dx}{dt}\Delta t = x(t) + f(x(t),t)\Delta t$$

Here *x(t)* ↔ *y(t)*



Error: $O(\Delta t^2)$

Graphically, the Euler method corresponds to the linear extrapolation up to $(t+\Delta t)$ by using a line tangent to $x(t)$ at t.

A larger interval Δt, would cause a larger discrepancy between x_{true} and x_{Euler}.

How to systematically improve it?

How to systematically improve it?

A smaller Δt? Of course! However, the cost would be the reduction of the efficiency…
Another obvious answer is to keep to higher orders in the Taylor expansion. For example, we could take an iterative approach

$$x(t_{i+1}) \approx x(t_i) + \underline{\frac{dx}{dt}}\Delta t + \frac{1}{2}\frac{d^2x}{dt^2}\Delta t^2$$

f

$$= x(t_i) + f_i\Delta t + \frac{1}{2}\frac{d^2x}{\underline{dt^2}}\Delta t^2 \qquad (1)$$

Remember: $$\frac{d}{dt}x(t) = f(x,t)$$

Then $$\frac{d}{dt}\left(\frac{d}{dt}x(t)\right) = \frac{d}{dt}(f(x,t))$$

$$\frac{d^2}{dt^2}(x(t)) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial t}\frac{dt}{dt} = f(x,t)\frac{\partial f}{\partial x} + \frac{\partial f}{\partial t} \qquad (2)$$

(2) $\Rightarrow$ (1), we obtain the approximation of $x(t_{i+1})$ until the second order in the Taylor expansion as

$$x(t_{i+1}) \approx x(t_i) + f_i \Delta t + \frac{1}{2}\left[f_i \left.\frac{\partial f}{\partial x}\right|_i + \left.\frac{\partial f}{\partial t}\right|_i \right] (\Delta t)^2 ,$$

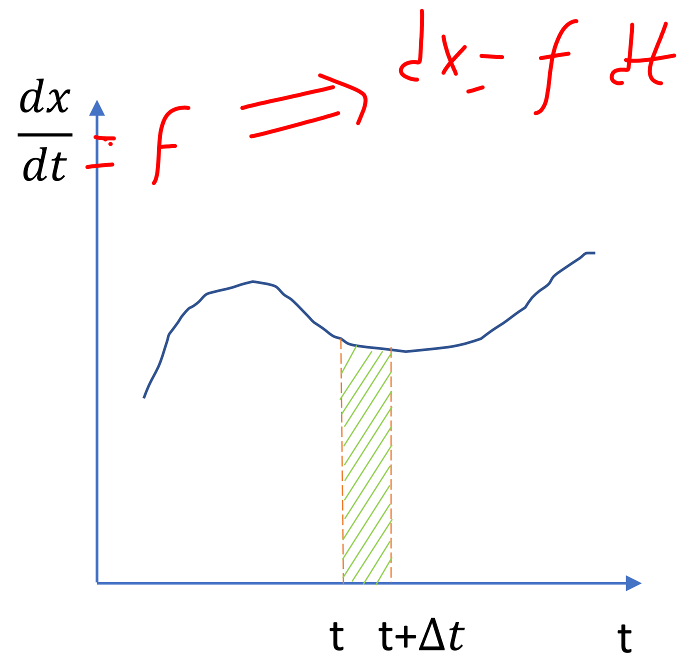
where the subscript *i* on the derivatives indicates evaluation at x_i and t_i. This would produce a higher order approximation locally, however, the partial derivatives of *f(x, t)* must be evaluated.

How about other methods without evaluating the partial derivatives of *f(x, t)*?

One possible approach is to consider the problem as the integration of dx/dt in the range of t to $\mathrm{t} + \Delta t$.

What is the integration of $\int_t^{t+\Delta t} \frac{dx}{dt} dt$?

It equals $x(t+\Delta t) - x(t)$, or it can be taken as the area covered by the $\frac{dx}{dt}$ curve and the t axis, with boundaries of t=t and t= t+Δt.



This area can be irregular shape. How to compute it?

-- The mean value theorem

if f is a continuous function on the closed interval [a, b] and differentiable on the open interval (a, b), then there exists a point c in (a, b) such that the tangent at c is parallel to the secant line through the endpoints (a, f(a)) and (b, f(b)), that is

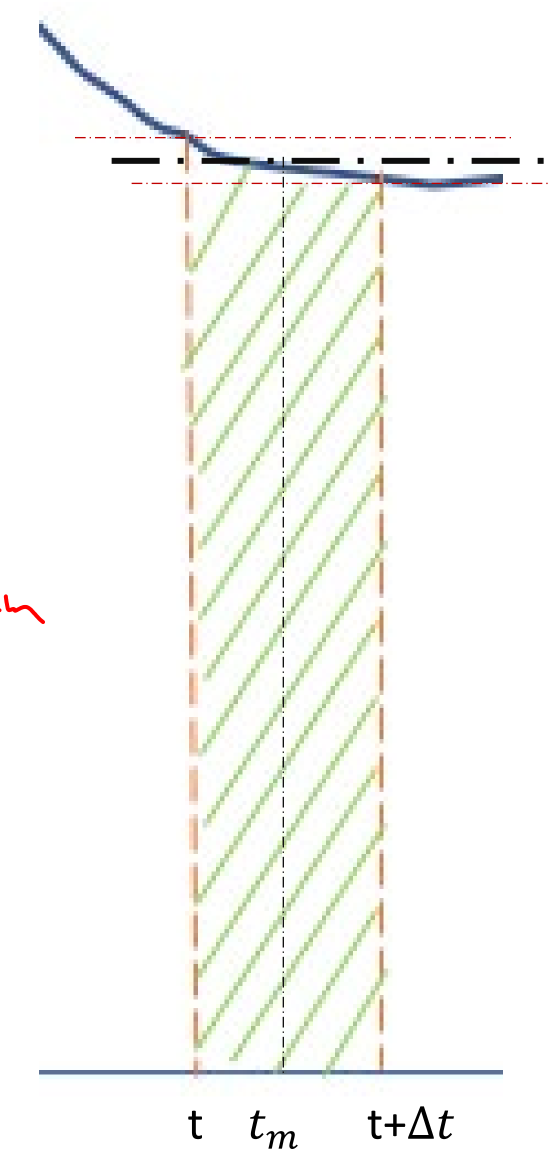
$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Or

$$f(b) - f(a) = f'(c) * (b - a)$$
$$= the\ integration\ area$$

Here $x(t) \leftrightarrow y(t)$

That just means there exists a point t_m between t and t+Δt, and the rectangular area made by the height $\frac{dy}{dt} = \frac{dy}{dt}|_{t=t_m}$, equals to the irregular area $\int_t^{t+\Delta t} \frac{dy}{dt} dt$:



$$y(t+\Delta t) - y(t) = \frac{dy}{dt}|_{t=t_m} \Delta t$$

$= f|_{t=t_m}$

Or
$$y(t+\Delta t) = y(t) + \frac{dy}{dt}|_{t=t_m} \Delta t$$

If we could find the point t_m, we then can compute $y(t+\Delta t)$ from $y(t)$ accurately!

Yet, it is not easy to know the exact value of t_m and we will try to estimate it and $\frac{dy}{dt}|_{t=t_m}$ effectively.

- A popular estimation method is the second-order Runge-Kutta method

$$y(t + \Delta t) = y(t) + f(t', y')\Delta t$$

where

$$y' = y(t) + \frac{1}{2} f\big(t, y(t)\big)\Delta t$$

$\leftarrow$ Euler

$$t' = t + \frac{1}{2}\Delta t$$

$O(h^2)$

- In other words, the slope $\frac{dy}{dt}|_{t=t_m}$ is estimated as the value $f(t', y')$ where t' is the midpoint of the interval and y' is the Euler approximated value of y at t'.

- QDE to be solved

$$\frac{dy}{dt} = f(t,y) \quad \Rightarrow \qquad dy = f(t,y)dt$$

Integrate on both sides from n-th to (n+1)-th point:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t,y)dt$$

$$\int_n^{n+1} dy = y\Big|_n^{n+1} = y_{n+1} - y_n$$

- Since the idea is using the midpoint. Take Taylor series at it:

$$f(t,y) \approx f\left(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}\right) + \left(t - t_{n+\frac{1}{2}}\right)\frac{df}{dt}\Big|_{t_{n+1/2}} + O(h^2)$$

- Substitute f(t,y) into the integral *and do the integration*

Note: $\int_{t_n}^{t_{n+1}}\left(t - t_{n+\frac{1}{2}}\right)dt = \frac{1}{2}t^2\Big|_{t_n}^{t_{n+1}} - t_{n+\frac{1}{2}}(t_{n+1} - t_n) = 0$

$$t_{n+\frac{1}{2}} = \frac{1}{2}(t_n + t_{n+1})$$

$$= \frac{1}{2}\left(t_{n+1}^2 - t_n^2\right)$$

- So $\int_{t_n}^{t_{n+1}} f(t,y)dt = f\left(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}\right)h + O(h^3)$

$$\Rightarrow \qquad \boldsymbol{y_{n+1} \approx y_n + hf\left(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}}\right)} \qquad +O(h^3)$$

The price for the improved precision: to evaluate *f(t,y)* in the middle of an interval by Euler's algorithm: $y_{n+\frac{1}{2}} \approx y_n + \frac{dy}{dt}\frac{h}{2} = y_n + \frac{h}{2} f(t_n, y_n)$

In summary, the 2nd-order Runge-Kutta algorithm:

$$\vec{y}_{n+1} \approx \vec{y}_n + \vec{K}_2$$

$$\vec{K}_2 = h\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{K}_1}{2}\right)$$

$$\vec{K}_1 = h\vec{f}(t_n, \vec{y}_n)$$

$\vec{f}$: derivative function(s); known

$\vec{y}$: unknown; only the initial value(s) required

The algorithm is self-starting

Code (FORTRAN)

```
c second-order Runge-Kutta subroutine
        Subroutine rk2(t, dt, y, n)
c declarations
        Real*8 deriv, h, t, dt, y(1)
        Real*8 k1(5), k2(5), t1(5)
        Integer i, n
        h=dt/2.0
        Do i = 1,n
           k1(i) = dt * deriv(t, y, i)
           t1(i) = y(i) + 0.5*k1(i)
        enddo
        Do i = 1,n
           k2(i) = dt * deriv(t+h, t1, i)
           y(i) = y(i) + k2(i)
        enddo
        Return
        End
```

$$\vec{y}_{n+1} \approx \vec{y}_n + \vec{K}_2$$

$$\vec{K}_2 = h\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{K}_1}{2}\right)$$

$$\vec{K}_1 = h\vec{f}(t_n, \vec{y}_n)$$

f

$\omega^2 = k/m$

$$\boxed{\frac{d^2x}{dt^2} + \frac{k}{m}x = 0}$$

```
c function which returns the derivatives
        Function deriv(x, temp, i)
C 2 SHO function components: dx/dt = v(t),   dv/dt = -w**2*x(t)
C the second term is damping term
        Real*8 deriv, x, temp(2), omega, alpha
        Integer i
        data omega /3.13d0/
        data alpha /0.5d0/
c
        If (i .EQ. 1) deriv=temp(2)
        If (i .EQ. 2) deriv=-temp(1) * omega**2
        Return
        End
```

$f^{(1)}$

$$\frac{dy^{(1)}(t)}{dt} = y^{(2)}(t)$$

$$\frac{dy^{(2)}(t)}{dt} = -\frac{k}{m}y^{(1)}(t)$$

$f^{(2)}$

Fourth-order Runge-Kutta $\boldsymbol{O(h^5)}$

On excellent balance of power, precision, and programming simplicity.

$$\vec{y}_{n+1} = \vec{y}_n + \frac{1}{6}(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4)$$

$$\vec{K}_1 = h\vec{f}(t_n, \vec{y}_n)$$ — Euler

$$\vec{K}_2 = h\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{K}_1}{2}\right)$$ — 2nd-order

$$\vec{K}_3 = h\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{K}_2}{2}\right)$$

$$\vec{K}_4 = h\vec{f}(t_n + h, \vec{y}_n + \vec{K}_3)$$

4 gradient (**K**'s) terms; $\Rightarrow$ better approximation to *f(t, y)* at midpoint.
The 4 K's can be programmed easily just as 4 functions/subroutines.

Code (FORTRAN)

```
c fourth-order Runge-Kutta subroutine
        Subroutine rk4(x, xstep, y, n)
c declarations
        Real*8 deriv, h, x, xstep, y(5)
        Real*8 k1(5), k2(5),k3(5), k4(5), t1(5), t2(5), t3(5)
        Integer i, n
        h=xstep/2.0
        Do i = 1,n
           k1(i) = xstep * deriv(x, y, i)
           t1(i) = y(i) + 0.5*k1(i)
        Enddo
        Do i = 1,n
           k2(i) = xstep * deriv(x+h, t1, i)
           t2(i) = y(i) + 0.5*k2(i)
        Enddo
        Do i = 1,n
           k3(i) = xstep * deriv(x+h, t2, i)
           t3(i) = y(i) + k3(i)
        Enddo
        Do i = 1,n
           k4(i) = xstep * deriv(x+xstep, t3, i)
           y(i) = y(i) + (k1(i) + (2.*(k2(i) + k3(i))) + k4(i))/6.0
        Enddo
c
        Return
        End
```

t h y_n ($\bar{y}_{n+1}$)

$$\vec{y}_{n+1} = \vec{y}_n + \frac{1}{6}(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4)$$

$$\vec{K}_1 = h\vec{f}(t_n, \vec{y}_n)$$

$$\vec{K}_2 = h\vec{f}\left(t_n + \frac{h}{2}, \underbrace{\vec{y}_n + \frac{\vec{K}_1}{2}}_{t1}\right)$$

$$\vec{K}_3 = h\vec{f}\left(t_n + \frac{h}{2}, \underbrace{\vec{y}_n + \frac{\vec{K}_2}{2}}_{t2}\right)$$

$$\vec{K}_4 = h\vec{f}\left(t_n + h, \vec{y}_n + \vec{K}_3\right)$$