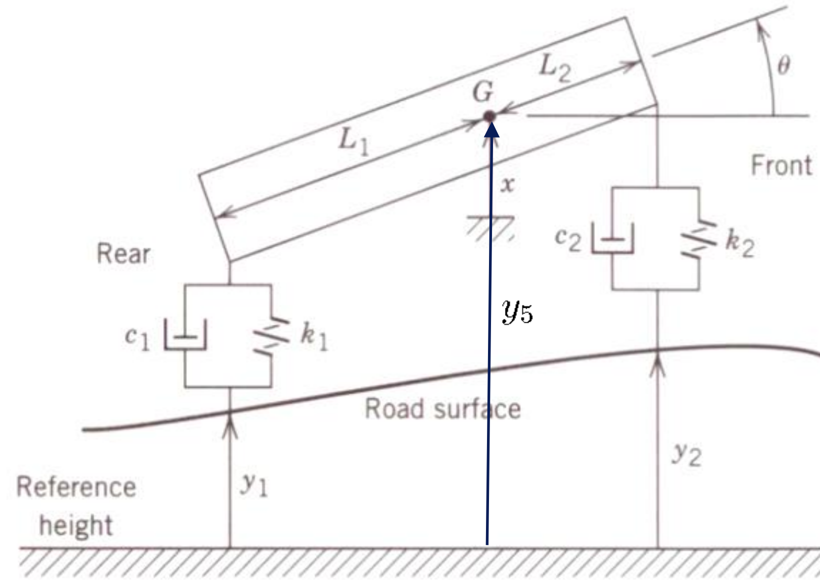
상미분 방정식의 풀이 소개
(Introduction to Ordinary Differential Equations)

서강대학교 공과대학 컴퓨터공학과
임 인 성

Vehicle Suspension Design



2nd-order ODE system

$$
\begin{aligned}
m \cdot y_5'' &= -(k_1+k_2)y_5 - (c_1+c_2)y_5' + (k_1L_1 - k_2(L-L_1))\theta \\
&\quad + (c_1L_1 - c_2(L-L1))\theta' + (k_1y_1 + k_2y_2 + c_1y_1' + c_2y_2') \\
I \cdot \theta'' &= (L_1k_1 - (L-L_1)k_2)y_5 + (L_1c_1 - (L-L_1)c_2)y_5' + \\
&\quad - (L_1^2k_1 + (L-L_1)^2k_2)\theta - (L_1^2c_1 + (L-L_1)^2c_2)\theta' \\
&\quad + (-L_1k_1y_1 + (L-L_1)k_2y_2 - L_1c_1y_1' + (L-L_1)c_2y_2')
\end{aligned}
$$

Ordinary Differential Equations (ODE)

- Differential equations:
  - 독립 변수와 (종속 변수와 종속 변수들의 미분 값들)간의 관계
  - Ordinary: 독립 변수 1개
  - Partial: 독립 변수 2개 이상
  - Order of equations: the highest derivatives involved
- **First-order ODE**

$$\boxed{\text{Given} \begin{Bmatrix} y'(t) = f(t, y) \\ y(t_0) = y_0 \end{Bmatrix}, \text{ find } y = y(t)!}$$

$$y'(t) = f(t, y) = -20y + 7e^{-0.5t},\ y(0) = 5$$

Analytic solution

$$y = 5e^{-20t} + \frac{7}{19.5}\left(e^{-0.5t} - e^{-20t}\right)$$

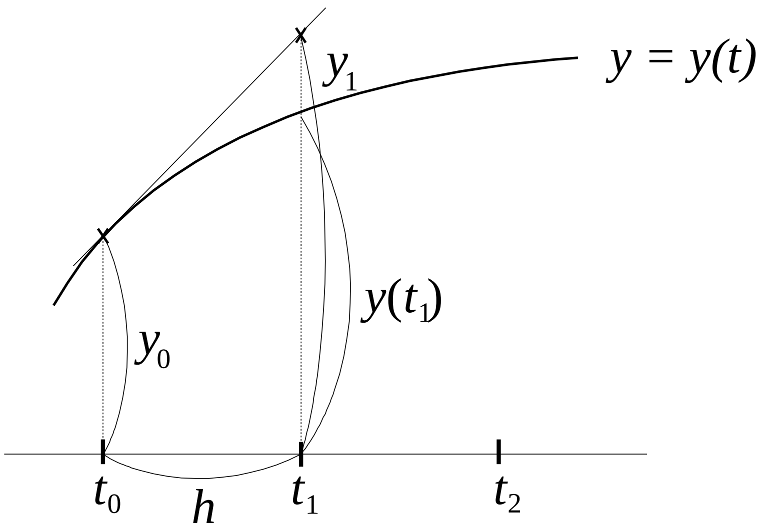
Numerical solution

t	t_0	t_1	t_2	$\cdots$	t_m
y	y_0	y_1	y_2	$\cdots$	y_m

Mesh Points

Euler's Method: 1st Order Taylor Method

$$\text{Problem: } \left\{ \begin{array}{l} y'(t) = f(t, y) \\ y(t_0) = y_0 \end{array} \right\}$$



$$\begin{aligned} y(t) &= y(t_0) + y'(t_0)(t - t_0) \\ &= y_0 + f(t_0, y_0)(t - t_0) \end{aligned}$$

$$\begin{aligned} y_1 &= y_0 + f(t_0, y_0)(t_1 - t_0) \\ &= y_0 + h \cdot f(t_0, y_0) \end{aligned}$$

$$\boxed{y_{i+1} = y_i + h \cdot f(t_i, y_i), i = 0, 1, 2, \cdots}$$

Example : $y'(t) = f(t,y) = -20y + 7e^{-0.5t}, y(0) = 5$

$$\boxed{y_{i+1} = y_i + h(-20y_i + 7e^{-0.5t_i}), i = 0, 1, 2, \cdots}$$

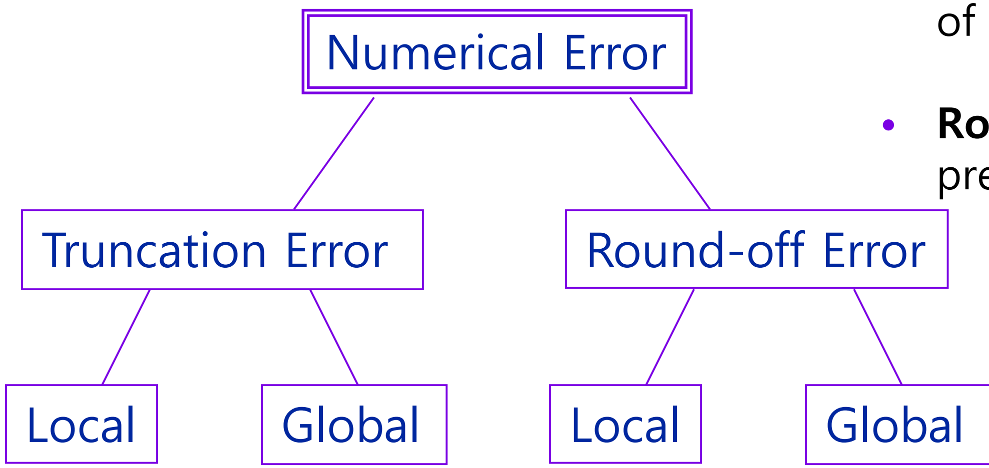
$$\begin{aligned} t_0 &= 0, & y_0 &= y(0) = 5 \\ t_1 &= 0.01, & y_1 &= y_0 + h(-20y_0 + 7e^{-0.5t_0}) = 4.07 \\ &\vdots \end{aligned}$$

t	$h = 0.01$	$h = 0.001$	$h = 0.0001$
0.01	4.07000 (8.693)*	4.14924 (0.769)*	4.15617 (0.076)*
0.02	3.32565 (14.072)	3.45379 (1.259)	3.46513 (0.124)
0.03	2.72982 (17.085)	2.88524 (1.544)	2.89915 (0.153)
0.04	2.25282 (18.440)	2.42037 (1.684)	2.43554 (0.167)
0.05	1.87087 (18.658)	2.04023 (1.722)	2.05574 (0.171)
0.06	1.56497 (18.125)	1.72932 (1.690)	1.74454 (0.168)
0.07	1.31990 (17.119)	1.47496 (1.613)	1.48949 (0.169)
0.08	1.12352 (15.839)	1.26683 (1.507)	1.28041 (0.150)
0.09	0.96607 (14.427)	1.09646 (1.387)	1.10895 (0.138)
0.10	0.83977 (12.979)	0.95696 (1.261)	0.96831 (0.126)

*(error) × 100

1. Magnitudes of errors are approximately proportional to h. (Why?)
2. Further reduction of h without using double precision is not advantageous. (Why?)

Error Analysis



- **Truncation Error** : due to the truncation of Taylor series for approximation
- **Round-off Error** : due to the use of finite precision in calculation

Taylor Series의 의미

$$y(t) = y(t_i) + (t - t_i)y'(t_i) + \frac{(t-t_i)^2}{2} y''(\xi_i)$$

$$y(t_{i+1}) = y(t_i) + h \cdot f(t_i, y_i) + \frac{h^2}{2} y''(\xi_i)$$

$$y(t_{i+1}) \leftarrow y(t_i) + h \cdot f(t_i, y_i), \text{ Error } = o(h^2)$$ ← Local Truncation Error

$$h = \frac{b-a}{n} \rightarrow n = \frac{b-a}{h}$$

Global Truncation Error

$$\boxed{\text{When the entire interval } [a, b] \text{ is done, Error } = o(h^2) \cdot n = o(h)}$$

Taylor Polynomial Approximation

- **Taylor's theorem**

For a function $f \in C^{n+1}[a,b]$, $\quad f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1},$ ⟵ Error term

where $E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}$ for some $\xi = \xi(c,x) \in (\min(c,x), \max(c,x))$.

For a function $f \in C^{n+1}[a,b]$, $\quad f(x+h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!}h^k + E_{n+1},$

where $E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$ for some $\xi = \xi(x,h) \in (x, x+h)$.

- Example

$$\sqrt{1+h} = 1 + \frac{1}{2}h - \frac{1}{8}h^2 + \frac{1}{16}h^3\xi^{-\frac{5}{2}},\ \xi \in (1, 1+h), h > 0, \quad \sqrt{1-h} = 1 - \frac{1}{2}h - \frac{1}{8}h^2 - \frac{1}{16}h^3\xi^{-\frac{5}{2}},\ \xi \in (1+h, 1), h < 0$$

$$\sqrt{1.00001} \approx 1 + 0.5\times10^{-5} - 0.125\times10^{-10} = 1.00000\ 49999\ 87500$$

$$\frac{1}{16}h^3\xi^{-\frac{5}{2}} < \frac{1}{16}10^{-15} = 0.00000\ 00000\ 00000\ 0625$$

Higher-Order Taylor Methods

$$
\begin{aligned}
y(t_{i+1}) &= y(t_i) + \frac{(t_{i+1}-t_i)}{1!}y'(t_i) + \frac{(t_{i+1}-t_i)^2}{2!}y''(t_i) + \frac{(t_{i+1}-t_i)^3}{3!}y^{(3)}(t_i) + \cdots \\
&= y(t_i) + \frac{h}{1!}y'(t_i) + \frac{h^2}{2!}y''(t_i) + \frac{h^3}{3!}y^{(3)}(t_i) + \cdots \\
&= y(t_i) + h \cdot f(t_i, y(t_i)) + \frac{h^2}{2}f'(t_i, y(t_i)) + \frac{h^3}{6}f''(t_i, y(t_i)) + \cdots
\end{aligned}
$$

[Taylor Method of order 1 - Euler]

$$y(t_{i+1}) = y(t_i) + h \cdot f(t_i, y(t_i)) + o(h^2) \longrightarrow \boxed{y_{i+1} = y_i + h \cdot f(t_i, y_i)}$$

[Taylor Method of order 2]

$$y(t_{i+1}) = y(t_i) + h \cdot f(t_i, y(t_i)) + \tfrac{h^2}{2}f'(t_i, y(t_i)) + o(h^3)$$

$$\longrightarrow \boxed{y_{i+1} = y_i + h \cdot f(t_i, y_i) + \tfrac{h^2}{2}f'(t_i, y_i)}$$

[Taylor Method of order n]

???

Example : $y' = y - t^2 + 1 \ (0 \leq t \leq 2),\ y(0) = 0.5$

$$
\boxed{
\begin{aligned}
f(t, y(t)) &= y(t) - t^2 + 1 \\
f'(t, y(t)) &= \frac{d}{dt}(y - t^2 + 1) = y' - 2t = y - t^2 + 1 - 2t \\
f''(t, y(t)) &= \frac{d}{dt}(y - t^2 + 1 - 2t) = y - t^2 - 2t - 1 \\
f'''(t, y(t)) &= \frac{d}{dt}(y - t^2 - 2t - 1) = y - t^2 - 2t - 1
\end{aligned}
}
$$

[Taylor Method of order 2]

$$
\begin{aligned}
y_{i+1} &= y_i + h \cdot f(t_i, y_i) + \frac{h^2}{2} f'(t_i, y_i) \\
&= y_i + h \cdot (y_i - t_i^2 + 1) + \frac{h^2}{2}(y_i - t_i^2 + 1 - 2t_i) \\
&= y_i + h\{(1 + \frac{h}{2})(y_i - t_i^2 + 1) - ht_i\}
\end{aligned}
$$

[Taylor Method of order 4]

$$
\begin{aligned}
y_{i+1} &= y_i + h\{(1 + \frac{h}{2} + \frac{h^2}{6} + \frac{h^3}{24})(y_i - t_i^2) \\
&\quad -(1 + \frac{h}{3} + \frac{h^2}{12})ht_i + 1 + \frac{h}{2} - \frac{h^2}{6} - \frac{h^3}{24}\}
\end{aligned}
$$

Euler's and Higher-Order Taylor Methods

- Euler
  - 장점: 계산이 간단함. 즉 계산량이 적음
  - 단점: first order -> 부정확
- Higher-Order Taylor Methods
  - 장점: higher-order!
  - 단점: $f(t, y)$의 미분 값들을 계산해야 함 -> complicated & time-consuming

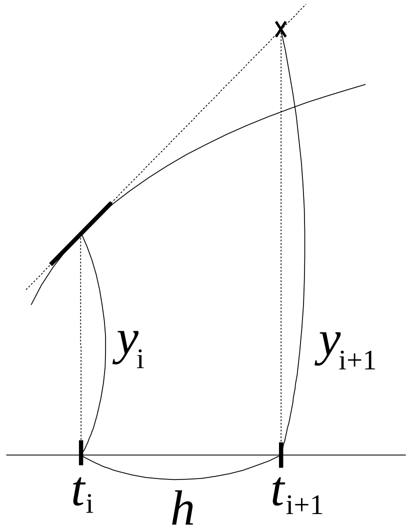
$$f'(t,y) = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y}\frac{dy}{dt}$$

$$f''(t,y) = \frac{\partial[\partial f/\partial t + (\partial f/\partial y)(dy/dt)]}{\partial t} + \frac{\partial[\partial f/\partial t + (\partial f/\partial y)(dy/dt)]}{\partial y}\frac{dy}{dt}$$

  - 이 방법들은 일반적으로 잘 쓰이지 않음

- **문제:** 어떻게 하면 $f(t, y)$ 정보만 사용하여 "higher-order" 방법을 만들 수 있을까?

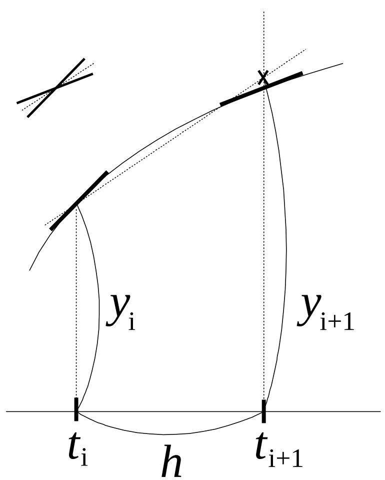
Modified Euler Method



$$\boxed{y_{i+1} = y_i + \Delta y = y_i + s \cdot h}$$

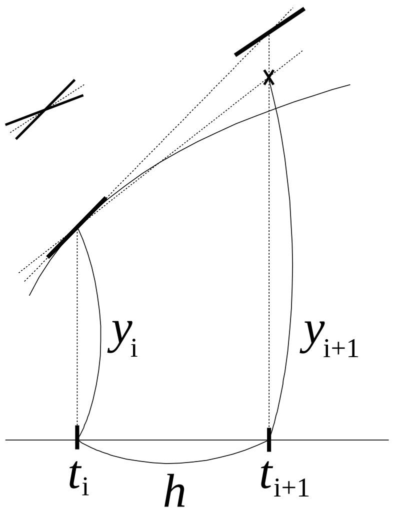
$$\text{Euler: Slope} = f(t_i, y_i)$$

$$\to \text{GTE: } o(h)$$



$$\text{“Better”: Slope} = \frac{f(t_i, y_i) + f(t_{i+1}, y(t_{i+1}))}{2}$$

$$\to \text{GTE: } o(h^2) \quad ???$$



[Predictor-Corrector Scheme]

1. Predictor step:

$$y^*_{i+1} = y_i + h \cdot f(t_i, y_i)$$

2. Corrector step:

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

$$k_1 = f(t_i, y_i)$$
$$k_2 = f(t_i + h, y_i + k_1 \cdot h)$$
$$y_{i+1} = y_i + \tfrac{1}{2}\{k_1 + k_2\} \cdot h$$

Euler Predictor-Corrector Method
or
Second-Order Runge-Kutta Method
or
Modified Euler

Second-Order Runge-Kutta Methods

Find a_1, a_2, p_1, q_{11} such that

$$\boxed{y_{i+1} = y_i + f(t_i, y_i)h + \frac{f'(t_i,y_i)}{2}h^2}(+o(h^3)) \text{ is identical with}$$

$$\boxed{y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h}$$

where $k_1 = f(t_i, y_i)$ and $\boxed{k_2} = f(t_i + p_1 h, y_i + q_{11} k_1 h)$

$$\boxed{f'(t_i, y_i) = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y}\frac{dy}{dt}}$$

$$\boxed{y_{i+1} = y_i + f(t_i, y_i)h + \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt}\right)\frac{h^2}{2}}$$

$$\boxed{k_2 = f(t_i + p_1 h, y_i + q_{11} k_1 h) = f(t_i, y_i) + (p_1 h \frac{\partial f}{\partial t} + q_{11} k_1 h \frac{\partial f}{\partial y}) + o(h^2)}$$

$$\boxed{y_{i+1} = y_i + [a_1 f(t_i, y_i) + a_2 f(t_i, y_i)]h + [a_2 p_1 \frac{\partial f}{\partial t} + a_2 q_{11} f(t_i, y_i)\frac{\partial f}{\partial y}]h^2 + o(h^3)}$$

$$\boxed{a_1 + a_2 = 1,\ a_2 p_1 = \frac{1}{2},\ a_2 q_{11} = \frac{1}{2}}$$

Notice that both have the same truncation error o(h^3)!

[Modified Euler: $a_2 = \frac{1}{2} \rightarrow a_1 = \frac{1}{2}$, $p_1 = q_{11} = 1$]

$$y_{i+1} = y_i + (\tfrac{1}{2}k_1 + \tfrac{1}{2}k_2)h,$$
$$\text{where } k_1 = f(t_i, y_i), \text{ and } k_2 = f(t_i + h, y_i + k_1 h)$$

[Midpoint Method: $a_2 = 1 \rightarrow a_1 = 0$, $p_1 = q_{11} = \frac{1}{2}$]

$$y_{i+1} = y_i + k_2 h,$$
$$\text{where } k_1 = f(t_i, y_i), \text{ and } k_2 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_1 h)$$

[Ralston's Method: $a_2 = \frac{2}{3} \rightarrow a_1 = \frac{1}{3}$, $p_1 = q_{11} = \frac{3}{4}$]

$$y_{i+1} = y_i + (\tfrac{1}{3}k_1 + \tfrac{2}{3}k_2)h,$$
$$\text{where } k_1 = f(t_i, y_i), \text{ and } k_2 = f(t_i + \tfrac{3}{4}h, y_i + \tfrac{3}{4}k_1 h)$$

[Heun's Method: $a_2 = \frac{3}{4} \rightarrow a_1 = \frac{1}{4}$, $p_1 = q_{11} = \frac{2}{3}$]

$$y_{i+1} = y_i + (\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2)h,$$
$$\text{where } k_1 = f(t_i, y_i), \text{ and } k_2 = f(t_i + \tfrac{2}{3}h, y_i + \tfrac{2}{3}k_1 h)$$

Example: $L \cdot I'(t) + R \cdot I(t) = E,\ I(0) = 0$

$$\rightarrow I'(t) = f(t, I) = -\frac{R}{L} I(t) + \frac{E}{L},\ I(0) = 0$$

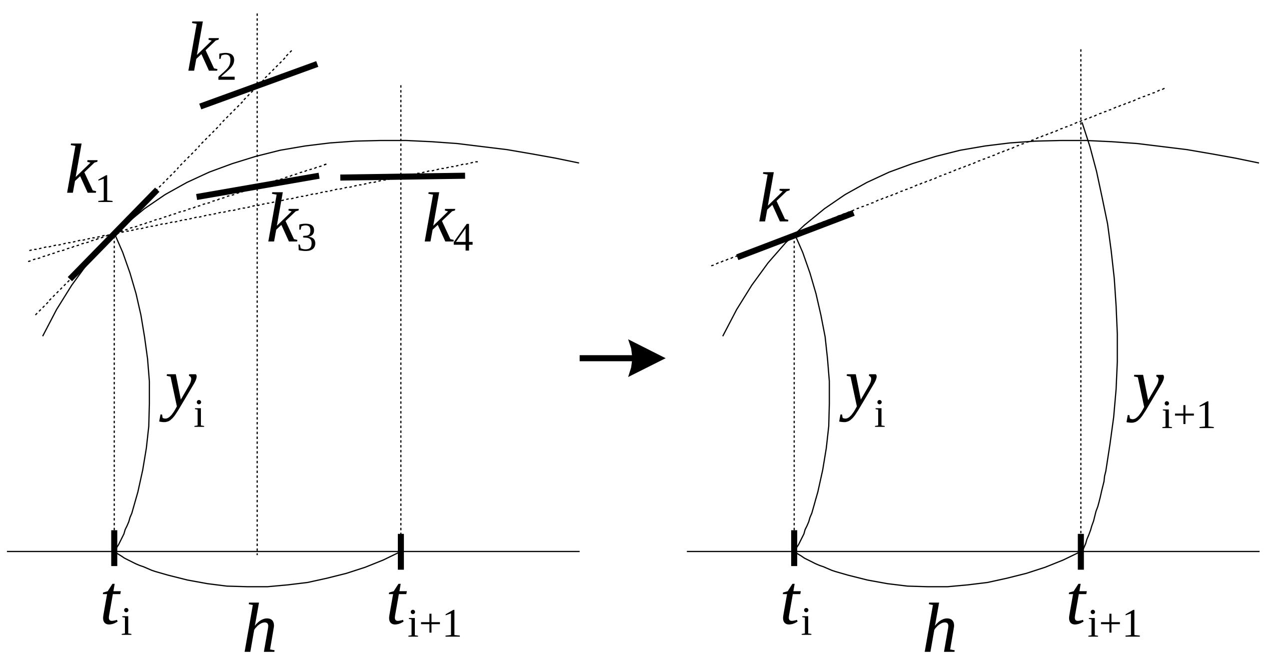
$$k_1 = f(t_i, I_i) = -\frac{R}{L} I_i + \frac{E}{L}$$

$$k_2 = f(t_i + h, I_i + k_1 \cdot h) = -\frac{R}{L}(I_i + k_1 \cdot h) + \frac{E}{L}$$

$$I_{i+1} = I_i + \frac{1}{2}(k_1 + k_2) h$$

"Classical" Fourth-Order Runge-Kutta Method

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h, \text{ where}$$
$$k_1 = f(t_i, y_i),$$
$$k_2 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_1 h),$$
$$k_3 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_2 h),$$
$$k_4 = f(t_i + h, y_i + k_3 h)$$



Example: $f(t,y) = 4e^{0.8t} - 0.5y,\ t_0 = 0, y_0 = 2\ (h = 0.5)$

$$
\begin{aligned}
k_1 &= f(0,2) = 4e^{0.8\cdot 0} - 0.5\cdot 2 = 3\\
k_2 &= f(0+\tfrac{1}{2}\cdot 0.5, 2+\tfrac{1}{2}\cdot 3\cdot 0.5) = f(0.25, 2.75) = 3.510611\\
k_3 &= f(0+\tfrac{1}{2}\cdot 0.5, 2+\tfrac{1}{2}\cdot 3.510611\cdot 0.5) = f(0.25, 2.877653) = 3.446785\\
k_4 &= f(0+0.5, 2+3.446785\cdot 0.5) = f(0.5, 3.723392) = 4.105603
\end{aligned}
$$

$$
y_1 = 2 + \tfrac{1}{6}(3 + 2\cdot 3.510611 + 2\cdot 3.446785 + 4.105603) = 3.503399
$$

각 방법의 비교

	Euler	Runge-Kutta		Talyor method of order n
		2nd order	4th order	
LTE	$O(h^2)$	$O(h^3)$	$O(h^5)$	$O(h^{n+1})$
GTE	$O(h)$	$O(h^2)$	$O(h^4)$	$O(h^n)$
Func. Eval.	f 1번	f 2번	f 4번	$f, f', f'', \ldots, f^{n-1}$ 각 1번

Example: $y' = f(t, y) = y - t^2 + 1,\ y(0) = 0.5\ (0 \le t \le 2)$

Euler

t_i	w_i	$y_i = y(t_i)$	$\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0.0000000
0.2	0.8000000	0.8292986	0.0292986
0.4	1.1520000	1.2140877	0.0620877
0.6	1.5504000	1.6489406	0.0985406
0.8	1.9884800	2.1272295	0.1387495
1.0	2.4581760	2.6408591	0.1826831
1.2	2.9498112	3.1799415	0.2301303
1.4	3.4517734	3.7324000	0.2806266
1.6	3.9501281	4.2834838	0.3333557
1.8	4.4281538	4.8151763	0.3870225
2.0	4.8657845	5.3054720	0.4396874

A

2nd–order RK

t_i	$y(t_i)$	Midpoint Method	Error B	Modified Euler Method	Error C	Heun's Method	Error D
0.0	0.5000000	0.5000000	0	0.5000000	0	0.5000000	0
0.2	0.8292986	0.8280000	0.0012986	0.8260000	0.0032986	0.8273333	0.0019653
0.4	1.2140877	1.2113600	0.0027277	1.2069200	0.0071677	1.2098800	0.0042077
0.6	1.6489406	1.6446592	0.0042814	1.6372424	0.0116982	1.6421869	0.0067537
0.8	2.1272295	2.1212842	0.0059453	2.1102357	0.0169938	2.1176014	0.0096281
1.0	2.6408591	2.6331668	0.0076923	2.6176876	0.0231715	2.6280070	0.0128521
1.2	3.1799415	3.1704634	0.0094781	3.1495789	0.0303627	3.1635019	0.0164396
1.4	3.7324000	3.7211654	0.0112346	3.6936862	0.0387138	3.7120057	0.0203944
1.6	4.2834838	4.2706218	0.0128620	4.2350972	0.0483866	4.2587802	0.0247035
1.8	4.8151763	4.8009586	0.0142177	4.7556185	0.0595577	4.7858452	0.0293310
2.0	5.3054720	5.2903695	0.0151025	5.2330546	0.0724173	5.2712645	0.0342074

t_i	Runge-Kutta Order Four w_i	Exact $y_i = y(t_i)$	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.5000000	0
0.2	0.8292933	0.8292986	0.0000053
0.4	1.2140762	1.2140877	0.0000114
0.6	1.6489220	1.6489406	0.0000186
0.8	2.1272027	2.1272295	0.0000269
1.0	2.6408227	2.6408591	0.0000364
1.2	3.1798942	3.1799415	0.0000474
1.4	3.7323401	3.7324000	0.0000599
1.6	4.2834095	4.2834838	0.0000743
1.8	4.8150857	4.8151763	0.0000906
2.0	5.3053630	5.3054720	0.0001089

E

4^{th}-order RK

High-order Taylor Method

t_i	Taylor Order 2 w_i	Error $\lvert y(t_i) - w_i \rvert$	Taylor Order 4 w_i	Error $\lvert y(t_i) - w_i \rvert$	Exact $y(t_i)$
0.0	0.5000000	0	0.5000000	0	0.5000000
0.2	0.8300000	0.0007014	0.8293000	0.0000014	0.8292986
0.4	1.2158000	0.0017123	1.2140910	0.0000034	1.2140877
0.6	1.6520760	0.0031354	1.6489468	0.0000062	1.6489406
0.8	2.1323327	0.0051032	2.1272396	0.0000101	2.1272295
1.0	2.6486459	0.0077868	2.6408744	0.0000153	2.6408591
1.2	3.1913480	0.0114065	3.1799640	0.0000225	3.1799415
1.4	3.7486446	0.0162446	3.7324321	0.0000321	3.7324000
1.6	4.3061464	0.0226626	4.2835285	0.0000447	4.2834838
1.8	4.8462986	0.0311223	4.8152377	0.0000615	4.8151763
2.0	5.3476843	0.0422123	5.3055554	0.0000834	5.3054720

F

G

- 문제: 구간 h의 크기를 줄여 Euler 방법을 사용하는 것과 2차 또는 4차 RK 방법을 사용하는 것 중 어느 것이 더 좋은 방법일까?

t_i	Exact	Euler $h = 0.025$	Midpoint $h = 0.05$	Runge-Kutta Order Four $h = 0.1$
0.0	0.5000000	0.5000000	0.5000000	0.5000000
0.1	0.6574145	0.6554982	0.6573726	0.6574144
0.2	0.8292986	0.8253385	0.8292127	0.8292983
0.3	1.0150706	1.0089334	1.0149386	1.0150701
0.4	1.2140877	1.2056345	1.2139076	1.2140869
0.5	1.4256394	1.4147264	1.4254094	1.4256384

Simple ODE Solvers - Derivation

- From *http://www.math.ubc.ca/~feldman/math/**odesolvers**.pdf*

Higher-Order ODE

- First-order ODE system 형태로 해결을 할 수 있음.
  - 예: Second-order ODE

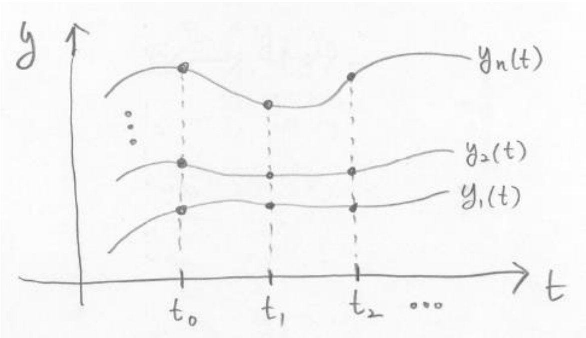
$$\boxed{y'' + 3y' + y = 0,\ y(0) = 1,\ y'(0) = 0}$$

$$y_1(t) \equiv y(t)$$

$$\boxed{\begin{aligned} y_1' &= y_2, & y_1(0) &= 1 \\ y_2' &= -y_1 - 3y_2, & y_2(0) &= 0 \end{aligned}}$$

- First-order ODE system

$$\begin{aligned} y_1'(t) &= f_1(t, y_1, y_2, \cdots, y_n), & y_1(t_0) &= y_{10} \\ y_2'(t) &= f_2(t, y_1, y_2, \cdots, y_n), & y_2(t_0) &= y_{20} \\ &\vdots \\ y_n'(t) &= f_n(t, y_1, y_2, \cdots, y_n), & y_n(t_0) &= y_{n0} \end{aligned}$$



First-Order Euler Method for First-Order ODE System

$n=1$ $\quad y'(t) = f(t, y), \; y(t_0) = y_0$

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

$n>1$

$$\boxed{\begin{aligned} y_{1,i+1} &= y_{1i} + h \cdot f_1(t_i, y_{1i}, y_{2i}, \cdots, y_{ni}) \\ y_{2,i+1} &= y_{2i} + h \cdot f_2(t_i, y_{1i}, y_{2i}, \cdots, y_{ni}) \\ &\vdots \\ y_{n,i+1} &= y_{ni} + h \cdot f_n(t_i, y_{1i}, y_{2i}, \cdots, y_{ni}) \end{aligned}}$$

Second-Order RK Method for First-Order ODE System

$n=1$

$$\left\{\begin{array}{l} k_1 = f(t_i, y_i) \\ k_2 = f(t_i + h,\ y_i + k_1 \cdot h) \end{array}\right\} \Rightarrow y_{i+1} = y_i + \frac{k_1 + k_2}{2} \cdot h$$

$n=2$

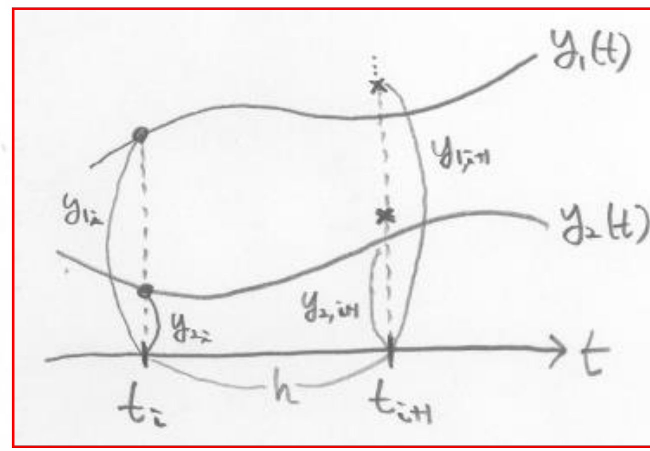
$$\left\{\begin{array}{ll} y_1' = f_1(t, y_1, y_2), & y_1(t_0) = y_1' \\ y_2' = f_2(t, y_1, y_2), & y_2(t_0) = y_2' \end{array}\right.$$

$$\left\{\begin{array}{l} (k_1)_1 = f_1(t_i, y_{1i}, y_{2i}) \\ (k_1)_2 = f_2(t_i, y_{1i}, y_{2i}) \\ (k_2)_1 = f_1(t_i + h,\ y_{1i} + (k_1)_1 \cdot h,\ y_{2i} + (k_1)_2 \cdot h) \\ (k_2)_2 = f_2(t_i + h,\ y_{1i} + (k_1)_1 \cdot h,\ y_{2i} + (k_1)_2 \cdot h) \end{array}\right.$$

$$\Rightarrow y_{1,i+1} = y_{1i} + \frac{(k_1)_1 + (k_2)_1}{2} \cdot h$$

$$y_{2,i+1} = y_{2i} + \frac{(k_1)_2 + (k_2)_2}{2} \cdot h$$

$(k_i)_j$: j 번째 함수의 k_i

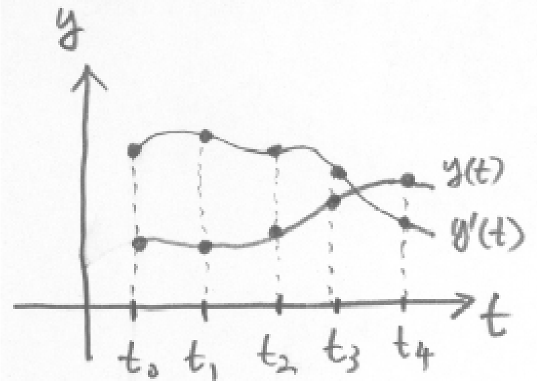


Second-Order RK Method for Second-Order ODE

- 문제

$$y''(t) + a\,y'(t) + b\,y(t) = g(t)$$

$$\left.\begin{array}{l} y(t_0) = y_0 \\ y'(t_0) = y_0' \end{array}\right\} \text{initial values}$$



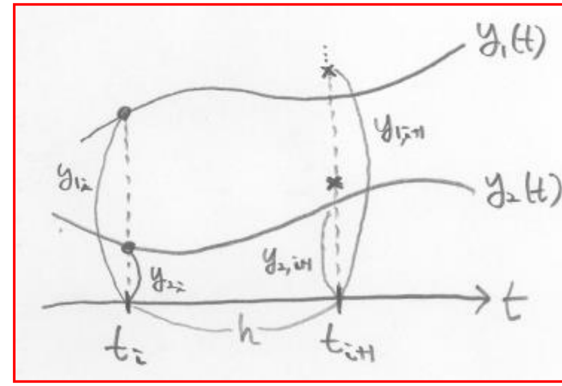
- First-order ODE system으로의 변환

$y_1(t) \equiv y(t)$, $y_2(t) \equiv y'(t)$ 라 하면,

$$\begin{cases} y_1'(t) = f_1(t, y_1, y_2) = y_2(t), & y_1(t_0) = y_0 \\ y_2'(t) = f_2(t, y_1, y_2) = -a y_2(t) - b y_1(t) + g(t), & y_2(t_0) = y_0' \end{cases}$$

$$y_{1,i+1} = y_{1i} + \frac{(k_1)_1 + (k_2)_1}{2} h$$

$$y_{2,i+1} = y_{2i} + \frac{(k_1)_2 + (k_2)_2}{2} h$$

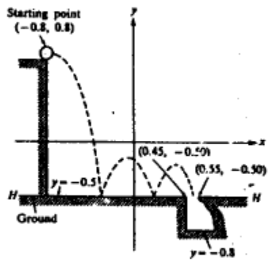


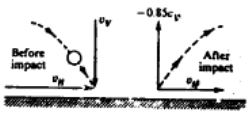
$$\left\{\begin{aligned} (k_1)_1 &= f_1(t_i, y_{1i}, y_{2i}) = y_{2i} \\ (k_1)_2 &= f_2(t_i, y_{1i}, y_{2i}) = -a y_{2i} - b y_{1i} + g(t_i) \end{aligned}\right.$$

$$\left\{\begin{aligned} (k_2)_1 &= f_1(t_i + h,\ y_{1i} + (k_1)_1 h,\ y_{2i} + (k_1)_2 h) \\ &= y_{2i} + (k_1)_2 h \\ (k_2)_2 &= f_2(t_i + h,\ y_{1i} + (k_1)_1 h,\ y_{2i} + (k_1)_2 h) \\ &= -a(y_{2i} + (k_1)_2 h) - b(y_{1i} + (k_1)_1 h) + g(t_i + h) \end{aligned}\right.$$

Example: A Bouncing Ball

Bouncing Ball Problem





문제 공이 어떠한 초기속도로 움직일때 구멍에 빠지게 되는지를 계산하라.

1. 조건

① $\begin{cases} v_H(t): \text{the horizontal velocity at time } t \\ v_V(t): \text{the vertical velocity at time } t \end{cases}$

$\begin{cases} x(t): \text{the } x \text{ coordinate of the ball at time } t \\ y(t): \text{the } y \text{ " " " " " "} \end{cases}$

$$\left.\begin{cases} x'(t) = \dfrac{dx(t)}{dt} = v_0 \\ y''(t) = \dfrac{d^2y(t)}{dt^2} = -g \end{cases}\right\} (*)$$

v_0 : an initial horizontal velocity

g : 32.2 ft/sec²

$v_V(t) = y'(t)$ 이므로 (*)이

$$\begin{cases} x'(t) = v_0, & x(0) = x_0 \\ v_V'(t) = -g, & v_V(0) = v_0 \\ y'(t) = v_V(t), & y(0) = y_0 \end{cases}$$

t	t_0	t_1	t_2	⋯	t_n
x(t)	x_0	x_1	x_2	⋯	x_n
y(t)	y_0	y_1	y_2	⋯	y_n

② It the ball bounces, the horizontal velocity remains constant as v_0, and the vertical velocity velocity v_V after one bounce is equal to the negative value of 85% of the vertical velocity before the bounce.

2. (the second-order Runge-Kutta method)

$$\begin{cases} x_{i+1} = x_i + h \cdot v_0 \\ v_{i+1} = v_i + h \cdot (-g) \\ y_{i+1} = y_i + \frac{1}{2} h \cdot (v_i + v_{i+1}) \end{cases}, \quad h = \Delta t$$

3. 방법

① 주어진 초기속도 v_0에 대해 $x(t) \geq 0.58$이 되는 시간까지 궤도를 계산

② 만약 공이 구멍에 빠지지 않으면 초기속도를 0.005 ft/sec 씩 감소시키며 ①의 계산 반복.

③ 공이 구멍에 빠진경우 $y = -0.8$ 이 될때까지 궤도 계산.

※ $\begin{cases} \Delta t = 0.005 \text{ sec로 할것} \\ v_0 = 0.78 \text{ ft/sec로 부터 시작할것.} \end{cases}$

$$\left\{\begin{array}{l} x'(t) = \dfrac{dx(t)}{dt} = v_0 \\ y''(t) = \dfrac{d^2y(t)}{dt^2} = -g \end{array}\right\} (*)$$

v_0 : an initial horizontal velocity

g : 32.2 ft/sec^2

2nd-order ODE system

$$\begin{aligned} y_1(t) &\equiv x(t) \\ y_2(t) &\equiv y(t) \\ y_3(t) &\equiv y_2'(t) = y'(t) \end{aligned}$$

$$\begin{aligned} y_1' &= f_1(t, y_1, y_2, y_3) = v_0, \qquad y_1(0) = x_0 \\ y_2' &= f_2(t, y_1, y_2, y_3) = y_3, \qquad y_2(0) = y_0 \\ y_3' &= f_3(t, y_1, y_2, y_3) = -g, \qquad y_3(0) = v_0 \end{aligned}$$

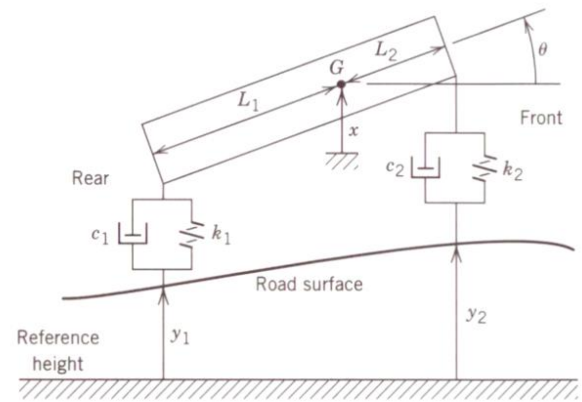
1st-order ODE system

2nd-order RK

$$\begin{aligned} y_{1,i+1}(t) &= y_{1i} + v_0 h \\ y_{3,i+1}(t) &= y_{3i} + (-g)h \\ y_{2,i+1}(t) &= y_{2i} + \frac{y_{3i} + y_{3,i+1}}{2}h \end{aligned}$$

Example: Vehicle Suspension Design

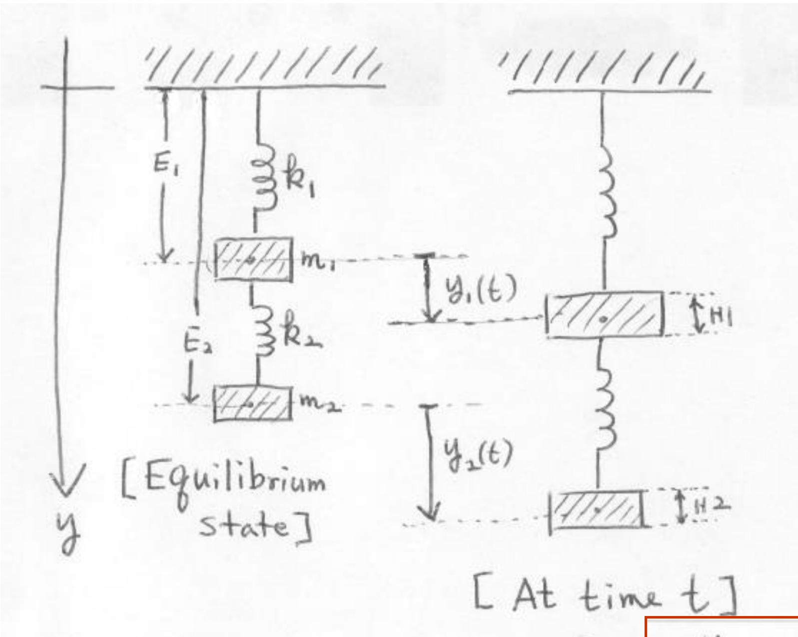
$$
\begin{aligned}
m \cdot y_5'' &= -(k_1 + k_2)y_5 - (c_1 + c_2)y_5' + (k_1 L_1 - k_2(L - L_1))\theta \\
&\quad + (c_1 L_1 - c_2(L - L1))\theta' + (k_1 y_1 + k_2 y_2 + c_1 y_1' + c_2 y_2') \\
I \cdot \theta'' &= (L_1 k_1 - (L - L_1)k_2)y_5 + (L_1 c_1 - (L - L_1)c_2)y_5' + \\
&\quad - (L_1^2 k_1 + (L - L_1)^2 k_2)\theta - (L_1^2 c_1 + (L - L_1)^2 c_2)\theta' \\
&\quad + (-L_1 k_1 y_1 + (L - L_1)k_2 y_2 - L_1 c_1 y_1' + (L - L_1)c_2 y_2')
\end{aligned}
$$



만약 2nd-order RK 방법을 사용하여 3D 게임 엔진을 구현하려면 …

???

Example: A Spring System



$$\begin{cases} m_1 \cdot y_1'' = -k_1 \cdot y_1 - k_2 (y_1 - y_2) \\ m_2 \cdot y_2'' = k_2 (y_1 - y_2) \end{cases}$$

$$y_1(0) = A,\quad y_1'(0) = B,\quad y_2(0) = C,\quad y_2'(0) = D$$

$$y_{10} \equiv y_1,\quad y_{11} \equiv y_1',\quad y_{20} \equiv y_2,\quad y_{21} \equiv y_2'$$

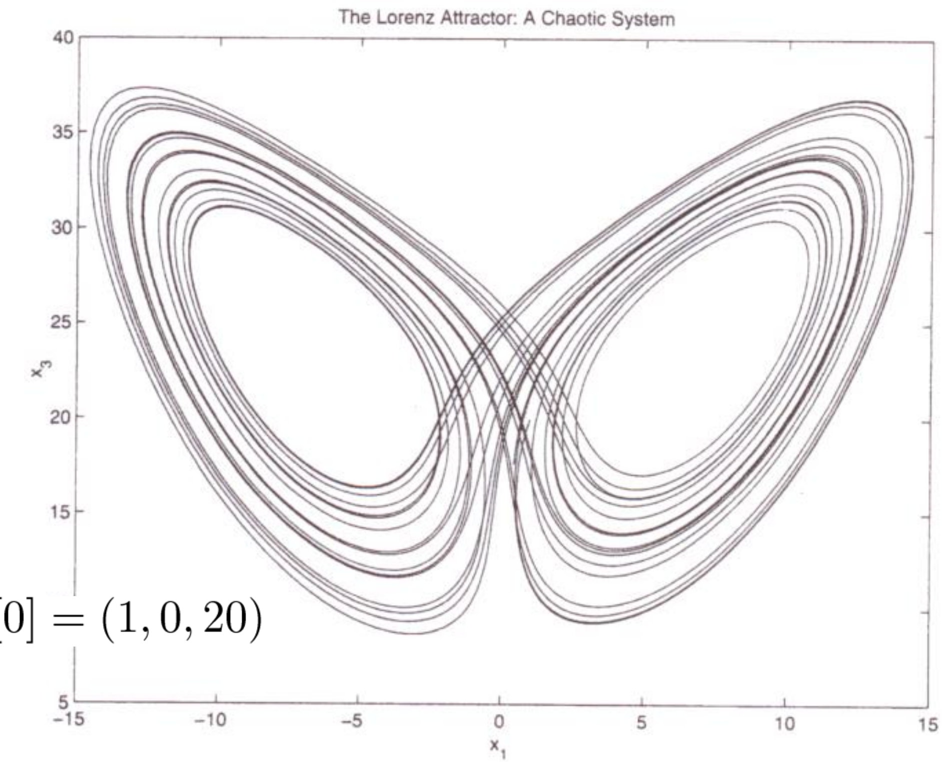
$$\begin{cases} y_{10}' = y_{11} = f_{10}(t, y_{10}, y_{11}, y_{20}, y_{21}), & y_{10}(0) = A \\ y_{11}' = \frac{1}{m_1}\{-k_1 \cdot y_{10} - k_2 \cdot (y_{10} - y_{20})\}, & y_{11}(0) = B \\ \quad = f_{11}(t, y_{10}, y_{11}, y_{20}, y_{21}) & \\ y_{20}' = y_{21} = f_{20}(t, y_{10}, y_{11}, y_{20}, y_{21}), & y_{20}(0) = C \\ y_{21}' = \frac{k_2}{m_2}\{y_{10} - y_{20}\} = f_{21}(t, y_{10}, y_{11}, y_{20}, y_{21}), & y_{21}(0) = D \end{cases}$$

Example: Chaos System (Lorenz Attractor System)

$$
\begin{aligned}
\frac{dx_1}{dt} &= \sigma(x_2 - x_1) \\
\frac{dx_2}{dt} &= (1 + \lambda - x_3)x_1 - x_2 \\
\frac{dx_3}{dt} &= x_1 x_2 - \gamma x_3 \\
& (\sigma, \lambda, \gamma > 0)
\end{aligned}
$$

Chaotic behavior is observed when $\sigma > \gamma + 1$ and $\lambda > \frac{(\sigma+1)(\sigma+\gamma+1)}{\sigma-\gamma-1}$.

When $\sigma = 10$, $\lambda = 24$, and $\gamma = 2$ with $x[0] = (1, 0, 20)$



Courtesy of Schilling and Harris

Fehlberg Fourth-Fifth-Order Runge-Kutta Method

`subroutine rkf45(f,neqn,y,t,tout,relerr,abserr,iflag,work,iwork)`

- Subroutine **`rkf45`** integrates a system of **`neqn`** first order ordinary differential equations of the form

  `dy(i)/dt = f(t,y(1),y(2),...,y(neqn))` where the y(i) are given at t .
- Typically the subroutine is used to integrate from **`t`** to **`tout`** but it can be used as a one-step integrator to advance the solution a single step in the direction of **`tout`**.
- On return the parameters in the call list are set for continuing the integration.
- The user has only to call **`rkf45`** again (and perhaps define a new value for **`tout`**).
- Actually, **`rkf45`** is an interfacing routine which calls subroutine **`rkfs`** for the solution.
- **`rkfs`** in turn calls subroutine fehl which computes an approximate solution over one step.

Function Parameters

- **f** : subroutine f(t,y,yp) to evaluate derivatives
  yp(i)=dy(i)/dt
  (cf. void eval_f(double *t, double *y, double *yp);
- **neqn** : number of equations to be integrated
- **y(*)** : solution vector at t
- **t** : independent variable
- **tout** : output point at which solution is desired
- **relerr, abserr** : relative and absolute error tolerances for local error test. At each step the code requires that abs(local error) .le. relerr*abs(y) + abserr
  for each component of the local error and solution vectors
- **iflag** : indicator for status of integration
- **work(*)** : array to hold information internal to rkf45 which is necessary for subsequent calls. Must be dimensioned at least 3+6*neqnc.
- **iwork(*)** : integer array used to hold information internal to rkf45 which is necessary for subsequent calls. Must be dimensioned at least 5.

A Sample Usage (From C to Fortran)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define NEQN 2

double work[3+6*NEQN+10];
int iwork[10], neqn = NEQN;

void ODE_I(double *t, double *y, double
   *yp) {
  yp[0] = -4.0*y[0] + 3.0*y[1] + 6.0;
  yp[1] = -2.4*y[0] + 1.6*y[1] + 3.6;
}

double ExactI1(double t) {
  return -3.375*exp(-2*t) + 1.875*exp(-
   0.4*t) + 1.5;
}

double ExactI2(double t) {
  return -2.25*exp(-2.0*t) + 2.25*exp(-
   0.4*t);
}

double abserr(double src, double dest) {
  return (src > dest) ? src-dest :
                              dest-src;
}
```

텍스트

```c
int main(void){
  double y[2] = { 0.0, 0.0 };
  double err = 0.00000000001;
  double t = 0.0, tinit = 0.0;
  int iflag = +1;

  printf("%3s  %15s
   %15s\t\t%15s\t%15s\n", "t",
   "w1","w2","|I1(t)-w1|","|I2(t)-w2|");
  printf("----------------------------------
   ----------------------------------------
   ------------\n");
  printf("%3f   %15.10f
   %15.10f\t%15.10f\t%15.10f\n",t,0.0,0.0,0
   .0,0.0);

  for( t = 0.1; t < 0.6 ; t += 0.1 ) {
    rkf45_(ODE_I, &neqn, y, &tinit, &t,
         &err, &err, &iflag, work, iwork);

    printf("%3f   %15.10f
        %15.10f\t%10.10E   %10.10E\n",t,
        y[0],y[1],abserr(ExactI1(t),y[0]),
        abserr(ExactI2(t),y[1]));
  }
  return 0;
}
```