[수치해석] Differential Equation

F = ma와 같은 물리 법칙을 **Ordinary Differential Equation (ODE)**로 표현할 수 있다.

위 ODE를 Analytic하게 풀 수 도 있고 Numerical하게 풀 수 있다.

- Analytic: 식을 정확히 풀어 **공식**을 통한 정확한 해를 구함
- Numerical: 컴퓨터로 **근사 계산**

하지만, 대부분의 물리 법칙은 Analytic하게 풀 수 없다. **거의 모든 계산은 Numerical하게 진행**된다.

Differential equation

> **미지 함수와 그 도함수 간의 관계를 표현한 방정식**

Differential equation을 푼다는 것은 도함수로부터 미지 함수를 구하는 것이다.

Defferential equation의 종류

1. **Ordinary Differential Equation**: 편미분을 포함하지 않는 미분방정식
2. **Partial Differential Equation**: 편미분을 포함하는 미분방정식

Linear ODE: y, y′, y′′끼리 서로 곱해져 있지 않은 형태

Nonlinear ODE: y, y′, y′′끼리 서로 곱해져 있는 형태

- 일반적으로 **Analytic solution을 찾기 힘들다.**
- **Numerical** 방법을 이용하거나 **Linearization**을 이용한다.

Problem의 형태

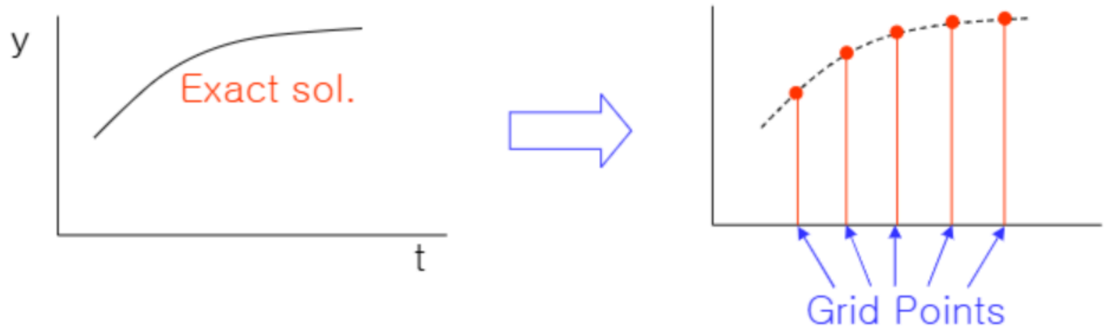
1. **Initial Value Problem (IVP): 초깃값을 주고, 특정 시간이 지났을 때의 값이 어떻게 되는지 찾는 문제**

2. **Boundary Value Problem (BVP): 두 구간의 끝 점의 정보가 주어지고, 두 구간 사이의 함수를 찾는 문제**
    - 일반적으로 더 어렵다.

Error in solving differential equation

Defferential equation을 풀 때, Analytic하게 해결되지 않는 경우에는 Numerical하게 풀어야 한다.

이 경우, 각 구간을 **Discretization** 해야 한다.



미분을 표현할 때 Δt를 사용하는데, 컴퓨터는 $\Delta t \rightarrow 0$를 정확히 표현하지 못하여 **Discretization error**가 생긴다.

- Analytic하게 구할 때 미분의 정의를 사용하는 것을 생각해보면 된다.

Discretization을 이용하여 컴퓨터로 Numerical 계산을 수행할 때, Round-off error 등이 누적된다. 이를 **Stability error**라고 한다.

정확한 Solution: y_e, Discretization만 했을 때의 Solution: y_d, Numerical method로 구한 Solution: y_n 이라고 하면 error를 다음과 같이 작성할 수 있다.

Discretization error: $y_e - y_d$

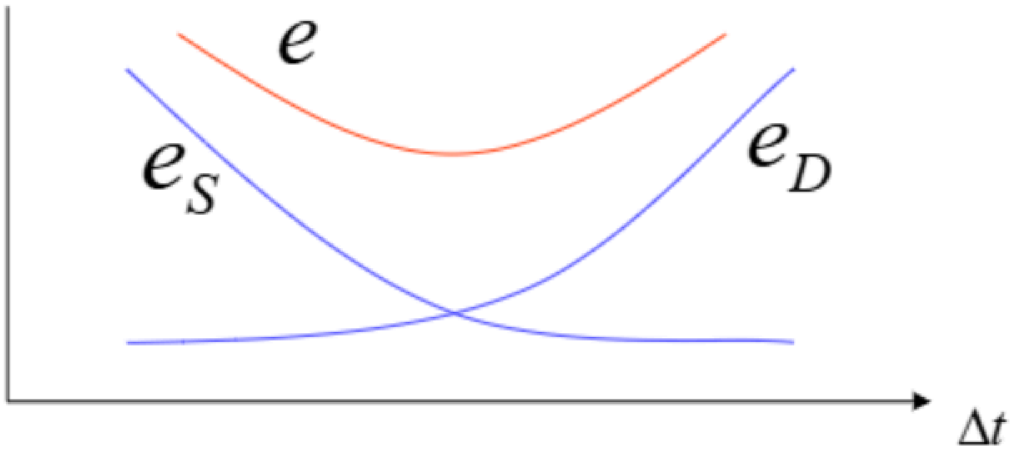
- Δt가 0에 가까울수록 작다.
- Trucation error이다.

Stability error: $y_d - y_n$

- Δt가 0에 가까울수록 계산 값이 작아져 커진다.
- Round-off error이다.

Total error: Discretization error + Stability error

- 두 Error는 **Trade-off**이다.



Local error: 이전 Step인 모두 올바르다고 가정했을 때, 한 step에서만 발생한 오차

Global error: 전체 step의 누적 오차

Useful concepts

1. Consistency

스텝 크기 h → 0일 때, 변형된 미분 방정식이 원래 미분방정식을 만족하는가?

2. Order

Discretizaion error e_D가 Δt^p에 비례한다.

- p: 차수

Step size를 줄일수록 오차가 빠르게 감소함을 의미한다.

- **error** = $O(h^p)$

3. Convergence

Step 크기 Δt → 0일수록 Exact solution에 가까워진다.

4. Stable

Numerical method를 반복할 때, Round-off error가 누적되지 않는 것

- e_s가 커지지 않는 것이다.

결론

Convergence한 것은 반드시 Consistency하면서 Stable하다.

Consistency하면서 Stable한 것은 반드시 Convergence하다.

Euler Method 기본

> **미분 방정식이 주어졌을 때, 해당 미분 방정식을 이용하여 여러 점을 얻고 해당 점을 이은 선을 미지 함수로 선택하는 방법**

위 공식에서 Error $O(h)$를 무시하고 정리하면 다음과 같다.

$$y(f+h) = y(f) + hy'$$

즉, 초기값부터 미분 방정식에 대입해서 구한 $y(f)$와 y'을 이용하여 다음 Step의 함숫값을 얻는 작업을 계속 반복한다.

얻은 함수값들을 하나의 직선으로 잇는다면 Numerical하게 계산한 함수를 얻을 수 있다.

Explicit Euler method

> **현재 값으로 기울기를 바로 계산하는 방법**

$$y_{n+1} = y_n + hy'(t_n, y_n)$$

Implicit Euler method

> **다음 예측 값으로 기울기를 계산하도록 하는 방법**

$$y_{n+1} = y_n + hy'(t_{n+1}, y_{n+1})$$

- 양변에 미지수 y_{n+1}이 나와 방정식을 풀어야 한다.

예시를 살펴보자. 주어진 함수는 다음과 같다.

$$y' + y = 1.2, \quad y(0) = 0.2 \Rightarrow \quad y' = \underbrace{1.2 - y}_{= f}$$

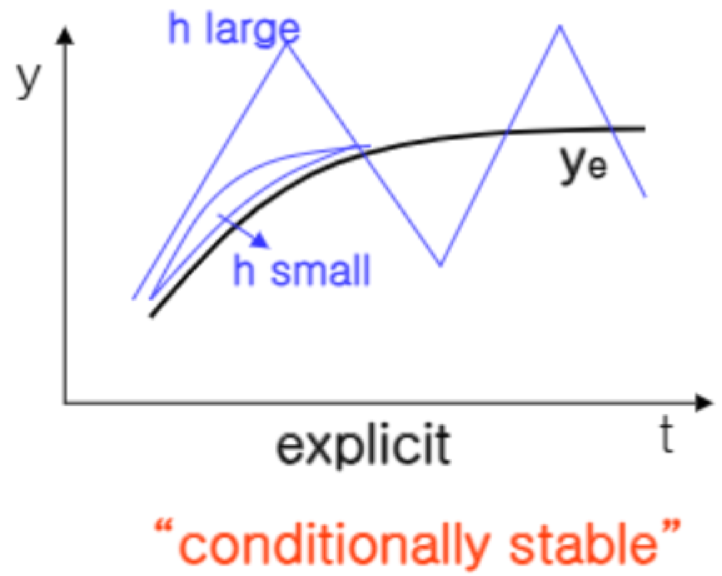
Explicit 방법으로 구하면 아래와 같다.

$$y_{n+1} = y_n + hf_n$$
$$y_{n+1} = y_n + h(1.2 - y_n) = 1.2h - (1 - h)y_n$$

Implicit 방법으로 구하면 아래와 같다.

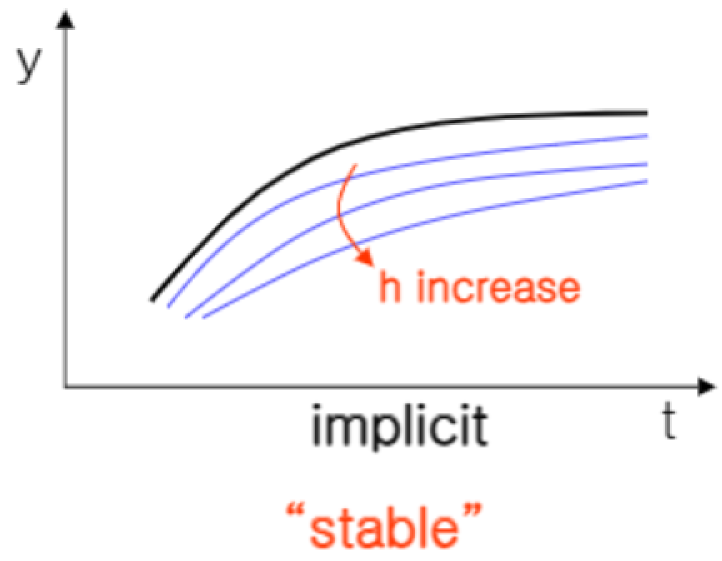
$$y_{n+1} = y_n + hf_{n+1}$$
$$y_{n+1} = y_n + h(1.2 - y_{n+1})$$
$$= \frac{y_n + 1.2h}{1 + h}$$

Explicit 방법으로 구한 것은 **Conditionally stable**한 것을 확인할 수 있다.



- y_n의 계수가 $(1-h)$이다. 이 경우, $||1-h|| \leq 1$ **인 경우에만 Stable**하다.
- 작은 Step size가 안정적이다.

Implicit 방법으로 구한 것은 **항상 Stable**하다.



- y_n의 계수가 $\frac{1}{1+h}$이다. 따라서 **조건과 상관없이 항상 Stable**하다.

Stability

eg. $y' = -Ay, \quad y(0) = y_0$

Exact sol. $y = y_0 e^{-At}$

Euler method $y_{n=1} = y_n - hAy_n$

$$= (1 - hA) y_n$$

$$= (\underline{1 - hA})^{n+1} y_0$$

Amplification factor

For stability

$$|1 - hA| \le 1 \quad \rightarrow \quad 0 < h \le \frac{2}{A}$$

위 상황에서 **Amplification factor**에 의해 오차가 존재한다면, **오차도 계속 같은 비율로 증폭된다.**

따라서 **| Amplification factor | ≤ 1**이어야 **Stability**하다.

- Euler method에서 y_n의 계수가 중요한 것이다.
- **| Amplification factor | ≤ 1이면 되고, | Amplification factor |가 작을수록 좋거나 한 것은 아니다.**

Modified Differential Equation

> **실제 미분 방정식을 Euler method 등을 이용하여 풀 때, 근사값을 계산하다 보니 생기는 변형되는 식**

$y' + Ay = g(t)$의 미분 방정식을 풀 때, Euler method에 의해 $y_{n+1} = y_n + h(g(t) - Ay)$ 로 변형된 것도 **Modified Differential equation**이다.

위 Modified Differential equation가 **Consistency**를 만족하는지 확인해보자.

$$y_{n+1} = y_n + hy'|_n + \frac{1}{2!}h^2 y''|_n + \cdots$$

$$\cancel{y_n} + hy'|_n + \frac{1}{2!}h^2 y''|_n + \cdots = \cancel{y_n} + h(g_n - Ay_n)$$

$$y'|_n + Ay_n = g_n - \frac{1}{2!}\underline{hy''} - \cdots$$

; consistent

$$\text{Let } h \rightarrow 0 \quad y'|_n + Ay_n = g_n$$

1. 먼저 Talyor 2차 전개를 한다.
2. 2차 전개를 한 식과 $y_n + h(g(t) - Ay)$가 같기 때문에 겹치는 항을 소거한다.
3. (2)의 식에서 h → 0이라면 원래의 미분 방정식과 동일하게 나와 Consistency를 만족하는 것을 확인할 수 있다.

Order를 확인해보면 위에 (2)번 식에서 h → 0으로 보내기 전에 O(h)임을 확인할 수 있다.

<Order>

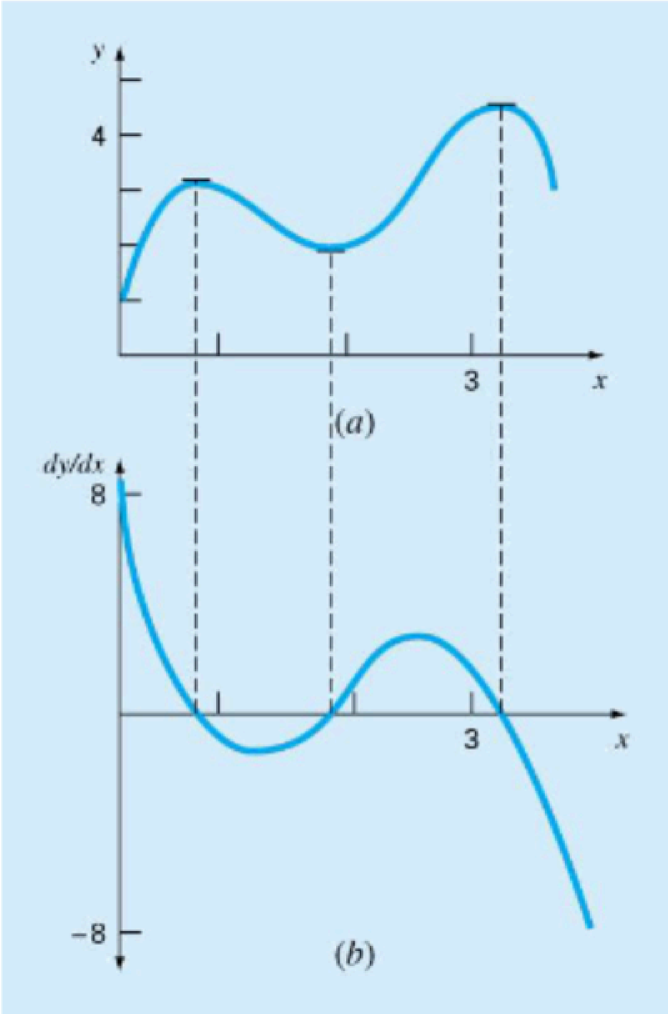
$$y'|_n + Ay_n = g_n + O(h)$$

Initial value problem

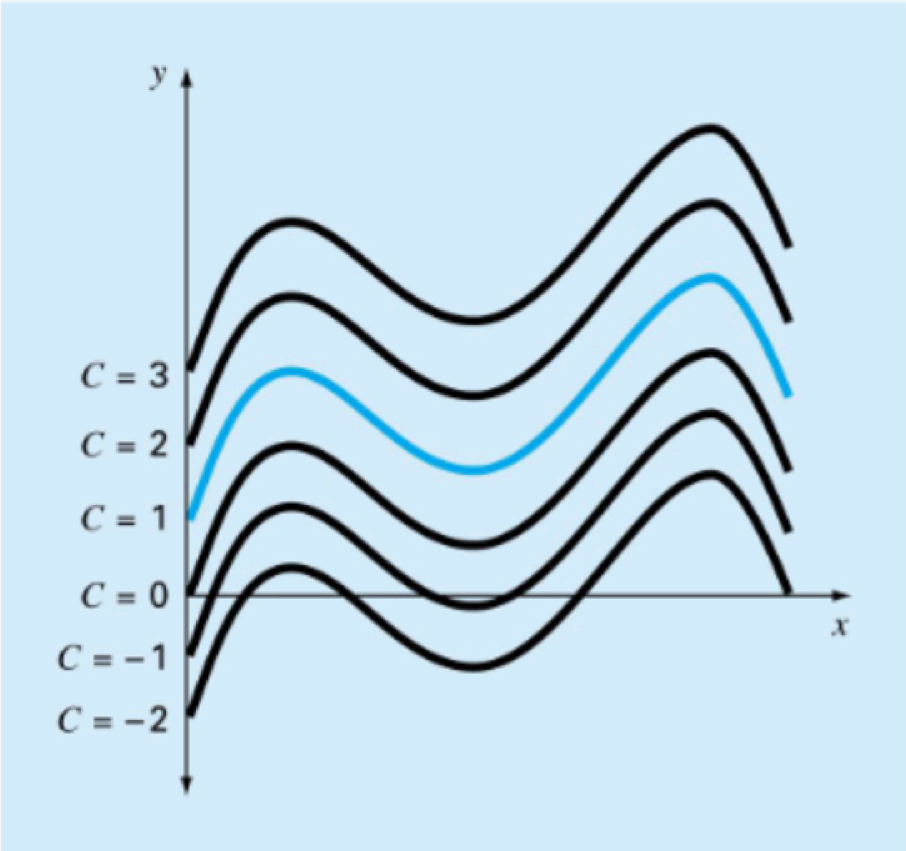
> **미분 방정식과 초기값이 주어지면 Solution이 결정되는 문제**

- Euler method는 Initial value problem을 푸는 방법이다.

초기값에 따라 기울기가 정해져 하나의 곡선이 정해진다.



초기값에 따라 여러 가능한 곡선 (Solution)이 존재할 수 있다.



두 가지 상황으로 나눌 수 있다.

Simultaneous D.E.: 여러 변수가 존재하고 각 변수끼리 서로 영향을 미치는 상황

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2, \cdots, y_n), \quad y_1(t_0) = y_{10}$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$\frac{dy_n}{dt} = f_n(t, y_1, y_2, \cdots, y_n), \quad y_n(t_0) = y_{n0}$$

High order D.E: 1차 도함수 이상이 존재하는 상황

$$y^{(n)} = f(t, y, y', y'', \cdots, y^{(n-1)})$$

$$y^{(t_0)} = y_0,\ y'(t_0) = y_0', \cdots, y^{(n-1)}(t_0) = y_0^{(n-1)}$$

High order D.E는 가상의 변수를 도입해 **Simulaneous D.E로 변형하여 풀 수 있다.**

- 단 Initial valur를 N개 제공해야 한다.
- N차라면 N개

예시: $y'' = -y$ **(이계 미분 방정식) 풀기**

1. 새로운 변수 y_1, y_2를 도입합니다.
2. $y_1 = y$ (원래 위치)
3. $y_2 = y'$ (속도)라고 정의합니다.
4. 그러면 식을 두 개의 1차 식으로 쪼갤 수 있습니다.
    - $y_1' = y_2$ (위치를 미분하면 속도)
    - $y_2' = -y_1$ (속도를 미분하면 가속도인데, 원래 식에 의해 $-y$니까)

Numerical method로 미분 방정식을 푸는 방법에 앞서 **Well-posed problem에 대해 정의해야 한다.**

> **함수 f와 그것을 y로 미분한 f_y가 끊어지지 않고 부드럽게 이어져 있다면(Continuous), 해당 Initial value problem은 Unique한 Solution을 갖고 Well-posed problem이다.**

Well-posed problem임이 증명된다면, Numerical method를 이용하여 미분 방정식을 해결할 수 있다.

Taylor series

Taylor 전개를 하면 다음과 같다.

$$y(t_0+h) = y_0 + y_0'\,h + \frac{1}{2!}y_0''\,h^2 + \cdots + \frac{1}{n!}y_0^{(n)}\,h^n + R_n$$

- 고차 미분이 있지만, **컴퓨터로 구현하기는 매우 복잡하다.**

R_n은 **Truncated error**로 다음과 같다.

$$R_n = \frac{h^{\,n+1}}{(n+1)!}\,y^{(n+1)}(\xi), \qquad t_0 \le \xi \le t_0 + h$$

이때, Talyor 전개에서 **1차항 까지만 사용**한다면 **Euler method와 동일**해진다.

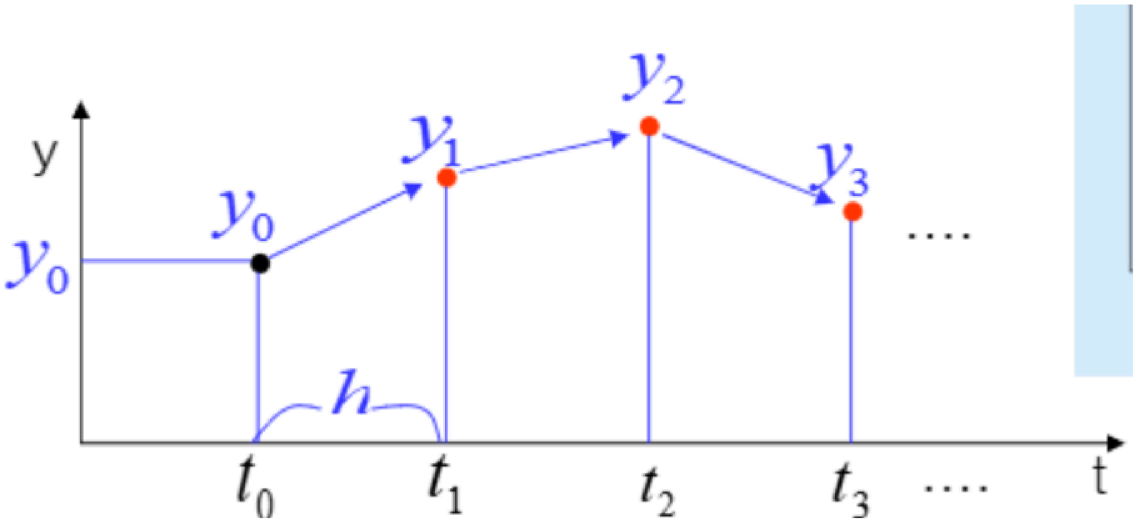
$$y(t_0+h) = y_0 + y_0'\,h + \frac{h^{\,2}}{(2)!}\,y^{(2)}(\xi)$$

위 식에서 오차를 무시하면 **Euler method의 공식과 동일**하다.

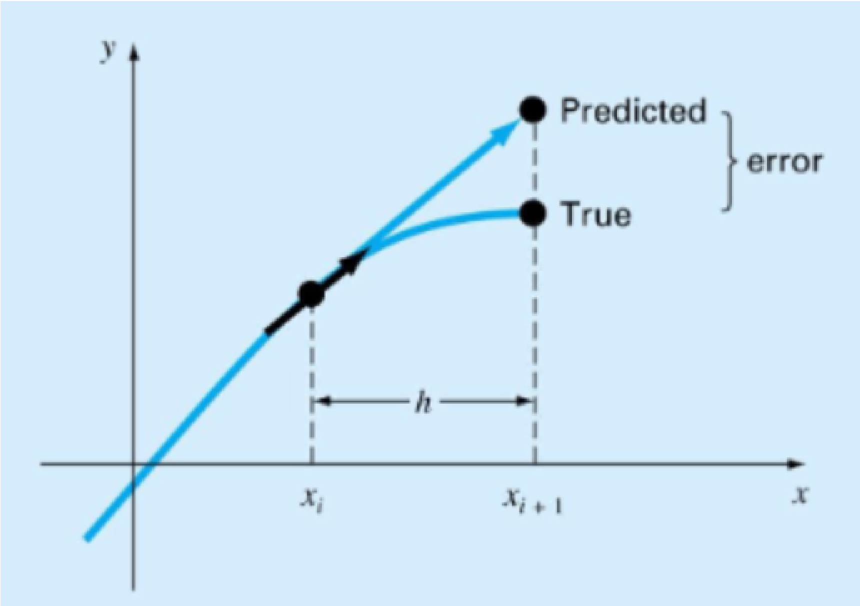
$$y(t_0+h) = y_0 + y_0'\,h$$

Euler Method 심화

Euler method는 **한 점에서의 기울기 방향으로 직선으로 이동**한다.



직선 이동하기 때문에 곡선 함수와의 **Error**가 생길 수 밖에 없다.



Taylor 전개에서 유도한 Euler method의 공식을 보자.

$$y(t_0+h) = y_0 + y_0'\,h + O(h^2)$$

- 여기서 $O(h^2)$은 Truncated error로, 한 Step에서의 **Local error**이다.
- $y_0 + y_0$'h가 Euler method의 **Approximation**이다.

Accumalated truncated error를 구해보자.

$$e_t = \sum_{i=0}^{n-1} \frac{h^2}{2} y^{''}(\xi_i)$$

이때 전체 $t_n - t_0$의 구간을 Step h로 이동한 횟수는 $\frac{t_n - t_0}{h}$이다.

또한, 위 식에서 $y^{''}$ 부분은 다음과 같이 바꾸어 쓸 수 있다.

$$\bar{y}''(\xi) = \frac{1}{n}\sum_{i=0}^{n-1} y''(\xi_i)$$

$\bar{y}''(\xi)$을 이용하여 e_t를 표현하면 다음과 같다.

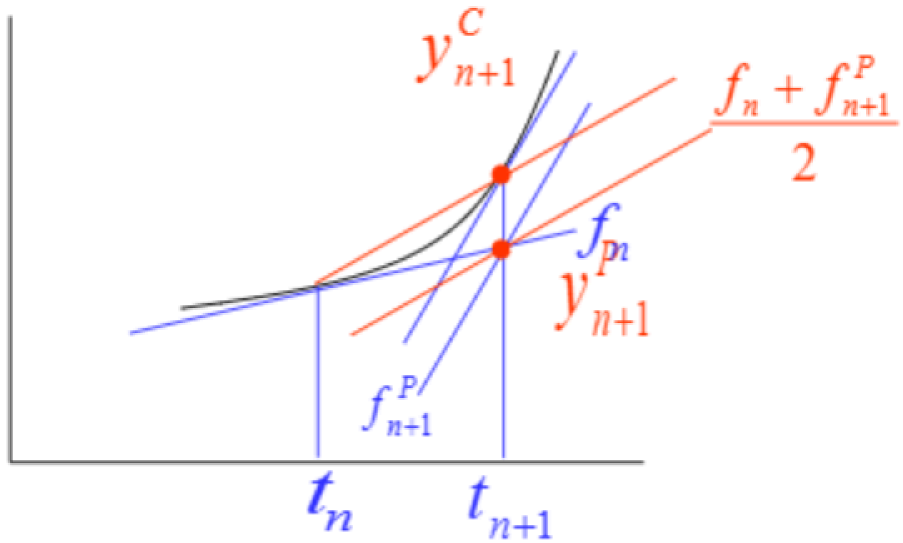
$$e_t = \sum_{i=0}^{n-1} \frac{h^2}{2} y^{''}(\xi_i) = n\bar{y}''(\xi)\frac{h^2}{2} = \frac{t_n - t_0}{h}\bar{y}''(\xi)\frac{h^2}{2} = \frac{1}{2}(t_n - t_0)\bar{y}''(\xi)h = O(h)$$

따라서 **Total error**는 $O(h)$이다.

- 누적 Error는 **선형적으로 증가**한다.

Modified Euler method

> **Euler method가 선형적으로만 이동할 수 있어 Error가 커지는 문제를 해결하기 위해 평균 기울기를 사용하는 방법**



1. 우선 Euler method의 공식대로 y^p_{n+1}를 얻는다.
   - $y^p_{n+1} = y_n + hy'_n$
2. y_n에서와 y^p_{n+1}에서의 기울기의 평균을 구한다.
   - $\bar{y} = \frac{f(y_n)+f(y^p_{n+1})}{2}$
3. (2)에서 구한 $\bar{y}$를 이용하여 Euler method를 하여 y_{n+1}을 얻는다.
   - $y_{n+1} = y_n + h\bar{y} = y_n + \frac{f(t_n,y_n)+f(t_{n+1},y_{n+1})}{2}h$

Heun's method with iteration

> **Modified Euler method 중 하나**

$$y_{i+1}^{j} \leftarrow\ y_i^m +\ \frac{f(t_i, y_i^m) + f(t_{i+1}, y_{i+1}^{j-1})}{2}\, h$$

과정

1. 먼저 Euler method를 통해 초기값을 얻는다.
    - $y_{i+1}^{(0)} = y_i + hf(t_i, y_i)$
2. 아래 Modified Euler method의 공식을 수렴할 때까지 반복한다.
    - $y_{i+1}^{(k+1)} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_{i+1}^{(k)})]$
3. 수렴한다면 $y_{i+1}^{(k+1)}$를 y_{i+1}로 사용한다.

Error

가장 먼저 **talyor 전개**를 생각해보자

- $y_{n+1} = y_n + hy'_n + \frac{1}{2}h^2y''_n + \frac{1}{3!}h^3y'''(\xi)$

Talyor 전개에 따르면 $y_n^{"}$은 다음과 같다.

- $y'_{n+1} = y'_n + h \cdot y''_n + \frac{1}{2}h^2 \cdot y'''_n + \cdots$
- = $\frac{y'_{n+1} - y'_n}{h} = y''_n + \frac{1}{2}h \cdot y'''_n + \cdots$
- $y''n = \frac{y'_{n+1} - y'_n}{h} - (\frac{1}{2}h \cdot y'''n + \cdots)$
- $y''n = \frac{y'n+1 - y'_n}{h} + O(h)$

따라서 맨 위의 Taylor 전개는 다음과 같이 적을 수 있다.

- $y_{n+1} = y_n + hy'_n + \frac{1}{2}h^2 \left\{ \frac{y'_{n+1} - y'_n}{h} + O(h) \right\} + \frac{1}{3!}h^3y'''(\xi)$

위 식을 전개하면 다음과 같다.

- $y_{n+1} = y_n + \frac{h}{2}[y'_n + y'_{n+1}] + O(h^3)$

따라서 Modified Euler method의 **Local truncated error**는 $O(h^3)$이다.

하지만 이 오차는 계산 동안 N번 누적되고, $N = \frac{t_n - t_0}{h}$이므로 **Total truncated error**는 $O(h^2)$이다.

- 따라서 **Huen's method**는 **2st order method**이다.

Euler method에 비해 훨씬 나은 성능 향상을 보인다.

Runge-kutta method

> **ODE의 Solution을 Euler method보다 정확하게 근사적으로 구하는 방법**

고차 미분 ($y^{''}, y^{'''}$) 등을 직접 구하지 않아 Source code로 구현하기 쉽다.

- 계산이 간단하다.

기본적인 공식은 다음과 같다.

$$y_{n+1} = y_n + h\phi(t_n, y_n, h)$$

$\phi(t_n, y_n, h)$는 다음과 같다.

$$\phi = a_1k_1 + a_2k_2 + ... + a_nk_n$$
$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \alpha_1, y_n + \beta_1)$$
$$k_n = f(t_n + \alpha_{n-1}, y_n + \beta_{n-1})$$

- f: 기울기 (한번 미분한 값)

즉, **미리 구해놓은 각 구간의 기울기의 가중 평균을 이용하여 업데이트**한다.

Second - order Runge-kutta method

> **현재 구간과 직전 구간의 평균 기울기를 이용하는 방법**

기본 수식은 아래와 같다.

$$y_{n+1} = y_n + h(a_1f(t_n, y_n) + a_2f(t_n + \alpha_1, y_n + \beta_1))$$

- $k_1 = f(t_n, y_n)$
- $k_2 = f(t_n + \alpha_1, y_n + \beta_1)$

위 공식을 유도하는 방법은 아래와 같다.

먼저 y_{n+1}을 **Taylor 전개**할 수 있다.

$$y_{n+1} = y_n + h(f)_n + \frac{h^2}{2}(f_t + f_y f)_n + O(h^3)$$

- **Second order이므로 기본 공식의 변형이 Taylor의 2차 전개와 동일**해야 한다.
- f_t: $\frac{\partial f}{\partial t}(t_n, y_n)$
- f_y : $\frac{\partial f}{\partial y}(t_n, y_n)$

$f(t_n + \alpha_1, y_n + \beta_1)$에 대한 **Taylor 1차 전개**를 확인해보자.

$$k_2 = f(t_n + \alpha_1, y_n + \beta_1) \approx f(t_n, y_n) + \alpha_1 f_t + \beta_1 f_y + R_n$$

기본 수식에 위에서 k_2를 근사한 식을 대입해보자.

$$y_{n+1} = y_n + h(a_1 f(t_n, y_n) + a_2(f(t_n, y_n) + \alpha_1 f_t + \beta_1 f_y + R_n))$$

전개하면 다음과 같다.

$$y_{n+1} = y_n + h(a_1 f(t_n, y_n) + a_2 f(t_n, y_n) + a_2\alpha_1 f_t + a_2\beta_1 f_y + a_2 R_n))$$

$$y_{n+1} = y_n + h(a_1 + a_2) f(t_n, y_n) + a_2 h(\alpha_1 f_t + \beta_1 f_y + R_n)$$

$a_2 h R_n$은 오차이므로 hR_n라고 쓸 수 있다.

$$y_{n+1} = y_n + h(a_1 + a_2)(f)_n + ha_2(\alpha_1 f_t + \beta_1 f_y)_n + hR_n$$

위 식을 아까 구한 Taylor 2차 전개와 같다고 하고 전개해보자

$$y_n + h(f)_n + \frac{h^2}{2}(f_t + f_y f)_n + O(h^3) = y_n + h(a_1 + a_2)(f)_n + ha_2(\alpha_1 f_t + \beta_1 f_y)_n + hR_n$$

두 식이 같아지기 위해선 아래 조건을 만족하면 된다.

1. $a_1 + a_2 = 1$
2. $a_2\alpha_1 = \frac{h}{2}$
3. $a_2\beta_1 = \frac{2}{h} f(t_n, y_n)$

위 조건을 만족하면 **2st order임을 증명**할 수 있다.

위 조건을 어떻게 설정하는지에 따라 다른 방식으로 동작한다.

$a_2 = \frac{1}{2}$인 경우에는 **Modified Euler method와 동일**하다.

- $a_1 = \frac{1}{2}, \alpha_1 = h, \beta_1 = hf(t_n, y_n)$

위 경우에 **2st order runge-kutta method**를 적용하면 다음과 같다.

- Modified Euler method와 동일하다.

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))]$$

Modified Euler method은 2st order runge-kutta method의 종류 중 하나이다.

다른 **2st order runge-kutta method도 있다.**

1. Midpoint method: 중간 기울기만 반영 ($a_1 = 0, a_2 = 1, \alpha_1 = \frac{h}{2}, \beta_1 = \frac{h}{2}k_1$)

$$y_{n+1} = y_n + h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

- 구간의 중간에서의 기울기를 이용하여 업데이트한다.

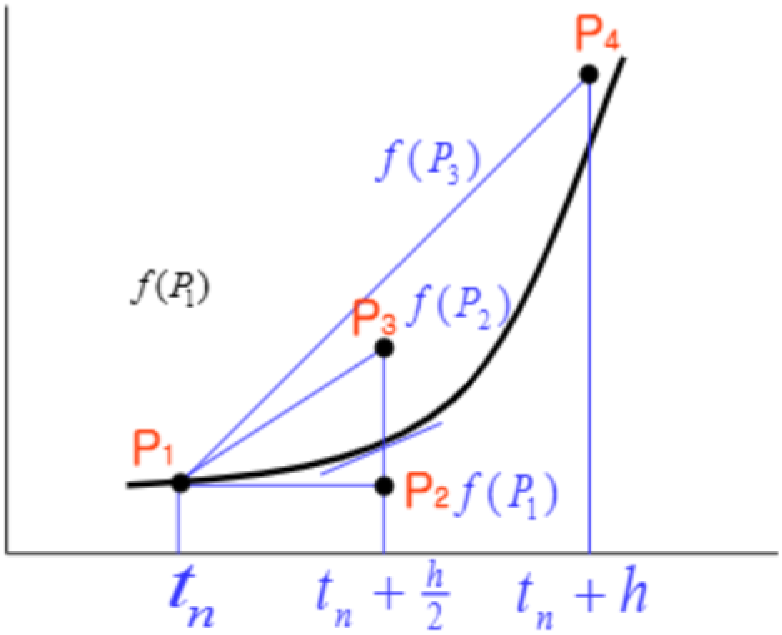
2. Ralston's method: 이 비율대로 하면 오차가 최소화된다. **(**$a_1 = \frac{1}{4}, a_2 = \frac{3}{4}, \alpha_1 = \frac{2}{3}h, \beta_1 = \frac{2}{3}hk_1$)

$$y_{n+1} = y_n + h\left(\frac{1}{4}k_1 + \frac{3}{4}k_2\right)$$
$$k_1 = f(t_n, y_n)$$
$$k_2 = f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_1\right)$$

4th order Runge-kutta method

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\
k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\
k_4 &= f(t_n + h, y_n + hk_3)
\end{aligned}
$$



- k_1 **(시작점 기울기):** 현재 지점(t_n)에서의 기울기.
- k_2 **(중간점 기울기 1):** k_1을 이용해 절반($h/2$)만큼 갔을 때의 기울기를 예측합니다.
- k_3 **(중간점 기울기 2):** 방금 구한 k_2를 이용해 다시 절반($h/2$)만큼 갔을 때의 기울기를 더 정교하게 예측합니다.
- $k_4$$ **(끝점 기울기):** k_3를 이용해 끝까지(h) 갔을 때의 기울기를 예측합니다.