4. Roundoff & Truncation Errors

HoHee Kim

Errors : 수치해법은 근사값을 다루므로 오차발생→참 값을 모르니 오차 또한 정확한 계산이 어려워 오차에 대한 근사값으로 오차 예측

- **Roundoff errors(반올림 오차)** : 컴퓨터가 수용할 수 있는 유효숫자 (significant figures) 자리수의 한계로 발생하는 오차

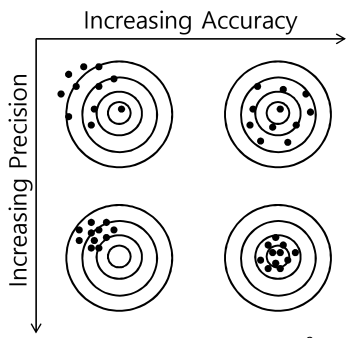
  신뢰를 갖고 사용할 수 있는 숫자

- **Truncation errors(절단오차)** : 수치해법이 수학적 연산의 근사적 방법을 취하므로 발생하는 오차

Accuracy(정확도) : 측정값이 얼마나 참값에 가까운가를 나타냄

Precision(정밀도) : 측정값들이 얼마나 서로 가까이 있는가를 나타냄

☞ 수치해법은 accurate, precise 해야 함



Error Definitions

- **True Error(참 오차)** : E_t = 참값 – 근사값
  흔히 절대치로 표현되므로 절대오차 라고도 함
- **True fractional relative error (참 상대오차)** : $\frac{\text{참값}-\text{근사값}}{\text{참값}}$
  다루는 양의 크기를 고려
- **True percent relative error (참 백분율 상대오차)** :

$$\boxed{\varepsilon_t = \left|\frac{\text{참값}-\text{근사값}}{\text{참값}}\right| \times 100\%}$$

- **Approximate percent relative error (근사 백분율 상대오차)** :
  수치해법은 참값을 모르는 상태에서 오차를 추측하는데
  수치해법 중 반복법에서 현재근사값은 이전근사값을 토대로 만들어지므로

tolerance (허용오차)

$$\boxed{\varepsilon_a = \left|\frac{\text{현재근사값}-\text{이전근사값}}{\text{현재근사값}}\right| \times 100\%}$$

- $|\varepsilon_a| < \varepsilon_s$: 이 조건 만족하면 반복실행 멈춤 (stopping criterion)

Ex) Error estimates for iterative methods to estimate $e^{0.5}$ (ε_s = 0.05%)

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \ldots$$

Maclaurin series

(true value : $e^{0.5}$ = 1.648721)

- 1개 항까지 : $e^{0.5} = 1 \Rightarrow 1$
- 2개 항까지 : $e^{0.5} = 1 + 0.5 \Rightarrow 1.5$

현재 근사값 이전 근사값

$$\varepsilon_t = \left|\frac{1.648721-1.5}{1.648721}\right| \times 100\% = 9.02\%, \qquad \varepsilon_a = \left|\frac{1.5-1}{1.5}\right| \times 100\% = 33.3\%$$

Terms	Result	$\lvert\varepsilon_t\rvert$ %	$\lvert\varepsilon_a\rvert$ %
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	**0.0158**

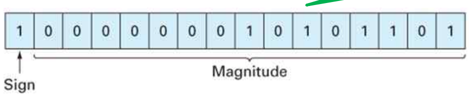
```
function [ fx, ea, iter ] = IterMeth( x, es, maxit )
if nargin<2 | isempty(es), es=0.0001;  end
if nargin<3 | isempty(maxit), maxit=50;  end

% initialization
iter = 1; sol = 1; ea = 100;
% iterative calculation
while (1)
  solold = sol;
  sol = sol + x ^ iter / factorial(iter);
  iter = iter + 1;
  if sol~=0 , ea=abs((sol - solold)/sol)*100; , end
  if ea<=es | iter>=maxit, break, end
end
fx = sol;
end
```


① Roundoff errors : 유효숫자의 생략으로 발생하는 오차

Chopping(잘림) , Rounding(반올림)

- 컴퓨터 내에서 수 체계의 한계로 인하여 발생

Integer 표현 : 2's complement



Floating-point 표현 : 저장공간 낭비를 막기 위해 normalized

$\pm(1+f)X2^{e}$

Mantissa (가수) Exponent (지수)

예) 1101.1 → 1.1011 X 2^3 =(1+0.1011) X 2^3

- 컴퓨터 내에서 실제 산술적 연산 방법으로 인한 발생

어떤 컴퓨터의 4-digit 가수 와 1-digit 지수를 가정, (base-10 으로 설명)

예) 큰 수 + 작은 수의 연산

유효숫자 소멸

$$\begin{array}{rl} & 0.4000 \times 10^4 \\ + & 0.0000001 \times 10^4 \\ \hline & 0.4000001 \times 10^4 \end{array}$$

예) 서로 크기가 비슷한 수 끼리 뺄셈

$$\begin{array}{rl} & 0.7642 \times 10^3 \\ - & 0.7641 \times 10^3 \\ \hline & 0.0001 \times 10^3 \end{array} \Rightarrow 0.1000$$

유효숫자 아님

☞ 크기 비슷한 두수의 뺄셈을 뺄셈의 무효화(subtractive cancellation)

→계속 연산하다 보면 유효숫자인 것처럼 작용하여 반올림오차 유발

예) 연산횟수가 많은 경우

```
 s=0 ;
for i = 1 : 10000
 s = s + 0.0001;
end
disp(s)              % 실행결과 s = 0.99999999999991
```

0.0001 이 2진법으로 정확히 표현 안되므로 반올림오차가 누적

② **Truncation errors** : 수치해법의 근사적 표현으로 발생되는 오차

- **Taylor series :** 함수를 근사적으로 표현하는데 널리 사용

☞ 어떤 점에서의 함수 값을 다른 점에서의 함수 값과 도함수들로 예측

a 와 x 를 포함구간에서 f와 f의 도함수들이 연속이라 가정

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$$

$$f(x_{i+1}) = \underbrace{f(x_i)}_{\text{0차 항}} + \underbrace{f'(x_i)h}_{\text{1차 항}} + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + \underbrace{R_n}_{\text{절단오차}}$$

$$(h = x_{i+1} - x_i)$$

$$\boxed{R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}} \quad \boxed{?}$$

$$(x_i \le \xi \le x_{i+1})$$

급수에 포함될 항의 개수 제어가능,

f(x) 를 얻기 위해 h 를 통해 제어,

$R_n = O(h^{n+1})$이므로 R_n 는 h^{n+1} 에 비례

☞ Taylor series에 기초를 둔 수치해법의 비교오차를 판단하는데 유용

$$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n+R_n$$

절단오차를 다음과 같이 표현가능

$$\boxed{R_n=\int_a^x \frac{(x-t)^n}{n!}f^{(n+1)}(t)\,dt}$$

$$=\left[-f^{(n+1)}(t)\frac{(x-t)^{n+1}}{(n+1)!}\right]_a^x+\int_a^x f^{(n+2)}(t)\frac{(x-t)^{n+1}}{(n+1)!}dt$$

$$=f^{(n+1)}(a)\frac{(x-a)^{n+1}}{(n+1)!}+\left[-f^{(n+2)}(t)\frac{(x-t)^{n+2}}{(n+2)!}\right]_a^x+\int_a^x f^{(n+3)}(t)\frac{(x-t)^{n+2}}{(n+2)!}dt$$

$$=\underbrace{f^{(n+1)}(a)\frac{(x-a)^{n+1}}{(n+1)!}}_{(n+1)\text{차 항}}+\underbrace{f^{(n+2)}(a)\frac{(x-a)^{n+2}}{(n+2)!}}_{(n+2)\text{차 항}}+\int_a^x f^{(n+3)}(t)\frac{(x-t)^{n+2}}{(n+2)!}dt$$

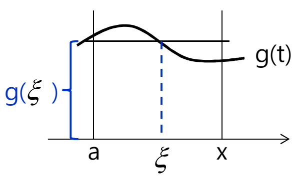
뒤에 계속

<적분의 평균값 제1정리>

함수 g 가 a 와 x 사이에서 연속, 적분가능,

$$\int_a^x g(t)dt=g(\xi)(x-a)$$

a 와 x 사이에 만족하는 ξ 가 존재한다.



<적분의 평균값 제2정리>

함수 g 와 h 가 a 와 x 사이에서 연속, 적분가능,

h(t)가 구간내에서 부호가 바뀌지 않는다면, $\int_a^x g(t)h(t)dt=g(\xi)\int_a^x h(t)dt$

a 와 x 사이에 만족하는 ξ 가 존재한다. (h(t) = 1 경우, 제1정리와 같다)

$$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n+R_n$$

$$R_n=\int_a^x \underbrace{\frac{(x-t)^n}{n!}}_{h(t)}\underbrace{f^{(n+1)}(t)}_{g(t)}dt \;\Rightarrow\; \boxed{R_n=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}}$$

제2정리에 의해

예) 함수 f(x) = e^x 이고, x=1 에서의 함수 근사값($e^1 \cong 2.7???$)이 절단오차가 $5\text{X}10^{-4}$ 보다 작으려면 Maclaurin series 에서 몇 차 항까지 구해야 하나?

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + R_n \qquad R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}$$

: a=0 인 Taylor series

≒ 2.7

$$R_n = \frac{e^{\xi}}{(n+1)!}(1)^{n+1} \le \frac{e^1}{(n+1)!}(1)^{n+1} < 5\times10^{-4}$$

☞ n=7

$$(0 \le \xi \le 1)$$

$$\frac{e^1}{(7+1)!}(1)^{7+1} = 0.00006696$$

☞ 7차 항까지의 근사값이 소수 4번째 까지 정확

$$e^1 = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{7!} = \underline{2.7182}539683$$

$$e^1 = \underline{2.7182}818285$$ (참 값)

Ex) Use Taylor series expansion to approximate $f(x) = \cos x$ at x_{i+1} = π/3 on the basis of the value of f(x) and derivatives at x_i = π/4 .

- Zero-order approximation

$$f(\frac{\pi}{3}) \cong \cos(\frac{\pi}{4}) = 0.707106781 \longrightarrow \varepsilon_t = \left|\frac{0.5 - 0.707106781}{0.5}\right| \times 100\% = 41.4\%$$

- First-order approximation (1차 도함수를 포함하는 항까지)

$$f(\frac{\pi}{3}) \cong \cos(\frac{\pi}{4}) - \sin(\frac{\pi}{4})\underline{(\frac{\pi}{3} - \frac{\pi}{4})} = 0.521986659 \qquad \boxed{h = \frac{\pi}{3} - \frac{\pi}{4} = \frac{\pi}{12}}$$

- Second-order approximation

$$f(\frac{\pi}{3}) \cong \cos(\frac{\pi}{4}) - \sin(\frac{\pi}{4})(\frac{\pi}{12}) - \frac{\cos(\pi/4)}{2}(\frac{\pi}{12})^2 = 0.497754491$$

참 값 : f(π/3) = 0.5

☞ 대부분 몇 개 항만 추가하면 참 값에 근사

Order n	f(π/3)	$\lvert\varepsilon_t\rvert$ %
0	0.707106781	41.4
1	0.521986659	4.40
2	0.497754491	0.449
3	0.499869147	$2.62 \text{ X } 10^{-2}$

Numerical Differentiation (수치미분)

- **Finite-difference approximations of the first derivative**

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + R_1$$

절단오차 공식에서

$$R_1 = \frac{f''(\xi)}{2!}h^2$$

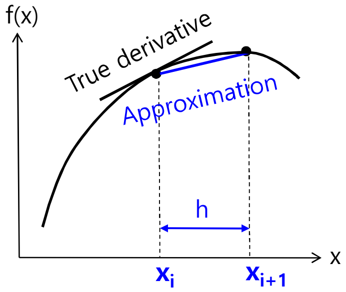
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{R_1}{h}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

Forward finite difference

x_i 와 x_{i+1} 에서의 데이터를 이용해 도함수를 구하므로

$$h = x_{i+1} - x_i$$



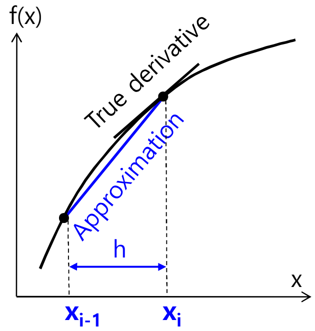
$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \cdots \quad ①$$

여기서 $h = x_i - x_{i-1}$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$$

Backward finite difference

x_i 와 x_{i-1} 에서의 데이터 이용하므로



$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \ldots \quad ②$$

여기서 $h = x_{i+1} - x_i$

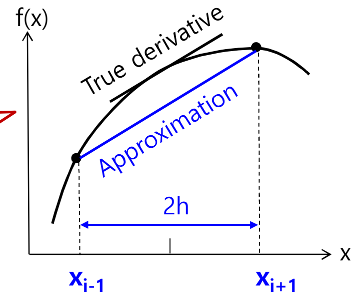
②-① :

$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + 2\frac{f^{(3)}(x_i)}{3!}h^3 + \ldots$$

여기서 $2h = x_{i+1} - x_{i-1}$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2)$$

Centered finite difference



구간길이가 1/2 으로 줄면
오차가 1/4 로 줄어지므로

☞ 도함수 표현에 Centered finite difference 가 더 정확

Ex) Use forward, backward, and centered finite difference approximations to estimate the first derivative of $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$ at x=0.5 using a step size h=0.5 . Repeat the computation using h=0.25 .

→ $f'(x) = -0.4x^3 - 0.45x^2 - 1.0x - 0.25$ 이므로 참 값 : $f'(0.5) = -0.9125$

- h=0.5 경우

$$\begin{bmatrix} x_{i-1} = 0 & f(x_{i-1}) = 1.2 \\ x_i = 0.5 & f(x_i) = 0.925 \\ x_{i+1} = 1.0 & f(x_{i+1}) = 0.2 \end{bmatrix} \rightarrow$$

Forward $f'(0.5) = \frac{0.2 - 0.925}{0.5} = -1.45$

Backward $f'(0.5) = \frac{0.925 - 1.2}{0.5} = -0.55$

Centered $f'(0.5) = \frac{0.2 - 1.2}{1.0} = -1.0$

- h=0.25 경우

$$\begin{bmatrix} x_{i-1} = 0.25 & f(x_{i-1}) = 1.10351563 \\ x_i = 0.5 & f(x_i) = 0.925 \\ x_{i+1} = 0.75 & f(x_{i+1}) = 0.63632813 \end{bmatrix} \rightarrow$$

Forward
$$f'(0.5) = \frac{0.63632813 - 0.925}{0.25} = -1.155$$

Backward $f'(0.5) = \frac{0.925 - 1.10351563}{0.25} = -0.714$

Centered $f'(0.5) = \frac{0.63632813 - 1.10351563}{0.5} = -0.934$

구간길이 줄이니 더 정확

- **Finite-difference approximations of higher derivative**

$$f(x_{i+2}) = f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2!}(2h)^2 + \dots \quad ①$$

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \dots \quad ②$$

①- 2X② : $f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 + \dots$

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h)$$

second forward finite difference

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} + O(h)$$

second backward finite difference

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2)$$

☞ 2차 도함수 표현에 second centered finite difference 가 더 정확

$$f''(x_i) \cong \frac{\dfrac{f(x_{i+1}) - f(x_i)}{h} - \dfrac{f(x_i) - f(x_{i-1})}{h}}{h}$$

21장

☞ 2차 도함수는 1차 도함수의 도함수

Total Numerical Error : 전체 수치오차 = 반올림 오차 + 절단오차

- 반올림 오차 줄이려면 → 컴퓨터의 유효숫자 자리 수 늘리고,
  연산 횟수 줄이기
- 절단오차 줄이려면 → 근사식의 항 수 늘리거나 구간간격을 줄이기

☞ 두 오차는 상반되므로 반올림 오차와 절단오차를 각 각 구해서 그 합이 최소가 되도록 해야 함