韓國數學敎育學會誌<數學敎育>
1982. 12. Vol. XXI. No. 1.

Journal of the Korea Society of Mathematical Education
Dec. 1982. Vol. XXI. No. 1

Computer의 數値計算에 對한 誤差의 推定

白 淸 鎬

1. 緖 論

computer에 依한 數値計算에서 看過하기 쉬운 誤差의 影響을 代數的인 四則演算의 種類別로 推定하고자 한다. 特히 近似値의 거듭된 四則演算의 結果는 誤差로 因해 完全히 意味가 없어지는 境遇도 있기 때문에 誤差를 看過한채 computer에 依한 數値計算結果를 믿는다는 것은 위험스러운 일이다.

本 硏究에서는 便宜性을 위해 十進數만을 使用하지만, 主要原則들은 어느 進數에도 適用될 수 있을 것이다.

2. 近似値와 正確値

近似値(Approximate number)에 對應하는 말로 正確値(Exact number)라는 말을 使用한다. 正確値란 그 값이 하나의 記號로써 完全하게 表現될 수 있는 數로서 모든 有理數 및 π, $\sqrt{2}$, e…等이다. 그러나 모든 正確値가 十進數로서 正確하게 表現될 수 있는 것은 아니며 그것은 2進數나 16進數로 한다 해도 마찬가지이다. 例로서 正確値인 有理數 $\frac{2}{3}$의 十進數 表現은 0.6666…으로 無限小數이다. 小數點 아래로 6을 無限히 쓸 수는 없으므로 十進數로써 正確하게 表現하는 것은 不可能하다. 따라서 그의 近似値로서 .6667 또는 좀 더 正確히 한다면 .666667等으로 쓸 수 있다.

3. 絕對誤差와 相對誤差

正確値 Q의 近似値를 Q_1이라 하면 誤差는 Q_1-Q, 相對誤差는 $(Q_1-Q)/Q$이다. 다음의 境遇에 近似値를 使用한다.

(1) 正確値를 十進數로 正確히 쓸 수 없을 때
(2) 正確値를 알 수 없을 때(測定時)

위의 境遇에는 誤差도 물론 十進數로 正確히 나타낼 수 없으므로 近似値가 正確値에 얼마나 近接해 있는가를 알기 위해 $|Q_1-Q|\leqq\Delta Q$라 할 때 ΔQ와 같은 誤差의 絕對값의 上限을 생각하게 되는데 이것을 앞으로 簡單히 絕對誤差라고 부르며 이는 近似値의 正確度를 말한다.

$|Q|=|Q_1-(Q_1-Q)|\geqq|Q_1|-|Q_1-Q|\geqq|Q_1|-\Delta Q$에서 相對誤差의 크기는 $|Q_1-Q|/|Q|\leqq\Delta Q/|Q|\leqq\Delta Q/(|Q_1|-\Delta Q)$로 되어 $Q_1, \Delta Q$가 주어지면 相對誤差 크기의 上限을 大略 알 수 있고 이것을 簡單히 相對誤差라고 부르며 特히, ΔQ가 $|Q_1|$에 比해 아주 작으면 相對誤差를 簡單히 $\Delta Q/|Q_1|$이라 할 수 있고 이는 近似値의 精密度를 말한다.

4. 有效숫자

十進數의 表現에서 1, 2, 3, …, 9는 恒常 有效숫자이고, 0은 小數點의 位置를 表示하기 위해 使用된 境遇를 除外하고는 언제나 有效숫자이다.

例 1. $0.0158=1.58\times10^{-2}(1,5,8)$,
$4076=4.076\times10^{3}(4,0,7,6)$,
$38500=3.85\times10^{4}(3,8,5)$,
$38500=3.850\times10^{4}(3,8,5,0)$,
$38500=3.8500\times10^{4}(3,8,5,0,0)$
에서 괄호안은 有效숫자이다.

4-1. 近似値를 얻는 方法

近似値를 얻는 方法은 올림, 버림, 반올림의 세가지가 있는데 올림과 버림의 境遇, 絕對誤差는 近似値 最終位 單位數와 같고, 반올림의 境遇는 最終位 單位數의 半과 같아서 絕對誤差가 가장 작으므로 널리 使用된다. computer에서는

반올림과 버림이 使用된다.

例 2. 다음 數들은 小數點아래 셋째 자리에서 반올림한 것이다.

46.12416⇒46.12, 31.34792⇒31.35,

52.27500⇒52.28, 2.38500⇒2.38,

2.38501⇒2.39

위의 例에서 반올림해야 할 數가 近似値 最終位 單位數의 半과 같으면 交番으로 반올림과 버림을 行하여 誤差의 偏重을 막을 수 있다.

大部分의 計算에서는 相對誤差가 絶對誤差 보다 關心이 높으며; 相對誤差는 近似値의 有效숫자의 個數와 關係가 깊다.

n個의 有效숫자를 가진 近似値를 얻으려면 왼쪽에서 세어 n個 未滿의 數를 반올림하거나 버리면 된다(round off). 이 方法은 記憶場所가 限定된 computer에 使用할 近似値를 얻는 方法이다.

例 3. 반올림으로 다음을 4個의 有效숫자를 갖는 近似値로 만들면,

316.8972⇒316.9, 4.167500⇒4.168,

.0011118⇒.001112, 19.265001⇒19.27

4-2. 正確한 有效숫자

絶對誤差가 單位數의 半以下가 되는 有效숫자를 正確한 **有效숫자**(correct significant digit)라 한다. 便宜上 앞으로 이것을 CSD라고 쓰겠다.

例를 들어 $\frac{2}{3}$의 近似値를 .66667 또는 .66699842593으로 할 때 後者는 前者보다 더 많은 有效숫자를 갖고 있으나 前者보다 더 不正確한 表現이다. 前者는 有效숫자 5個, CSD 5個, 後者는 有效 숫자 11個에 CSD는 3個뿐이다. 筆算에서는 近似値를 나타낼 때 正確한 有效숫자만을 쓰겠지만 computer는 記憶에 指定된 자릿수 때문에 有效숫자의 正確性을 不問하고 指定된 자릿수 만큼의 個數의 有效숫자를 지니고 다닌다.

4-3. 相對誤差와 正確한 有效숫자

一般的으로 近似値의 CSD의 個數가 주어지면 相對誤差를 알 수 있고, 그 逆도 成立한다.

例 4. 近似値 34.152의 CSD가 5個일 때 相對誤差를 구하라.

CSD의 定義에 依해 $\Delta Q \leqq 0.0005$이다.

$$\text{相對誤差} = \frac{\Delta Q}{|Q_1| - \Delta Q}$$

$$= \frac{0.0005}{34.152 - 0.0005} = \frac{1}{68303}$$

$$< \frac{1}{10000}$$

윗式에서 $\frac{1}{68303}$ 보다 $\frac{1}{10000}$을 더 자주 쓰게 되는 데, 이것은 前者가 正確하지만 效用性이 낮기 때문이다.

定理 1. n個의 CSD를 가진 近似値의 相對誤差는 5×10^{-n}보다 작다. 단 1과 $n-1$個의 CSD 0으로 된 近似値는 除外한다.

證明 近似値에서 小數點이 마지막 CSD 다음에 있음을 假定하고 Q를 참값, Q_1을 近似値, ΔQ를 絶對誤差라 하면 $\Delta Q \leqq 0.5$이고, Q_1은 1다음 $n-1$個의 0으로 된 數가 아니므로 $|Q_1| \geqq 1\times10^{n-1}+1$

$$\text{相對誤差} = \frac{\Delta Q}{|Q|} \leqq \frac{\Delta Q}{|Q_1| - \Delta Q}$$

$$= \frac{.5}{1\times10^{n-1}+1-0.5}$$

$$= \frac{.5}{1\times10^{n-1}-0.5} < 5\times10^{-n}$$

相對誤差는 小數點의 位置에 關係가 없으므로 證明은 完了되었다.

1다음 $n-1$個의 CSD 0이 오는 境遇의 相對誤差는 다음과 같다.

n	Q_1	相對誤差	定理 1에 依한 값
1	1.	.5/.5=1	1/2
2	10.	.5/9.5=1/(1.9×10)	1/(2×10)
3	100.	.5/99.5=1/(1.99×10²)	1/(2×10²)
4	1000.	.5/999.5=1/(1.999×10³)	1/(2×10³)

$Q_1=1$일 때 定理 1에 依한 相對誤差는 可能한 相對誤差의 半이 되지만 實際로 이것은 問題가 되지 않으므로 定理 1이 모든 境遇에 適用되는 것으로 생각한다. 例 4를 定理 1에 依해 풀면 5個의 CSD가 있으므로 相對誤差는 $5\times10^{-5}=1/20000$ 보다 작다.

例 5. 近似値 31.546824의 相對誤差가 1/100000 以下일 때 CSD의 個數를 推定해보자.

$$\frac{\Delta Q}{|Q|} \leqq \frac{1}{100000}, \quad \Delta Q \leqq \frac{1}{100000}|Q|.$$

$|Q| \leqq |Q_1| + \Delta Q$에서

$$\Delta Q \leqq \frac{1}{100000}(|Q_1| + \Delta Q)$$

$$= 0.00031546824 + \frac{1}{100000}\Delta Q$$

$$\frac{99999}{100000}\Delta Q \leqq 0.00031546824$$

$$\Delta Q \leqq 0.00032$$

ΔQ가 小數點아래 3位 單位數의 半보다 작으므로 小數點아래 3位까지가 CSD라고 推定할 수 있으므로 적어도 5個의 CSD를 갖는다고 할 수 있다.

定理 2. 한 近似値의 相對誤差가 $.5\times10^{-n}$ 以下이면 CSD는 적어도 n個이다.

4-4. computer memory에서의 數의 記憶

Computer의 memory에 記憶되는 數는 fixed point(固定 小數點) 形態이거나 floatiog point(流動 小數點)形態이다. fixed point 形態에서는 2進數의 小數點이 數의 오른쪽 끝에 位置하여 小數部分은 round off 되어야 한다.

floating point 形態에서는 實數部分과 指數部分으로 나누어져 記憶된다. 例로써 1 word 32bit로 된 記憶裝置는 實數部分이 24 bit인데 여기에는 約 七位의 十進數를 記憶시킬 수 있다.

모든 十進數는 $f\times10^e$으로 쓸 수 있다(단 $0\leqq|f|<1$)(normalize). f가 記憶되는 實數部分은 자릿수가 限定되어 確保된다. t를 f가 記憶될 限定確保된 자릿수라 하면 t個의 CSD에 對한 相對誤差는 定理 1에 依하여 5×10^{-t}以下이다.

computer에서는 精密을 要하는 計算을 위해 double precision(倍精度)의 記憶場所를 주어 誤差를 줄일 수 있으나 single precision에 比해 計算速度는 느리다.

5. 四則演算에서의 誤差의 推定

誤差推定이 可能한 두 近似値의 各 演算의 結果에 對한 誤差의 推定도 可能하다. 推定을 위해 먼저 隱數만 取扱한다.

두 近似値 u_1, u_2에 對한 絶對誤差를 各各 Δu_1, Δu_2라 하고 相對誤差를 r_1, r_2라 하면 $u_1-\Delta u_1\leqq u_1$의 참값$\leqq u_1+\Delta u_1$, $u_2-\Delta u_2\leqq u_2$의 참값$\leqq u_2+\Delta u_2$이고 $r_1=\Delta u_1/u_1$, $r_2=\Delta u_2/u_2$이다.

5-1. 덧 셈

u_1+u_2의 참값은 $(u_1+\Delta u_1)+(u_2+\Delta u_2)=u_1+u_2+(\Delta u_1+\Delta u_2)$와 $(u_1-\Delta u_1)+(u_2-\Delta u_2)=u_1+u_2-(\Delta u_1+\Delta u_2)$ 사이에 있으므로 絶對誤差는 $\Delta u_1+\Delta u_2$이다. u_1+u_2의 相對誤差를 r라 하면

$$r=\frac{\Delta u_1+\Delta u_2}{u_1+u_2}=\frac{u_1}{u_1+u_2}\left(\frac{\Delta u_1}{u_1}\right)+\frac{u_2}{u_1+u_2}\left(\frac{\Delta u_2}{u_2}\right)$$

$$=\frac{u_1}{u_1+u_2}r_1+\frac{u_2}{u_1+u_2}r_2 \quad \cdots\cdots(1)$$

여기서 $\theta=\dfrac{u_1}{u_1+u_2}$이라 하면

$$r=\theta r_1+(1-\theta)r_2 \quad \cdots\cdots(2)$$

이것은 線型補間法의 公式이며 $0\leqq\theta\leqq1$이므로 r의 값은 r_1과 r_2 사이에 存在한다.

例 1. 近似値 476.6과 3.11918의 合을 구하자. floating point 形態를 위한 computer 記憶場所의 實數部分 자릿수 $t=7$이라 하면 두 數는 記憶을 위해 $.4766000\times10^3$, $.3119180\times10$으로 되고 덧셈을 위해 指數部分을 같게 해야 하므로 後者가 $.0031191\times10^3$으로 자리가 移動되며, 끝자리 숫자 8이 버려지면서 round off 誤差가 發生하여 絶對誤差, 特히 相對誤差가 크게 增加하게 된다. 덧셈의 結果는 $.4797191\times10^3=479.7191$이 되지만 實際로는 小數 1位까지만 正確하므로 CSD는 4, 7, 9, 7의 4個이다.

小數 t자리까지 記憶할 수 있는 computer에서의 덧셈 結果는 小數 t자리만 記憶되므로 $5\times10^{-t}(u_1+u_2)$ 만큼의 round off 誤差가 생긴다. 따라서 相對誤差는 (1)式 代身,

$$r=\frac{u_1}{u_1+u_2}r_1+\frac{u_2}{u_1+u_2}r_2+5\times10^{-t} \quad \cdots\cdots(3)$$

이 된다.

5-2. 뺄 셈

$u_1>u_2$일 때 u_1-u_2의 참값은 $u_1+\Delta u_1-(u_2-\Delta u_2)=(u_1-u_2)+(\Delta u_1+\Delta u_2)$와 $u_1-\Delta u_1-(u_2+\Delta u_2)=(u_1-u_2)-(\Delta u_1+\Delta u_2)$ 사이에 있으므로, 差의 絶對誤差는 $\Delta u_1+\Delta u_2$이고 相對誤差를 구하면

$$r=\frac{\Delta u_1+\Delta u_2}{u_1-u_2}$$

$$=\frac{u_1}{u_1-u_2}r_1+\frac{u_2}{u_1-u_2}r_2 \quad \cdots\cdots\cdots\cdots\cdots\cdots(4)$$

$$=\frac{u_1+u_2}{u_1-u_2}\left(\frac{u_1}{u_1+u_2}r_1+\frac{u_2}{u_1+u_2}r_2\right) \cdots\cdots(5)$$

이다. (5)式에서 $\frac{u_1+u_2}{u_1-u_2}\geqq 1$이며 u_1, u_2가 가까운 값이면 r도 r_1, r_2보다 매우 커지게 되는데 이 點은 數値計算에서 매우 重要하다.

例 1. 近似値 754.8에서 37.68151을 빼보자. $t=7$이라면 記憶을 위해서 $.7548000\times10^3$, $.3768151\times10^2$으로 normalize되고, 뺄셈을 위해 後者는 $.0376815\times10^3$이 되면서 round off 誤差가 발생, 뺄셈의 結果는 $.7171185\times10^3=717.1185$가 되고 小數 1位까지가 CSD이며 round off 誤差로 因하여 相對誤差는 (4)式 代身,

$$r=\frac{u_1}{u_1-u_2}r_1+\frac{u_2}{u_1-u_2}r_2+5\times10^{-t} \quad \cdots\cdots(6)$$

이 된다.

例 2. 近似値 938.67827에서 938.67804를 빼보자. $t=7$이라 하면 記憶을 위해 $.9386782\times10^3$, $.9386780\times10^3$로 되고 뺄셈의 結果는 $.0000002\times10^3$으로서 normalize되면 $.2000000\times10^{-3}$이 되어 元來의 數는 CSD가 8個였으나 結果는 한 個뿐이다. 어떤 computer에서는 그 뒤의 6位에 다른 register의 값을 移動시켜 0이 아닌 CSD처럼 보이게 하는 수도 있으나, 이 結果値가 다시 演算에 使用된다면 最終結果의 正確度는 매우 떨어지게 된다.

近接한 두 近似値間의 뺄셈에서 CSD의 喪失은, 大部分의 計算에서 不正確의 큰 要因이지만 computer는 無防備 狀態이므로 program上의 防備나 programmer의 注意가 必要하다.

例 3. $a=1.00020000$, $b=1.00010000$, $c=0.00020001$일 때 a^2-b^2-c를 計算해 보자. $t=7$일 때 $a=.1000200\times10$, $b=.1000100\times10$, $c=.2000100\times10^{-3}$이 되며 一般 代數的인 演算順位에 따라 計算하면 $a^2=.1000400\times10, b^2=.1000200\times10$, $a^2-b^2=.0000200\times10=.2000000\times10^{-3}$이며 $a^2-b^2-c=-.0000100\times10^{-3}=-.1000000\times10^{-7}$ 即, 最終結果値는 $-.0000001$이다. 그러나 $t=14$(double precision)이면 倍精度이므로

$a^2=.10004000400000\times10$,

$b^2=1000200100000\times10$,

$a^2-b^2=.00002000300000\times10$

$=.20003000000000\times10^{-3}$

이며, $a^2-b^2-c=.00002000000000\times10^{-3}$

$=.200000000000000\times10^{-7}$

$=.00000002$

가 되어 $t=7$일 때의 값과는 符號마저 달라서 $t=7$일 때의 값은 아무 意味가 없을 뿐 아니라 오히려 해롭다. 그러나 $t=7$인 single precision에서도 一般的인 代數的 演算順序를 달리함으로써 正確한 結果를 얻을 수 있다.

$a^2-b^2-c=(a+b)(a-b)-c$이므로

$a+b=.2000300\times10$, $a-b=.0000100\times10$

$=.10000000\times10^{-3}$,

$(a+b)(a-b)=.2000300\times10^{-3}$,

$(a+b)(a-b)-c=.0000200\times10^{-3}$

$=.2000000\times10^{-7}$.

이것은 $t=14$일 때 얻은 結果와 같다. 結局 演算順位도 誤差에 作用됨을 알 수 있다.

5-3. 곱 셈

곱 u_1u_2의 참값은 $(u_1+\Delta u_1)(u_2+\Delta u_2)=u_1u_2+u_1\Delta u_2+u_2\Delta u_1+\Delta u_1\Delta u_2$와 $(u_1-\Delta u_1)(u_2-\Delta u_2)=u_1u_2-u_1\Delta u_2-u_2\Delta u_1+\Delta u_1\Delta u_2$ 사이에 있다, 여기서 $\Delta u_1\Delta u_2$는 比較的 작은 값이므로 無視하면, 곱의 絶對誤差는 $u_1\Delta u_2+u_2\Delta u_1$이고 相對誤差는

$$\frac{u_1\Delta u_2+u_2\Delta u_1}{u_1u_2}=\frac{\Delta u_1}{u_1}+\frac{\Delta u_2}{u_2}$$

로 近似된다. 그러나 자릿수 t에 對한 round-off 誤差가 있으므로 곱 u_1u_2의 相對誤差 $r=r_1+r_2+5\times10^{-t}$이다.

5-4. 나눗셈

$\frac{u_1}{u_2}$의 몫의 참값은 $\frac{u_1+\Delta u_1}{u_2-\Delta u_2}$과 $\frac{u_1-\Delta u_1}{u_2+\Delta u_2}$ 사이에 있다. 처음 式의 分母, 分子에 $u_2+\Delta u_2$를, 둘째 式의 分母, 分子에 $u_2-\Delta u_2$를 各各 곱하면

$$\frac{u_1u_2+u_1\Delta u_2+u_2\Delta u_1+\Delta u_1\Delta u_2}{u_2^2-\Delta u_2^2},$$

$$\frac{u_1u_2-u_1\Delta u_2-u_2\Delta u_1+\Delta u_1\Delta u_2}{u_2^2-\Delta u_2^2}$$

가 되는데 $\Delta u_1\Delta u_2$, Δu_2^2의 값은 작으므로 無視하

면 첫 式은

$$\frac{u_1u_2+u_1\Delta u_2+u_2\Delta u_1}{u_2^2}=\frac{u_1}{u_2}+\frac{u_1}{u_2}\left(\frac{\Delta u_2}{u_2}+\frac{\Delta u_1}{u_1}\right)$$

이고 두번째 式은

$$\frac{u_1u_2-u_1\Delta u_2-u_2\Delta u_1}{u_2^2}=\frac{u_1}{u_2}-\frac{u_1}{u_2}\left(\frac{\Delta u_2}{u_2}+\frac{\Delta u_1}{u_1}\right)$$

이 되어 몫의 絶對誤差는 $\frac{u_1}{u_2}\left(\frac{\Delta u_2}{u_2}+\frac{\Delta u_1}{u_1}\right)$이고 相對誤差를 구하기 위해 $\frac{u_1}{u_2}$로 나누면 $r=\frac{\Delta u_2}{u_2}+\frac{\Delta u_1}{u_1}$이다. 結果는 곱셈과 같으므로 몫의 相對誤差 $r=r_1+r_2+5\times10^{-t}$이다.

6. 誤差의 傳播

지금까지 陽數에 對한 誤差를 推定하였으나 그 公式은 陰數에 對해서도 마찬가지로 適用될 수 있음은 分明하다.

6-1. 演算의 順序

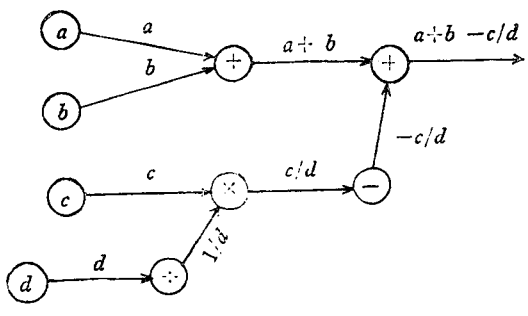
5-2의 例 3에서와 같이 演算順序에 따라 結果가 전혀 달라지는 수가 있다. computer의 演算에서 덧셈, 곱셈의 交換法則은 成立하지만 結合, 配分法則은 반드시 成立하지는 않는다.

演算의 段階에서 괄호를 쓰면 精密度에 도움을 줄 수 있다.

6-2. 樹形圖

演算을 各 마디에, 演算結果를 各 線에 表示함으로써 代數的인 演算式을 樹形圖로 나타낼 수 있다.

例 1. $(a+b)-c/d$의 樹形圖



다음과 같은 規則을 따르면 誤差推定을 樹形圖를 通해 쉽게 구할 수 있다.

(1) 符號交換마디나 逆數마디는 相對誤差와 無關하다.

(2) 곱셈마디는 들어오는 두 線의 相對誤差가 더해지고 5×10^{-t}의 round off 誤差가 더해진다.

(3) 덧셈마디는 들어오는 값의 크기에 따른 比重이 주어져서 相對誤差가 더해지고 round off 誤差 5×10^{-t}이 더해진다.

例 2. $y_1=(a+(b+(c+d)))$의 誤差를 推定하라. 단 a,b,c,d는 正確한 陽數이다.

a,b,c,d의 相對誤差는 0이므로 $c+d$의 相對誤差는 5×10^{-t}이고, $b+c+d$의 相對誤差는 $\frac{c+d}{b+c+d}\times5\times10^{-t}+5\times10^{-t}$이며 $a+b+c+d$의 相對誤差는 $\frac{b+c+d}{a+b+c+d}\left(\frac{c+d}{b+c+d}\times5\times10^{-t}+5\times10^{-t}\right)+5\times10^{-t}=\frac{a+2b+3c+3d}{a+b+c+d}\times5\times10^{-t}$이다.

위의 結果는 $c+d$와 같이 먼저 더해진 數들은 나중에 더해진 數들보다 誤差의 比重이 높음을 나타낸다. 따라서 여러 數의 덧셈에서는 작은 數부터 더해지도록 하는 것이 좋다.

$y_2=(a+b)+(c+d)$를 例 2와 같은 條件으로 誤差를 推定하면 a,b,c,d의 相對誤差는 0, $a+b$, $c+d$의 相對誤差는 5×10^{-t}이며 $(a+b)+(c+d)$의 相對誤差는

$$\frac{a+b}{a+b+c+d}\times5\times10^{-t}+\frac{c+d}{a+c+c+d}\times5\times10^{-t}$$

$$+5\times10^{-t}=\frac{2a+2b+2c+2d}{a+b+c+d}\times5\times10^{-t+1}$$

이것을 例 2의 y_1과 比較하면 y_1,y_2는 代數的으로는 같은 演算式이지만 $a<c+d$이면 y_1의 相對誤差가 더 크고 그렇지 않으면 y_2의 相對誤差가 더 크다.

6-3. 最大誤差에 對한 期待값

只今까지 推定한 誤差는 最大(最惡)의 可能한 境遇에 對한 것이었으나 大部分의 實際誤差는 公式에 依한 것보다는 훨씬 작을 것이다. 例를 들어 round off 誤差는 덧셈, 곱셈의 各 段階마다 5×10^{-t}만큼씩 正確히 發生하지만, round off가 正確히 進行되면(반올림) 誤差는 陽數 또는 陰數가 되어 덧셈演算에서는 相殺될 수도 있다.

確率論에서 Gauss의 誤差의 法則에 따르면 많은 演算에서 發生한 round off 誤差의 合은 正規分布를 따른다. 즉, -5×10^{-t}과 5×10^{-t} 사이에

任意로 存在하는 N個의 round off 誤差의 合은 平均이 0, 標準偏差가 $\sqrt{N/3}\times5\times10^{-t}$인 正規分布를 따르게 되고, 例를 들어 誤差가 標準偏差의 6倍, 즉 $\sqrt{12N}\times5\times10^{-t}$을 넘지 않을 確率은 0.999999998이다. 萬一 $N=10^7$이면 앞에서의 公式은 round off 誤差에 依해 7個의 有效숫자를 잃을 것으로 推定되지만(最惡의 境遇), 確率的인 推定에서는 4個以上의 有效숫자를 잃을 기회는 매우 적게 된다.

參 考 文 獻

1. Wilfred J. Dixon and Frank J. Massey, Jr., *Introduction to Statistical Analysis*, 3rd ed., Tokyo Kogakusha(McGraw-Hill), 1969.
2. Ralph H. Pennington, *Introductory Computer Methods and Numerical Analysis*, 2nd Ed., Macmillan, 1970.
3. Ira Pohl and Alan Shaw, *The Nature of Computation*, Rockville, Md., Computer Science Press, 1981.