1 Hiba, pontosság, stabilitás

A számítógépek a számokat véges pontossággal, általában lebegőpontos alakban tárolják. A véges pontosság azt jelenti, hogy az aritmetikai műveletek eredményeként kapott érték nem egyezik meg a művelet egzakt értelemben vett eredményével, hanem csak megközelíti azt. Ha egy $\mathcal{A}$ számot közelítő értékével a-val ábrázolunk, akkor ennek az ábrázolásnak az abszolút hibája

$$\Delta = |\mathcal{A} - a|. \tag{1}$$

Az ábrázolás relatív hibája

$$\delta = \frac{\Delta}{|\mathcal{A}|}. \tag{2}$$

Ha egy számítógép a tízes számrendszert használja, és a számokat 8 tizedesjegy pontossággal tárolja, akkor az $1/3$ művelet eredménye 0.33333333 lesz, ami több mint $3{\cdot}10^{-9}$-nel tér el a valódi értéktől. Egy tízes számrendszerben felírt szám értékes jegyeinek nevezzük azokat a számjegyeket, amelyek nem nulla értékűek, valamint azokat a nulla számjegyeket, amelyek értékes jegyek között vannak.

Mivel a számítógépek többnyire nem a tízes számrendszert használják, hanem a kettest, ezért nem a tárolt tizedes jegyek számát szokták az ábrázolás pontosságának jellemzésére használni. Általában az ábrázolás pontosságának szokták nevezni azt az ϵ_m legkisebb abszolút értékű valós számot, amelynek ha a lebegőpontos alakját hozzáadjuk az 1.0 valós szám lebegőpontos alakjához, akkor az eredmény eltér 1.0-tól. Ez a szám 32 bites mantissza esetén $\epsilon_m = 3 \cdot 10^{-8}$ érték. Azt lehet mondani, hogy az ϵ_m érték a számok lebegő pontos ábrázolás tört részének a pontossága. Ennek a pontosságnak, és az aritmetikai műveletek pontosságának elegendően jónak kell lennie ahhoz, hogy az eredményünk a számítás végén elfogadható legyen. Ez nem jelenti azt, hogy ha két tizedes jegy pontosságú eredményre van szükségünk, akkor $\epsilon_m = 10^{-2}$ megfelel. A számítás során fellép ugyanis egy ún. kerekítési hiba, amelynek az értéke legalább ϵ_m, és amely a számítások során halmozódik. Ha egy számításhoz N aritmetikai műveletre van szükségünk, akkor szerencsés esetben $\epsilon_m \approx \sqrt{N}$ nagyságú kerekítési hibára számíthatunk.

Az elemi aritmetikai műveleteket figyelembe véve azt lehet mondani, hogy közelítő számok összegének abszolút hibája nem haladja meg a számok abszolút hibáinak összegét,

$$\Delta\left(\sum a_i\right) \leq \sum \Delta_{a_i} \tag{3}$$

és közelítő számok különbségének abszolút hibája nem haladja meg a két szám abszolút hibáinak összegét, azaz

$$\Delta_{a-b} \leq \Delta_a + \Delta_b. \tag{4}$$

Egy másik hibalehetőség, amelyet azonban elkerülhetünk, ha két egymáshoz közeli számot vonunk ki egymásból. Ilyen esetben az értékes bitek nagy része azonos, és csak néhány olyan bit lesz, amelyik valóban értékes eredményt ad. Így az eredményül kapott szám pontossága jóval alatta lesz a valódi számábrázolási pontosságnak. Ezt úgy is megérthetjük, ha megvizsgáljuk a kivonás eredményeként kapott abszolút, és relatív pontosságot. Ha két közel azonos számot vonunk ki egymásból, akkor az abszolút hiba ugyan nem nő meg radikálisan, de a relatív hiba igen, ugyanis a

$$\delta_{a-b} = \frac{\Delta_{a-b}}{|a-b|} \tag{5}$$

képletben $a - b$ értéke igen kicsi lehet.

Azt hihetnénk, hogy ilyen kivonás csak nagyon ritkán fordul elő. Néhány képlet azonban magában hordozza az ilyen jellegű kivonásokat.

Ennek példájaképpen oldjunk meg egy másodfokú egyenletet! A megoldás során a számolási pontosság ne legyen több, mint 10^{-2}.

$$(x-100)(x-0.1) = x^2 - 100.1x + 10 = 0 \tag{6}$$

Ha csak két értékes jegy pontossággal számolunk, akkor az egyenlet alakja:

$$(x-100)(x-0.1) = x^2 - 100x + 10 = 0 \tag{7}$$

Használva a jólismert megoldó képletet:

$$x_{1,2} = \frac{100 \pm \sqrt{100^2 - 4 \cdot 10}}{2} \tag{8}$$

Kiszámolva a determinánst:

$$x_{1,2} = \frac{100 \pm \sqrt{10000}}{2} \tag{9}$$

A helyes érték persze 9960 *lenne, de mivel csak két tizedes jegy pontossággal számolunk a* 40*-t hiába vonjuk le, a detemináns értékén nem változtat. A megoldóképlet ezek után így alakul:*

$$x_{1,2} = \frac{100 \pm 100}{2} \tag{10}$$

Ebből:

$$x_1 = \frac{100 + 100}{2} = 100 \tag{11}$$

és

$$x_2 = \frac{100 - 100}{2} = 0 \tag{12}$$

Ennek a második eredménynek a relatív hibája 100%. Ha ebben az esetben nem a

$$x_2 = \frac{-b - \sqrt{b^2 - 4 \cdot a \cdot c}}{2a} \tag{13}$$

képletet használjuk, hanem ismerve x_1, valamint az együtthatók a, b, c értékeit, a

$$x_2 = \frac{c}{x_1} \tag{14}$$

képletet, akkor kiküszöböljük azt a hibát, amely az utolsó lépésben teljes pontosságvesztéshez vezetett, és megkapjuk a helyes 0.1 eredményt.

További elemi aritmetikai műveleteket megvizsgálva, azt tapasztaljuk, hogy közelítő számok szorzatának relatív hibája nem haladja meg a számok relatív hibáinak összegét, két szám hányadosának relatív hibája nem haladja meg a számok relatív hibáinak összegét, valamint egy szám m-edik hatványának relatív hibája a szám relatív hibája az m-ediken.

Ha nem csak az elemi műveletekre szorítkozunk, hanem általánosabb állítást akarunk kimondani, akkor az mondható, hogyha egy

$$u = f(x_1, x_2, \ldots, x_n) \tag{15}$$

függvényben a paraméterek abszolút hibái $\Delta_{x_1}, \Delta_{x_2}, \ldots, \Delta_{x_n}$, akkor az eredmény abszolút hibája

$$\Delta_u = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| \Delta_{x_i} \tag{16}$$

Van egy másik hibafajta is, amelyet a program határoz meg. A legtöbb numerikus módszer diszkrét módon közelít valamilyen folytonos mennyiséget. Egy integrált például úgy számítunk ki numerikusan, hogy kiszámítjuk a függvény értékét véges sok pontban, de korántsem minden pontban. Egy másik példa, amikor egy függvény értékét számítjuk ki valamely a függvényt közelítő sor első néhány tagjának összegeként, de természetesen sohasem veszünk végtelen sok tagot. Ezekben az esetekben van egy paraméterünk, az első esetben a pontok száma, a másodikban a tagoké, amelyeket megfelelően nagyra választva, az eredmény megfelelő potossággal kapjuk meg. Ennek a paraméternek a megválasztása persze egy kompromisszum eredménye, mert igaz ugyan, hogy minél nagyobb ez a paraméter, annál pontosabb az eredmény de annál tovább is fut a program. Azt a hibát, amely így adódik, csonkítási hibának hívják. Ez a csonkítási hiba még akkor is meglenne, ha a számítógépünk abszolút pontossággal tárolná a számokat, és végezné a műveleteket. Addig, amíg a kerekítési hibával a programozó semmit sem tud kezdeni, a csonkítási hiba teljesen a hatalma alatt van. Egy kis túlzással azt állíthatjuk, hogy a numerikus analízis nem más, mint ennek a hibának a minimalizálása. A kerekítési és csonkítási hiba az esetek döntő többségében független egymástól. Egy számítás hibája kezelhető úgy, mint a csonkítási hiba, amely egy végtelenül pontosan számoló számítógépen keletkezne, plusz a kerekítési hiba, amely a műveletek számától függ.

Ugyanakkor, néha mégis előfordul, hogy egy máskülönben vonzó eljárás nem stabilis. Ez azt jelenti, hogy a kerekítési hiba, amely a számítás egy korai stádiumában keletkezik, a számítás során megnő, és a számítás végére az eredményként kapott számokban dominánssá válik. Egy ilyen nem stabilis eljárás nagyon jól használható lenne egy abszolút pontos számítógépen. Nézzünk meg egy egyszerű, bár kissé mesterkélt példát!

Tegyük fel, hogy ki szeretnénk számolni az aranymetszési arány egész hatványait. Ez az arány:

$$\Phi = \frac{\sqrt{5}-1}{2} \approx 0.61803398 \tag{17}$$

Egyszerűen belátható, hogy $\Phi^{n+1} = \Phi^{n-1} - \Phi^n$. Ha tudjuk az első két értéket, $\Phi^0 = 1.0$ és $\Phi^1 \approx 0.61803398$, akkor könnyen, és gyorsan, csupán kivonásokat alkalmazva lassú szorzások helyett, egymás után kiszámíthatjuk Φ egész hatványait. Sajnos ennek a rekurzív egyenletnek van egy másik megoldása is, $\Phi = -\frac{1}{2}(\sqrt{5}+1)$. Mivel az összefüggés lineáris, és mivel ennek a megoldásnak az abszolút értéke nagyobb mint 1, bármilyen kicsiny kerekítési hiba exponenciális gyorsasággal nő. Egy tipikus, 32 bites mantisszával rendelkező számábrázolásnál, $n = 16$-nál már teljesen hibás eredményt ad. Az összefüggés nem stabilis, és ezért nem használható Φ egész hatványainak kiszámításához.