Kapitel 2. Feluppskattning och räknenoggrannhet

Sedan datorerna togs i bruk på 1950–talet, har det blivit möjligt att utföra beräkningar i långt större skala än tidigare. Liksom vid beräkningar för hand (eller med räknemaskiner) kan också datorberäkningar leda till felaktiga resultat, men orsaken till dessa fel är ofta av ett annat slag. Detta beror på, att det utrymme, som reservats för varje tal i datorns minne, alltid är begränsat. Avrundningsfel är därför mycket viktiga, och kan lätt leda till helt oanvändbara resultat. Som vi tidigare nämnt, insågs detta redan 1947 av v. Neumann och Goldstine. Ett exempel på detta är Taylor-utvecklingen av en exponentialfunktion, som för negativa värden av argumentet är en alternerande serie. Kancelleringar leder till onogranna resultat.

Ett annat exempel är ekvationssystemet

$$1.985x - 1.358y = 1$$

$$0.953x - 0.652y = 0.$$

Om detta ekvationssystem löses enligt *Cramers regel* på en hypotetisk dator med sex siffrors räknenoggrannhet fås resultatet

$$x = 13040.0$$

$$y = 19060.0,$$

medan det korrekta svaret är approximativt

$$x = 14173.9$$

$$y = 20717.4.$$

I många fall går det att undvika dessa avrundningsproblem genom omskrivning av det uttryck, som skall beräknas. Ett exempel därpå är $\sqrt{x+\epsilon} - \sqrt{x}$, där ϵ är ett litet tal. Om t.ex $x = 10000$, $\epsilon = 10^{-7}$, och vi räknar med en noggrannhet av tio siffror, så blir uttrycket 0. Genom omskrivning kan uttrycket lätt bringas i formen

$$\frac{\epsilon}{\sqrt{x+\epsilon} + \sqrt{x}},$$

som efter beräkning blir $5 \cdot 10^{-10}$.

Detta är några exempel på problem som har dålig "kondition" (*ill-conditioned problems* på engelska). Vi skall senare stöta på flera andra sådana problem.

2.1. Talframställning i datorer

När man gör beräkningar för hand använder man vanligen tiopotensframställningen, men det är naturligtvis också möjligt att använda andra baser än tio. Varje naturligt tal $b \geq 2$ kan användas som bas.

Den vanligaste talframställningen i en dator är den s.k. flyttalsrepresentationen[1]. Flyttalsrepresentationen karaktäriseras av en heltalig bas b och en precision p. Om t.ex. $b = 10$ och $p = 4$, så kan talet 0.1 uttryckas som $1.000 \cdot 10^{-1}$. Om $b = 2$ och $p = 24$, så kan man inte framställa talet 0.1 exakt, utan endast approximativt som $1.10011001100110011001101 \cdot 2^{-4}$. I allmänhet kan ett flyttal anges som $\pm d.dd \ldots d \cdot b^e$, där $d.dd \ldots d$ som kallas signifikanden (tidigare mantissan) har p siffror.

Allmänt kan man visa, att varje positivt reellt tal har en entydig framställning av formen

$$a_n b^n + a_{n-1} b^{n-1} + \ldots + a_0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots,$$

där koefficienterna a_i är positiva heltal, som uppfyller villkoret $0 \leq a_i \leq b - 1$. Termen flyttal används för att beteckna ett reellt tal som kan representeras i ett givet format av detta slag.

[1] En utförligare framställning är David Goldberg: *What every computer scientist should know about floating-point arithmetic*, `http://portal.acm.org/citation.cfm?id=103163`

På grund av att räkneoperationerna oftast blir enklare, ju mindre basen är, har man valt basen 2 (det *binära talsystemet*) som en standard för de flesta datamaskinerna.

Om talen uttrycks i det binära talsystemet, blir de givetvis mycket längre än i decimalsystemet, och man brukar därför ofta ordna de binära siffrorna i grupper om tre eller fyra, vilket är detsamma som att använda 8 (det *oktala* systemet) eller 16 (det hexadecimala systemet) som bas. Några exempel på förvandlingar mellan systemen:

$$(13.25)_{10} = (1101.01)_2 = (15.2)_8 = (D.4)_{16}.$$

Siffrorna i det binära systemet kallas *bitar*, som kommer från det engelska uttrycket "binary digit" . Åtta påvarandra följande bitar brukar kallas "byte", och hälften därav kallas ibland "nibble" ("munsbit").

Två parametrar, som brukar associeras med flyttalsrepresentationen, är den största resp. minsta exponenten, $e_{\max}$ och $e_{\min}$. Eftersom antalet möjliga signifikander är b^p, och antalet möjliga exponenter $e_{\max} - e_{\min} + 1$, så kan antalet bitar i ett flyttal uttryckas som

$$\lceil \log_2(e_{\max} - e_{\min} + 1) \rceil + \lceil \log_2(b^p) \rceil + 1,$$

där $\lceil x \rceil$ betecknar det minsta heltal, som är större eller lika med x ("ceiling", eller "entier"-funktionen). Signifikandens teckenbit har också beaktats.

Ett reellt tal kan inte alltid uttryckas exakt i flyttalsform. Det vanligaste fallet har vi redan studerat då vi försökte uttrycka 0.1 i binär form. Fastän talets decimalframställning är avslutad, så är den binära framställningen ett oavslutat binärt tal. Ett mindre vanligt fall är att det reella talet är så stort, att dess absoluta värde är större än $b \cdot b^{e\max}$ eller så litet, att det är mindre än $1.0 \cdot b^{e\min}$.

Flyttalsrepresentationen är inte nödvändigtvis unik. Både $0.01 \cdot 10^1$ och $1.00 \cdot 10^{-1}$ representerar sålunda talet 0.1. Om den ledande siffran (a_n i framställningen ovan) är olika noll, så säges framställningen vara normaliserad. Sålunda är flyttalet $1.00 \cdot 10^{-1}$ normaliserat, medan $0.01 \cdot 10^1$ inte är det. Om man kräver, att flyttalet skall vara normaliserat, så blir framställningen unik. Detta leder dock till, att det blir omöjligt att framställa talet 0! Man brukar därför framställa talet 0 i formen $1.0 \cdot b^{e\min -1}$. Då exponenten lagras i ett fält med vidden k, så kan man endast använda $2^k - 1$ heltal som exponenter, eftersom ett av dem används för att representera noll.

För att framställa flyttal används numera huvdsakligen IEEE-standarden, varav det existerar två varianter. IEEE 754 är en binär standard, som kräver $b = 2$ och $p = 24$ i enkel precision och $p = 53$ i dubbel precision. IEEE 854 specificerar endast $b = 2$ eller $b = 10$, medan precisionen inte är fastställd (däremot fastställs gränser för de tillåtna värdena av p i enkel och dubbel precision). Att denna standard tillåter $b = 10$ är naturligt, med tanke på den mänskliga kommunikationen och användningen av decimalsystemet t.ex. i fickräknare.

För datorer lämpar sig dock $b = 2$ bättre än andra baser, bl.a. på grund av att precisionen blir bättre. Ett större värde av basen är oförmånligare. Likväl användes på IBM 370 basen $b = 16$, antagligen på grund av att exponentens räckvidd då blev större.

IEEE-standarden tillåter fyra olika slags precision: enkel precision, dubbel precision, utvidgad enkel precision och utvidgad dubbel precision. I IEEE 754 upptar enkel precision ett 32-bitars ord, medan dubbel precision kräver två konsekutiva 32-bitars ord. De utvidgade formaten kräver större utrymme.

Nedan visas en sammanställning av formatparametrarna för IEEE 754:

Parameter	enkel prec.	utvidgad	dubbel prec.	utvidgad
p	24	32	53	64
e_{max}	$+127$	$+1023$	$+1023$	$> +16383$
e_{min}	-126	≤ -1022	-1022	≤ -16382
Exponentvidd	8	≤ 11	11	15
Formatvidd	32	43	64	79

Antalet siffror som kan användas i signifikanden och exponenten bestäms av datorns ordlängd. En vanligen förekommande ordlängd är 32 bitar, eller 4 bytes. Ett tal uttryckt i binär potensform: $a = r \cdot 2^m$, där r betecknar signifikanden och m exponenten framställs då i enkel precision med 24 bitar i signifikanden och 8 bitar i exponenten (se tabellen ovan). I signifikanden och exponenten ingår också förtecknen.

Vanligen reserveras en bit för förtecknet i signifikanden, även om man ibland använder tvåans komplement för heltal. Flyttal är alltid normaliserade, varför den mest signifikanta biten i signifikanden alltid är 1. För att spara utrymme utsätts den inte ("dold" bit).

I IEEE standarden lagras exponenten som ett heltal i intervallet $[0, 255]$, så att 127 måste subtraheras för att man skall få det riktiga värdet. Det största positiva talet som kan representeras på detta sätt är $1.11\ldots1 \cdot 2^{127} \approx 10^{38}$, och det minsta positiva talet är $\approx 10^{-38}$, dvs $e_{max} = +127$ och $e_{min} = -126$ i tabellen. Om en beräkning leder till ett resultat, som ligger utanför dessa gränser, får man antingen "överflöde" eller "underflöde". Observera, att man avsiktligt har valt $|e_{\min}| < e_{\max}$ för att inversa värdet av det minsta tal som kan framställas ($2^{-e_{min}}$) inte skall leda till överflöde. Beräkning av inversa värdet av det största tal som kan framställas kommer visserligen att leda till underflöde, men detta är oftast inte så allvarligt som ett överflöde. Signifikandens 24 bitar motsvarar ca 7 decimalers noggrannhet.

I IEEE standarden anges bara en undre gräns för antalet extra bitar som behövs för utvidgad precision. I standarden rekommenderas att formatet för utvidgad precision skall ha så stor vidd som möjligt. Idén att använda utvidgad precision har man lånat från fickräknarna, som ofta visar endast 10 siffror, men utför beräkningar med 13 siffrors noggrannhet. För att funktionsberäkningar skall kunnas utföras med 10 siffrors noggrannhet i svaret, behövs några extra siffrors precision i mellanstegen. Enligt IEEE standarden kan räkneoperationerna utföras i utvidgad precision, och svaret anges sedan med korrekt antal siffror i enkel eller dubbel precision.

Dubbelprecisionsformatet i IEEE-standarden finns implementerat i MATLAB. Detta betyder att 11 bitar används för den binära exponenten, som alltså varierar inom intervallet $[-2^{10}, +2^{10}]$. Signifikanden har 53 bitar, så att ett normaliserat flyttal kan uttryckas i formen $a = r \cdot 2^e$, där $r \in [1, 2)$ är

$$r = 1 + \sum_{k=1}^{52} a_k 2^{-k}.$$

Eftersom $2^{10} = 1024 \approx 10^3$ så kommer 53 signifikanta bitar att ge upphov till ca 16 decimalers räknenoggrannhet. Maskinprecisionen ges av MATLAB-variabeln `eps`$= 2^{-52} \approx 2.2204 \cdot 10^{-16}$. Det minsta positiva talet anges av `realmin`$\approx 2.2251 \cdot 10^{-308}$, och det största positiva talet av `realmax`$\approx 1.7977 \cdot 10^{+308}$. Resultatet av en räkneoperation, som ger ett meningslöst resultat (såsom $0/0$, ∞/∞ etc.) anges med `NaN` (= "Not a Number").

2.2. Avrundningsfel vid flyttalsaritmetik

Även om operanderna i en aritmetisk operation har en exakt flyttalsrepresentation, behöver inte resultatet av räkneoperationen vara ett flyttal, eftersom produkten av två tal med n siffror antingen har $2n$ eller $2n-1$ siffror. Det är därför viktigt att det finns något sätt att mäta avrundningsfelet vid en räkneoperation.

v. Neumann och Goldstine uttryckte detta år 1947 som följer: En maskin som framställer två tal x och y som två fysiska storheter $\tilde{x}$ och $\tilde{y}$ bildar summan $x + y$ eller produkten xy som två fysiska storheter $\tilde{x} \oplus \tilde{y}$ och $\tilde{x} \otimes \tilde{y}$. De motsvarar inte de sanna värdena $x + y$ och xy, utan $x + y + \epsilon^{(s)}$ och $xy + \epsilon^{(p)}$, där $\epsilon^{(s)}$ och $\epsilon^{(p)}$ är slumpmässiga "brus"-variabler.

Låt oss för enkelhetens skull tänka oss ett flyttalsformat, där $b = 10$ och $p = 3$. Om resultatet av en flyttalsoperation t.ex. blir $6.22 \cdot 10^{-2}$, och vi vet, att det exakta svaret är 0.0624, så är det klart, att felet är 2 enheter i den sista decimalen. Likaledes, om man uttrycker det reella talet 0.059763 som $5.97 \cdot 10^{-2}$, så är felet 0.63 enheter i den sista decimalen. I allmänhet, om flyttalet $d.d \ldots d \cdot b^e$ får representera det reella talet a, så är felet $|d.d \ldots d - (a/b^e)|b^{p-1}$ enheter i den sista decimalen. För "enheter i den sista decimalen" brukar användas förkortningen ulp (från engelskan: "units in the last place"). Om resultatet av en beräkning är det flyttal, som är närmast det korrekta resultatet, så kan felet likväl vara så stort som $0.5ulp$.

Ett annat sätt att ange skillnaden mellan flyttalet och det reella talet är använda det relativa felet, som är de två talens skillnad dividerad med det reella talet. Om man t.ex. approximerar 5.9713 med $5.97 \cdot 10^0$, så är det relativa felet $0.0013/5.9713 = 0.0002$.

Vilket är det relativa fel, som motsvarar $0.5\ ulp$? Som vi lätt inser, kan felet vara så stort som $0.00\ldots00b' \cdot b^e$, då ett reellt tal approximeras med det närmaste flyttalet $d.dd\ldots dd \cdot b^e$ (här har vi antagit att b' betecknar $b/2$, och att det finns p enheter i flyttalets signifikand, och lika många nollor i felets signifikand). Detta fel kan uttryckas $(b/2)b^{-p} \cdot b^e$. Eftersom alla tal av formen $d.dd\ldots dd \cdot b^e$ har samma absoluta fel, men värden mellan b^e och $b \cdot b^e$, så kommer det relativa felet att befinna sig mellan det större talet $(b/2)b^{-p} \cdot b^e/b^e$ och det mindre $(b/2)b^{-p} \cdot b^e/b^{e+1}$, så att

$$\tfrac{1}{2}b^{-p} \leq \tfrac{1}{2}ulp \leq \frac{b}{2}b^{-p}.$$

Det relativa fel, som motsvarar $0.5\ ulp$, kan alltså variera med en faktor b. Denna faktor kallas "vinglingen" (eng. "wobble"). Om man sätter $\epsilon = (b/2)b^{-p}$ lika med den största av gränserna i ovanstående olikhet, får man en övre gräns för det relativa felet (maskinprecisionen). I exemplet ovan var det relativa felet $0.0013/5.9713 = 0.0002$. För att undvika små tal, brukar man vanligen uttrycka det relativa felet som en konstant gånger ϵ_m, som i detta fall är $\epsilon_m = (b/2)b^{-p} = 5 \cdot 10^{-3} = 0.005$. Det relativa felet kan då uttryckas som $((0.0013/5.9713)/0.005)\epsilon_m \approx 0.04\epsilon_m$.

Vanligen är resultatet av en räkneoperation lika med det avrundade eller avkortade värdet av räkneoperationens exakta resultat. Om $x \circ y$ är det exakta värdet av en räkneoperation, och $fl(x \circ y)$ betecknar det approximativa värde, som uträknats av maskinen, så kan *felet* vid en dylik räkneoperation uttryckas i formen

$$|fl(x \circ y) - x \circ y| \leq |x \circ y| \cdot u,$$

där u är ett mått på datorns avrundningsfel.

Ett annat sätt att uttrycka maskinprecisionen ϵ_{m} är att definera den som det minsta flyttal, som man kan addera till 1.0, och få till resultat ett flyttal större än 1.0. Om x och y är två positiva flyttal ($x > y$), så är $x + y = x\left(1 + \frac{y}{x}\right)$. Vi inser att högra membrum av denna ekvation kommer att vara x, om inte $\frac{y}{x} > \epsilon_{\mathrm{m}}$. Med 32 bitars ordlängd är $\epsilon_{\mathrm{m}} = 2^{-22} \approx 10^{-7}$.

Detta förklarar förlusten av precision vid subtraktion av tal, som endast obetydligt skiljer sig från varandra (katastrofal kancellering). Som exempel kan vi ta uttrycket $a - \sqrt{a^2 - \delta^2}$, där $\delta \ll a$. Genom serieutveckling finner vi

$$a - \sqrt{a^2 - \delta^2} = a - a\left[1 - \tfrac{1}{2}\left(\frac{\delta}{a}\right)^2 + \ldots\right] = \frac{a}{2}\left(\frac{\delta}{a}\right)^2 + \ldots,$$

som skiljer sig från noll endast om $\delta > a\sqrt{\epsilon_{\mathrm{m}}}$. Om t.ex. $a = 1000$ och $\epsilon_{\mathrm{m}} = 10^{-7}$, så finner man, att $\delta \geq 0.32$ bör gälla för att detta villkor skall vara uppfyllt.

Denna förlust av precision kan dock undvikas, om man omskriver uttrycket :

$$a - \sqrt{a^2 - \delta^2} = \frac{\delta^2}{a + \sqrt{a^2 - \delta^2}}.$$

Ett annat exempel är ytan av en triangel, som kan beräknas då man känner längden av de tre sidorna a, b och c:

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = (a+b+c)/2$$

Detta är Herons formel, vars bevis gavs av *Heron av Alexandria* år 60 e.kr., men möjligen redan var känd av Arkimedes. Om triangeln är mycket trubbig, dvs $a \approx b+c$, så är $s \approx a$, och faktorn $(s-a)$ är skillnaden mellan två nästan lika stora tal, som kan ha avrundningsfel. Om t.ex. $a = 10$ och $b = c = 5.02$, så är det exakta värdet av $s = 10.02$, och $A = 2.238\ldots$. Om s beräknats till 10.04, dvs ett fel om endast $2\ ulp$, så blir $A \approx 3.18$, ett fel på $94\ ulp$!

Man kan omskriva Herons formel, så att den fungerar också för trubbvinkliga trianglar:

$$A = \frac{\sqrt{(a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c))}}{4}, \quad a \geq b \geq c.$$

Om a, b och c inte uppfyller villkoret $a \geq b \geq c$, så byter man ut beteckningarna innan formeln tillämpas. Som man lätt visar, är de båda formlerna identiska.

Ett exempel på ett annat uttryck som kan omskrivas så att kancellering undviks, är $(1+x)^n$, där $x \ll 1$. Detta uttryck förekommer t.ex. då man beräknar ränta på ränta. Om man t.ex. dagligen skulle sätta in 100 euro på en (hypotetisk) bank där årsräntan 6 % dagligen tilläggs till kapitalet, så skulle det på ett år växa till $100 \cdot \frac{(1+r/n)^n - 1}{r/n}$, där $n = 365$ och $r = 0.06$. Om detta uttryck beräknas med $b = 2$ och $p = 24$ fås 37615.45 euro, istället för 37614.05 euro, som är det rätta svaret. Felet är 1.40 euro, som beror på additionen $1 + r/n = 1 + 0.0001643836$, vilket leder till ett avrundningsfel. Detta fel förstoras vid potenseringen.

Emellertid kan uttrycket $(1 + r/n)^n$ omskrivas som $e^{n \ln(1+r/n)}$, varigenom man istället måste beräkna uttrycket $\ln(1 + x)$ för ett litet värde av x. Ett sätt att göra det är att använda approximationen $\ln(1 + x) \approx x$, men det leder till ett sämre resultat. Ett mycket bättre sätt är att använda formeln:

$$\ln(1+x) = \begin{cases} x & \text{om } 1 \oplus x = 1 \\ \frac{x \ln(1+x)}{(1+x)-1} & \text{om } 1 \oplus x \neq 1, \end{cases}$$

som blir intressant då $x \ll 1$. Här anger $\oplus$ flyttalsaddition.

Det är inte alltid möjligt att förändra räknemetoden för att förbättra räknenoggrannheten. Svårigheten kan bero på själva problemets natur. Ett exempel på detta är fjärdegradsekvationen

$$x^4 - 4x^3 + 8x^2 - 16x + 15.99999999 = (x-2)^4 - 10^{-8} = 0,$$

som har de exakta lösningarna $x_1 = 2.01, \quad x_2 = 1.99, \quad x_3 = 2 + 0.01i, \quad x_4 = 2 - 0.01i.$ Om man använder en dator med $\epsilon_m > 10^{-10}$ så blir fjärdegradsekvationens konstanta term automatiskt avrundad till 16.0, och ekvationen övergår i

$$(x-2)^4 = 0,$$

som har fyra sammanfallande rötter $x_k = 2, k = 1, 2, 3, 4$. Felet i rötterna kommer sålunda att uppgå till 0.5% . Eftersom detta resultat fås oberoende av den använda räknemetoden är det själva problemet som har dålig kondition.

2.3. Feluppskattningsmetoder

För uppskatta felet i en approximation $\tilde{x}$ till ett tal x används antingen det *absoluta* eller det *relativa* felet. Det absoluta felet definieras som $|x - \tilde{x}|$, vilket i praktiken innebär att ett visst antal decimaler överensstämmer i talen. Det relativa felet definieras i sin tur som $|x - \tilde{x}|/|x| = |1 - \tilde{x}/x|$, vilket innebär att ett visst antal signifikanta siffror överensstämmer.

I allmänhet lämpar sig det absoluta felet bäst för storheter, vars värde är nära 1, medan det relativa felet kan användas för storheter, vars värde är stort, eller nära 0.

För att uppskatta felet vid approximation av funktioner, används andra mätmetoder. Dessa metriker, eller normer uttrycks vanligen som absoluta fel, men man kan lätt transformera dem till relativa fel. De tre vanligaste normerna som används vid approximation av en funktion f inom ett slutet intervall $[a, b]$ med en annan funktion p (t.ex. ett polynom) är

Likformighetsmetriken eller L_∞:

$$\|f - p\|_\infty = \max_{a \le x \le b} |f(x) - p(x)|,$$

L_1–*normen*:

$$\|f - p\|_1 = \int_a^b |f(x) - p(x)| dx,$$

L_2 eller *minsta-kvadrat-normen*:

$$\|f - p\|_2 = \sqrt{\int_a^b |f(x) - p(x)|^2 dx}.$$

Som ett exempel skall vi tillämpa dessa normer på approximation av sinuskurvan med en rät linje inom intervallet $[0, \pi/4]$, dvs $f(x) = \sin x$ och $p(x) = x$. Normerna får då följande värden:

$$\|\sin x - x\|_\infty = \frac{\pi}{4} - \sin\frac{\pi}{4} \approx 7.8291 \cdot 10^{-2}$$

$$\|\sin x - x\|_1 = \int_0^{\pi/4} (x - \sin x) dx = \left[\frac{x^2}{2} - \cos x\right]_0^{\pi/4} = \frac{\pi^2}{32} + \frac{1}{\sqrt{2}} - 1 \approx 1.5532 \cdot 10^{-2}$$

$$\| \sin x - x \|_2 = \sqrt{\int_0^{\pi/4} (x - \sin x)^2 dx} = \sqrt{\left[\frac{x^3}{3} - 2 \sin x + 2x \cos x + \frac{x}{2} - \frac{\sin 2x}{4} \right]_0^{\pi/4}}$$

$$= \sqrt{\frac{\pi^3}{192} + \frac{2\sqrt{2}+1}{8}(\pi - 4) + \frac{1}{4}} \approx 2.6406 \cdot 10^{-2}.$$

Om vi endast känner funktionsvärdena i ett antal punkter $x_0, x_1, \ldots, x_n$, så övergår integralerna givetvis i summor:

$$\| f - p \|_\infty = \max_{k=0,1,\ldots,n} |f(x_k) - p(x_k)|$$

$$\| f - p \|_1 = \sum_{k=0}^{n} |f(x_k) - p(x_k)|$$

$$\| f - p \|_2 = \sqrt{\sum_{k=0}^{n} |f(x_k) - p(x_k)|^2}$$

Den sistnämnda normen används då man gör en anpassning till experimentella data med hjälp av minsta kvadratmetoden.

Om funktionen är känd endast i diskreta punkter kan man använda polynominterpolation för att bestämma ett polynom som överensstämmer med funktionen i de givna punkterna:

$$f(x_k) = p(x_k) \text{ då } x_k = x_0, x_1, \ldots, x_n.$$

Men detta är inte alltid en god idé, eftersom funktionsvärdena kan ha mätfel. Vi skall återkommer till dessa problem senare.