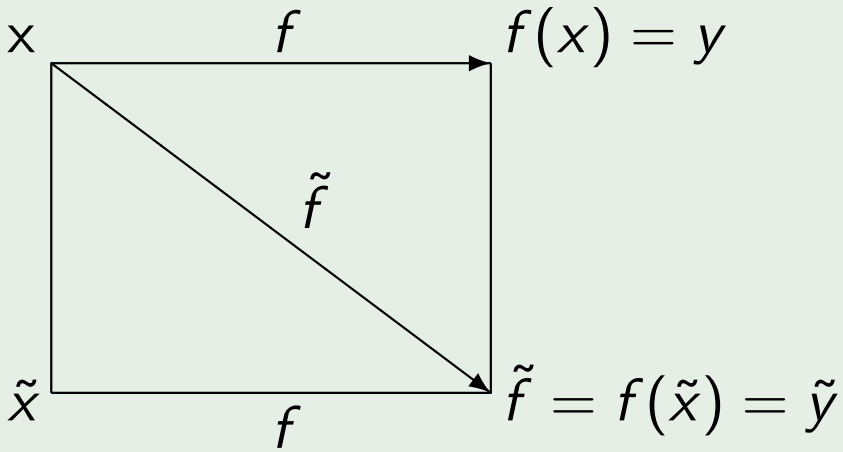
Numerisk Analys, MMG410. Lecture 2.

Example



Framåtfelet: $|y - \tilde{y}|$; Bakåtfelet: $|x - \tilde{x}|$;
$f(x) = \sqrt{x} = y$; $f(\tilde{x}) = \sqrt{2} \approx 1.4 = \tilde{y}$
Låt $y = 1.41421\ldots$
Framåtfelet: $|y - \tilde{y}| = |1.4 - 1.41421| \approx 0.014 \approx 1\%$
Bakåtfelet: $(1.4)^2 = 1.96 = \tilde{x}$,
$\sqrt{1.96} = 1.4$ och $|x - \tilde{x}| = |1.96 - 2| = 0.04 \approx 4\%$

Diskretiseringsfel och avrundningsfel

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

- För stora h dominerar diskretiseringsfelet, man kan bortse från avrundningsfelet.
- För små h dominerar avrundningsfelet.
  Se approximation av $f'(x)$: kancellation i täljaren för små h och division med litet tal förstärker felet i täljaren.

Diskretiseringen för $f'(x)$: första ordningen noggrannhet

Approximativt värde (Taylor's theorem):

$$f(x+h) = f(x) + f'(x)h + \frac{f''(Q)h^2}{2!},$$

$Q \in [x, x+h]$

$$f(x+h) - f(x) = f'(x)h + \frac{f''(Q)h^2}{2!}.$$

Dividera med h:

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f''(Q)h}{2!}$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(Q)h}{2!}$$

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Trunkeringsfel för $f'(x)$

Trunkeringsfel:

$$\frac{f''(Q)h}{2} = \frac{f(x+h) - f(x)}{h} - f'(x).$$

Låt $M \leq |f''(Q)|$, då trunkeringsfel ε är bounded med

$$\varepsilon < \frac{Mh}{2}.$$

Vi fick för $f'(x)$ första ordning noggrannhet.

Diskretiseringen för $f'(x)$: andra ordning noggrannhet.

Approximativt värde (Taylor's theorem):

$$(*)\ f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(Q)h^3}{3!}$$

$$(**)\ f(x-h) = f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(Q)h^3}{3!}$$

$(*) - (**)$:

$$f(x+h) - f(x-h) = 2f'(x)h + 2\frac{f'''(Q)}{3!}h^3$$

$$2f'(x)h = f(x+h) - f(x-h) - 2\frac{f'''(Q)}{3!}h^3$$

Eller

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{2f'''(Q)h^3}{3! \cdot 2h}$$

Trunkeringsfel för $f'(x)$: andra ordning noggrannhet.

Trunkeringsfel:

$$\frac{2f'''(Q)h^3}{3! \cdot 2h} = \frac{f(x+h) - f(x-h)}{2h} - f'(x).$$

Låt $M \leq |f'''(Q)|$, då trunkeringsfel ε är begränsad med

$$\varepsilon < \frac{Mh^2}{6}.$$

Vi fick för $f'(x)$ andra ordning noggrannhet.

Diskretiseringen för $f''(x)$: andra ordning noggrannhet.

Approximativt värde (Taylor's theorem):

$$(*)\ f(x-h) = f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f'''(x)h^3}{3!} + \dots$$

$$(**)\ f(x+h) = f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \dots$$

$(*) + (**)$:

$$f(x+h) + f(x-h) = 2f(x) + \frac{2f''(x)}{2!}h^2 + O(h^4)$$

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x) - O(h^4)}{h^2}$$

$O(h^4) = \frac{2f^{(4)}(x)}{24}h^4$; $\frac{f^{(4)}(x)}{12}\frac{h^4}{h^2} = \frac{f^{(4)}(x)}{12}h^2 \longrightarrow$ Låt $M \leq |f^{(4)}(Q)|$, då trunkeringsfel ε är begränsad med $\varepsilon < \frac{Mh^2}{12}$, och $f''(x)$ har andra ordningen noggrannhet.

Representation av tal i dator

- Alla tal lagras i ett begränsat antal bitar i minnet, vanligen i binär form.
- För heltal: lagring i dator är utan problem. Heltal upp till en viss storlek lagras exakt.
- Reella tal lagras exakt utan måste avrundas. Representation av reella tal kallas flyttalsrepresentation och talen kallas flyttal.

IEEE flyttalsrepresentation

IEEE 754 (Institute of Electrical and Electronics Engineers, Inc.) definierar enkel och dubbel precision (bland annat).

- Under 60- och 70-talen hade varje datortillverkare sitt eget flyttalsystem.
- En flyttalstandard utvecklades under tidigt 80-tal och följdes av tillverkare som Intel och Motorola.
- IEEE standarden har 3 viktiga krav: konsistent flyttalsrepresentation, korrekt avrundningsaritmetik, konsistent hantering av exceptionella situationer.

Flyttalsaritmetik

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

var

$$0 \leq d_k \leq \beta - 1, L \leq e \leq U,$$

Här:

- β bas (s vi kommer att ha $\beta = 2$)
- e exponent (heltal)
- t precision
- $[L, U]$ exponentomfång
- $d_k, k = 0, ..., t - 1$ mantissa (heltal)

Vi antar att $= 2$ från och med nu:

bas	t	L	U	
2	24	-126	127	32 bitar
2	53	-1022	1023	64 bitar

Flyttalsaritmetik

- IEEE enkel precision: tecknet ("+" 0, "-" 1) 1 bit, exponent 8 bitar, mantissa 23 bitar = 32 bitar:

$$\pm e_1 e_2 \ldots e_8 d_0 d_1 \ldots d_{22}$$

- IEEE dubbel precision: tecknet ("+" 0, "-" 1) 1 bit, exponent 11 bitar, mantissa 52 bitar = 64 bitar:

$$\pm e_1 e_2 \ldots e_{11} d_0 d_1 \ldots d_{51}$$

Ett tal $\neq 0$ är normaliserat om $d_0 \neq 0$.
Om $\beta = 2$ så är $d_0 = 1$ varför man inte lagrar d_0.

Det finns speciella bitformat för diverse specialfall, t.ex:
[0, -0]
ans = 0000000000000000 8000000000000000
Antalet olika tal räknas som:

$$2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

och är

- $4.2614 \cdot 10^9$ i enkel precision
- $1.8429 \cdot 10^{19}$ i dubbel precision.
- Att testa en funktion för alla flyttal i dubbel precision är nästan omöjligt. 109 tester per sekund ger 584.4 år.
- Minsta positiva representerbara normaliserade talet är $2^L \approx 1.1710^{-38}$ i enkel och $\approx 2.2 \cdot 10^{-308}$ i dubbel precision.
- Det största representerbara talet har största exponenten och ettor i hela mantissan. I enkel precision $\approx 3.4 \cdot 10^{38}$, i dubbel $\approx 1.8 \cdot 10^{308}$.
- Tal större än största representerbara talet ger overflow och mindre än minsta positiva representerbara normaliserade talet ger underflow.

Example

När vi skriver i Matlab:

- $1e-200^2 = 10^{-200^2}$ ger underflow, svar 0.
- $1e200^2$ ger overflow, svar infinity.
- *log*(0), svar -infinity
- sin(1/0), svar NaN (not a number)
- exp(log(10000)) − log(exp(10000)), *svar* :exp(log(10000)) = 10000, exp(10000) = Inf och log(exp(10000)) = Inf och exp(log(10000)) − log(exp(10000)) = −Inf
- exp(log(10)) − log(exp(10)), svar: exp(log(10)) = 10, log(exp(10)) = 10 och exp(log(10000)) − log(exp(10000)) = 0.

Skriv talet i binär form som flyttal i dator.

Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

Example

$$-3.25: \ -[1.625] \cdot 2^1 = -\underbrace{[1 + 0.625]}_{\textit{mantissa}} \cdot 2^1$$

Exponenten $e = 1$ lagras som: $1 + 1023 = 1024 = 2^{10}$. Mantissa: 1 kodas inte,

$$0.625 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \ldots = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \ldots$$

$\frac{1}{2} = 0.5 < 0.625; \quad \frac{1}{4} = 0.25 : 0.625 - 0.5 = 0.125; 0.125 > 0.125$? Vi får följande binär representation för -3.25 :

1	10000000000	1010 0
tecken	exponent 11 bitar	mantissa 52 bitar

Example

-3.25 i binär form lagras som:

1	10000000000	1010 0
tecken	exponent 11 bitar	mantissa 52 bitar

-3.25 i hexadecimalt (bas 16) form lagras som:

c00a000000000000

Bas 16:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										a	b	c	d	e	f

Nu grupperar vi om binär form för -3.25 i 4 bitar:

1100	0000	0000	1010	0000	....	0000

och kodar första fyra bitar: $\boxed{1100} = c$

$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 12 = c$

0000 koderas som 0, och sedan

$\boxed{1010} = a$

$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 10 = a$

Example

$-9.28 := -[1.16] \cdot 2^3 = -[1 + 0.16] \cdot 2^3$

$$0.16 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \frac{1}{32}x_5 + \ldots =$$

$$= \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 0 + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 0 + \ldots$$

Exponenten 3 lagras som: $3 + 1023 = 1026 = 1024 + 2 = 1 \cdot 2^{10} + 1 \cdot 2^1 + 0 \cdot 2^0$

1	10000000010	0010 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Example

$6.28 = +[1.57] \cdot 2^2 = +[1 + 0.57] \cdot 2^2$

$$0.57 = \frac{1}{2} + 0.07 =$$

$$= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} + 0 \cdot \frac{1}{32} + \ldots + \frac{1}{256} + \ldots$$

Exponenten 2 lagras som: $2 + 1023 = 1025 = 1024 + 1 = 1 \cdot 2^{10} + 1 \cdot 2^0$

0	10000000001	10010 ....10...
tecken	exponent 11 bitar	mantissa 52 bitar

Flyttal-räkning

Om x är ett godtyckligt reellt tal betecknar vi det avrundade flyttalet med $fl(x)$ (floating). Normalt (kan ändras) är fl(x) det flyttal som ligger närmast x.

Example

Exempel: Låt oss anta att vi räknar decimalt med fyra siffror.

$$\begin{aligned} fl(\pi) &= fl(3.141592653589...) = 3.142, \\ fl(31415926.53589...) &= 3.142 \cdot 10^7. \end{aligned} \tag{1}$$

Hur stort kan det absoluta felet bli vid avrundning till närmaste flyttal? Maximalt en halv enhet i fjärde siffran. Så om vårt tal är

$$\pm s_1 . s_2 s_3 s_4 ... 10^e$$

(där $s_1, s_2, ...$ betecknar decimala siffror) är absolutbeloppet av absoluta felet maximalt $0.0005 \cdot 10^e$. Relativa felet är maximalt (för ett normaliserat tal)

$$\left| \frac{fl(x) - x}{x} \right| \leq \frac{0.0005 \cdot 10^e}{1.0000.... \cdot 10^e} = 0.0005$$

Denna begränsning kallas relativa maskinnoggrannheten

Maskinnoggrannheten ε_{mach}

ε_{mach} beror på method, som vi använder i avrundning. Definitioner för ε_{mach} för precision t:

- I rounding by choping:

$$\varepsilon_{mach} = \beta^{1-p} = \beta^{-t}$$

- I rounding to nearest:

$$\varepsilon_{mach} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}\beta^{-t}$$

I dubbel precision (när $t = 53$, 64 bits dator) gäller att $\varepsilon_{mach} = 2^{-t} \approx 1.11 \cdot 10^{-16}$ och i enkel precision (när $t = 24$, 32 bits dator) $\varepsilon_{mach} = 2^{-t} \approx 6 \cdot 10^{-8}$.

Example

	chop rounding to 2 digits	to nearest to 2 digits
1.849	1.8	1.8
1.850	1.8	1.9
1.851	1.8	1.9
1.899	1.8	1.9