Zapis zmiennopozycyjny, arytmetyka, błędy numeryczne

Plan wykładu:

1. zapis zmiennopozycyjny
2. arytmetyka zmiennopozycyjna
3. reprezentacja liczb w standardzie IEEE754
4. błędy w obliczeniach numerycznych
5. definicje

Własności zapisu zmiennopozycyjnego

Liczbę rzeczywistą w komputerze reprezentuje liczba zmiennoprzecinkowa:

$$F = M\beta^E$$

gdzie:
M – znacznik („mantysa”) jest liczbą ułamkową ze znakiem
β – stanowi bazę reprezentacji i jest liczbą całkowitą (np.: 2, 10, 16)
E – wykładnik („cecha”) jest znakowaną liczbą całkowitą
Powyższy zapis jest niejednoznaczny:

$$F = M\beta^E = M_i\beta^{E+i} = (M\beta^{-i})\beta^{E+i}$$

$$i = -m, \ldots, -1, 0, 1, \ldots, m$$

m jest liczbą pozycji znacznika w bazie.
Można odróżnić

$$k = mlog_\beta 2$$

różnych reprezentacji tej samej liczby.
Ograniczeniem znacznika jest

$$|M| < \beta$$

ale problem nadal pozostaje.

Jednoznaczność osiągamy dla warunku

$$\beta^{p-1} \leq |M| < \beta^p$$

wtedy dla dowolnego i≠0 mamy

$$\beta^{p-i-1} \leq |M|\beta^{-i} < \beta^p$$

$$|M|\beta^{-i} \notin [\beta^{p-1}, \beta^p)$$

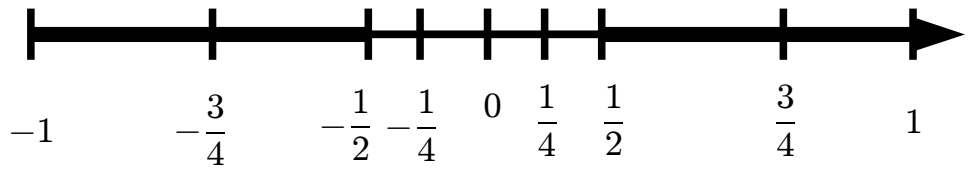
w praktyce stosuje się

$$p = 0, 1$$

Znaczniki spełniające powyższy warunek nazywamy **znormalizowanymi (liczby znormalizowane)**.
Nie ma liczb znormalizowanych

$$|F| < \beta^{p-1-E_{min}}$$

w tym 0. Tworzą one osobną grupę zwana **liczbami zdenormalizowanymi**.



Znormalizowane wartości dla $\beta = 2, \quad p = 0$

Podstawowe operacje arytmetyczne. Wykonujemy operacje na dwóch znormalizowanych liczbach

$$F_1 = M_1\beta^{E_1}, \quad F_2 = M_2\beta^{E_2}$$

Czy przeprowadzenie 4 podstawowych operacji da znormalizowany wynik?

Mnożenie

$$\begin{aligned} F_1F_2 &= M_1\beta^{E_1}M_2\beta^{E_2} \\ &= (M_1M_2)\beta^{E_1+E_2} = M_W\beta^W \end{aligned}$$

sprawdzamy czy nie wystąpił **nadmiar** ($E_W>E_{max}$) lub **niedomiar** ($E_W<E_{min}$) zmiennopozycyjny. Jeśli wartość znacznika wychodzi poza dozwolony zakres tj.:

$$\beta^{2p-2} \leq |M_W| = |M_1M_2| < \beta^{p-1}$$

to wykonujemy **postnormalizację**

$$\begin{aligned} F_1F_2 &= (M_1M_2\beta^{-p+\varepsilon})\beta^{E_1+E_2+p-\varepsilon} \\ \varepsilon &= 0,1 \end{aligned}$$

Dzielenie

$$\begin{aligned} F_1/F_2 &= (M_1\beta^{E_1})/(M_2\beta^{E_2}) \\ &= (M_1/M_2)\beta^{E_1-E_2} \end{aligned}$$

ponieważ

$$|M_W| = M_1/M_2 \in (\beta^{-1}, \beta)$$

konieczna jest postnormalizacja ilorazu do postaci

$$\begin{aligned} F_1/F_2 &= (\beta^{p-\varepsilon}M_1/M_2)\beta^{E_1-E_2-p+\varepsilon} \\ \beta^{-1} &\leq |M_W| < 1 \Longrightarrow \varepsilon = 0 \\ 1 &\leq |M_W| < \beta \Longrightarrow \varepsilon = 1 \end{aligned}$$

Dodatkowo należy zapewnić obsługę liczb zdenormalizowanych, dzielenia przez 0 oraz próby dzielenia 0/0.

Dodawanie i odejmowanie

Wymagane jest wstępne wyrównanie wykładników. Powoduje to denormalizację operandu z mniejszym wykładnikiem i utratę dokładności (na najmniej znaczących pozycjach znacznika).

Załóżmy

$$E_1 \geq E_2$$

$$\begin{aligned} F_1 \pm F_2 &= (M_1\beta^{E_1}) \pm (M_2\beta^{E_2}) \\ &= (M_1 \pm M_2\beta^{-(E_1-E_2)})\beta^{E_1} \end{aligned}$$

a) jeśli

$$E_1 = E_{max}$$

oraz

$$\beta^p \leq |M_W| < 2\beta^p$$

to wówczas normalizacja doprowadzi do nadmiaru.

b) jeśli

$$|M_W| < \beta^i \leq \beta^{p-1}$$

oraz $E_1 - (p-i) \leq E_{min}$

to wystąpi niedomiar.

Jeśli spełniony jest warunek

$$p - i \geq |m log_\beta 2|$$

wówczas może dojść do wyzerowania wszystkich znaczących pozycji znacznika sumy lub różnicy.

Wybór reprezentacji

Liczbę zmiennopozycyjną zapisujemy na n pozycjach, z czego:

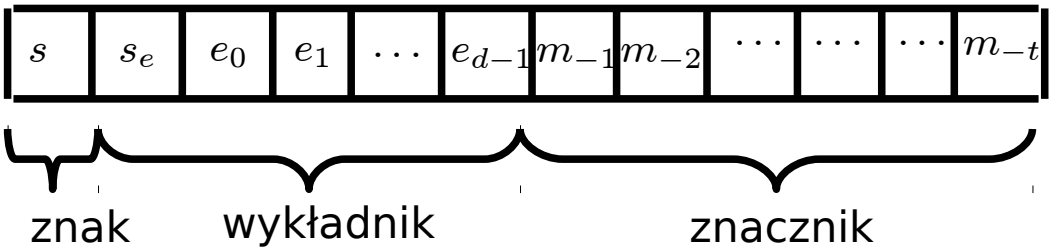
a) 1 pozycję przeznaczamy na znak liczby

b) t pozycji na zapis znacznika

c) d pozycji na zapis wykładnika

$$n = 1 + d + t$$

$$x = s\left(\sum_{i=1}^{t} m_{-i}\beta^{-i}\right)\beta^E$$



Dokładność reprezentacji

Jeśli liczba zmiennopozycyjna jest reprezentowana przez skończoną liczbę bitów to dokładność reprezentacji określa liczba bitów znacznika a zakres reprezentowanych liczb zależy od liczby bitów wykładnika.

Jeśli zachodzi warunek

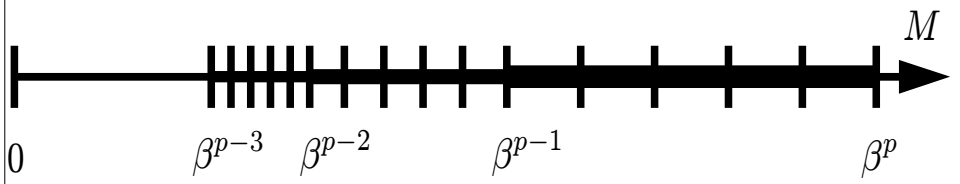
$$M\beta^E \leq x \leq (M + ulp)\beta^E, \quad x \in R$$

(ulp – najmniej znacząca pozycja znacznika) wówczas zaokrąglając liczbę x do bliższej reprezentacji dostajemy **błąd bezwzględny**

$$|fl(x) - x| \leq \frac{1}{2} ulp \beta^E$$

gdzie fl(x) oznacza reprezentację liczby x w zapisie zmiennopozycyjnym.

Wartość błędu bezwzględnego wynika z nierównomiernego rozłożenia liczb w reprezentacji zmiennopozycyjnej (rysunek).



Błąd względny

$$|\varepsilon(x)| = \frac{|fl(x) - x|}{x} \leq \frac{\frac{1}{2} ulp \beta^E}{M\beta^E} = \frac{ulp}{2M}$$

rośnie ze zmniejszaniem znacznika aż do wartości

$$MRRE = \max_M |\varepsilon(x)| = \frac{1}{2} ulp \beta^{p-1}$$

MRRE – maximum relative representation error.

Jeśli znacznik zakodowano na m pozycjach

$$\begin{aligned} MRRE &= 2^{-(m-plg\beta)-1} \beta^{1-p} \\ &= 2^{-m-1} \beta^p \beta^{1-p} = 2^{-m-1} \beta \end{aligned}$$

to nie zależy on od przedziału normalizacji znacznika.

Dobór zakresu wykładnika

Jednym z wymagań jest aby dla każdej znormalizowanej liczby x możliwe było obliczenie jej odwrotności. Wykładnik kodowany jest na d pozycjach z czego λ pozycji musimy zarezerwować na 0 oraz wielkości nieznormalizowane (nieskończoności itp.).
Rozpiętość wykładnika wynosi

$$s = E_{max} - E_{min} = 2^d - \lambda$$

i aby istniały znormalizowane reprezentacje odwrotności małych liczb muszą być spełnione warunki

$$\left(\beta^{E_{min}}\beta^{p-1}\right)^{-1} \quad < \quad \beta^{E_{max}}\beta^{p}$$

(oraz dla E_{min})

$$\left(\beta^{E_{min}-1}\beta^{p-1}\right)^{-1} \quad \geq \quad \beta^{E_{max}-1}\beta^{p}$$

co daje ograniczenie na E_{min}

$$-p - \frac{1}{2}(s-1) < E_{min} \leq -p - \frac{1}{2}(s-1) + 1$$

I pozwala wyliczyć minimalną i maksymalną wartość wykładnika

$$E_{min} = -\left[\frac{1}{2}(s-1)\right] - (p-1)$$

$$E_{max} = \left[\frac{1}{2}(s+1)\right] - (p-1)$$

Uwaga

Najmniejszy znormalizowany znacznik ma postać

{1,0,...,0,0}

a największy

{β-1,β-1,...,β-1,β-1}

zatem wartość znacznika jest ograniczona

$$\beta^{p-1} = 1\beta^{p-1} + \sum_{i=-m}^{p-1} 0 \cdot \beta^i \leq M$$

$$M \leq \sum_{i=-m}^{p-1} (\beta - 1)\beta^i < \beta^p$$

dla β=2, β-1=1 i p=0
najmniejszy znacznik ma wartość
0,100000...000
a największy
0,111111...111
najbardziej znaczący bit nie musi być kodowany (**bit ukryty**). Pominięcie go prowadzi do pełnego wykorzystania przestrzeni kodowej znacznika (100%). W innych przypadkach liczby znormalizowane zajmują jedynie (1-1/β)100% przestrzeni. Bit ukryty jest odtwarzany w trakcie operacji arytmetycznych.

Schematy zaokrąglania liczb

Podczas wykonywania operacji arytmetycznych może dojść do zwiększenia się liczby bitów wynikowych, np. przy mnożeniu dwóch znaczników

$$m_w = 2m$$

Aby zapisać wynik należy uciąć ostanie m bitów. Eliminacja nadmiarowych bitów nazywana jest **zaokrąglaniem**.

Reguły zaokrąglania

$$x \leq y \Rightarrow fl(x) \leq fl(y)$$

$$x \in fl \Rightarrow fl(x) = x$$

$$\underbrace{F_1 = M\beta^E \leq x \leq (M + ulp)\beta^E = F_2}_{x=F_1 \vee x=F_2}$$

odcięcie (najprostsze)

$$M \leq x < M + ulp \Leftrightarrow T(x) = M$$

Jeśli
m - liczba pozycji znacznika
d - liczba uciętych bitów

$$x = M + i2^{-d}ulp, \quad 0 \leq i \leq 2^d - 1$$

i standaryzowanym błędem losowym obcinania znacznika jest

$$\frac{T(x) - x}{ulp} = -i2^{-d}$$

Dla równomiernego rozkładu wartości x w przedziale [M, M+ulp] miarą średniego standaryzowanego błędu obcinania jest

$$\delta_T = 2^{-d} \sum_{i=0}^{2^d-1} (-i)2^{-d} = -(2^{d-1} - \frac{1}{2})2^{-d}$$

Błąd ten jest zawsze ujemny – estymator T(x) jest ujemnie obciążony.
Skutek obcinania: **niedoszacowanie**.

Zaokrąglanie do najbliższej wartości
(lub po prostu - **zaokrąglanie**)

Reguły zaokrąglania

$$R(x) = \begin{cases} M, & x - M < \frac{ulp}{2} \\ M + ulp, & x - M \geq \frac{ulp}{2} \end{cases}$$

błąd standaryzowany

$$\frac{R(x) - x}{ulp} = \begin{cases} -i2^{-d}, & 0 \leq i < 2^{d-1}, \quad (0 \leq x - M < \frac{1}{2}ulp) \\ 1 - i2^{-d}, & 2^{d-1} \leq i < 2^{d} \quad (\frac{1}{2}ulp \leq x - M < ulp) \end{cases}$$

i średnia wartość błędu standaryzowanego

$$\delta_R = 2^{-d} \left\{ \sum_{i=0}^{2^{d-1}-1} (-i)2^{-d} + \sum_{i=2^{d-1}}^{2^{d}-1} (1 - i2^{-d}) = \frac{1}{2}2^{-d} \right\}$$

jest bliska 0. Estymator R(x) jest jednak obciążony dodatnio.

zaokrąglanie symetryczne

Reguły zaokrąglania

$$S(x) = \begin{cases} M - ulp, & -ulp \leq x - M < -\frac{1}{2}ulp \\ M, & -\frac{1}{2}ulp \leq x - M \leq +\frac{1}{2}ulp \\ M + ulp, & +\frac{1}{2}ulp \leq x - M < +ulp \end{cases}$$

standaryzowany błąd

$$\frac{S(x) - x}{ulp} = \begin{cases} -i2^{-d} - 1, & -2^d \leq i < -2^{d-1} \\ -i2^{-d}, & -2^{d-1} \leq i \leq 2^{d-1} \\ -i2^{-d} + 1, & 2^{d-1} < i < 2^d \end{cases}$$

i średni błąd standaryzowany

$$\begin{aligned} \delta_s \quad = \quad & 2^{-2d} \left\{ \left(- \sum_{i=-2^d}^{-2^{d-1}-1} (i2^{-d} + 1) - \sum_{i=-2^{d-1}}^{-1} i2^{-d} \right) \right. \\ & \left. + \left(- \sum_{i=0}^{2^{d-1}} i2^{-d} - \sum_{i=2^{d-1}+1}^{2^d-1} (i2^{-d} - 1) \right) \right\} = 0 \end{aligned}$$

Wadą zwykłego zaokrąglania oraz zaokrąglania symetrycznego jest konieczność wykonania 2m-pozycyjnego dodawania.

Reprezentacja liczb w standardzie IEEE754

Formaty liczb zmiennopozyjcyjnych:
1) zwykły pojedynczej precyzji
   - single (real)
2) rozszerzony pojedynczej precyzji
   - single extended
3) zwykły podwójnej precyzji
   - double
4) rozszerzony podwójnej precyzji
   - double extended

Wykładnik jest reprezentowany w kodzie z obciążeniem, a znacznik w kodzie znak-moduł. Wartość liczby w IEEE754

$$F = (-1)^s 2^{E_B - B}(1, f)$$

gdzie: E_B - wykładnik, B – przesunięcie,
($E=E_B-B$ -"prawdziwy" wykładnik)
(1,f) – wartość modułu znacznika
Jeśli d oznacza liczbę bitów wykładnika to wielkością przemieszczenia jest

$$B = 2^d - 1$$

Bez przesunięcia, najmniejszą wartością wykładnika jest

$$E_B = E_{min} + B = 00...001_2$$

a największą

$$E_B = E_{max} + B = 11...110_2$$

Zakres wykładnika jest ograniczony z obawy przed uzyskaniem nadmiaru podczas obliczania odwrotności liczb.

Liczby zdenormalizowane z zakresu $[-2^{Emin}, +2^{Emin}]$ łącznie z zerami można zapisać

$$F = (-1)^s 2^{E_{min}}(0, f)$$

	symbol	single	Single extended	Double	Double extended
Rozmiar formatu	n	32	≥ 43	64	≥ 79
Rozmiar znacznika	m	23(+1)	≥ 32	52(+1)	≥ 64
Rozmiar wykładnika	d	8	≥ 11	11	≥ 15
obciążenie	B	127	≥ 1023	1023	≥ 16383
Zakres wykładnika	E	[-126,127]	[-(B-1),B]	[-1022,1023]	[-(B-1),B]
dokładność	ulp	$2^{-23 \approx 10^{-7}}$	2^{-m+1}	$2^{-52} \approx 10^{-15}$	2^{-m+1}
Zakres formatu	RNG	$\approx 2^{128}$ $\approx 3,8 \cdot 10^{38}$	$\geq 2^{1024}$	$\approx 2^{1024}$ $\approx 9 \cdot 10^{307}$	$\geq 2^{16384}$

Uwaga: dla formatów rozszerzonych istnieje tylko wymaganie co do wartości minimalnych parametrów (n,m,d,B,ulp)

wykładnik	ułamek	kod dwójkowy	wielkość
$E_B = B + E_{min} - 1$	$f = 0$	$s00\ldots00\ 00\ldots00$	± 0
$E_B = B + E_{min} - 1$	$f \neq 0$	$s00\ldots00\ xx\ldots xx$	$(-1)^s 2^{E_{min}} 0, f$
$B + E_{min} \leq E_B \leq B + E_{max}$	-	$syy\ldots yy\ xx\ldots xx$	$(-1)^s 2^{E_B - B} 1, f$
$E_B = B + E_{min} + 1$	$f = 0$	$s11\ldots11\ 00\ldots00$	$\pm\infty$
$E_B = B + E_{min} + 1$	$f \neq 0$	$s11\ldots11\ xx\ldots xx$	NaN
$E_B = B + E_{min} - 1$	$f = 00\ldots01$	$s00\ldots00\ 00\ldots01$	$\pm F_{min,den} = (-1)^s 2^{E_{min} - m}$
$E_B = B + E_{min}$	$f = 00\ldots01$	$s00\ldots01\ 00\ldots00$	$\pm F_{min} = (-1)^s 2^{E_{min}}$
$E_B = B + E_{max}$	$f = 11\ldots10$	$s11\ldots10\ 11\ldots11$	$\pm F_{max} = (-1)^s 2^{E_{max}+1} (1 - 2^{-m-1})$

Nie-liczby

Pojawiają się podczas wykonywania operacji np.:

$$0/0, \quad \sqrt{-|x|}$$

Bez ich obsługi program przerywałby działanie. Obsługa nie-liczb pozwala je wykryć i np. zrestartować schemat iteracyjny z innym parametrem.

Znakowane zero

Po co?
Dla liczb rzeczywistych mamy

$$1/-\infty = 0, \quad 1/(1/-\infty) = +\infty$$

zatem relacja

$$1/(1/x) = x$$

nie jest spełniona. Dla znakowanego 0 mamy

$$x = -\infty, \; 1/x = -0$$
$$1/(1/x) = 1/-0 = -\infty = x$$

Wyjątki w IEEE754

Standard zapewnia obsługę specyficznych wyników operacji:

1. nadmiar (F_{max}, nieskończoność)
2. niedomiar (F_{min}, l. denormalizowane)
3. dzielenie przez 0
4. niepoprawna operacja (NAN)
5. niedokładność (zaokrąglenie wyniku)

Błędy numeryczne

Najprostszy podział:
1. wejściowe
2. zaokrągleń
3. obcięcia

Błędy wejściowe

Występują gdy dane liczbowe wprowadzane do pamięci komputera odbiegają od wartości dokładnych. Kiedy występują?
gdy wprowadzane dane pomiarowe są obarczone błędami pomiarowymi (np. pomiar wielkości fizycznych takich jak oporu czy napięcia)
gdy ze względu na skończoną długość słowa binarnego dochodzi do wstępnego zaokrąglenia liczb (ułamki dziesiętne lub zaokrąglanie liczb niewymiernych jak np.: e, π)

Przykład – zapis 8 bitowy
Liczba

$$x_{(10)} = 3.25$$

ma reprezentację

$$x_{(z2)} = \underbrace{(0)1101}_{M}\underbrace{(0)10}_{W}$$

Ale dla liczby x=0.2 pojawia się problem

$$x_{(2)} = 0.0011(0011)....$$

po zaokrągleniu wyniku do najbliższej liczby

$$\begin{aligned} x'_{(z2)} &= (0)1100(1)10 \\ x'_{(2)} &= 0.001100 \\ x'_{(10)} &= 0.1875 \end{aligned}$$

co daje błąd bezwzględny równy 0.0125
i błąd względny na poziomie 6.25%.

<u>Błędy obcięcia</u> - powstają podczas zmniejszania liczby działań np.:

a) przy obliczaniu wartości szeregów (ucięcie szeregu)
b) wyznaczaniu granic (obliczanie wartości całki)
c) zastępowaniu pochodnej funkcji ilorazem różnicowym

Przykład

Chcemy 1wyznaczyć wartość e^x, więc korzystamy z rozwinięcia

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad -\infty < x < +\infty$$

ale numerycznie lepiej zrobić to tak

$$e^x = e^{E(x)} e^q$$

gdzie: E(x) jest częścią całkowitą liczby x,
q jest częścią ułamkową $0 \leq x < 1$

Pierwszy wyraz jest potęgą a drugi liczymy wg rozwinięcia.
Do wyznaczenia pozostaje tylko błąd obcięcia. Szereg ucinamy na n-tym wyrazie.

Reszta szeregu:

$$R_n(x) = \frac{e^{\theta q}}{(n+1)!} q^{n+1}, \quad 0 < \theta < 1$$

szacujemy maksymalny błąd obcięcia tj.

$$\theta \approx 1, \quad q \approx 1$$

$$0 \leq R_n(q) < \frac{3}{(n+1)!} q^{n+1}$$

widzimy że szereg jest więc szybko zbieżny.
Dokładniejsze oszacowanie

$$R_n(q) =$$

$$\frac{q^{n+1}}{(n+1)!} + \frac{q^{n+2}}{(n+2)!} + \frac{q^{n+3}}{(n+3)!} + \dots$$

$$= \frac{q^{n+1}}{(n+1)!} \left[1 + \frac{q}{n+2} + \frac{q^2}{(n+2)(n+3)} + \dots \right]$$

$$< \frac{q^{n+1}}{(n+1)!} \left[1 + \frac{q}{n+2} + \left(\frac{q}{n+2} \right)^2 + \dots \right]$$

Stosując wzór na sumę szer. geom.

$$R_n(q) < \frac{q^{n+1}}{(n+1)!}\frac{1}{1-\frac{q}{n+2}}$$

dla $0 \leq q < 1$

oraz

$$\frac{n+2}{n+1} < \frac{n+1}{n}$$

dostajemy warunek

$$0 < R_n(q) < u_n\frac{q}{n}, \quad 0 < q < 1$$

gdzie

$$u_n = \frac{q^n}{n!}$$

jest ostatnim wyrazem użytym przy sumowaniu elementów.
Wyrażenie na e^x (małe x)przyjmuje postać:

$$e^x = u_0 + u_1 + u_2 + \ldots + R_n(x)$$

gdzie

$$u_0 = 1, \quad u_k = \frac{xu_{k-1}}{k}, \quad k = 1, 2, 3, \ldots, n$$

Wówczas schemat iteracyjny obliczania wartości sumy jest następujący

$$\begin{aligned} u_k &= \frac{x}{k}u_{k-1} \\ s_k &= s_{k-1} + u_k \\ k &= 0, 1, 2, \ldots, n \end{aligned}$$

z warunkami

$$u_0 = 1, \quad s_{-1} = 0, \quad s_0 = 1$$

Załóżmy, że ε jest maksymalną wartością błędu obcięcia szeregu.

Proces sumowania przerywamy gdy spełniony będzie poniższy warunek

$$
\begin{aligned}
& |R_n(x)| \leq R_n(|x|) \\
< \quad & \frac{|x|^{n+1}}{(n+1)!} \cdot \frac{1}{1 - \frac{|x|}{n+2}} \\
< \quad & \frac{2|x|^{n+1}}{(n+1)!} = \frac{2|x|}{n+1} \cdot \frac{|x|^n}{n!} \\
< \quad & |u_n| \leq \varepsilon
\end{aligned}
$$

Ostatecznie warunek ten przyjmuje bardziej „przystępną” postać

$$|u_n(x)| < \varepsilon$$

np. obliczmy wartość $\sqrt{e}$
z dokładnością 2.5x10^{-6}

$$
\begin{aligned}
u_0 &= 1 \\
u_1 &= 0.5000000 \\
u_2 &= 0.1250000 \\
u_3 &= 0.0208333 \\
u_4 &= 0.0026042 \\
u_5 &= 0.0002604 \\
u_6 &= 0.0000217 \\
u_7 &= 0.0000016
\end{aligned}
$$

wynik

$$\sqrt{e} = 1.648721$$

Błędy zaokrągleń

Pojawiają się podczas wykonywania operacji arytmetycznych. Wynikają z ograniczonej reprezentacji liczb zmiennopozycyjnych.

Wielkość błędów zależy od:

a) dokładności reprezentacji
b) sposobu zaokrąglania wyniku
c) rodzaju przeprowadzanej operacji

Lemat Wilkinsona – błędy zaokrągleń powstające podczas wykonywania działań zmiennopozycyjnych są równoważne zastępczemu zaburzeniu liczb, na których wykonujemy działania.

Po przeprowadzeniu operacji dostajemy

$$fl(x) = x(1+\varepsilon_x), \quad fl(y) = y(1+\varepsilon_y)$$

Błędy względne zaokrągleń:

mnożenia

$$\frac{fl(x)fl(y) - xy}{xy} = (1+\varepsilon_x)(1+\varepsilon_y) - 1 =$$
$$= \varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y \approx \varepsilon_x + \varepsilon_y$$

dzielenia

$$\frac{fl(x)/fl(y) - x/y}{x/y} = \frac{1+\varepsilon_x}{1+\varepsilon_y} - 1 =$$
$$= \frac{\varepsilon_x - \varepsilon_y}{1-\varepsilon_y} \approx \varepsilon_x - \varepsilon_y$$

dodawania i odejmowania

$$\frac{fl(x) \pm fl(y) - (x \pm y)}{x \pm y} = \frac{x\varepsilon_x \pm y\varepsilon_y}{x \pm y} =$$
$$= \frac{x}{x \pm y}\varepsilon_x - \frac{y}{x \pm y}\varepsilon_y$$

Zwłaszcza przy odejmowaniu możemy dostać duży błąd, gdy

$$x \approx y, \quad \varepsilon_x \approx -\varepsilon_y$$

Wykonywanie kolejnych operacji na wynikach poprzednich operacji prowadzi do kumulacji błędów zaokrągleń.

Można je zmniejszyć ustalając odpowiednio sposób i kolejność wykonywanych działań lub zwiększając precyzję obliczeń (nie zawsze można).

Przykład
chcemy obliczyć sumę trzech liczb 0.48, 0.24, 0.12 w zapisie zmiennopozycyjnym 8 bitowym (5 bitów-M, 3 bity-W)

$$
\begin{aligned}
0.48_{(z2)} &= (0)1111(1)01 \\
0.24_{(z2)} &= (0)1111(1)10 \\
0.12_{(z2)} &= (0)1111(1)11
\end{aligned}
$$

Podejście pierwsze: wynik=(a+b)+c

$$a + b = (0)1111(1)01 + (0)1111(1)10$$

uzgadniamy wykładniki

$$a + b = (0)1111(1)01 + (0)01111(1)01$$ <-przepełnienie, trzeba uciąć ostatni bit w M2

$$\begin{aligned} fl(a+b) &= fl\big((0)1111(1)01 + (0)0111(1)01\big) \\ &= fl\big((0)0111(1)00 + (0)0011(1)00\big) \\ &= (0)1010(1)00 \end{aligned}$$

$$\begin{aligned} fl((a+b)+c) &= fl\big((0)1010(1)00 + (0)1111(1)11\big) \\ &= fl\big((0)1010(1)00 + (0)0001(1)00\big) \\ &= (0)1011(1)00 \\ &= 0.68125 \end{aligned}$$

$$\varepsilon = 18.8\%$$

Podejście drugie: wynik=a+(b+c)

$$
\begin{aligned}
fl(c+b) &= fl\big((0)1111(1)11 + (0)1111(1)10\big) \\
&= fl\big((0)0111(1)10 + (0)1111(1)10\big) \\
&= fl\big((0)0011(1)01 + (0)0111(1)01\big) \\
&= (0)1010(1)01
\end{aligned}
$$

$$
\begin{aligned}
fl((c+b)+a) &= fl\big((0)1010(1)01 + (0)1111(1)01\big) \\
&= fl\big((0)0101(1)00 + (0)0111(1)00\big) \\
&= (0)1100(1)00 \\
&= 0.75
\end{aligned}
$$

$$\varepsilon = 10.7\%$$

Szacowanie błędów zaokrągleń

a) Sumowanie liczb (jedna z częściej wykonywanych operacji)

$$s = \sum_{i=1}^{n} x_i$$

oznaczenie $s_k = fl\big(s_{k-1} + x(k)\big) = s_{k-1}^{'} + x^{'}(k)$

Zgodnie z lematem Wilkinsona:

$$s_{k-1}^{'} = s_{k-1}(1+\varepsilon_{k-1}^{1}) \qquad |\varepsilon_{k-1}^{1}| \le \varepsilon$$

$$x^{'}(k) = x(k)(1+\varepsilon_{k}^{2}) \qquad |\varepsilon_{k}^{2}| \le \varepsilon$$

Indeks 1 - suma
indeks 2 - x

Obliczona wartość sumy:

$$\begin{aligned} s &= x(1)(1+\varepsilon_1^2)(1+\varepsilon_2^1)\cdots(1+\varepsilon_{n-1}^1) \\ &+ x(2)(1+\varepsilon_2^2)(1+\varepsilon_2^1)\cdots(1+\varepsilon_{n-1}^1) \\ &+ x(3)(1+\varepsilon_3^2)(1+\varepsilon_3^1)\cdots(1+\varepsilon_{n-1}^1)+\cdots \\ &+ x(n-1)(1+\varepsilon_{n-1}^2)(1+\varepsilon_{n-1}^1)+x(n)(1+\varepsilon_n^2) \end{aligned}$$

Obliczona suma jest sumą zaburzonych składników. Wielkość zaburzeń zależy od kolejności wykonywania sumowania. Nie znamy wielkości poszczególnych mnożników, ale możemy oszacować maksymalne dopuszczalne zmiany składników:

$$x(1)_{min} = x(1)(1-\varepsilon)^n$$
$$x(i)_{min} = x(i)(1-\varepsilon)^{n+1-i}$$
$$x(1)_{max} = x(1)(1+\varepsilon)^n$$
$$x(i)_{max} = x(i)(1+\varepsilon)^{n+1-i}$$

Najmniej zaburzony jest składnik ostatni bo tylko (1+ε) lub (1-ε) razy.

Można stąd wysunąć wniosek odnośnie sumowania: **liczby należy sumować od najmniejszej do największej wg wartości bezwzględnej** - trzeba zmienić algorytm na dokładniejszy.

Przykład
Można też zmienić sposób wyliczania sumy (algorytm Kahana).

Znacznie ogranicza kumulację błędów gdy kolejne składniki różnią się znacznie wielkością

```
S=X[1]
C=0
for j=2 to N
{
   Y=X[j]-C
   T=S+Y
   C=(T-S)-Y
   S=T
}
```

Przykład
Wyznaczamy pole trójkąta wzorem Herona

$$S = \sqrt{q(q-a)(q-b)(q-c)}$$
$$q = (a+b+c)/2$$

ale wówczas błąd jest duży gdy

$$a \approx b+c, \quad q-a \approx 0$$

Dokładność może zwiększyć modyfikacja Kahana wzoru Herona:

$$S = \frac{1}{4}\sqrt{[a+(b+c)][c-(a-b)][c+(a-b)][a+(b-c)]}$$

b) Obliczanie wartości wielomianu

$$w(x) = x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n$$

W „tradycyjny” ale nieoptymalny sposób:

$$w(x) = \underbrace{x x \cdots x}_{n} + a_1 \underbrace{x \cdots x}_{n-1} + \cdots + a_{n-1} x + a_n$$

Wykonujemy

$$M = \frac{(n-1)(n+2)}{2}$$ operacji mnożenia

$$D = n$$ operacji dodawania

Schemat **Hornera**:

$$w(x) = x\Big(x\Big(x\cdots(x+a_1)+a_2\Big)+\cdots+a_{n-1}\Big)+a_n$$

Wykonujemy tylko **M=n-1** mnożeń i **D=n** dodawań.

Schemat Hornera pozwala znacznie zmniejszyć liczbę wykonywanych działań. Wyniki uzyskane wg powyższych algorytmów mogą się różnić. Natomiast oszacowana największa możliwa wartość błędu w obu przypadkach jest taka sama.

c) Błędy zaokrągleń w algorytmie iteracyjnym

Przykład

Chcemy numerycznie wyznaczyć wartość

$$y = \sqrt{x}$$

Zapisujemy funkcję w postaci uwikłanej

$$F(x, y) = 0$$

Jeśli przez y_n oznaczymy przybliżone rozwiązanie to z tw. Lagrange'a otrzymamy

$$\begin{aligned} \Delta F(x, y_n) &= F(x, y_n) - F(x, y) \\ &= (y_n - y) F'(x, \bar{y}_n) \end{aligned}$$

gdzie: $\bar{y}_n \in (y, y_n)$

stąd otrzymujemy

$$y_{n+1} = y_n - \frac{F(x, y_n)}{F'(x, y_n)}, \quad n = 0, 1, 2, \ldots$$

Jeśli naszą funkcję zdefiniujemy jako

$$F(x, y) = y^2 - x = 0, \quad F'(x, y) = 2y$$

To wówczas otrzymamy przepis iteracyjny na kolejne przybliżenia (*proces Herona*)

$$\begin{aligned} y_{n+1} &= y_n - \frac{y_n^2 - x}{2y_n} \\ &= \frac{1}{2}\left(y_n + \frac{x}{y_n}\right) \\ & y_0 \in R^+ \end{aligned}$$

Co ze zbieżnością elementów tego ciągu (dokładnością wyniku)?

Możemy zapisać warunek

$$y_i = p_i \sqrt{x}$$

wówczas ciąg p_i powinien być zbieżny do 1

$$p_0 > 0, \ldots, \quad p_{i+1} = \frac{1}{2}\left(p_i + \frac{1}{p_i}\right)$$

Proces jest zbieżny jeśli w kolejnych iteracjach uzyskujemy lepsze przybliżenie pierwiastka

$$\begin{aligned} q_i &= \frac{y_{i+1} - \sqrt{x}}{y_i - \sqrt{x}} \\ &= \frac{p_{i+1} - 1}{p_i - 1} \\ &= \frac{p_i - 1}{2p_i} \end{aligned}$$

co zachodzi gdy

$$|q_i| < 1 \quad , \qquad i = 0, 1, 2, \ldots$$

$$p_i > \frac{1}{3}$$

Możemy uzyskać zbieżność, ale co z błędami?
Przybliżenie wyniku w i+1 iteracji

$$fl(y_{i+1}) = \frac{1}{2}\left(y_i(1+\varepsilon_1^i) + \frac{x(1+\varepsilon_2^i)(1+\varepsilon_3^1)}{y_i}\right)$$

możemy przyjąć warunek na błędy zaokrągleń

$$|\varepsilon_k^i| \leq \varepsilon, \ k = 1, 2, 3$$

przy założeniu

$$y_{i+1}^{'} = fl(y_{i+1})$$

$$\begin{aligned} q_i^{'} &= \frac{y_{i+1}^{'} - \sqrt{x}}{y_i - \sqrt{x}} \\ &= \frac{p_i - 1}{2p_i} + \frac{p_i\varepsilon_1^i + \frac{\varepsilon_2^i + \varepsilon_3^i + \varepsilon_2^i\varepsilon_3^i}{p_i}}{2(p_i - 1)} \end{aligned}$$

pierwszy wyraz znika gdy

$$\lim_{p_i \to 1} \frac{p_i - 1}{2p_i} = 0$$

Ale drugi wyraz nie znika, co więcej jego wartość może rosnąć ze względu postać mianownika i błędy zaokrągleń.

W i+1 iteracji możemy nie uzyskać lepszego przybliżenia niż w i-tej iteracji. Rozwiązanie może oscylować wokół pewnej wartości. W takim przypadku możemy posłużyć się pojęciem: **maksymalnej granicznej dokładności**. Jest to oszacowanie błędu rozwiązania, którego nie można zmniejszyć dla danej metody iteracyjnej.
W powyższym przykładzie oszacowanie to otrzymamy jeśli wyznaczymy p_i dla którego błąd w kolejnej iteracji nie zmniejsza się, czyli

$$|q_i| \geq 1$$

Jeśli założymy że $\quad |\varepsilon_k^i| \ll 1, \quad p_i \approx 1$

to wówczas otrzymamy oszacowanie

$$\begin{aligned} |q_i^{'}| &= \left| \frac{p_i - 1}{2p_i} + \frac{p_i\varepsilon_1^i + \frac{\varepsilon_2^i + \varepsilon_3^i + \varepsilon_2^i\varepsilon_3^i}{p_i}}{2(p_i - 1)} \right| \\ &\approx \left| \frac{\varepsilon_1^i + \varepsilon_2^i + \varepsilon_3^i}{2(p_i - 1)} \right| \end{aligned}$$

Najgorsze oszacowanie uzyskamy, gdy założymy

$$\begin{aligned} \varepsilon = \varepsilon_k^i \quad , \quad k = 1, 2, 3 \\ \frac{3\varepsilon}{2|p_i - 1|} &\geqslant 1 \\ |p_i - 1| &\geqslant \frac{3}{2}\varepsilon \\ 1 - \frac{3}{2}\varepsilon &\leqslant p_i \leqslant 1 + \frac{3}{2}\varepsilon \end{aligned}$$

czyli przybliżenie

$$y_i = p_i\sqrt{x}$$

uzyskamy z maksymalnym błędem równym

$$\frac{3}{2}\sqrt{x}\varepsilon$$

Błąd ten stanowi **maksymalną graniczną dokładność** danej metody.

Zadanie numeryczne to jasny i niedwuznaczny opis powiązania funkcjonalnego między danymi wejściowymi i danymi wyjściowymi. Dane te składają się ze skończonej liczby wielkości rzeczywistych.

Algorytm numeryczny dla zadania numerycznego to opis poprawnie określonych operacji (arytmetycznych lub logicznych), które należy wykonać aby przekształcić wektor danych wejściowych na wektor danych wyjściowych.

Przykład
Określić największy pierwiastek rzeczywisty równania

$$x^3 + a_2x^2 + a_1x + a_0 = 0$$

dla wektora danych wejściowych

$$(a_0, a_1, a_2)$$

jest to zadanie numeryczne. Daną wyjściową jest szukany pierwisatek. Algorytm dla tego zadania – metoda Newtona, schemat iteracyjny, wzory Cardana.

Uwarunkowanie zadania
dla danych

$$\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$$

poszukujemy wyniku

$$\boldsymbol{w} = (w_1, w_2, \ldots, w_m)$$

czyli

$$\boldsymbol{w} = \varphi(\boldsymbol{d}), \quad \varphi : R_d \to R_w$$

Jeśli niewielkie względne zmiany danych zadania powodują duże względne zmiany rozwiązania, to zadanie takie jest źle uwarunkowane.
Wskaźnikiem uwarunkowania zadania nazywamy wielkość charakteryzującą wpływ zaburzeń danych na zaburzenie rozwiazania.
Przykład
„Perfidny” wielomian Wilkinsona

$$w(x) = \prod_{k=1}^{20}(x - k)$$

$$a_{19} = -210 \to a_{19}(\varepsilon) = -(210 + 2^{-23})$$

$$x_{15} = 13.9923 \pm i2.5188$$

Przykład
Jakie jest uwarunkowanie obliczania iloczynu skalarnego?

$$S = \sum_{i=1}^{n} a_i b_i \neq 0$$

zaburzamy dane wejściowe

$$a_i(\alpha) = a_i(1+\alpha_i)$$
$$b_i(\beta) = b_i(1+\beta_i)$$
$$\alpha_i \beta_i \approx 0$$

i liczmy względną zmianę wyniku

$$\left| \frac{\sum_{i=1}^{n} a_i(1+\alpha_i)b_i(1+\beta_i) - \sum_{i=1}^{n} a_i b_i}{\sum_{i=1}^{n} a_i b_i} \right| \approx$$
$$\approx \left| \frac{\sum_{i=1}^{n} a_i b_i(\alpha_i + \beta_i)}{\sum_{i=1}^{n} a_i b_i} \right| \leq$$
$$\leq \max_i |\alpha_i + \beta_i| \left(\sum_{i=1}^{n} |a_i b_i| / \left| \sum_{i=1}^{n} a_i b_i \right| \right)$$

Za wskaźnik uwarunkowania przyjmujemy

$$cond(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sum_{i=1}^{n} |a_i b_i|}{|\sum_{i=1}^{n} a_i b_i|}$$

dla $|a_i b_i| = a_i b_i \rightarrow cond(\boldsymbol{a}, \boldsymbol{b}) = 1$

czyli **zadanie** jest dobrze uwarunkowane.

Algorytmy numerycznie poprawne

Są to algorytmy numerycznie najwyższej jakości, dla których obliczone rozwiązanie jest „nieznacznie” zaburzonym rozwiązaniem dla „nieznacznie” zaburzonych danych.

„nieznaczne ” zaburzenie – zaburzenie na poziomie reprezentacji (rd(**d**), rd(**w**))

$$\|\boldsymbol{d} - rd(\boldsymbol{d})\| \leq \rho_d \|\boldsymbol{d}\|$$
$$\|\boldsymbol{w} - rd(\boldsymbol{w})\| \leq \rho_w \|\boldsymbol{w}\|$$

Algorytm A jest numerycznie poprawny w klasie zadań

$$\{\varphi; D\}$$

jeśli dla

$$\boldsymbol{d} \in D$$

i dostatecznie silnej arytmetyki, istnieje

$$\widetilde{\boldsymbol{d}} \in D_0$$

takie że

$$\|\boldsymbol{d} - rd(\widetilde{\boldsymbol{d}})\| \leq \rho_d K_d \|\boldsymbol{d}\|$$
$$\|\varphi(\widetilde{\boldsymbol{d}}) - fl(A(\boldsymbol{d}))\| \leq \rho_w K_w \|\varphi(\widetilde{\boldsymbol{d}})\|$$

K_d i K_w są stałymi kumulacji algorytmu. Charakteryzują one jakość algorytmu numerycznie poprawnego. Można je minimalizować stosując silniejszą arytmetykę (arytmetykę wyższej precyzji).

Przykład

Czy algorytm obliczania iloczynu skalarnego jest numerycznie poprawny?

$$S = \sum_{i=1}^{n} a_i b_i \neq 0$$

a) zaburzamy dane na **poziomie reprezentacji**

$$\widehat{a}_i = rd(a_i) = a_i(1+\alpha_i)$$
$$\widehat{b}_i = rd(b_i) = b_i(1+\beta_i)$$
$$|\alpha_i|, |\beta_i| \leq 2^{-t}, \quad i = 1, 2, \ldots, n$$

b) działania wykonujemy w arytmetyce fl

$$fl(A(\boldsymbol{a}, \boldsymbol{b})) =$$
$$(\ldots(\widehat{a}_1\widehat{b}_1(1+\varepsilon_1) + \widehat{a}_2\widehat{b}_2(1+\varepsilon_2))(1+\delta_2) + \ldots$$
$$+\widehat{a}_n\widehat{b}_n(1+\varepsilon_n))(1+\delta_n)$$

$$fl(A(\boldsymbol{a}, \boldsymbol{b})) = \sum_{i=1}^{n} a_i(1+\alpha_i)b_i(1+E_i)$$

$$1 + E_i = (1+\beta_i)(1+\varepsilon_i)\prod_{j=i}^{n}(1+\delta_j)$$

$$\delta_1 = 0, \quad |E_i| \lessapprox (n-i+3)2^{-t}$$

Interpretacja wyniku – dostaliśmy wynik dokładny dla trochę zaburzonych danych (K_w=0)

$$\widetilde{\boldsymbol{a}} = (a_1(1+\alpha_1), \ldots, a_n(1+\alpha_n))$$

$$\widetilde{\boldsymbol{b}} = (b_1(1+E_1), \ldots, b_n(1+E_n))$$

$$\|\boldsymbol{a} - \widehat{\boldsymbol{a}}\| \leq 2^{-t}\|\boldsymbol{a}\|, \quad (K_{d_t} = 1)$$

$$\|\boldsymbol{b} - \widehat{\boldsymbol{b}}\| \leq (n+1)2^{-t}\|\boldsymbol{b}\|, \quad (K_{d_2} \leq n+1)$$

Algorytmy numerycznie stabilne

- dokładne rozwiązanie:

$$\boldsymbol{w} = \varphi(\boldsymbol{d}), \quad \|\boldsymbol{d} - \widehat{\boldsymbol{d}}\| \leq \rho_d\|\boldsymbol{d}\|$$

- dane zaburzone na poziomie reprezentacji: $\widehat{\boldsymbol{d}}$
- dokładny wynik dla danych $\widehat{\boldsymbol{d}}$:

$$\widehat{\boldsymbol{w}} = \varphi(\widehat{\boldsymbol{d}}), \quad \|\widehat{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\| \leq \rho_w\|\widehat{\boldsymbol{w}}\|$$

- zaburzony wynik dla zaburzonych danych:

$$\widetilde{\boldsymbol{w}} = fl\,(\widehat{\boldsymbol{w}})$$

Dostajemy oszacowanie

$$\begin{aligned} \|\boldsymbol{w} - \widetilde{\boldsymbol{w}}\| &\leq \|\boldsymbol{w} - \widehat{\boldsymbol{w}}\| + \|\widehat{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\| \leq \\ &\leq (1+\rho_w)P(\boldsymbol{d}, \varphi) \end{aligned}$$

gdzie

$$P(\boldsymbol{d}, \varphi) = \rho_w\|\boldsymbol{w}\| + \max\|\varphi(\boldsymbol{d}) - \varphi(\widetilde{\boldsymbol{d}})\|$$

jest optymalnym poziomem błędu rozwiązania w danej arytmetyce (fl).

Algorytm A jest numerycznie stabilny, jeśli dla każdego $\boldsymbol{d} \in D$

istnieje stała K (ograniczenie od góry) i dla dostatecznie silnej arytmetyki zachodzi

$$\|\varphi(\boldsymbol{d}) - fl(A(\boldsymbol{d}))\| \leq K \cdot P(\boldsymbol{d}, \varphi)$$

Wskaźnik stabilności K powinien być jak najmniejszy – jego wielkość może służyć do oceny algorytmu.
Stabilność numeryczna jest własnością jakiej powinniśmy oczekiwać od algorytmu.

Złożoność obliczeniowa

rozważamy problem

$$\boldsymbol{w} = \varphi(\boldsymbol{d}), \quad \varphi : D \subset R_d \to R_w$$

Minimalną liczbę działań potrzebnych do obliczenia wyniku definiujemy jako

$$z(\varphi, D) = \sup_{d \in D} z(\varphi, D)$$

Wielkość z(φ,D) nazywamy złożoności obliczeniową zadania. Jeśli zadanie ma n danych istotnych tj.

$$\bigvee_{d \in D} \bigwedge_{1 \le j \le n} \bigvee_{\delta} \boldsymbol{d} + \delta \boldsymbol{e}_j \in D \Rightarrow \varphi(\boldsymbol{d}) \neq \varphi(\boldsymbol{d} + \delta \boldsymbol{e}_j)$$

$$\boldsymbol{e}_j = (0, \ldots, 0, 1, 0 \ldots, 0)^T$$

wówczas

$$z(\varphi, D) \ge n/2$$

i liczba istotnych danych określa oszacowanie z dołu złożoności obliczeniowej. Analiza algorytmów numerycznych powinna zawierać oprócz analizy dokładności metody również analizę jej złożoności obliczeniowej (pozwala oszacować koszt metody).