# Metody numeryczne

Janusz Szwabiński

2 października 2002

Plan wykładu

1. Dokładność w obliczeniach numerycznych
2. Złożoność obliczeniowa algorytmów
3. Interpolacja funkcji
4. Całkowanie numeryczne funkcji
5. Różniczkowanie numeryczne funkcji
6. Układy równań liniowych
7. Równania nieliniowe
8. Zagadnienia własne
9. Zagadnienia początkowe i brzegowe dla równań różniczkowych zwyczajnych

---

Literatura


1. Å. Björck, G. Dahlquist, "Metody numeryczne"
2. S. E. Koonin, "Computational Physics"
3. J. Stoer, "Wstęp do metod numerycznych"
4. D. Potter, "Metody obliczeniowe fizyki"
5. W. H. Press, S. A. Teutolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in ..." (http://www.nr.com)
6. K. A. Ross, Ch. R. B. Wright, "Matematyka dyskretna"
7. P. Wróblewski, "Algorytmy, struktury danych i techniki programowania"


---

... the purpose of computing is insight, not numbers ...

(Hamming)

---

Dokładność w obliczeniach numerycznych

1. Analiza błędów
   - Źródła błędów
   - Błędy bezwzględne i względne
   - Cyfry poprawne i znaczące

- Ucinanie i zaokrąglanie
- Przenoszenie się błędów

2. Przedstawienia liczb
   - Liczby całkowite
   - Reprezentacja zmiennopozycyjna
   - Arytmetyka zmiennopozycyjna i błędy zaokrągleń
3. Analiza zaburzeń pozornych

---

Źródła błędów

(i) błędy danych wejściowych

(ii) zaokrąglenia w czasie obliczeń

(iii) błędy zakresu

(iv) błędy obcięcia

(v) uproszczenia modelu matematycznego

(vi) błędy człowieka i maszynowe

---

Błędy bezwzględne i względne

$\tilde{a}$ - wartość przybliżona
$a$ - wartość dokładna

**Definicja** Błędem bezwzględnym wartości $\tilde{a}$ nazywamy $\Delta a = \tilde{a} - a$.

**Definicja** Błędem względnym wartości $\tilde{a}$ nazywamy $r(a) = \frac{\Delta a}{a}$, jeśli $a \neq 0$.

Jeśli $a$ pozostaje nieznane, wówczas szacuje się

$$r(a) \simeq \frac{\Delta a}{\tilde{a}}$$

---

Cyfry ułamkowe i istotne

**Definicja** Cyfry ułamkowe to wszystkie cyfry po kropce dziesiętnej w ułamku. Cyfry istotne to cyfry jakie pozostaną po pominięciu zer na początku ułamka.

**Przykład** Liczba 0,00245 ma pięć cyfr ułamkowych i trzy istotne. Liczba 12,12 ma dwie cyfry ułamkowe i cztery istotne

---

Cyfry poprawne i znaczące

**Definicja** Jeżeli $|\Delta a| \leq \frac{1}{2} * 10^{-t}$, to $\tilde{a}$ ma $t$ poprawnych cyfr ułamkowych. Cyfry istotne występujące w $\tilde{a}$ do pozycji $t$–tej po kropce to cyfry znaczące.

**Przykład** Liczba $0,001234 \pm 0,000004$ ma pięć cyfr poprawnych (ponieważ $0,000004 < \frac{1}{2} * 10^{-5}$) i trzy znaczące.

| | | |
|---|---|---|
| cyfry poprawne | $\rightarrow$ | błąd bezwzględny |
| cyfry znaczące | $\rightarrow$ | błąd względny |

---

Ucinanie i zaokrąglanie (1)

Cel: skrócenie liczby do danej długości $t$ cyfr ułamkowych

Metody:

1. ucinanie, czyli odrzucenie cyfr na prawo od $t$–tej
   - błąd bezwzględny $\leq 10^{-t}$
   - błąd ma systematycznie znak przeciwny niż sama liczba

---

Ucinanie i zaokrąglanie (2)

2. zaokrąglanie, czyli wybór liczby najbardziej zbliżonej do danej
   - jeśli $|a| = 0, \alpha_1\alpha_2 \ldots \alpha_t\alpha_{t+1} \ldots$, to

$$|\tilde{a}| := \begin{cases} 0, \alpha_1\alpha_2 \ldots \alpha_t & \text{jeśli } 0 \leq \alpha_{t+1} \leq 4 \\ 0, \alpha_1\alpha_2 \ldots \alpha_t + 10^{-t} & \text{jeśli } \alpha_{t+1} \geq 5 \end{cases}$$

   - $\Delta a \in (-\frac{1}{2} * 10^{-t}, \frac{1}{2} * 10^{-t})$

**Przykład** $t = 3$

| liczba | zaokrąglanie | ucinanie |
|---|---|---|
| 0,2379 | 0,238 | 0,237 |
| -0,2379 | -0,238 | -0,237 |
| 0,2375 | 0,238 | 0,237 |

---

Przenoszenie się błędów - dodawanie i odejmowanie (1)

Niech

$$\begin{aligned} x_1 &= \tilde{x}_1 \pm \epsilon_1 \\ x_2 &= \tilde{x}_2 \pm \epsilon_2 \end{aligned}$$

Zachodzą następujące związki

$$\begin{aligned} \tilde{x}_1 - \epsilon_1 + (\tilde{x}_2 - \epsilon_2) \leq \quad x_1 + x_2 \quad &\leq \tilde{x}_1 + \epsilon_1 + (\tilde{x}_2 + \epsilon_2) \\ \tilde{x}_1 - \epsilon_1 - (\tilde{x}_2 + \epsilon_2) \leq \quad x_1 - x_2 \quad &\leq \tilde{x}_1 + \epsilon_1 - (\tilde{x}_2 - \epsilon_2) \end{aligned}$$

czyli

$$\begin{aligned} \tilde{x}_1 + \tilde{x}_2 - (\epsilon_1 + \epsilon_2) \leq \quad x_1 + x_2 \quad &\leq \tilde{x}_1 + \tilde{x}_2 + (\epsilon_1 + \epsilon_2) \\ \tilde{x}_1 - \tilde{x}_2 - (\epsilon_1 + \epsilon_2) \leq \quad x_1 - x_2 \quad &\leq \tilde{x}_1 - \tilde{x}_2 + (\epsilon_1 + \epsilon_2) \end{aligned}$$

---

Przenoszenie się błędów - dodawanie i odejmowanie (2)

Zatem

$$\begin{aligned} x_1 + x_2 &= \tilde{x}_1 + \tilde{x}_2 \pm (\epsilon_1 + \epsilon_2) \\ x_1 - x_2 &= \tilde{x}_1 - \tilde{x}_2 \pm (\epsilon_1 + \epsilon_2) \end{aligned}$$

**Twierdzenie 1** *Oszacowanie błędu bezwzględnego wyniku dodawania lub odejmowania jest sumą oszacowań błędów bezwzględnych składników.*

**Dowód** Metodą indukcji matematycznej

---

Przenoszenie się błędów - mnożenie i dzielenie (1)

Z definicji błędu względnego $r$ wynika

$$\tilde{x} = x + xr = x(1 + r)$$

Stąd

$$\tilde{x}_1\tilde{x}_2 = x_1(1 + r_1)x_2(1 + r_2) = x_1x_2(1 + r_1)(1 + r_2)$$

Wobec tego

$$r(x_1 * x_2) = (1 + r_1)(1 + r_2) - 1 = r_1 + r_2 + r_1r_2 \simeq r_1 + r_2$$

jeżeli tylko $|r_1| \ll 1$ i $|r_2| \ll 1$.

---

Przenoszenie się błędów - mnożenie i dzielenie (2)

Podobnie

$$r\left(\frac{x_1}{x_2}\right) = \frac{1 + r_1}{1 + r_2} - 1 = \frac{r_1 - r_2}{1 + r_2} \simeq r_1 - r_2$$

jeżeli tylko $|r_1| \ll 1$ i $|r_2| \ll 1$.

Niech

$$|r_1| \leq \rho_1, \quad |r_2| \leq \rho_2.$$

Wówczas

$$|r_1 + r_2| \leq \rho_1 + \rho_2 \text{ i } |r_1 - r_2| \leq \rho_1 + \rho_2.$$

---

Przenoszenie się błędów - mnożenie i dzielenie (3)

**Twierdzenie 2** *W mnożeniu i dzieleniu oszacowania błędu względnego argumentów (o ile tylko błędy te są $\ll 1$) dodają się.*

---

Przenoszenie się błędów - znoszenie się składników

$$y = x_1 - x_2 \quad \Rightarrow |\Delta y| \leq |\Delta x_1| + |\Delta x_2| \quad \Rightarrow |\frac{\Delta y}{y}| \leq \frac{|\Delta x_1| + |\Delta x_2|}{|x_1 - x_2|}$$

$\Rightarrow$ jeżeli $x_1$ i $x_2$ niewiele się różnią między sobą, ich różnica może mieć małą dokładność względną

**Przykład** Niech $x_1 = 0,5765 \pm 0,00005$ i $x_2 = 0,5763 \pm 0,00005$. Mamy

$$x_1 - x_2 = 0,0001 \pm 0,0001$$

$\Rightarrow$ oszacowanie błędu jest tak samo duże, jak otrzymana różnica

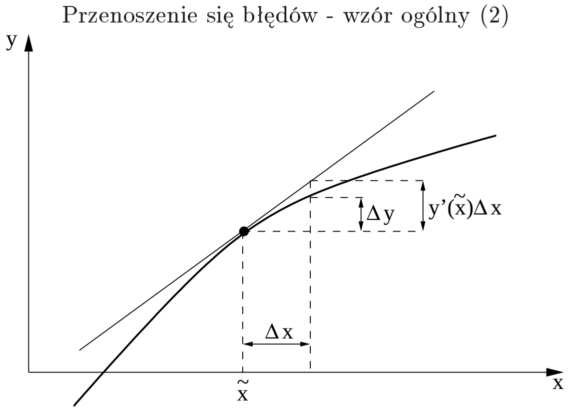
---

Przenoszenie się błędów - wzór ogólny (1)

Niech

| | | |
|---|---|---|
| $y(x)$ | - | funkcja jednej zmiennej |
| $x^{(0)}$ | - | dokładna wartość argumentu |
| $\tilde{x}$ | - | przybliżona wartość argumentu |
| $\Delta y = y(\tilde{x}) - y(x^{(0)})$ | - | błąd, którego oszacowania szukamy |

Przenoszenie się błędów - wzór ogólny (2)




$\Rightarrow$ $\Delta y$ można przybliżyć za pomocą różniczki funkcji $y$. $|y'(x)|$ to przy tym miara wrażliwości $y(x)$ na zakłócenie argumentu $x$

Przenoszenie się błędów - wzór ogólny (3)

**Twierdzenie 3** *Jeśli $y$ jest funkcją $x = (x_1, \ldots, x_n)$, $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ to wartość przybliżona argumentu $x_0 = (x_1^0, \ldots, x_n^0)$, oraz $\Delta x_i = \tilde{x}_i - x_i^0$ i $\Delta y = y(\tilde{x}) - y(x_0)$, to wówczas ogólny wzór na przenoszenie się błędów ma postać:*

$$\Delta y \approx \sum_{i=1}^{n} \frac{\partial y}{\partial x_i}(\tilde{x})\Delta x_i \quad \Rightarrow |\Delta y| \leq \sum_{i=1}^{n} |\frac{\partial y}{\partial x_i}(\tilde{x})||\Delta x_i|.$$

**Dowód** Załóżmy, że z $x_0$ do $\tilde{x}$ przesuwamy się w $n$ krokach, zmieniając w każdym z nich tylko jedną współrzędną. Zatem dla $i = 1, 2, \ldots, n-1$ mamy

$$x_i = (\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}^0, \ldots, x_n^0)$$

Przenoszenie się błędów - wzór ogólny (4)

**Dowód** (ciąg dalszy)
oraz

$$x_n = \tilde{x}.$$

Gdy przechodzi się z $x_{i-1}$ do $x_i$, zmienia się $i$–ta współrzędna z $x_i^0$ na $\tilde{x}_i$. Z twierdzenia o wartości średniej wynika

$$y(x_i) - y(x_{i-1}) = \frac{\partial y}{\partial x_i}(\xi_i)(\tilde{x}_i - x_i^0) \simeq \frac{\partial y}{\partial x_i}(\tilde{x})\Delta x_i,$$

gdzie $\xi_i \in (x_{i-1}, x_i)$.

Przenoszenie się błędów - wzór ogólny (5)

**Dowód** (ciąg dalszy)

Ponieważ

$$\Delta y = y(x_n) - y(x_0) = \sum_{i=1}^{n}(y(x_i) - y(x_{i-1})),$$

otrzymujemy

$$\Delta y \approx \sum_{i=1}^{n} \frac{\partial y}{\partial x_i}(\tilde{x})\Delta x_i. \qquad \blacksquare$$

---

Znoszenie się błędów

**Przykład** Niech $y = z_1 + z_2$, gdzie $z_1 = (x^2+1)^{1/2}$, $z_2 = 200 - x$ oraz $x = 100 \pm 1$. Błędy wielkości $z_1$ i $z_2$ znoszą się, ponieważ

$$\frac{\Delta y}{\Delta x} \simeq \frac{dy}{dx} = \frac{dz_1}{dy} + \frac{dz_2}{dy} \approx -\frac{1}{2x^2}.$$

Zatem $|\Delta y| \lesssim \frac{1}{2} * 10^{-4}$ i $z_1$ oraz $z_2$ należało by wyznaczyć z czterema cyframi ułamkowymi, choć nawet cyfry jedności nie są pewne. Jeśli jednak przekształcimy $y$ w następujący sposób:

$$y = (\sqrt{x^2+1} - x) + 200 = \frac{1}{x + \sqrt{x^2+1}} + 200,$$

otrzymamy $y$ z żądaną dokładnością przy niskiej dokładności pierwiastka kwadratowego. Wynik: $y = 200,0050 \pm \frac{1}{2} * 10^{-4}$.

---

Zaburzenia eksperymentalne

Skomplikowane zadania obliczeniowe:

- związki między danymi WE/WY zbyt skomplikowane, aby stosować wzór na przenoszenie się błędów

$\Rightarrow$ obliczenia testowe, w których bada się wrażliwość danych WY na zaburzenie danych WE

---

Przedstawienia liczb

Maszyny cyfrowe:

- skończona pamięć
- skończona liczba operacji WE/WY w jednostce czasu

$\Rightarrow$ zbiór liczb reprezentowanych w maszynie skończony

---

Liczby całkowite (1)

Maszynowa, $N$-bitowa liczba całkowita $z$ reprezentowana jest w następujący sposób:

$$z = \{\alpha_0, \alpha_1, \ldots, \alpha_{N-1}\}, \quad \alpha_n \in \{0, 1\},$$

przy czym

$$z = -\alpha_{N-1} * 2^{N-1} + \sum_{n=0}^{N-2} \alpha_n * 2^n.$$

Zakres: $-2^{N-1} \leq z \leq 2^{N-1} - 1$.
Oszacowanie błędu: $|\Delta z| \leq 0,5$ .

---

Liczby całkowite (2)

**Przykład**

| Liczba | Reprezentacja 8-bitowa | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\alpha_7$ | $\alpha_6$ | $\alpha_5$ | $\alpha_4$ | $\alpha_3$ | $\alpha_2$ | $\alpha_1$ | $\alpha_0$ |
| +1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| +2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| +117 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| -118 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| Wartościowość | $-2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

---

Liczby całkowite (3)

Liczby ujemne:

- w liczbie dodatniej o tym samym module zmień każdy bit na przeciwny
- dodaj 1

$\Rightarrow$ dzięki temu nie trzeba rozróżniać na liczby dodatnie i ujemne przy dodawaniu

**Przykład**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| +2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| +1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

---

Reprezentacja zmiennopozycyjna (1)

Liczba maszynowa $g$ w reprezentacji zmiennopozycyjnej ma postać:

$$g = (-1)^s * m * B^e, \quad 1/B \leq m < 1,$$

| | | |
|---|---|---|
| $s$ | - | bit znaku, |
| $m$ | - | mantysa, |
| $e$ | - | wykładnik (cecha), |
| $B$ | - | podstawa (ustalona), |

$\Rightarrow$ operacje na ustalonej liczbie cyfr istotnych.

---

Reprezentacja zmiennopozycyjna (2)

32–bitowy, zmiennopozycyjny standard IEEE (*Institute of Electrical and Electronics Engineers*):

$$g = (-1)^s 1.f * 2^{e-127},$$

| | | |
|---|---|---|
| $s$ | - | bit znaku, |
| $f$ | - | część ułamkowa (23 bity), |
| $e$ | - | wykładnik przesunięty o 127 (8 bitów), |

Zakres: $-10^{38} \leq g \leq 10^{38}$.

---

Reprezentacja zmiennopozycyjna (3)

Przesunięcie wykładnika:

- ogólnie o $2^{k-1} - 1$, gdzie $k$ to liczba bitów w polu wykładnika,

- wykładnik reprezentowany przez liczby dodatnie,
- najmniejsza liczba maszynowa może być odwrócona bez wykroczenia poza zakres.

---

Reprezentacja zmiennopozycyjna (4)

Wartości specjalne:

| | |
|---|---|
| 0 00000000 00000000000000000000000 | 0 |
| 1 00000000 00000000000000000000000 | -0 |
| 0 11111111 00000000000000000000000 | $+\infty$ |
| 1 11111111 00000000000000000000000 | $-\infty$ |
| 0 11111111 00000100000000000000000 | NaN |
| 1 11111111 00100010001001010101010 | NaN |

---

Reprezentacja zmiennopozycyjna (5)

**Przykład** Chcemy przekształcić liczbę 5,375 do formatu IEEE:

- przekształcamy do postaci dwójkowej:
  $5,375 = 4 + 1 + 0,25 + 0,125 =$
  $1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3} = 101.011$
- renormalizujemy otrzymaną liczbę: $101.011 * 2^0 = 1.01011 * 2^2$
- pomijamy wiodącą jedynkę w mantysie $\rightarrow$ 01011
- wyznaczamy wykładnik: $2 + 127 = 129 = 10000001$
- znak dodatni $\rightarrow$ 0

$\Rightarrow$

| znak | wykładnik | mantysa |
|---|---|---|
| 0 | 10000001 | 01011000000000000000000 |

---

Reprezentacja zmiennopozycyjna (6)

**Przykład** -113,3125:

- przekształcamy wartość bezwzględną do postaci dwójkowej:
  - część całkowita: $113 = 1110001$
  - część ułamkowa:
    $0,3125 * 2 = 0,625 \quad \rightarrow \ 0$
    $0,625 * 2 = 1,25 \quad \rightarrow \ 1$
    $0,25 * 2 = 0,5 \quad \rightarrow \ 0$
    $0,5 * 2 = 1,0 \quad \rightarrow \ 1$

  $\Rightarrow \quad 113,3125 = 1110001,0101$
- renormalizujemy otrzymaną liczbę: $1110001,0101 * 2^0 = 1,1100010101 * 2^6$
- pomijamy wiodącą jedynkę w mantysie $\rightarrow$ 1100010101

---

Reprezentacja zmiennopozycyjna (7)

**Przykład** (ciąg dalszy)

- wyznaczamy wykładnik: $6 + 127 = 133 = 10000101$
- znak ujemny $\rightarrow 1$

$\Rightarrow$

| znak | wykładnik | mantysa |
|---|---|---|
| 1 | 10000101 | 11000101010000000000000 |

---

Reprezentacja zmiennopozycyjna (8)

**Przykład** 1,7:

- przekształcamy część ułamkową do postaci dwójkowej:

$$\begin{array}{ll}
0,7 * 2 = 1,4 & \rightarrow 1 \\
0,4 * 2 = 0,8 & \rightarrow 0 \\
0,8 * 2 = 1,6 & \rightarrow 1 \\
0,6 * 2 = 1,2 & \rightarrow 1 \\
0,2 * 2 = 0,4 & \rightarrow 0 \\
0,4 * 2 = 0,8 & \rightarrow 0 \\
0,8 * 2 = 1,6 & \rightarrow 1 \\
0,6 * 2 = 1,2 & \rightarrow 1 \\
0,2 * 2 = 0,4 & \rightarrow 0 \\
\vdots & \\
\end{array}$$

$$\Rightarrow \quad 1,7 \simeq 1,10110011001100110011001 * 2^0$$

---

Reprezentacja zmiennopozycyjna (9)

**Przykład** (ciąg dalszy)

- pomijamy wiodącą jedynkę w mantysie $\rightarrow 10110011001100110011001$
- wyznaczamy wykładnik:
  $0 + 127 = 127 = 01111111$
- znak dodatni $\rightarrow 0$

$\Rightarrow$

| znak | wykładnik | mantysa |
|---|---|---|
| 0 | 01111111 | 10110011001100110011001 |

---

Arytmetyka zmiennopozycyjna (1)

Dla dowolnych liczb maszynowych $x$ i $y$ mamy

$$\begin{aligned}
x +^* y &:= rd(x + y), \\
x -^* y &:= rd(x - y), \\
x \times^* y &:= rd(x \times y), \\
x /^* y &:= rd(x/y),
\end{aligned}$$

gdzie $rd$ to odwzorowanie zaokrąglania (lub ucinania), przekształcające liczbę rzeczywistą na maszynową.

---

Arytmetyka zmiennopozycyjna (2)

**Przykład** Niech $a = 0,1234567 * 10^0$, $b = 0,4711325 * 10^4$ oraz $c = -b$. Wówczas mamy:

$$a +^* b = 0,0000123 * 10^4 + 0,4711325 * 10^4 = 0,4711448 * 10^4,$$
$$(a +^* b) +^* c = 0,0000123 * 10^4 = 0,123 * 10^0.$$

Z drugiej strony

$$\begin{aligned} b +^* c &= 0, \\ a +^* (b +^* c) &= 0,1234567 * 10^0 \neq (a +^* b) +^* c, \end{aligned}$$

czyli $+^*$ nie jest łączne.

---

Arytmetyka zmiennopozycyjna (3)

$\Rightarrow$ algorytmy równoważne matematycznie nie zawsze są równoważne numerycznie.

**Definicja** Algorytmy nazywamy *równoważnymi matematycznie*, jeśli dają te same wyniki dla jednakowych danych wejściowych, o ile tylko obliczenia wykonuje się bez zaokrągleń.

**Definicja** Algorytmy nazywamy *równoważnymi numerycznie*, gdy wyniki dla jednakowych danych wejściowych różnią się tylko o tyle, o ile mogą zmienić się dokładne dane wyjściowe zadania, gdy dane wejściowe zaburzy się o niewiele $u$, gdzie $u$ jest względnym błędem zaokrąglenia lub ucięcia.

---

Analiza zaburzeń pozornych (1)

Z definicji $op^*$ wynika

$$|x \; op^* \; y \quad - \quad x \; op \; y| \leq |x \; op \; y| u,$$

gdzie $u$ to jednostka maszynowa.

$$\Rightarrow \quad \exists \delta, \; x \; op^* \; y = (x \; op \; y)(1 + \delta), \quad |\delta| \leq u.$$

$\Rightarrow$ wynik każdego z działań $op^*$ można przedstawić jako wynik dokładnego działania wykonanego na zaburzonych argumentach, przy czym zaburzenie względne nie przewyższa $u$

---

Analiza zaburzeń pozornych (2)

**Definicja** Dane wyjściowe otrzymane za pomocą algorytmu $A$ interpretujemy jako dokładne dane wyjściowe tych samych obliczeń, ale ze zmienionymi danymi wejściowymi. Wówczas *wskaźnik uwarunkowania $C_A$ algorytmu $A$*, wskazuje wystarczający zakres takiej względnej zmiany danych wejściowych z $u$ jako jednostką.

**Definicja** *Wskaźnik uwarunkowania $C_P$ zadania $P$* z pewnymi danymi wejściowymi jest największą zmianą względną, mierzoną w jednostkach $u$, dokładnych danych wyjściowych, spowodowaną zakłóceniem względnym danych wejściowych o wielkości $u$.

Analiza zaburzeń pozornych (3)

Jeśli $r$ jest oszacowaniem błędu względnego danych wejściowych, to

$$C_P(r + C_A)$$

jest oszacowaniem błędu względnego danych wyjściowych. $C_A$ i $C_P$ zależą od danych wejściowych, natomiast są wzajemnie niezależne.

---

Analiza zaburzeń pozornych (4)

**Przykład** $P$ - zadanie obliczenia wartości funkcji $f$ w punkcie $x$. Z definicji $C_P$ wynika, że wskaźnik ten jest większą z wartości wyrażenia

$$\left| \frac{\frac{f(x+\Delta x) - f(x)}{f(x)}}{\frac{\Delta x}{x}} \right|$$

dla $\Delta x = ux$ i $\Delta x = -ux$ (o ile tylko funkcja $f$ jest dostatecznie reguralna w otoczeniu $x$). Stąd w praktyce przyjmuje się często

$$C_P = \left| \frac{\frac{f'(x)}{f(x)}}{\frac{1}{x}} \right|.$$

---

Analiza zaburzeń pozornych (5)

**Przykład** Chcemy obliczyć

$$x = \frac{1}{1 - \alpha^2}.$$

Mamy

$$C_P = \left| \frac{2\alpha^2}{1 - \alpha^2} \right|$$

$\Rightarrow$ zadanie źle uwarunkowane dla $\alpha^2 \approx 1$

$\Rightarrow$ zadanie bardzo dobrze uwarunkowane dla $\alpha^2 \ll 1$

---

Złożoność obliczeniowa algorytmów

1. Notacja $O$
2. Złożoność obliczeniowa algorytmów
   - Uproszczone zasady analizy algorytmów
   - Kilka słów o rekurencji

---

Notacja $O$ (1)

**Definicja** Niech $f$ i $g$ będą ciągami liczb rzeczywistych. Piszemy

$$f(n) = O(g(n))$$

wtedy, gdy istnieje stała dodatnia $C$ taka, że

$$|f(n)| \leq C|g(n)|$$

dla dostatecznie dużych wartości $n$.

Notacja $O$ (2)

**Twierdzenie 4** *W poniższej hierarchii ciągów każdy z nich jest O od wszystkich ciągów na prawo od niego:*

$$1, \log_2 n, \ldots, \sqrt[4]{n}, \sqrt[3]{n}, \sqrt{n}, n, n \ln n, n\sqrt{n}, n^2, n^3, \ldots, 2^n, n!, n^n.$$

**Dowód**

(i) $n$ można przedstawić w postaci

$$n = 2 * \frac{3}{2} * \frac{4}{3} \ldots \frac{n-1}{n-2} \frac{n}{n-1}.$$

Ponieważ po prawej stronie jest $n-1$ czynników i każdy z nich jest co najwyżej równy 2, więc $n \leq 2^{n-1}$, a zatem również $n \leq 2^n$.

Notacja $O$ (3)

**Dowód** (ciąg dalszy)

(ii) Ponieważ $n \leq 2^{n-1}$, więc

$$\log_2 n \leq \log_2 2^{n-1} = n - 1$$

dla $n \geq 1$.

(iii) Dla $n > 4$ mamy

$$n! = (4!) * 5 * \ldots * (n-1) * n.$$

Pierwszy czynnik $4!$ jest większy od $2^4$, a każdy z pozostałych $n-4$ czynników jest większy od 2. Zatem

$$n! > 2^4 * 2^{n-4} = 2^n.$$

Notacja $O$ (4)

**Dowód** (ciąg dalszy)

(iv) Nierówność $n! < n^n$ jest oczywista, ponieważ $n!$ jest iloczynem liczb naturalnych, z których wszystkie poza jedną są mniejsze od $n$. ∎

Notacja $O$ (5)

**Twierdzenie 5**

*(a) Jeśli* $f(n) = O(g(n))$ *i* $c$ *jest stałą, to*

$$c * f(n) = O(g(n)).$$

*(b) Jeśli* $f(n) = O(g(n))$ *i* $h(n) = O(g(n))$*, to*

$$f(n) + h(n) = O(g(n)).$$

*(c) Jeśli $f(n) = O(a(n))$ i $g(n) = O(b(n))$, to*

$$f(n) * g(n) = O(a(n) * b(n)).$$

*(d) Jeśli $a(n) = O(b(n))$ i $b(n) = O(c(n))$, to*

$$a(n) = O(c(n)).$$

---

Notacja $O$ (6)

**Dowód**

(a) Z definicji notacji $O$ mamy

$$|f(n)| \leq C|g(n)|$$

dla dostatecznie dużych $n$. Zatem:

$$|cf(n)| = c|f(n)| \leq c * C|g(n)|,$$

czyli $cf(n) = O(g(n))$.

---

Notacja $O$ (7)

**Dowód** (ciąg dalszy)

(b) Jeśli $f(n) = O(g(n))$ i $h(n) = O(g(n))$, to istnieją liczby dodatnie $C$ i $D$ takie, że

$$|f(n)| \leq C|g(n)|$$

oraz

$$|h(n)| \leq D|g(n)|$$

dla dostatecznie dużych $n$. Ponieważ $|x+y| \leq |x|+|y|$ dla $x, y \in \mathbb{R}$, więc wynika stąd, że

$$|f(n) + h(n)| \leq |f(n)| + |h(n)| \leq (C + D)|g(n)|$$

dla dostatecznie dużych $n$. Zatem $f(n) + h(n) = O(g(n))$.

---

Notacja $O$ (8)

**Dowód** (ciąg dalszy)

(c) Jeśli $f(n) = O(a(n))$ i $g(n) = O(b(n))$, to istnieją liczby dodatnie $C$ i $D$ takie, że

$$|f(n)| \leq C|a(n)|$$

oraz

$$|g(n)| \leq D|b(n)|$$

dla dostatecznie dużych $n$. Stąd wynika, że

$$|f(n) * g(n)| \leq C * D|a(n)b(n)|$$

dla dostatecznie dużych $n$. Zatem $f(n) * g(n) = O(a(n) * b(n))$.

Notacja $O$ (9)

**Dowód** (ciąg dalszy)

(d) Jeśli $a(n) = O(b(n))$ i $b(n) = O(c(n))$, to istnieją liczby dodatnie $C$ i $D$ takie, że

$$|a(n)| \leq C|b(n)|$$

oraz

$$|b(n)| \leq D|c(n)|$$

dla dostatecznie dużych $n$. Zatem

$$|a(n)| \leq C|b(n)| \leq CD|c(n)|,$$

dla dostatecznie dużych $n$, a więc $a(n) = O(c(n))$. ∎

---

Notacja $O$ (10)

**Twierdzenie 6** *Dla dowolnych ciągów $a(n)$ i $b(n)$ mamy*

*(a)* $O(a(n)) + O(b(n)) = O(\max\{|a(n)|, |b(n)|\})$.

*(b)* $O(a(n)) * O(b(n)) = O(a(n) * b(n))$.

---

Notacja $O$ (11)

**Dowód**

(a) Niech $f(n) = O(a(n))$ oraz $g(n) = O(b(n))$. Wtedy istnieją liczby dodatnie $C$ i $D$ takie, że

$$|f(n)| \leq C|a(n))|$$

oraz

$$|g(n)| \leq C|b(n))|$$

dla dostatecznie dużych $n$. Zachodzi zatem

$$\begin{aligned} &|f(n) + g(n)| \\ &\quad \leq |f(n)| + |g(n)| \leq C|a(n)| + D|b(n)| \\ &\quad \leq C \max\{|a(n)|, |b(n)|\} + D \max\{|a(n)|, |b(n)|\} \\ &\quad = (C + D) \max\{|a(n)|, |b(n)|\} \end{aligned}$$

---

Notacja $O$ (12)

**Dowód**

(a) (ciąg dalszy)
dla dostatecznie dużych $n$. Czyli $f(n) + g(n) = O(\max\{|a(n)|, |b(n)|\})$.

(b) Dowód wynika natychmiast z poprzedniego twierdzenia (podpunkt c) ∎

---

**Przykład** Zestawienie czasów wykonania algorytmów różnych klas. Założenie: elementarny czas wykonania wynosi jedną mikrosekundę ($10^{-6}$s).

| $n$ | 10 | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|
| $O(n)$ | 0,00001s | 0,00002s | 0,00003s | 0,00004s | 0,00005s | 0,00006s |
| $O(n^2)$ | 0,0001s | 0,0004s | 0,0009s | 0,0016s | 0,0025s | 0,0036 |
| $O(n^3)$ | 0,001s | 0,008s | 0,027s | 0,064s | 0,125s | 0,216s |
| $O(2^n)$ | 0,001s | 1,048s | 17,9min | 12,7dni | 35,7lat | 366w |
| $O(3^n)$ | 0,059s | 58min | 6,5lat | 3855w | 227*$10^6$w | 1,3*$10^{13}$w |
| $O(n!)$ | 3,6s | 771w | 8,4*$10^{16}$w | 2,6*$10^{32}$w | 9,6*$10^{47}$w | 2,6*$10^{66}$w |

($w$ oznacza tutaj wiek)

---

Uproszczone zasady analizy algorytmów (1)

1. w analizie programu należy zwracać uwagę tylko na najbardziej "czasochłonne" operacje,
2. należy wybrać wiersz programu znajdujący się w najgłębiej położonej instrukcji iteracyjnej, następnie zliczyć liczbę jego wywołań i na tej podstawie wnioskować o klasie algorytmu.

---

Uproszczone zasady analizy algorytmów (2)

**Przykład** Przeanalizujmy poniższy fragment programu:

```
while(i<N)
  {
   while(j<=N)
     {
      suma=suma+2;
      j=j+1;
     }
   i=i+2;
  }
```

Wiersz `suma=suma+2` będzie tu wykonywany $\frac{N(N+1)}{2}$ razy. Powyższy fragment jest zatem klasy $O(n^2)$ (rozmiarem danych jest tu wielkość $N$).

---

Kilka słów o rekurencji (1)

$$\begin{aligned} 0! &= 1 \\ n! &= n*(n-1)! \end{aligned}$$

```
int silnia(int n)
  {
   if(n>0)
     return n*silnia(n-1);
   else
     return 1;
  }
```

Niech czas sprawdzenia warunku wynosi $t_c$. Załóżmy, że jest to najbardziej czasochłonna operacja. Wówczas

$$\begin{aligned} T(0) &= t_c, \\ T(n) &= t_c + T(n-1), \quad \text{dla } n \geq 1. \end{aligned}$$

Kilka słów o rekurencji (2)

$$\begin{aligned} T(n) &= t_c + T(n-1), \\ T(n-1) &= t_c + T(n-2), \\ &\vdots \\ T(1) &= t_c + T(0), \\ T(0) &= t_c, \end{aligned}$$

$\Rightarrow$ $T(n) = (n+1)t_c$

$\Rightarrow$ algorytm jest klasy $O(n)$