Lecture 1

October 14, 2019

[1]:
```
%%HTML
<style>
.rendered_html table, .rendered_html th, .rendered_html tr, .rendered_html td {
     font-size: 100%;
}
</style>
```

```
<IPython.core.display.HTML object>
```

1 Metody Numeryczne

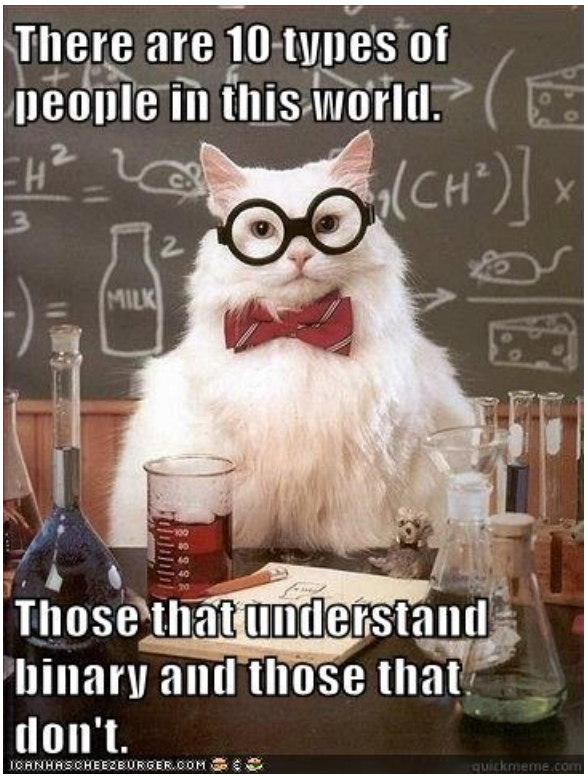
1.1 Elementy analizy numerycznej

1.1.1 dr hab. in. Jerzy Baranowski, Prof. nadzw.

1.2 Informacje ogólne

- Katedra Automatyki i Robotyki, C3, p. 214
- Konsultacje
  - Czwartki 11:00-12:00 (o ile nie ma Kolegium Wydziaowego lub seminarium)
- jb@agh.edu.pl
- wykady dostpne tutaj: https://github.com/KAIR-ISZ/public_lectures

2 Reprezentacja liczb



2.1 Kod binarny

- Zapis liczby z wykorzystaniem dwóch symboli **1** i **0**
- Podstawa współczesnego sposobu reprezentacji informacji

2.2 Zamierzcha historia

- Pingala, Chandastra i Prozodia
  - Ok. 4 wiek pne
  - Wykorzystanie zapisu w formie zer i jedynek do opisu metrum
- Chiny, hexagramy, Shao Yong, I-Ching
- Leibniz

2.3 Algebra Boole'a

$$
\begin{aligned}
x \wedge y &= xy && \text{Koniunkcja} \\
x \vee y &= x + y - xy && \text{Alternatywa} \\
\neg x &= 1 - x && \text{Negacja} \\
x \rightarrow y &= (\neg x \vee y) && \text{Implikacja} \\
x \oplus y &= (x \vee y) \wedge \neg(x \wedge y) && \text{EXOR} \\
x = y &= \neg(x \oplus y) && \text{Rwnowano}
\end{aligned}
$$

2.4 Nieco mniej zamierzcha historia

- 1937 Shannon – przekanikowa realizacja operacji binarnych i algebry Boole'a
- 1937 Stibitz – Pierwszy komputer przekanikowy (dodawanie)

2.5 Kod binarny

	0	**0**	**1**	**0**	**1**	**0**	**1**	**1**	
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Co daje $ =$2^{5+2}3+2^{1+2}0=32+8+2+1=43$

2.6 Liczby naturalne

- Ogólnie zakres od 0 do 2n-1
- 8 bit – zakres od 0 do 255
- 16 bit – zakres od 0 do 65,535 (short, int)
- 32 bit – zakres od 0 do 4,294,967,295 (long)

W Pythonie i matlabie za bardzo nie przejmujemy si typami, chyba e je wymusimy

2.7 Operacje na liczbach binarnych

- Dodawanie
  - 0+0=0
  - 0+1=1
  - 1+0=1
  - 1+1=0, przenie 1
- Jak w dodawaniu pisemnym

```
    1 1 1 1 1(cyfry przenoszone)
0 1 1 0 1(1310)
+   1 0 1 1 1(2310)
------------
=1 0 0 1 0 0 (3610)
```

2.8 Operacje na liczbach binarnych

- Odejmowanie
  - 0-0=0
  - 0-1=1, poyczka 1
  - 1-0=1
  - 1-1=0,
- Analogicznie

```
        *    * * *(poyczki)
1 1 0 1 1 1 0(11010)
-     1 0 1 1 1(2310)
---------------
= 1 0 1 0 1 1 1 (8710)
```

2.9 Co z liczbami ujemnymi?

Uzupeniamy zapis o tzw. bit znaku

	1	**0**	**1**	**0**	**1**	**0**	**1**	**1**
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Co daje $ =(-1)^{1(2}3+2^{1+2}0)=-(8+2+1)=-11$
Zmieniaj si zakresy: - 8 bit (-128 do 127) - 16 bit (32,768 do 32,767) - itd

2.10 Problemy

- Niepraktyczny zapis
- Trzeba przekodowywa wyniki operacji
- Potencjalnie podatniejsze na bdy

2.11 Kod uzupenienia do 2 (U2)

	1	**1**	**1**	**1**	**1**	**0**	**1**	**1**
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

Co daje $ =-2^{7+2}6+2^{5+2}4+2^{3+2}1+2^0$
$= -128 + 64 + 32 + 16 + 8 + 2 + 1 = -5$

2.12 Bardzo atwa konwersja

- Liczby dodatnie s takie same jak byy
- Aby zamieni liczb na jej przeciwn wystarczy zanegowa wszystkie bity i do wyniku doda 1 (*w obie strony*)

	0	**0**	**0**	**0**	**0**	**1**	**0**	**1**	510	orygina
1	1	1	1	1	0	1	0			negacja
1	**1**	**1**	**1**	**1**	**0**	**1**	**1**	-510		dodanie 1
0	0	0	0	0	1	0	0			negacja
0	**0**	**0**	**0**	**0**	**1**	**0**	**1**	510		dodanie 1
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0			

2.13 Jaka z tego korzy?

- Odejmowanie staje si dodawaniem (prawie)

$$A - B = A + \neg B + 1$$

- Przykad 13 – 7 (na 8 bitach)

```
      1 1 1 1        1(cyfry przenoszone)
0 0 0 0 1 1 0 1(1310)
1 1 1 1 1 0 0 0(zanegowane 710)
+                    1(jedynka)
-----------------
= 0 0 0 0 0 1 1 0 (610)
```

2.14 Operacje na liczbach binarnych

Mnoenie równie przypomina mnoenie pisemne

```
              1 0 1 1 1110
* 1 0 1 0 1010
-----------
0 0 0 0
+     1 0 1 1
+   0 0 0 0
+ 1 0 1 1
---------------
= 1 1 0 1 1 1 0 11010
```

2.15 A co z uamkami?

S dwa sposoby zapisu liczb niecakowitych - Staoprzecinkowy (staopozycyjny) - Zmienno-przecinkowy (zmiennopozycyjny)

2.16 Zapis staoprzecinkowy

		$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}		

$$2^1 + 2^{-1} + 2^{-2} + 2^{-3} = 2 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = 2.875$$

2.17 Zalety zapisu staoprzecinkowego

- Nie ma rónicy w kodowaniu
- Mamy stale okrelon dokadno, któr moemy w miar dokadnie ksztatowa
- Stosunkowa prostota
- Mae wymagania sprztowe

2.18 Wady zapisu staoprzecinkowego

Problemy z dokadnoci, np. nie da si dokadnie przedstawi liczby 0.1 - Na 3 bitach czci uamkowej rónica wynosi 0.025 - Na 7 bitach czci uamkowej rónica wynosi ok. 0.001

2.19 Jak wykonujemy dziaania?

- Dziaania wykonujemy traktujc zapis liczby staoprzecinkowej jako normaln binarn
- Kod U2 dalej dziaa
- Naley pamita, e wtedy liczba jest pomnoona przez 2n gdzie n to ilo bitów czci uamkowej
- W liczbach poddanych dziaaniu liczba bitów czci cakowitej i uamkowej musi by równa

2.20 Dziaania staoprzecinkowe

- Dodawanie wykonujemy identycznie
- W przypadku mnoenia wynik musimy podzieli przez 2n
- Mnoenie liczb staoprzecinkowych przez potg 2 polega tylko na przesuwaniu bitów (bardzo proste w realizacji)

				1	0	1	1	1	0	0	0	
0	0	1	0	1	1	1	0					Podzielenie przez 2^2
2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}					

2.21 Format zmiennoprzecinkowy

- Bardziej zaawansowany sposób przedstawiania liczb
- Ustandaryzowany norm IEEE
- Dajcy pod pewnymi wzgldami wiksz dokadno

2.22 Format zmiennoprzecinkowy

Reprezentacja liczby

$$x = S \cdot M \cdot B^E$$

- S – znak (*sign*)
- M – mantysa (*mantissa*, take *fraction*)
- B – podstawa (*base*, zazwyczaj 2, rzadziej 10)
- E - wykadnik (*exponent*)

2.23 Mantysa

- Liczba odpowiadajca za uamkow cz zapisu
- Format staoprzecinkowy, zazwyczaj liczba z przedziau [1,2)

2.24 Podstawa i wykadnik

- Pozwalaj na okrelenie szerokiego zakresu

- Ze wzgldu na kodowanie, zazwyczaj podstawa to 2
- Wykadnik moe by ujemny lub dodatni.
- Wykadnik koduje si w U2, lub te wprowadza si przesunicie

2.25 Dziaania na liczbach zmiennoprzecinkowych

Dodawanie i odejmowanie

$$x_1 \pm x_2 = \left(M_1 \pm M_2 \cdot B^{E_2 - E_1}\right) \cdot B^{E_1}$$

Mnoenie i dzielenie

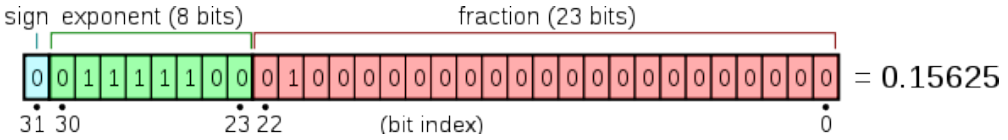
$$x_1 \cdot x_2 = (S_1 \cdot S_2) \cdot (M_1 \cdot M_2) \cdot B^{E_1 + E_2}$$

$$x_1 / x_2 = (S_1 \cdot S_2) \cdot (M_1 / M_2) \cdot B^{E_1 - E_2}$$

2.26 Dzielenie

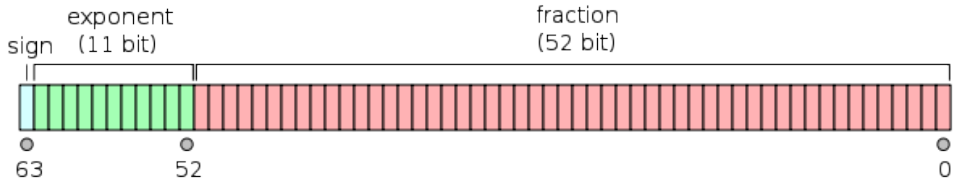
- Majc moliwo zapisu liczby ulamkowej mona sformuowa operacj dzielenia.
- Istnieje wiele algorytmów np.
  - *restoring division*
  - *non-restoring division*
  - SRT
  - algorytm Newtona-Raphsona
  - algorytm Goldschmidta
- S one ju zaimplementowane, jedno dzielenie zazwyczaj wymaga przeprowadzenia 3-4 mnoe

2.27 Wane formaty – IEEE Single precision



- 8 bitów wykadnika, wykadnik przesunity o 127 (zamiana z -126 do 127 na 1 do 244) - 24 bity mantysy, ale zawsze koduje si tylko 23 po kropce, przed kropk jest 1 - Specjalne zapisy nieskoczonoci i bdów - w NumPy - `float32`

2.28 Wane formaty – IEEE Double precision



- 11 bitów wykadnika, wykadnik przesunity o 1023 (zamiana z -1022 do 1023 na 1 do 2046) - 53 bity mantysy, ale zawsz koduje si tylko 52 po kropce, przed kropk jest 1 - Specjalne zapisy nieskoczonoci i bdów - w

NumPy - `float64`, ale w zasadzie kada liczba w Pythonie i Matlabie to double, chyba e wymusimy inaczej

2.29 Wywietlanie liczb

- Normalnie
- Notacja inynierska
  - $3700 = 3.7 \cdot 10^3$, $0.12 = 120 \cdot 10^{-3}$
- Notacja naukowa
  - `3700=3.7E3`, `0.12=1.2E-1`

3 Bdy numeryczne

3.1 Podstawowe definicje

Warto dokadna

$$y = \tilde{y} + \varepsilon$$

- $\tilde{y}$ - warto przybliona - ε - bd

3.2 Bd bezwzgldny

Warto bezwzgldna rónicy midzy rozwizaniem dokadnym i przyblionym

$$\varepsilon = |y - \tilde{y}|$$

3.3 Bd wzgldny

Stosunek bdu bezwzgldnego do wartoci bezwzgldnej rozwizania

$$\eta = \frac{|y - \tilde{y}|}{|y|} = \left|\frac{y - \tilde{y}}{y}\right| = \left|1 - \frac{\tilde{y}}{y}\right|$$

Czasami bd wzgldny wyraamy w procentach

3.4 Przykady

Pierwiastek kwadratowy ze 122

$$\begin{aligned} y &= \sqrt{122} \approx 11.04536 \\ \tilde{y} &= 11 \\ \varepsilon &= |y - \tilde{y}| = 0.04536 \\ \eta &= \frac{|y - \tilde{y}|}{|y|} = 0.00411 \end{aligned}$$

3.5 Przykady

Liczba obywateli Polski (stan na ostatni spis powszechny z 2011)

$$
\begin{aligned}
y &= 38\,538\,447 \\
\tilde{y} &= 38\,500\,000 \\
\varepsilon &= |y - \tilde{y}| = 38\,447 \\
\eta &= \frac{|y - \tilde{y}|}{|y|} = 9.97627 \cdot 10^{-4} \approx 0.001
\end{aligned}
$$

3.6 Przykady

Obliczanie staej grawitacji

$$
\begin{aligned}
y &= 6.673841 \cdot 10^{-11} \\
\tilde{y} &= 6.7 \cdot 10^{-11} \\
\varepsilon &= |y - \tilde{y}| = 2.6159 \cdot 10^{-13} \\
\eta &= \frac{|y - \tilde{y}|}{|y|} = 0.00391
\end{aligned}
$$

3.7 róda bdów

Bdy powstajce przy formuowaniu zagadnienia - Bdy pomiaru - Bdy wynikajce z przyjcia okrelonych przyblie opisu zjawisk fizycznych

Bdy powstajce przy obliczeniach - Bdy grube (pomyki) - Bdy metody (obcicia) - Bdy zaokrgle

3.8 Bdy grube

- Bd przy wpisywaniu wzoru do komputera np. `x=A/b` zamiast `x=A\b`
- Za implementacja algorytmu
- Niewaciwa kolejno wykonywania dziaa

3.9 Bdy metody (obcicia)

- Bdy obcicia s nieodcznym elementem oblicze numerycznych.
- Bd obcicia jest to bd wynikajcy z tego, e do uzyskania dokadnego rozwizania potrzebujemy wykona nieskoczenie wiele oblicze

3.10 Przykady bdów metody

Mona wykaza, e

$$
\begin{aligned}
\sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \\
&= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}
\end{aligned}
$$

Bdem odcicia bdzie

$$
\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!}
$$

3.11 Przykady bdów metody

Metoda bisekcji

[2]:
```
def bisection(f,a,b,N):
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = (a_n + b_n)/2
        f_m_n = f(m_n)
        if f(a_n)*f_m_n < 0:
            a_n = a_n
            b_n = m_n
        elif f(b_n)*f_m_n < 0:
            a_n = m_n
            b_n = b_n
    return (a_n + b_n)/2
```

Szukamy pierwiastka wielomianu $x^2 - 2$, w przedziale $[1, 2]$. Rozwizanie to $\sqrt{2}$.

[3]:
```
f = lambda x: x**2 - 2 # definicja funkcji
bisection(f,1,2,5) # 5 kroków
```

[3]:
```
1.421875
```

[4]:
```
bisection(f,1,2,10) # 10 kroków
```

[4]:
```
1.41455078125
```

[5]:
```
bisection(f,1,2,15) # 15 kroków
```

[5]:
```
1.4141998291015625
```

[6]:
```
import numpy as np
np.sqrt(2)
```

[6]:
```
1.4142135623730951
```

3.12 Bd metody - podsumowanie

- Praktycznie wszystkie metody numeryczne maj jaki bd metody
- Dobre algorytmy podaj jednak jego oszacowanie, w ten sposób wiemy jak daleko jestemy od rozwizania nawet jak przerwiemy obliczenia

3.13 Bdy zaokrgle

Kolejne nieusuwalne w peni ródo bdów, nad którym mamy mniejsz kontrol ni nad bdem metody

3.14 Zaokrglenie i cyfry znaczce

Liczba $\tilde{y} = \mathrm{rd}(y)$ jest poprawnie zaokrglona do d miejsc po przecinku, jeeli

$$\varepsilon = |y - \tilde{y}| \leq \frac{1}{2} \cdot 10^{-d}$$

k-t cyfr dziesitn liczby $\tilde{y}$ nazwiemy znaczc gdy

$$|y - \tilde{y}| \leq \frac{1}{2} \cdot 10^{-k}$$

oraz

$$|\tilde{y}| \geq 10^{-k}$$

3.15 Rzeczywiste obliczenia zmiennoprzecinkowe

$$\begin{aligned} \mathrm{fl}(x+y) &= \mathrm{rd}(x+y) \\ \mathrm{fl}(x-y) &= \mathrm{rd}(x-y) \\ \mathrm{fl}(x \cdot y) &= \mathrm{rd}(x \cdot y) \\ \mathrm{fl}(x/y) &= \mathrm{rd}(x/y) \end{aligned}$$

3.16 Liczby maszynowe

- Liczba maszynowa, to taka liczba jak mona przedstawi w komputerze. Zbiór tych liczb oznaczamy A
- Dokadno maszynow (epsilon maszynowy) – eps, ε_m, definiujemy:

$$\mathrm{eps} = \min\{x \in A \colon \mathrm{fl}(1+x) > 1,\ x > 0\}$$

  Innymi sowy, jest to najmniejsza liczba, któr moemy doda do 1, aby uzyska co wikszego od 1.

3.17 Epsilon maszynowy w rónych formatach

Zaley on od liczby bitów na cz uamkow - Single precision $\varepsilon_m = 2^{-24} \approx 5.96 \cdot 10^{-8}$ - Double precision $\varepsilon_m = 2^{-52} \approx 1.11 \cdot 10^{-16}$

Przykad

[7]:
```
a=10**(-15)
b=10**(-17)
1+a>1,1+b>1
```

[7]:
```
(True, False)
```

3.18 Maksymalny bd reprezentacji

Dla kadej liczby rzeczywistej x istnieje taka liczba ε, taka e $|\varepsilon| < \varepsilon_m$, e $\mathrm{fl}(x) = x(1+\varepsilon)$

Oznacza to, e **bd wzgldny midzy liczb rzeczywist, a jej najblisz reprezentacj zmiennoprzecinkow jest zawsze mniejszy od** ε_m

3.19 Lemat Wilkinsona

Bedy zaokrgle powstae podczas wykonywania dziaa zmiennoprzecinkowych s równowane zastpczemu zaburzeniu liczb, na których wykonujemy dziaania

$$\begin{aligned} \mathrm{fl}(x+y) &= (x+y)(1+\varepsilon_1) \\ \mathrm{fl}(x-y) &= (x-y)(1+\varepsilon_2) \\ \mathrm{fl}(x\cdot y) &= (x\cdot y)(1+\varepsilon_3) \\ \mathrm{fl}(x/y) &= (x/y)(1+\varepsilon_4) \\ |\varepsilon_i| &< \varepsilon_m \end{aligned}$$

(dla kadej pary liczb x, y zaburzenia zastpcze ε_i s inne)

3.20 Konsekwencja lematu Wilkinsona

Prawa cznoci i rozdzielnoci operacji matematycznych s ogólnie nieprawdziwe dla oblicze zmiennoprzecinkowych

3.20.1 Przykad

[8]:
```
a=np.float32(0.23371258*10**(-4))
b=np.float32(0.33678429*10**(2))
c=np.float32(-0.33677811*10**(2))
print([a,b,c])
```

```
[2.3371258e-05, 33.67843, -33.67781]
```

Chcemy obliczy `a+b+c`

3.21 Obliczenia

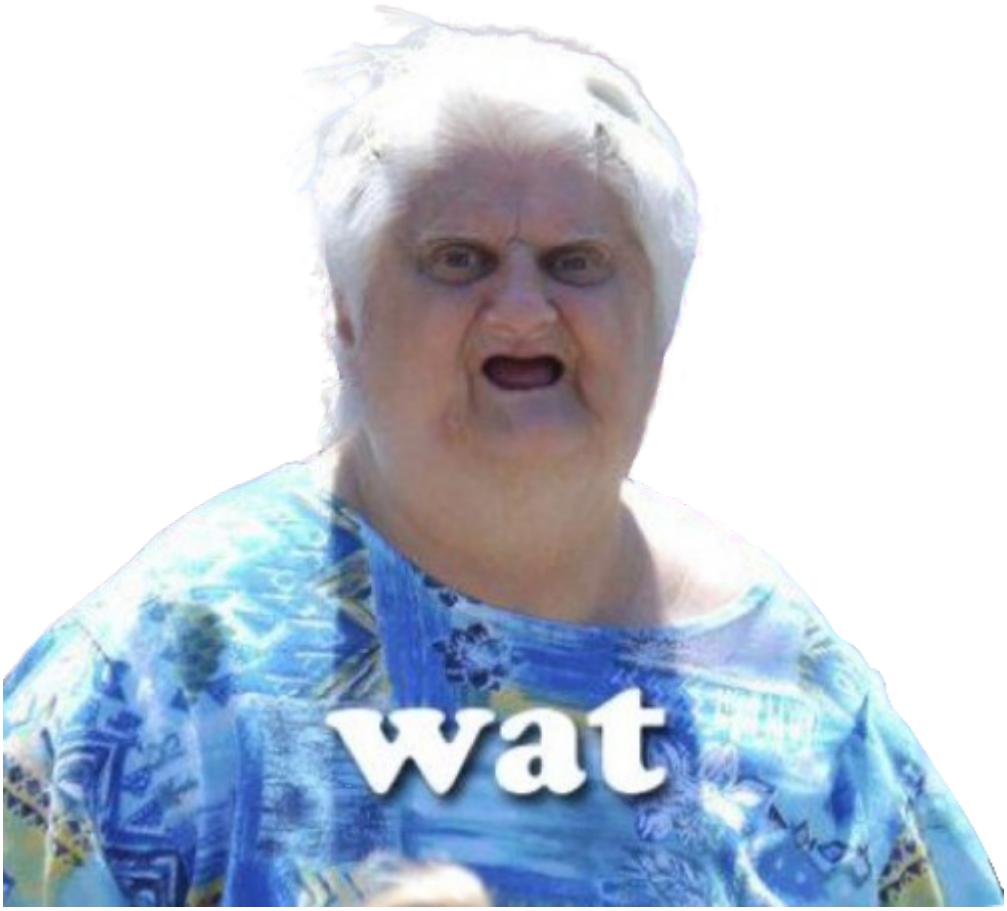
[9]:
```
## Podejcie 1
d=b+c
wynik_1=a+d
print(wynik_1)
```

```
0.0006413522
```

[10]:
```
## Podejcie 2
e=a+b
wynik_2=e+c
print(wynik_2)
```

```
0.00064086914
```

3.22 Co tu si porobio?



3.23 Konsekwencje obliczen zmiennoprzecinkowych

[11]:
```
m_a, e_a = np.frexp(a)
print(m_a,e_a)
m_b,e_b = np.frexp(b)
print(m_b,e_b)
m_c,e_c = np.frexp(c)
print(m_c,e_c)
```

```
0.7658294 -15
0.52622545 6
-0.5262158 6
```

Wykadnik `a` od wykadników `b` i `c` róni si o 21. Oznacza to, e z 23 bitów mantysy liczby `a` po sprowadzeniu do wspólnego wykadnika z `b` zostan nam tylko 2 najbardziej znaczce.

3.24 Konsekwencje cd..

Jeeli dodajemy ma liczb do duej, zawsze musimy si liczy z zaokrgleniem i to normalne. W tym przypadku jednak dwie due liczby `b` i `c` s przeciwnych znaków i bliskie co do wartoci bezwzgldnej. Wynik tego dziaania:

[12]:
```
m_d,e_d = np.frexp(d)
print(m_d,e_d)
print(wynik_2)
```

```
0.6328125 -10
0.00064086914
```

W konsekwencji dodajc `a` do `d` na zaokrgleniu stracimy jedynie 5 bitów mantysy `a`.

3.25 O ile si pomylilimy (w stosunku do dokadniejszych oblicze)

[13]:
```
a_dbl=(0.23371258*10**(-4))
b_dbl=(0.33678429*10**(2))
c_dbl=(-0.33677811*10**(2))
d_dbl=b_dbl+c_dbl
wynik_dbl=a_dbl+d_dbl
epsilon_1=np.abs((wynik_1)-wynik_dbl)
eta_1=epsilon_1/np.abs(wynik_dbl)
print("Metoda 1: Bd bezwzgldny %10.2e, Bd wzgldny %10.2e"%(epsilon_1,eta_1))
epsilon_2=np.abs((wynik_2)-wynik_dbl)
eta_2=epsilon_2/np.abs(wynik_dbl)
print("Metoda 2: Bd bezwzgldny %10.2e, Bd wzgldny %10.2e"%(epsilon_2,eta_2))
```

```
Metoda 1: Bd bezwzgldny   1.91e-08, Bd wzgldny   2.97e-05
Metoda 2: Bd bezwzgldny   5.02e-07, Bd wzgldny   7.83e-04
```

4 Analiza algorytmów numerycznych

4.1 Notacja O due

- Mówimy, e dla wielkoci zalenej od parametru np. $F(n)$ zachodzi

$$F(n) = O(G(n))$$

jeeli istnieje taka staa C, e przy n zmierzajcym do nieskoczonoci (odpowiednio duym), mamy

$$F(n) C G(n)$$

- Jeeli interesuje nas $O(c)$, gdzie c jest sta, zaleno ta ma zachodzi niezalenie od wielkoci parametru.
- Mówimy potocznie, gdy bd jest równy $O(n^2)$, e bd jest rzdu n^2

4.2 Ocena algorytmu

- Naszym celem jest obliczenie pewnej wielkoci $f(x)$, zalenej od danych wejciowych x
- W przypadku oblicze komputerowych zawsze mamy do czynienia z obliczaniem przyblionym std algorytm obliczania $f(x)$ bdziemy oznacza jako $f^*(x)$
- Dane w komputerze równie s reprezentowane w sposób zaokrglony, wic bdziemy je oznacza jako x^*

4.3 Uwarunkowanie problemu

- Mówimy, e problem $f(x)$ jest dobrze uwarunkowany, jeeli maa zmiana x powoduje ma zmian w $f(x)$
- Problem jest le uwarunkowany, jeeli maa zmiana x powoduje du zmian w $f(x)$
- Miar uwarunkowania jest staa κ (kappa), która (nieformalnie) okrela najwikszy iloraz zaburze $f(x)$ wywoanych przez najmniejsze zaburzenia x.
- Sta κ mona wyliczy tylko w niektórych probemach

4.4 Dokadno algorytmu

- Algorytm jest dokadny, jeeli
$$\frac{\|f^*(x) - f(x)\|}{\|f(x)\|} = O(\varepsilon_m)$$
- Zagwarantowanie, e algorytm jest dokadny wg tej definicji jest niezwykle trudne, zwaszcza dla le uwarunkowanych problemów

4.5 Stabilno algorytmu

Mówimy, e algorytm jest stabilny, gdy dla kadego x, zachodzi
$$\frac{\|f^*(x) - f(x^*)\|}{\|f(x^*)\|} = O(\varepsilon_m)$$
dla takich x^*, e
$$\frac{\|x - x^*\|}{\|x\|} = O(\varepsilon_m)$$
Innymi sowy **Stabilny algorytm daje prawie dobr odpowied na prawie dobre pytanie**

4.6 Stabilno wsteczna algorytmu

Algorytm jest stabilny wstecznie, jeeli dla kadego x, zachodzi
$$f^*(x) = f(x^*)$$
dla takich x^*, e
$$\frac{\|x - x^*\|}{\|x\|} = O(\varepsilon_m)$$
Innymi sowy **Stabilny wstecznie algorytm daje prawidow odpowied na prawie dobre pytanie**

4.7 Dokadno algorytmów stabilnych wstecznie przy zym uwarunkowaniu

Jeli algorytm jest stabilny wstecznie, to jego bd wzgldny pogarsza si proporcjonalnie do staej uwarunkowania tj. $O(\kappa\varepsilon_m)$