MOwNiT – lab 1

Arytmetyka komputerowa

Gabriel Cyganek

Dane techniczne

Do napisania programu wykonującego obliczenia wykorzystałem język Python. Wykonywanie programu odbywało się na systemie Windows 10 x64 na komputerze z procesorem Intel® Core™ i5-7300HQ CPU @ 2.50GHz.

Polecenie

Do wykonania otrzymałem zadanie o następującej treści:

Moduł liczby zespolonej $z = x + iy$ *może być obliczany przy pomocy wzorów:*

- $|z| = \sqrt{(x^2 + y^2)}$
- $|z| = v\left[1 + \left(\frac{w}{v}\right)^2\right]^{1/2}$
- $|z| = 2v\left[\frac{1}{4} + \left(\frac{w}{2v}\right)^2\right]^{1/2}$

gdzie $v = \max\{|x|, |y|\}$, $w = \min\{|x|, |y|\}$. *Porównać wyniki dla dużych i małych liczb x i y. Dobrać dane tak, aby widoczne były różnice pomiędzy wynikami uzyskanymi różnymi wzorami.*

Program wykonujący obliczenia

```
import math
import sys


def formula_1(x, y):
    return math.sqrt(x ** 2 + y ** 2)


def formula_2(v, w):
    return v * math.sqrt(1 + (w / v) ** 2)


def formula_3(v, w):
    return 2 * v * math.sqrt(1 / 4 + (w / (2 * v)) ** 2)


x = float(sys.argv[1])
y = float(sys.argv[2])

v = max(abs(x), abs(y))
w = min(abs(x), abs(y))

print(formula_1(x, y))
print(formula_2(v, w))
print(formula_3(v, w))
```

Liczby zmiennoprzecinkowe w Pythonie

Liczby zmiennoprzecinkowe w Pythonie są reprezentowane przez typ danych *float* o zapisie 64-bitowym o podwójnej precyzji, według standardu IEEE 754. W tym wypadku maksymalna liczba zmiennoprzecinkowa w Pythonie wynosi $1.8 * 10^{308}$, gdzie jakiekolwiek większe wartości są traktowane jako nieskończoność (inf), a najmniejsza $5.0 * 10^{-324}$, gdzie jakiekolwiek mniejsze wartości są traktowane jako zero.

Przebieg eksperymentu

Po wykonaniu pierwszej serii obliczeń, gdzie podstawiałem różne wartości pod x oraz y, gdzie $z = x + iy$ otrzymałem następujące wyniki:

x	y	$\sqrt{(x^2+y^2)}$	$v\left[1+\left(\frac{w}{v}\right)^2\right]^{1/2}$	$2v\left[\frac{1}{4}+\left(\frac{w}{2v}\right)^2\right]^{1/2}$	Wolfram Alpha
5	6	7.810249675906654	7.810249675906656	7.810249675906656	7.810249675906654**39**
5	15	15.811388300841896	15.811388300841898	15.811388300841898	15.811388300841896**65**
25	30	39.051248379533327	39.051248379533328	39.051248379533328	39.051248379533271**97**
55	101	115.004347743900054	115.004347743900052	115.004347743900052	115.004347743900534675
100	501	510.882569677220043	510.882569677222205	510.882569677222205	510.882569677220442161
501	403	642.969672690710**9**	642.96967269**0711**	642.96967269**0711**	642.969672690710973**0**
501	99	510.68777153951**9**	510.687771539519**13**	510.687771539519**13**	510.68777153951904**6**
5000001	99999999	100124921.023689**23**	100124921.023689**21**	100124921.023689**21**	100124921.023689212**19**
512341234	153453419	534828469.5670494**4**	534828469.567**0495**	534828469.567**0495**	534828469.567**0494287**
0.00212123	0.0000053	0.0021212366211481**454**	0.0021212366211481**45**	0.0021212366211481**45**	0.00212123662114814**527**

Tabela 1. *Wyniki obliczeń dla pierwszej serii danych*

Na zielono zaznaczyłem wyniki, które są bliższe wynikom uzyskanym z Wolfram Alpha, który obliczał moduły korzystając ze wzoru $\sqrt{(x^2+y^2)}$. Należy brać pod uwagę, że Wolfram Alpha korzysta z o wiele większych zasobów, więc nie jest aż tak ograniczony dostępną pamięcią jak program napisany w Pythonie wykonany na systemie Windows 10 x64, dzięki czemu otrzymujemy zdecydowanie większą liczbę precyzyjnych cyfr w wynikach i nie ma granicy wartości z góry i z dołu jak zmienna typu *float* w Pythonie. Jak widać w ***Tabeli 1***., bardziej dokładny w zdecydowanej większości zbadanych przypadków był wzór $\sqrt{(x^2+y^2)}$.

Drugą serię obliczeń wykonałem dla bardzo małych liczb:

x	y	$\sqrt{(x^2+y^2)}$	$v\left[1+\left(\frac{w}{v}\right)^2\right]^{1/2}$	$2v\left[\frac{1}{4}+\left(\frac{w}{2v}\right)^2\right]^{1/2}$	Wolfram Alpha
2131e-145	2137e-150	2.13100000010**71507**e-142	2.13100000010**7151**e-142	2.13100000010**7151**e-142	2.13100000010**7150844671** e-142
25559e-133	88223e-133	9.18507496431**0309**e-129	9.18507496431**0307**e-129	9.18507496431**0307**e-129	9.18507496431**0307685361** e-142
1e-161	1e-161	1.40579606748**0928**e-161	1.41421356237**0951**e-161	1.41421356237**0951**e-161	1.41421356237**095048801** e-161
9e-161	9e-161	1.272**6143119844308**e-160	1.272**7922061357856**e-160	1.272**7922061357856**e-160	1.272**792206135785543921** e-160
1e-162	1e-162	0	1.414213562373095e-162	1.414213562373095e-162	1.414213562373095048801 e-162
1e-320	1e-320	0	1.414e-320	1.414e-320	1.414213562373095048872 e-162

Tabela 2. *Wyniki obliczeń dla serii danych o małych wartościach*

Możemy zauważyć, że tym razem w większości przypadków dokładniejsze wyniki otrzymujemy dla wzorów $v\left[1+\left(\frac{w}{v}\right)^2\right]^{1/2}$ oraz $2v\left[\frac{1}{4}+\left(\frac{w}{2v}\right)^2\right]^{1/2}$. Dla wzoru $\sqrt{(x^2+y^2)}$ po przekroczeniu pewnej wartości zarówno dla x jak i dla y nie otrzymamy już żadnego innego wyniku poza zerem, ma to związek z tym, że obie te zmienne podniesione do kwadratu pod pierwiastkiem otrzymują wartości mniejsze od wartości granicznej dla zmiennej typu *float* w Pythonie i są dalej traktowane jako zera. Dla dwóch pozostałych wzorów taka sytuacja nie zachodzi, gdyż operacje matematyczne są przeprowadzane w inny sposób, w innej kolejności i przeprowadzając je nie narusza się dolnej wartości granicznej *float*.

Trzecią serię obliczeń przeprowadziłem dla bardzo dużych liczb:

x	y	$\sqrt{(x^2+y^2)}$	$v\left[1+\left(\frac{w}{v}\right)^2\right]^{1/2}$	$2v\left[\frac{1}{4}+\left(\frac{w}{2v}\right)^2\right]^{1/2}$	Wolfram Alpha
1.2391 e+147	1.23391 e+149	1.233972214023**071** e+149	1.233972214023**0708**e+149	1.233972214023**0708**e+149	1.233972214023**0711**e+149
1e+153	1e+153	1.414213562373**095** e+153	1.414213562373**0952**e+153	1.414213562373**0952**e+153	1.414213562373**0950**4e+153
1e+154	1e+154	inf	1.414213562373**0953**e+154	1.414213562373**0953**e+154	1.414213562373**09504**e+154
1e+307	1e+307	inf	1.41421356237**3095**e+307	1.41421356237**3095**e+307	1.41421356237**309504**e+307
1e+308	1e+308	inf	1.4142135623730951e+308	inf	1.41421356237309504e+308

Tabela 3. *Wyniki obliczeń dla serii danych o dużychh wartościach*

Dla bardzo dużych liczb zachodzi analogiczna sytuacja jak dla bardzo małych. Dla wzoru $\sqrt{(x^2+y^2)}$ w pewnym momencie dane wartości do obliczeń są zbyt duże i w wyniku przeprowadzanych operacji matematycznych osiągamy górną graniczną wartość dla zmiennej typu *float*, w wyniku czego dostajemy same nieskończoności. Dla dwóch pozostałych wzorów jesteśmy w stanie dalej uzyskiwać dosyć dokładne wyniki. Przy samej granicy górnej granicznej wartości zmiennej typu *float* zaczynamy dostawać wynik równy nieskończoności również dla wzoru $2v\left[\frac{1}{4}+\left(\frac{w}{2v}\right)^2\right]^{1/2}$.

Podsumowanie

Na podstawie uzyskanych obliczeń można wywnioskować, że w zależności od kolejności wykonywanych operacji matematycznych ich sposobu wykonywania możemy otrzymać różne wartości. Biorąc to pod uwagę, a także arytmetykę komputerową i ograniczenia reprezentacji liczb zmiennoprzecinkowych możemy napotkać na problemy w wykonywaniu tych samych obliczeń danym sposobem, podczas gdy innych dostępnych sposobów te problemy mogą nie dotyczyć, z czego można korzystać.