

Laboratorium 1, Zadanie 7

Filip Dziurdzia

06 Marzec 2023

1 Specyfikacja sprzętowa

Jednostka: MacBook Pro 2020

System operacyjny: macOS Ventura 13.0

Procesor: Apple M1

Architektura procesora: ARM

Pamięć RAM: 8GB

2 Środowisko

Zadanie rozwiązałem przy użyciu języka **Python 3.9.13** oraz dwóch bibliotek pomocniczych - **time** i **numpy**. Biblioteka **time** jest zawarta w paczce podstawowych bibliotek instalowanych wraz z językiem. Wykorzystana została ona do pomiaru długości czasu działania programu. Biblioteka **numpy** została zainstalowana na własną rękę korzystając z menedżera paczek Python Conda. Python w podstawowej wersji nie jest statycznie typizowany i posiada tylko jeden rodzaj typu zmiennej zmiennoprzecinkowej. Biblioteka pozwala nam na zaimplementowanie typów float, double oraz long double.

3 Zadany problem

Dana jest zależność rekurencyjna $3x_{k-1} - 10x_k + 3x_{k+1} = 0$. Wartości początkowe $x_0 = 1, x_1 = 1/3$. Wyznaczyć wartości x_k i x_{k+1} dla $k = 45$. Następnie korzystając z wyznaczonych wartości x_k i x_{k+1} obliczyć x_1 i x_0 , wykonując rekurencję w tył. Porównać wyznaczone wartości x_1 i x_0 z dokładnymi wartościami początkowymi 1 i $1/3$. Wykonać obliczenia dla różnej precyzji zmiennych (float, double, long double). Skomentować różnice. Co będzie, jeśli wszędzie liczbę 3 zastąpimy przez liczbę 2 lub 20, lub 30?

4 Omówienie teoretyczne

Przekształcamy równanie rekurencyjne

$$3x_{k-1} - 10x_k + 3x_{k+1} = 0 \quad (1)$$

$$x_{k+1} = \frac{10}{3}x_k - x_{k-1} \quad (2)$$

$$x_0 = 1 \quad (3)$$

$$x_1 = \frac{1}{3} \quad (4)$$

$$x_2 = \frac{10}{3} \cdot x_1 - x_0 = \frac{10}{3} \cdot \frac{1}{3} - 1 = \frac{1}{9} \quad (5)$$

$$x_3 = \frac{10}{3} \cdot x_2 - x_1 = \frac{10}{3} \cdot \frac{1}{9} - \frac{1}{3} = \frac{1}{27} \quad (6)$$

Widzimy zatem, że wzór jawny na k -ty wyraz przy zadanych $x_0 = 1$ i $x_1 = 1/3$ ma postać

$$x_k = \frac{1}{3^k} \quad (7)$$

Ta wiedza sprzyja nam się do weryfikacji poprawności wyników i potencjalnej analizy niedokładności wyników.

5 Wyniki

Wstęp

Poniżej zamieściłem wyniki programu zarówno w postaci informacji zwrotnej z terminalu, jak i tabeli. Program został wykonany dla wartości parametru $k=45$.

Wyniki programu

```
Funckja: rekurencjaPrzod
Typ: <class 'numpy.float32'>
Czas wykonania: 0.0004298686981201172
Wartość: [148609730000.0, 445829200000.0]
-----
Funckja: rekurencjaTyl
Typ: <class 'numpy.float32'>
Czas wykonania: 0.00024318695068359375
Wartość: [5.0429345e+23, 1.6809781e+23]
-----
Funckja: rekurencjaPrzod
Typ: <class 'numpy.float64'>
Czas wykonania: 5.984306335449219e-05
Wartość: [-13312.59789030802, -39937.79367092405]
```

```

-----
Funkcja: rekurencjaTyl
Typ: <class 'numpy.float64'>
Czas wykonania: 5.91278076171875e-05
Wartość: [-2389977427.261684, -796659142.4205614]
-----

Funkcja: rekurencjaPrzod
Typ: <class 'numpy.float128'>
Czas wykonania: 0.00013208389282226562
Wartość: [-12717.465660928486034, -38152.396982785458103]
-----

Funkcja: rekurencjaTyl
Typ: <class 'numpy.float128'>
Czas wykonania: 0.0001220703125
Wartość: [396709.4230040512679, 132236.47433468375597]

```

Wyniki w tabelach

Precyzja zmiennych	x_{44}	x_{45}	Czas wykonania
float	$1.5 \cdot 10^{10}$	$4.5 \cdot 10^{11}$	$4.3 \cdot 10^{-4}$ s
double	$-1.3 \cdot 10^4$	$-4.0 \cdot 10^4$	$6.0 \cdot 10^{-5}$ s
long double	$-1.3 \cdot 10^4$	$-3.8 \cdot 10^4$	$1.3 \cdot 10^{-4}$ s

Tabela 1: Wyniki dla rekurencji w przód przy parametrze $k = 45$

Precyzja zmiennych	x_0	x_1	Czas wykonania
float	$5.0 \cdot 10^{23}$	$1.7 \cdot 10^{23}$	$2.4 \cdot 10^{-4}$ s
double	$-2.4 \cdot 10^9$	$-8.0 \cdot 10^8$	$6.0 \cdot 10^{-5}$ s
long double	$-1.3 \cdot 10^4$	$-3.8 \cdot 10^4$	$1.3 \cdot 10^{-4}$ s

Tabela 2: Wyniki dla rekurencji w tył przy parametrze $k = 45$

Wnioski dla $k = 45$

Jak widzimy, wyniki otrzymane dla każdego rodzaju precyzji zmiennych kompletnie nie przypominają oczekiwanych rezultatów. Przy $k = 45$, liczba $(1/3)^{45}$ jest rzędu 10^{-22} i przepełnia nasze zmienne powodując bezsensowne rezultaty. Aby zaobserwować wyniki, z których możemy zaobserwować niedokładności w zapisie liczb zmiennoprzecinkowych, przeprowadziłem obliczenia dla parametru $k = 35$.

6 Wyniki dla $k = 35$

Poniżej zamieściłem wyniki działania programu dla parametru $k = 35$.

Wyniki programu

```
Funckja: rekurencjaPrzod
Typ: <class 'numpy.float32'>
Czas wykonania: 0.0003058910369873047
Wartość: [2516718.5, 7550155.5]
-----
Funckja: rekurencjaTyl
Typ: <class 'numpy.float32'>
Czas wykonania: 0.0001881122589111328
Wartość: [150198080000000.0, 50066027000000.0]
-----
Funckja: rekurencjaPrzod
Typ: <class 'numpy.float64'>
Czas wykonania: 4.8160552978515625e-05
Wartość: [-0.22545001423068997, -0.6763500426920701]
-----
Funckja: rekurencjaTyl
Typ: <class 'numpy.float64'>
Czas wykonania: 4.673004150390625e-05
Wartość: [1.0808332968748744, 0.36027776562495806]
-----
Funckja: rekurencjaPrzod
Typ: <class 'numpy.float128'>
Czas wykonania: 0.00010395050048828125
Wartość: [-0.21537139766852076224, -0.6461141930055624466]
-----
Funckja: rekurencjaTyl
Typ: <class 'numpy.float128'>
Czas wykonania: 9.870529174804688e-05
Wartość: [1.0000042468023278289, 0.33333474893410924186]
```

Wyniki w tabelach

Precyzja zmiennych	x_{44}	x_{45}	Czas wykonania
float	$2.5 \cdot 10^6$	$7.6 \cdot 10^6$	$3.1 \cdot 10^{-4}$ s
double	-0.23	-0.68	$4.8 \cdot 10^{-5}$ s
long double	-0.22	-0.65	$1.0 \cdot 10^{-4}$ s

Tabela 3: Wyniki dla rekurencji w przód przy parametrze $k = 35$

Precyzja zmiennych	x_0	x_1	Czas wykonania
float	$1.5 \cdot 10^{14}$	$5.0 \cdot 10^{13}$	$1.9 \cdot 10^{-4}$ s
double	1.0808333	0.3602777	$4.7 \cdot 10^{-5}$ s
long double	1.0000042	0.3333347	$9.9 \cdot 10^{-5}$ s

Tabela 4: Wyniki dla rekurencji w tył przy parametrze $k = 35$

Wnioski dla $k = 35$

Pomimo ponownego przepełnienia zmiennych podczas rekurencji w przód, dla precyzji **double** i **long double**, możemy zauważyć wartości zbliżone do początkowych dla rekurencji w tył. Błąd znacząco maleje wraz ze wzrostem precyzji i dla typu **long double** jest on rzędu 10^{-6} . Wartości dla typu **float** możemy zignorować.

7 Alternatywne przypadki

Zamiana liczby 3 w równaniu rekurencyjnym na liczbę 2, 20 lub 30 znacząco wpłynie na otrzymywane wyniki. Zakładamy takie same wartości początkowe tj. $x_0 = 1$ oraz $x_1 = 1/3$.

Liczba 2

Zamieniając liczby 3 w równaniu na liczbę 2, x_k rosłoby wraz z parametrem k . Ponieważ dokładność dla dużych liczb jest większa niż dla liczb zmierzających do 0, to nasze zmienne przepełniłyby się dopiero dla większych parametrów k .

Liczba 20

Liczba 20 sprawia, że x_k "skacze" po przybliżonym przedziale $(0.01, 1)$, dzięki czemu nie musimy się martwić o przepełnienie naszych zmiennych. Dopuszczalne są naprawdę duże wartości parametru k .

Liczba 30

Dla liczby 30, równanie zachowuje się podobnie, jak dla liczby 20, jednak zakres po którym skaczą wartości jest większy i zbudowany wokół 0, dlatego pojawiają się wartości ujemne.

8 Kod źródłowy

```
1  # Importy
2  import numpy as np
3  import time
4
5  # Stałe
6
7  # Wartości początkowe
8  x_0 = 1
9  x_1:np.longdouble = 1/3
10
11 # Współczynniki w zależności rekurencyjnej
12 a = 3
13 b = -10
14 c = 3
15 d = 0
16
17 # Wartość parametru k
18 k = 45
19
20 # Typy zmiennych
21 types = [np.single, np.double, np.longdouble]
22
23 # Funkcje
24
25 def lab1(x_k_1, x_k, k, k_max, type):
26     for type in types:
27
28         # Rekurencja w przód
29         start = time.time()
30         result1 = rekurencjaPrzod(x_k_1, x_k, k, k_max, type)
31         end = time.time()
32         print("Funkcja: ", rekurencjaPrzod.__name__)
33         print("Typ: ", type)
34         print("Czas wykonania: ", end - start)
35         print("Wartość: ", result1)
36         print("-----")
37
38         # Rekurencja w tył
39         start = time.time()
40         result2 = rekurencjaTyl(result1[0], result1[1], k_max, type)
41         end = time.time()
42         print("Funkcja: ", rekurencjaTyl.__name__)
43         print("Typ: ", type)
44         print("Czas wykonania: ", end - start)
```

```

45         print("Wartość: ", result2)
46         print("-----")
47
48     # Funkcja do rekurencji w przód
49     def rekurencjaPrzod(x_k_1, x_k, k, k_max, type) -> type[2]:
50         result = type((-b*x_k - a*x_k_1)/c)
51         if k == k_max:
52             return [x_k, result]
53         else:
54             return rekurencjaPrzod(x_k, result, k+1, k_max, type)
55
56     # Funkcja do rekurencji w tył
57     def rekurencjaTyl(x_k_1, x_k, k, type) -> type[2]:
58         result = type((-b*x_k - c*x_k_1)/a)
59         if k == 1:
60             return [result, x_k_1]
61         else:
62             return rekurencjaTyl(result, x_k_1, k-1, type)
63
64
65     # Wywołanie głównej funkcji
66     lab1(x_0, x_1, 1, k, types[1])

```