Obliczenia Naukowe - Laboratorium 2

Krzysztof Zając (przykład)

Listopad 2025

1 Zadanie 1 - Iloczyn skalarny

1.1 Opis problemu

Obliczenie iloczynu skalarnego na cztery różne sposoby, tak jak w zadaniu 1 z laboratorium 1, tym razem dla innych argumentów i porównanie wyników.

1.2 Rozwiązanie

Zaimplementowano algorytmy.

1.3 Wyniki

Tabela 1: Porównanie wyników iloczynu skalarnego

Metoda	`Float32`	`Float64`
(a) "w przód"	-0.4999443	-0.004296342739891585
(b) "w tył"	-0.4543457	-0.004296342998713953
(c) Sort (najw. $\rightarrow$ najmn.)	-0.5	-0.004296342842280865
(d) Sort (najmn. $\rightarrow$ najw.)	-0.5	-0.004296342842280865
Poprzednia wartość (bliska)	$-1.00657107 \times 10^{-11}$	

1.4 Wnioski

1. `Float32` ma zbyt małą precyzję, aby zauważyć różnicę.
2. Wartości `Float64` drastycznie się zmieniły, co sugeruje, że ten iloczyn skalarny jest wrażliwy na małe zmiany danych.

2 Zadanie 2 - Błąd anulowania przy liczeniu granicy

2.1 Opis problemu

Analiza numeryczna funkcji $f(x) = e^x \ln(1 + e^{-x})$. Obliczenie granicy $lim_{x\to\infty} f(x)$ i porównanie jej z wykresem wygenerowanym komputerowo.

2.2 Rozwiązanie

Granicę obliczono analitycznie. Dla $t = e^{-x}$, gdy $x \to \infty$, to $t \to 0^+$.

$$\lim_{x\to\infty} e^x \ln(1 + e^{-x}) = \lim_{t\to 0^+} \frac{\ln(1+t)}{t}$$

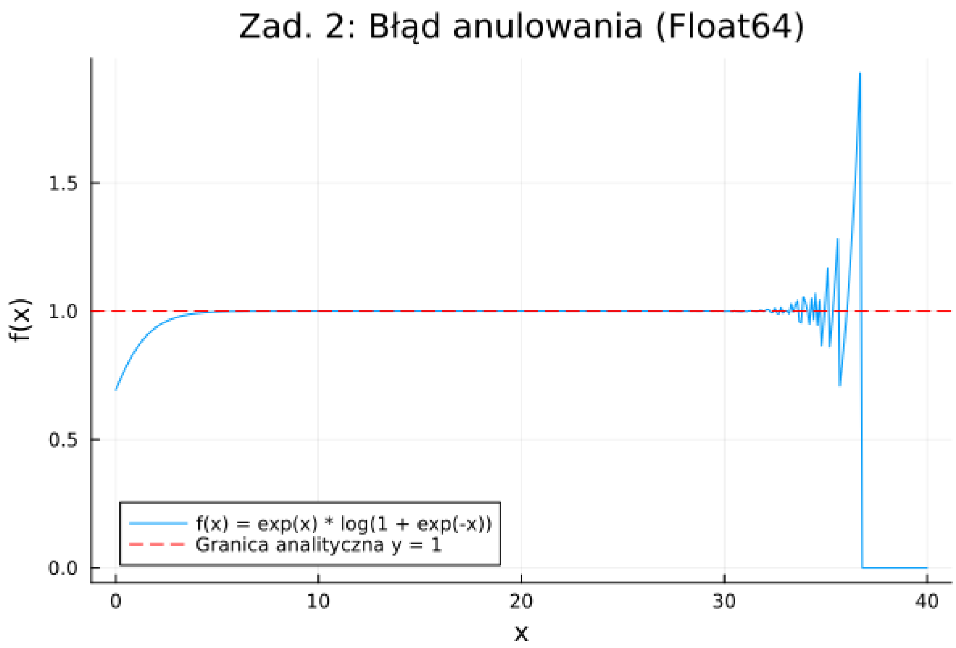
Jest to definicja pochodnej $\ln(u)$ w $u = 1$, lub (z reguły de l'Hospitala):

$$\lim_{t \to 0^+} \frac{\frac{1}{1+t}}{1} = 1$$

Funkcję zwizualizowano w Julii, obserwując jej zachowanie dla rosnących x.

2.3 Wyniki

Analityczna granica funkcji wynosi 1. Wykres numeryczny (w arytmetyce Float64) pokazuje, że funkcja poprawnie zbiega do 1, jednak od $x \approx 32$ zachowuje się niestabilnie, a dla $x \approx 37$ spada do 0.



Rysunek 1: Wykres f(x). Widoczny spadek do 0 dla x > 36.

2.4 Wnioski

Nagły spadek do zera jest błędem numerycznym. Dla $x > 36.7$, e^{-x} staje się mniejsze niż $\approx 10^{-16}$ (epsilon maszynowy). W rezultacie $\mathrm{fl}(1 + e^{-x})$ jest zaokrąglane do 1.0. Następnie ln(1.0) daje 0, a całe wyrażenie $f(x)$ jest obliczane jako 0. Jest to skutek błędu zaokrąglenia (underflow) i anulowania.

3 Zadanie 3 - Uwarunkowanie układów równań

3.1 Opis problemu

Rozwiązanie układu $Ax = b$, gdzie $x_{dokl} = (1, ..., 1)^T$ a $b = Ax$. Porównanie stabilności dwóch metod: eliminacji Gaussa (`A\b`) oraz jawnego użycia macierzy odwrotnej (`inv(A)*b`). Analiza dla macierzy Hilberta H_n (źle uwarunkowana) i losowych macierzy R_n o kontrolowanym wskaźniku uwarunkowania c.

3.2 Rozwiązanie

Zaimplementowano testy dla H_n przy $n \in [2, 20]$ oraz dla R_n przy $n \in \{5, 10, 20\}$ i $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$. Mierzono błąd względny $||x_{obl} - x_{dokl}||_2 / ||x_{dokl}||_2$.

3.3 Wyniki

(a) Macierze Hilberta H_n

n	cond(A)	rank	Błąd (A\b)	Błąd (inv(A)*b)
2	1.928e+01	2	5.66105e-16	1.40433e-15
4	1.551e+04	4	4.13741e-14	0.00000e+00
6	1.495e+07	6	2.61891e-10	2.01638e-10
8	1.526e+10	8	6.12409e-08	3.07748e-07
10	1.602e+13	10	8.67039e-05	2.50149e-04
12	1.752e+16	11	1.33962e-01	2.58994e-01
14	6.201e+17	11	1.45541e+00	8.71499e+00
16	7.046e+17	12	5.41552e+01	2.98488e+01
18	2.248e+18	12	1.02576e+01	2.47621e+01
20	1.148e+18	13	1.08318e+02	1.14344e+02

(b) Macierze losowe R_n o zadanym 'c'

n	c (zadane)	c (realne)	rank	Błąd (A\b)	Błąd (inv(A)*b)
5	1.0e+00	1.000e+00	5	1.98603e-16	4.96507e-17
5	1.0e+01	1.000e+01	5	2.53170e-16	5.25453e-16
5	1.0e+03	1.000e+03	5	5.16917e-14	2.43335e-14
5	1.0e+07	1.000e+07	5	1.68858e-10	1.56188e-10
5	1.0e+12	1.000e+12	5	4.62578e-05	1.46009e-05
5	1.0e+16	1.612e+16	4	3.28100e-01	4.77624e-01
10	1.0e+00	1.000e+00	10	3.25581e-16	2.60370e-16
10	1.0e+01	1.000e+01	10	3.58036e-16	6.98647e-16
10	1.0e+03	1.000e+03	10	2.16665e-14	3.69209e-14
10	1.0e+07	1.000e+07	10	3.76855e-10	2.05499e-10
10	1.0e+12	1.000e+12	10	2.88106e-05	4.76602e-05
10	1.0e+16	1.547e+16	9	3.91277e-01	1.22035e-01
20	1.0e+00	1.000e+00	20	5.03899e-16	5.45028e-16
20	1.0e+01	1.000e+01	20	6.30974e-16	5.17178e-16
20	1.0e+03	1.000e+03	20	6.47415e-14	4.13263e-14s
20	1.0e+07	1.000e+07	20	5.99724e-10	3.20098e-10
20	1.0e+12	1.000e+12	20	4.66512e-05	1.60382e-05s
20	1.0e+16	8.942e+15	19	3.42596e-01	6.00913e-01

Dla macierzy Hilberta (a) wskaźnik uwarunkowania 'cond(A)' rośnie wykładniczo, co powoduje szybki wzrost błędu. Już dla $n = 12$ wskaźnik osiąga $\approx 1.7 \times 10^{16}$, czyli granicę precyzji maszynowej. Błąd względny obu metod skacze wtedy do $\approx 0.1 - 0.2$. Dla $n \geq 14$ błąd przekracza 1.0, co oznacza całkowitą utratę dokładności. W przypadku macierzy losowych (b) błąd rośnie stabilnie wraz z zadanym wskaźnikiem 'c'. Gdy 'c' osiąga 10^{16}, macierze również tracą rząd, a błąd wzrasta do $\approx 0.1 - 0.6$. W obu eksperymentach metody `A\b` oraz `inv(A)*b` dają wyniki o bardzo zbliżonym rzędzie wielkości błędu.

3.4 Wnioski

Problem rozwiązywania $Ax = b$ jest źle uwarunkowany, gdy 'cond(A)' jest duże. Błąd względny rozwiązania jest w przybliżeniu ograniczony przez $c \cdot \epsilon_{mach}$. Metoda `A\b` jest numerycznie stabilniejsza. Jawne obliczanie macierzy odwrotnej `inv(A)` jest kosztowniejsze i wprowadza dodatkowe błędy zaokrągleń, pogarszając wynik.

4 Zadanie 4 - Wielomian Wilkinsona

4.1 Opis problemu

Analiza uwarunkowania problemu znajdowania zer wielomianu. (a) Obliczenie zer z_k wielomianu $P(x)$ (postać naturalna $p(x) = \prod_{k=1}^{20}(x-k)$) i porównanie z k. (b) Analiza wpływu współczynnika a_{19} (z -210 na $-210 - 2^{-23}$) na zera.

4.2 Rozwiązanie

Użyto funkcji `roots()` z pakietu `Polynomials`. (a) Obliczono $z_k = \text{roots}(P)$ i błędy $|z_k - k|$. (b) Zmieniono współczynnik a_{19} i ponownie obliczono zera.

4.3 Wyniki

(a) Obliczone zera z_k znacznie różnią się od dokładnych $k \in \{1, \ldots, 20\}$. Pierwiastki $k \leq 8$ są wyznaczone poprawnie. Dla $k \geq 9$ błędy gwałtownie rosną. Pierwiastki $k \in [10, 18]$ są obliczone jako liczby zespolone o znaczących częściach urojonych (np. $z_{14}, z_{15} \approx 13.99 \pm 2.5i$).

(b) Niewielka zamiana a_{19} ($\delta \approx 1.19 \times 10^{-7}$) spowodowała drastyczne zmiany. Już z_{10} i z_{11} stały się zespolone. Błędy dla pierwiastków $k \geq 12$ są rzędu 2-3 (np. $z_{15} \approx 17.6 \pm 2.8i$).

4.4 Wnioski

(a) Problem znajdowania zer wielomianu na podstawie współczynników w bazie naturalnej jest źle uwarunkowany. Błędy reprezentacji współczynników w Float64

(b) są silnie wzmacniane, prowadząc do całkowicie błędnych wyników dla większych pierwiastków. Forma iloczynowa $p(x)$ jest dobrze uwarunkowana; postać naturalna $P(x)$ - nie.

5 Zadanie 5 - Model logistyczny

5.1 Opis problemu

Badanie wrażliwości modelu logistycznego $p_{n+1} := p_n + rp_n(1-p_n)$ dla $r = 3$ i $p_0 = 0.01$. Porównanie trajektorii (1) w arytmetyce Float32 z obcięciem po 10 iteracjach oraz (2) w arytmetyce Float32 vs Float64.

5.2 Rozwiązanie

Przeprowadzono symulacje iteracyjne dla 40 kroków zgodnie ze scenariuszami.

5.3 Wyniki

1. **Float32 vs Float32 z obcięciem**: Obie trajektorie są identyczne do $n = 10$. Po obcięciu p_{10} (z $\approx 0.7224...$ do 0.722), trajektorie natychmiast się rozbiegają. W $n = 40$ wartość niezakłócona to ≈ 0.654, a zakłócona ≈ 0.638

2. **Float32 vs Float64**: Trajektorie są zgodne przez ok. 20 iteracji. Następnie błędy zaokrąglenia Float32 kumulują się i trajektorie rozchodzą się. W $n = 40$ Float64 daje≈ 0.64459, a Float32 ≈ 0.65415.

```
(1) Float32 vs Float32 z obcięciem p_10 = 0.722
n    | p_n (F32)       | p_n (F32 Obcięte)
---------------------------------------
0    | 0.01000000      | 0.01000000
10   | 0.72293061      | 0.72293061
11   | 1.32383645      | 1.32414794
40   | 0.25860548      | 1.09356797
```

```
(2) Float32 vs Float64
n    | p_n (F32)        | p_n (F64)
----------------------------------------
0    | 0.01000000       | 0.01000000
10   | 0.72293061       | 0.72291298
20   | 0.57990360       | 0.59788019
30   | 0.75292093       | 0.28965945
40   | 0.25860548       | 0.02494137
```

5.4 Wnioski

Model logistyczny dla $r = 3$ (w obszarze zachowań chaotycznych) jest wysoce wrażliwy na warunki początkowe i błędy zaokrągleń. Minimalna zmiana (obcięcie, różna precyzja) jest wykładniczo wzmacniana w kolejnych iteracjach, prowadząc do zupełnie różnych trajektorii w długim okresie.

6 Zadanie 6 - Równanie rekurencyjne $x_{n+1} = x_n^2 + c$

6.1 Opis problemu

Obserwacja zachowania ciągów generowanych przez $x_{n+1} := x_n^2 + c$ dla 7 różnych par (c, x_0) w arytmetyce Float64 (40 iteracji).

6.2 Rozwiązanie

Zaimplementowano 7 pętli iteracyjnych, zapisując generowane ciągi.

6.3 Wyniki

Zaobserwowano następujące zachowania:

1. $c = -2, x_0 = 1$: Ciąg zbiega do punktu stałego $x = -1$ (w 1 iteracji). ($x_1 = 1^2 - 2 = -1$, $x_2 = (-1)^2 - 2 = -1$).
2. $c = -2, x_0 = 2$: Ciąg pozostaje w punkcie stałym $x = 2$. ($x_1 = 2^2 - 2 = 2$).
3. $c = -2, x_0 = 1.99...9$: Ciąg wykazuje zachowanie chaotyczne. Wartości oscylują w sposób aperiodyczny w przedziale $[-2, 2]$.
4. $c = -1, x_0 = 1$: Ciąg wchodzi w cykl (0, -1) po 1 iteracji. ($x_1 = 0, x_2 = -1, x_3 = 0, \dots$).
5. $c = -1, x_0 = -1$: Ciąg wchodzi w cykl (-1, 0) natychmiast. ($x_1 = 0, x_2 = -1, x_3 = 0, \dots$).
6. $c = -1, x_0 = 0.75$: Ciąg asymptotycznie zbiega do cyklu (0, -1).
7. $c = -1, x_0 = 0.25$: Ciąg asymptotycznie zbiega do cyklu (0, -1).

6.4 Wnioski

Badane równanie rekurencyjne (związane ze zbiorem Mandelbrota) wykazuje dużą dynamikę. W zależności od parametrów c i x_0, trajektorie mogą zbiegać do punktów stałych, zbiegać do cykli lub wykazywać zachowanie chaotyczne (w przypadku 3).