Sprawozdanie z Listy nr 2

Sara Żyndul
279686

Spis treści

1 Zadanie 1

1.1 Opis

Zadanie polega na powtórzeniu eksperymentu z zadania 5 z listy 1 (wykorzystaniu czterech algorytmów sumowania do obliczenia iloczynu skalarnego dwóch wektorów) przy wprowadzeniu niewielkich zmian w wektorze x. Wektory x, y z poprzedniej listy:

$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$

$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$.

Zmodyfikowany wektor x':

$x' = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$

Rozwiązanie znajduje się w pliku `task1.jl`.

1.2 Wyniki

Algorytm	Float32 (oryginalny x)	Float32 (zmodyfikowany x')
a	-0.4999442994594573975	-0.4999442994594573975
b	-0.4543457031250000000	-0.4543457031250000000
c	-0.5000000000000000000	-0.5000000000000000000
d	-0.5000000000000000000	-0.5000000000000000000

Algorytm	Float64 (oryginalny x)	Float64 (zmodyfikowany x')
a	$1.025188136829667182 \times 10^{-10}$	$-4.296342739891585369 \times 10^{-3}$
b	$-1.564330887049436569 \times 10^{-10}$	$-4.296342998713953421 \times 10^{-3}$
c	0.000000000000000000	$-4.296342842280864716 \times 10^{-3}$
d	0.000000000000000000	$-4.296342842280864716 \times 10^{-3}$

1.3 Wnioski

`Float32` w tym przykładzie „zatopił" perturbację zaokrągleniami — wynik nie zmienił się widocznie. `Float64` był wystarczająco precyzyjny, by wykryć różnicę i wynik przesunął się o 4.3×10^{-3}.

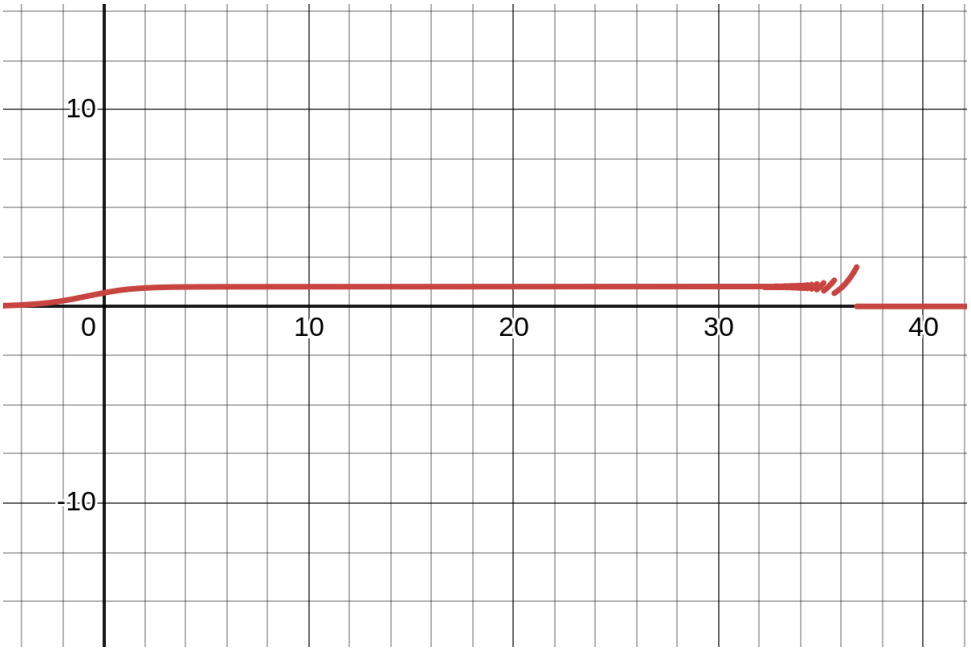
Drobna zmiana danych znacząco zmieniła wynik iloczynu skalarnego $x' \cdot y$ względem $x \cdot y$. Możemy więc wywnioskować, że zadanie obliczenia iloczynu skalarnego dla tych danych jest źle uwarunkowane.

2 Zadanie 2

2.1 Opis

Zadanie polega na wykorzystaniu co najmniej dwóch programów do wizualizacji do narysowania funkcji $f(x) = e^x \ln(1 + e^{-x})$ oraz na obliczeniu granicy funkcji przy $x \to \infty$.

2.2 Rozwiązanie



Rysunek 1: Wizualizacja funkcji f z użyciem kalkulatora graficznego Desmos

Rysunek 2: Wizualizacja funkcji f z użyciem kalkulatora graficznego Geogebra

Obliczenie granicy funkcji f:

$$\lim_{x\to\infty} e^x \ln(1+e^{-x}) = \lim_{x\to\infty} \frac{\ln(1+e^{-x})}{e^{-x}} \overset{\text{(forma } 0/0)}{=\!=\!=} \overset{\text{l'Hopital}}{=\!=\!=} \lim_{x\to\infty} \frac{\frac{d}{dx}\ln(1+e^{-x})}{\frac{d}{dx}e^{-x}}$$

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

2.3 Wnioski

Obserwacja: na kalkulatorach graficznych funkcja wygląda najpierw jak zbliżająca się do 1, potem pojawiają się oscylacje, a w końcu wykres „spada" do 0. Dla dużych x wartości e^x stają się bardzo duże (możliwe przepełnienie), a e^{-x} — bardzo małe (możliwe podpełnienie). Gdy e^{-x} podpełnia do 0, $\ln(1+e^{-x})$ przyjmuje wartość 0 i mnożenie przez (zawodny) e^x może dać 0 lub NaN/Inf — stąd skoki i „oscylogramy". Dodatkowo rzadkie próbkowanie punktów i łączenie ich liniami na wykresie potęguje artefakty (widoczne oscylacje).

3 Zadanie 3

3.1 Opis

Zadanie polega na rozwiązaniu układów $Ax = b$ dla dwóch rodzin macierzy: Hilberta H_n dla rosnącego n oraz losowych macierzy R_n wygenerowanych z zadanym wskaźnikiem uwarunkowania c. Dla każdej macierzy rozwiązano układ dwoma metodami: `A\b` oraz `inv(A)*b`. Porównano otrzymane wektory z rozwiązaniem dokładnym, licząc względne błędy rel_2, oraz sprawdzono $\text{cond}(A)$ i $\text{rank}(A)$.

3.2 Wyniki

Tabela 1: Macierze Hilberta

n	$\text{cond}(A)$	$\text{rank}(A)$	rel_2(`A\b`)	rel_2(`inv(A)*b`)
2	1.928×10^{1}	2	5.661×10^{-16}	1.404×10^{-15}
3	5.241×10^{2}	3	8.023×10^{-15}	0
4	1.551×10^{4}	4	4.137×10^{-14}	0
5	4.766×10^{5}	5	1.683×10^{-12}	3.354×10^{-12}
6	1.495×10^{7}	6	2.619×10^{-10}	2.016×10^{-10}
7	4.754×10^{8}	7	1.261×10^{-8}	4.713×10^{-9}
8	1.526×10^{10}	8	6.124×10^{-8}	3.077×10^{-7}
9	4.932×10^{11}	9	3.875×10^{-6}	4.541×10^{-6}
10	1.602×10^{13}	10	8.670×10^{-5}	2.501×10^{-4}
11	5.223×10^{14}	10	1.583×10^{-4}	7.618×10^{-3}
12	1.761×10^{16}	11	1.340×10^{-1}	2.590×10^{-1}
13	3.191×10^{18}	11	1.104×10^{-1}	5.331
14	9.276×10^{17}	11	1.455	8.715
15	3.676×10^{17}	12	4.697	7.345

n — rozmiar macierzy Hilberta H_n. $\text{cond}(A)$ — zmierzony wskaźnik uwarunkowania w normie 2. $\text{rank}(A)$ — rząd (liczba liniowo niezależnych kolumn); wartości mniejsze niż n wskazują na utratę rzędu numerycznego. rel_2(`A\b`) i rel_2(`inv(A)*b`) — względne błędy w normie 2 dla dwóch metod rozwiązania (kolejno: `A\b`, `inv(A)*b`).

Interpretacja: rosnące $\text{cond}(A)$ koreluje z narastaniem błędu; `A\b` zwykle daje mniejsze błędy niż `inv(A)*b`.

Tabela 2: Macierze losowe

n	c	cond(A)	rank(A)	rel$_2$(`A\b`)	rel$_2$(`inv(A)*b`)
5	1.0×10^{0}	1.000×10^{0}	5	2.483×10^{-16}	2.627×10^{-16}
5	1.0×10^{1}	1.000×10^{1}	5	9.930×10^{-17}	1.404×10^{-16}
5	1.0×10^{3}	1.000×10^{3}	5	2.693×10^{-14}	2.883×10^{-14}
5	1.0×10^{7}	1.000×10^{7}	5	2.983×10^{-10}	2.855×10^{-10}
5	1.0×10^{12}	1.000×10^{12}	5	4.297×10^{-6}	5.327×10^{-6}
5	1.0×10^{16}	1.633×10^{16}	4	1.036×10^{-1}	9.607×10^{-2}
10	1.0×10^{0}	1.000×10^{0}	10	2.107×10^{-16}	2.979×10^{-16}
10	1.0×10^{1}	1.000×10^{1}	10	2.107×10^{-16}	2.432×10^{-16}
10	1.0×10^{3}	1.000×10^{3}	10	2.776×10^{-15}	6.723×10^{-15}
10	1.0×10^{7}	1.000×10^{7}	10	1.723×10^{-10}	9.155×10^{-11}
10	1.0×10^{12}	1.000×10^{12}	10	3.408×10^{-6}	5.117×10^{-6}
10	1.0×10^{16}	1.220×10^{16}	9	2.678×10^{-1}	1.065×10^{-1}
20	1.0×10^{0}	1.000×10^{0}	20	5.166×10^{-16}	3.846×10^{-16}
20	1.0×10^{1}	1.000×10^{1}	20	5.601×10^{-16}	5.027×10^{-16}
20	1.0×10^{3}	1.000×10^{3}	20	4.817×10^{-15}	4.037×10^{-15}
20	1.0×10^{7}	1.000×10^{7}	20	2.362×10^{-10}	2.207×10^{-10}
20	1.0×10^{12}	9.999×10^{11}	20	3.343×10^{-7}	5.245×10^{-6}
20	1.0×10^{16}	6.226×10^{15}	19	4.816×10^{-2}	5.241×10^{-2}

n — rozmiar macierzy losowej. c — zadany współczynnik uwarunkowania użyty przy generowaniu macierzy. cond(A) — zmierzony wskaźnik uwarunkowania macierzy (może różnić się od żądanego z powodu ograniczeń precyzji). rank(A) — rząd macierzy (sprawdza, czy następuje utrata rzędu numerycznego przy bardzo dużym c). rel$_2$(`A\b`) i rel$_2$(`inv(A)*b`) — względne błędy w normie 2 dla dwóch metod rozwiązania (kolejno: `A\b`, `inv(A)*b`).

Interpretacja: rosnące c prowadzi do wzrostu cond(A) i błędów; przy ekstremalnych c może wystąpić spadek rzędu i znaczne pogorszenie dokładności.

3.3 Wnioski

1. Dla macierzy Hilberta błąd rośnie bardzo szybko wraz ze wzrostem n — problem staje się silnie źle uwarunkowany (w tabeli już od $n \approx 10$ widać wyraźny wzrost błędów).

2. `A\b` daje zwykle mniejsze błędy niż `inv(A)*b` — odwracanie macierzy jest mniej stabilne numerycznie.

3. Wzrost cond(A) koreluje z rosnącym błędem; ekstremalnie duże wartości cond prowadzą do błędów rzędu jedności lub większych.

4. Spadek rzędu numerycznego (rank(A) $< n$) wiąże się z gwałtownym pogorszeniem jakości rozwiązania (przykład: c bardzo duże $\rightarrow$ rank spada $\rightarrow$ duże relacje błędów).

5. W praktyce: zawsze sprawdzać cond(A) i rank(A); używać `A\b` zamiast `inv(A)*b`; dla źle uwarunkowanych macierzy rozważyć regularyzację lub arytmetykę o większej precyzji.

6. Dla cond(A) $> 10^{12}$ wyniki mogą nie być wiarygodne bez dodatkowych zabiegów.

4 Zadanie 4

4.1 Opis

Zadanie: obliczyć miejsca zerowe wielomianu Wilkinsona w postaci naturalnej $P(x)$ za pomocą funkcji `roots` z pakietu `Polynomials` z Julii. Porównać otrzymane pierwiastki z_k z pierwiastkami „oczekiwanymi” $k = 1, \ldots, 20$ przez obliczenie wartości $|P(z_k)|, |p(z_k)|$ (ewaluacja przez iloczyn $(x-1)\cdots(x-20)$) oraz $|z_k - k|$. Potem powtórzenie eksperymentu po maleńkiej perturbacji współczynnika -210 o -2^{-23}.

Rozwiązanie znajduje się w pliku `task4.jl`.

4.2 Wyniki

Opis kolumn:

- k — wartość oczekiwanego pierwiastka (1..20).
- z_k / z'_k — wartość znalezionego pierwiastka
- $|P(z_k)|$ — wartość modułu wielomianu w postaci naturalnej.
- $|p(z_k)|$ — wartość modułu ewaluacji przez iloczyn $(x-1)\cdots(x-20)$.
- $|z_k - k|$ — przesunięcie względem oczekiwanego pierwiastka.

Bez perturbacji

Tabela 3: Wyniki bez perturbacji: pierwiastki są bliskie wartościom właściwym, ale można zauważyć duże błędy przy obliczaniu wartości funkcji $|P(z_k)|$ i $|p(z_k)|$. Sugeruje to, że problem wyznaczania pierwiastków tego wielomianu jest źle uwarunkowany.

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	1.000000000000	2.332×10^{4}	2.331×10^{4}	1.916×10^{-13}
2	2.000000000011	6.461×10^{4}	7.316×10^{4}	1.143×10^{-11}
3	2.999999999817	1.885×10^{4}	1.303×10^{5}	1.832×10^{-10}
4	3.999999983819	2.636×10^{6}	2.031×10^{6}	1.618×10^{-8}
5	5.000000688671	2.371×10^{7}	2.161×10^{7}	6.887×10^{-7}
6	5.999988371602	1.264×10^{8}	1.217×10^{8}	1.163×10^{-5}
7	7.000112910766	5.230×10^{8}	5.062×10^{8}	1.129×10^{-4}
8	7.999279406282	1.798×10^{9}	1.740×10^{9}	7.206×10^{-4}
9	9.003273831141	5.122×10^{9}	5.264×10^{9}	3.274×10^{-3}
10	9.989265687778	1.416×10^{10}	1.415×10^{10}	1.073×10^{-2}
11	11.027997558570	3.586×10^{10}	3.693×10^{10}	2.800×10^{-2}
12	11.948273958400	8.511×10^{10}	8.162×10^{10}	5.173×10^{-2}
13	13.082031971970	2.214×10^{11}	2.044×10^{11}	8.203×10^{-2}
14	13.906800565193	3.812×10^{11}	3.852×10^{11}	9.320×10^{-2}
15	15.081439299377	8.809×10^{11}	9.126×10^{11}	8.144×10^{-2}
16	15.942404318674	1.675×10^{12}	1.675×10^{12}	5.760×10^{-2}
17	17.026861831476	3.307×10^{12}	3.512×10^{12}	2.686×10^{-2}
18	17.990484623391	6.166×10^{12}	6.644×10^{12}	9.515×10^{-3}
19	19.001981084996	1.407×10^{13}	1.275×10^{13}	1.981×10^{-3}
20	19.999803908064	3.285×10^{13}	2.384×10^{13}	1.961×10^{-4}

Po perturbacji $(-210 \to -210 - 2^{-23})$

Tabela 4: Po bardzo małej perturbacji współczynnika część pierwiastków przesuwa się znacznie; pojawiają się pary pierwiastków zespolonych sprzężonych (dla wyższych indeksów), a przesunięcia $|z'_k - k|$ mogą być rzędu jednostek.

k	z'_k	$\lvert P(z'_k)\rvert$	$\lvert p(z'_k)\rvert$	$\lvert z'_k - k\rvert$
1	1.000000000000	2.169×10^{3}	2.377×10^{3}	1.954×10^{-14}
2	1.999999999999	2.995×10^{4}	9.132×10^{3}	1.426×10^{-12}
3	3.000000000105	2.390×10^{5}	7.476×10^{4}	1.051×10^{-10}
4	3.999999995007	9.393×10^{5}	6.269×10^{5}	4.993×10^{-9}
5	5.000000034713	7.449×10^{6}	1.089×10^{6}	3.471×10^{-8}
6	6.000005852511	1.469×10^{7}	6.123×10^{7}	5.853×10^{-6}
7	6.999704466217	5.818×10^{7}	1.325×10^{9}	2.955×10^{-4}
8	8.007226654065	1.395×10^{8}	1.738×10^{10}	7.227×10^{-3}
9	8.917396943382	2.460×10^{8}	1.349×10^{11}	8.260×10^{-2}
10	$10.095290344779 - 0.64327709im$	2.291×10^{9}	1.482×10^{12}	6.503×10^{-1}
11	$10.095290344779 + 0.64327709im$	2.291×10^{9}	1.482×10^{12}	1.110×10^{0}
12	$11.793588728372 - 1.65225355im$	2.078×10^{10}	3.294×10^{13}	1.665×10^{0}
13	$11.793588728372 + 1.65225355im$	2.078×10^{10}	3.294×10^{13}	2.046×10^{0}
14	$13.992330537348 - 2.51881964im$	9.391×10^{10}	9.545×10^{14}	2.519×10^{0}
15	$13.992330537348 + 2.51881964im$	9.391×10^{10}	9.545×10^{14}	2.713×10^{0}
16	$16.730730080370 - 2.81262730im$	9.592×10^{11}	2.742×10^{16}	2.906×10^{0}
17	$16.730730080370 + 2.81262730im$	9.592×10^{11}	2.742×10^{16}	2.825×10^{0}
18	$19.502438958684 - 1.94033202im$	5.050×10^{12}	4.252×10^{17}	2.454×10^{0}
19	$19.502438958684 + 1.94033202im$	5.050×10^{12}	4.252×10^{17}	2.004×10^{0}
20	$20.846908874105 + 0.00000000im$	4.859×10^{12}	1.374×10^{18}	8.469×10^{-1}

4.3 Wnioski

1. Wielomian Wilkinsona jest bardzo źle uwarunkowany — małe zmiany współczynników powodują duże przesunięcia pierwiastków.

2. Przed perturbacją pierwiastki były bliskie wartościom całkowitym, ale odchyłki rosły wraz z k.

3. Po perturbacji: część pierwiastków rozszczepiła się na pary zespolone sprzężone — to typowy efekt niestabilności: realne pierwiastki zamieniają się lokalnie w pary zespolone.

4. Różnice $|P(z_k)|$ vs $|p(z_k)|$ pokazują, że ewaluacja w postaci współczynnikowej i iloczynowej ma różną kondycję numeryczną — wartości dużych modułów świadczą o utracie dokładności przy ewaluacji.

5 Zadanie 5

5.1 Opis

Zadanie: porównać trajektorie rekurencji wzrostu populacji $p_{n+1} = p_n + rp_n(1 - p_n)$ dla $p_0 = 0.01,\ r = 3$ w trzech wariantach:

- `Float32` (bez modyfikacji),
- `Float32` z jednorazowym **obcięciem** po 10. iteracji (odrzucenie cyfr po trzecim miejscu po przecinku),
- `Float64`.

Wykonano 40 iteracji i zapisano wartość (p_n) dla każdej iteracji.

Rozwiązanie znajduje się w pliku `task5.jl`.

5.2 Wyniki

Tabela 5: Kolumny pokazują wartości p_n w kolejnych iteracjach dla trzech wariantów obliczeń. Wersja z obcięciem została ucięta raz po 10. iteracji i kontynuowano iteracje od tej uciętej wartości.

n	`Float32`	`Float32` z obcięciem	`Float64`
1	$3.970000147820 \times 10^{-2}$	$3.970000147820 \times 10^{-2}$	$3.970000000000 \times 10^{-2}$
2	$1.540717333555 \times 10^{-1}$	$1.540717333555 \times 10^{-1}$	$1.540717300000 \times 10^{-1}$
3	$5.450726151466 \times 10^{-1}$	$5.450726151466 \times 10^{-1}$	$5.450726260444 \times 10^{-1}$
4	$1.288978099823 \times 10^{0}$	$1.288978099823 \times 10^{0}$	$1.288978001189 \times 10^{0}$
5	$1.715188026428 \times 10^{-1}$	$1.715188026428 \times 10^{-1}$	$1.715191421092 \times 10^{-1}$
6	$5.978190898895 \times 10^{-1}$	$5.978190898895 \times 10^{-1}$	$5.978201201071 \times 10^{-1}$
7	$1.319113373756 \times 10^{0}$	$1.319113373756 \times 10^{0}$	$1.319113792414 \times 10^{0}$
8	$5.627322196960 \times 10^{-2}$	$5.627322196960 \times 10^{-2}$	$5.627157764626 \times 10^{-2}$
9	$2.155928611755 \times 10^{-1}$	$2.155928611755 \times 10^{-1}$	$2.155868392326 \times 10^{-1}$
10	$7.229306101799 \times 10^{-1}$	$7.220000028610 \times 10^{-1}$	$7.229143011796 \times 10^{-1}$
11	$1.323836445808 \times 10^{0}$	$1.324147939682 \times 10^{0}$	$1.323841944168 \times 10^{0}$
12	$3.771698474884 \times 10^{-2}$	$3.648841381073 \times 10^{-2}$	$3.769529725473 \times 10^{-2}$
13	$1.466002166271 \times 10^{-1}$	$1.419594436884 \times 10^{-1}$	$1.465183827136 \times 10^{-1}$
14	$5.219259858131 \times 10^{-1}$	$5.073803663254 \times 10^{-1}$	$5.216706214352 \times 10^{-1}$
15	$1.270483732224 \times 10^{0}$	$1.257216930389 \times 10^{0}$	$1.270261773935 \times 10^{0}$
16	$2.395482063293 \times 10^{-1}$	$2.870845198631 \times 10^{-1}$	$2.403521727782 \times 10^{-1}$
17	$7.860428094864 \times 10^{-1}$	$9.010854959488 \times 10^{-1}$	$7.881011902353 \times 10^{-1}$
18	$1.290581345558 \times 10^{0}$	$1.168476819992 \times 10^{0}$	$1.289094302790 \times 10^{0}$
19	$1.655247211456 \times 10^{-1}$	$5.778930187225 \times 10^{-1}$	$1.710848467019 \times 10^{-1}$
20	$5.799036026001 \times 10^{-1}$	$1.309691071510 \times 10^{0}$	$5.965293124947 \times 10^{-1}$
21	$1.310749769211 \times 10^{0}$	$9.289216995239 \times 10^{-2}$	$1.318575587983 \times 10^{0}$
22	$8.880424499512 \times 10^{-2}$	$3.456818163395 \times 10^{-1}$	$5.837760825943 \times 10^{-2}$
23	$3.315584063530 \times 10^{-1}$	$1.024239540100 \times 10^{0}$	$2.232865975994 \times 10^{-1}$
24	$9.964407086372 \times 10^{-1}$	$9.497582316399 \times 10^{-1}$	$7.435756763952 \times 10^{-1}$
25	$1.007080554962 \times 10^{0}$	$1.092910766602 \times 10^{0}$	$1.315588346001 \times 10^{0}$
26	$9.856885075569 \times 10^{-1}$	$7.882812023163 \times 10^{-1}$	$7.003529560278 \times 10^{-2}$
27	$1.028008580208 \times 10^{0}$	$1.288963079453 \times 10^{0}$	$2.654263545206 \times 10^{-1}$
28	$9.416294097900 \times 10^{-1}$	$1.715748310089 \times 10^{-1}$	$8.503519690601 \times 10^{-1}$
29	$1.106519818306 \times 10^{0}$	$5.979855656624 \times 10^{-1}$	$1.232112462387 \times 10^{0}$
30	$7.529209256172 \times 10^{-1}$	$1.319182157516 \times 10^{0}$	$3.741464896393 \times 10^{-1}$
31	$1.311013936996 \times 10^{0}$	$5.600392818451 \times 10^{-2}$	$1.076629171429 \times 10^{0}$
32	$8.778309822083 \times 10^{-2}$	$2.146063894033 \times 10^{-1}$	$8.291255674005 \times 10^{-1}$
33	$3.280147910118 \times 10^{-1}$	$7.202578186989 \times 10^{-1}$	$1.254154650050 \times 10^{0}$
34	$9.892780780792 \times 10^{-1}$	$1.324717283249 \times 10^{0}$	$2.979069414723 \times 10^{-1}$
35	$1.021098971367 \times 10^{0}$	$3.424143791199 \times 10^{-2}$	$9.253821285571 \times 10^{-1}$
36	$9.564665555954 \times 10^{-1}$	$1.334483325481 \times 10^{-1}$	$1.132532262670 \times 10^{0}$
37	$1.081381440163 \times 10^{0}$	$4.803679585457 \times 10^{-1}$	$6.822410727153 \times 10^{-1}$
38	$8.173682689667 \times 10^{-1}$	$1.229211807251 \times 10^{0}$	$1.332605646962 \times 10^{0}$
39	$1.265200376511 \times 10^{0}$	$3.839622139931 \times 10^{-1}$	$2.909156902851 \times 10^{-3}$
40	$2.586054801941 \times 10^{-1}$	$1.093567967415 \times 10^{0}$	$1.161123802975 \times 10^{-2}$

Tabela wyników (wartości p_n w kolejnych iteracjach)

5.3 Wnioski

1. W początkowych iteracjach wartości funkcji w obu artymetykach są zbieżne.
2. Już niewielka modyfikacja (obcięcie raz po 10. iteracji) prowadzi do **znacznie odmiennych** trajektorii. Już przy 19 iteracji model z obcięciem daje duży błąd względem pozostałych dwóch modeli.
3. Różnice między `Float32` a `Float64` kumulują się — różnice rosną z liczbą iteracji (od iteracji 22 widać znaczne różnice w wartościach funkcji) i zachodzi efekt kumulacji błędów.
4. Model logistyczny przy ($r = 3$) jest numerycznie niestabilny — niewielkie błędy kumulują się dając duże rozbieżności końcowe. Wynika to z czułości na drobne zmiany wartości i precyzję arytmetyki.

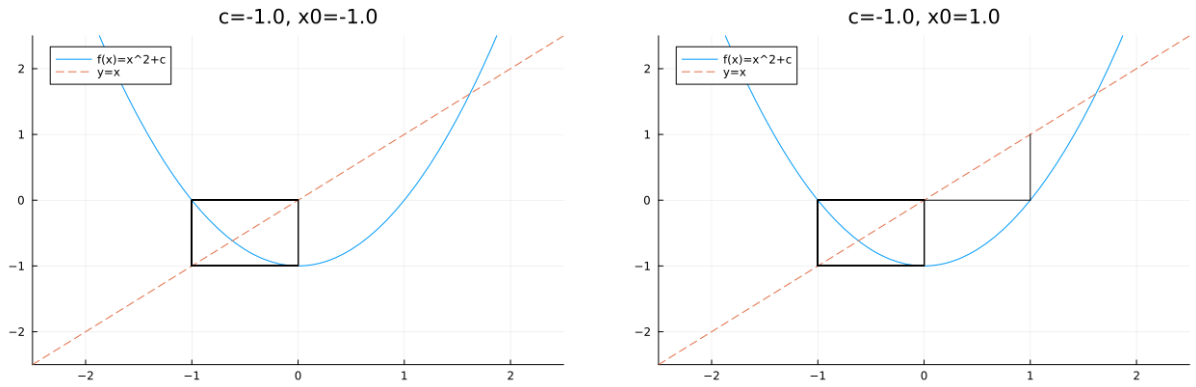
6 Zadanie 6

6.1 Opis

Zadanie polega na obliczaniu w arytmetyce `Float64` kolejnych wartości wyrażenia rekurencyjnego $x_{n+1} = x_n^2 + c$. Rozwiązanie znajduje się w pliku `task6.jl`.

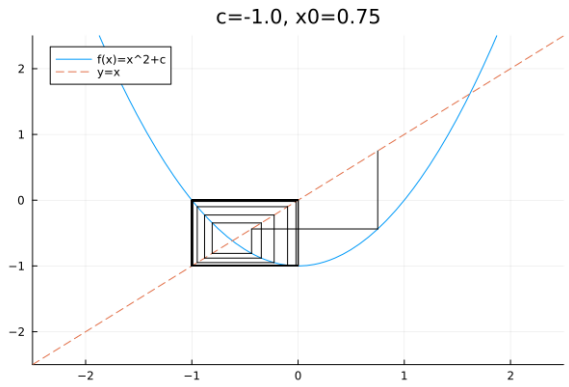
6.2 Wyniki

Poniższe graficzne iteracje ciągów to siatka, w której oś pozioma określa wartość x_n a oś pozioma to wartość $f(x_n)$.

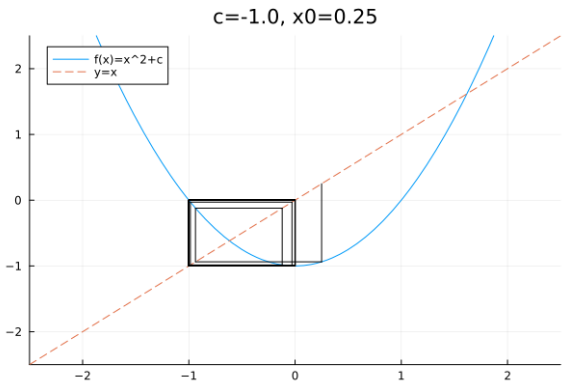


(a) $x_0 = -1.0$ (b) $x_0 = 1.0$

Rysunek 3: Graficzne iteracje ciągu $x_{n+1} = x_n^2 - 1$. Dla $x_0 = -1.0$ ciąg: $(1 \mapsto 0 \mapsto -1 \mapsto 0 \mapsto -1 \ldots)$. Szybkie wejście w cykl okresu $(0, -1)$. Dla $x_0 = 1.0$ już w cyklu: $(1 \mapsto 0 \mapsto -1 \ldots)$. Ciąg pozostaje w okresie-2 (punkt startowy na cyklu).

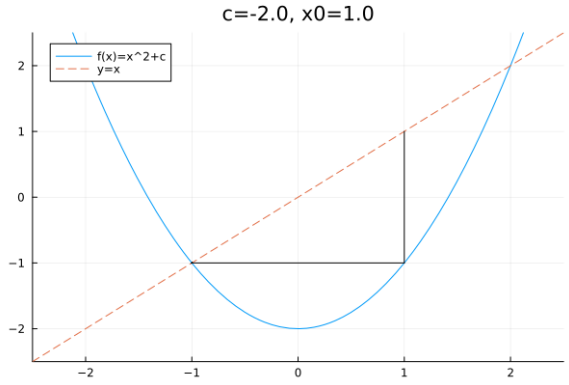


(a) $x_0 = 0.75$

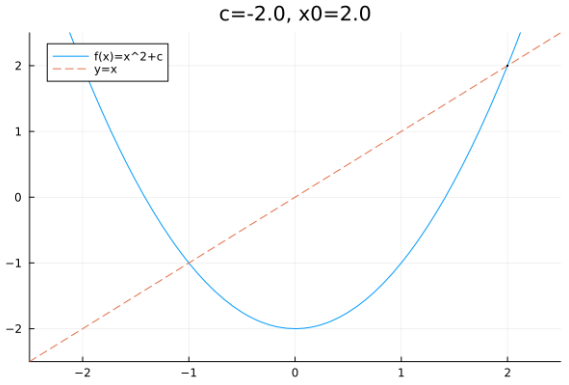


(b) $x_0 = 0.25$

Rysunek 4: Graficzne iteracje ciągu $x_{n+1} = x_n^2 - 1$. Dla $x_0 = 0.75$ początkowo nieregularne wartości, ale w krótkim czasie następuje zbliżenie do cyklu $(0, -1)$ — w tabelce widoczne częste wartości -1 i 0 od pewnej iteracji (przyciąganie cyklu). Dla $x_0 = 0.25$ również po kilku iteracjach następuje przejście ku okresowi-2 $(0, -1)$.

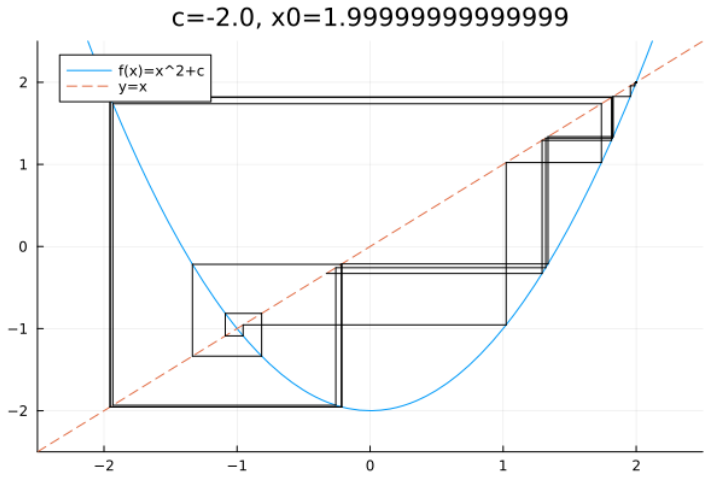


(a) $x_0 = 1.0$



(b) $x_0 = 2.0$

Rysunek 5: Graficzne iteracje ciągu $x_{n+1} = x_n^2 - 2$. Dla $x_0 = 1.0$ Szybkie przejście: $(1 \mapsto 0 \mapsto -1 \mapsto -1 \mapsto \ldots)$. Ciąg „zatrzymuje się” w -1 (punkt stały osiągnięty dokładnie). Dla $x_0 = 2.0$ ciąg równa się 2 we wszystkich iteracjach.



Rysunek 6: Start bardzo blisko 2. Przez kilka iteracji $x_n \approx 2$, potem wartości stopniowo oddalają się od 2 i pojawiają się duże oscylacje/zmiany (przejście przez wartości > 1, < 0, aż do dużych ujemnych) — wrażliwość na maleńkie perturbacje.

iter	(case 1) $c = -2, x_0 = 1$	(case 2) $c = -2, x_0 = 2$	(case 3) $c = -2, x_0 = 1.99\ldots9$	(case 4) $c = -1, x_0 = 1$	(case 5) $c = -1, x_0 = -1$	(case 6) $c = -1, x_0 = 0.75$	(case 7) $c = -1, x_0 = 0.25$
1	-1.0000000000	2.0000000000	2.0000000000	0.0000000000	0.0000000000	-0.4375000000	-0.9375000000
2	-1.0000000000	2.0000000000	2.0000000000	-1.0000000000	-1.0000000000	-0.8085937500	-0.1210937500
3	-1.0000000000	2.0000000000	2.0000000000	0.0000000000	0.0000000000	-0.3461761475	-0.9853363037
4	-1.0000000000	2.0000000000	2.0000000000	-1.0000000000	-1.0000000000	-0.8801620749	-0.0291123686
5	-1.0000000000	2.0000000000	2.0000000000	0.0000000000	0.0000000000	-0.2253147219	-0.9991524700
6	-1.0000000000	2.0000000000	2.0000000000	-1.0000000000	-1.0000000000	-0.9492332761	-0.0016943417
7	-1.0000000000	2.0000000000	1.9999999998	0.0000000000	0.0000000000	-0.0989561875	-0.9999971292
8	-1.0000000000	2.0000000000	1.9999999993	-1.0000000000	-1.0000000000	-0.9902076730	-0.0000057416
9	-1.0000000000	2.0000000000	1.9999999974	0.0000000000	0.0000000000	-0.0194887644	-1.0000000000
10	-1.0000000000	2.0000000000	1.9999999895	-1.0000000000	-1.0000000000	-0.9996201881	-0.0000000001
11	-1.0000000000	2.0000000000	1.9999999581	0.0000000000	0.0000000000	-0.0007594796	-1.0000000000
12	-1.0000000000	2.0000000000	1.9999998324	-1.0000000000	-1.0000000000	-0.9999994232	0.0000000000
13	-1.0000000000	2.0000000000	1.9999993294	0.0000000000	0.0000000000	-0.0000011536	-1.0000000000
14	-1.0000000000	2.0000000000	1.9999973178	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
15	-1.0000000000	2.0000000000	1.9999892712	0.0000000000	0.0000000000	-0.0000000000	-1.0000000000
16	-1.0000000000	2.0000000000	1.9999570848	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
17	-1.0000000000	2.0000000000	1.9998283411	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
18	-1.0000000000	2.0000000000	1.9993133938	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
19	-1.0000000000	2.0000000000	1.9972540465	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
20	-1.0000000000	2.0000000000	1.9890237264	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
21	-1.0000000000	2.0000000000	1.9562153843	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
22	-1.0000000000	2.0000000000	1.8267786299	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
23	-1.0000000000	2.0000000000	1.3371201626	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
24	-1.0000000000	2.0000000000	-0.2121096709	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
25	-1.0000000000	2.0000000000	-1.9550094875	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
26	-1.0000000000	2.0000000000	1.8220620963	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
27	-1.0000000000	2.0000000000	1.3199102828	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
28	-1.0000000000	2.0000000000	-0.2578368453	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
29	-1.0000000000	2.0000000000	-1.9335201612	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
30	-1.0000000000	2.0000000000	1.7385002138	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
31	-1.0000000000	2.0000000000	1.0223829935	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
32	-1.0000000000	2.0000000000	-0.9547330147	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
33	-1.0000000000	2.0000000000	-1.0884848707	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
34	-1.0000000000	2.0000000000	-0.8152006863	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
35	-1.0000000000	2.0000000000	-1.3354478410	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
36	-1.0000000000	2.0000000000	-0.2165790640	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
37	-1.0000000000	2.0000000000	-1.9530935090	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
38	-1.0000000000	2.0000000000	1.8145742551	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000
39	-1.0000000000	2.0000000000	1.2926797272	0.0000000000	0.0000000000	0.0000000000	-1.0000000000
40	-1.0000000000	2.0000000000	-0.3289791230	-1.0000000000	-1.0000000000	-1.0000000000	0.0000000000

Tabela 6: Iteracje ciągu $x_{n+1} = x_n^2 + c$ dla różnych parametrów (precyzja Float64).

6.3 Wnioski

1. Proces generowania ciągów równaniem rekurencyjnym ma różną stabilność w zależności od przyjętych parametrów.

2. Dla $c = -2$ starty dokładnie w (2) lub w (-1) dają stałe ciągi (cases 1–2). Natomiast starty bardzo blisko punktu niestabilnego (case 3) **szybko odchodzą** od niego:

maleńka różnica rośnie i prowadzi do dużych zmian — przykład silnej wrażliwości (numeryczna niestabilność).

3. Dla $c = -1$ obserwujemy zachowanie zbieżności do **przyciągającego cyklu okresu 2** $(0, -1)$. Z różnych początków (cases 4–7) ciągi wchodzą w ten cykl albo w niego od razu trafiają.