Sprawozdanie z Laboratorium
Obliczenia Naukowe - Lista 2

Karol Wziątek

10 listopada 2025

1 Zadanie 1

1.1 Krótki opis problemu

Powtórzyć zadanie 5 z listy 1, ale usunąć ostatnią 9 z x4 i ostatnią 7 z x5. Sprawdzić, jakie zmiany nastąpią w wyniku dla algorytmów:

1. **w przód** – $\sum_{i=1}^{n} x_i y_i$
2. **w tył** – $\sum_{i=n}^{1} x_i y_i$
3. **sortowanie rosnąco** – sortujemy iloczyny $x_i y_i$ rosnąco (w zależności od wartości bezwzględnej, osobno dodajemy ujemne i dodatnie)
4. **sortowanie malejąco** – analogicznie jak wyżej, ale sortujemy malejąco

Po modyfikacji x mamy dane:

$$\tilde{x} = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$$

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

1.2 Rozwiązanie

Podejście do problemu jest takie samo jak w zadaniu 5. z listy 1. i znaduje się w pliku zadanie1.jl.

1.3 Wyniki oraz ich interpretacja

Alg.	Float32 x * y	Float32 $\tilde{x}$ * y	Float64 x * y	Float64 $\tilde{x}$ * y
1.	-0.49994430	-0.49994430	$1.025188136829667 \cdot 10^{-10}$	-0.004296342739891585
2.	-0.45434570	-0.45434570	$-1.564330887049437 \cdot 10^{-10}$	-0.004296342998713953
3.	-0.5	-0.5	0	-0.004296342842280865
4.	-0.5	-0.5	0	-0.004296342842280865

Tabela 1: Porównanie wyników dla Float64 i Float32 oraz normalnego i zaburzonego x

Dla Float32 zaburzony x nie daje innych wyników, z kolei dla Float64 wyniki bardzo się różnią. Prawdziwy wynik to $-1.00657107 \cdot 10^{-11}$. Błąd wynika z dużej różnicy między rzędami wielkości mnożonych liczb. Float32 jest niewystarczająco precyzyjny, żeby zaburzenie miało na wpływ na rezultat. Inaczej jest w przypadku Float64, który zwraca diamertalnie inny wynik, mimo niewielkiego zaburzenia.

1.4 Wnioski

Można wnioskować, że algorytmy wyliczania iloczynu skalarnego są wrażliwe nawet na drobne zmiane, co znaczy, że zadanie jest źle uwarunkowane.

2 Zadanie 2

2.1 Krótki opis problemu

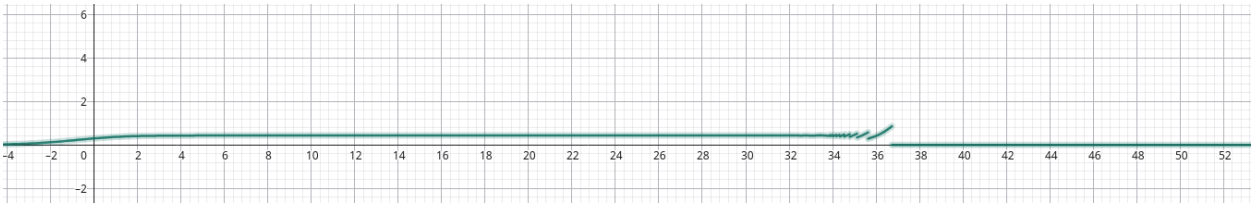
Wyznaczyć w dwóch programach graficznych wykres funkcji $f(x) = e^x \ln(1 + e^{-x})$. Obliczyć granicę tej funkcji dla $\lim_{x\to\infty} f(x)$.

2.2 Rozwiązanie

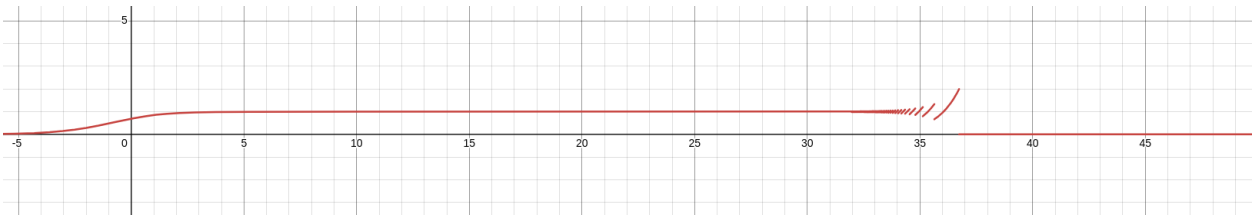
2.2.1 $\lim_{x\to\infty} f(x)$

Wykorzystując podstawowe przekształcenia i reguły de l'Hopitala, mamy $\lim_{x\to\infty} f(x) = 1$, co zostało potwierdzone przez Wolfram alpha.

2.2.2 Wykres funkcji



Rysunek 1: $f(x)$ GeoGebra



Rysunek 2: $f(x)$ Desmos

2.3 Wyniki oraz interpretacja

Wykresy funkcji nie pokrywają się z jej granicą. Już dla $x = 40$ funkcja przyjmuje wartości, które wydają się być zerem. Wcześniej występują oscylacje, które również nie przedstawiają rzeczywistego wykresu - funkcja powinna być ciągła. Prawdopodobnie dla $x > 40 : ln(1 + e^{-x}) = 0$, co wynika z ograniczonych możliwości używanej arytmetyki. Stąd również wartość funkcji wynosi 0. Osycylacje wynikają z faktu, że występuje duża różnica rzędu wielkości między czynnikami e^x oraz $ln(1 + e^{-x})$. Mnożenie takich liczb powoduje duży błąd.

2.4 Wnioski

Programy do wyznaczania wykresów również nie są odporne na błędy wynikające z obliczeń zmneinnoprzecinkowych.

3 Zadanie 3

3.1 Krótki opis problemu

Rozwiązać układ równań liniowych: $\mathbf{Ax} = \mathbf{b}$, dla macierzy $A \in \mathbb{R}^{n \times n}$ i wektora prawych stron $b \in \mathbb{R}^n$ metodami:

- eleminacji Gaussa, czyli $x = A/b$
- wprost, czli $x = A^{-1}b$

Eksperymenty należy wykonać dla macierzy Hilberta H_n z rosnącym stopniem oraz dla macierzy losowej R_n, gdzie $n = 5, 10, 20$ i rosnącym wskaźnikiem uwarunkowania c = 1, 10, 103, 107, 1012, 1016. Porównwać obliczony $\tilde{x}$ z dokładnym x, czyli policzyć błąd względny.

3.2 Rozwiązanie

Rozwiązanie zadania znajduje się w pliku zadanie3.jl. Zostały tam również dodane metody $hilb(n)$ oraz $matcond(n, c)$, a sam program wyświetla błąd względny dla jednego oraz drugiego algorytmu, wskaźnik uwarunkowania oraz rząd macierzy dla obu rodzajów macierzy.

3.3 Wyniki i interpretacja

n	cond(A)	rank(A)	błąd wz. Gauss	błąd wzg. Inwersja
2	1.928e+01	2	5.661e-16	1.404e-15
3	5.241e+02	3	8.023e-15	0.000e+00
4	1.551e+04	4	4.137e-14	0.000e+00
5	4.766e+05	5	1.683e-12	3.354e-12
6	1.495e+07	6	2.619e-10	2.016e-10
7	4.754e+08	7	1.261e-08	4.713e-09
8	1.526e+10	8	6.124e-08	3.077e-07
9	4.932e+11	9	3.875e-06	4.541e-06
10	1.602e+13	10	8.670e-05	2.501e-04
11	5.223e+14	10	1.583e-04	7.618e-03
12	1.752e+16	11	1.340e-01	2.590e-01

Tabela 2: Porównanie wyników błędów względnych dla różnych metod dla macierzy Hilberta

c	rank(A)	błąd wz. Gauss	błąd wzg. Inwersja
1.0e+00	5	1.490e-16	1.790e-16
1.0e+01	5	3.878e-16	7.161e-16
1.0e+03	5	1.070e-14	1.210e-14
1.0e+07	5	1.110e-11	9.935e-11
1.0e+12	5	1.249e-05	7.716e-06
1.0e+16	4	1.136e-01	1.227e-01

Tabela 3: Porównanie wyników błędów względnych dla różnych metod dla macierzy losowej dla n = 5

c	rank(A)	błąd wz. Gauss	błąd wzg. Inwersja
1.0e+00	10	3.511e-16	1.955e-16
1.0e+01	10	3.331e-16	4.877e-16
1.0e+03	10	5.096e-15	8.485e-15
1.0e+07	10	7.364e-11	6.421e-11
1.0e+12	10	5.300e-06	6.773e-06
1.0e+16	9	8.656e-03	1.474e-02

Tabela 4: Porównanie wyników błędów względnych dla różnych metod dla macierzy losowej dla n = 10

c	rank(A)	błąd wz. Gauss	błąd wzg. Inwersja
1.0e+00	20	3.941e-16	4.271e-16
1.0e+01	20	6.799e-16	5.769e-16
1.0e+03	20	2.617e-14	2.728e-14
1.0e+07	20	1.035e-10	1.575e-10
1.0e+12	20	1.762e-05	1.254e-05
1.0e+16	18	1.480e-02	8.841e-02

Tabela 5: Porównanie wyników błędów względnych dla różnych metod dla macierzy losowej dla n = 20

- Dla macierzy Hilberta wraz ze wzrostem wielkości bardzo szybko rośnie zarówno wskaźnik uwarunkowania jak i błąd względy w obu metodach.
- Dla macierzy losowych im większy wskaźnik uwarunkowania, tym większy błąd, rozmiar macierzy ani algorytm nie wykazują wpływu na błąd względny.

3.4 Wnioski

Stąd można wywnioskować, że przy dużym wskaźniku uwarunkowania, tzn. źle uwarunkowanym zadaniu, błąd względny jest duży.

4 Zadanie 4

4.1 Krótki opis problemu

Dany jest wielomian P w postaci naturalnej oraz w postaci iloczynowej $p = (x-20)(x-19)...(x-1)$. Trzeba znaleźć pierwiastki z_k tego wielomianu, korzystając z postaci naturalnej. Policzyć $|P(z_k)|$, $|p(z_k)|$ oraz $|z_k - k|$.
W drugiej części zadania należy powtórzyć eksperyment - zamiast -210 ma być $-210 - 2^{-23}$ przy x^{19}.

4.2 Rozwiązanie

Rozwiązanie znajduje się w pliku zadania4.jl.
Na początku przy pomoocy funkcji *roots*() znajduję pierwiastki dla postaci normalnej i od razu obliczam $|z_k - k|$. Później poprzez podstawienie do $P(x)$ i $p(x)$ wyliczam $|P(z_k)|$ oraz $|p(z_k)|$.
Analogicznie postępuję dla podpunktu b.

4.3 Wyniki i interpretacja

k	z_k	$\|z_k - k\|$	$\|P(z_k)\|$	$\|p(z_k)\|$
1	0.999999999999699	0.000000000000301	3.569650964788257e+04	5.518479490350445e+06
2	2.000000000028318	0.000000000028318	1.762526002666841e+05	7.378697629901740e+19
3	2.999999999592097	0.000000000407903	2.791576968824087e+05	3.320413931687579e+20
4	3.999999983737532	0.000000016262468	3.027109298899109e+06	8.854437035384718e+20
5	5.000000665769791	0.000000665769791	2.291747375656708e+07	1.844675205654569e+21
6	5.999989245824773	0.000010754175227	1.290241728420510e+08	3.320394888870117e+21
7	7.000102002793008	0.000102002793008	4.805112754602064e+08	5.423593016891273e+21
8	7.999355829607762	0.000644170392238	1.637952021896114e+09	8.262050140110275e+21
9	9.002915294362053	0.002915294362053	4.877071372550003e+09	1.196559421646277e+22
10	9.990413042481725	0.009586957518275	1.363863819545813e+10	1.655260133520688e+22
11	11.025022932909318	0.025022932909318	3.585631295130865e+10	2.247833297924790e+22
12	11.953283253846857	0.046716746153143	7.533332360358197e+10	2.886944688412679e+22
13	13.074314032447340	0.074314032447340	1.960598812433082e+11	3.807325552826988e+22
14	13.914755591802127	0.085244408197873	3.575134782310432e+11	4.612719853150334e+22
15	15.075493799699476	0.075493799699476	8.216271236455970e+11	5.901011420218566e+22
16	15.946286716607972	0.053713283392028	1.551497888049407e+12	7.010874106897764e+22
17	17.025427146237412	0.025427146237412	3.694735918486229e+12	8.568905825736165e+22
18	17.990921352716480	0.009078647283520	7.650109016515867e+12	1.014479936104443e+23
19	19.001909818299438	0.001909818299438	1.143527374972120e+13	1.199037620237126e+23
20	19.999809291236637	0.000190708763363	2.792410639368073e+13	1.401911741431813e+23

Tabela 6: Obliczenie pierwiastków postaci normalnej, błędu bezwzględnego, i wartości wielomianu dla postaci normalnej i iloczynowej dla obliczonych wcześniej pierwiastków

k	z_k	$\|z_k - k\|$	$\|P(z_k)\|$	$\|p(z_k)$
1	0.9999999999998357 + 0.0im	1.6431300764452317e-13	2.025987231341821e+04	3.01310012768
2	2.0000000000550373 + 0.0im	5.503730804434781e-11	3.465414137593836e+05	7.37869763029
3	2.99999999660342 + 0.0im	3.3965799062229962e-9	2.258059700119701e+06	3.32041392011
4	4.000000089724362 + 0.0im	8.972436216225788e-8	1.054263179039548e+07	8.85443781742
5	4.99999857388791 + 0.0im	1.4261120897529622e-6	3.757830916585153e+07	1.84467269740
6	6.000020476673031 + 0.0im	2.0476673030955794e-5	1.314094332556945e+08	3.32045019528
7	6.99960207042242 + 0.0im	0.00039792957757978087	3.939355874647618e+08	5.42236652891
8	8.007772029099446 + 0.0im	0.007772029099445632	1.184986961371896e+09	8.28939986098
9	8.915816367932559 + 0.0im	0.0841836320674414	2.225522123307771e+09	1.16074725017
10	10.095455630535774 - 0.6449328236240688im	0.6519586830380407	1.067792123293016e+10	1.72128928536
11	10.095455630535774 + 0.6449328236240688im	1.1109180272716561	1.067792123293016e+10	1.72128928536
12	11.793890586174369 - 1.6524771364075785im	1.665281290598479	3.140196234442949e+10	2.85684010040
13	11.793890586174369 + 1.6524771364075785im	2.0458202766784277	3.140196234442949e+10	2.85684010040
14	13.992406684487216 - 2.5188244257108443im	2.518835871190904	2.157665405951858e+11	4.93464714768
15	13.992406684487216 + 2.5188244257108443im	2.7128805312847097	2.157665405951858e+11	4.93464714768
16	16.73074487979267 - 2.812624896721978im	2.9060018735375106	4.850110893921027e+11	8.48469471356
17	16.73074487979267 + 2.812624896721978im	2.825483521349608	4.850110893921027e+11	8.48469471356
18	19.5024423688181 - 1.940331978642903im	2.4540214463129764	4.557199223869993e+12	1.31819478206
19	19.5024423688181 + 1.940331978642903im	2.0043294443099486	4.557199223869993e+12	1.31819478206
20	20.84691021519479 + 0.0im	0.8469102151947894	8.756386551865696e+12	1.59110840814

Tabela 7: Obliczenie pierwiastków postaci normalnej z zaburzeniem, błędu bezwzględnego, i wartości wielomianu dla postaci normalnej i iloczynowej dla obliczonych wcześniej pierwiastków

Pierwiastki wielomianu w postaci naturalnej zostały obliczone z dobrą dokładnością (mogłoby się wydawać), ponieważ błąd bezwzględny $< 10^{-1}$. Jednakże przy próbie podstawienia tych pierwiastków do $P(x)$ i $p(x)$ otrzymujemy coraz większe wartości, im większe k. Warto zauważyć, że wartości $|p(z_k)|$ rosną znacznie szybciej niż dla $|P(z_k)|$. W przypadku $P(z_k)$ błąd ten wynika stąd, że współczynniki przy najniższych potęgach x są bardzo duże i wychodzą poza arytmetyke, co generuje duży błąd. Z kolei w przypadku obliczania wartości $p(z_k)$ dochodzi do mnożenia liczb skrajne różnych rzędów, co też prowadzi do dużego błędu.
W podpunkcie b widać, że nawet lekkie zaburzenie prowadzi do powstania częsci urojnej przy wyliczaniu z_k, a błąd bezwględny rośnie nawet do 2.

4.4 Wnioski

Wielomiany o dużych współczynnikach lub wielu pierwiastkach stanowią problem dla maszyny, ponieważ wyliczone wartości z takich wielomianów obraczone są dużym błędem. Ponadto, zadanie jest źle uwarunkowane, skoro mała zamiana w jednym ze współczynników prowadzi do tak dużych rozbieżności w wyniku.

5 Zadanie 5

5.1 Krótki opis problemu

Należy policzyć kolejne wartości funkcji rekurencyjnej. Zobaczyć różnice między wyliczaniem tych wartości we Float32 i Float64 oraz różnicę między brakiem ingerencji w wartości, a obcięiem kilku znaczących cyfr po otrzymaniu 10. wartości.

5.2 Rozwiązanie

Rowiązanie znajduje się w pliku zadanie5.jl.
Program jest prosty - implementuje tę funkcję rekurencyjną kilkukrotnie tzn. raz dla Float32, raz dla Float64, a raz z obcięciem.

5.3 Wyniki i interpretacja

n	wartości bez obcięcia	wartości z obcięciem
1	0.0396999978	0.0396999978
2	0.1540717334	0.1540717334
3	0.5450726151	0.5450726151
4	1.2889779806	1.2889779806
5	0.1715191603	0.1715191603
6	0.5978201628	0.5978201628
7	1.3191138506	1.3191138506
8	0.0562714860	0.0562714860
9	0.2155864984	0.2155864984
10	0.7229133844	0.7229133844
11	1.3238422871	1.3241480589
12	0.0376940966	0.0364882238
13	0.1465138495	0.1419587135
14	0.5216564536	0.5073780417
15	1.2702494860	1.2572147846
16	0.2403967381	0.2870922685
17	0.7882151604	0.9011031985
18	1.2890112400	1.1684519053
19	0.1713951081	0.5779681206
20	0.5974515676	1.3097310066
21	1.3189611435	0.0927380323
22	0.0568690598	0.3451511264
23	0.2177739739	1.0232166052
24	0.7288193703	0.9519498348
25	1.3217444420	1.0891739130
26	0.0459526218	0.7977962494
27	0.1774755567	1.2817484140
28	0.6154094934	0.1983565390
29	1.3254514933	0.6753902435
30	0.0313411988	1.3331050873
31	0.1224179864	0.0009130480
32	0.4447134733	0.0036496911
33	1.1855436563	0.0145588033
34	0.5256333351	0.0575993359
35	1.2736620903	0.2204443067
36	0.2280029058	0.7359901071
37	0.7560556531	1.3189160824
38	1.3093621731	0.0570453405
39	0.0941608250	0.2184188515
40	0.3500445187	0.7305550575

Tabela 8: Porównanie wyników między przeprowadzeniem symulacji bez i z obcięciem cyfr znaczących

n	wartości Float32	wartości Float64
1	0.0396999978	0.0397000000
2	0.1540717334	0.1540717300
3	0.5450726151	0.5450726260
4	1.2889779806	1.2889780012
5	0.1715191603	0.1715191421
6	0.5978201628	0.5978201201
7	1.3191138506	1.3191137924
8	0.0562714860	0.0562715776
9	0.2155864984	0.2155868392
10	0.7229133844	0.7229143012
11	1.3238422871	1.3238419442
12	0.0376940966	0.0376952973
13	0.1465138495	0.1465183827
14	0.5216564536	0.5216706214
15	1.2702494860	1.2702617739
16	0.2403967381	0.2403521728
17	0.7882151604	0.7881011902
18	1.2890112400	1.2890943028
19	0.1713951081	0.1710848467
20	0.5974515676	0.5965293125
21	1.3189611435	1.3185755880
22	0.0568690598	0.0583776083
23	0.2177739739	0.2232865976
24	0.7288193703	0.7435756764
25	1.3217444420	1.3155883460
26	0.0459526218	0.0700352956
27	0.1774755567	0.2654263545
28	0.6154094934	0.8503519691
29	1.3254514933	1.2321124624
30	0.0313411988	0.3741464896
31	0.1224179864	1.0766291714
32	0.4447134733	0.8291255674
33	1.1855436563	1.2541546501
34	0.5256333351	0.2979069415
35	1.2736620903	0.9253821286
36	0.2280029058	1.1325322627
37	0.7560556531	0.6822410727
38	1.3093621731	1.3326056470
39	0.0941608250	0.0029091569
40	0.3500445187	0.0116112380

Tabela 9: Porównanie wyników między przeprowadzeniem symulacji dla Float32 i Float64

Widać, że w przypadku obcięcia wartości do $0,722$ kolejne wyniki zaczynają coraz bardziej się różnić. Po kilku wywołaniach (od $n = 19$) wartości wydają się być od siebie niezależne aż do końca eksperymentu. Inaczej wygląda sytuacja przy porównywaniu Float32 i Float64 bez zewnętrznej ingerencji w wartości. Na drugim miejscu po przecinku dochodzi do różnicy dopiero dla $n = 23$, a na pierwszym dla $n = 27$. Wartości wydają się być bliskie sobie aż do samego końca.

5.4 Wnioski

Drobna modyfikacja liczby wpływa na zupełni inny "przebieg symulacji". Błąd się skaluje z każdą iteracją. Dodatkowo można by się zastanawiać, który z tych procesów najdokładniej odzwierciedli faktyczny stan populacji przy danych założeniach początkowych. Otóż okazuje się, że żaden, ponieważ każdej iteracji towarzyszy podnoszenie obliczanej wartości do kwadratu. W związku z tym potrzebna jest za każdym razem 2 razy większa precyzja, żeby nie zgubić dokładności, co przy Float32 i Float64 jest niemożliwe. Błąd ten znacznie powiększa się z każdą iteracją. Już po kilku wywołaniach równie dobrze można by generować wartości losowe. Mamy tu doczynienia z **numeryczną niestabilnością**.

6 Zadanie 6

6.1 Krótki opis problemu

Należy policzyć kolejne wartości funkcji rekurencyjnej $x_{n+1} = x_n^2 + c$. Zaobserwować zachowanie ciągów.

6.2 Rozwiązanie

Rowiązanie znajduje się w pliku zadanie6.jl.
Program jest prosty - implementuje tę funkcję rekurencyjną, dla ułatwienia liczę wartości iteracyjnie.

6.3 Wyniki i interpretacja

n	$x_0 = 1$	$x_0 = 2$	$x_0 = 1.99999999$
1	1.000000000000000	2.000000000000000	1.999999999999990
2	-1.000000000000000	2.000000000000000	1.999999999999960
3	-1.000000000000000	2.000000000000000	1.999999999999840
4	-1.000000000000000	2.000000000000000	1.999999999999361
5	-1.000000000000000	2.000000000000000	1.999999999997442
6	-1.000000000000000	2.000000000000000	1.999999999989768
7	-1.000000000000000	2.000000000000000	1.999999999959073
8	-1.000000000000000	2.000000000000000	1.999999999836291
9	-1.000000000000000	2.000000000000000	1.999999999345164
10	-1.000000000000000	2.000000000000000	1.999999997380655
11	-1.000000000000000	2.000000000000000	1.999999989522621
12	-1.000000000000000	2.000000000000000	1.999999958090484
13	-1.000000000000000	2.000000000000000	1.999999832361938
14	-1.000000000000000	2.000000000000000	1.999999329447781
15	-1.000000000000000	2.000000000000000	1.999997317791575
16	-1.000000000000000	2.000000000000000	1.999989271173494
17	-1.000000000000000	2.000000000000000	1.999957084809083
18	-1.000000000000000	2.000000000000000	1.999828341078044
19	-1.000000000000000	2.000000000000000	1.999313393778961
20	-1.000000000000000	2.000000000000000	1.997254046543948
21	-1.000000000000000	2.000000000000000	1.989023726436175
22	-1.000000000000000	2.000000000000000	1.956215384326049
23	-1.000000000000000	2.000000000000000	1.826778629873910
24	-1.000000000000000	2.000000000000000	1.337120162564000
25	-1.000000000000000	2.000000000000000	-0.212109670864823
26	-1.000000000000000	2.000000000000000	-1.955009487525616
27	-1.000000000000000	2.000000000000000	1.822062096315173
28	-1.000000000000000	2.000000000000000	1.319910282828443
29	-1.000000000000000	2.000000000000000	-0.257836845283740
30	-1.000000000000000	2.000000000000000	-1.933520161214129
31	-1.000000000000000	2.000000000000000	1.738500213821511
32	-1.000000000000000	2.000000000000000	1.022382993457439
33	-1.000000000000000	2.000000000000000	-0.954733014689007
34	-1.000000000000000	2.000000000000000	-1.088484870662841
35	-1.000000000000000	2.000000000000000	-0.815200686338098
36	-1.000000000000000	2.000000000000000	-1.335447840993894
37	-1.000000000000000	2.000000000000000	-0.216579063984746
38	-1.000000000000000	2.000000000000000	-1.953093509043491
39	-1.000000000000000	2.000000000000000	1.814574255067817
40	-1.000000000000000	2.000000000000000	1.292679727154924

Tabela 10: Porównanie wyników funkcji rekrencyjnej dla $c = -2$ i stanów początkowych $x_0 = 1, 2, 1.9999999$

n	$x_0 = 1$	$x_0 = -1$	$x_0 = 0.75$	$x_0 = 0.25$
1	1.000000000000000	-1.000000000000000	0.750000000000000	0.250000000000000
2	0.000000000000000	0.000000000000000	-0.437500000000000	-0.937500000000000
3	-1.000000000000000	-1.000000000000000	-0.808593750000000	-0.121093750000000
4	0.000000000000000	0.000000000000000	-0.346176147460938	-0.985336303710938
5	-1.000000000000000	-1.000000000000000	-0.880162074929103	-0.029112368589267
6	0.000000000000000	0.000000000000000	-0.225314721856496	-0.999152469995123
7	-1.000000000000000	-1.000000000000000	-0.949233276114730	-0.001694341702646
8	0.000000000000000	0.000000000000000	-0.098956187516497	-0.999997129206195
9	-1.000000000000000	-1.000000000000000	-0.990207672952200	-0.000005741579369
10	0.000000000000000	0.000000000000000	-0.019488764426589	-0.999999999967034
11	-1.000000000000000	-1.000000000000000	-0.999620188061125	-0.000000000065931
12	0.000000000000000	0.000000000000000	-0.000759479620641	-1.000000000000000
13	-1.000000000000000	-1.000000000000000	-0.999999423190706	0.000000000000000
14	0.000000000000000	0.000000000000000	-0.000001153618256	-1.000000000000000
15	-1.000000000000000	-1.000000000000000	-0.999999999998669	0.000000000000000
16	0.000000000000000	0.000000000000000	-0.000000000002662	-1.000000000000000
17	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
18	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
19	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
20	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
21	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
22	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
23	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
24	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
25	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
26	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
27	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
28	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
29	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
30	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
31	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
32	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
33	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
34	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
35	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
36	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
37	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
38	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000
39	-1.000000000000000	-1.000000000000000	-1.000000000000000	0.000000000000000
40	0.000000000000000	0.000000000000000	0.000000000000000	-1.000000000000000

Tabela 11: Porównanie wyników funkcji rekrencyjnej dla $c = -1$ i stanów początkowych $x_0 = 1, -1, 0.75, 0.25$

W przypadku $c = -2$ oraz $x_0 = 1, 2$ widzimy, że wartość już od 2 iteracji pozostaje bez zmian aż do końca, kolejne iteracje nie mają wpływu na zmianę wyniku - **zachowanie stabilne**. Z kolei dla $c == -2$ oraz $x_0 = 1.9999999$ widać, że dla każdej kolejnej wartości odchylenie od 2 jest coraz większe i wzrasta też błąd, który wynika z podnoszenia do kwadratu na ograniczonej precyzji - **zachowanie chaotyczne**

Dla $c = -1$ obserwujemy zachowanie stabilne dla $x_0 = 1, -1$ i zachowanie niestabilne w pozostałych przypadkach. Ciekawe jest to, że $x_0 = 0.75, 0.25$ po pewnym czasie również zaczynają przyjmować cykliczne wartości 0 i -1, ale to wynika tylko z błędu zaokrąlenia w systemie zmiennopozycyjnym - w rzeczywistości wartości tych funkcji dla takich danych początkowych nigdy nie są całkowite.

6.4 Wnioski

Analiza tego typu układów nie jest prosta, ponieważ stany stabilne i niestabilne przeplatają się ze sobą i zależą od warunków początkowych.