Obliczenia Naukowe

Laboratorium 2

Bartosz Grelewski

Listopad 2022

1 Zadanie 1

1.1 Opis Problemu

Powtórz zadanie 5 z listy 1, ale usuń ostatnią 9 z x4 i ostatnią 7 z x5. Jaki wpływ na wyniki mają niewielkie zmiany danych?

1.2 Rozwiązanie

Należało zmienić tylko odpowiednie wartości w jednym z podanego ciągu wejścia. Następnie porównuję wartości otrzymane przed i po wykonaniu treści polecenia i Formułuje wniosek.

1.3 Interpretacja wyniku

Wyniki dla Float32 zad 5

Algorytm	Wynik
1	-0.4999443
2	-0.4543457
3	-0.5
4	-0.5

Wyniki dla Float64 zad 5

Algorytm	Wynik
1	1.0251881368296672e-10
2	-1.5643308870494366e-10
3	0.0
4	0.0

Wyniki dla Float32 zad 1

Algorytm	Wynik
1	-0.4999443
2	-0.4543457
3	-0.5
4	-0.5

Wyniki dla Float64 zad 1

Algorytm	Wynik
1	-0.004296342739891585
2	-0.004296342998713953
3	-0.004296342842280865
4	-0.004296342842280865

1.4 Wnioski

Analizując powyższe wyniki można łatwo zauważyć, że nie zmieniły się dla Float32 względem zadania 5 z listy 1. Jest to spowodowane niewielką precyzją arytmetyki. Porównując z arytmetyką Float64 można dojść do wniosku, że pomimo nie wprowadzenia zmian w algorytmach struktura Float32 dała zgoła inne wyniki niż Float64. Tutaj niewielka zmiana rzutowała na wynik iloczynu skalarnego bardzo mocno. Do rozwiązywania takich zadań należy używać maksymalnej precyzji. Zadanie jest źle uwarunkowane, małe decyzje w danych wejściowych powodują większe zaburzenia. Wgłębiając się bardziej w strukturę problemu błędu możemy dojść do wniosku, że modyfikacja danych wejściowych o najmniejszą cyfrę znaczącą (w tym wypadku rzędu 10^{-10}) jest inaczej zapamiętywana przez różne standardy. Właśnie ze względu na różnicę liczby bitów przeznaczonych na zapamiętanie mantysy. Mamy sytuację, w której Float32 nie widzi różnicy bo wszystkie zmiany zachodzą poza zakresem mantysy. Zaś Float64 ma większą precyzję, większą liczbę bitów na zapamiętanie mantysy co za tym idzie obejmuje swoim obszarem mantysy zmiany, co widać w wynikach.

2 Zadanie 2

2.1 Opis Problemu

Narysować wykres funkcji f(x) = $e^x \ln(1 + e^{-x})$ w co najmniej dwóch dowolnych programach do wizualizacji. Następnie policzyć granicę funkcji $\lim_{x \to \infty} f(x)$. Porównać wykres funkcji z policzoną granicą. Wyjaśnić zjawisko.

2.2 Rozwiązanie

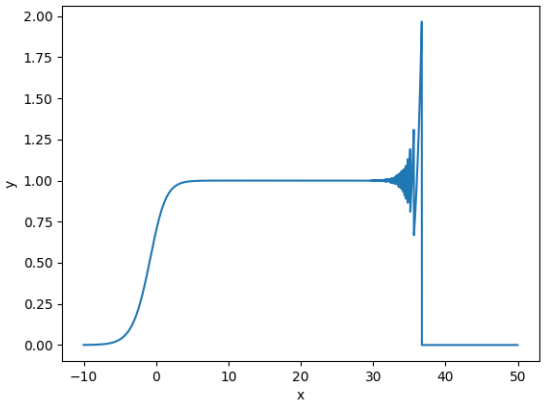


Figure 1: Wykres wygenerowany za pomocą Pythona mathplotlib.

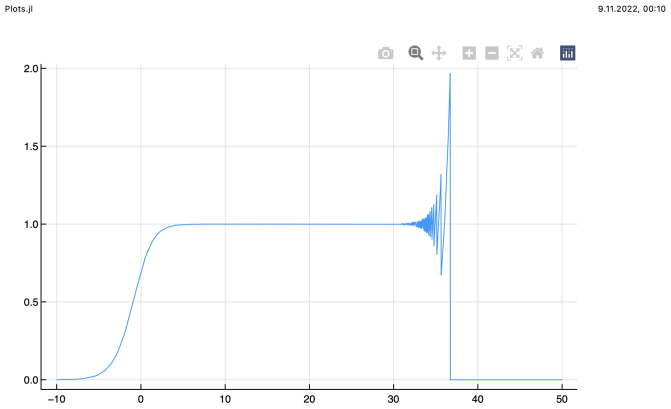


Figure 2: Wykres wygenerowany za pomocą Julia plots.

Wykresy funkcji wykonałem przy użyciu biblioteki Plotly w języku Julia oraz matplotlib w języku Python.

$\lim_{x \to \infty} f(x) = 1$

2.3 Interpretacja wyniku i wnioski

Patrząc na powyższe wykresy można dostrzec, że dla wartości powyżej 30 wykres zaczyna odbiegać od oczekiwanej wartości i schodzi do zera. Nie zgadza się to z wyliczoną przez nas granicą. Moment pojawienia się oscylacji w wykresie jest różny dla Float16, Float32, Float64. Obliczając e^{-x} dla wartości 30/40 można zauważyć, że przez ujemny wykładnik zbliża się ona do okolic epsilona maszynowego, a na końcu spada do 0 co sprawia, że wykres zaczyna oscylować wokół 1 i spadać do zera.

3 Zadanie 3

3.1 Opis Problemu

Rozważmy zadanie rozwiązywania układu równań liniowych A_x = b. dla danej macierzy współczynników A Rn×n i wektora prawych stron b należy do R_n. Macierz A generować w

następujący sposób: (a) A = Hn, gdzie Hn jest macierzą Hilberta stopnia n wygenerowaną za pomocą funkcji A=hilb(n) (źródła w języku Julia na stronie domowej), (b) A = Rn, gdzie Rn jest losową macierzą stopnia n z zadanym wskaźnikiem uwa- runkowania c wygenerowaną za pomocą funkcji A=matcond(n,c) (źródła w języku Julia na stronie domowej). Wektor b zadany jest następująco b = Ax, gdzie A jest wygenerowaną macierzą, a x = (1,...,1)T. Zatem wiemy jakie jest rozwiązanie dokładne Ax = b dla A i b. Rozwiązać Ax = b za pomocą dwóch algorytmów: eliminacji Gaussa (x=A\) oraz x = A-1b (x=inv(A)*b). Eksperymenty wykonać dla macierzy Hilberta Hn z rosnącym stopniem n > 1 oraz dla macierzy losowej Rn, n = 5,10,20 z rosnącym wskaźnikiem uwarunkowania c = 1, 10, 103, 107, 1012, 1016. Porównać obliczony x z rozwiązaniem do- kładnym x = (1, . . . , 1)T ,tj. policzyć błędy względne. W pakiecie LinearAlgebra znajdują się dwie użyteczne funkcje cond(A) i rank(A). Za pomocą funkcji cond(A) można sprawdzić jaki jest wskaźnik uwarunkowania wygenero- wanej macierzy. Natomiast za pomocą funkcji rank(A) można sprawdzić jaki jest rząd macierzy.

3.2 Rozwiązanie

Specyfikacja zadania obejmuje następującą listę kroków: Dane z zadania:
A - macierz współczynników
n - liczba wierszy A
1. Rozwiązuje układ równań Ax = b, gdzie prawdziwym rozwiązaniem jest wektor jedynek.
2. Najpierw wyznacza b = A * transposed(1,1,....,1). N
3. Następnie wyznacza rozwiązania metodą Gaussa i metodą inwersji.
4. Ostatecznie wyznacza błędy względne uzyskane dla obu metod.

3.3 Interpretacja wyniku

Tabela 1: Dla macierzy Hilberta.

n	cond(A)	Rank(A)	Gaussian err	Inversion err
1	1.0	1	0.0	0.0
5	476607.2502419222	5	2.0813331825304352e-13	8.58170264218487e-12
9	4.931539099297127e11	9	9.820864124028257e-6	6.7115610658087385e-6
13	6.738714767183357e17	11	1.9340328542478238	2.176661432102452
17	6.968485027955343e17	12	5.702847199359304	7.400447213245778
21	4.834811654527856e18	13	20.04080147772674	65.89364256450993
25	2.0216257487362834e19	13	54.375510677875205	52.526514863373635
29	2.6372941826255836e18	14	38.587634774940206	42.526167209158594
33	2.3517543604793573e19	14	48.708547648531024	31.858210936424033
37	5.804866262311895e18	15	18.21576326054527	30.84424032015582
41	8.630672125675325e18	15	28.78445680346668	52.86238215771376
45	6.376167235577815e18	15	281.0917293715779	291.4241044101379
49	9.384035584706191e18	16	53.48685784306524	51.65935529196263

Tabela 2: Dla macierzy o losowych współczynnikach.

n	c	Rank(A)	Gaussian err	Inversion err
5	1.0000000000000002	5	1.719950113979703e-16	2.482534153247273e-16
5	10.000000000000005	5	4.864753555590494e-16	3.9409007944299576e-16
5	999.9999999999898	5	1.8804438326238797e-14	1.8427963223201653e-14
5	1.0000000001993531e7	5	1.4427810447648212e-10	1.4794708173767986e-10
5	1.0000432600764006e12	5	8.890991966530342e-5	9.284378429068607e-5
5	8.833890737847085e15	4	0.6531608762030892	0.6882616824214273
10	1.0000000000000007	10	2.4575834280036907e-16	2.937374022976103e-16
10	9.999999999999996	10	3.98754029708303e-16	3.6654177513682335e-16
10	999.9999999999654	10	1.9408273551324266e-15	3.1448916673247165e-15
10	9.999999994078564e6	10	4.953203833547687e-12	2.769819410134542e-11
10	9.999959767634231e11	10	1.8026351820032064e-5	1.0185820951160972e-5
10	2.208878925809199e16	9	0.052044290007979915	0.032238146305432085
20	1.0000000000000013	20	3.2272943992214547e-16	3.2558130188798225e-16
20	10.000000000000002	20	3.640109616122045e-16	2.9582808634907537e-16
20	1000.0000000000488	20	4.658760923771699e-14	4.30447082914865e-14
20	9.999999999973744e6	20	6.736623052192738e-10	7.122469456279812e-10
20	1.0000677079866104e12	20	1.7128721774840067e-5	1.8114197310498265e-5
20	3.951205557453794e15	19	0.6646624809964294	0.6022487416370418

3.4 Wnioski

Po dokonaniu obliczeń możemy zauważyć, że dla macierzy Hilberta osiągane są duże wskaźniki uwarunkowania nawet dla małych wielkości. Łatwo można dojść do wniosku, że po załadowaniu danych i sprawdzeniu rezultatów metoda Gaussa umożliwia nam dokładniejsze wyniki dla tych macierzy. Mianownik wspólny dla obu metod to duże błędy względne. Patrząc na macierze losowe o ustalonym wskaźniku uwarunkowania nie jest już takie proste wskazanie dokładnych różnic pomiędzy algorytmami. Ponadto rzędy błędów są podobne dla macierzy o różnych rozmiarach jeśli ich współczynnik uwarunkowania jest zbliżony. Rozwiązanie układu równań liniowych Hilbert jest zadaniem źle uwarunkowanym. Jednak jeśli znamy współczynnik uwarunkowania macierzy A, to jesteśmy w stanie oszacować błędy względne.

4 Zadanie 4

4.1 Opis Problemu

("złośliwy wielomian", Wilkinson) Zainstalować pakiet Polynomials.
(a) Użyć funkcji roots (z pakietu Polynomials) do obliczenia 20 zer wielomianu P w

postaci naturalnej

$$\begin{aligned} P(x) &= x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} \\ &\quad -1672280820x^{15} + 40171771630x^{14} - 756111184500x^{13} \\ &\quad +11310276995381x^{12} - 135585182899530x^{11} \\ &\quad +1307535010540395x^{10} - 10142299865511450x^{9} \\ &\quad +63030812099294896x^{8} - 311333643161390640x^{7} \\ &\quad +1206647803780373360x^{6} - 3599979517947607200x^{5} \\ &\quad +8037811822645051776x^{4} - 12870931245150988800x^{3} \\ &\quad +13803759753640704000x^{2} - 8752948036761600000x \\ &\quad +2432902008176640000 \end{aligned}$$

(współczynniki wielomianu P umieszczone są na stronie domowej w pliku wielomian.txt). P jest postacią naturalną wielomianu Wilkinsona p

$$\begin{aligned} p(x) &= (x-20)(x-19)(x-18)(x-17)(x-16) \\ &\quad (x-15)(x-14)(x-13)(x-12)(x-11) \\ &\quad (x-10)(x-9)(x-8)(x-7)(x-6) \\ &\quad (x-5)(x-4)(x-3)(x-2)(x-1) \end{aligned}$$

Sprawdzić obliczone pierwiastki obliczając $|P(z_k)|$, $|p(z_k)|$ i $|z_k - k|$. Wyjaśnić rozbieżności.

(b) Powtórzyć eksperyment Wilkinsona, tj. zmienić współczynnik -210 na $-210na - 210 - 2^{-23}$. Wyjaśnić zjawisko.

4.2 Rozwiązanie

Należało zaimportować dane z pliku dostarczonego w specyfikacji zadania. Następnie znaleźć pierwiastki za pomocą funkcji roots z pakietu Polynomials. Kolejnym krokiem było sprawdzenie wartości osiąganych dla pierwiastków przez wielomian w postaci iloczynowej, p(x) i naturalnej P(x). Na koniec wystarczyło porównać jak wypadają wyniki znalezionych pierwiastków względem faktycznych wyników.

4.3 Interpretacja wyniku

Tabela 1: Mamy w niej wartości dla znalezionych wartości pierwiastków wielomianu.

k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999999833	1603.9573920179469	304572.04260797694	1.6653345369377348e-14
2	2.00000000000086	7032.126244053075	7.378697629496515e19	8.602007994795713e-13
3	3.000000000336467	363020.53660426877	3.320413934571126e20	3.3646685437815904e-10
4	3.99999996989576	4.1297007568024816e6	8.854436933250461e20	3.0104239989725556e-8
5	5.000000877361879	3.075695036659063e7	1.844675459361676e21	8.773618791479976e-7
6	5.999986963459685	1.4257935012621844e8	3.3203908470754844e21	1.3036540314814715e-5
7	7.000117697963099	5.24632388586803e8	5.423631524959937e21	0.00011769796309923919
8	7.999291601888181	1.7358866242071867e9	8.261841653520567e21	0.0007083981118194416

9	9.003039097772565	5.1002248626897955e9	1.1966108418294833e22	0.003039097772564503
10	9.990574635182616	1.166611991635824e10	1.655343433307367e22	0.009425364817383652
11	11.022998173545064	3.127208130563727e10	2.246553909518422e22	0.02299817354506395
12	11.959433407930353	6.1418141120587135e10	2.89153886820604e22	0.040566592069646745
13	13.059504524010793	1.5300549815452048e11	3.7940459609615415e22	0.05950452401079254
14	13.935190089077999	3.0985557081238544e11	4.633597782215305e22	0.06480991092200128
15	15.053983018339713	5.010255098229966e11	5.875191008208141e22	0.05398301833971253
16	15.963909038160269	9.394123675655612e11	7.034662578633065e22	0.03609096183973115
17	17.01653936390092	2.3971581253245e12	8.555214312879987e22	0.0165393639009217
18	17.994397630685246	5.6709570563811105e12	1.0150796567230493e23	0.005602369314754441
19	19.00114510391641	7.741019369389632e12	1.1988901833144277e23	0.0011451039164107613
20	19.999888806572073	1.2178388919649346e13	1.401928756196897e23	0.00011119342792653697

Tabela 2: Uzyskane pierwiastki zaburzonego wielomianu i wartości w nich.

k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999999225 + 0.0im	10130.71275321466	1.4221252872463423e6	7.749356711883593e-14
2	2.000000000008296 + 0.0im	56308.28771988236	7.378697629606253e19	8.29603052920902e-12
3	3.0000000008906005 + 0.0im	807092.163108728	3.320413936717737e20	8.906004822506475e-10
4	3.9999999438558906 + 0.0im	8.084860855411538e6	8.854436741110107e20	5.614410936161107e-8
5	5.000001086829692 + 0.0im	3.879198510577002e7	1.844675710521722e21	1.0868296920207854e-6
6	5.9999948068494735 + 0.0im	1.4732257627258718e8	3.320404736791115e21	5.193150526494605e-6
7	6.999771695064889 + 0.0im	4.1411759028405327e8	5.422782649701153e21	0.0002283049351108346
8	8.006980931819212 + 0.0im	5.02184047848722e8	8.286826509116029e21	0.006980931819212444
9	8.917724186859578 + 0.0im	1.2260299344962182e9	1.16152393552561e22	0.08227581314042176
10	10.09498228100687 - 0.643555596973669im	1.711412152637466e9	1.7209911934799487e22	0.6505270479399019
11	10.09498228100687 + 0.643555596973669im	1.711412152637466e9	1.7209911934799487e22	1.1105047852610384
12	11.793686395399911 - 1.6525606512888438im	1.922343284948516e10	2.856699145630138e22	1.665389446835571
13	11.793686395399911 + 1.6525606512888438im	1.922343284948516e10	2.856699145630138e22	2.0460081179778995
14	13.992434729751384 - 2.5188744692410605im	2.78104055609644e11	4.934686280169574e22	2.518885830105916
15	13.992434729751384 + 2.5188744692410605im	2.78104055609644e11	4.934686280169574e22	2.712916579182928

16	16.73076220872404 - 2.8126245020564427im	1.0504983293359739e12	8.48472076147016e22	2.9060058491454366
17	16.73076220872404 + 2.8126245020564427im	1.0504983293359739e12	8.48472076147016e22	2.8254814771679904
18	19.50244401519765 - 1.9403279701116138im	6.884949792026641e12	1.3181950355066861e23	2.454019284846941
19	19.50244401519765 + 1.9403279701116138im	6.884949792026641e12	1.3181950355066861e23	2.004325976483215
20	20.84690820687191 + 0.0im	9.72908249032009e11	1.5911079405980614e23	0.8469082068719089

4.4 Wnioski

Na początku patrząc na wyniki, które otrzymaliśmy można śmiało rzec, że dla zaburzonego wielomianu żadne z policzonych zer wstawione jako argument nie zwróciło oczekiwanego wyniku (zera). Niedokładność opiera się na korelacji zbyt dużych współczynników z reprezentacją cyfr znaczących w odpowiedniej arytmetyce, która nie może zostać bezbłędnie zapisana ze względu na brak miejsca. Zaburzenie wyników w nieznacznym stopniu 2^{-23} znacząco wpłynęło na wyniki. Stąd wiemy, że zadanie jest źle uwarunkowane, czyli niewielkie względne zmiany danych zadania powodują duże względne zmiany jego rozwiązania. Patrząc na drugą tabelkę można dostrzec liczby zespolone, które w wyniku interpretujemy jako skutek marginalnych zaburzeń przy znajdowaniu pierwiastków wielomianu Wilkinsona. Przy czym wartości funkcji dla pierwiastków są duże.

5 Zadanie 5

5.1 Opis Problemu

Rozważmy równanie rekurencyjne (model logistyczny, model wzrostu populacji)

$$p_{n+1} := p_n + r p_n(1 - p_n), n = 0, 1, ..., (1)$$

gdzie r jest pewną daną stałą, r(1 - pn) jest czynnikiem wzrostu populacji, a p0 jest wielkością populacji stanowiąca procent maksymalnej wielkości populacji dla danego stanu środowiska. Przeprowadzić następujące eksperymenty: 1. Dla danych p0 = 0.01 i r = 3 wykonać 40 iteracji wyrażenia (1), a następnie wykonać ponownie 40 iteracji wyrażenia (1) z niewielką modyfikacją tj. wykonać 10 iteracji, zatrzymać, zastosować obcięcie wyniku odrzucając cyfry po trzecim miejscu po przecinku (daje to liczbę 0.722) i kontynuować dalej obliczenia (do 40- stej iteracji) tak, jak gdyby był to ostatni wynik na wyjściu. Porównać otrzymane wyniki. Obliczenia wykonać w arytmetyce Float32 (w języku Julia). 2. Dla danych p0 = 0.01 i r = 3 wykonać 40 iteracji wyrażenia (1) w arytmetyce Float32 i Float64 (w języku Julia). Porównać otrzymane wyniki

5.2 Rozwiązanie

Należało zaimplementować funkcję podaną z treści zadania, a następnie umiejętnie rozplanować pętle żeby wyliczyć populację.

5.3 Interpretacja wyniku

n	Float32	Float32 z obcięciem	Float64
0	0.01	0.01	0.01
1	0.0397	0.0397	0.0397
2	0.15407173	0.15407173	0.15407173000000002
3	0.5450726	0.5450726	0.5450726260444213
4	1.2889781	1.2889781	1.2889780011888006
5	0.1715188	0.1715188	0.17151914210917552
6	0.5978191	0.5978191	0.5978201201070994
7	1.3191134	1.3191134	1.3191137924137974
8	0.056273222	0.056273222	0.056271577646256565
9	0.21559286	0.21559286	0.21558683923263022
10	0.7229306	0.722	0.722914301179573
11	1.3238364	1.3241479	1.3238419441684408
12	0.037716985	0.036488414	0.03769529725473175
13	0.14660022	0.14195944	0.14651838271355924
14	0.521926	0.50738037	0.521670621435246
15	1.2704837	1.2572169	1.2702617739350768
16	0.2395482	0.28708452	0.24035217277824272
17	0.7860428	0.9010855	0.7881011902353041
18	1.2905813	1.1684768	1.2890943027903075
19	0.16552472	0.577893	0.17108484670194324
20	0.5799036	1.3096911	0.5965293124946907
21	1.3107498	0.09289217	1.3185755879825978
22	0.088804245	0.34568182	0.058377608259430724
23	0.3315584	1.0242395	0.22328659759944824
24	0.9964407	0.94975823	0.7435756763951792
25	1.0070806	1.0929108	1.315588346001072
26	0.9856885	0.7882812	0.07003529560277899
27	1.0280086	1.2889631	0.26542635452061003
28	0.9416294	0.17157483	0.8503519690601384
29	1.1065198	0.59798557	1.2321124623871897
30	0.7529209	1.3191822	0.37414648963928676
31	1.3110139	0.05600393	1.0766291714289444
32	0.0877831	0.21460639	0.8291255674004515
33	0.3280148	0.7202578	1.2541546500504441
34	0.9892781	1.3247173	0.29790694147232066
35	1.021099	0.034241438	0.9253821285571046
36	0.95646656	0.13344833	1.1325322626697856
37	1.0813814	0.48036796	0.6822410727153098
38	0.81736827	1.2292118	1.3326056469620293
39	1.2652004	0.3839622	0.0029091569028512065

5.4 Wnioski

Wyniki uzyskane po obliczeniach wskazują na wniosek, że obcinanie cyfr znaczących i zmiana arytmetyki przy wielu iteracjach bazując na modyfikowaniu poprzednich danych bardzo wpływa na wyniki końcowe. Dlatego też, gdy mamy równania rekurencyjne, które bazują na wynikach z poprzednich wywołań są bardzo narażone na wszelkie zmiany dokładności i arytmetyki.

6 Zadanie 6

6.1 Opis Problemu

Rozważmy równanie rekurencyjne $x_{n+1} := x_{2n} + c$ dla n = 0, 1, . . . , gdzie c jest pewną daną stałą. Przeprowadzić następujące eksperymenty.

Dla danych:
1. c = -2 i $x_0 = 1$
2. c = -2 i $x_0 = 2$
3. c = -2 i $x_0 = 1.99999999999999$
4. c = -1 i $x_0 = 1$
5. c = -1 i $x_0 = -1$
6. c = -1 i $x_0 = 0.75$
7. c = -1 i $x_0 = 0.25$

wykonać, w języku Julia w arytmetyce Float64, 40 iteracji wyrażenia (2). Zaobserwo- wać zachowanie generowanych ciągów. Wsk. Przeprowadzić iteracje graficzną $x_{n+1} := x_{2n} + c$.

6.2 Rozwiązanie

n	$x_0 = 1$	$x_0 = 2$	$x_0 = 1.99999999999999$
1	-1.0	2.0	1.99999999999996
2	-1.0	2.0	1.9999999999998401
3	-1.0	2.0	1.9999999999993605
4	-1.0	2.0	1.999999999997442
5	-1.0	2.0	1.9999999999897682
6	-1.0	2.0	1.9999999999590727
7	-1.0	2.0	1.999999999836291
8	-1.0	2.0	1.9999999993451638
9	-1.0	2.0	1.9999999973806553
10	-1.0	2.0	1.999999989522621
11	-1.0	2.0	1.9999999580904841
12	-1.0	2.0	1.9999998323619383
13	-1.0	2.0	1.9999993294477814
14	-1.0	2.0	1.9999973177915749
15	-1.0	2.0	1.9999892711734937
16	-1.0	2.0	1.9999570848090826
17	-1.0	2.0	1.999828341078044

18	-1.0	2.0	1.9993133937789613
19	-1.0	2.0	1.9972540465439481
20	-1.0	2.0	1.9890237264361752
21	-1.0	2.0	1.9562153843260486
22	-1.0	2.0	1.82677862987391
23	-1.0	2.0	1.3371201625639997
24	-1.0	2.0	-0.21210967086482313
25	-1.0	2.0	-1.9550094875256163
26	-1.0	2.0	1.822062096315173
27	-1.0	2.0	1.319910282828443
28	-1.0	2.0	-0.2578368452837396
29	-1.0	2.0	-1.9335201612141288
30	-1.0	2.0	1.7385002138215109
31	-1.0	2.0	1.0223829934574389
32	-1.0	2.0	-0.9547330146890065
33	-1.0	2.0	-1.0884848706628412
34	-1.0	2.0	-0.8152006863380978
35	-1.0	2.0	-1.3354478409938944
36	-1.0	2.0	-0.21657906398474625
37	-1.0	2.0	-1.953093509043491
38	-1.0	2.0	1.8145742550678174
39	-1.0	2.0	1.2926797271549244
40	-1.0	2.0	-0.3289791230026702

Table 6: Tabela: c = -1

n	$x_0 = 1$	$x_0 = -1$	$x_0 = 0.75$	$x_0 = 0.25$
1	0.0	0.0	-0.4375	-0.9375
2	-1.0	-1.0	-0.80859375	-0.12109375
3	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	0.0	0.0	-0.2253147218564956	-0.9991524699951226
6	-1.0	-1.0	-0.9492332761147301	-0.0016943417026455965
7	0.0	0.0	-0.0989561875164966	-0.9999971292061947
8	-1.0	-1.0	-0.9902076729521999	-5.741579369278327e-6
9	0.0	0.0	-0.01948876442658909	-0.9999999999670343
10	-1.0	-1.0	-0.999620188061125	-6.593148249578462e-11
11	0.0	0.0	-0.0007594796206411569	-1.0
12	-1.0	-1.0	-0.9999994231907058	0.0
13	0.0	0.0	-1.1536182557003727e-6	-1.0
14	-1.0	-1.0	-0.9999999999986692	0.0
15	0.0	0.0	-2.6616486792363503e-12	-1.0
16	-1.0	-1.0	-1.0	0.0
17	0.0	0.0	0.0	-1.0
18	-1.0	-1.0	-1.0	0.0

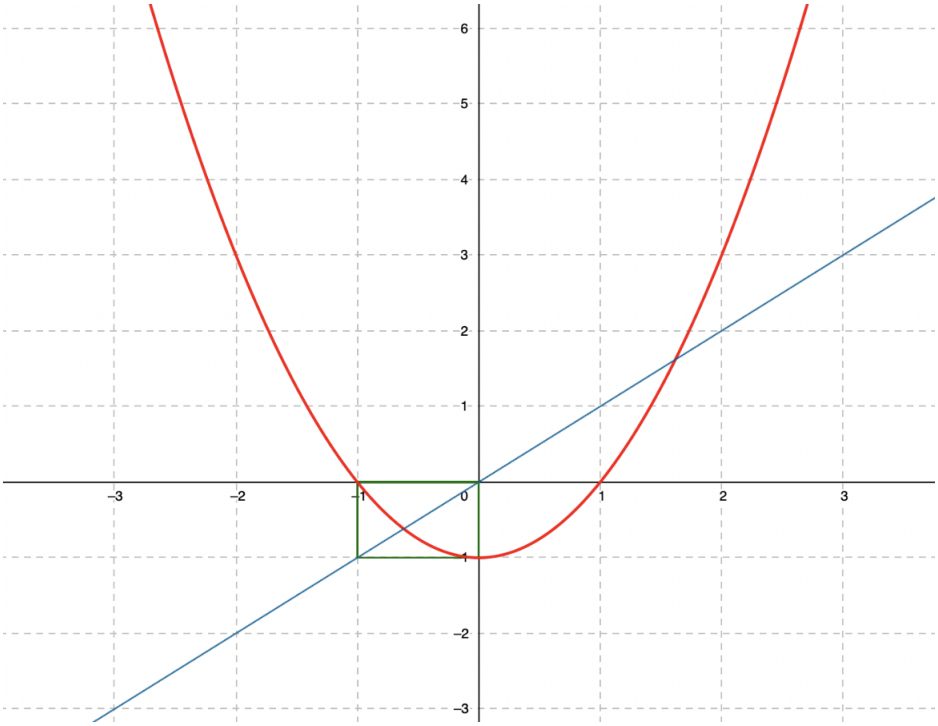
19	0.0	0.0	0.0	-1.0
20	-1.0	-1.0	-1.0	0.0
21	0.0	0.0	0.0	-1.0
22	-1.0	-1.0	-1.0	0.0
23	0.0	0.0	0.0	-1.0
24	-1.0	-1.0	-1.0	0.0
25	0.0	0.0	0.0	-1.0
26	-1.0	-1.0	-1.0	0.0
27	0.0	0.0	0.0	-1.0
28	-1.0	-1.0	-1.0	0.0
29	0.0	0.0	0.0	-1.0
30	-1.0	-1.0	-1.0	0.0
31	0.0	0.0	0.0	-1.0
32	-1.0	-1.0	-1.0	0.0
33	0.0	0.0	0.0	-1.0
34	-1.0	-1.0	-1.0	0.0
35	0.0	0.0	0.0	-1.0
36	-1.0	-1.0	-1.0	0.0
37	0.0	0.0	0.0	-1.0
38	-1.0	-1.0	-1.0	0.0
39	0.0	0.0	0.0	-1.0
40	-1.0	-1.0	-1.0	0.0

Table 7: Tabela: c = -2

6.3 Wnioski i Interpretacja wyniku

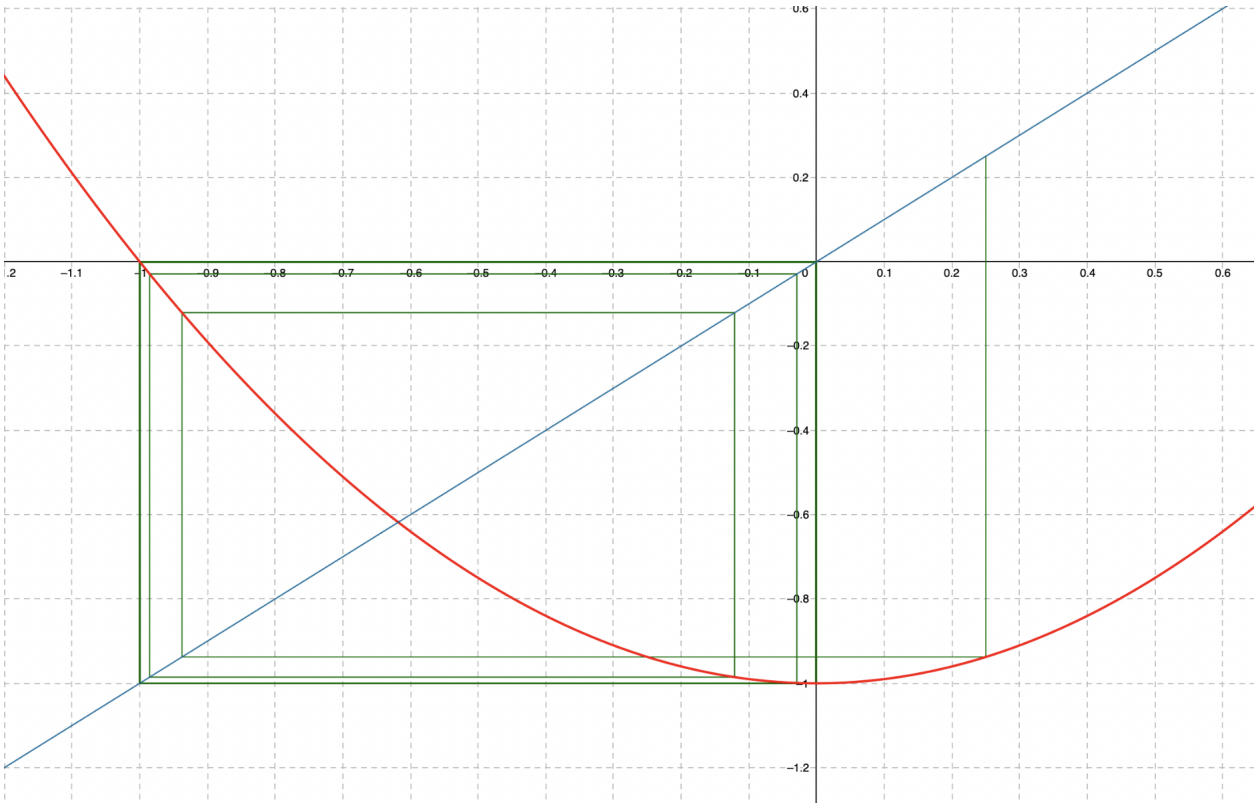
Wyznaczam iterację graficzną. Jak to zrobić?

Do przeprowadzenia iteracji trzeba najpierw wyznaczyć wykres funkcji y — ax(1 —x) oraz dwusieczną (przekątną kwadratu). Następnie należy zaznaczyć punkt x_q na osi x i przeprowadzić linię pionową wychodzącą z punktu x_0, a kończącą się w momencie przecięcia wykresu funkcji. Od tego punktu zaczynamy rysować linię poziomą do punktu przecięcia z przekątną, a stąd znowu linię pionową do punktu przecięcia z wykresem itd.



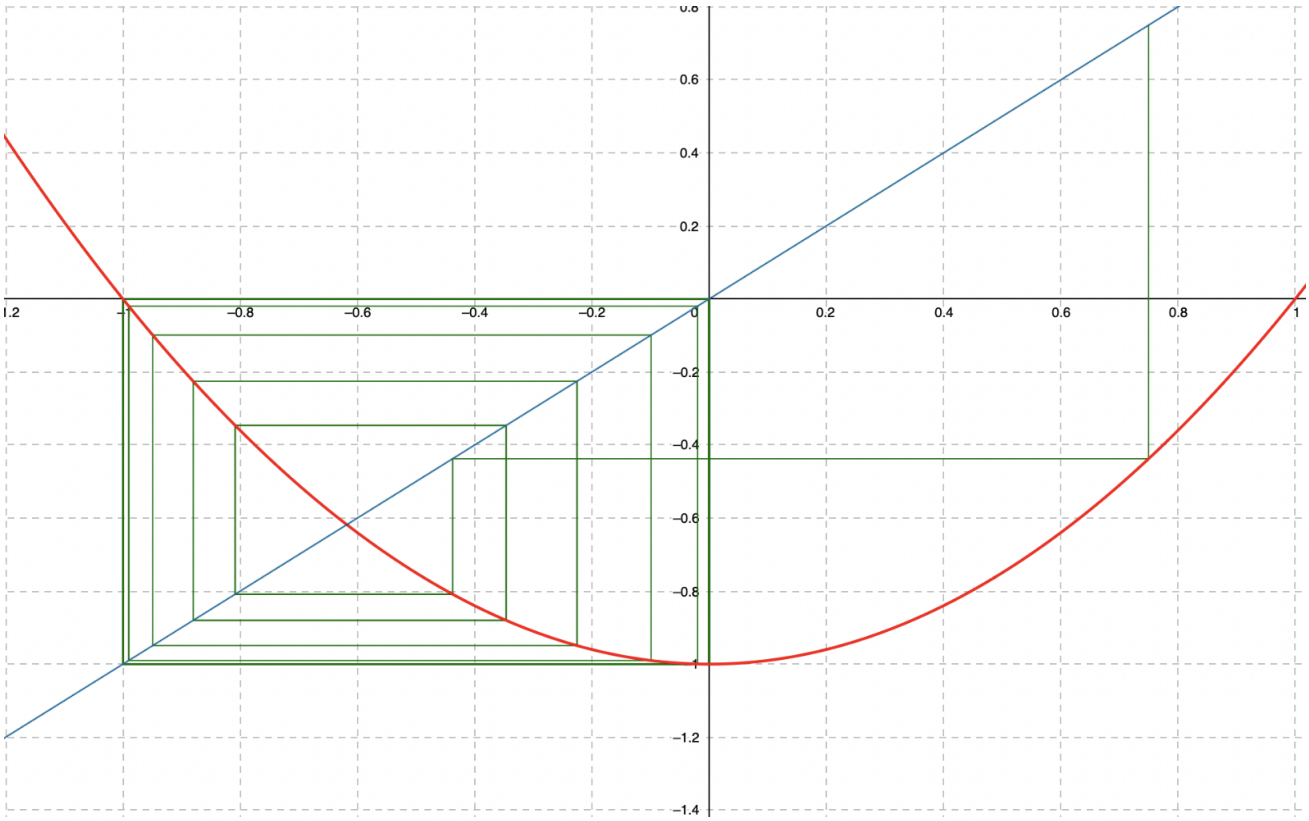
$$x_{n+1} := x_n^2 - 1 \text{ dla } x_0 = -1$$

Jak widzimy dla -1 widać zacyklenie się procesu. Jest one wręcz natychmiastowe. Oznacza to, że jest stabilne.



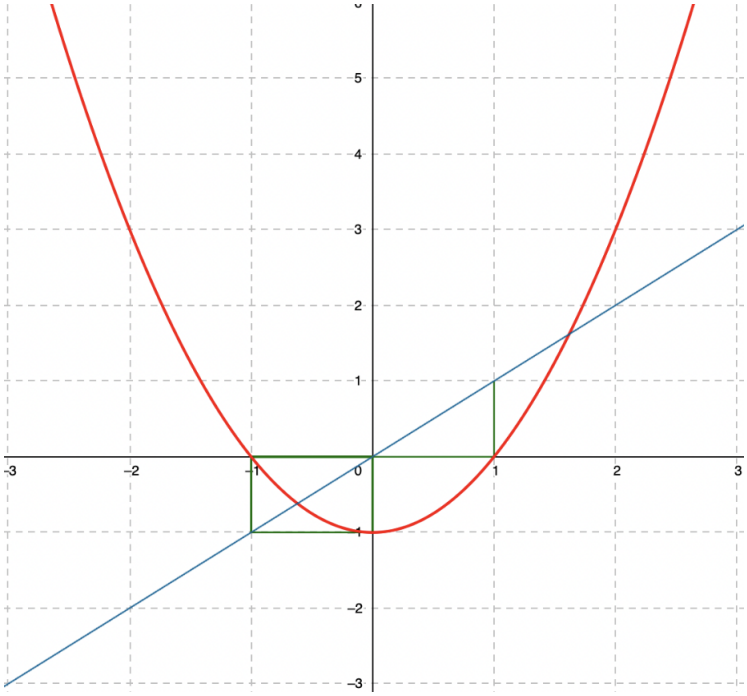
$$x_{n+1} := x_n^2 - 1 \text{ dla } x_0 = 0.25$$

Dla 0.25 powoli widać zacyklanie się procesu. Tzn. powolną stabilizację.



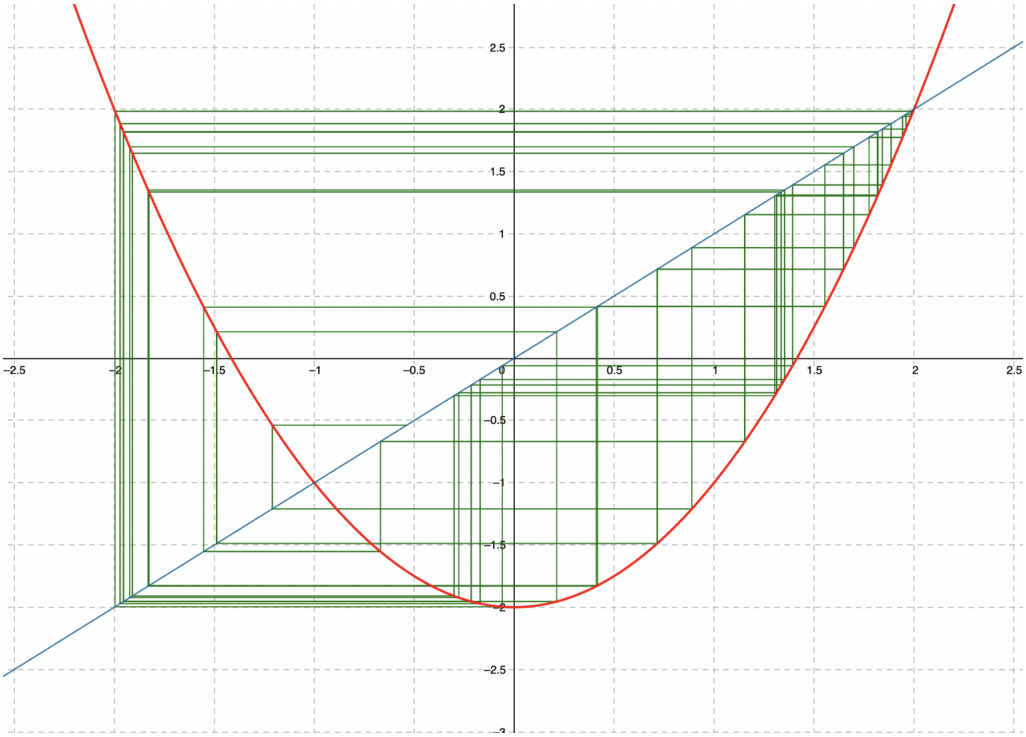
$x_{n+1} := x_n^2 - 1$ dla $x_0 = 0.75$

Jeszcze mocniejsze zacyklanie. Nieskończony cykl.



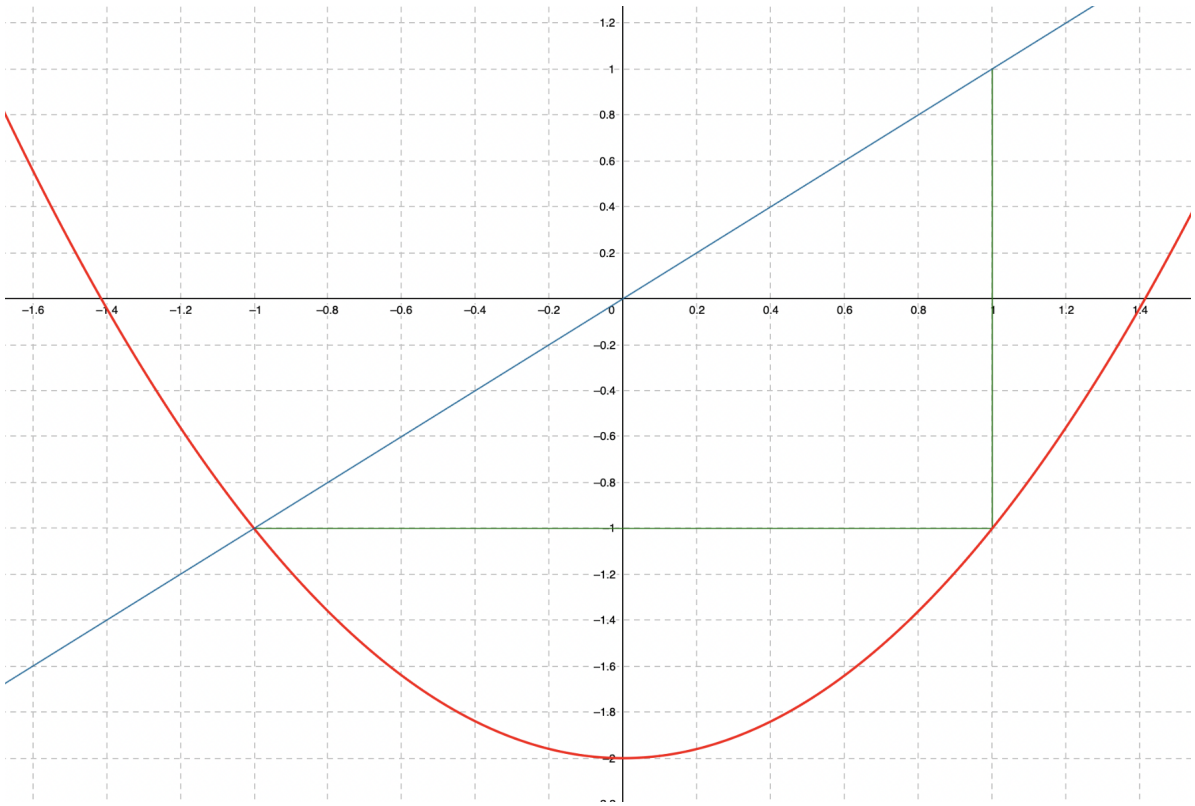
$x_{n+1} := x_n^2 - 1$ dla $x_0 = 1$

Jak widzimy dla 1 widać zacyklenie się procesu. Jest one wręcz natychmiastowe. Oznacza to, że jest stabilne.



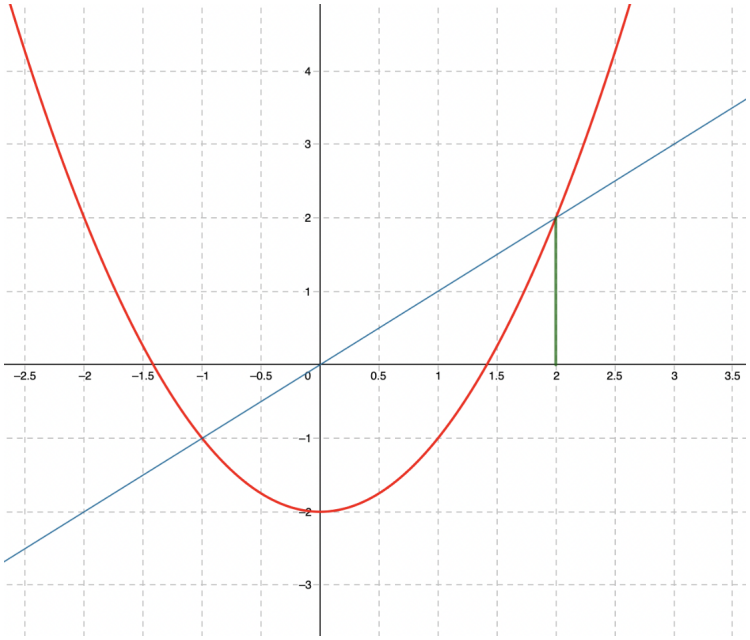
$$x_{n+1} := x_n^2 - 2 \text{ dla } x_0 = 1.9999999999999$$

Patrząc na uzyskany wykres widzimy, że proces wyraźnie nie zatrzymuje się. Zamiast uporządkowanego zachowania w granicy stopniowo (w kolejnych iteracjach jest mocniej zapełniony) zapełnia on (powoli) całą dostępną przestrzeń. Zjawisko to, nazywane mieszaniem, jest wskaźnikiem niestabilnego stanu systemu. Spoglądając na dane z tabeli znajdziemy potwierdzenie naszej graficznej reprezentacji.



$$x_{n+1} := x_n^2 - 2 \text{ dla } x_0 = 1$$

Zielone odcinki rysują się do punktów stałych w funkcji. Patrząc na tabelę możemy zinterpretować to jako brak zmian w kolejnych wartościach ciągu.



$$x_{n+1} := x_n^2 - 2 \text{ dla } x_0 = 2$$

Tak samo jak w poprzednim przykładzie.

Otrzymane wyniki dla całkowitych wartości c i x_0 nie zaskakują i są zgodne z intuicją. Przy liczbach zmiennoprzecinkowych takich jak np. $x_0 = 0.25$ i $x_0 = 0.75$ wartości zbiegają do całkowitych przez pewien czas, a potem po 40 iteracjach zwracają taką wartość, że jej zachowanie wygląda podobnie jakbyśmy przybrali wartości $x_0 = 1$ i $x_0 = 2$. Przybierając $x_0 = 1.99999999999999$ znacząco wpływamy na wyniki po 40 iteracjach. Wtedy mamy sytuację, że zamiast oczekiwanych wyników gdzie powinniśmy otrzymywać wyniki oddalone od zera to się do niego przybliża. Jest to całkowita rozbieżność w wynikach, które oczekujemy, a wynikami otrzymywanymi.

Reasumując, gdy dane są wartościami całkowitymi ciągi mają poprawne wartości. W przypadku, gdy dane są ułamkami dla małych x rekurencja wyznacza poprawne wartości, ale jak widać na wykresach od pewnego momentu te wartości są zaokrąglane do wartości całkowitych. Najpierw wartości są całkowite i od pewnego momentu liczby przestają być zaokrąglane do wartości całkowitych. Wyznaczanie kolejnych wyrazów ciągów rekurencyjnych może mieć różną stabilność. Zależy to od parametrów, które ustawimy. Całkowicie stabilne to np. $x_0 = 1$, stabilizujące się np. $x_0 = 0.75, x_0 = 0.25$, niestabilne np. $x_0 \approx 1.99$.