Vadym Liss

257264

Zadanie1

Powtórzyć zadanie 5 z listy 1, ale usunąć ostatnią 9 z x_4 i ostatnią 7 z x_5.

a)

arytmetyka	x1*y	x2*y
Float32	-0.4999443	-0.4999443
Float64	1.0251881368296672e-10	-0.004296342739891585

b)

arytmetyka	x1*y	x2*y
Float32	-0.4543457	-0.4543457
Float64	-1.5643308870494366e-10	-0.004296342998713953

c)

arytmetyka	x1*y	x2*y
Float32	-0.5	-0.5
Float64	0.0	-0.004296342842280865

d)

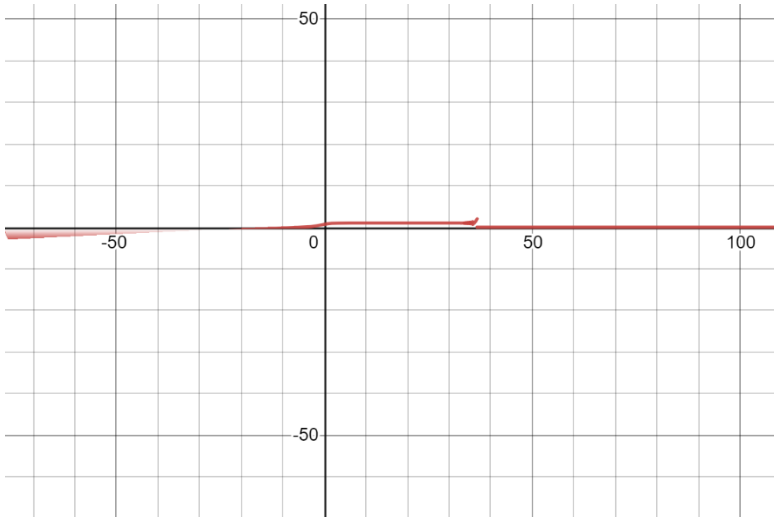
arytmetyka	x1*y	x2*y
Float32	-0.5	-0.5
Float64	0.0	-0.004296342842280865

Wniosek: Na podstawie powyższych eksperymentów można zauważyć, że niewielka zmiana danych wpływa na otrzymywane wyniki. W przypadku Float32 zmiany są praktycznie niewidoczne, ponieważ dokonana perturbacja danych pojawiła się na bitach, które nie mają znaczenia w tej arytmetyce (powodem jest jej precyzja). Natomiast w przypadku arytmetyki Float64, dokonana zmiana ma wielki wpływ na wynik, różnica między wynikami sięga 10 rzędów! Dzieje się tak, ponieważ wskaźnik uwarunkowania zadania policzenia iloczynu skalarnego wektorów x, y wyraża się wzorem $\frac{\sum_{i=1}^{n}|x_i y_i|}{|\sum_{i=1}^{n} x_i y_i|}$ natomiast wektory z zadania są prawie prostopadłe, zatem dół ułamka będzie dążył do 0, a co za tym idzie całe wyrażenie do nieskończoności. Zatem zadanie to jest zatem źle uwarunkowane, ponieważ niewielkie względne zmiany danych powodują duże względne odkształcenia wyników.

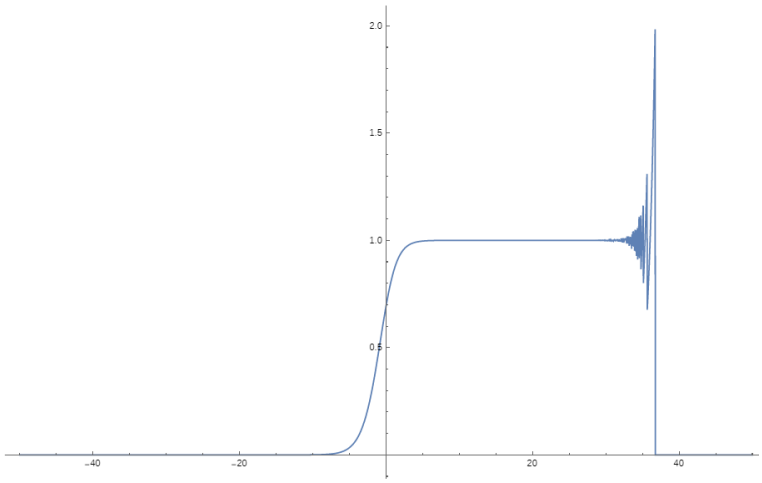
Zadanie2

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 i wyjaśnić zjawisko.

Desmos:



Wolfram Cloud:



Wniosek: w przedziale x ∈ [30, 40] wykresy różnią się i nie są do końca prawdziwe. By zweryfikować wykresy, policzyłem granicę funkcji w nieskończoności:

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

Zatem oba programy nie są odporne na błędy obliczeń. Dla dużych wartości x wyrażenie $1 + e^{-x}$ wynosi 1. Skoro dwa popularne programy nie poradziły sobie z tym problemem można stwierdzić, że jest on trudny bądź niemożliwy do uniknięcia.

Zadanie3

Rozważmy zadanie rozwiązywania układu równań liniowych ***Ax = b*** *dla danej macierzy* współczynników **A** ∈ $\mathbb{R}^{n\times n}$ *i wektora prawych stron* **b** ∈ $\mathbb{R}^n$.

Macierz Hilberta:

n	rank(A)	cond(A)	eliminacja gaussa	inv
2	2	19.28147006790397	5.661048867003676e-16	1.4043333874306803e-15
3	3	524.0567775860644	8.022593772267726e-15	0.0
4	4	15513.73873892924	4.4515459601812086e-13	4.0996530221808757e-13
5	5	476607.25024331047	1.6828426299227195e-12	3.2543043465682462e-12
6	6	1.495105864125091e7	2.618913302311624e-10	1.2883031376157472e-10
7	7	4.7536735647344047e8	1.1328142969056265e-8	4.032805581748429e-9
8	8	1.525757551642611e10	3.539844668926116e-7	3.862822472100174e-7
9	9	4.931539500738534e11	6.015315045425784e-6	1.137323420456882e-5
10	10	1.6024868379056498e13	0.00017899791220199793	0.00045037990900565275
11	10	5.223733117324263e14	0.008044067151200653	0.010953978875396859
12	11	1.7341930983785612e16	0.23809370828812224	0.7646686036530203
13	11	1.1522843303590932e18	57.83000687827706	35.15351393266081
14	11	6.06112400617363e17	2.982952532671996	4.875801223197704
15	12	6.504049098176895e17	3.06964399281068	22.079244508503688
16	12	3.498527647064917e17	18.63410314277541	18.63410314277541
17	12	5.046792958377091e17	1.6664937063354397	13.57336424125635
18	12	1.0190210372627103e18	55.65965311013928	70.63403951454899
19	13	1.1330805457366684e18	16.810267879293978	14.650337924995926
20	13	2.5382496382297713e18	21.403720415848714	109.81438885705134
21	13	3.085325530528195e18	14.131586614264583	61.56657777614069
22	13	1.8512631782844216e18	285.2622854691671	223.34823988261655
23	13	1.1448363665742652e18	23.698931371171994	26.244669007643207
24	13	4.2781222739562583e18	12.664694162539185	13.860939760430663
25	13	6.0919576838107685e19	28.046639859820335	30.218040979040115
26	14	2.0040729185184236e18	34.836394727496724	65.90131453638766
27	14	5.004966870318743e18	14.207882496312248	21.36570003109647
28	14	5.97438732511109e18	18.56568607769768	27.896223281690236
29	14	2.726491576645145e19	21.993500810059164	33.98938404618491
30	14	2.4515310428313155e18	67.54658072741391	65.16287386394825

Macierz losowa:

n	rank(A)	cond(A)	eliminacja gaussa	inv
5	5	1.0000000000000007	1.719950113979703e-16	2.432376777795247e-16
5	5	9.999999999999995	1.3136335981433191e-16	2.482534153247273e-16
5	5	999.999999999986	6.121482228636979e-14	7.169090995165241e-14
5	5	1.0000000002008274e7	5.334237472387399e-11	1.594082445095893e-11
5	5	1.0000902672721001e12	3.264461609936889e-5	3.245856839932655e-5
5	4	9.089764276515404e15	0.13119624866857832	0.1956559480312316
10	10	1.0000000000000013	2.482534153247273e-16	1.890641683868921e-16
10	10	10.0	2.1065000811460205e-16	2.696722356863272e-16
10	10	999.9999999999707	6.107939166631972e-14	5.920893397792411e-14
10	10	9.999999997007772e6	1.147530324466724e-10	1.1111032898084421e-10
10	10	1.0000374783591967e12	2.0477680098558766e-5	1.8393804094301138e-5
10	9	8.796002223592517e15	0.08913038031526217	0.10539638721275033
20	20	1.0000000000000013	5.617333549722722e-16	5.484097192022683e-16
20	20	9.999999999999982	4.977465500091126e-16	4.971270767759187e-16
20	20	1000.0000000000276	2.2989821141483903e-14	1.7900706681602682e-14
20	20	9.999999998100726e6	1.910010328305373e-11	4.441464088356969e-11
20	20	1.0000137138021631e12	2.1423025386914042e-5	1.9956646258987636e-5
20	19	4.967842488947862e15	0.04636334924349563	0.07714225647707616

Wniosek: Dla macierzy Hilberta wraz z rozmiarem macierzy rośnie wskaźnik uwarunkowania macierzy, a co za tym idzie rosną błędy obliczeniowe dla obydwu metod. Rośnie również rząd macierzy. Dla macierzy losowej błędy również wzrastają razem z rozmiarem i wskaźnikiem uwarunkowania macierzy, jednak nie w takim tempie jak błędy przybliżeń dla macierzy Hilberta, a rząd macierzy jest prawie zawsze równy *n*. Zadanie pokazuje, że zadanie rozwiązania układu równań liniowych $\boldsymbol{Ax = b}$ jest źle uwarunkowane dla macierzy Hilberta.

Zadanie4

a) *Użyć funkcji roots (z pakietu Polynomials) do obliczenia 20 zer wielomianu P w postaci naturalnej podanej w zadaniu.*

K	z_k	p(z_k)	P(z_k)|	|z_k-k|
1	0.9999999999996989	36626.425482422805	36352.0	3.0109248427834245e-13
2	2.0000000000283182	181303.93367257662	181760.0	2.8318236644508943e-11
3	2.9999999995920965	290172.2858891686	209408.0	4.0790348876384996e-10
4	3.9999999837375317	2.0415372902750901e6	3.106816e6	1.626246826091915e-8
5	5.000000665769791	2.0894625006962188e7	2.4114688e7	6.657697912970661e-7
6	5.999989245824773	1.1250484577562995e8	1.20152064e8	1.0754175226779239e-5
7	7.000102002793008	4.572908642730946e8	4.80398336e8	0.00010200279300764947
8	7.999355829607762	1.5556459377357383e9	1.682691072e9	0.0006441703922384079
9	9.002915294362053	4.687816175648389e9	4.465326592e9	0.002915294362052734
10	9.990413042481725	1.2634601896949205e10	1.2707126784e10	0.009586957518274986
11	11.025022932909318	3.300128474498415e10	3.5759895552e10	0.025022932909317674
12	11.953283253846857	7.388525665404988e10	7.216771584e10	0.04671674615314281
13	13.07431403244734	1.8476215093144193e11	2.15723629056e11	0.07431403244734014
14	13.914755591802127	3.5514277528420844e11	3.65383250944e11	0.08524440819787316

15	15.075493799699476	8.423201558964254e11	6.13987753472e11	0.07549379969947623
16	15.946286716607972	1.570728736625802e12	1.555027751936e12	0.05371328339202819
17	17.025427146237412	3.3169782238892363e12	3.777623778304e12	0.025427146237412046
18	17.99092135271648	6.34485314179128e12	7.199554861056e12	0.009078647283519814
19	19.00190981829944	1.228571736671966e13	1.0278376162816e13	0.0019098182994383706
20	19.999809291236637	2.318309535271638e13	2.7462952745472e13	0.00019070876336257925

Wniosek: Zaimplementowana przeze mnie funkcja poly(x) realizująca wielomian *p* daje złe wyniki, odbiegające od 0, ale podobne do wartości ***p***. Wartości ***p*** też są mocno nieprawidłowe, różnica między dobrym wynikiem, a tym otrzymanym jest większa o kilkanaście rzędów niż różnica między obliczonym pierwiastkiem a faktycznym. Błędy w tym eksperymencie wynikają z ograniczeń architektury Float64, w której możliwe jest zapisanie tylko 15-17 cyfr znaczących w systemie dziesiętnym.

b) *Powtórzyć eksperyment Wilkinsona, tj. zmienić współczynnik −210 na* $-210-2^{-23}$

K	z_k	p(z_k)	P(z_k)	|z_k-k|
1	0.9999999999998357 + 0.0im	19987.872313406835	20496.0	1.6431300764452317e-13
2	2.0000000000550373 + 0.0im	352369.4138087958	339570.0	5.503730804434781e-11
3	2.99999999660342 + 0.0im	2.4162415582518433e6	2.2777455e6	3.3965799062229962e-9
4	4.000000089724362 + 0.0im	1.1263702300292023e7	1.0488020625e7	8.972436216225788e-8
5	4.99999857388791 + 0.0im	4.475744423806908e7	4.1239073125e7	1.4261120897529622e-6
6	6.000020476673031 + 0.0im	2.1421031658039317e8	1.406328934140625e8	2.0476673030955794e-5
7	6.99960207042242 + 0.0im	1.7846173427860644e9	4.122812662421875e8	0.00039792957757978087
8	8.007772029099446 + 0.0im	1.8686972170009857e10	1.030790127257812 5e9	0.007772029099445632
9	8.915816367932559 + 0.0im	1.3746309775142993e11	2.1574055781816406e9	0.0841836320674414
10	10.095455630535774 + 0.6449328236240688im	1.490069535200058e12	9.3841476056471 82e9	0.6519586830380407
11	10.095455630535774 + 0.6449328236240688im	1.490069535200058e12	9.3841476056471 82e9	1.1109180272716561
12	11.793890586174369 + 1.6524771364075785im	3.2962792355717145e13	3.0012060598372482e10	1.665281290598479
13	11.793890586174369 + 1.6524771364075785im	3.2962792355717145e13	3.0012060598372482e10	2.0458202766784277
14	13.992406684487216 + 2.5188244257108443im	9.546022365750216e14	2.0030917431984006e11	2.518835871190904
15	13.992406684487216 + 2.5188244257108443im	9.546022365750216e14	2.0030917431984006e11	2.7128805312847097
16	16.73074487979267 - 2.812624896721978im	2.742106076928478e16	1.1583329328642004e12	2.9060018735375106
17	16.73074487979267 + 2.812624896721978im	2.742106076928478e16	1.1583329328642004e12	2.825483521349608
18	19.5024423688181 - 1.940331978642903im	4.2524858765203725e17	5.867381806750561e12	2.4540214463129764
19	19.5024423688181 + 1.940331978642903im	4.2524858765203725e17	5.867381806750561e12	2.0043294443099486
20	20.84691021519479 + 0.0im	1.37437435599976e18	9.550552334336e12	0.8469102151947894

Wniosek: Zaburzenie współczynnika ***-210*** spowodowało, że obliczone przez funkcję roots pierwiastki są zespolone. Zadanie jest na tyle źle uwarunkowane, że aż wypadliśmy z dziedziny rozwiązania. Naturalnie również wzrosły różnice między otrzymanymi pierwiastkami, a tymi faktycznymi.

Zadanie5

Mamy równanie rekurencyjne: $p_{n+1} := p_n + rp_n\,(1 - p_n)$*, dla n = 0, 1, . . .,*

gdzie r jest pewną daną stałą, $r\,(1 - p_n)$ *jest czynnikiem wzrostu populacji, a* p_0 *jest wielkością populacji stanowiąca procent maksymalnej wielkości populacji dla danego stanu środowiska.*

a) *Dla danych p0 = 0.01 i r = 3 wykonujemy 40 iteracji wyrażenia, a następnie wykonujemy ponownie 40 iteracji wyrażenia (1) z niewielką modyfikacją tj. wykonujemy 10 iteracji, zatrzymujemy i zastosujemy obcięcie wyniku odrzucając cyfry po trzecim miejscu po przecinku*

I	Symulacja 1	Symulacja 2
1	0.0397	0.0397
2	0.15407173	0.15407173
3	0.5450726	0.5450726
4	1.2889781	1.2889781
5	0.1715188	0.1715188
6	0.5978191	0.5978191
7	1.3191134	1.3191134
8	0.056273222	0.056273222
9	0.21559286	0.21559286
10	0.7229306	0.722
11	1.3238364	1.3241479
12	0.037716985	0.036488414
13	0.14660022	0.14195944
14	0.521926	0.50738037
15	1.2704837	1.2572169
16	0.2395482	0.28708452
17	0.7860428	0.9010855
18	1.2905813	1.1684768
19	0.16552472	0.577893
20	0.5799036	1.3096911
21	1.3107498	0.09289217
22	0.088804245	0.34568182
23	0.3315584	1.0242395
24	0.9964407	0.94975823
25	1.0070806	1.0929108
26	0.9856885	0.7882812
27	1.0280086	1.2889631
28	0.9416294	0.17157483
29	1.1065198	0.59798557
30	0.7529209	1.3191822
31	1.3110139	0.05600393
32	0.0877831	0.21460639
33	0.3280148	0.7202578
34	0.9892781	1.3247173
35	1.021099	0.034241438
36	0.95646656	0.13344833
37	1.0813814	0.48036796
38	0.81736827	1.2292118
39	1.2652004	0.3839622
40	25860548	1.093568

b) *Dla danych p0 = 0.01 i r = 3 wykonuję 40 iteracji wyrażenia w arytmetyce Float32 i Float64:*

I	Float 32	Float 64
1	0.0397	0.0397
2	0.15407173	0.15407173000000002
3	0.5450726	0.5450726260444213
4	1.2889781	1.2889780011888006
5	0.1715188	0.17151914210917552
6	0.5978191	0.5978201201070994
7	1.3191134	1.3191137924137974
8	0.056273222	0.056271577646256565
9	0.21559286	0.21558683923263022
10	0.7229306	0.722914301179573
11	1.3238364	1.3238419441684408

12	0.037716985	0.03769529725473175
13	0.14660022	0.14651838271355924
14	0.521926	0.521670621435246
15	1.2704837	1.2702617739350768
16	0.2395482	0.24035217277824272
17	0.7860428	0.7881011902353041
18	1.2905813	1.2890943027903075
19	0.16552472	0.17108484670194324
20	0.5799036	0.5965293124946907
21	1.3107498	1.3185755879825978
22	0.088804245	0.058377608259430724
23	0.3315584	0.22328659759944824
24	0.9964407	0.7435756763951792
25	1.0070806	1.315588346001072
26	0.9856885	0.07003529560277899
27	1.0280086	0.26542635452061003
28	0.9416294	0.8503519690601384
29	1.1065198	1.2321124623871897
30	0.7529209	0.37414648963928676
31	1.3110139	1.0766291714289444
32	0.0877831	0.8291255674004515
33	0.3280148	1.2541546500504441
34	0.9892781	0.29790694147232066
35	1.021099	0.9253821285571046
36	0.95646656	1.1325322626697856
37	1.0813814	0.6822410727153098
38	0.81736827	1.3326056469620293
39	1.2652004	0.0029091569028512065
40	0.25860548	0.011611238029748606

Wniosek: Na podstawie powyższych eksperymentów można zaobserwować, że zarówno wybór arytmetyki, jak i drobna manipulacja danych wpływa na otrzymywany wynik. Analizując wyniki pierwszego eksperymentu, można zauważyć, że mamy do czynienia z numeryczną niestabilnością. Raz popełniony błąd (obcięcie wyniku) kumuluje się, powodując poważną utratę dokładności obliczeń. Drugi eksperyment obrazuje fakt, że dokładność obliczeń ściśle zależy od arytmetyki. Wyniki zwracane przez Float64 są dokładniejsze. Także widać jak duże znaczenie ma zaburzenie wyniku pośredniego dla wyniku końcowego.

Zadanie6

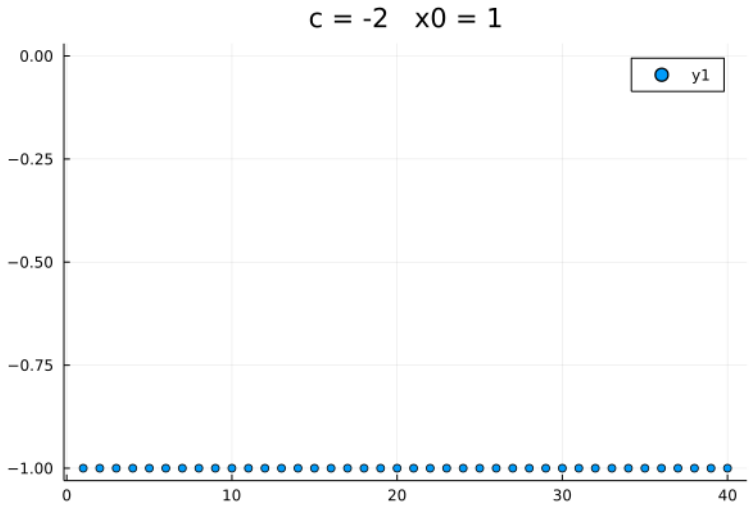
Rozważmy równanie rekurencyjne

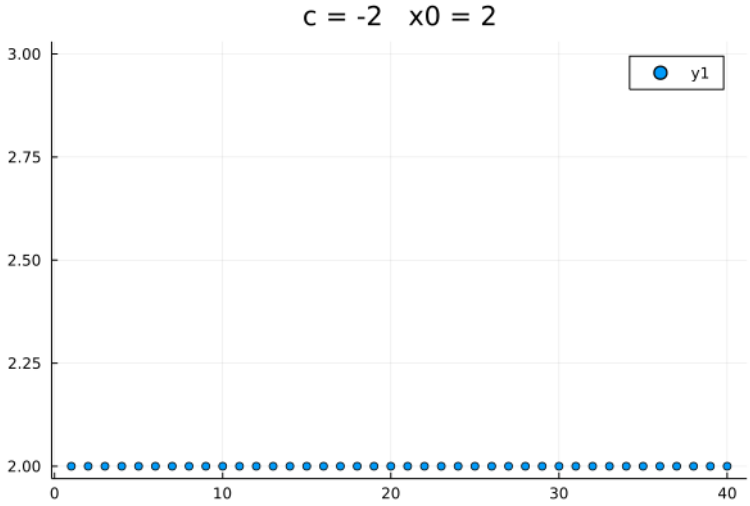
$$x_{n+1} := x_n^2 + c \; dla \; n = 0,1, \dots ,$$

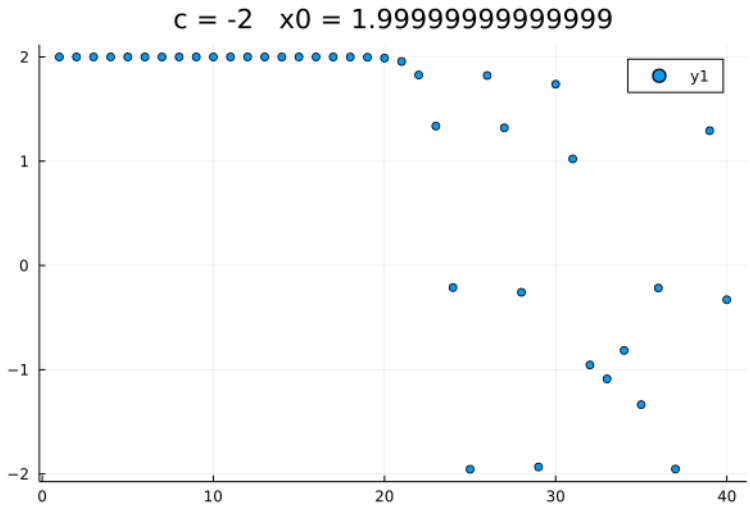
gdzie c jest pewną daną stałą. Przeprowadzić eksperymenty dla podanych danych w arytmetyce Float64:

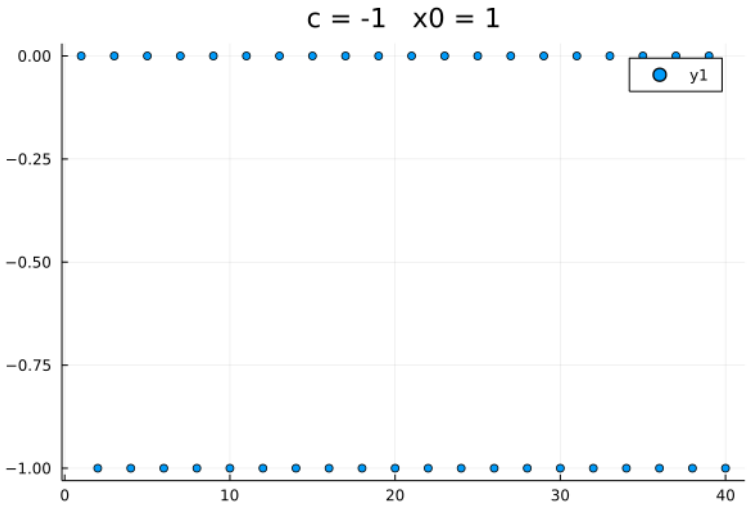
1. c = −2 i x_0 = 1
2. c = −2 i x_0 = 2
3. c = −2 i x_0 = 1.99999999999999
4. c = −1 i x_0 = 1
5. c = −1 i x_0 = −1
6. c = −1 i x_0 = 0.75
7. c = −1 i x_0 = 0.25

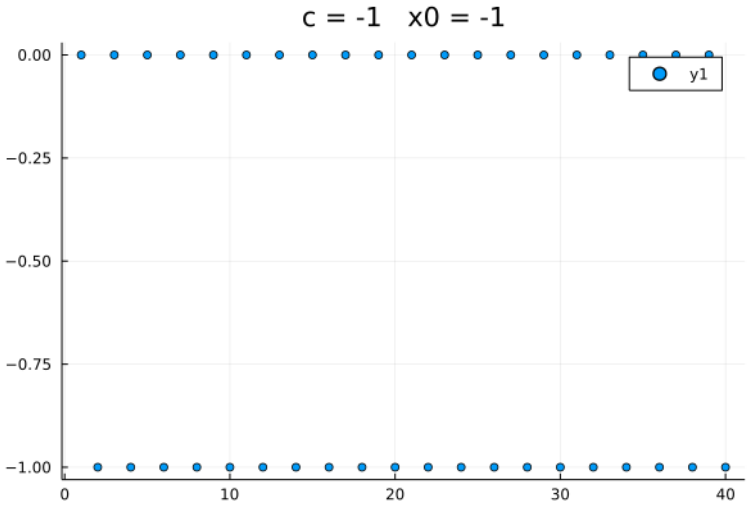
W plikach, gdzie nazwa to jest **`c + x₀.txt`** podaję dokładne wyniki obliczeń. Nie ma sensu robić tego tutaj w tablicach, bo dla niektórych par zajdzie coś takiego, że wynik będzie taki sami dla wszystkich iteracji. Także graficzna reprezentacja jest lepsza do interpretacji oraz wywnioskowania wniosków.

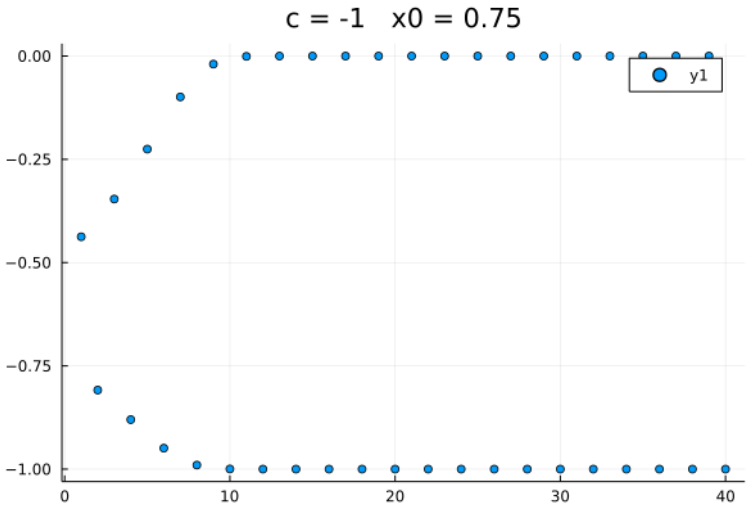

c = -2 x0 = 1
y1
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40

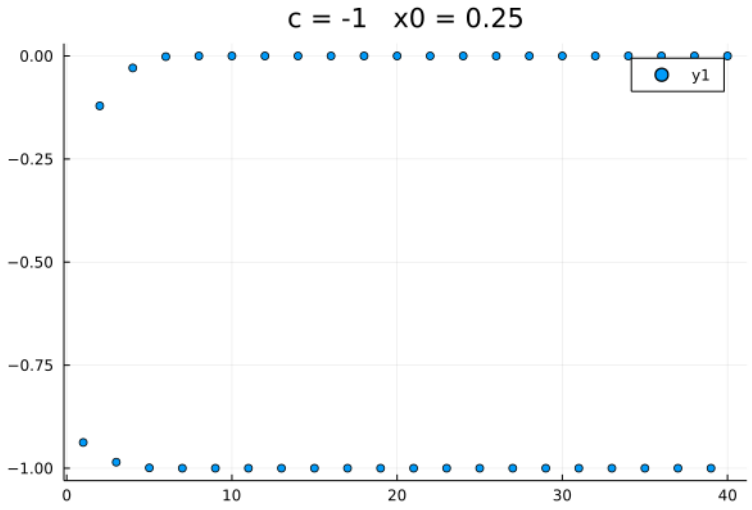

c = -2 x0 = 2
y1
3.00
2.75
2.50
2.25
2.00
0
10
20
30
40


c = -2 x0 = 1.99999999999999
y1
2
1
0
−1
−2
0
10
20
30
40


c = -1 x0 = 1
y1
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40


c = -1 x0 = -1
y1
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40


c = -1 x0 = 0.75
y1
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40



Wniosek: W tym zadaniu, podobnie jak w poprzednim badamy zjawisko stabilności numerycznej. Przypadek trzeci *(c = -2 x_0 = 1.99999999999999)* dostarcza nam najwięcej praktycznych informacji (reszta przypadków jest stabilna), ponieważ to w tym przypadku obserwujemy numeryczną niestabilność. Małe błędy nawarstwiają się powodując, że punkty należące do wykresu, od pewnego momentu są bardzo "rozrzucone". Zauważmy, że dla danych *(c = -2 x_0 = 1.99999999999999)* wyniki kompletnie się rozjeżdżają i są rozproszone - można było się spodziewać czegoś podobnego do wykresu dla danych *(c = -2 x_0 = 2)*, ponieważ mają takie samo $\boldsymbol{c}$ i bardzo bliskie $\boldsymbol{x_0}$, tymczasem okazało się, że niewielkie zaburzenie danych wejściowych miało duży wpływ na wynik. Dla danych *(c = −1 i x_0 = 0.75) i (c = −1 i x_0 = 0.25)* z kolei po kilku iteracjach pojawił się błąd zaokrąglenia, przez co wykres wpadł w cykl {-1, 1}. Zadanie jest zatem źle uwarunkowane, w szczególności dla danych w przykładzie 3.