Obliczenia naukowe

Paweł Kędzierski 272400

Lista 2

Zadanie 1.

Powtórz zadanie 5 z listy 1, ale usuń ostatnią cyfrę 9 z x_4 i ostatnią cyfrę 7 z x_5. Jaki wpływ na wyniki mają niewielkie zmiany danych?
Funkcja `1`: liczy „w przód"'
Funkcja `2`: liczy „w tył"
Funkcja `3`: od największego do najmniejszego
Funkcja `4`: od najmniejszego do największego

Sposób	Float32 stare	Float32 nowe	Float64 stare	Float64 nowe
1	-0.4999443	-0.4999443	1.0251881368296672e-10	-0.004296342739891585
2	-0.4543457	-0.4543457	-1.5643308870494366e-10	-0.004296342998713953
3	-0.5	-0.5	0.0	-0.004296342842280865
4	-0.5	-0.5	0.0	-0.004296342842280865

gdzie wartość prawidłowa wynosi:

`-0.004296343192495245`

Wyniki dla typu `Float32` nie uległy zmianie, ponieważ jest to niewystarczająca do zauważenia różnicy precyzja zapisu liczby zmiennopozycyjnej w tym typie.

Inną sytuacje można zaobserwować w typie `Float64`, gdzie różnica jest znaczna mimo że dokonaliśmy tak niewielkiej zmiany danych. W tej sytuacji osiągnięte wyniki są znacznie bliższe

---

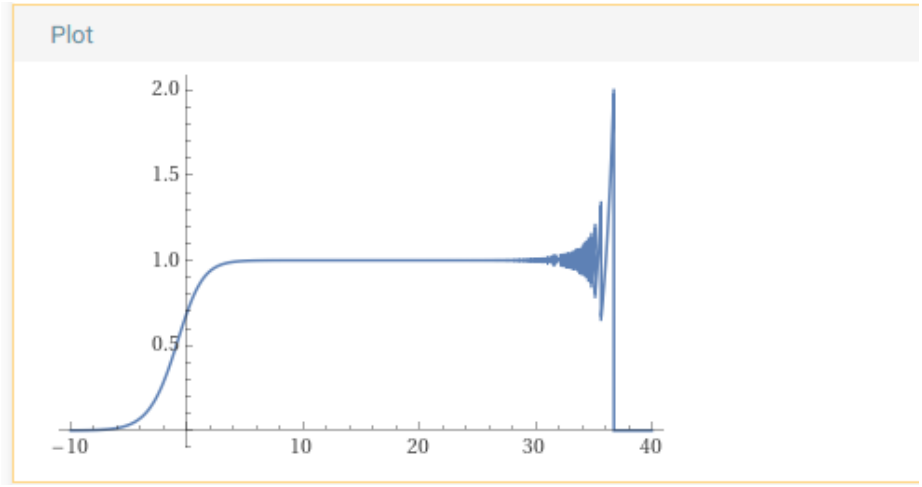
Zadanie 2. Narysować wykres funkcji

$$f(x) = e^x \ln(1 + e^{-x})$$

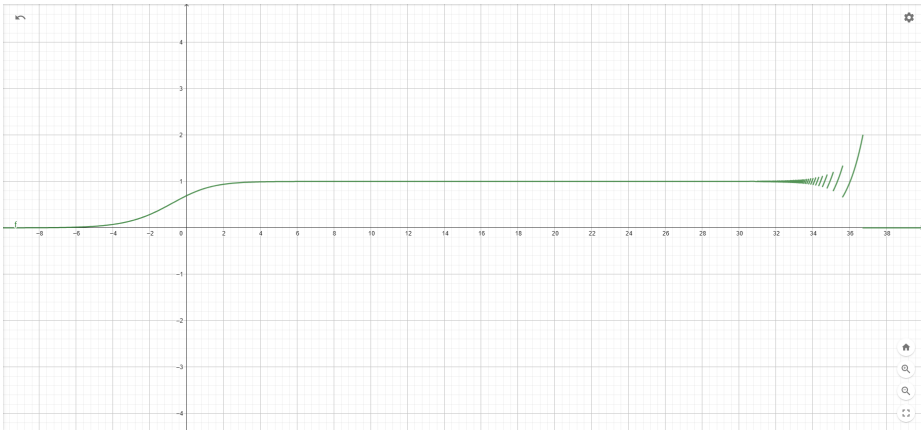
w co najmniej dwóch dowolnych programach do wizualizacji. Następnie policzyć granicę funkcji

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

Porównać wykres funkcji z policzoną granicą. Wyjaśnić zjawisko



Rysunek 1: WolframAlpha: `plot` $e^x * \ln(1+e^{-x})$



Rysunek 2: GeoGebra: f(x)=$e^x * \ln(1+e^{-x})$

Granica tej funkcji wynosi:

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

Możemy zauważyć że dla wartości powyżej x=30 wykresy zaczynają pokazywać błędne wartości. Wartości na wykresach zaczynają coraz bardziej odchodzić od 1, a następnie spadają do 0. Co ciekawe Wolfram pokazał mi troche inny wykres niż Geogebra.

Wytłumaczeniem tego zjawiska jest to że dla `x > 30` wartości e^x są już tak duże, że mnożenie jej z niewielką wartością $ln(1+e^{-x})$ skutkuje błędami przybliżenia. Wiemy, że przybliżenie $1 + e^{-x} \approx 1$, i tym samym przybliżenie całej wartości logarytmu do 0. Tak więc algorytm obliczający wartości `f(x)` nie jest stabilny numerycznie.

Zadanie 3.
Rozważmy zadanie rozwiązywania układu równań liniowych:
(a) A = Hn, gdzie Hn jest macierzą Hilberta stopnia n wygenerowaną za pomocą funkcji A=hilb(n)
(b) A = Rn, gdzie Rn jest losową macierzą stopnia n z zadanym wskaźnikiem uwarunkowania c wygenerowaną za pomocą funkcji A=matcond(n,c)

Macierze były generowane poprzez dostarczone przez prowadzącego funkcje: generującą macierz Hilberta n-tego stopnia oraz generującą losową macierz `n`-tego stopnia z zadanym wskaźnikiem uwarunkowania.
Uwarunkowanie obliczałem przez paczke LinearAlgebra z funkcji cond()
Rząd macierzy obliczałem przez paczke LinearAlgebra z funkcji rank()

n	Uwarunkowanie	Rząd macierzy	Błąd metody Gaussa	Błąd metody inwersji
1	1.0	1	0.0	0.0
2	19.28147006790397	2	5.661048867003676e-16	1.4043333874306803e-15
3	524.0567775860644	3	8.022593772267726e-15	9.6877260381103824e-15
4	15513.73873892924	4	4.137409622430382e-14	3.9612448750230811e-13
5	476607.2502422687	5	1.6828426299227195e-12	3.3544360584359632e-12
6	1.49510586424659e7	6	2.618913302311624e-10	2.0163759404347654e-10
7	4.753673565921816e8	7	1.2606867224171548e-8	4.713280397232037e-9
8	1.5257575538072489e10	8	6.124089555723088e-8	3.07748390309622e-7
9	4.931537556012197e11	9	3.8751634185032475e-6	4.541268303176643e-6
10	1.602441350036382e13	10	8.67039023709691e-5	0.0002501493411824886
11	5.222703245009594e14	10	0.00015827808158590435	0.007618304284315809
12	1.760619121841585e16	11	0.13396208372085344	0.258994120804705
13	3.1905581877988255e18	11	0.11039701117868264	5.331275639426837
14	9.27636978936766e17	11	1.4554087127659643	8.71499275104814
15	3.67568286586649e17	12	4.696668350857427	7.344641453111494
16	7.063115212292111e17	12	54.15518954564602	29.84884207073541
17	8.07124989431416e17	12	13.707236683836307	10.516942378369349
18	1.4135073701749765e18	12	10.257619124632317	24.762070989128866
19	5.190132496359103e18	13	102.15983486270827	109.94550732878284
20	1.3193976166344822e18	13	108.31777346206205	114.34403152557572

Tabela 1: Macierze Hilberta

Patrząc na tabele możemy zauważyć, że wraz ze wzrostem uwarunkowania, rośnie błąd metody Gaussa oraz błąd metody inwersji

Ponownie można zaobserwować rosnący błąd metody Gaussa oraz błąd metody

N	C	Uwarunkowanie	Rząd macierzy	Błąd metody Gaussa	Błąd metody inwersji
5	0	1.0000000000000009	5	1.7901808365247238e-16	9.930136612989092e-17
5	1	10.000000000000002	5	3.7812833677378426e-16	3.475547814546182e-16
5	3	999.9999999999902	5	4.171248415136761e-15	1.9895989601924898e-14
5	7	9.999999991507452e6	5	1.9949035820038333e-10	3.903603812005627e-10
5	12	9.999617593192695e11	5	2.9101068345710293e-6	3.4119689598080286e-6
5	16	1.18603034534176398e16	4	1.4895204919483638e-16	0.09688759998704684
10	0	1.0000000000000013	10	1.7901808365247238e-16	2.7194799110210365e-16
10	1	10.000000000000012	10	2.220446049250313e-16	7.850462293418875e-17
10	3	999.9999999999665	10	2.5974378891700425e-14	2.5468041891659954e-14
10	7	1.0000000003941769e7	10	2.348041921353544e-10	2.33204444342896e-10
10	12	9.998906510080537e11	10	8.725223875849085e-6	1.0074701420779615e-5
10	16	7.52542020410852e15	9	0.25077556682423396	0.1656130228570318
20	0	1.0000000000000016	20	5.008324230288946e-16	4.697175049207787e-16
20	1	10.000000000000005	20	4.570830125558456e-16	4.781698474367165e-16
20	3	999.9999999999193	20	1.649822793411015e-14	1.8641170262348222e-14
20	7	9.999999993462453e6	20	3.1821431180065117e-10	2.9061963586979877e-10
20	12	9.999502299696078e11	20	2.4371075346762986e-6	4.023556348748967e-6
20	16	1.2900131176993747e17	19	0.451950530517431	0.4542752104142102

Tabela 2: Macierze losowe

inwersji. Funkcja cond() w tej tabeli nie oblicza dokładnego uwarunkowania. W dodatku wartości tych przybliżeń są różne dla różnych procesorów.

Patrząc na te dwie tabele można wywnioskować, że zadanie to było źle uwarunkowane. Dla macierzy Hilberta mogliśmy zauważyć bardzo mocny wzrost wskaźnika uwarunkowania oraz błędu względnego, gdy rósł stopień macierzy. W przypadku tabeli drugiej również występowało to zjawisko, lecz w mniejszym stopniu.

Zadanie 4.
(a) Użyć funkcji roots (z pakietu Polynomials) do obliczenia 20 zer wielomianu P w postaci naturalnej
(b) Powtórzyć eksperyment Wilkinsona. Wyjaśnić zjawisko

Patrząc na tabele można zaobserwować, że obliczone miejsca zerowe są bardzo bliskie wartościom prawidłowym, lecz im nie równe. To właśnie sprawia, że wartości funkcji $P(z_k)$ oraz $p(z_k)$ są od siebie różne. .

Zmniejszenie współczynnika przy x^{19} o 2^{-23} spowodowała pojawienie się pierwiastków zespolonych Ponownie, pomimo dokonania bardzo małej zmiany, możemy zauważyć znaczną różnice w rozmieszczeniu pierwiastków. Uważam, że wyznaczanie pierwiastków wielomianu Wilkinsona jest bardzo źle uwarunkowanym zadaniem.

k	z_k	$\lvert Pz_k\rvert$	$\lvert pz_k\rvert$	$\lvert z_k - k\rvert$
1	0.9999999999998084	23323.616390897252	1307.6163909064744	1.9162449405030202e-13
2	2.0000000000114264	64613.550791712885	17818.44920890725	1.1426415369442111e-11
3	2.9999999998168487	18851.098984644806	42588.90101001011	1.8315127192636282e-10
4	3.999999983818672	2.6359390809003003e6	302428.90731865726	1.6181327833209025e-8
5	5.000000688670983	2.3709842874839526e7	3.02987280453671e6	6.88670983350903e-7
6	5.999988371602095	1.2641076289358065e8	2.625006128965147 6e7	1.162839790502801e-5
7	7.000112910766231	5.2301629899144447e8	4.5146144961963445e7	0.00011291076623098917
8	7.999279406281878	1.798432141726085e9	4.1787611441695905e8	0.0007205937181220534
9	9.003273831140774	5.1218815526720 67e9	1.7512547500017788e9	0.003273831140774064
10	9.989265687778465	1.4157542666785017e10	3.807588977407851e9	0.010734312221535092
11	11.027997558569794	3.586354765112257e10	9.907860486640614e9	0.027997558569794023
12	11.94827395840048	8.510931555828575e10	2.824260800886529e10	0.051726041599520656
13	13.082031971969954	2.2136146301419052e11	5.488190539472615e10	0.08203197196995404
14	13.906800565193148	3.812024574451268e11	1.6036435611967e11	0.09319943480685211
15	15.081439299377482	8.809029239560208e11	3.458559438392481e11	0.0814392993774824
16	15.942404318674466	1.6747434633806333e12	8.510902604527806e11	0.05759568132553383
17	17.026861831476396	3.3067827086376123e12	1.4792219527172427e12	0.026861831476395537
18	17.99048462339055	6.166202940769282e12	3.145883109080695e12	0.009515376609449788
19	19.001981084996206	1.406783619602919e13	7.707479532874811e12	0.001981084996206306
20	19.999803908064397	3.284992217648231e13	1.3655149320405107e13	0.00019609193560299332

Tabela 3: (a)

k	z_k	$\lvert Pz_k\rvert$	$\lvert pz_k\rvert$	$\lvert z_k - k\rvert$
1	0.9999999999999805 + 0.0i	2168.9361669986724	391.0638330014476	1.9539925233402755e-14
2	1.9999999999985736 + 0.0i	29948.438957395843	22787.561042399666	1.4264145420384011e-12
3	3.000000000105087 + 0.0i	239010.53520956426	177245.4648523173	1.0508705017286957e-10
4	3.9999999950066143 + 0.0i	939293.8049425513	1.530821130249216e6	4.993385704921138e-9
5	5.000000034712704 + 0.0i	7.448680396795 52e6	5.239193368182596e6	3.4712703822492585e-8
6	6.000005852511414 + 0.0i	1.4689332508961653e7	1.2659098560700657e7	5.852511414161654e-6
7	6.999704466216799 + 0.0i	5.817946400915084e7	8.540920686197355e6	0.00029553378320112955
8	8.007226654064777 + 0.0i	1.3954205929609105e8	7.22117656634685e7	0.007226654064777386
9	8.917396943382494 + 0.0i	2.459617755654851e8	6.760723571954966e7	0.082603056617506
10	10.09529034477879 - 0.6432770896263527i	2.291018560461982e9	1.878624279867249e9	0.6502965968281023
11	10.09529034477879 + 0.6432770896263527i	2.291018560461982e9	1.878624279867249e9	1.11009232692088712
12	11.793588728372308 - 1.6522535463872843i	2.077690789102519e10	9.342560958397488e9	1.6650968123818863
13	11.793588728372308 + 1.6522535463872843i	2.077690789102519e10	9.342560958397488e9	2.0458176697496047
14	13.99233053734825 - 2.5188196443048287i	9.390730597798799e10	1.0169486172930309e11	2.5188313205122075
15	13.99233053734825 + 2.5188196443048287i	9.390730597798799e10	1.0169486172930309e11	2.7129043747424584
16	16.73073008036981 - 2.81262729869721 36i	9.592356563898315e11	1.2191084776152224e12	2.906000476898456
17	16.73073008036981 + 2.81262729869721 36i	9.592356563898315e11	1.2191084776152224e12	2.8254873227453055
18	19.50243895868367 - 1.9403320231930836i	5.050467401799687e12	1.268640962637178e13	2.4540193937292005
19	19.50243895868367 + 1.9403320231930836i	5.050467401799687e12	1.268640962637178e13	2.004328632592893
20	20.84690887410499 + 0.0i	4.858653129933677e12	4.083550822479832e13	0.8469088741049902

Tabela 4: (b)

Zadanie 5. Rozważmy równanie rekurencyjne (model logistyczny, model wzrostu populacji):

$$p_{n+1} := p_n + rp_n(1 - p_n), \quad \text{dla} \quad n = 0, 1, \ldots$$

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.

Wykonałem trzy eksperymenty:

1. 40 iteracji modelu logistycznego dla danych $p_0 = 0.01$ i $r = 3$ w `Float32`
2. 10 iteracji modelu logistycznego dla tych samych danych, obcięcie wyniku do trzech miejsc po przecinku, i kontynuowanie iteracji aż do 40-tej, w `Float32`
3. powtórzenie 1-go eksperymentu dla `Float64`

W tabeli poniżej zaprezentowane są wyniki dla powyższych eksperymentów.

n	1. eksperyment	2. eksperyment (z obcięciem)	3. eksperyment (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	0.7229306	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
40	0.25860548	1.093568	0.011611238029748606

Tabela 5: Eksperymenty

Gdy popatrzymy na tabele możemy zauważyć, że dla ostatniej iteracji `Float32` wynik po obcięciu różni się od tego bez ponad czterokrotnie, w porównaniu do `Float64` otrzymujemy stukrotnie większy wynik.

Uważam, że powyższe zadanie zostało źle uwarunkowane. Pomimo obcięcia jedynie niewiele znaczących cyfr nastąpiła ogromna zmiana ostatecznego wyniku.

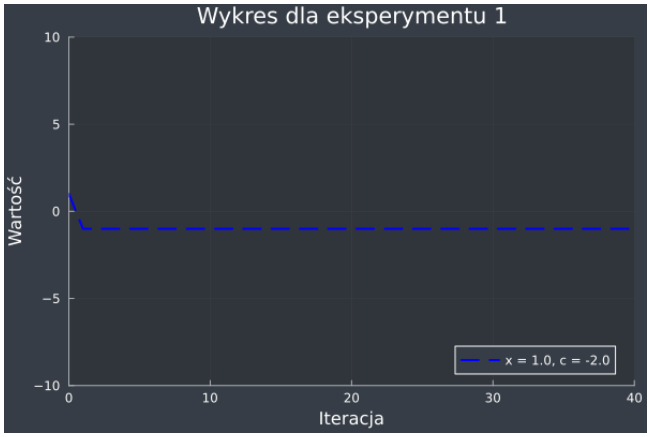
Zadanie 6.
Rozważmy równanie rekurencyjne

$$x_{n+1} := x_n^2 + c \quad \text{dla} \quad n = 0, 1, \ldots$$

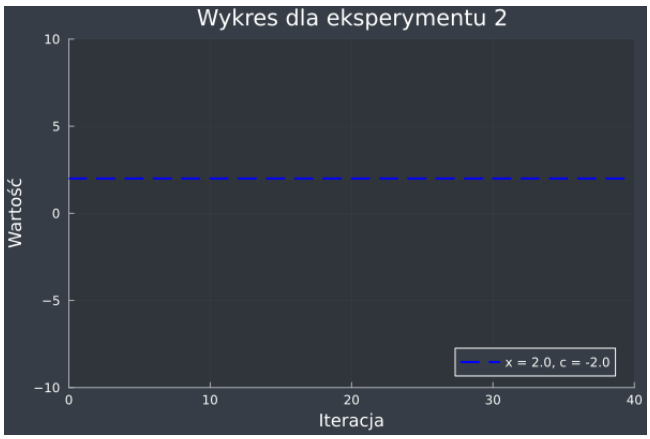
wykonać, w języku Julia w arytmetyce Float64, 40 iteracji wyrażenia. Zaobserwować zachowanie generowanych ciągów.

Wnioski z poniższych wykresów:

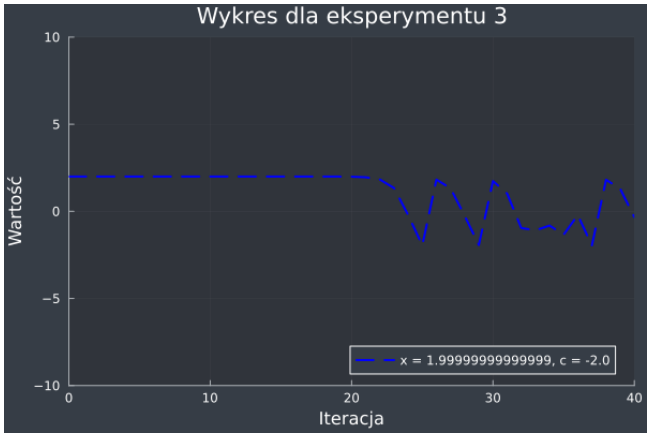
Na wykresach, na których `x` był wartością całkowitą, wyniki były poprawne. Tam gdzie mieliśmy doczynienia z liczbami zmiennoprzeciwnkowymi pojawiają się błędy. Niewielkie zmiany doprowadzają do różnych wyników, czyli zadanie jest źle uwarunkowane



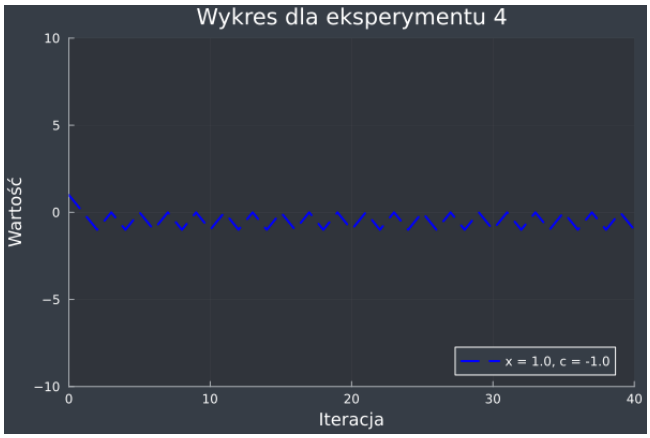
Rysunek 3: 1 eksperyment - wykres jest stały



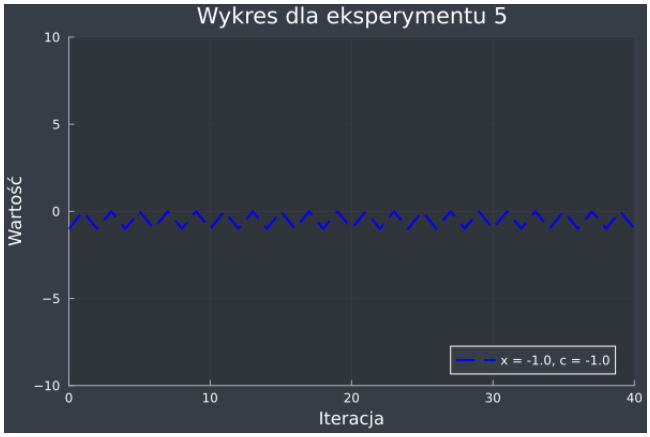
Rysunek 4: 2 eksperyment - wykres jest stały



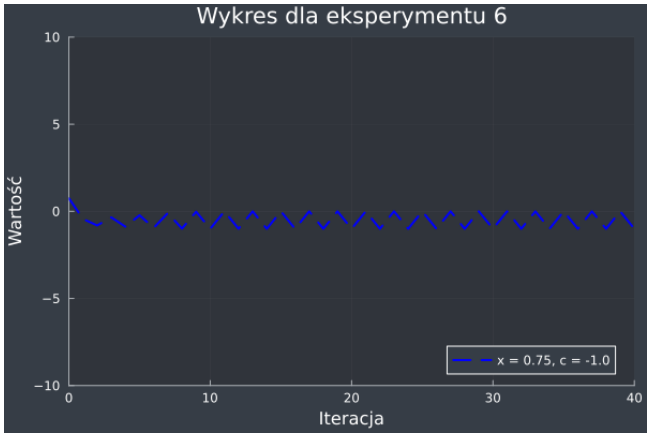
Rysunek 5: 3 eksperyment - po pewnym czasie otrzymujemy wyniki które nie są poprawne, powinny być dalej bliskie 2



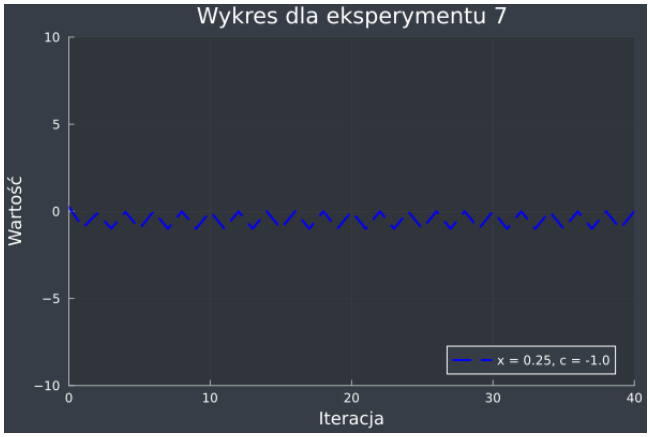
Rysunek 6: 4 eksperyment - wynik oscyluje wokół poprawnej wartości



Rysunek 7: 5 eksperyment - wynik oscyluje wokół poprawnej wartości



Rysunek 8: 6 eksperyment - wynik oscyluje wokół poprawnej wartości



Rysunek 9: 7 eksperyment - wynik oscyluje wokół poprawnej wartości