Obliczenia Naukowe 2

Adrian Herda 268449

5 November 2023

1 Zadanie 1

Zadanie polegało na powtórzeniu eksperymetnu z poprzedniej listy zadań (Zadanie 5.) z minimalnie zmienionymi danymi wejściowymi i porównanie wyników z wynikami otrzymanymi w normalnej wersji zadania. Dane przed zmianami wyglądały następsująco:

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

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

Zmiany dotyczyły jedynie wektora x który miał mieć postać:

$$x = [2.718281828, 3.141592654, 1.414213562, 0.577215664, 0.301029995]$$

1.1 Rozwiązanie

Rozwiązanie zadania znajduje się w pliku zad1.jl i polega na dosłownej implementacji algorytmów podanych w zadaniu 5. z listy 1. Następnie należało przetestować je dla danych z listy 1 i dla danych z listy 2.

1.2 Wyniki oraz ich interpretacja

1.2.1 Rożnica dla Float32

Tabela 1: Float32

	Lista 1	Lista 2	Prawdziwy wynik
Funkcja a:	-0.4999443	-0.4999443	-1.00657107000000e-11
Funkcja b:	-0.4543457	-0.4543457	-1.00657107000000e-11
Funkcja c:	-0.39291382	-0.39291382	-1.00657107000000e-11
Funkcja d:	-0.5	-0.5	-1.00657107000000e-11

1.2.2 Różnica dla Float64

Tabela 2: Float64

	Lista 1	Lista 2	Prawdziwy wynik
Funkcja a:	1.0251881368296672e-10	-0.004296342739891585	-1.00657107000000e-11
Funkcja b:	-1.5643308870494366e-10	-0.004296342998713953	-1.00657107000000e-11
Funkcja c:	1.4068746168049984e-12	-0.00429634284087399	-1.00657107000000e-11
Funkcja d:	0.0	-0.004296342842280865	-1.00657107000000e-11

W arytmetyce Float32 uzyskaliśmy wyniki takie same jak dla danych z obu list. W aytmetyce Float64 widizimy natomiast dużą różnicę w wynikach, rzędu nawet 10^{-3}.

1.3 Wnioski

W arytmetyce Float32 wyniki pozostają takie same, ze względu na to że precyzja tej arytmetyki jest mniejsza i niż precyzja podawanych liczb, przez co te są zaokrąglane i liczby najmniej znaczące, czyli te w których dokonaliśmy zmian, nie mają znaczenia.
Wyniki dla arytmetyki Float64 pokazują nam natomiast, że małe zmiany w danych wpływają na błędy względne w znacznym stopniu jednocześnie nie wpływając na błąd bezwzględny. Wnioskować z teog możemy, że to zadanie jest źle uwarunkowane.

2 Zadanie 2

Zadanie to polegało na narysowaniu wykresu w funkcji

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

w wybranych programch do wizualizacji. Następnie należało obliczyć granicę $lim_{x->\infty} f(x)$ oraz porównać ja z wykresem i wyjasnić co się dzieje.

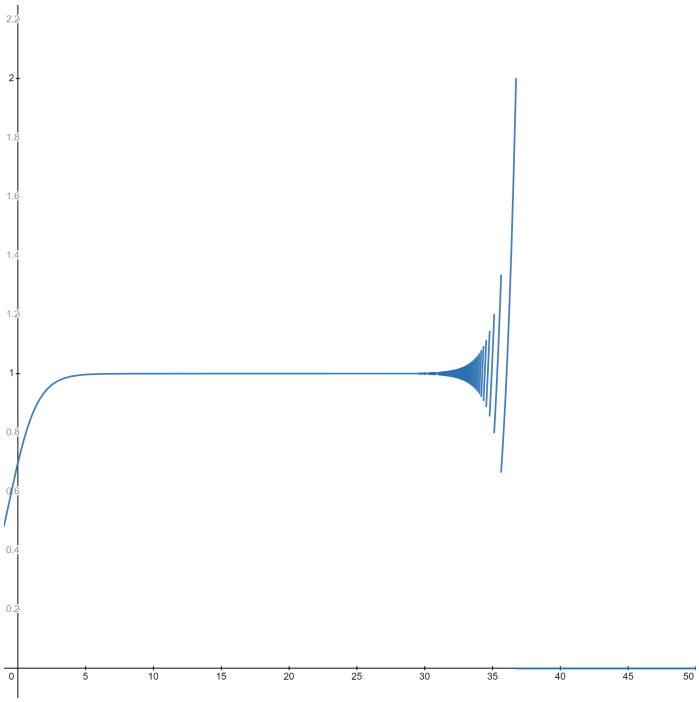
2.1 Rozwiązanie

Podaną funkcję narysowałem za dwóch pakietów do wizulaiazcji. Pierwszym był kalkulator graficzny Desmos a drugim był biblioteka MatPlotLib.Pyplot języka Python.

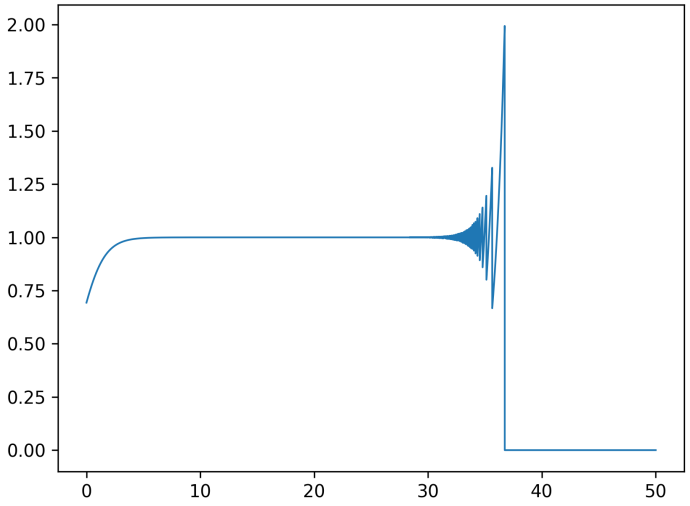
2.2 Wyniki oraz ich interpretacja

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

Jak widać wykresy nie odzwierciedlają tego co wynik obliczonego przez mnie limitu. Na wykresach wygląda jakby, podana w zadaniu funckja, dla wartości $x >= 30$ zaczynała odbiegać od oczekiwanej wartości a następnie zbiegała do zera i więcej już nie zmieniała wartości. Według obliczonej granicy, jednak, funkcja ta przy x dążących do nieskończoności powinna dążyć do 1.



Rysunek 1: Graf kalkulatora Desmos



Rysunek 2: Graf biblioteki MatPlotLib.Pyplot

2.3 Wnioski

Obliczając wyrażenie e^{-x} dla danych $30 <= x < 40$ zauważamy że wyniki zbliżają się do epsilona maszynowego a jak wiemy z poprzedniej listy przy ograniczonej precyzji bardzo małe liczby potrafią powodować błędy, stąd takie nieoczekiwane wachania wyników. Dla $x > 36$ wcześniej wymienione wyrażenie jest tak małe że przy takiej precyzji zaokrągla się ono do 0 powodując że wartość logarytmu naturalnego również spada do zera zmieniając wynik całej funkcji w 0. Możemy z tego wnioskować że zadanie jest źle uwarunkowane.

3 Zad 3

Zadanie to polegało na porównaniu metod rozwiązywania układu równań liniowych postaci

$$Ax = b$$

Porównywane przez nas metody to

- Metoda eliminacji Gaussa $x = A \backslash b$
- Metoda inwersji $x = A^{-1}b$

Porównanie miało się dzielić również na dwa rodzaje macierzy

- Macierze Hilberta stopnia $n \in \{1, 4, 7, 10, ..., 52\}$
- Macierze losowe stopnia $n \in \{5, 10, 15\}$ o współczynniku uwarunkowania $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$

Wyniki mieliśmy porównywać pod względem błędów w metodach obliczania wyników.

3.1 Rozwiązanie

Rozwiązanie znjaduje się w pliku zad3.jl, a funkcje tworzące macierze są zaawarte w plikach danych razem z listą zadań odpowiednio hilb.jl oraz matcond.jl. Algorytm najpierw macierz A wybranym sposobem, potem tworzy wektor $x = \{1, 1, ..., 1\}$ następnie oblicza wektor $b = Ax$ a na koniec oblicza z powrotem x i porównuje z prawidłowym wynikiem i oblicza błędy.

3.2 Wyniki oraz ich interpretacja

3.2.1 Macierze Hilberta

Tabela 3: Macierze Hilberta

n	$cond(A)$	$rank(A)$	Błąd eliminacji Gaussa	Błąd metody inwersji
1	1.0	1	0.0	0.0
4	15513.73873892924	4	4.137409622430382e-14	0.0
7	4.753673567446793e8	7	1.2606867224171548e-8	4.713280397232037e-9
10	1.602441698742836e13	10	8.67039023709691e-5	0.0002501493411824886
13	3.1883950689209334e18	11	0.11039701117868264	5.331275639426837
16	7.046389953630175e17	12	54.15518954564602	29.84884207073541
19	6.472700911391398e18	13	102.15983486270827	109.94550732878284

22	1.093638219471544e19	13	17.003425713362045	102.60161611995359
25	1.3309197502927598e18	13	7.095757204652332	21.04404299195525
28	5.889050001520599e18	14	276.91498822022265	290.1167291705239
31	2.4040817319121502e19	14	21.45662601984968	23.74575780277118
34	4.56520144655429e18	14	88.87380459381126	95.99116506490786
37	5.871718859396612e18	15	13.974714130452178	16.39248770656996
40	6.584302662074187e18	15	23.926484807638683	140.97274594056717
43	3.891698791143783e19	15	53.18930954995268	54.52704305417691
46	1.5700045048756816e19	15	69.14584939886464	109.17112219679052
49	6.145459250718421e18	16	24.15062009750964	35.92139018094681
52	1.1181915944118233e19	16	204.2938614173205	106.1172474183331

3.2.2 Macierze Losowe

Tabela 4: Macierze Losowe

n	$cond(A)$	$rank(A)$	Błąd eliminacji Gaussa	Błąd metody inwersji
5	1.0000000000000004	5	3.8459253727671276e-16	3.1401849173675503e-16
5	10.000000000000004	5	3.813741331030777e-16	3.3674731643402955e-16
5	999.9999999999794	5	2.115864892837535e-14	2.2250597837287583e-14
5	9.999999999595143e6	5	5.0011211277438676e-11	4.692851652099372e-11
5	1.0000535547718358e12	5	1.1645574434398402e-5	1.0092740291841853e-5
5	1.174661470095863e18	4	0.3525642720764605	0.31156108951536293
10	1.000000000000002	10	3.5631068175681536e-16	2.482534153247273e-16
10	10.000000000000007	10	7.97508059416606e-16	4.762326219937489e-16
10	1000.000000000013	10	3.423399295234108e-14	3.656610166950196e-14
10	9.999999997550212e6	10	8.6851707752283e-11	9.260782143506989e-11
10	9.99981878858708e11	10	2.9067646430993557e-5	3.4010558641139e-5
10	1.640731211680382e16	9	0.09701938822410716	0.23876104359275194
20	1.0000000000000016	20	5.289603976864219e-16	5.444622457435256e-16
20	9.99999999999999	20	3.5803616730494477e-16	4.927689594407735e-16
20	1000.000000000024	20	7.253674226423259e-15	6.3085681947966634e-15
20	1.000000000132515e7	20	1.611332251999396e-10	1.6907626369009887e-10
20	9.999648986333438e11	20	5.60436198343498e-6	5.670153234937571e-6
20	3.6387718615923e16	19	0.3044614942558878	0.31806567165347316

Macierze Hilberta wykazują duże wskaźniki uwarunkowania nawet dla stosunkowo małych wielkości. Obie metody obliczania wyników skutkują dużymi błędami, jednakże metoda Gaussa zazwyczaj ma mniejsze błedy wyników niż metoda inwersji. Obliczanie wyników dla macierzy losowych o ustalonym współczynniku uwarunkowania, daje małe błedy i nie wykazuje widocznych różnic w wymienionych metodach. Można jednak zauważyć pewną regularność w tych błędach, mianowicie błędy wyników dla macierzy o podobnych wartościach współczynnika uwarunkowania są podobne.

3.3 Wnioski

Na podstawie wyników przedstawionych powyżej, można stwierdzić, że algorytmy są efektywne, co implikuje, że rozwiązanie układu równań z macierzą Hilberta jako współczynnikami jest problemem o złym uwarunkowaniu. Co więcej znając współczynnik uwarunkowania jesteśmy w stanie oszacować błędy względne rozwiązań.

4 Zad 4

Zadanie 4 dotyczyło złośliwego wielomianu Wilinsona podanego wzorem:

$$\prod_{i=1}^{20}(x-i)$$

W podpunkcie a) mieliśmy, używając pakietu *Polynomials*, mieliśmy sprawdzić jak pakiet ten radzi sobie z obliczaniem pierwiastkó na dwa różne sposoby:

- Przy podanej formie naturalnej
- Przy podanej formie iloczynowej

W podpunkcie b) tego zadania, mieliśy sprawdzić jak na te wyniki wpływają minimalne zmiany przy współczynnikach formy naturalnej tego wielomianu. Zmiana polegałą na odjęciu wartości 2^{-23} w drugim współczynniku.

4.1 Rozwiązanie

Rozwiązanie zawarte jest w pliku zad4.jl i polega na dosłownej implemntacji algorytmu i obliczeń podanych w zadaniu.

4.2 Wyniki oraz ich interpretacja

4.2.1 Podpunkt a)

Tabela 5: Podpunkt a)

k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	23323.616390897252	24347.616390897056	1.9162449405030202e-13
2	64613.550791712885	80997.5507918065	1.1426415369442111e-11
3	18851.098984644806	101795.09897958105	1.8315127192636282e-10
4	2.6359390809003003e6	2.37379508196076e6	1.6181327833209025e-8
5	2.3709842874839526e7	2.306984278668964e7	6.88670983350903e-7
6	1.2641076289358065e8	1.2508366146559621e8	1.162839790502801e-5
7	5.2301629899144447e8	5.2055763533357024e8	0.00011291076623098917
8	1.798432141726085e9	1.7942387274786062e9	0.0007205937181220534
9	5.121881552672067e9	5.115158339932115e9	0.003273831140774064
10	1.4157542666785017e10	1.4147337313301882e10	0.010734312221535092
11	3.586354765112257e10	3.584844758752149e10	0.027997558569794023

12	8.510931555828575e10	8.508835580052173e10	0.051726041599520656
13	2.2136146301419052e11	2.2133136429365445e11	0.08203197196995404
14	3.812024574451268e11	3.811638891032201e11	0.09319943480685211
15	8.809029239560208e11	8.808498064028993e11	0.0814392993774824
16	1.6747434633806333e12	1.6746775852847332e12	0.05759568132553383
17	3.3067827086376123e12	3.3066967643946914e12	0.026861831476395537
18	6.166202940769282e12	6.16609562105193e12	0.009515376609449788
19	1.406783619602919e13	1.4067702719729383e13	0.001981084996206306
20	3.284992217648231e13	3.2849758339688676e13	0.00019609193560299332

4.2.2 Podpunkt b)

Tabela 6: Podpunkt b)

k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	2168.9361669986724	2168.9361669986324	1.9539925233402755e-14
2	29948.438957395843	13564.438957442584	1.4264145420384011e-12
3	239010.53520956426	48546.53517157824	1.0508705017286957e-10
4	939293.8049425513	1.201437802324579e6	4.993385704921138e-9
5	7.44868039679552e6	4.333159313828003e6	3.4712703822492585e-8
6	1.4689332508961653e7	5.5871717686950475e7	5.852511414161654e-6
7	5.817946400915084e7	1.2508396452132258e9	0.00029553378320112955
8	1.3954205929609105e8	1.742241327643653e10	0.0072266540647767386
9	2.459617755654851e8	1.3471875977601944e11	0.082603056617506
10	2.291018560461982e9	1.4798961582209915e12	0.6502965968281023
11	2.291018560461982e9	1.4798961582209915e12	1.110092326920887
12	2.077690789102519e10	3.2933037979480902e13	1.6650968123818863
13	2.077690789102519e10	3.2933037979480902e13	2.0458176697496047
14	9.390730597798799e10	9.54583157311972e14	2.5188313205122075
15	9.390730597798799e10	9.54583157311972e14	2.7129043747424584
16	9.592356563898315e11	2.7421143234744428e16	2.906000476898456
17	9.592356563898315e11	2.7421143234744428e16	2.8254873227453055
18	5.050467401799687e12	4.2524516249528954e17	2.4540193937292005
19	5.050467401799687e12	4.2524516249528954e17	2.004328632592893
20	4.858653129933677e12	1.3743527553999101e18	0.8469088741049902

Po wynikach łatwo zauważyć, że żadne z policzonych zer i policzonych dla nich wartości wileomianu Wilkinsona nie daje prawidłowego wyniku jakim byłoby zero. Na dodatek wyniki te są bardzo daleko od zera

4.3 Wnioski

Zaburzenie wyników w minimalnym stopniu sprawiło że otrzymane wyniki znacznie się zmieniły. Wynika to z faktu że przy tak dużych współczynnikach wielomianu brakowało liczb znaczących do precyzyjjnych obliczeń. Zwiększyły się też błędy bezwzględne w obliczanych pierwiastkach. W takiej sytacji możemy jasno stwierdzić że zadanie jest źle uwarunkowane.

5 Zadanie 5

Zadanie 5 polegało na przeanalizowaniu funkcji rekurencyjnnej

$$p_{n+1} = p_n + rp_n(1 - p_n)$$

będącej modelem wzrostu populacji.
Podpunkt pierwszy zadania polegała na wykonaniu 40 iteracji podanej funkcji dla danych $p_0 = 0.01$ oraz $r = 3$ a nastęnie powtórzeniu eksperymentu i zaokrągleniu w dół wyniku przy 10 iteracji do rzędu 10^{-3}. Wszystkie obliczenia w tym podpunkcie miały być wykonywane w arytmetyce Float32. Podpunkt drugi tego zadania wymagał zrobienia 40 iteracji podanej funkcji, dla danych podanych w podpunkcie pierwszym, i porówania wyników dla arytmetyk Float32 oraz Float64.

5.1 Rozwiązanie

Rozwiązanie tego zadaina znajduje się w pliku zad5.jl i polega na dosłownej implementacji treści zadania. Zaokrąglenie w dół do rzędu 10^{-3} osiągnąłem przez wykonanie działania

$$floor(x * 1000)/1000$$

wykorzystyjąc funkcjie floor języka Julia.

5.2 Wyniki oraz ich interpretacja

5.2.1 Porównanie odcięcia i jego braku

Tabela 7: Podpunkt 1.)

n	Bez odcięcia	Z odcięciem
0	0.01	0.01
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	0.25860548	1.093568

5.2.2 Porównanie Float32 z Float64

Tabela 8: Podpunkt 2.)

n	Float32	Float64
0	0.01	0.01
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

Porównując pary wyników widać że para wyników Float32 z Float64 są zbliżone podobne do siebie dużo dłużej niż para Float32 z Float32 z odcięciem. Para Float32 z Float64 są bardzo podobne do siebie aż do 22 iteracji podczas gdy para Float32 z Flat32 z odcięciem zaczynają być już widocznie różne od 16 iteracji. Mimo wszystko oba prównania po tych punktach rozbiegają się bardzo szybko. Wyniki końcowe eksperymentów różnią się od siebie już znacząco:

Tabela 9: Wyniki końcowe

Float32	Float32 z odcięciem	Float64
0.25860548	1.093568	0.011611238029748606

5.3 Wnioski

Funkcje rekurencyjne/iteracyjne z racji polegania na wynikach z poprzednich wywołań/iteracji są bardzo podatne na duże błędy gdyż najmniejszy błąd w początkowej fazie, kumuluje się wraz kolejnymi obliczeniami i bardzo szybko powoduje oddalenie się od prawidłowego wyniku.

6 Zadanie 6

Zadanie to polegało na przeanalizowaniu wyników rekurencyjnej funkcji

$$x_{n+1} = x_n^2 + c$$

Gdzie c jest pewną wybraną stałą. Mieliśmy wykonać 40 iteracji wyrażnia a następnie zaobserwoać jakzachowują się generwoane ciągi 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 $x_0 = 0.25$

6.1 Rozwiązanie

Pełne rozwiązanie znajduje się w pliku zad6.jl. Liczę w nim iteracyjnie kolejne wartości na tablicy która służyła do trzymania początkowych wartości x_0

6.2 Wyniki oraz ich interpretacja

Tabela 10: Wyniki dla $c = -2$ i ustalonego x_0

n	$x_0 = 1.0$	$x_0 = 2.0$	$x_0 = 1.99999999999999$
0	1.0	2.0	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

Tabela 11: Wyniki dla $c = -1$ i ustalonego x_0

n	$x_0 = 1.0$	$x_0 = -1.0$	$x_0 = 0.75$	$x_0 = 0.25$
0	1.0	-1.0	0.75	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

Na wynikach widać że niektóre dane początkowe, jak np. $(c = -2, x_0 = 1)$ oraz $(c = -2, x_0 = 2)$, prowadzą do punktów stałych, niektóre dane nie prowadzą do żadnego punktu stałego, np. $(c = -2, x_0 = 1.99999999999999)$, a jeszcze inne mogą oscylować wokół kilku wartości. do takich punktów stałych lub oscylacji mogą dojść nie tylko od razu ale i po kilku iteracjach jak np. $(c = -1, x_0 = 0.75)$

6.3 Wnioski

Z tego zadania możemy wynieść ponowne przypomnienie że funkcje rekurencyjne potrafią kumulować błędy, które ostatecznie mocno wpłyną na nasze wyniki, przykładem są tego $x_0 = 0.75$ oraz $x_0 = 0.35$, które nie powinny wpaść w oscylacje ale z racji skończonej precyzjii chciały dodać liczby małe do dużej w wyniku czego zabrakło liczb znaczących i doprowadziło do oscylacji.