Obliczenia Naukowe
lista 2

Mateusz Tofil

15 listopada 2021

1 Zadanie 1

1.1 Opis zadania

Jest to powtórka zadania z listy poprzedniej - listy 1 zadania 5. Jedyna różnica jaka występuje, to usunięcie z x_4 ostatniej cyfry tj. 9. i z x_5 usunąc ostatnią 7.

1.2 Metoda rozwiązania

Metoda niczym się nie różni od wcześniejszego zdaniaz wcześniejszej liczby. Dane wejśćiowe są tylko inne - a dokładniej tylko x_4 i x_5. Rozwiązanie znajduję się w pliku `zad1.jl` i jest dokłądną duplikacją kodu z zadania 5 z poprzedniej listy ze zmienionymi danymi wejściowymi.

1.3 Otrzymane wyniki

Tabele poniżej przedstawiając sumy obliczone tak jak w zadaniu 5 z listy 1, natomiast lekko zmienionymi wartościami na wejśćiu. Wyniki przedstawione dla arytmetyki `Float64` i `Float32`.

podpunkt	sumy z listy 2	sumy z listy 1
a	-0.004296342739891585	1.0251881368296672e-10
b	-0.004296342998713953	-1.5643308870494366e-10
c	-0.004296342842280865	0.0
d	-0.004296342842280865	0.0

Tablica 1: Porównanie sum z poprzedniej listy i obecnej dla `Float64`

podpunkt	suma z listy 2	suma z listy 1
a	-0.3472038161889941	-0.3472038161853561
b	-0.3472038162872195	-0.3472038162872195
c	-0.5	-0.5
d	-0.5	-0.5

Tablica 2: Wyniki dla `Float32`

1.4 Wnioski

Zmiana wartości liczby, rzędu nawet 2^{-10} wpływa znacząco na wyniki ostateczny. Niewielkie zmiany spowodowały duże względne odkształcenia wyników, zatem możemy stwierdzić, że zadanie jest źle uwarunkowane.

2 Zadanie 2

2.1 Opis zadania

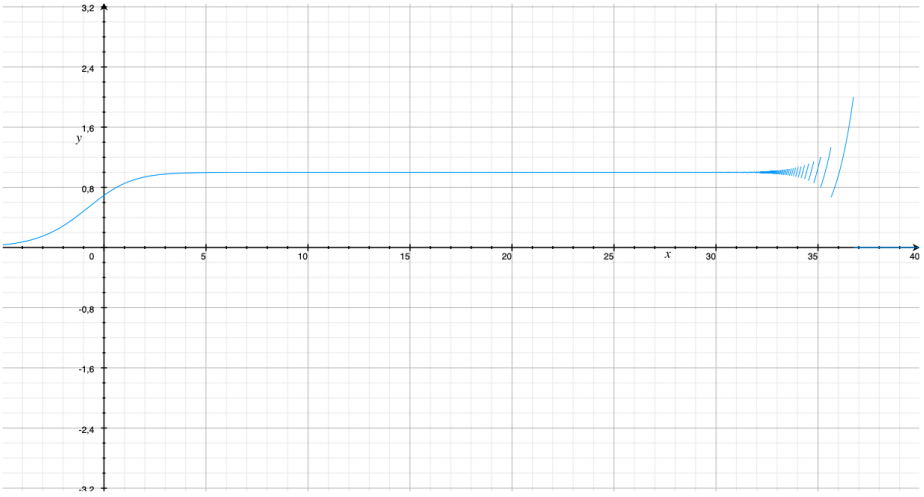
Należało narysować funckję $f(x) = e^x ln(1 + e^{-x})$ w conajmniej dwóch rożnych program do rysowania wykresu, zbadać faktyczną granicę funkcji i porównać z otrzymanymi wykresami.

2.2 Metoda rozwiązania

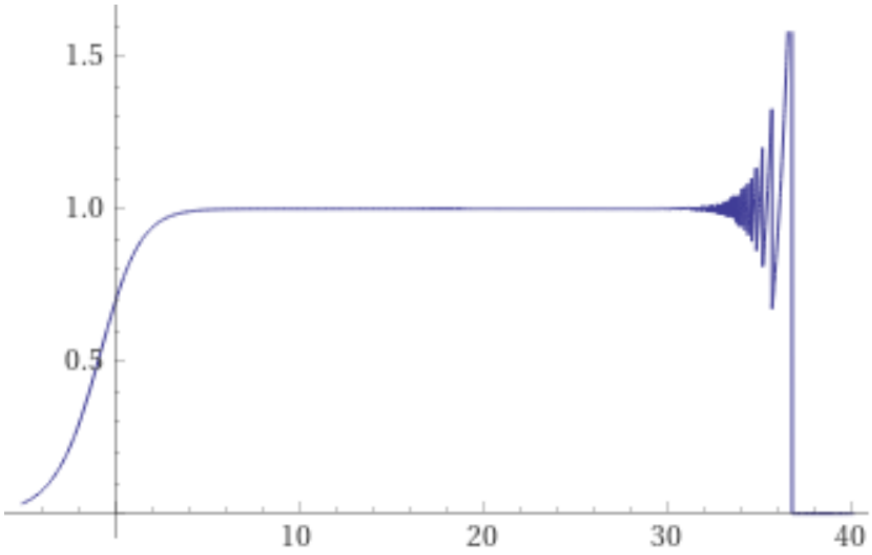
Funkcje $f(x)$ wpisać do programu umożliwiającego rysowanie wykresów. Programy, które wybrałem do narysowania funkcji $f(x)$ to: WolframAplha, Grapher oraz postanowiłem napisać programy przedstawiający wykresy w języku programowania Python z wykorzystaniem biblioteki matplotlib in numpy

2.3 Otrzymane wyniki

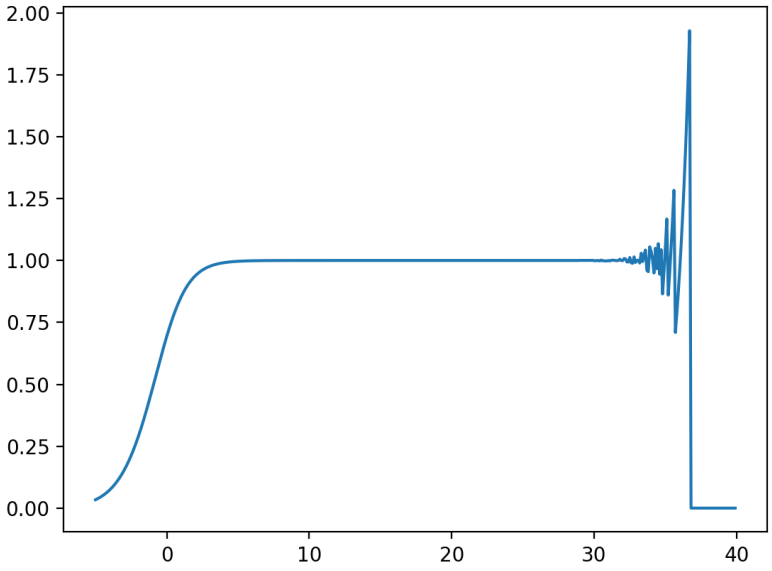
W programie Grapher należało wpisać funkcję w wyznaczone miejsce. Następnie przeskalować oś x, wykonując następujące czynności: Viev > Frame Limit ... i zmienić skalowanie.Na stronie WolframAplha należało wpisać `plot <funkcja> from -5 to 40`. Program napisany w pythonie znajduję się w pliku o ścieżce `./zad2/plotpython.py`



Rysunek 1: Wykres funkcji $f(x)$ w programie Grapher



Rysunek 2: Wykres funkcji $f(x)$ w programie WolframAplha



Rysunek 3: Wykres funkcji $f(x)$ w języku Python z wykorzystamiem biblioteki matplotlib

Obliczmy teraz granicę funkcji:

$$\lim_{x\to\infty} e^x \ln(e^{-x}+1) = \lim_{x\to\infty} \frac{e^x \ln(e^{-x}+1)e^{-x}}{e^{-x}} = \lim_{x\to\infty} \frac{\ln(e^{-x}+1)}{e^{-x}} \overset{H}{=}$$
$$\overset{H}{=} \lim_{x\to\infty} \frac{\frac{d}{dx}\ln(e^{-x}+1)}{\frac{d}{dx}e^{-x}} = \lim_{x\to\infty} \frac{-\frac{e^{-x}}{e^{-x}+1}}{-e^{-x}} = \lim_{x\to\infty} \frac{1}{e^{-x}+1} = 1 \quad (1)$$

Jak łatwo zauważyć, granica funkcji, którą obliczyliśmy powyżej nie pokrywa się z granicą funkcji odczytując ją z wykresu. Na wykresie granica funkcji dąży do 0 natomiast z matematycznego punktu widzenia dąży do 1. Powyżej argumnetów powyżej 30 obserwujemy, że wykresy zaczynają pokazywać nieprawidłowe wyniki.

2.4 Wnioski

Wartości e^{-x} dla każdego następnego argumentu zbliżając się do epsilona maszynowego, po przekroczeniu epsilona, funkcja spada do zera. Przed przekroczeniem epsiolona maszynowego, funkcja dąży do jedynki, tak jak powinna.

wykładnik	$exp(wykladnik)$	$exp(wykladnik) - epsilon$
30	9.357622968840175e-14	9.335418508347672e-14
31	3.442477108469977e-14	3.4202726479774736e-14
32	1.2664165549094176e-14	1.2442120944169144e-14
33	4.658886145103398e-15	4.4368415401783664e-15
34	1.713908431542013e-15	1.4918638266169817e-15
35	6.305116760146989e-16	4.084670710896676e-16
36	2.319522830243569e-16	9.907678099325606e-18
37	8.533047625744066e-17	-1.3671412866759066e-16
38	3.1391327920480296e-17	-1.90653277004551e-16
39	1.1548224173015786e-17	-2.104963807520155e-16
40	4.248354255291589e-18	-2.1779625066973972e-16

Tablica 3: Wartości exp() dla kolejnych argumentów porównanie z epsilonem maszynowym

3 Zadanie 3

3.1 Opis zadania

Porównanie rozwiązań równań macierzowych dwoma metodami: macierzy odwrotnej i eliminacja Gausa dla dwóch rodzajów maccierzy: Macierzy Hilberta oraz macierzy losowej.

3.2 Metoda rozwiązania

Rozwiązanie znajduję się w pliku `zad3.jl`, zawiera one funkcję napisane przez prowadzącego do generowania odpowiednich macierzy. Przez wbudowane funkcje w język Julia, bez trudności jest zaimplementowanie powyższego problemu.

3.3 Otrzymane wyniki

Rozwiązania przedstawie w dwóch tabelakch z podziałem na macierze, gdzie w pierwszej tabeli rozpatrujemy macierz Hilberta, natomiast w kolejnej macierz losową stopnia n ze wskaźnikiem uwarunkowania.

size	rank()	cond()	met. el. Gaussa	met. m. odwrotną
1	1	1.0	0.0	0.0
2	2	19.3	5.66e-16	1.4e-15
3	3	524.0	8.02e-15	0.0
4	4	15500.0	4.14e-14	0.0
5	5	477000.0	1.68e-12	3.35e-12
6	6	1.5e7	2.62e-10	2.02e-10
7	7	4.75e8	1.26e-8	4.71e-9
8	8	1.53e10	6.12e-8	3.08e-7
9	9	4.93e11	3.88e-6	4.54e-6
10	10	1.6e13	8.67e-5	0.00025
11	10	5.22e14	0.000158	0.00762
12	11	1.75e16	0.134	0.259
13	11	3.34e18	0.11	5.33
14	11	6.2e17	1.46	8.71
15	12	3.67e17	4.7	7.34
16	12	7.87e17	54.2	29.8
17	12	1.26e18	13.7	10.5
18	12	2.24e18	9.13	7.58
19	13	6.47e18	9.72	12.2
20	13	1.36e18	7.55	22.1

Tablica 4: Wyniki dla macierzy Hilberta używając metody macierzy odwrotenj i eliminacji Gaussa

Macierz Hilberta, jest bardzo dobrym przykładem maceirzy źle uwarunkonej, wraz ze wzrostem rozmiaru macierzy, wzrasta cond(x) i wyniki są coraz bardziej rozstrzelane.

size	rank()	cond()	met. el. Gaussa	met. m. odwrotną
5	5	1.0	9.93e-17	9.93e-17
5	5	10.0	1.11e-16	2.81e-16
5	5	1000.0	2.31e-14	2.6e-14
5	5	1.0e7	1.18e-10	9.39e-11
5	5	1.0e12	5.46e-6	3.81e-6
5	4	1.46e16	0.33	0.161
10	10	1.0	2.72e-16	1.79e-16
10	10	10.0	2.19e-16	3.39e-16
10	10	1000.0	3.22e-14	2.69e-14
10	10	1.0e7	1.18e-10	1.38e-10
10	10	1.0e12	1.53e-5	1.26e-5
10	9	9.52e15	0.121	0.148
20	20	1.0	3.84e-16	2.66e-16
20	20	10.0	5.17e-16	7.69e-16
20	20	1000.0	9.71e-15	8.34e-15
20	20	1.0e7	2.04e-10	1.84e-10
20	20	1.0e12	3.54e-7	4.35e-6
20	19	1.2e16	0.156	0.125

Tablica 5: Wyniki dla macierzy generowanej losowo z użwyając metody macierzy odwrotenj i eliminacji Gaussa

Warto zauważyć, że dla rozmiaru macierzy 20x20 i wskaźniku uwarunkowania błedy są mniejsze niż dla macierzy rozmiaru 5x5 i wskaźniku uwarunkowania $1.46e16$

3.4 Wnioski

Uwarunkowanie macierzy ma wpływ na otrzymane wyniki. Im większy cond(x) otrzymujemy coraz większe błedy przy dowolnej metodzie. Dla macierzy Hilberta, wraz ze wzrostem rozmiaru macierzy, zwiększa się cond(x) i z każdą iteracją otrrzymujemy to coraz mniej dokładne wyniki. Dla macierzy losowej wyniki dla dużych rozmiarów ale z małym wskaźnikiem uwarunkowania dają lepsze wyniki niż dla małych macierzy z dużym wskaźnikiem.

4 Zadanie 4

4.1 Opis zadania

Problem w zadaniu to znalezienie zer wielomianu Wilkinsona. Do tego celu posługujemy się pakietem Polynomials. Obliczami pierwiastki wielomianu z postaci ogólnej i ilorazowej. Porównujemy otrzymane wyniki wraz z wynikami nam dobrze znanymi. Cały eksperyment powtarzamy, zaburzając wartośc drugiego współcznnika o 2^{-23}.

4.2 Metoda rozwiązania

Rozwiązanie znajduje się w pliku `zad4.jl`. Wykorzystujemy w niej obiekty z pakietu `Polynomials`. Do stworzenia postaci ogólnej używamy konstruktora `Polynomials` (w przypadku listy to duże P) oraz `fromroots()` do zbudowania obeiktu reprezentującej postać iloczynową. Na koniec używamy funckji `roots` w celu znalezienia zer wielomianu Wilkinsona.

4.3 Otrzymane wyniki

Rozwiązania przedstawie w dwóch tabelkach, w pierwszech rozpatrujemy wielomian Wilkinsona o współczynnikach całkowitych, następnie z lekko zaburzonym współczynnikiem drugim o 2^{-23}.

k	$P(z_k)$	$p(z_k)$	$\lvert z_k - k\rvert$
1	35696.50964788257	36720.50964788227	3.0109248427834245e-13
2	176252.60026668405	192636.60026691604	2.8318236644508943e-11
3	279157.6968824087	362101.69687113096	4.0790348876384996e-10
4	3.0271092988991085e6	2.7649652999648857e6	1.626246826091915e-8
5	2.2917473756567076e7	2.2277473671348542e7	6.657697912970661e-7
6	1.2902417284205095e8	1.2769707122070245e8	1.0754175226779239e-5
7	4.805112754602064e8	4.780526156335614e8	0.00010200279300764947
8	1.6379520218961136e9	1.6337585675856934e9	0.0006441703922384079
9	4.877071372550003e9	4.870348427548107e9	0.002915294362052734
10	1.3638638195458128e10	1.362843071072106e10	0.009586957518274986
11	3.585631295130865e10	3.584087897760478e10	0.025022932909317674
12	7.533332360358197e10	7.531256581876213e10	0.04671674615314281
13	1.9605988124330817e11	1.9602984002587503e11	0.07431403244734014
14	3.5751347823104315e11	3.574748406282602e11	0.08524440819787316
15	8.21627123645597e11	8.215740477766903e11	0.07549379969947623
16	1.5514978880494067e12	1.5514314565843672e12	0.05371328339202819
17	3.694735918486229e12	3.6946500070912217e12	0.025427146237412046
18	7.650109016515867e12	7.650001670877033e12	0.009078647283519814
19	1.1435273749721195e13	1.14351402511197e13	0.0019098182994383706
20	2.7924106393680727e13	2.7923942556843e13	0.00019070876336257925

Tablica 6: Wyniki dla wielomianu Wilkinsona

k	$P(z_k)$	$p(z_k)$	$\|z_k - k\|$
1	20259.872313418207	20259.87231341787	1.6431300764452317e-13
2	346541.4137593836	362925.41376118705	5.503730804434781e-11
3	2.2580597001197007e6	2.448523699173658e6	3.3965799062229962e-9
4	1.0542631790395478e7	1.0280487766874775e7	8.972436216225788e-8
5	3.757830916585153e7	4.691967282113626e7	1.4261120897529622e-6
6	1.3140943325569446e8	2.037447840252475e8	2.0476673030955794e-5
7	3.939355874647618e8	1.7130336640684276e9	0.00039792957757978087
8	1.184986961371896e9	1.870372834263971e10	0.007772029099445632
9	2.2255221233077707e9	1.3757961713967935e11	0.0841836320674414
10	1.0677921232930157e10	1.491107673054507e12	0.6519586830380407
11	1.0677921232930157e10	1.491107673054507e12	1.1109180272716561
12	3.1401962344429485e10	3.296740218390893e13	1.665281290598479
13	3.1401962344429485e10	3.296740218390893e13	2.0458202766784277
14	2.157665405951858e11	9.545850646509295e14	2.518835871190904
15	2.157665405951858e11	9.545850646509295e14	2.7128805312847097
16	4.850110893921027e11	2.7421389644464932e16	2.9060018735375106
17	4.850110893921027e11	2.7421389644464932e16	2.825483521349608
18	4.557199223869993e12	4.252503605819883e17	2.4540214463129764
19	4.557199223869993e12	4.252503605819883e17	2.0043294443099486
20	8.756386551865696e12	1.3743593161201708e18	0.8469102151947894

Tablica 7: Wyniki dla wilominau Wilkinsona z zaburzonym współczynnikiem

4.4 Wnioski

Zadanie jest źle uwarunkowane ze względu na zaburzenia współczynników. Zaburzenie wspołczynnika drugiego niewiele wpłynęło na wyniki i nie różnią się zbyt wiele od prawdziwego wielomianu Wilkinosona. Przy zbyt dużych liczbach, tak jak współczynniki wielomianu, brakuje cyfr znaczących do dokładnej reprezentacji.

5 Zadanie 5

5.1 Opis zadania

W tym zadaniu należało zbadać jakie wyniki zwraca rekurencyjny wzór modelu logistycznego, model wzrostu populacji. Należało przeprowadzić badania dla 3 metodyk. Przeprowadzić 40 iteracji, iteracja za iteracją. Druga metoda to przeprowadzić 4 razy po 10 iteracji, poczym po każdym zatzrymaniu obcinamy odrzucając cyfry po trzecim miejscu po przecinku. Pierwszą metodykę należało przetestować dla dwóch typów zmiennoprzecinkowych `Float32` i `Float64`. Wzór modelu populacji:

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

dla zadanego p_0 i r.

5.2 Metoda rozwiązania

Rozwiązanie znajduje się w pliku `zad5.jl` i prezentuje implementacje powyższego wzoru rekurencyjnego. Ponadto zawiera też przeprowadzone testy 40-iteracji wyrażenia rekrencyjnego oraz w 4 seriach po 10 interacji z zapmaiętaniem wyniku i ponownym odpaleniu wzoru rekrentyjengo z wcześniej zapamiętamy wynikiem.

5.3 Otrzymane wyniki

Wynikiem dla `Float64` po 40-iteracjach bezprzerywania:

- 0.011611238029748606

Wynikiem dla `Float32` po 40-iteracjach bezprzerywania:

- 0.25860548

Wynikiem dla `Float32` po 40-iteracjach w 4 seriach po 10-iteracji:

- 0.71587336

5.4 Wnioski

Równiania rekurecyjne które polegają na wywołaniach poprzednich rekurencji, które zawierają błedy, potęgują błedy z każdą iteracją. Błąd który już instneiej i wykorzystujemy go do kolejnych obliczeń, powoduje kolejne błedy. Obcinanie cyfr znaczących jak i zmiana arytmetyki z `Float32` na `Float64` przy kolejnych iteracjach wpływa na wyniki końcowe.

6 Zadanie 6

6.1 Opis zadania

W zadaniu przeprowadzić eksperymenty dla zadanego wzoru rekurentycjnego:

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

dla n = 0,1,2,3,... i dla pewnego c. Eksperymenty należało przeprowadzić w arytmetyce `Float64` z liczbą iteracji równą 40. Warto było zrobić interpretacje graficzną zadanego ciągu x_{n+1}.

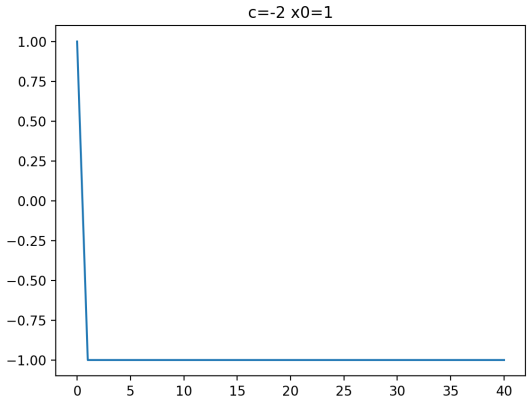
6.2 Metoda rozwiązania

W pliku `zad6.jl` znajduję się implementacja tego zadania.

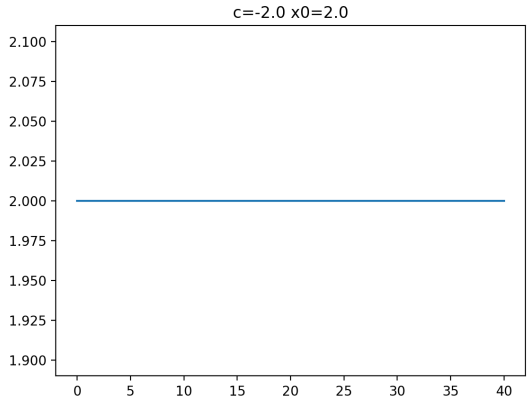
6.3 Otrzymane wyniki

Wykresy dla każdej pary c i x:

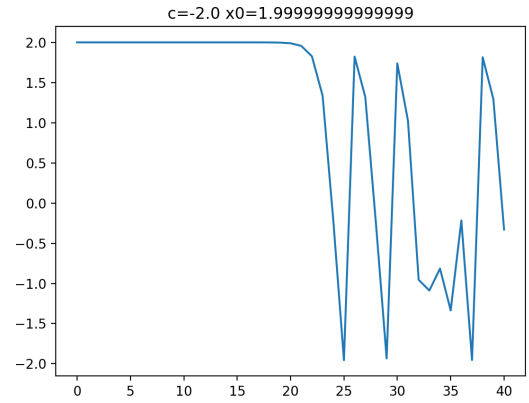
- c = -2, $x_0 = 1$
- c = -2, $x_0 = 2$
- c = -2, $x_0 = 1.99999999999999$
- c = -1, $x_0 = 1$
- c = -1, $x_0 = -1$
- c = -1, $x_0 = 0.75$
- c = -1, $x_0 = 0.25$



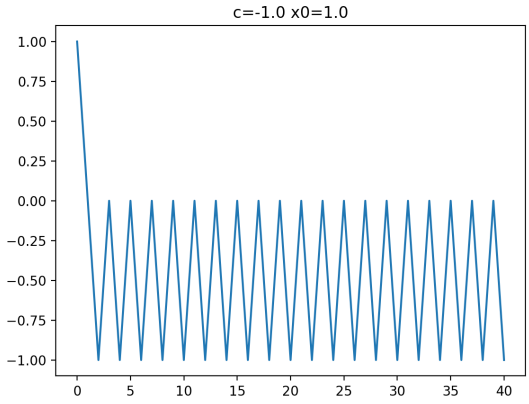
Rysunek 4: Wykres ciągu $c = -2$ i $x_0 = 1$



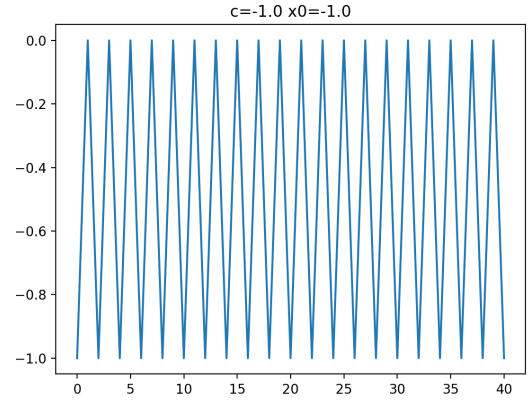
Rysunek 5: Wykres ciągu $c = -2$ i $x_0 = 2$



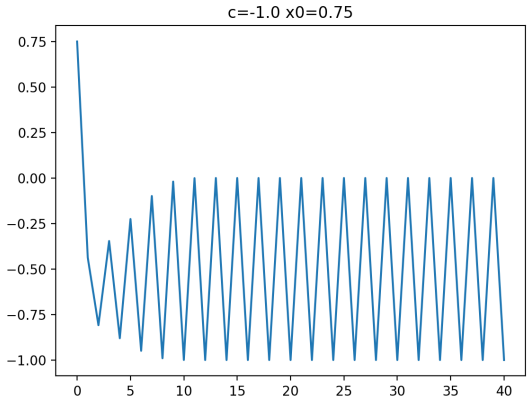
Rysunek 6: Wykres ciągu $c = -2$ i $x_0 = 1.99999999999999$



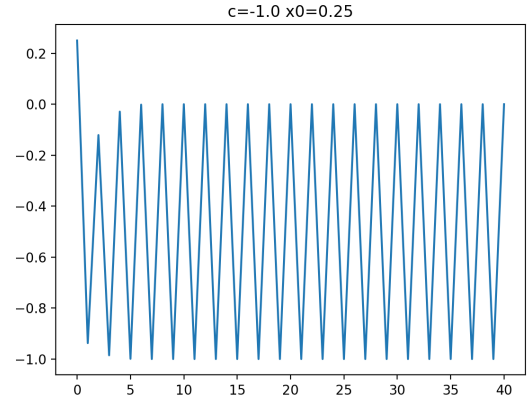
Rysunek 7: Wykres ciągu $c = -1$ i $x_0 = 1$



Rysunek 8: Wykres ciągu $c = -1$ i $x_0 = -1$



Rysunek 9: Wykres ciągu $c = -1$ i $x_0 = 0.75$



Rysunek 10: Wykres ciągu $c = -1$ i $x_0 = 0.25$

Otrzymane wyniki dla 40-iteracji dla danych wejściowych całkowitych działają poprawnie, co nie zaskakuje, w przeciwieństwie do wartości rzeczywistych, gdzie wykresy np. dla $x_0 = 0.25$ lub $x_0 = 0.75$ po kolejncyh iteracjach zbiega do wartości całkowitej z pominięciem wartości bezwględnej. Dla $x_0 = 1.9999999999$ w pierwszej fazie zbiega do liczb całkowitych, nastepnie po osiągnięciu pewnego progu, wpada w niekontrolowany przebieg.

6.4 Wnioski

Skończona dokładność arytmetyki sprawia, że przez kumulacje błędów i przy każdym ich wykorzystaniu w następnych operacjach, otrzymujemy coraz to

mniej dokładne wyniki.