SPRAWOZDANIE NR 2

OBLICZENIA NAUKOWE

AUTOR: MARCIN ADAMCZYK
NR INDEKSU: 221 429

ZAD 1

1. Wstęp
   Zadanie polegało na delikatnej zmianie danych wejściowych dla zadania 5 z listy poprzedniej i porównanie uzyskanych wyników.
   Modyfikacja danych wyglądała następująco:

$$x_4 = 0.5772156649$$
$$x'_4 = 0.577215664$$

$$x_5 = 0.3010299957$$
$$x'_5 = 0.301029995$$

   Jak widać jest ona minimalna i mogłoby się wydawać nieznacząca.

2. Wyniki

Kolejność działania i arytmetyka	Wyniki dla standardowych danych	Wyniki dla zmodyfikowanych danych
forward 64:	1.0251881368296672e-10	-0.004296342739891585
backward 64:	-1.5643308870494366e-10	-0.004296342998713953
maxFirst 64:	0.0	-0.004296342842280865
minFirst 64:	0.0	-0.004296342842280865
forward 32:	-0.4999443	-0.4999443
backward 32:	-0.4543457	-0.4543457
maxFirst 32:	-0.5	-0.5
minFirst 32:	-0.5	-0.5

3. Interpretacja:
   Od razu można zauważyć, że na wyniki w arytmetyce Float32 zmiana nie miała żadnego wpływu ze względu na stosunkowo niską precyzję obliczeń. Analiza zapisu bitowego pozwala zauważyć, że reprezentacja x_4 i x'_4 w arytmetyce Float32 wygląda identycznie, a w przypadku x_5 i x'_5 różnica występuje jedynie na ostatnim – najmniej znaczącym – bicie.

   W arytmetyce Float64 różnice pomiędzy liczbami zmodyfikowanymi, a niezmodyfikowanymi są już wyraźnie widoczne. Tak samo zmiany jakie powstały w wynikach. Po pierwsze nie uzyskaliśmy wyników zerowych ponieważ tym razem nie weszliśmy w granice zera maszynowego. Wyniki jednak są dalekie od oczekiwanych, ale jest to spowodowane faktem, że pracujemy na wektorach prawie prostopadłych, co zawsze powodować będzie błędy.

4. Podsumowanie
   Podczas pracy na wektorach prostopadłych każde zauważalne w zapisie komputerowym zmiany danych powodują znaczne odchylenia końcowych wyników.

ZAD 2

1. Wstęp
   Zadanie to polegało na obserwacji zachowań rozwiązań układów równań nieliniowych w zależności od typu użytej macierzy.

2. Rozwiązanie (notacja jak w treści zadania)
   a. Za pomocą załączonych metod obliczam macierz A,
   b. Jako x przyjmuję tablicę „jedynek”,
   c. Za pomocą A oraz x obliczam wektor b (b = Ax),
   d. Stosując dwie różne metody obliczania wektora x:

$$x = A\backslash b$$
$$x = A^{-1}b$$

   wykonuję obliczenia używając macierzy Hilberta oraz losowych (z zadanym rzędem oraz stopniem uwarunkowania).

3. Wyniki
   Tabele przedstawiające błędy względne liczone za pomocą wzoru:

$$\frac{\|x - x'\|}{\|x\|}$$

n	Macierz Hilberta $x = A\backslash b$	Macierz Hilberta $x = A^{-1}b$
1	0.0	0.0
2	5.661048867003676e-16	1.1240151438116956e-15
3	7.803480619076007e-15	1.7798690734292922e-14
4	4.771929771129762e-13	2.3982306577889774e-13
5	7.781912611742795e-13	8.570321509875875e-12
6	4.348887405734815e-10	3.999474326234716e-10
7	1.4178651011536586e-8	9.49155253157802e-9
8	1.8200901573254684e-7	4.437305576592919e-7
9	1.094603705009265e-5	1.0335179886769292e-5
10	0.00012200718344818623	0.00023094985118418104
11	0.007817692255462364	0.009999393319240569
12	0.2560179095424327	0.25716746002941454
13	5.1042061856444265	5.854248952226193
14	0.9233446266331113	2.8790314865123943
15	2.045398212000293	3.935296436865408
16	4.037502661816079	24.322225494109322
17	4.760220133630642	7.62679661899025
18	8.166550120131358	10.818783988611026
19	5.335024162253255	3.9325939823478704
20	26.308128702224078	202.66020538362136

Macierze losowe generowane za pomocą funkcji *matcond(n,c)*, gdzie *n* jest żądanym rzędem macierzy, a *c* jej wskaźnikiem uwarunkowania.

n	c	Macierz losowa $x = A\backslash b$	Macierz losowa $x = A^{-1}b$
5	1	2.220446049250313e-16	1.85775845048325e-16
5	10	2.9790409838967276e-16	3.1006841635969763e-16
5	10^3	2.5266441837367804e-14	4.923185078068423e-15
5	10^7	8.358146137677174e-11	1.632620452475937e-10
5	10^{12}	1.928778092418009e-6	3.0080541739642993e-5
5	10^{16}	0.013827874428365418	0.13288227732391025
10	1	2.432376777795247e-16	2.1065000811460205e-16
10	10	2.1925147983971603e-16	2.3551386880256624e-16
10	10^3	9.490686183531022e-15	4.4382767531585776e-14
10	10^7	3.17939196892781e-10	1.247824359300068e-10
10	10^{12}	2.3373006071963992e-6	6.369969746846817e-5
10	10^{16}	0.09355615310227149	0.04224270188566521
20	1	5.644695297351525e-16	4.697175049207787e-16
20	10	5.709828552468728e-16	6.582227588661398e-16
20	10^3	2.5666713991868842e-14	3.084034902906653e-14
20	10^7	1.1379734950094825e-10	1.2953028200740527e-10
20	10^{12}	1.1082652868575399e-5	8.84258562892245e-6
20	10^{16}	0.1656828265777694	0.22135073957281343

Bez problemu można zauważyć zależność błędu względnego obliczeń od wskaźnika uwarunkowania.
Dla macierzy Hilberta błędy wzrastają wraz z jej wielkością (co wiąże się z rosnącym wskaźnikiem uwarunkowania).
Adnotacja:
Dla macierzy Hilberta wielkości 20 wskaźnik ten wynosi w przybliżeniu $2.78 * 10^{19}$.

Brak wpływu samej wielkości macierzy na błędy widać przy macierzy losowej, gdzie ewidentnie istnieje bezpośrednią zależność błędu od wskaźnika uwarunkowania (c).

4. Podsumowanie
   Używanie macierzy o wysokim stopniu uwarunkowania generuje duże błędy w obliczeniach.

ZAD 3

1. Wstęp
   W zadaniu tym należało przeanalizować błędy powstałe przez używanie wielomianu Wilkinsona. Wielomian ten w programie zapisany był w dwóch postaciach:

$$p(x) = (x-20)(x-19)\dots(x-1)$$
$$P(x) = x^{20} - 210x^{19} + 20615x^{18}\dots$$

   (kompletna postać funkcji P(x) w treści zadania).
2. Rozwiązanie
   a. utworzenie wielomianu $P(x)$ przy użyciu współczynników z pliku tekstowego,
   b. policzenie pierwiastków tego wielomianu i zapisanie ich w tablicy A,
   c. utworzenie wielomianu $p(x)$,
   d. policzenie wartości wielomianów $P(x)$ oraz $p(x)$ dla wcześniej policzonych pierwiastków znajdujących się w tablicy A,
   e. analiza wyników i rozbieżności.

3. Wyniki

A[k] oznacza k-te miejsce zerowe wielomianu

k	P(A[k])	p(A[k])
1	229376.0 + 0.0im	230400.0 + 0.0im
2	1.209856e6 + 0.0im	1.22624e6 + 0.0im
3	5.04832e6 + 0.0im	5.122048e6 + 0.0im
4	2.34368e7 + 0.0im	2.3698944e7 + 0.0im
5	1.1296512e8 + 0.0im	1.1373312e8 + 0.0im
6	5.03290368e8 + 0.0im	5.04617472e8 + 0.0im
7	1.968515584e9 + 0.0im	1.97132544e9 + 0.0im
8	6.86119424e9 + 0.0im	6.86539008e9 + 0.0im
9	2.1149393408e10 + 0.0im	2.1155347456e10 + 0.0im
10	6.454844928e10 + 0.0im	6.4558998016e10 + 0.0im
11	1.43521440768e11 + 0.0im	1.43537213952e11 + 0.0im
12	5.32673099264e11 + 0.0im	5.32696562176e11 + 0.0im
13	6.02947259392e11 + 0.0im	6.02975376896e11 + 0.0im
14	2.473342619648e12 + 3.6045803326e11im	2.47339307776e12 + 3.6046094056e11im
15	2.473342619648e12 - 3.6045803326e11im	2.47339307776e12 - 3.6046094056e11im
16	7.95998408192e12 + 0.0im	7.960054000128e12 + 0.0im
17	1.183580855552e13 + 0.0im	1.1835897310208e13 + 0.0im
18	2.248445502208e13 + 0.0im	2.2484563546624e13 + 0.0im
19	3.7714572212736e13 + 0.0im	3.7714711862272e13 + 0.0im
20	6.5522804164608e13 + 0.0im	6.5522968009216e13 + 0.0im

Jak widać „perfidny” wielomian Wilkinsona każde, nawet minimalne odchylenie na poziomie przekazanego argumentu, przekształca w bardzo duży błąd w końcowym wyniku. To co chcieliśmy uzyskać to zera, a uzyskane wyniki wchodzą na poziom bilionów.

Różnice pomiędzy prawdziwymi pierwiastkami, a tymi obliczonymi przez program:

k	A[k] - k
1	-1.8831602943691905e-12 + 0.0im
2	1.8919177335874338e-10 + 0.0im
3	-7.380310584892413e-9 + 0.0im
4	1.9601274114933176e-7 + 0.0im
5	-3.6977964725792845e-6 + 0.0im
6	4.8439601833649704e-5 + 0.0im
7	-0.0004423699590061503 + 0.0im
8	0.0028910698579363014 + 0.0im
9	-0.013306957810753417 + 0.0im
10	0.04997403713946724 + 0.0im
11	-0.11398306473093456 + 0.0im
12	0.35865751923029876 + 0.0im
13	-0.4388066058601936 + 0.0im
14	0.5189593087228292 + 0.2133045589544431im
15	-0.48104069127717075 - 0.2133045589544431im
16	0.2067945870631469 + 0.0im
17	-0.11428331176867701 + 0.0im
18	0.030097274474776725 + 0.0im
19	-0.006097819409536243 + 0.0im
20	0.0005420937027018624 + 0.0im

Wyraźnie widać wzrost błędu wraz ze wzrostem wartości obliczanego pierwiastka. Teraz widać, że regularny wzrost odchylenia przy obliczaniu wartości wielomianu, bezpośrednio związany jest ze wzrostem odchylenia wartości samych argumentów przekazywanych do wielomianu.
Błąd na poziomie 12 miejsc po przecinku w argumencie, przekształcony w wynik 229376.0 zamiast 0.0 daje proste wnioski na temat specyfiki tego wielomianu.

Eksperyment nr 2
Drugi eksperyment na wielomianie Wilkinsona, przebiegać miał identycznie jak pierwszy, jednak tym razem zmieniony został współczynnik przy x^{19} z -210 na $-210 - 2^{-23}$.
Zmiana wydawać by się mogło pomijalnie mała (w zapisie bitowym prezentuje się następująco):
11000000011010100100000000000000000000**0**00000000000000000000
11000000011010100100000000000000000000**1**00000000000000000000

Jeżeli chodzi o różnice, to zauważalne są zdecydowanie wyższe błędy obliczonych pierwiastków, dodatkowo podniesione przez znacznie częściej występujące i wyższe części urojone liczb. Efekty takiego odchylenia widać doskonale podczas obliczania wartości obu wielomianów dla „pierwiastków".
W przypadku obliczeń dla P(x) wyniki wykraczają w część urojoną na poziomie od 10^{10} do 10^{13}. Błędy samej części rzeczywistej pozostają na poziomie poprzedniego eksperymentu.

Natomiast obliczenia dla p(x) generują już znacznie wyższe odchylenia, nawet na poziomie 10^{17} zarówno w przypadku części rzeczywistej jak i urojonej.

4. Podsumowanie
„Perfidny" wielomian Wilkinsona potęguje błędy w tempie tak szybkim, że jakiekolwiek obliczenia są w zasadzie niemożliwe.

ZAD 4

1. Wstęp
W zadaniu tym należało przeprowadzić symulację tzw. modelu wzrostu populacji.

2. Rozwiązanie:
Równanie rekurencyjne, na podstawie którego przeprowadzane były obliczenia przedstawia się w następujący sposób:

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

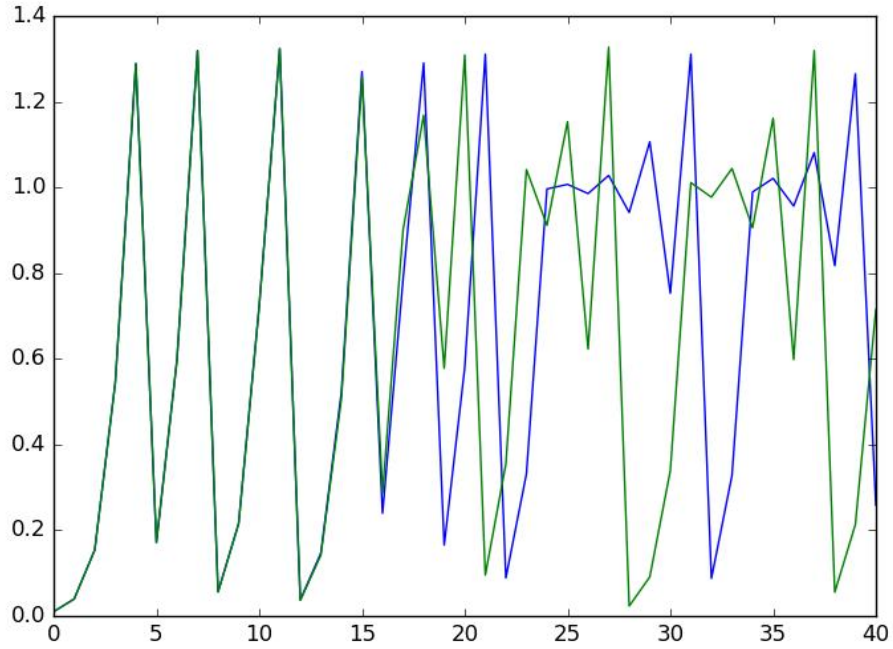
gdzie n = 0, 1,…, , a r jest pewną stałą.

Iteracje wykonane zostały na dwa sposoby:
   a. Normalny – zgodny z równaniem (obliczenia dla Float32 i Float64)
   b. Co 10. Iterację wynik był ucinany do 3 miejsc po przecinku (tylko Float32)

3. Wyniki

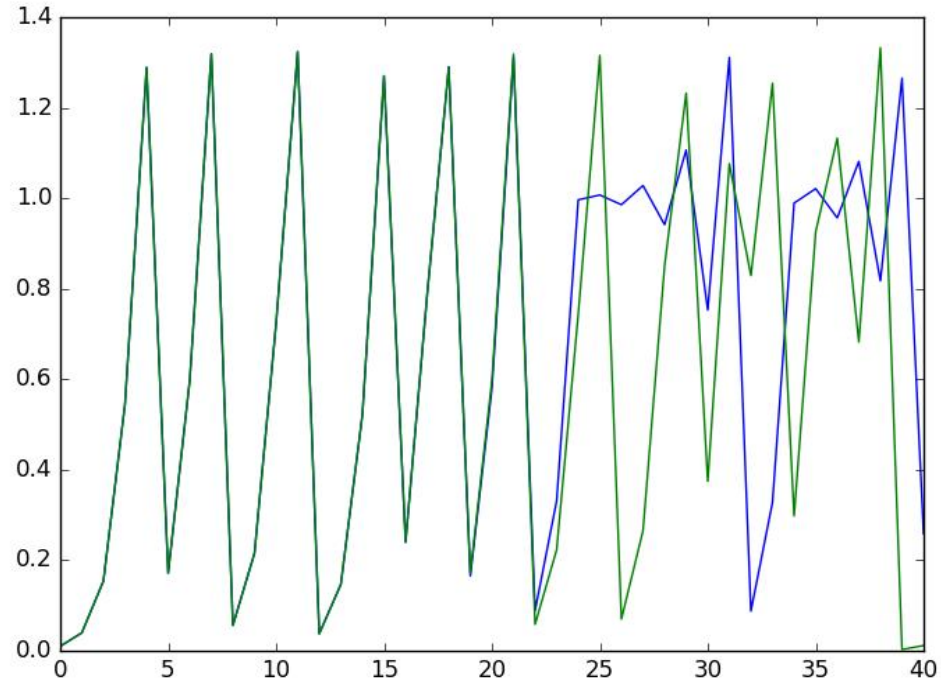
n	Float64	Float32	Float32 z ucinaniem
0	0.01	0.01	0.01
1	0.0397	0.0397	0.0397
2	0.15407173000000002	0.15407173	0.15407173
3	0.5450726260444213	0.5450726	0.5450726
4	1.2889780011888006	1.2889781	1.2889781
5	0.17151914210917552	0.1715188	0.1715188
6	0.5978201201070994	0.5978191	0.5978191
7	1.3191137924137974	1.3191134	1.3191134
8	0.056271577646256565	0.056273222	0.056273222
9	0.21558683923263022	0.21559286	0.21559286
10	0.722914301179573	0.7229306	0.722
11	1.3238419441684408	1.3238364	1.3241479
12	0.03769529725473175	0.037716985	0.036488414
13	0.14651838271355924	0.14660022	0.14195944
14	0.521670621435246	0.521926	0.50738037
15	1.2702617739350768	1.2704837	1.2572169
16	0.24035217277824272	0.2395482	0.28708452
17	0.7881011902353041	0.7860428	0.9010855
18	1.2890943027903075	1.2905813	1.1684768
19	0.17108484670194324	0.16552472	0.577893
20	0.5965293124946907	0.5799036	1.309
21	1.3185755879825978	1.3107498	0.095556974
22	0.058377608259430724	0.088804245	0.3548345
23	0.22328659759944824	0.3315584	1.0416154
24	0.7435756763951792	0.9964407	0.91157377
25	1.315588346001072	1.0070806	1.1533948
26	0.07003529560277899	0.9856885	0.62262046
27	0.26542635452061003	1.0280086	1.3275132
28	0.8503519690601384	0.9416294	0.023178816
29	1.2321124623871897	1.1065198	0.091103494
30	0.37414648963928676	0.7529209	0.339
31	1.0766291714289444	1.3110139	1.0112369
32	0.8291255674004515	0.0877831	0.9771474
33	1.2541546500504441	0.3280148	1.0441384
34	0.29790694147232066	0.9892781	0.90587854
35	0.9253821285571046	1.021099	1.1616664
36	1.1325322626697856	0.95646656	0.59825915
37	0.6822410727153098	1.0813814	1.3192946
38	1.3326056469620293	0.81736827	0.05556369
39	0.0029091569028512065	1.2652004	0.21299279
40	0.011611238029748606	0.25860548	0.715

Wyniki w takiej formie jednak nie przekazują jednak wielu informacji dlatego omówienie rezultatów przeprowadzę przy pomocy wykresów.



Powyższy wykres przedstawia przebieg iteracji w arytmetyce Float32. Niebieska linia przedstawia iterację normalną, zielona, z ucinaną końcówką wyniku. Tutaj widać doskonale, jak początkowo ledwo zauważalny błąd wraz z kolejnymi iteracjami błąd staje się tak duży, że ostatecznie mamy do czynienia z dwoma różnymi iteracjami.

Porównanie Float32 (niebieska linia) i Float64 (zielona linia):



To co można zauważyć na tym wykresie, to różnica w regularności przebiegu wykresu. W tym wypadku niższa precyzja Float32 spowodowała, że początkowo minimalne różnice z każdą kolejną iteracją się powiększały. Momentem kulminacyjnym jest uzyskanie wyniku

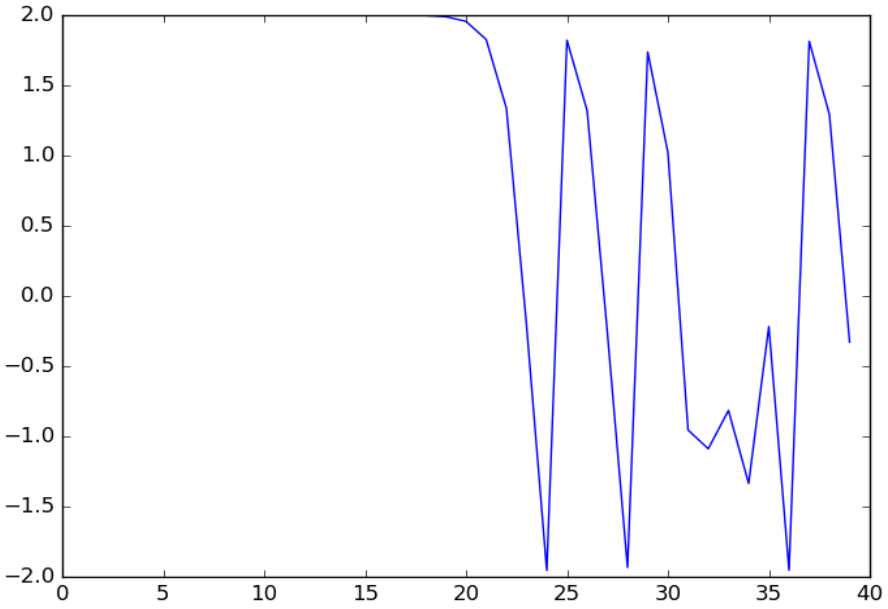
zbliżonego do 1.0, co powoduje oscylowanie następnych iteracji w pobliżu właśnie tej liczby. Widać, że iteracja w arytmetyce Float64 również napotyka taką sytuację, ale znacznie później niż Float32.

4. Podsumowanie:
   Powyższy eksperyment jest typowym przykładem sprzężenia zwrotnego, czyli procesu w którym dane wyjściowe, stanowią dane wejściowe do następnych obliczeń. Można więc zauważyć, że błąd obliczeniowy w wyniku, zostanie przeniesiony na wejście następnej operacji jedynie potęgując powstałe błędy. Choć stosowanie zwiększonej precyzji może pomóc, to jednak w daleko wybiegających symulacjach, błędy będą zniekształcały obliczenia do tego stopnia, że wyniki będą bezużyteczne.

ZAD 5

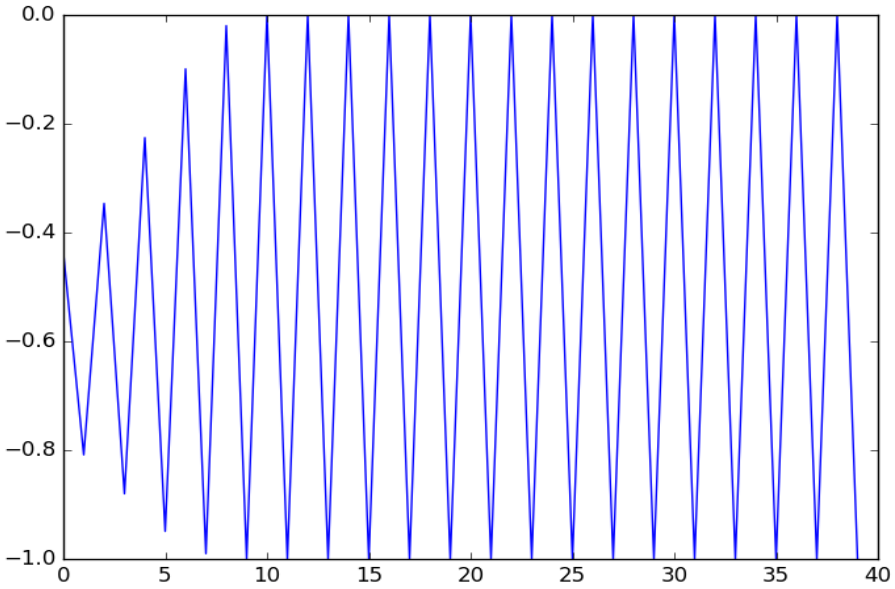
1. Wstęp
   W zadaniu tym należało przeprowadzić serię eksperymentów obliczając wartości równania rekurencyjnego:
   $$x_{n+1} := x_n^2 + c$$
   dla n=0, 1,…, gdzie c jest pewną stała.
2. Wyniki
   Wyniki eksperymentów o numerach 1, 2, 4, 5, pozostawię bez omówienia, gdyż wyniki dla tych równań są zwyczajnie nieciekawe i przewidywalne. Poniżej znajdują się wykresy trzech najciekawszych przypadków. W każdym z nich mamy do czynienia ze zjawiskiem obserwowanym w poprzednim zadaniu czyli sprzężeniem zwrotnym.

   a. $c = -2, x_0 = 1.999999999999999$

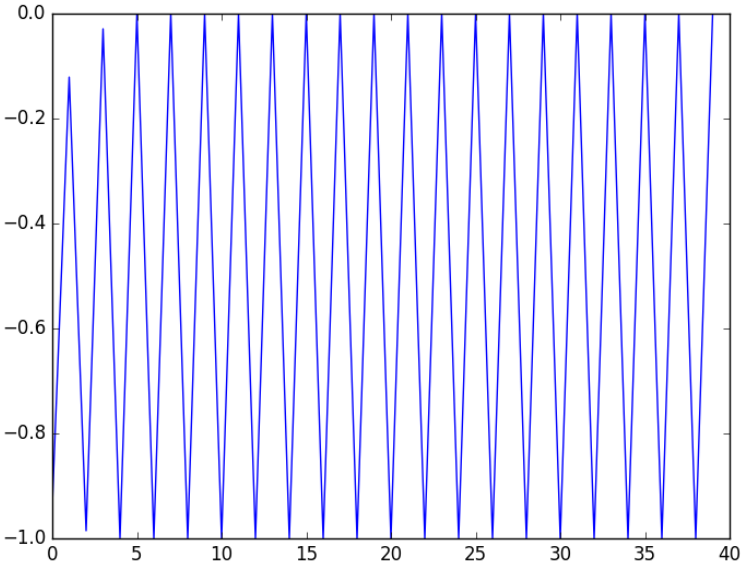


W tym wypadku obserwujemy typowy przypadek potęgowania się błędu podczas używania sprzężenia zwrotnego. Do pewnego momentu błąd jest niezauważalny, a sam komputer przedstawia wyniki jako te same liczby (choć zapis bitowy wykazuje różnice). Błędy jednak ciągle występują, w pewnym momencie stają się zauważalne, aż w końcu można by powiedzieć wywracają wszystko do góry nogami.

b. $c = -1, x_0 = 0.75$



c. $c = -1, x_0 = 0.25$



W powyższych dwóch wypadkach mamy do czynienia z czymś co w książce „Fraktale Granice Chaosu" nazywane jest stabilnością. Wspomniane wcześniej przykłady (nieomawiane ze względu na brak ciekawych zachowań) również były stabilne. Oznacza to, że albo od samego początku(nieomówione przykłady), albo dopiero od pewnego momentu (powyższe dwa przypadki), wyniki stają się regularne, powtarzalne i co najważniejsze przewidywalne.

3. Podsumowanie
W tym zadaniu można było zaobserwować różne zachowania układów sprzężonych:
Te pożądane, czyli stabilizacja dająca przewidywalność wyników,
Te zdecydowanie niepożądane, czyli chaotyczne generowanie wyników spowodowane małymi odchyleniami.