Sprawozdanie z listy drugiej

Karolina Bąk

Listopad 2019

1 Zadanie 1

Zadanie polegało na powtórzeniu zadania 5 z poprzedniej listy dla nieco zaburzonych danych. Ponownie obliczyłam iloczyn skalarny dla wektora:

$$x = (2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957)$$

oraz:

$$y = (1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049)$$

gdzie usunęłam ostatnie 9 z x_4 oraz ostatnie 7 z x_5. Uzyskałam następujące wyniki:

	Float32	Float64
1	-0.4999443	-0.004296342739891585
2	-0.4543457	-0.004296342998713953
3	-0.5	-0.004296342842280865
4	-0.5	-0.004296342842280865

Poprzednie wyniki wyglądały następująco:

	Float32	Float64
1	-0.4999443	1.0251881368296672*10^{-10}
2	-0.4543457	-1.5643308870494366*10^{-10}
3	-0.5	0
4	-0.5	0

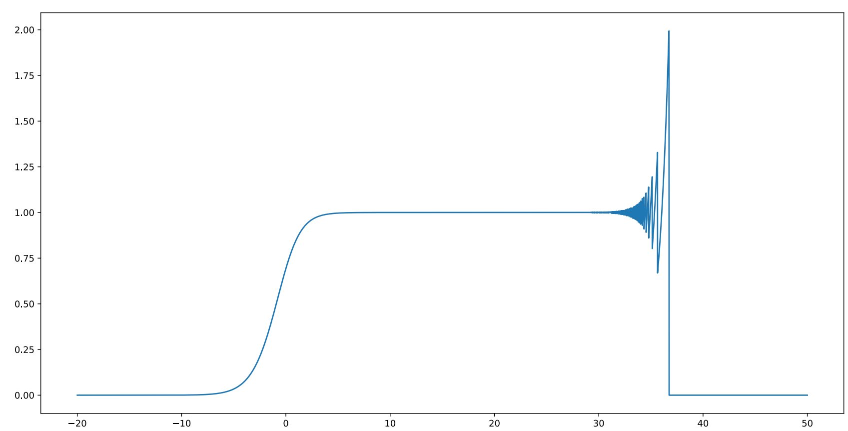
Zmniejszenie liczby cyfr w części dziesiętnej x spowodowało znaczną utratę bliskości do poprawnego wyniku przy Float64 (-1.00657107000000*10^{-11}). Niewielkie zmiany rzędu 10^{-10} spowodowały ogromną zmianę (10^7 razy większy wynik). Potwierdza to wniosek z pierwszej listy, że zadanie jest źle uwarunkowane dla danych tego typu (różne znaki przy współrzędnych).

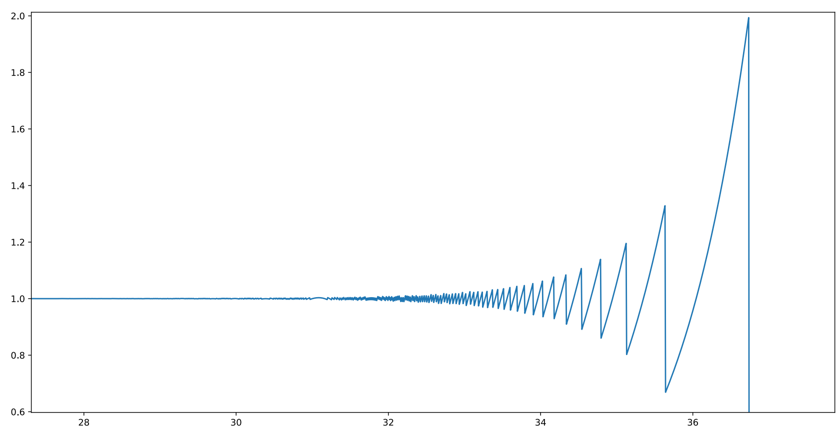
2 Zadanie 2

Celem drugiego zadania było przedstawienie graficznie wykresu funkcji:

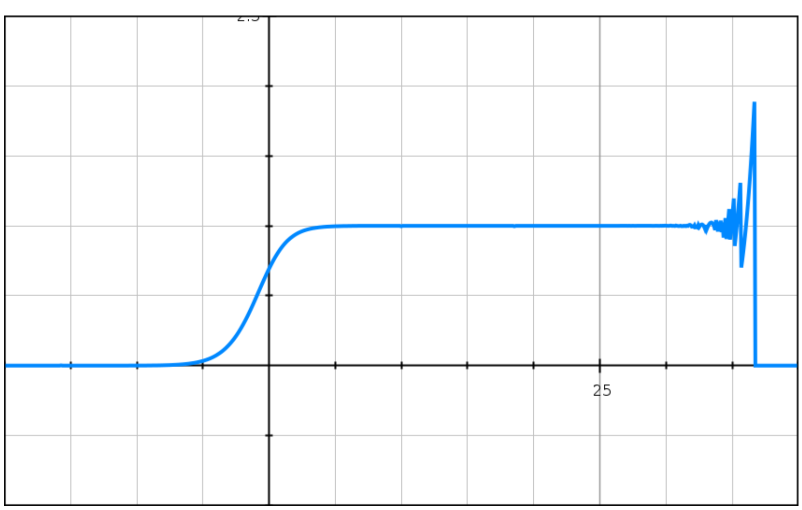
$$f(x) = e^x * ln(1 + e^{-x})$$

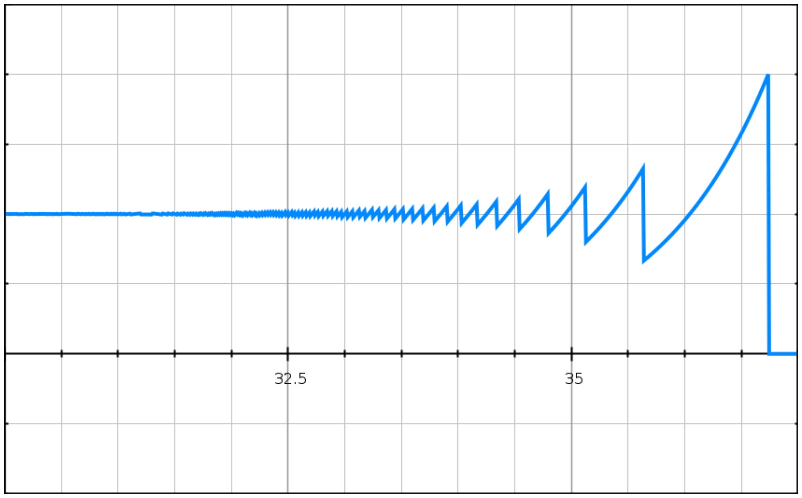
w co najmniej dwóch programach do wizualizacji. Jeden wykres wykonałam przy użyciu Pythona i paczki matplotlib, a drugi on-line na stronie graphsketch.com. Poniżej przedstawiam otrzymane wykresy w Pythonie:





Wykresy z graphsketch.com:





Granicą powyższej funkcji jest 1. Przybliżając wykresy oraz testując dokładniej funkcję dla odpowiednich wartości ustaliłam, że zaburzenia pojawiły się około x = 16, a około x = 18 zaczęły przekraczać 1. Im dalej tym większe było zaburzenie, aż funkcja zmieniła się w stałe 0. Stało się tak, ponieważ działałam na bardzo dużych i bardzo małych liczbach. Mnożąc je ze sobą, stale zwiększałam błąd w obliczeniach, który rósł aż e(-x) zostało pochłonięte przez 1, co wyzerowało logarytm. Dlatego każda następna wartość była równa 0.

3 Zadanie 3

Zadanie polegało na rozwiązaniu układu równań liniowych

$$Ax = b$$

gdzie A jest daną macierzą współczynników A$\in \mathbb{R}^{n\times n}$, a b$\in \mathbb{R}^n$ jest wektorem prawych stron. Macierz A jest generowana na dwa sposoby. Pierwszy to macierz Hilberta o danym stopniu n generowana przez funkcję hilb(n). Drugim sposobem jest generowanie losowej macierzy o zadanym wskaźniku uwarunkowania c przez funkcję matcond(n,c). Wektor b jest generowany przez powyższe równanie, gdzie x=(1,...,1)$^{\mathrm{T}}$. Znane więc są dokładne A oraz b.

Test w zadaniu polega na rozwiązaniu układu na dwa różne sposoby: eliminacją Gaussa x = A\b oraz x = A^{-1}b. Dla macierzy Hilberta uzyskałam poniższe wyniki.

Gauss:

n	$\frac{\|x-xp\|}{\|x\|}$	cond(A)	rank(A)
1	0.0	1.0	1
2	5.661048867003676*10^{-16}	19.28147006790397	2

3	$8.022593772267726*10^{-15}$	524.0567775860644	3
4	$4.137409622430382*10^{-14}$	15513.73873892924	4
5	$1.6828426299227195*10^{-12}$	476607.25024259434	5
6	$2.618913302311624*10^{-10}$	$1.4951058642254665*10^{7}$	6
7	$1.2606867224171548*10^{-8}$	$4.75367356583129*10^{8}$	7
8	$6.124089555723088*10^{-8}$	$1.5257575538060041*10^{10}$	8
9	$3.8751634185032475*10^{-6}$	$4.931537564468762*10^{11}$	9
10	$8.67039023709691*10^{-5}$	$1.6024416992541715*10^{13}$	10
11	0.00015827808158590435	$5.222677939280335*10^{14}$	10
12	0.13396208372085344	$1.7514731907091464*10^{16}$	11
13	0.11039701117868264	$3.344143497338461*10^{18}$	11
14	1.4554087127659643	$6.200786263161444*10^{17}$	11
15	4.696668350857427	$3.674392953467974*10^{17}$	12
16	54.15518954564602	$7.865467778431645*10^{17}$	12
17	13.707236683836307	$1.263684342666052*10^{18}$	12
18	9.134134521198485	$2.2446309929189128*10^{18}$	12
19	9.720589712655698	$6.471953976541591*10^{18}$	13
20	7.549915039472976	$1.3553657908688225*10^{18}$	13

Odwrotna macierz:

n	$\frac{\|x-xp\|}{\|x\|}$	cond(A)	rank(A)
1	0.0	1.0	1
2	$1.4043333874306803*10^{-15}$	19.28147006790397	2
3	0.0	524.0567775860644	3
4	0.0	15513.73873892924	4
5	$3.3544360584359632*10^{-12}$	476607.25024259434	5
6	$2.0163759404347654*10^{-10}$	$1.4951058642254665*10^{7}$	6
7	$4.713280397232037*10^{-9}$	$4.75367356583129*10^{8}$	7
8	$3.07748390309622*10^{-7}$	$1.5257575538060041*10^{10}$	8
9	$4.541268303176643*10^{-6}$	$4.931537564468762*10^{11}$	9
10	0.0002501493411824886	$1.6024416992541715*10^{13}$	10
11	0.007618304284315809	$5.222677939280335*10^{14}$	10
12	0.258994120804705	$1.7514731907091464*10^{16}$	11
13	5.331275639426837	$3.344143497338461*10^{18}$	11
14	8.71499275104814	$6.200786263161444*10^{17}$	11
15	7.344641453111494	$3.674392953467974*10^{17}$	12
16	29.84884207073541	$7.865467778431645*10^{17}$	12
17	10.516942378369349	$1.263684342666052*10^{18}$	12
18	7.575475905055309	$2.2446309929189128*10^{18}$	12
19	12.233761393757726	$6.471953976541591*10^{18}$	13
20	22.062697257870493	$1.3553657908688225*10^{18}$	13

Dla macierzy losowych uzyskałam poniższe wyniki.
Gauss:

n	$\frac{\|x-xp\|}{\|x\|}$	cond(A)	rank(A)
5	1.7901808365247238*10^{-16}	1.0000000000000009	5
5	1.4043333874306804*10^{-16}	9.999999999999998	5
5	1.3136335981433191*10^{-16}	1000.0000000000316	5
5	3.635564360697878*10^{-10}	1.0000000004173718*10^{7}	5
5	9.930136612989092*10^{-17}	9.99966763751934*10^{11}	5
5	0.22407027119013165	1.1171692878820258*10^{16}	4
10	3.1985215122904827*10^{-16}	1.000000000000001	10
10	4.749367485114549*10^{-16}	10.000000000000007	10
10	3.1044345184083204*10^{-14}	1000.0000000000381	10
10	7.312885725957515*10^{-11}	9.99999999487321*10^{6}	10
10	3.5736334640683613*10-^{6}	9.999705522584362*10^{11}	10
10	0.007886044944081804	2.852771852948684*10^{16}	9
20	5.093734210850115*10^{-16}	1.0000000000000013	20
20	7.199349044417091*10^{-16}	9.999999999999991	20
20	1.9013586298626663*10^{-14}	999.9999999999523	20
20	8.344401704472224*10^{-11}	1.0000000003527017*10^{7}	20
20	4.029043035753852*10^{-5}	1.0000489898080726*10^{12}	20
20	0.12546327027700901	6.405885370975909*10^{15}	19

Odwrócona macierz:

n	$\frac{\|x-xp\|}{\|x\|}$	cond(A)	rank(A)
5	2.0471501066083611*10^{-16}	1.0000000000000007	5
5	2.2752801345137457*10^{-16}	9.999999999999998	5
5	1.53220431207386*10^{-14}	1000.0000000000089	5
5	1.0165530953424155*10^{-10}	9.999999993366444*10^{6}	5
5	1.7481138973067626*10^{-5}	9.9993293808524*10^{11}	5
5	0.17936451951089713	5.668621505285428*10^{15}	4
10	2.7866376757248753*10^{-16}	1.0000000000000009	10
10	4.550560269027491*10^{-16}	10.000000000000014	10
10	3.2824171342942656*10^{-14}	1000.0000000000537	10
10	4.020386326707943*10^{-10}	9.999999998237088*10^{6}	10
10	9.19195614625663*10^{-6}	9.999112458768989*10^{11}	10
10	0.12357124133296675	1.0162651221978986*10^{16}	9
20	3.394814396577995*10^{-16}	1.0000000000000013	20

20	5.489713268447767*10^{-16}	10.000000000000005	20
20	2.3174107273898665*10^{-14}	999.9999999999737	20
20	2.3524392971717174*10^{-10}	1.0000000006275691*10^{7}	20
20	1.0904881120709677*10^{-5}	1.0000824146741567*10^{12}	20
20	0.04533606607016923	8.635757378668062*10^{15}	19

Z otrzymanych danych wynika, że im większy wskaźnik uwarunkowania tym większy błąd względny, czyli jeśli wskaźnik uwarunkowania jest duży, to zadanie jest źle uwarunkowane. Przy macierzy Hilberta z rosnącym n bardzo szybko zwiększa się cond(A), przez co uzyskane wyniki szybko zaczynają być absurdalne. Od n=10 w macierzy Hilberta można zauważyć, że rząd zaczyna rosnąć coraz wolniej. Ilość tych samych rank() pod rząd rośnie co 1 (2 razy 10, 3 razy 11, 4 razy 12, itd). W losowych macierzach również można zauważyć spadek rzędu na cond()=10^{16}.

4 Zadanie 4

Celem zadania było obliczenie 20 zer wielomianu Wilkinsona w postaci naturalnej. Współczynniki do wykorzystania były podane w zadaniu. Pierwiastki wielomianu obliczyłam korzystając z funkcji roots() z pakietu Polynomials, postać naturalną stworzyłam funkcją Poly(), a iloczynową postać funkcją poly(). Wyniki przetestestowałam następnie z wartościami dla wielomianu w postaci naturalnej oraz iloczynowej.

k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	36352.0	5.517824*10^{6}	3.0109248427834245*10^{-13}
2	181760.0	7.378697629901744*10^{19}	2.8318236644508943*10^{-11}
3	209408.0	3.320413931687578*10^{20}	4.0790348876384996*10^{-10}
4	3.106816*10^{6}	8.854437035384718*10^{20}	1.626246826091915*10^{-8}
5	2.4114688*10^{7}	1.8446752056545675*10^{21}	6.657697912970661*10^{-7}
6	1.20152064*10^{8}	3.320394888870126*10^{21}	1.0754175226779239*10^{-5}
7	4.80398336*10^{8}	5.423593016891272*10^{21}	0.00010200279300764947
8	1.682691072*10^{9}	8.26205014011023*10^{21}	0.0006441703922384079
9	4.465326592*10^{9}	1.196559421646318*10^{22}	0.002915294362052734
10	1.2707126784*10^{10}	1.6552601335207813*10^{22}	0.009586957518274986
11	3.5759895552*10^{10}	2.2478332979247994*10^{22}	0.025022932909317674
12	7.216771584*10^{10}	2.8869446884129956*10^{22}	0.04671674615314281
13	2.15723629056*10^{11}	3.807325552825022*10^{22}	0.07431403244734014
14	3.65383250944*10^{11}	4.612719853149547*10^{22}	0.08524440819787316
15	6.13987753472*10^{11}	5.901011420239329*10^{22}	0.07549379969947623
16	1.555027751936*10^{12}	7.01087410689741*10^{22}	0.05371328339202819

17	$3.777623778304*10^{12}$	$8.568905825727875*10^{22}$	0.025427146237412046
18	$7.199554861056*10^{12}$	$1.0144799361089491*10^{23}$	0.009078647283519814
19	$1.0278376162816*10^{13}$	$1.1990376202486947*10^{23}$	0.0019098182994383706
20	$2.7462952745472*10^{13}$	$1.4019117414364248*10^{23}$	0.00019070876336257925

Otrzymane wartości dla wielomianu mimo dość niewielkich błędów przy pierwiastkach są bardzo oddalone od oczekiwanego zera. Dzieje się tak, gdyż współczynniki wielomianu są większe niż $4.5*10^{15}$, a od tego momentu liczby zmiennoprzecinkowe są oddalone od siebie o wartości większe od 1 i ich reprezentacja może być niedokładna. Do błędu przy wyznaczeniu pierwiastków przyczynia się również ogromna różnica między liczbami, na których operowałam. Także przy dość wysokich wykładnikach w potęgowaniu uzyskiwane wyniki są obarczone dużym błędem, co zauważyłam już w poprzednim zadaniu potęgując 10.

Następnym celem zadania było powtórzenie eksperymentu Wilkinsona. Polega ono na zaburzeniu dziewiętnastego współczynnika o 2^{-23}. Dla tych danych uzyskałam następujące wyniki.

k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	20992.0	$3.012096*10^{6}$	$1.6431300764452317*10^{-13}$
2	349184.0	$7.37869763029606*10^{19}$	$5.503730804434781*10^{-11}$
3	$2.221568*10^{6}$	$3.3204139201100146*10^{20}$	$3.3965799062229962*10^{-9}$
4	$1.046784*10^{7}$	$8.854437817429645*10^{20}$	$8.972436216225788*10^{-8}$
5	$3.9463936*10^{7}$	$1.8446726974084148*10^{21}$	$1.4261120897529622*10^{-6}$
6	$1.29148416*10^{8}$	$3.320450195282314*10^{21}$	$2.0476673030955794*10^{-5}$
7	$3.88123136*10^{8}$	$5.422366528916045*10^{21}$	0.00039792957757978087
8	$1.072547328*10^{9}$	$8.289399860984229*10^{21}$	0.00777202909944563 2
9	$3.065575424*10^{9}$	$1.1607472501770085*10^{22}$	0.0841836320674414
10	$7.143113638035824*10^{9}$	$1.7212892853671066*10^{22}$	0.6519586830380406
11	$7.143113638035824*10^{9}$	$1.7212892853671066*10^{22}$	1.1109180272716561
12	$3.357756113171857*10^{10}$	$2.8568401004080516*10^{22}$	1.665281290598479
13	$3.357756113171857*10^{10}$	$2.8568401004080516*10^{22}$	2.045820276678428
14	$1.0612064533081976*10^{11}$	$4.934647147685479*10^{22}$	2.5188358711909045
15	$1.0612064533081976*10^{11}$	$4.934647147685479*10^{22}$	2.7128805312847097
16	$3.3151034759817638*10^{11}$	$8.484694713574187*10^{22}$	2.9060018735375106
17	$3.315103475981763*10^{11}$	$8.484694713574187*10^{22}$	2.825483521349608
18	$9.539424609817828*10^{12}$	$1.318194782057474*10^{23}$	2.454021446312976
19	$9.539424609817828*10^{12}$	$1.318194782057474*10^{23}$	2.004329444309949
20	$1.114453504512*10^{13}$	$1.591108408283123*10^{23}$	0.8469102151947894

Jak widać błąd w wyznaczonych pierwiastkach o wiele się zwiększył. Niektóre różnice zaczęły dochodzić nawet do 3. W samych zerach wielomianu pojawiły się pierwiastki zespolone. Ta sytuacja pokazała, że mikroskopijne zmiany mogą w tym wypadku drastycznie wpłynąć na otrzymane wyniki, co znaczy, że wielomian Wilkinsona jest źle uwarunkowany.

5 Zadanie 5

Zadanie polega na przetestowaniu równania rekurencyjnego (model logistyczny, wzrostu populacji)

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

gdzie r jest pewną dana stałą, $\mathrm{r}(1 - \mathrm{p}_n)$ jest czynnikiem wzrostu populacji, a p_0 jest wielkością populacji stanowiącą procent maksymalnej wielkości populacji dla danego stanu środowiska.

Przeprowadziłam 3 różne eksperymenty:

- 40 iteracji dla Float64
- 40 iteracji dla Float32
- 40 iteracji, gdzie co dziesięć iteracji zaokrąglałam wynik do trzech miejsc po przecinku dla Float32

Poniżej przedstawiam uzyskane wyniki w kolejności od największej do najmniejszej dokładności:

n	Float64	Float32	Float32 modyfikacja
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	0.7229306
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	1.3096911
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	0.33951443
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	0.71587336

Najszybciej wyniki zaczęły się znacząco różnić dla trzeciego eksperymentu (około 14. iteracji), gdzie co 10. iterację ucinałam cyfry z rozszerzenia wyniku. Wyniki między Float64 a Float32 rozjechały się na poziomie 10^{-2} na 18. iteracji. Dzieje się tak, ponieważ co iteracje do dokładnego przedstawienia wyniku potrzeba 2 razy większej dokładności. Dlatego bardzo szybko pojawia się błąd, który ma wpływ na następne wyniki. Przez to parę iteracji później wyniki przestają być poprawne (około 16 iteracji było potrzebne by Float32 i Float64 różniły się na 3. miejscu po przecinku.).

6 Zadanie 6

Zadanie polegało na przetestowaniu kolejnego równania rekurencyjnego postaci

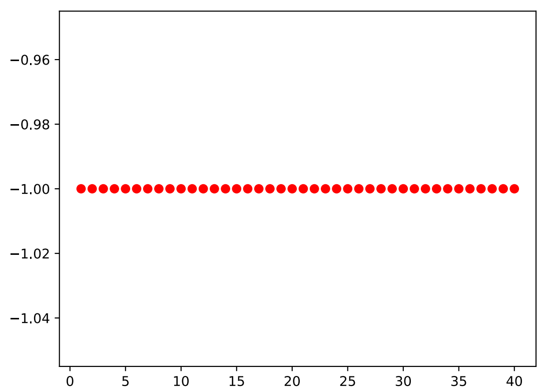
$$x_{n+1} := x_n^2 + c$$

dla n = 1,2.. , gdzie c jest pewną stałą. W zadaniu należało przeprowadzić po 40 iteracji dla poniższych danych:

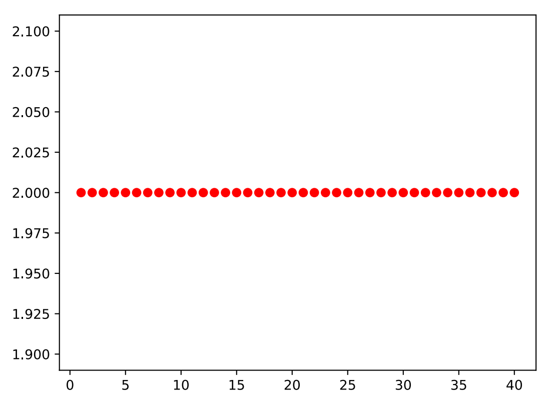
1. $c = -2$ i $x_0 = 1$
2. $c = -2$ i $x_0 = 2$
3. $c = -2$ i $x_0 = 1.99999999999999$
4. $c = -1$ i $x_0 = 1$
5. $c = -1$ i $x_0 =$ -1
6. $c = -1$ i x_0= 0.75
7. $c = -1$ i $x_0 = 0.25$

Wyniki przedstawię w postaci wykresów:

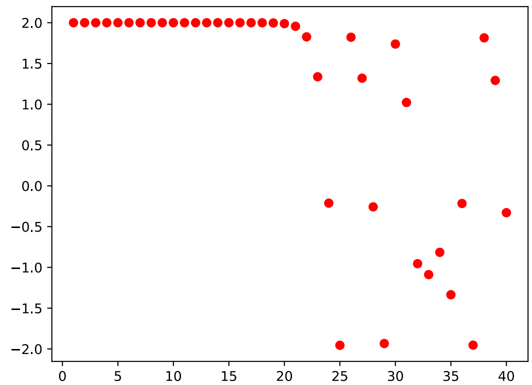
1.



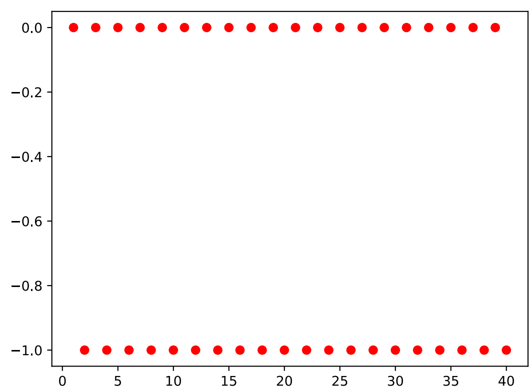
2.



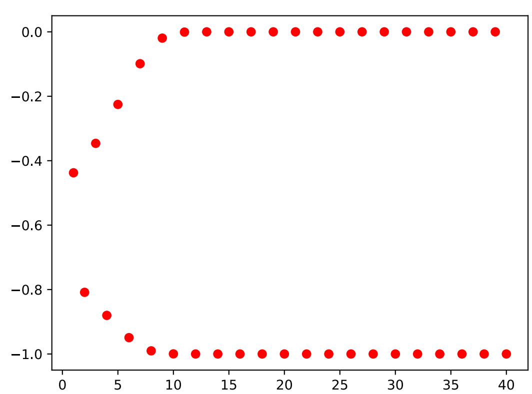
3.


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
5
10
15
20
25
30
35
40

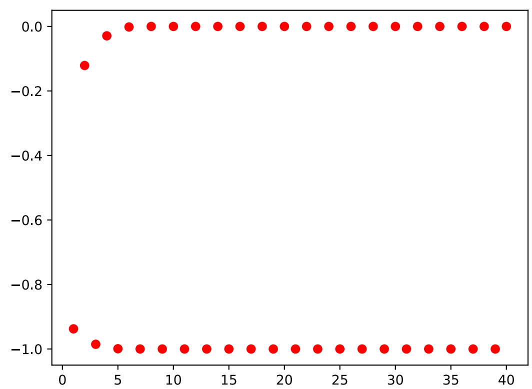
4 i 5.


0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
5
10
15
20
25
30
35
40

6.


0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
5
10
15
20
25
30
35
40

7.



Podobnie jak w poprzednim zadaniu zaburzenie wywołało niedorzeczne wyniki. Jednak pojawiły się one tylko przy teście numer 3 ($x_0 = 1.99999999999999$ i c =-2). Odchylenie od 2 wyniosło tu 10^{-14}i po około 20 iteracjach zaczęło się widocznie psuć. Stało się tak, ponieważ zaburzenie było bliskie precyzji arytmetyki i błąd szybko przeniósł się na następne iteracje. W reszcie testów nie było takich zaburzeń więc uzyskane tam wyniki można uznać za poprawne.