Sprawozdanie

Obliczenia naukowe, lista 2

Aleksandra Malinowska, 244925, WPPT INF, semestr 5, listopad 2019

1. Wstęp

Sprawozdanie to zawiera analizę problemów oraz wyniki ich rozwiązań z zadań z listy nr 2 doktora Pawła Zielińskiego z przedmiotu Obliczenia naukowe. Wszystkie programy potrzebne do rozwiązania zadań zostały napisane w języku Julia.

2. Lista zadań

2.1. Zadanie 1

2.1.1. Opis problemu i rozwiązanie

Problemem tego zadania było sprawdzenie, jakie znaczenie dla wyników będzie miało lekkie zaburzenie danych. Eksperyment polegał na zmianie ostatnich cyfr we współrzędnych wektora X z zadania 5 z listy nr 1, następnie policzenia iloczynu skalarnego wektorów X i Y oraz porównania wyników.

2.1.2. Wyniki

Poniżej znajdują się wyniki uzyskane dla niezmienionych danych (wyniki z listy nr 1).

Metoda	**Float32**	**Float64**
1 („w przód”)	`-0.4999443`	`1.0251881368296672e-10`
2 („w tył”)	`-0.4543457`	`-1.5643308870494366e-10`
3	`-0.5`	`0.0`
4	`-0.5`	`0.0`

Wyniki po zaburzeniu wektora X przedstawia poniższa tabela.

Metoda	**Float32**	**Float64**
1 („w przód”)	`-0.4999443`	`-0.004296342739891585`
2 („w tył”)	`-0.4543457`	`-0.004296342998713953`
3	`-0.5`	`-0.004296342842280865`
4	`-0.5`	`-0.004296342842280865`

2.1.3. Wnioski

Z powyższych danych wyraźnie widać, że dla arytmetyki Float32 zaburzenie wektora X w ogóle nie wpłynęło na zmianę wyniku iloczynu skalarnego. Wynika to z faktu, że zmiana cyfr w wektorze została dokonana na dziesiątym miejscu po przecinku, natomiast obliczenia wykonywane w arytmetyce Float32 nie są na tyle dokładne, aby zmiana ta mogła mieć wpływ na końcowy wynik.

W przypadku arytmetyki Float64 widać różnice w wynikach rzędu dziesięciu wielkości dla metod 1 i 2. Wynika to większej dokładności tej arytmetyki od arytmetyki Float32. Można stąd

wyciągnąć wniosek, że metody zastosowane do obliczenia iloczynu skalarnego wektorów są bardzo wrażliwe na niewielkie zmiany danych, a z tego natomiast wynika, że zadanie jest źle uwarunkowane.

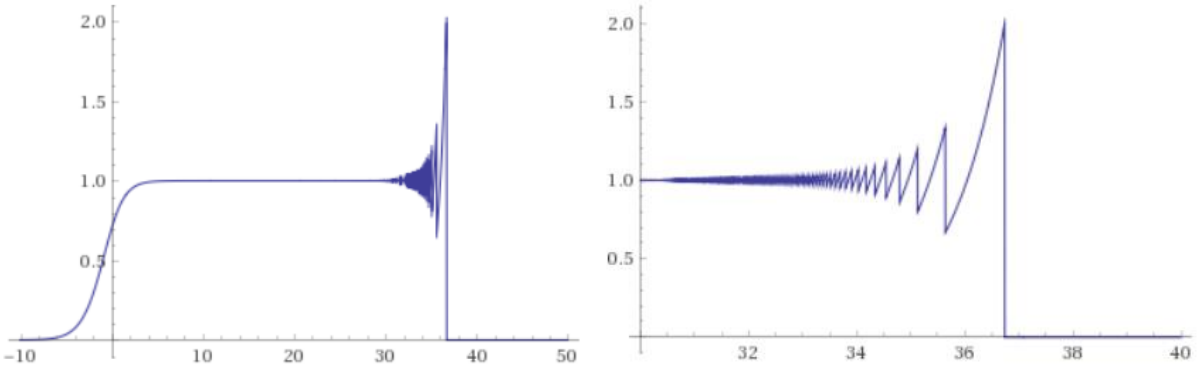
2.2. Zadanie 2

2.2.1. Opis problemu i rozwiązanie

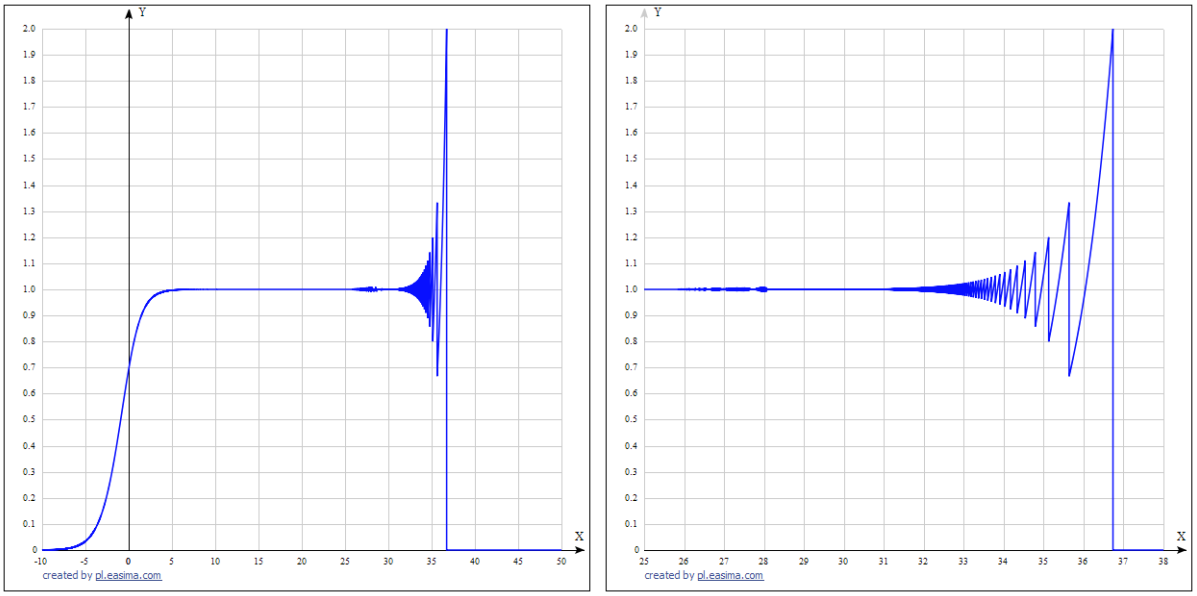
W tym zadaniu należało narysować wykres funkcji $f(x) = e^x * ln(1 + e^{-x})$ w dwóch wybranych programach graficznych, a następnie sprawdzić ich poprawność, porównując wykres z granicą funkcji $\lim_{x\to\infty} f(x)$.

2.2.2. Wyniki

Do narysowania wykresów użyłam programów *WolframAlpha* oraz *Easima*.



Ryc. 1 Wykresy narysowane w programie WolframAlpha



Ryc. 2 Wykresy narysowane w programie Easima

Z powyższych wykresów można odczytać, że funkcja w przedziale $[-10, 5]$ jest rosnąca, natomiast w przedziale $[5, 25]$ jest stała i wynosi 1. W kolejnym przedziale $[25, 37)$ wykres oscyluje w przedziale około $[0.6\,, 1.8]$, po czym dla $x > 37$ spada do 0. Granica tej funkcji wynosi $\lim_{x\to\infty} f(x) = 1$, natomiast z wykresów wynika, że wynosi ona 0, co jest błędnym wynikiem.

2.2.3. Wnioski

Odchylanie wykresu w przedziale $[25, 37]$ wynika z dużej niedokładności obliczeń spowodowanej mnożeniem liczby e^x, której wartość jest bardzo duża, z bardzo małym w tym przedziale logarytmem naturalnym. Natomiast zbieganie funkcji do 0 dla $x > 37$ wynika z dodawania do 1 bardzo małej liczby e^{-x}, co prowadzi do błędu $1 + e^{-x} \approx 1$, skąd bierze się $ln(1 + e^{-x}) \approx 0$. Dlatego właśnie wartość całej funkcji zbiega do 0. Widać tu wyraźnie, że jest to zadanie źle uwarunkowane.

2.3. Zadanie 3

2.3.1. Opis problemu

W tym zadaniu należało znaleźć rozwiązanie układu równań liniowych $Ax = b$ dla danej macierzy współczynników $A \in \mathbb{R}^{n \times n}$ i wektora prawych stron $b \in \mathbb{R}^n$, gdzie A jest macierzą Hilberta lub losową macierzą, natomiast $b = Ax$ $(x = (1, \ldots, 1)^T)$. Układ należało rozwiązać dwiema metodami: (a) metodą eliminacji Gaussa, (b) zgodnie ze wzorem $x = A^{-1}b$. Znając prawidłowe rozwiązanie układu należało je porównać z otrzymanym i obliczyć błąd względny.

2.3.2. Rozwiązanie

Do przeprowadzenia eksperymentów dla różnych danych napisałam dwie funkcje, odpowiadające dwóm metodom wyznaczania macierzy. Funkcja `countHilbert(n)` liczy oraz wyświetla błędy względne dla kolejnych macierzy $A = H_2, H_3, \ldots, H_n$, natomiast funkcja `countRandom()` robi to dla losowych macierzy wygenerowanych funkcją `matcond(n,c)` (plik źródłowy załączony do listy). Wszystkie obliczenia były wykonywane w arytmetyce Float64.

2.3.3. Wyniki

W poniższych tabelach znajdują się wyniki przeprowadzonych eksperymentów.

Pierwsza tabela przedstawia błędy względne dla każdej z metod, porównane ze współczynnikiem uwarunkowania macierzy Hilberta stopnia n.

n	Metoda (a)	Metoda (b)	cond(A)
2	5.661048867003676e-16	1.4043333874306803e-15	19.28147006790397
3	8.022593772267726e-15	0.0	524.0567775860644
4	4.137409622430382e-14	0.0	15513.73873892924
5	1.6828426299227195e-12	3.3544360584359632e-12	476607.25024259434
6	2.618913302311624e-10	2.0163759404347654e-10	1.4951058642254665e7
7	1.2606867224171548e-8	4.713280397232037e-9	4.75367356583129e8
8	6.124089555723088e-8	3.07748390309622e-7	1.5257575538060041e10
9	3.8751634185032475e-6	4.541268303176643e-6	4.931537564468762e11
10	8.67039023709691e-5	0.0002501493411824886	1.6024416992541715e13
11	0.00015827808158590435	0.007618304284315809	5.222677939280335e14
12	0.13396208372085344	0.258994120804705	1.7514731907091464e16

13	0.11039701117868264	5.331275639426837	3.344143497338461e18
14	1.4554087127659643	8.71499275104814	6.200786263161444e17
15	4.696668350857427	7.344641453111494	3.674392953467974e17
16	54.15518954564602	29.84884207073541	7.865467778431645e17
17	13.707236683836307	10.516942378369349	1.263684342666052e18
18	9.134134521198485	7.575475905055309	2.2446309929189128e18
19	9.720589712655698	12.233761393757726	6.471953976541591e18
20	7.549915039472976	22.062697257870493	1.3553657908688225e18

Następna tabela przedstawia zestawione ze sobą błędy względne obu metod dla danego rzędu macierzy n i współczynnika uwarunkowania macierzy c.

n	c	Metoda (a)	Metoda (b)
5	1.0	1.2161883888976234e-16	1.719950113979703e-16
5	10.0	2.7194799110210365e-16	3.1006841635969763e-16
5	1000.0	6.353153646316042e-15	7.105427357601002e-15
5	1.0e7	4.568473684293346e-10	4.3892783444774907e-10
5	1.0e12	6.298234240290979e-6	5.2234893560902225e-6
5	1.0e16	0.008692281011601586	0.01711632992203644
10	1.0	3.9720546451956367e-16	1.1102230246251565e-16
10	10.0	4.002960424867205e-16	4.1540741810552243e-16
10	1000.0	3.490234664007936e-15	1.1360354047922869e-14
10	1.0e7	1.2247764260064769e-10	9.966335733626855e-11
10	1.0e12	1.0935301377586183e-5	1.2467262912644195e-5
10	1.0e16	0.18446334644138096	0.17608480733726006
20	1.0	8.367281836914829e-16	3.236828524569469e-16
20	10.0	7.723833630868742e-16	7.271748933585094e-16
20	1000.0	2.4066500945406244e-14	2.0604973651765453e-14
20	1.0e7	2.7467273338354373e-10	2.935467944218844e-10
20	1.0e12	2.1225942474005693e-5	2.7786148569656765e-5
20	1.0e16	0.5547739737162748	0.3101473943949876

2.3.4. Wnioski

W pierwszej tabeli można zauważyć wprost proporcjonalną zależność błędów względnych od rzędu macierzy Hilberta. Razem z nimi rośnie również współczynnik uwarunkowania tej macierzy, skąd tak ogromny wpływ rzędu macierzy na poprawność wyniku.

W drugiej tabeli błędy również rosną wprost proporcjonalnie do rzędu macierzy i współczynnika jej uwarunkowania, jednak postęp ten nie jest tak szybki jak w przypadku macierzy Hilberta.

Prowadzi to do wniosków, że zadanie jest źle uwarunkowane dla macierzy Hilberta ze względu na wysoką zależność współczynnika uwarunkowania od rzędu macierzy. Macierz generowana losowo z drugiej strony również nie przedstawia w pełni poprawnych wyników, jednak błędy te są akceptowalnie małe.

2.4. Zadanie 4

2.4.1. Opis problemu

Celem tego zadania było obliczenie 20 pierwiastków wielomianu Wilkinsona w postaci naturalnej $P(x)$ oraz ilorazowej $p(x)$. Następnie należało sprawdzić obliczone pierwiastki z_k, $1 \leq z_k \leq 20$, obliczając $|P(z_k)|, |p(z_k)|, |z_k - k|$. Eksperyment ten należało powtórzyć ze zmienionym współczynnikiem.

2.4.2. Rozwiązanie

Do rozwiązania tego zadania należało zainstalować pakiet *Polynomials*. Współczynniki wielomianu w postaci naturalnej zostały podane w pliku `wielomian.txt` (plik źródłowy dołączony do listy). Do pierwszego eksperymentu napisałam funkcję `wilkinsonExperiment()`, która generuje wielomian przy użyciu funkcji `Poly()`, a następnie wyznacza pierwiastki tego wielomianu funkcją `roots()`. W dalszej kolejności funkcja sprawdza poprawność obliczonych pierwiastków – najpierw przy użyciu pętli podstawia je do wielomianu $p(x)$, potem przy użyciu funkcji `polyval()` oblicza wartość wielomianu $P(x)$. Na końcu funkcja liczy różnicę $|z_k - k|$.

Przy drugim eksperymencie (w którym zmieniamy wartość współczynnika przy x^{19}) użyłam funkcji `modifiedWilkinsonEx()`, która od poprzedniej różni się tym, że nie są w niej obliczane wartości wielomianu $p(x)$, ponieważ nie znamy postaci ilorazowej nowego wielomianu. Reszta algorytmu pozostaje niezmieniona.

2.4.3. Wyniki

W wyniku działania funkcji `wilkinsonExperiment()` zostaje wygenerowany następujący wielomian

$$P(x) = 2.43290200817664 * 10^{18} - 8.7529480367616 * 10^{18} * x + 1.3803759753640704 * 10^{19} * x^2 - 1.2870931245150988 * 10^{19} * x^3 + 8.037811822645051 * 10^{18} * x^4 - 3.599979517947607 * 10^{18} * x^5 + 1.2066478037803732 * 10^{18} * x^6 - 3.1133364316139066 * 10^{17} * x^7 + 6.30308120992949 * 10^{16} * x^8 - 1.014229986551145 * 10^{16} * x^9 + 1.307535010540395 * 10^{15} * x^{10} - 1.3558518289953 * 10^{14} * x^{11} + 1.1310276995381 * 10^{13} * x^{12} - 7.561111845 * 10^{11} * x^{13} + 4.017177163 * 10^{10} * x^{14} - 1.67228082 * 10^{9} * x^{15} + 5.3327946 * 10^{7} * x^{16} - 1.25685 * 10^{6} * x^{17} + 20615.0 * x^{18} - 210.0 * x^{19} + 1.0 * x^{20}$$

Poniżej znajduje się zestawienie wyników sprawdzania poprawności otrzymanych pierwiastków tego wielomianu.

k	z_k	$\lvert z_k - k \rvert$	$\lvert P(z_k) \rvert$	$\lvert p(z_k) \rvert$
1.0	0.9999999999996989	3.0109248427834245e-13	36352.0	36626.425482422805
2.0	2.0000000000283182	2.8318236644508943e-11	181760.0	181303.93367257662
3.0	2.9999999995920965	4.0790348876384996e-10	209408.0	290172.2858891686
4.0	3.9999999837375317	1.626246826091915e-8	3.106816e6	2.0415372902750901e6
5.0	5.000000665769791	6.657697912970661e-7	2.4114688e7	2.0894625006962188e7
6.0	5.999989245824773	1.0754175226779239e-5	1.20152064e8	1.1250484577562995e8
7.0	7.000102002793008	0.00010200279300764947	4.80398336e8	4.572908642730946e8
8.0	7.999355829607762	0.0006441703922384079	1.682691072e9	1.5556459377357383e9
9.0	9.002915294362053	0.002915294362052734	4.465326592e9	4.687816175648389e9
10.0	9.990413042481725	0.009586957518274986	1.2707126784e10	1.2634601896949205e10
11.0	11.025022932909318	0.025022932909317674	3.5759895552e10	3.300128474498415e10
12.0	11.953283253846857	0.04671674615314281	7.216771584e10	7.388525665404988e10
13.0	13.07431403244734	0.07431403244734014	2.15723629056e11	1.8476215093144193e11
14.0	13.914755591802127	0.08524440819787316	3.65383250944e11	3.5514277528420844e11
15.0	15.075493799699476	0.07549379969947623	6.13987753472e11	8.423201558964254e11
16.0	15.946286716607972	0.05371328339202819	1.555027751936e12	1.570728736625802e12
17.0	17.025427146237412	0.025427146237412046	3.777623778304e12	3.3169782238892363e12
18.0	17.99092135271648	0.009078647283519814	7.199554861056e12	6.34485314179128e12
19.0	19.00190981829944	0.0019098182994383706	1.0278376162816e13	1.228571736671966e13
20.0	19.999809291236637	0.00019070876336257925	2.7462952745472e13	2.318309535271638e13

Rezultatem działania funkcji `modifiedWilkinsonEx()` jest wielomian (w którym współczynnik -210 został zmieniony na $-210-2^{-23}$)

$$P_m(x) = 2.43290200817664 * 10^{18} - 8.7529480367616 * 10^{18} * x + 1.3803759753640704 * 10^{19} * x^2 - 1.2870931245150988 * 10^{19} * x^3 + 8.037811822645051 * 10^{18} * x^4 - 3.599979517947607 * 10^{18} * x^5 + 1.2066478037803732 * 10^{18} * x^6 - 3.1133364316139066 * 10^{17} * x^7 + 6.30308120992949 * 10^{16} * x^8 - 1.014229986551145 * 10^{16} * x^9 + 1.307535010540395 * 10^{15} * x^{10} - 1.3558518289953 * 10^{14} * x^{11} + 1.1310276995381 * 10^{13} * x^{12} - 7.561111845 * 10^{11} * x^{13} + 4.017177163 * 10^{10} * x^{14} - 1.67228082 * 10^9 * x^{15} + 5.3327946 * 10^7 * x^{16} - 1.25685 * 10^6 * x^{17} + 20615.0 * x^{18} - 210.0000001192093 * x^{19} + 1.0 * x^{20}$$

oraz poniższe zestawienie sprawdzenia poprawności obliczonych pierwiastków.

k	z_k	$\lvert z_k - k\rvert$	$\lvert P_m(z_k)\rvert$
1.0	0.9999999999998357 + 0.0im	1.6431300764452317e-13	20992.0
2.0	2.0000000000550373 + 0.0im	5.503730804434781e-11	349184.0
3.0	2.99999999660342 + 0.0im	3.3965799062229962e-9	2.221568e6
4.0	4.000000089724362 + 0.0im	8.972436216225788e-8	1.046784e7
5.0	4.99999857388791 + 0.0im	1.4261120897529622e-6	3.9463936e7
6.0	6.000020476673031 + 0.0im	2.0476673030955794e-5	1.29148416e8
7.0	6.99960207042242 + 0.0im	0.00039792957757978087	3.88123136e8
9.0	8.915816367932559 + 0.0im	0.0841836320674414	3.065575424e9
10.0	10.095455630535774 - 0.6449328236240688im	0.6519586830380406	7.143113638035824e9
11.0	10.095455630535774 + 0.6449328236240688im	1.1109180272716561	7.143113638035824e9
12.0	11.793890586174369 - 1.6524771364075785im	1.665281290598479	3.357756113171857e10
13.0	11.793890586174369 + 1.6524771364075785im	2.045820276678428	3.357756113171857e10
14.0	13.992406684487216 - 2.5188244257108443im	2.5188358711909045	1.0612064533081976e11
15.0	13.992406684487216 + 2.5188244257108443im	2.7128805312847097	1.0612064533081976e11
16.0	16.73074487979267 - 2.812624896721978im	2.9060018735375106	3.315103475981763e11
17.0	16.73074487979267 + 2.812624896721978im	2.825483521349608	3.315103475981763e11
18.0	19.5024423688181 - 1.940331978642903im	2.454021446312976	9.539424609817828e12
19.0	19.5024423688181 + 1.940331978642903im	2.004329444309949	9.539424609817828e12
20.0	20.84691021519479 + 0.0im	0.8469102151947894	1.114453504512e13

2.4.4. Wnioski

Na przykładzie powyższych danych wyraźnie widać, że pomimo precyzyjnego wyznaczenia zarówno wielomianu $P(x)$ jak i $P_m(x)$, obliczone pierwiastki są bardzo oddalone od poprawnych wyników. Szczególnie widać do w zmodyfikowanej wersji wielomianu, gdzie mamy do czynienia z pierwiastkami zespolonymi. Efektem tego są błędne wyniki $|P(z_k)|$, $|P_m(z_k)|$ oraz $|p(z_k)|$. Ma na to wpływ fakt, że arytmetyka Float64 ma w języku Julia od 15

do 17 cyfr znaczących w systemie dziesiętnym, natomiast współczynniki wielomianu są rzędu nawet 10^{19}. Stąd wniosek, że niepoprawność otrzymanych wyników jest rezultatem błędnych obliczeń spowodowanych ograniczeniami zastosowanej arytmetyki.

Dodatkowo w drugim eksperymencie widać wyraźnie, że zaburzenie jednego współczynnika o bardzo małą wartość doprowadziło do otrzymania pierwiastków zespolonych równania. To prowadzi do wniosku, że problem szukania pierwiastka wielomianu Wilkinsona to jest bardzo źle uwarunkowany.

2.5. Zadanie 5

2.5.1. Opis problemu

W tym zadaniu dane było równanie rekurencyjne (model logistyczny, model wzrostu populacji)

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

gdzie r jest pewną daną stałą, a p_0 jest wielkością populacji stanowiącą procent maksymalnej wielkości populacji dla danego stanu środowiska. Dla tego równania i danych

$$p_0 = 0.01 \text{ oraz } r = 3$$

należało przeprowadzić trzy eksperymenty polegające na wykonaniu 40 iteracji równania rekurencyjnego (1) w arytmetyce Float32, (2) w arytmetyce Float32 z modyfikacją (tj. obcięciem wyniku po 10 iteracji) oraz (3) w arytmetyce Float64. Następnie należało porównać otrzymane wyniki.

2.5.2. Rozwiązanie

Do rozwiązania tego zadania napisałam funkcję `experiment(type)`, która przeprowadza 40 iteracji równania w arytmetyce `type` oraz funkcję `experimentMod()`, która przeprowadza 40 iteracji wyrażenia z obcięciem po 10 iteracji w arytmetyce Float32.

2.5.3. Wyniki

Poniżej znajdują się wyniki działania programu dla kolejnych n-tych kroków iteracji.

n	Iteracja Float32	Iteracja z modyfikacją	Iteracja Float64
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	0.722	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

2.5.3.Wnioski

W powyższej tabeli wyraźnie widać, że w każdej kolejnej iteracji coraz bardziej pogłębia się błąd spowodowany niedokładnością obliczeń. Szczególnie widać to dla iteracji z modyfikacją, która w dziesiątym kroku zostaje zaburzona poprzez odcięcie cyfr na dalszych miejscach po przecinku. Można tu bardzo dobrze zaobserwować zjawisko sprzężenia zwrotnego, w którym każda kolejna wartość jest ściśle związana z poprzednią, a raz popełniony błąd prowadzi do coraz większych niedokładności wyniku końcowego. Proces ten można nazwać niestabilnym.

Dodatkowo w drugiej części zadania można zaobserwować jaki wpływ na wyniki ma precyzja danej arytmetyki. Porównując wyniki krok po kroku można dostrzec różnice w rezultatach już od drugiego kroku, jednak w szczególności można to zaobserwować od 18-stego kroku, gdzie widać zmianę już na drugim miejscu po przecinku. Dzięki temu możemy zaobserwować ścisłą zależność wyników od warunków początkowych i dokładności obliczeń.

2.6. Zadanie 6

2.6.1. Opis problemu i rozwiązanie

W tym zadaniu dane było równanie rekurencyjne dla $n = 0, 1, ...$

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

gdzie c jest pewną daną stałą, na którym należało przeprowadzić siedem eksperymentów dla różnych zestawów danych. Eksperymenty te miały polegać na wykonaniu 40 iteracji danego równania w arytmetyce Float64. Następnie należało zaobserwować zachowanie ciągów. Rozwiązaniem tego zadania jest program, który wykonuje dla każdego zestawu danych 40 iteracji równania i wyświetla wyniki.

2.6.2. Wyniki

Poniżej przedstawiam wyniki wywołania funkcji dla eksperymentów (1)-(5).

n	$x_0 = 1$ $c = -2$	$x_0 = 2$ $c = -2$	$x_0 = 1.99999999999999$ $c = -2.0$	$x_0 = 1.0$ $c = -1.0$	$x_0 = -1.0$ $c = -1.0$
1	-1.0	2.0	1.99999999999996	0.0	0.0
2	-1.0	2.0	1.9999999999998401	-1.0	-1.0
3	-1.0	2.0	1.9999999999993605	0.0	0.0
4	-1.0	2.0	1.999999999997442	-1.0	-1.0
5	-1.0	2.0	1.9999999999897682	0.0	0.0
6	-1.0	2.0	1.9999999999590727	-1.0	-1.0
7	-1.0	2.0	1.999999999836291	0.0	0.0
8	-1.0	2.0	1.9999999993451638	-1.0	-1.0
9	-1.0	2.0	1.9999999973806553	0.0	0.0
10	-1.0	2.0	1.999999989522621	-1.0	-1.0
11	-1.0	2.0	1.9999999580904841	0.0	0.0
12	-1.0	2.0	1.9999998323619383	-1.0	-1.0
13	-1.0	2.0	1.9999993294477814	0.0	0.0

14	-1.0	2.0	1.9999973177915749	-1.0	-1.0
15	-1.0	2.0	1.9999892711734937	0.0	0.0
16	-1.0	2.0	1.9999570848090826	-1.0	-1.0
17	-1.0	2.0	1.999828341078044	0.0	0.0
18	-1.0	2.0	1.9993133937789613	-1.0	-1.0
19	-1.0	2.0	1.9972540465439481	0.0	0.0
20	-1.0	2.0	1.9890237264361752	-1.0	-1.0
21	-1.0	2.0	1.9562153843260486	0.0	0.0
22	-1.0	2.0	1.82677862987391	-1.0	-1.0
23	-1.0	2.0	1.3371201625639997	0.0	0.0
24	-1.0	2.0	-0.21210967086482313	-1.0	-1.0
25	-1.0	2.0	-1.9550094875256163	0.0	0.0
26	-1.0	2.0	1.822062096315173	-1.0	-1.0
27	-1.0	2.0	1.319910282828443	0.0	0.0
28	-1.0	2.0	-0.2578368452837396	-1.0	-1.0
29	-1.0	2.0	-1.9335201612141288	0.0	0.0
30	-1.0	2.0	1.7385002138215109	-1.0	-1.0
31	-1.0	2.0	1.0223829934574389	0.0	0.0
32	-1.0	2.0	-0.9547330146890065	-1.0	-1.0
33	-1.0	2.0	-1.0884848706628412	0.0	0.0
34	-1.0	2.0	-0.8152006863380978	-1.0	-1.0
35	-1.0	2.0	-1.3354478409938944	0.0	0.0
36	-1.0	2.0	-0.21657906398474625	-1.0	-1.0
37	-1.0	2.0	-1.953093509043491	0.0	0.0
38	-1.0	2.0	1.8145742550678174	-1.0	-1.0
39	-1.0	2.0	1.2926797271549244	0.0	0.0
40	-1.0	2.0	-0.3289791230026702	-1.0	-1.0

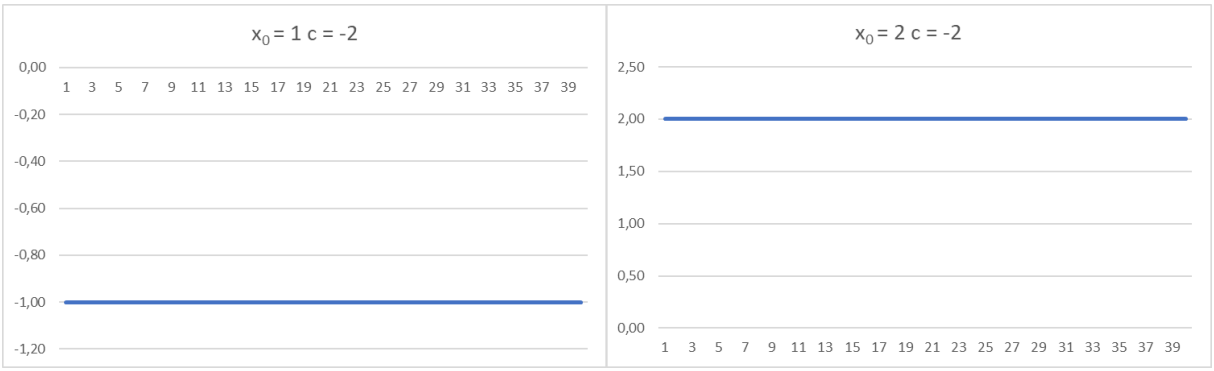
Poniższa tabela zawiera wyniki dla eksperymentów (6) i (7).

n	$x_0 = 0.75\ c = -1.0$	$x_0 = 0.25\ c = -1$
1	-0.4375	-0.9375
2	-0.80859375	-0.12109375
3	-0.3461761474609375	-0.9853363037109375
4	-0.8801620749291033	-0.029112368589267135
5	-0.2253147218564956	-0.9991524699951226
6	-0.9492332761147301	-0.0016943417026455965
7	-0.0989561875164966	-0.9999971292061947
8	-0.9902076729521999	-5.741579369278327e-6
9	-0.01948876442658909	-0.9999999999670343
10	-0.999620188061125	-6.593148249578462e-11
11	-0.0007594796206411569	-1.0
12	-0.9999994231907058	0.0
13	-1.1536182557003727e-6	-1.0
14	-0.9999999999986692	0.0
15	-2.6616486792363503e-12	-1.0
16	-1.0	0.0
17	0.0	-1.0
18	-1.0	0.0
19	0.0	-1.0
20	-1.0	0.0
21	0.0	-1.0
22	-1.0	0.0
23	0.0	-1.0
24	-1.0	0.0
25	0.0	-1.0
26	-1.0	0.0
27	0.0	-1.0
28	-1.0	0.0
29	0.0	-1.0
30	-1.0	0.0
31	0.0	-1.0
32	-1.0	0.0

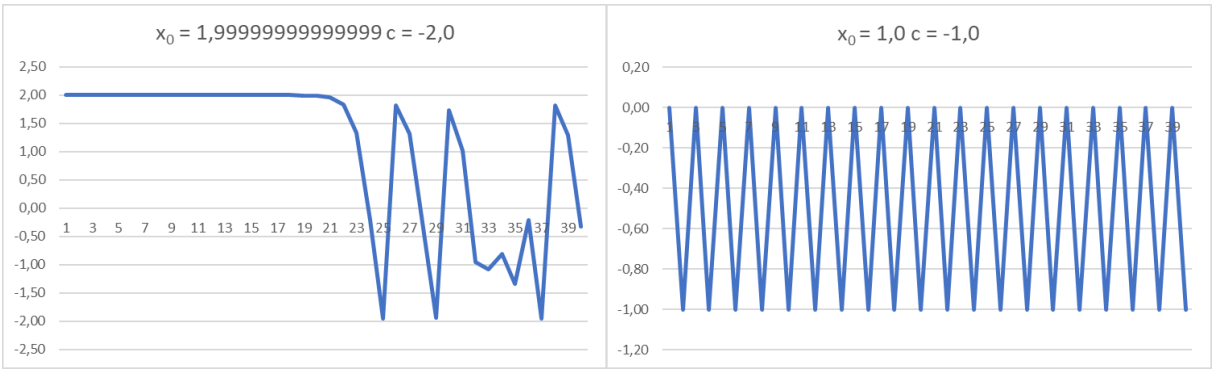
33	0.0	-1.0
34	-1.0	0.0
35	0.0	-1.0
36	-1.0	0.0
37	0.0	-1.0
38	-1.0	0.0
39	0.0	-1.0
40	-1.0	0.0

2.6.4. Wykresy

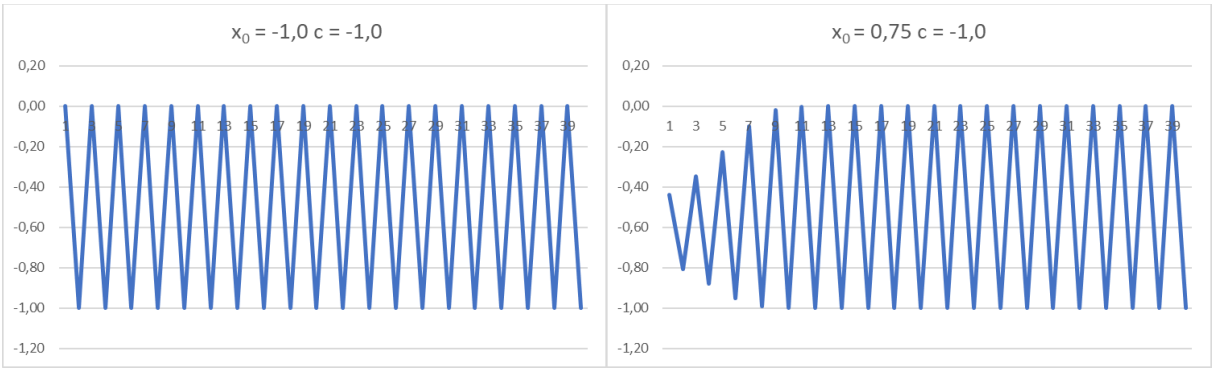
Powyższe dane przetworzyłam w programie *MS Excel* i wygenerowałam wykresy dla każdego zestawu danych początkowych. Wykresy przedstawiam poniżej.



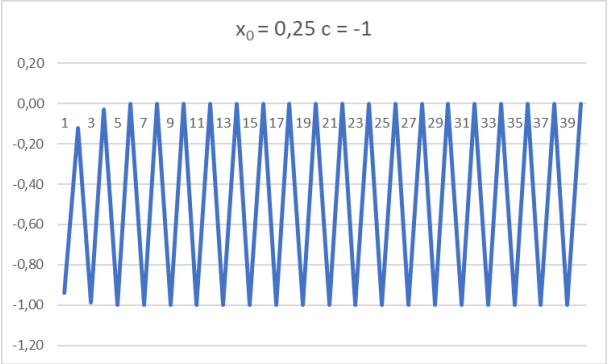
Ryc. 3 Wykresy do eksperymentów (1) i (2)



Ryc. 4 Wykresy do eksperymentów (3) i (4)



Ryc. 5 Wykresy do eksperymentów (5) i (6)



Ryc. 6 Wykres do eksperymentu (7)

2.6.5. Wnioski

Zarówno na wykresach jak i w tabeli powyżej widać, że przy pierwszych dwóch eksperymentach wykresem jest linia prosta, następnie otrzymujemy linię prostą, która przy $n > 20$ zaczyna zmieniać się w losowe oscylacje w przedziale $[-2, 2]$. W eksperymentach (4) i (5) otrzymujemy wartości równe 0 i -1 odpowiednio dla n nieparzystych i n parzystych. Eksperyment (6) pokazuje tę samą sytuację dla $n > 15$, natomiast w eksperymencie (7) wartości osiągają 0 (dla n parzyste) i -1 (dla n nieparzyste) dla $n > 10$.

Podobnie jak w poprzednim zadaniu mamy tu przykład sprzężenia zwrotnego, jednak tym razem w eksperymentach (4)-(7) możemy obserwować proces stabilny (wszędzie $c = -1$). W przeciwieństwie do eksperymentu (3), w którym wartości od pewnego momentu stają się losowe, w eksperymentach (6)-(7) wartości zbiegają odpowiednio do 0 i -1 mimo początkowych zmian w x_0. Widać tu, że dla $c = -2$ proces staje się niestabilny.

3. Podsumowanie

Przeprowadzone eksperymenty miały na celu wyszczególnienie problemów, jakie niosą ze sobą zadania źle uwarunkowane oraz procesy niestabilne. Zjawiska te skutkują znacznym odbieganiem otrzymywanych wyników od prawidłowych danych, co może powodować poważne błędy przy programowaniu wymagającym dokładnych obliczeń.