Obliczenia naukowe - sprawozdanie lista 2

Piotr Kołodziejczyk

Listopad 2019

1 Zadanie 1

1.1 Opis problemu

Zadanie polega na powtórzeniu zadania 5. z pierwszej listy nieznacznie zmieniając w nim dane i zaobserwowaniu różnic w wynikach.

1.2 Rozwiązanie

Używając kodu z zadania 5. listy 1. obliczam iloczyny skalarne dla nieco zmienionych danych - w x_4 usuwam ostatnią 9, a w x_5 ostatnią 7.

1.3 Wyniki

Tabela przedstawia porównanie wyników z pierwotnej wersji zadania z tymi ze zmodyfikowanymi danymi. W arytmetyce Float32 nie ma różnicy w wynikach - usunięte cyfry i tak wykraczały poza

	Float32		Float64	
Sposób	Stare	Nowe	Stare	Nowe
a)	-0.4999443	-0.4999443	1.0251881e-10	-0.004296342739891585
b)	-0.4543457	-0.4543457	-1.5643309e-10	-0.004296342998713953
c)	-0.5	-0.5	0.0	-0.004296342842280865
d)	-0.5	-0.5	0.0	-0.004296342842280865

Tabela 1: Porównanie wyników iloczynu skalarnego wyliczonego na różne sposoby

możliwą precyzję. Wyniki we Float64 różnią się i to znaczniej, niż można by się spodziewać przy zmianie danych początkowych o wielkość rzędu 10^{-10}.

1.4 Wnioski

Możemy wysunąć wniosek, że zadanie jest źle uwarunkowane - na poprzedniej liście uznaliśmy, że różne algorytmy dają niepoprawne wyniki, teraz dodatkowo widać, że naprawdę niewielka zmiana w danych daje *znaczną* różnicę w wyniku. Nowe wyniki we Float64 są bliskie poprawnym, natomiast Float32 nie wystarczył na zapisanie składowych wektorów, stąd błędy.

2 Zadanie 2

2.1 Opis problemu

Zadanie polega na narysowaniu wykresu funkcji $f(x) = e^x \ln(1 + e^{-x})$ w dowolnych dwu programach do wizualizacji i porównaniu ich z obliczoną granicą funkcji f.

2.2 Rozwiązanie

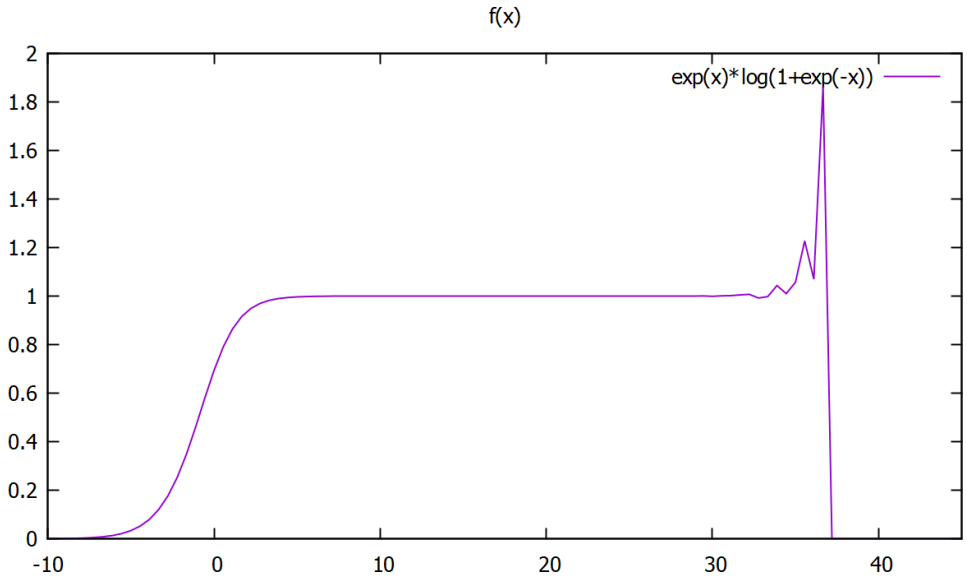
Granica funkcji f równa się:

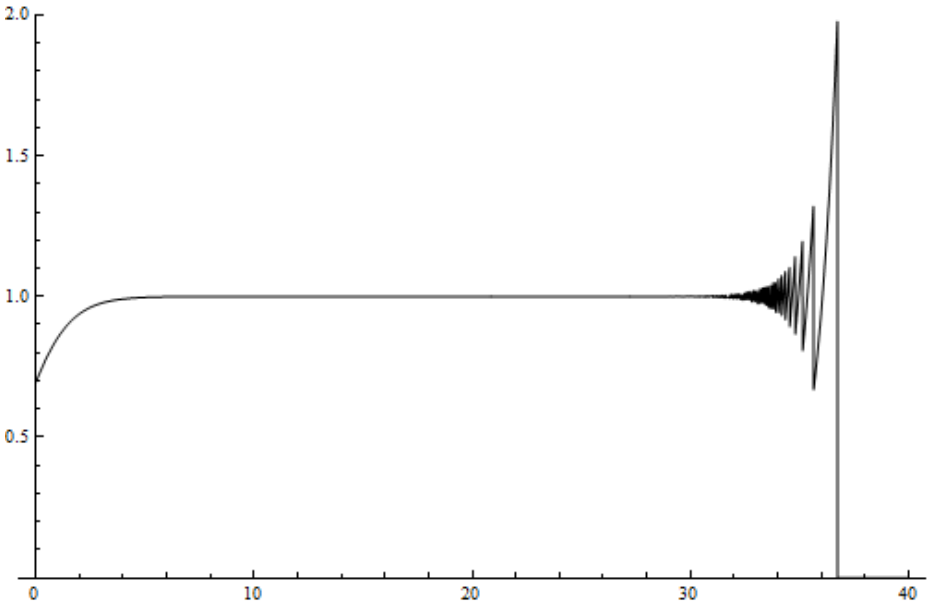
$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} \frac{\ln(1+e^{-x})}{e^{-x}} =^H \lim_{x\to\infty} \frac{e^{-x}}{(1+e^{-x})e^{-x}} = \lim_{x\to\infty} \frac{1}{1+e^{-x}} = 1$$

Wykresy zostały wygenerowane przy pomocy programów gnuplot oraz mathematica.

2.3 Wyniki

Wykresy funkcji dla argumentów rzędu 30-40 przestają być prawdziwe - oddalają się coraz bardziej od zera, by w końcu przyjąć wartość 0 i pozostać w niej, pomimo że granica funkcji jest inna.





2.4 Wnioski

Dla x rzędu 40, wartość e^{-x} spada poniżej wartości epsilonu maszynowego, więc logarytm zbiega do 0, wobec czego cała funkcja ma wartość 0. Jak widać, problemu tego nie omijają nawet numerycznie zaawansowane narzędzia - mając więc wątpliwości, czy wygenerowany wykres jest poprawny, warto policzyć granicę funkcji.

3 Zadanie 3

3.1 Opis problemu

Celem zadania jest numeryczne zmierzenie się z rozwiązywaniem układów równań liniowych. Równania postaci $\boldsymbol{A}x = b$ rozpatrujemy dla macierzy Hilberta stopnia n, oraz losowej macierzy stopnia n z zadanym wskaźnikiem uwarunkowania.

3.2 Rozwiązanie

Do wygenerowania macierzy używam funkcji `hilb` oraz `matcond`. x jest wektorem jednostkowym, a b to wynik Ax (predefiniowane dane mają na celu możliwość sprawdzenia dokładności obliczeń). Wartość x obliczam na dwa sposoby - metodą eliminacji Gaussa (`x=A\b`) oraz przez inwersję (`x=inv(A)*b`). Powtarzam dla różnych rozmiarów macierzy oraz różnych wskaźników uwarunkowania macierzy losowej.

3.3 Wyniki

Tabele 2. i 3. przedstawiają wyniki dla wygenerowanych macierzy Hilberta i losowych różnych rozmiarów. W przypadku macierzy Hilberta błąd rośnie wraz z rozmiarem, natomiast w macierzy losowej większą zależność widać pomiędzy błędem a wskaźnikiem uwarunkowania.

n	rank(H)	cond(H)	Błąd Gauss	Błąd Inwersja
1	1	1.0	0.0	0.0
2	2	19.28147006790397	5.661048867003676e-16	1.4043333874306803e-15
3	3	524.0567775860644	8.022593772267726e-15	0.0
4	4	15513.73873892924	4.137409622430382e-14	0.0
5	5	476607.25024259434	1.6828426299227195e-12	3.3544360584359632e-12
6	6	1.4951058642254665e7	2.618913302311624e-10	2.0163759404347654e-10
7	7	4.75367356583129e8	1.2606867224171548e-8	4.713280397232037e-9
8	8	1.5257575538060041e10	6.124089555723088e-8	3.07748390309622e-7
9	9	4.931537564468762e11	3.8751634185032475e-6	4.541268303176643e-6
10	10	1.6024416992541715e13	8.67039023709691e-5	0.0002501493411824886
11	10	5.222677939280335e14	0.00015827808158590435	0.007618304284315809
12	11	1.7514731907091464e16	0.13396208372085344	0.258994120804705
13	11	3.344143497338461e18	0.11039701117868264	5.331275639426837
14	11	6.200786263161444e17	1.4554087127659643	8.71499275104814
15	12	3.674392953467974e17	4.696668350857427	7.344641453111494
16	12	7.865467778431645e17	54.15518954564602	29.84884207073541
17	12	1.263684342666052e18	13.707236683836307	10.516942378369349
18	12	2.2446309929189128e18	9.134134521198485	7.575475905055309
19	13	6.471953976541591e18	9.720589712655698	12.233761393757726
20	13	1.3553657908688225e18	7.549915039472976	22.062697257870493

Tabela 2: Wyniki dla macierzy Hilberta, kolejno rozmiar, rząd macierzy, wskaźnik uwarunkowania oraz błędy względne obliczonego x metodą eliminacji Gaussa oraz metodą inwersji.

3.4 Wnioski

Macierz Hilberta jest przykładem macierzy źle uwarunkowanej, dla stosunkowo niewielkiego rozmiaru n=10, cond(H) jest rzędu 10^{13}. Oba algorytmy dla dużych rozmiarów dają duży błąd (złe uwarunkowanie jest cechą danych, nie algorytmu). Macierze losowe zachowują się dużo lepiej od macierzy Hilberta.

4 Zadanie 4

4.1 Opis problemu

Celem zadania jest zapoznanie się z wielomianem Wilkinsona (tzw. złośliwy wielomian) i przeprowadzeniem na nim eksperymentów numerycznych. Ze względu na wielkość współczynników spodziewamy się anomalii w wynikach.

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

4.2 Rozwiązanie

W skrypcie 4.jl wielomian Wilkinsona przedstawiony jest na dwa sposoby: w postaci normalnej $P(x)$ (po wymnożeniu) oraz jako iloczyn $p(x) = \prod_{i=1}^{20}(x - i)$ (matematycznie tożsame formy). Używając

n	rank(R)	cond(R)	Błąd Gauss	Błąd Inwersja
5	5	1	1.85775845048325e-16	2.0471501066083611e-16
5	5	10	5.301242283512285e-16	5.393444713726544e-16
5	5	10^3	6.577919173368772e-15	8.556067554929068e-15
5	5	10^7	2.4300725328443226e-10	2.471684674887078e-10
5	5	10^{12}	4.087478849305578e-5	3.323387377109452e-5
5	4	10^{17}	0.13874902269789224	0.16770509831248423
10	10	1	3.813741331030777e-16	4.168883761650163e-16
10	10	10	3.2177320244274193e-16	3.3857251850959236e-16
10	10	10^3	2.4660025306105033e-14	2.6281678192450302e-14
10	10	10^7	9.03459651022284e-11	1.441966491345168e-10
10	10	10^{12}	3.223882698565345e-5	4.3510966102932336e-5
10	9	10^{17}	0.13794878797333646	0.2225122889864737
20	20	1	3.8298671591131183e-16	3.1694874333121784e-16
20	20	10	6.161487731364982e-16	4.468561475845091e-16
20	20	10^3	2.8287820360246276e-15	3.689383200557469e-15
20	20	10^7	1.1001321368600536e-10	9.851133410157492e-11
20	20	10^{12}	1.055326849082849e-5	1.2005979408079833e-5
20	19	10^{17}	0.0909272240832228	0.12547213083085174

Tabela 3: Wyniki dla losowej macierzy z zadanym wskaźnikiem uwarunkowania, pokazane błędy względne obliczonego x metodą eliminacji Gaussa oraz metodą inwersji.

pakietu `Polynomials`, funkcją `roots` obliczam zera wielomianu, sprawdzam je, podstawiając wartości do obu form wielomianu oraz porównuję z faktycznymi wartości miejsc zerowych. W drugiej części powtarzam obliczenia, zmieniając współczynnik przy x^{19} o wartość 2^{-23}.

4.3 Wyniki

W tabeli przedstawione są wyliczone pierwiastki wielomianu, wartość wielomianu w tych punktach dla obu postaci wielomianu oraz różnica między pierwiastkiem faktycznym a obliczonym. Już na pierwszy rzut oka widać, że wyniki są niepoprawne - 3 ostatnie kolumny powinny być co najmniej bliskie zeru, a im wyższa potęga x, tym wartości w nich są większe. Ponadto, mała zmiana (2^{-30}) jednego ze współczynników sprawia, że pojawiają się pierwiastki zespolone.

x	$\tilde{x}$	$P(\tilde{x})$	$p(\tilde{x})$	$\Delta(x, \tilde{x})$
1	0.9999999999996989	36352.0	38400.0	3.0109248427834245e-13
2	2.0000000000283182	181760.0	198144.0	2.8318236644508943e-11
3	2.9999999995920965	209408.0	301568.0	4.0790348876384996e-10
4	3.9999999837375317	3.106816e6	2.844672e6	1.626246826091915e-8
5	5.000000665769791	2.4114688e7	2.3346688e7	6.657697912970661e-7
6	5.999989245824773	1.20152064e8	1.1882496e8	1.0754175226779239e-5
7	7.000102002793008	4.80398336e8	4.78290944e8	0.00010200279300764947
8	7.999355829607762	1.682691072e9	1.67849728e9	0.0006441703922384079
9	9.002915294362053	4.465326592e9	4.457859584e9	0.002915294362052734
10	9.990413042481725	1.2707126784e10	1.2696907264e10	0.009586957518274986
11	11.025022932909318	3.5759895552e10	3.5743469056e10	0.025022932909317674
12	11.953283253846857	7.216771584e10	7.2146650624e10	0.04671674615314281
13	13.07431403244734	2.15723629056e11	2.15696330752e11	0.07431403244734014
14	13.914755591802127	3.65383250944e11	3.653447936e11	0.08524440819787316
15	15.075493799699476	6.13987753472e11	6.13938415616e11	0.07549379969947623
16	15.946286716607972	1.555027751936e12	1.554961097216e12	0.05371328339202819
17	17.025427146237412	3.777623778304e12	3.777532946944e12	0.025427146237412046
18	17.99092135271648	7.199554861056e12	7.1994474752e12	0.009078647283519814
19	19.00190981829944	1.0278376162816e13	1.0278235656704e13	0.0019098182994383706
20	19.999809291236637	2.7462952745472e13	2.7462788907008e13	0.00019070876336257925

Tabela 4: Porównanie wartości pierwiastków wielomianu oraz wartości wielomianu w tych punktach, gdzie P(x) - postać normalna, p(x) - postać iloczynowa

Tabela 5: Porównanie dla wielomianu ze zmienionym współczynnikiem przy x^{19}

Pierwiastki $p'(x)$	$P(\tilde{x})$	$p(\tilde{x})$	$\Delta(x, \tilde{x})$
0.9999999999998357 + 0.0im	20992.0	22016.0	1.6431300764452317e-13
2.0000000000550373 + 0.0im	349184.0	365568.0	5.503730804434781e-11
2.99999999660342 + 0.0im	2.221568e6	2.295296e6	3.3965799062229962e-9
4.000000089724362 + 0.0im	1.046784e7	1.0729984e7	8.972436216225788e-8
4.99999857388791 + 0.0im	3.9463936e7	4.3303936e7	1.4261120897529622e-6
6.000020476673031 + 0.0im	1.29148416e8	2.06120448e8	2.0476673030955794e-5
6.99960207042242 + 0.0im	3.88123136e8	1.757670912e9	0.00039792957757978087
8.007772029099446 + 0.0im	1.072547328e9	1.8525486592e10	0.007772029099445632
8.915816367932559 + 0.0im	3.065575424e9	1.37174317056e11	0.0841836320674414
10.095455630535774 - 0.6449328236240688im	7.143113638035824e9	1.4912633816754019e12	0.6519586830380406
10.095455630535774 + 0.6449328236240688im	7.143113638035824e9	1.4912633816754019e12	1.1109180272716561
11.793890586174369 - 1.6524771364075785im	3.357756113171857e10	3.2960214141301664e13	1.665281290598479
11.793890586174369 + 1.6524771364075785im	3.357756113171857e10	3.2960214141301664e13	2.045820276678428
13.992406684487216 - 2.5188244257108443im	1.0612064533081976e11	9.545941595183662e14	2.5188358711909045
13.992406684487216 + 2.5188244257108443im	1.0612064533081976e11	9.545941595183662e14	2.7128805312847097
16.73074487979267 - 2.812624896721978im	3.315103475981763e11	2.7420894016764064e16	2.9060018735375106
16.73074487979267 + 2.812624896721978im	3.315103475981763e11	2.7420894016764064e16	2.825483521349608
19.5024423688181 - 1.940331978642903im	9.539424609817828e12	4.2525024879934694e17	2.454021446312976
19.5024423688181 + 1.940331978642903im	9.539424609817828e12	4.2525024879934694e17	2.004329444309949
20.84691021519479 + 0.0im	1.114453504512e13	1.3743733197249713e18	0.8469102151947894

4.4 Wnioski

Zadanie jest źle uwarunkowane ze względu na zaburzenia współczynników - mała zmiana danych wprowadza duży błąd. Nieprawidłowości w wartościach obliczonych zer wynikają z faktu, że współczynniki przy niektórych potęgach mają więcej cyfr znaczących, niż jest w stanie zapewnić arytmetyka Float64.

5 Zadanie 5

5.1 Opis problemu

Zadanie polegało na przeprowadzeniu eksperymentów numerycznych dla rekurencyjnego równania modelu wzrostu populacji danego wzorem

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

5.2 Rozwiązanie

Skrypt 5.jl iteracyjnie wylicza kolejne wartości p_n dla danych wartości początkowej $p_0 = 0.01$ (procentowa wielkość stanu populacji względem maksymalnej wartości) oraz stałej $r = 3$.
W pierwszej części wykonuję 40 kolejnych iteracji oraz tak samo 40 iteracji, ale z obcięciem do trzech cyfr wyniku w 10. iteracji. Te działania wykonuję w arytmetyce Float32. W drugiej części porównuję kolejnych 40 iteracji dla tych samych danych w arytmetyce Float32 i Float64.

5.3 Wyniki

Tabele na stronie 10. prezentują wyniki przeprowadzonych eksperymentów. Widać, że zarówno zmiana wyniku w 10. iteracji jak i zwiększenie dokładności skutkują całkowitymi zmianami kolejnych wyników.

5.4 Wnioski

Równania logiczne są bardzo wrażliwe na dokładność danych - niewielka zmiana warunków początkowych może dawać zupełnie różne wyniki w kolejnych iteracjach. Z numerycznego punktu widzenia występuje tu również numeryczna niestabilność - niedokładność przedstawienia liczby powoduje kumulację błędu w kolejnych wynikach.

6 Zadanie 6

6.1 Opis problemu

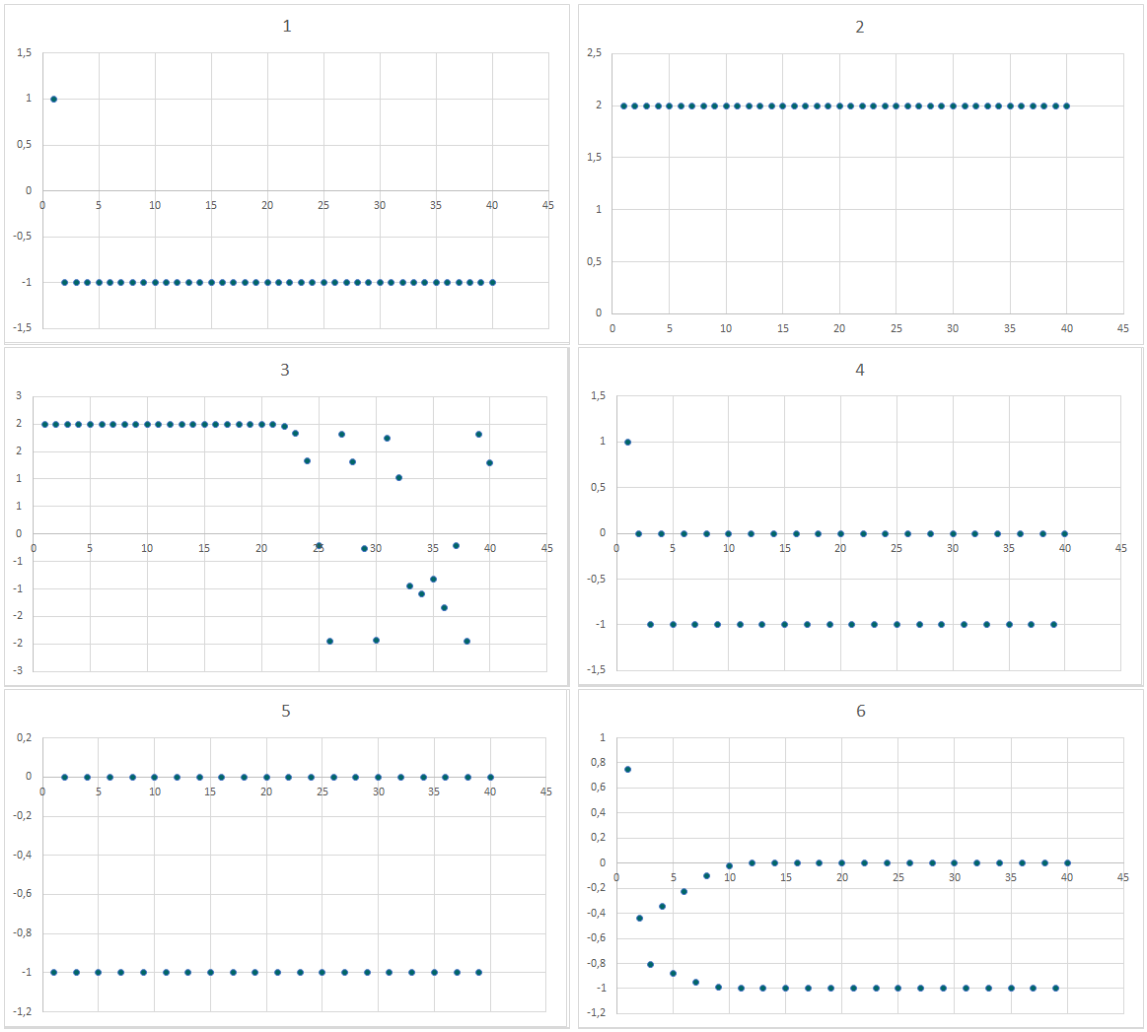
Zadanie polega na analizie rekurencyjnego równania $x_{n+1} = x_n^2 + c$ dla różnych danych wejściowych x_0 oraz c.

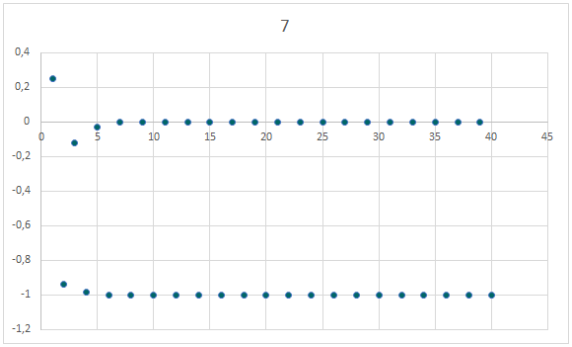
6.2 Rozwiązanie

Za pomocą skryptu 6.jl wyliczam kolejne iteracje równania dla podanych wartości początkowych, które następnie przedstawiam graficznie.

6.3 Wyniki

Wykresy przedstawiają zachowanie równania w kolejnych iteracjach. Dla 1., 2., 4., i 5. zestawu danych nie dzieje się nic niespodziewanego - obracamy się w wartościach całkowitych. Przykład 3. różni się od 2. tylko o 10^{-14} , a widać że po 20. iteracji wartości całkowicie rozbiegają się. W 6. i 7. przypadku po kilkunastu iteracjach wartości wpadają w okresowy ciąg dwu wartości.





6.4 Wnioski

Przykład 3. pokazuje że mała zmiana danych znacząco wpływa na wyniki, jeśli jest ona kumulowana. Tak samo w przypadkach 6. i 7. po kilku iteracjach wartość przestaje być dokładnie reprezentowana w arytmetyce IEEE754, wobec czego z każdą kolejną iteracją błąd narasta (występuje niestabilność numeryczna).

	Kolejne iteracje	Zmiana 10. wartości
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	1,324148
12	0,037716985	0,036488056
13	0,14660022	0,14195809
14	0,521926	0,5073761
15	1,2704837	1,2572129
16	0,2395482	0,28709882
17	0,7860428	0,9011181
18	1,2905813	1,1684309
19	0,16552472	0,5780312
20	0,5799036	1,3097646
21	1,3107498	0,09260833
22	0,088804245	0,34470442
23	0,3315584	1,0223544
24	0,9964407	0,9537921
25	1,0070806	1,0860103
26	0,9856885	0,805786
27	1,0280086	1,2752707
28	0,9416294	0,22213674
29	1,1065198	0,7405127
30	0,7529209	1,3169736
31	1,3110139	0,06463623
32	0,0877831	0,24601139
33	0,3280148	0,80248076
34	0,9892781	1,277997
35	1,021099	0,21215892
36	0,95646656	0,71360147
37	1,0813814	1,3267246
38	0,81736827	0,026303649
39	1,2652004	0,10313895
40	0,25860548	0,3806429

	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

Tabela 5: Kolejne iteracje równania logicznego z zadania 5.