Sprawozdanie z listy 2

Michał Gancarczyk 244923

Streszczenie

Celem doświadczeń przeprowadzonych w ramach listy zadań jest przeanalizowanie problemów związanych z właściwościami specyficznymi dla arytmetyki float. Wiele z wykonanych zadań obrazuje niebezpieczeństwa wynikające ze złego uwarunkowania algorytmów.

Sprawozdanie poświęca jedną sekcję każdemu z sześciu zadań. W skład każdej sekcji wchodzą kolejno: krótki opis przedstawionego w zadaniu problemu, wyniki uzyskane przy użyciu tego algorytmu, opcjonalne wykresy obrazujące przedstawione wcześniej dane, oraz wypływające z doświadczenia wnioski.

Do rozwiązania zadań użyty został język Julia, wersja 1.2.0.

1 Zadanie 1

Celem zadania jest powtórne wykonanie doświadczenia z piątego podpunku poprzedniej listy po zmodyfikowaniu niektórych wartości podanego w nim wektora x.

$$\tilde{x} = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$$
$$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$$

Wyniki dla $\tilde{x}$		
Algorytm	Float32	Float64
A	-0.4999443	-0.004296342739891585
B	-0.4543457	-0.004296342998713953
C	-0.5	-0.004296342842280865
D	-0.5	-0.004296342842280865

Wyniki dla x		
Algorytm	Float32	Float64
A	-0.4999443	$1.0251881368296672 \cdot 10^{-10}$
B	-0.4543457	$-1.5643308870494366 \cdot 10^{-10}$
C	-0.5	0.0
D	-0.5	0.0

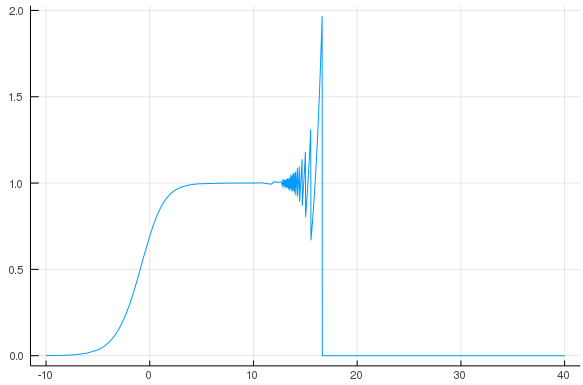
Obserwacja wyników przed i po wprowadzeniu opisanych wyżej zmian pozwala zauważyć, że precyzja arytmetyki **Float32** nadal nie pozwala na uzyskanie satysfakcjonująco bliskich poprawnym wyników. Zgoła odwrotny jest rezultat dla **Float64**, gdzie wyniki obliczeń po zmianie parametrów diametralnie różnią się od poprzednich. Analizując wyniki stwierdzić można, że wartości owe są dla zadanych obliczeń poprawne. Istotną obserwacją w przypadku arytmetyki **Float64** jest niewielkie odchylenie rezultatu od poprawnego wyniku niezależnie od wykorzystanego algorytmu. Doświadczenie pokazuje jak istotne jest dobranie odpowiedniej precyzji do wykonywanych obliczeń - choć w przypadku **Float32** możnaby mówić o złym uwarunkowaniu zadania, wykonanie tych samych obliczeń we **Float64** nie niesie już za sobą podobnych problemów.

2 Zadanie 2

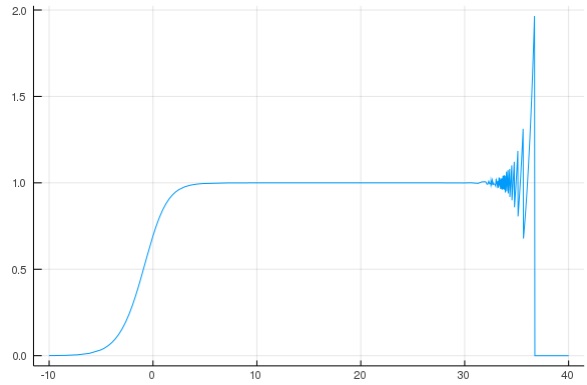
Doświadczenie opiera się na porównaniu generowanych przez różne programy wykresów funkcji $f(x) = e^x ln(1 + e^{-x})$. Poprawność wykresów zostanie dodatkowo zweryfikowana w oparciu o ręcznie obliczoną granicę funkcji, zaprezentowaną poniżej.

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^x \ln(1+e^{-x}) = \lim_{x\to\infty} \frac{\ln(1+e^{-x})}{e^{-x}} = \begin{bmatrix} 0 \\ \overline{0} \end{bmatrix} \overset{H}{=} \lim_{x\to\infty} \frac{(\ln(1+e^{-x}))'}{(e^{-x})'} =$$
$$= \lim_{x\to\infty} \frac{\frac{1}{1+e^{-x}} \times -e^{-x}}{-e^{-x}} = \lim_{x\to\infty} \frac{1}{1+e^{-x}} = 1.$$

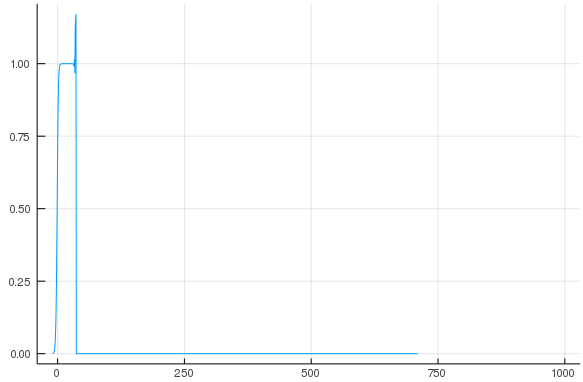
Poniższe wykresy prezentują wyniki, jakie można napotkać obliczając funkcję przy użyciu programów komputerowych.



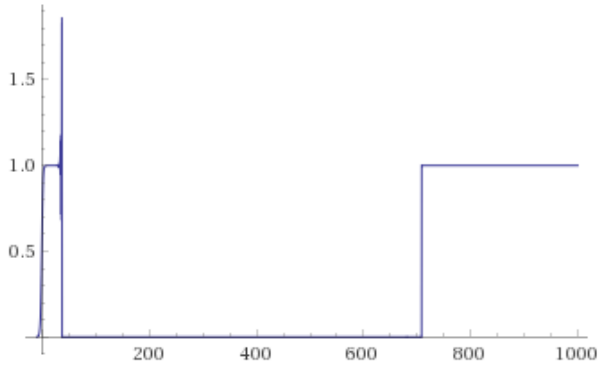
Rys. 1. Julia, Float32



Rys. 2. Julia, Float64



Rys. 3. Julia, Float64, większy zakres



Rys. 4. Wolfram Alpha

Uzyskana granica wskazuje, że dla całego przedziału funkcja powinna przyjmować wartość 1. Obserwacja powyższych wykresów pokazuje jednak, że w pewnym, zależnym od przecyzji obliczeń, momencie wartości funkcji odbiegają od 1 i zaczynają oscylować.

Powodem wystąpienia tej anomalii jest wykonanie operacji dodawania 1 i e podniesionego do coraz mniejszych liczb ujemnych. W pewnym momencie e^{-x} stanie się na tyle małe, że po dodaniu obu liczb wynik w dalszym ciągu będzie równy 1. W tej sytuacji wartość wyrażenia $ln(1+e^{-x})$ wyniesie 0, co tłumaczy pojawienie się zera na wykresach.

Dodatkowo gdy x przyjmie wystarczająco dużą wartość, wyrażenie e^x równe będzie ∞. Zgodnie z arytmetyką float $0 \cdot \infty = NaN$, które, jak przedstawiono na rysunku 3, na wykresie nie rysowane jest wogóle, w czego skutku zaobserwować można puste miejsca. Prawidłowość ta nie jest jednak zauważalna w wykresie wygenerowanym przez program *Wolfram Alpha*, który po napotkaniu wartości NaN zastępuje wyniki fukcji przez poprawnie wyliczoną granicę.

3 Zadanie 3

Zadanie polega na rozwiązaniu układ równań liniowych $Ax = b$, gdzie dla macierzy A i wektora prawych stron b przy użyciu algorytmów eliminacji Gaussa ($x = A\backslash b$) oraz $x = A - 1b(x = inv(A) \cdot b)$.

Rozwiązania zaprezentowane są w poniższych tabelach.

Macierz Hilberta				
n	rank	cond	Błąd względny dla EG	Błąd względny dla INV
1	1	1.0	0.0	0.0
2	2	19.28147006790397	$4.227603326225575 \cdot 10^{-16}$	$1.4043333874306803 \cdot 10^{-15}$
3	3	524.0567775860644	$6.312995352117186 \cdot 10^{-16}$	0.0
4	4	15513.73873892924	$2.1819484005185015 \cdot 10^{-15}$	$4.547473508864641 \cdot 10^{-13}$
5	5	476607.25024259434	$8.99737540851766 \cdot 10^{-12}$	$1.4911297558868156 \cdot 10^{-11}$
6	6	$1.4951058642254665 \cdot 10^{7}$	$1.8866554820446764 \cdot 10^{-10}$	$3.5689314550268525 \cdot 10^{-10}$
7	7	$4.75367356583129 \cdot 10^{8}$	$1.8614175017153493 \cdot 10^{-8}$	$1.231617281152939 \cdot 10^{-8}$
8	8	$1.5257575538060041 \cdot 10^{10}$	$1.9217530132542664 \cdot 10^{-7}$	$1.3503856270597747 \cdot 10^{-7}$
9	9	$4.931537564468762 \cdot 10^{11}$	$5.09208671414057 \cdot 10^{-6}$	$9.542631496737485 \cdot 10^{-6}$
10	10	$1.6024416992541715 \cdot 10^{13}$	$5.508778842434553 \cdot 10^{-5}$	0.0002289249459044258
11	10	$5.222677939280335 \cdot 10^{14}$	0.005207147400142043	0.008849600746399502
12	11	$1.7514731907091464 \cdot 10^{16}$	0.1407902677571603	0.4039460424541017
13	11	$3.344143497338461 \cdot 10^{18}$	355.6363996139929	212.78257870436312
14	11	$6.200786263161444 \cdot 10^{17}$	3.5720053119125916	1.7744875020766255
15	12	$3.674392953467974 \cdot 10^{17}$	9.583022643862336	5.293638550842115
16	12	$7.865467778431645 \cdot 10^{17}$	11.968066192267113	10.846229140698135
17	12	$1.263684342666052 \cdot 10^{18}$	5.455797564762254	8.982730231514317
18	12	$2.2446309929189128 \cdot 10^{18}$	26.309523680963647	20.038300611698062
19	13	$6.471953976541591 \cdot 10^{18}$	22.59171398621913	15.359549749376397
20	13	$1.3553657908688225 \cdot 10^{18}$	5.658540758579833	7.992554067235351

Macierz Losowa				
n	rank	cond	Błąd względny dla EG	Błąd względny dla INV
5	5	1.0000000000000004	$1.4043333874306804 \cdot 10^{-16}$	$1.4043333874306804 \cdot 10^{-16}$
5	5	9.999999999999998	$2.432376777795247 \cdot 10^{-16}$	$4.2130001622920406 \cdot 10^{-16}$
5	5	999.9999999998879	$1.532365195377229 \cdot 10^{-14}$	$1.9394042374125575 \cdot 10^{-14}$
5	5	$1.0000000007063102 \cdot 10^{7}$	$4.058556027137932 \cdot 10^{-11}$	$1.041250292910165 \cdot 10^{-10}$
5	5	$9.999732749016914 \cdot 10^{11}$	$4.628557486853792 \cdot 10^{-5}$	$5.097476201970202 \cdot 10^{-5}$
5	4	$4.820871440827623 \cdot 10^{15}$	0.5426564720655163	0.574864114378346
10	10	1.0000000000000013	$2.808666774861361 \cdot 10^{-16}$	$2.3022074639253675 \cdot 10^{-16}$
10	10	10.000000000000002	$3.080743865682491 \cdot 10^{-16}$	$2.5559253454202263 \cdot 10^{-16}$
10	10	999.9999999999893	$8.494983760466248 \cdot 10^{-15}$	$1.0043008996726448 \cdot 10^{-14}$
10	10	$9.999999994332584 \cdot 10^{6}$	$2.958231981690243 \cdot 10^{-10}$	$3.2382507449251424 \cdot 10^{-10}$
10	10	$1.0001245180111727 \cdot 10^{12}$	$4.505149590293919 \cdot 10^{-5}$	$3.60521688432706 \cdot 10^{-5}$
10	9	$1.3891260498988398 \cdot 10^{16}$	0.03572150085990611	0.0506307867063114
20	20	1.000000000000001	$4.996003610813204 \cdot 10^{-16}$	$7.03043854199395 \cdot 10^{-16}$
20	20	9.999999999999995	$4.198346204284728 \cdot 10^{-16}$	$4.3568297570458958 \cdot 10^{-16}$
20	20	999.9999999999586	$1.476229285231916 \cdot 10^{-14}$	$1.8281163050989988 \cdot 10^{-14}$
20	20	$1.0000000001047328 \cdot 10^{7}$	$1.815554896241027 \cdot 10^{-10}$	$1.9649969274654167 \cdot 10^{-10}$
20	20	$9.999726395629987 \cdot 10^{11}$	$1.712068253198343 \cdot 10^{-5}$	$2.156522077484519 \cdot 10^{-5}$
20	19	$1.2187785923051648 \cdot 10^{16}$	0.0741944675648783	0.12660107436846654

Jak można zaobserwować na pierwszej tabeli decydujący wpływ na błąd obliczeń ma wskaźnik uwarunkowania macierzy. Większy wskaźnik skutkuje w tej sytuacji większym błędem w rozwiązaniu układu równań.

Istotną własnością Macierzy Hilberta jest jej złe uwarunkowanie - im większy jest rozmiar macierzy, tym większe są błędy zaobserwowane w wynikach. Kolejną ciekawą własnością macierzy jest to, że w jej przypadku precyzyjniejszym algorytmem jest eliminacja Gaussa. Stanowi to odstępstwo od sytuacji "standardowej" (która w zadaniu rozpatrywana jest poprzez macierz losową), gdzie eliminacja Gaussa przynosi mniej precyzyjne wyniki.

4 Zadanie 4

Celem zadania jest obliczenie dwudziestu pierwiastków "złośliwego wielomianu" Wilkinsona, a następnie sprawdzeniu otrzymanych wyników, przez obliczenie wartości funkcji dla otrzymanych miejsc zerowych.

Wielomian				
k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999996989	36352.0	38400.0	$3.0109248427834245 \cdot 10^{-13}$
2	2.0000000000283182	181760.0	198144.0	$2.8318236644508943 \cdot 10^{-11}$
3	2.9999999995920965	209408.0	301568.0	$4.0790348876384996 \cdot 10^{-10}$
4	3.9999999837375317	$3.106816 \cdot 10^{6}$	$2.844672 \cdot 10^{6}$	$1.626246826091915 \cdot 10^{-8}$
5	5.000000665769791	$2.4114688 \cdot 10^{7}$	$2.3346688 \cdot 10^{7}$	$6.657697912970661 \cdot 10^{-7}$
6	5.999989245824773	$1.20152064 \cdot 10^{8}$	$1.1882496 \cdot 10^{8}$	$1.0754175226779239 \cdot 10^{-5}$
7	7.000102002793008	$4.80398336 \cdot 10^{8}$	$4.78290944 \cdot 10^{8}$	0.00010200279300764947
8	7.999355829607762	$1.682691072 \cdot 10^{9}$	$1.67849728 \cdot 10^{9}$	0.0006441703922384079
9	9.002915294362053	$4.465326592 \cdot 10^{9}$	$4.457859584 \cdot 10^{9}$	0.002915294362052734
10	9.990413042481725	$1.2707126784 \cdot 10^{10}$	$1.2696907264 \cdot 10^{10}$	0.009586957518274986
11	11.025022932909318	$3.5759895552 \cdot 10^{10}$	$3.5743469056 \cdot 10^{10}$	0.025022932909317674
12	11.953283253846857	$7.216771584 \cdot 10^{10}$	$7.2146650624 \cdot 10^{10}$	0.04671674615314281
13	13.07431403244734	$2.15723629056 \cdot 10^{11}$	$2.15696330752 \cdot 10^{11}$	0.07431403244734014
14	13.914755591802127	$3.65383250944 \cdot 10^{11}$	$3.653447936 \cdot 10^{11}$	0.08524440819787316
15	15.075493799699476	$6.13987753472 \cdot 10^{11}$	$6.13938415616 \cdot 10^{11}$	0.07549379969947623
16	15.946286716607972	$1.555027751936 \cdot 10^{12}$	$1.554961097216 \cdot 10^{12}$	0.05371328339202819
17	17.025427146237412	$3.777623778304 \cdot 10^{12}$	$3.777532946944 \cdot 10^{12}$	0.025427146237412046
18	17.99092135271648	$7.199554861056 \cdot 10^{12}$	$7.1994474752 \cdot 10^{12}$	0.009078647283519814
19	19.00190981829944	$1.0278376162816 \cdot 10^{13}$	$1.0278235656704 \cdot 10^{13}$	0.0019098182994383706
20	19.999809291236637	$2.7462952745472 \cdot 10^{13}$	$2.7462788907008 \cdot 10^{13}$	0.0001907087633625792 5

Wielomian ze zmienioną wartością przy x^{19}				
k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999998357 + 0.0im	20992.0	22016.0	$1.6431300764452317 \cdot 10^{-13}$
2	2.0000000000550373 + 0.0im	349184.0	365568.0	$5.503730804434781 \cdot 10^{-11}$
3	2.99999999660342 + 0.0im	$2.221568 \cdot 10^{6}$	$2.295296 \cdot 10^{6}$	$3.3965799062229962 \cdot 10^{-9}$
4	4.000000089724362 + 0.0im	$1.046784 \cdot 10^{7}$	$1.0729984 \cdot 10^{7}$	$8.972436216225788 \cdot 10^{-8}$
5	4.99999857388791 + 0.0im	$3.9463936 \cdot 10^{7}$	$4.3303936 \cdot 10^{7}$	$1.4261120897529622 \cdot 10^{-6}$
6	6.000020476673031 + 0.0im	$1.29148416 \cdot 10^{8}$	$2.06120448 \cdot 10^{8}$	$2.0476673030955794 \cdot 10^{-5}$
7	6.99960207042242 + 0.0im	$3.88123136 \cdot 10^{8}$	$1.757670912 \cdot 10^{9}$	0.00039792957757978087
8	8.007772029099446 + 0.0im	$1.072547328 \cdot 10^{9}$	$1.8525486592 \cdot 10^{10}$	0.007772029099445632
9	8.915816367932559 + 0.0im	$3.065575424 \cdot 10^{9}$	$1.37174317056 \cdot 10^{11}$	0.0841836320674414
10	10.095455630535774 - 0.6449328236240688im	$7.143113638035824 \cdot 10^{9}$	$1.4912633816754019 \cdot 10^{12}$	0.6519586830380406
11	10.095455630535774 + 0.6449328236240688im	$7.143113638035824 \cdot 10^{9}$	$1.4912633816754019 \cdot 10^{12}$	1.1109180272716561
12	11.793890586174369 - 1.6524771364075785im	$3.357756113171857 \cdot 10^{10}$	$3.2960214141301664 \cdot 10^{13}$	1.665281290598479
13	11.793890586174369 + 1.6524771364075785im	$3.357756113171857 \cdot 10^{10}$	$3.2960214141301664 \cdot 10^{13}$	2.045820276678428
14	13.992406684487216 - 2.5188244257108443im	$1.0612064533081976 \cdot 10^{11}$	$9.545941595183662 \cdot 10^{14}$	2.5188358711909045
15	13.992406684487216 + 2.5188244257108443im	$1.0612064533081976 \cdot 10^{11}$	$9.545941595183662 \cdot 10^{14}$	2.7128805312847097
16	16.73074487979267 - 2.812624896721978im	$3.315103475981763 \cdot 10^{11}$	$2.7420894016764064 \cdot 10^{16}$	2.9060018735375106
17	16.73074487979267 + 2.812624896721978im	$3.315103475981763 \cdot 10^{11}$	$2.7420894016764064 \cdot 10^{16}$	2.825483521349608
18	19.5024423688181 - 1.940331978642903im	$9.539424609817828 \cdot 10^{12}$	$4.2525024879934694 \cdot 10^{17}$	2.454021446312976
19	19.5024423688181 + 1.940331978642903im	$9.539424609817828 \cdot 10^{12}$	$4.2525024879934694 \cdot 10^{17}$	2.004329444309949
20	20.84691021519479 + 0.0im	$1.114453504512 \cdot 10^{13}$	$1.3743733197249713 \cdot 10^{18}$	0.8469102151947894

Pomimo nieznacznych odczyleń w obliczonych pierwiastkach, otrzymane podczas weryfikacji wyniki w znaczącym stopniu odbiegają od poprawnych. Źródłem zaobserwowanych w doświadczeniu błędów jest złe uwarunkowanie zadania - współczynniki wielomianu Wilkinsona nie są dokładnie reprezentowane nawet w arytmetyce **Float64**. Początkowo drobny błąd jest w wielomianie mnożony przez czynnik rzędu 19!. Operacja tego typu sprawia, że odchylenie rzędu 10^{-13} przekształca się w około 20000. Dla większych błędów taka amplifikacja skutkuje nieprawidłościami sięgającymi nawet bilionów.

Skalę problemów z reprezentacją współczynników zaobserwować można po zamianie wartości przy x^{19} z -210 na $-210 - 2^{-23}$. Dokonanie tej zmiany sprawia, że niektóre z obliczonych pierwiastków przyjmują wartości zespolone, chociaż wszystkie pierwiastki wielomianu powinny mieć wartości rzeczywiste.

5 Zadanie 5

Zadane doświadczenie polega na rozwiązaniu w arytmetykach **Float32** i **Float64** równania rekurencyjnego

$$p_n + 1 := p_n + rp_n(1 - p_n), \text{ dla } n = 0, 1, 2, ...$$

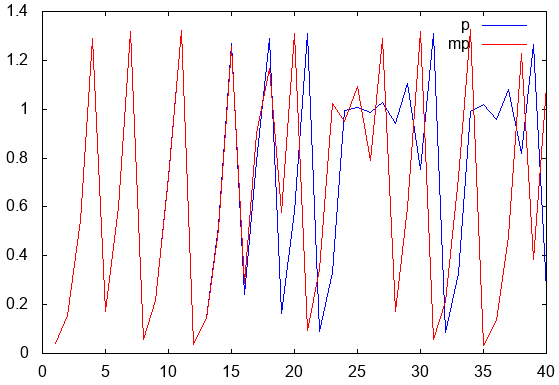
używając stałej $r = 3$, oraz $x_0 = 0.01$. Równanie iterowane jest czterdziestokrotnie. Dodatkowo, oprócz normalnego rozwiązania równania, w arytmetyce **Float32** wynik dla dziesiątej iteracji ograniczony ma zostać do trzech miejsc po przecinku.

Wyniki dla Float32		
n	Algorytm bazowy p	Algorytm zmodyfikowany mp
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	1.3241479
12	0.037716985	0.036488414
13	0.14660022	0.14195944
14	0.521926	0.50738037
15	1.2704837	1.2572169
16	0.2395482	0.28708452
17	0.7860428	0.9010855
18	1.2905813	1.1684768
19	0.16552472	0.577893
20	0.5799036	1.3096911
21	1.3107498	0.09289217
22	0.088804245	0.34568182
23	0.3315584	1.0242395
24	0.9964407	0.94975823
25	1.0070806	1.0929108
26	0.9856885	0.7882812
27	1.0280086	1.2889631
28	0.9416294	0.17157483
29	1.1065198	0.59798557
30	0.7529209	1.3191822

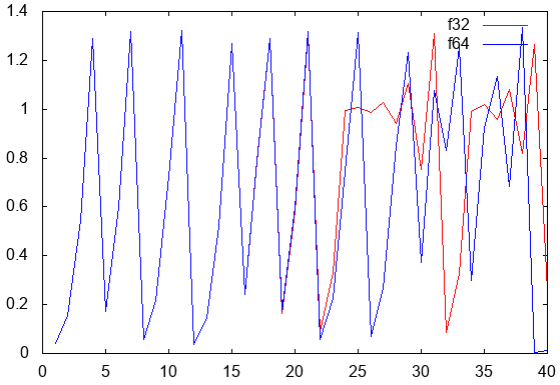
31	1.3110139	0.05600393
32	0.0877831	0.21460639
33	0.3280148	0.7202578
34	0.9892781	1.3247173
35	1.021099	0.034241438
36	0.95646656	0.13344833
37	1.0813814	0.48036796
38	0.81736827	1.2292118
39	1.2652004	0.3839622
40	0.25860548	1.093568

Wyniki dla różnych arytmetyk		
n	Float32	Float64
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



Rys. 5. Algorytmy przed i po modyfikacji



Rys. 6. Typy Float32 i Float64

Powodem anomalii w zaprezentowanych wynikach jest specyfika użytego algorytmu - iteracyjne kroki funkcji bazują na wartościach wyznaczonych w krokach poprzednich. Wystarczy, żeby jeden z wcześniejszych kroków wyznaczył obarczony błędem wynik, a jego następnicy zaczną pracować na danych wejściowych, których błąd zostanie nie tylko zachowany, ale często kumulowany w miarę działania programu.

Analizując wykres funkcji *mp* stwierdzić można, że pomimo nieznacznych zmian w jej przebiegu dla kroku dziesiątego, anomalie pojawiły się wcześniej niż w przypadku funkcji p (gdzie w żadnym kroku nie zaburzano sztucznie jej wartości). Ta obserwacja potwierdza zajście wcześniej opisanego zjawiska.

Pomimo podwyższonej precyzji arytmetyki **Float64** błąd nie został zniwelowany. Stanowi to poważny problem w algorytmach, gdzie obarczony błędem wynik użyty zostanie w kolejnych obliczeniach, które również generują wynik z nieuniknionym błędem. Efektem działania takiego algorytmu jest wynik, który już po kilku iteracjach potrafi znacząco odbiegać od oczekiwanej, poprawnej wartości.

6 Zadanie 6

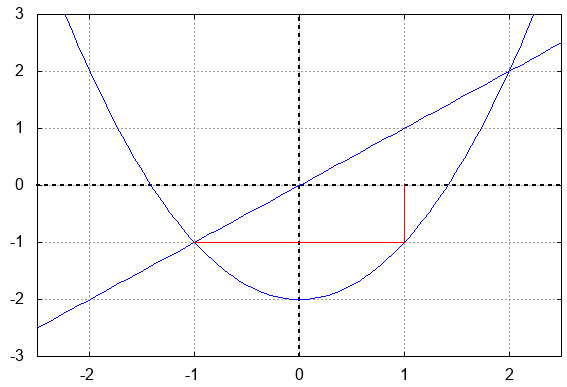
W zadaniu rozwiązać należy równanie

$$x_n + 1 := x_n^2 + c, \text{ dla } n = 0, 1 \text{ , ...}$$

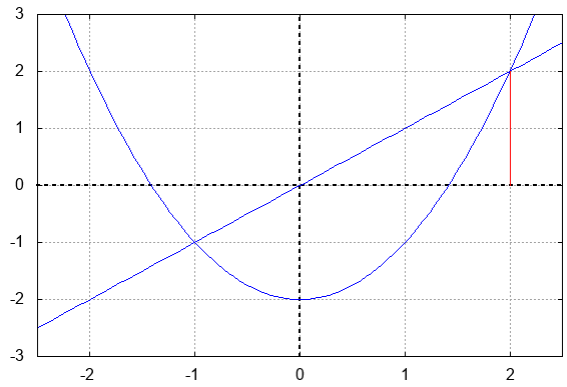
dla danych:

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

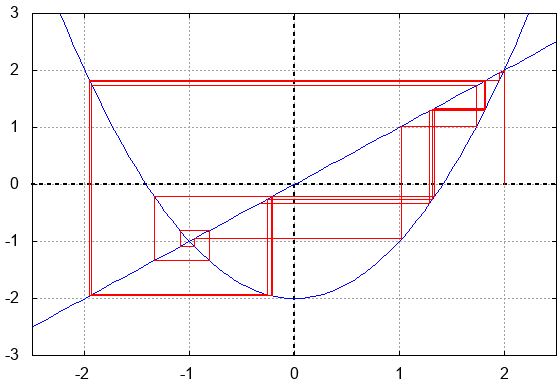
Wyniki							
n	$c=-2$			$c=-1$			
	$x_0=1$	$x_0=2$	$x_0=1.99999999999999$	$x_0=1$	$x_0=-1$	$x_0=0.75$	$x_0=0.25$
1	-1	2	1.99999999999996	0	0	-0.4375	-0.9375
2	-1	2	1.9999999999998401	-1	-1	-0.80859375	-0.12109375
3	-1	2	1.9999999999993605	0	0	-0.3461761474609375	-0.9853363037109375
4	-1	2	1.999999999997442	-1	-1	-0.8801620749291033	-0.029112368589267135
5	-1	2	1.9999999999897682	0	0	-0.2253147218564956	-0.9991524699951226
6	-1	2	1.9999999999590727	-1	-1	-0.9492332761147301	-0.0016943417026455965
7	-1	2	1.999999999836291	0	0	-0.0989561875164966	-0.9999971292061947
8	-1	2	1.9999999993451638	-1	-1	-0.9902076729521999	$-5.741579369278327 \cdot 10^{-6}$
9	-1	2	1.9999999973806553	0	0	-0.01948876442658909	-0.9999999999670343
10	-1	2	1.999999989522621	-1	-1	-0.999620188061125	$-6.593148249578462 \cdot 10^{-11}$
11	-1	2	1.9999999580904841	0	0	-0.0007594796206411569	-1.0
12	-1	2	1.9999998323619383	-1	-1	-0.9999994231907058	0.0
13	-1	2	1.9999993294477814	0	0	$-1.1536182557003727 \cdot 10^{-6}$	-1.0
14	-1	2	1.9999973177915749	-1	-1	-0.9999999999986692	0.0
15	-1	2	1.9999892711734937	0	0	$-2.6616486792363503 \cdot 10^{-12}$	-1.0
16	-1	2	1.9999570848090826	-1	-1	-1.0	0.0
17	-1	2	1.999828341078044	0	0	0.0	-1.0
18	-1	2	1.9993133937789613	-1	-1	-1.0	0.0
19	-1	2	1.9972540465439481	0	0	0.0	-1.0
20	-1	2	1.9890237264361752	-1	-1	-1.0	0.0
21	-1	2	1.9562153843260486	0	0	0.0	-1.0
22	-1	2	1.82677862987391	-1	-1	-1.0	0.0
23	-1	2	1.3371201625639997	0	0	0.0	-1.0
24	-1	2	-0.21210967086482313	-1	-1	-1.0	0.0
25	-1	2	-1.9550094875256163	0	0	0.0	-1.0
26	-1	2	1.822062096315173	-1	-1	-1.0	0.0
27	-1	2	1.319910282828443	0	0	0.0	-1.0
28	-1	2	-0.2578368452837396	-1	-1	-1.0	0.0
29	-1	2	-1.9335201612141288	0	0	0.0	-1.0
30	-1	2	1.7385002138215109	-1	-1	-1.0	0.0
31	-1	2	1.0223829934574389	0	0	0.0	-1.0
32	-1	2	-0.9547330146890065	-1	-1	-1.0	0.0
33	-1	2	-1.0884848706628412	0	0	0.0	-1.0
34	-1	2	-0.8152006863380978	-1	-1	-1.0	0.0
35	-1	2	-1.3354478409938944	0	0	0.0	-1.0
36	-1	2	-0.21657906398474625	-1	-1	-1.0	0.0
37	-1	2	-1.953093509043491	0	0	0.0	-1.0
38	-1	2	1.8145742550678174	-1	-1	-1.0	0.0
39	-1	2	1.2926797271549244	0	0	0.0	-1.0
40	-1	2	-0.3289791230026702	-1	-1	-1.0	0.0



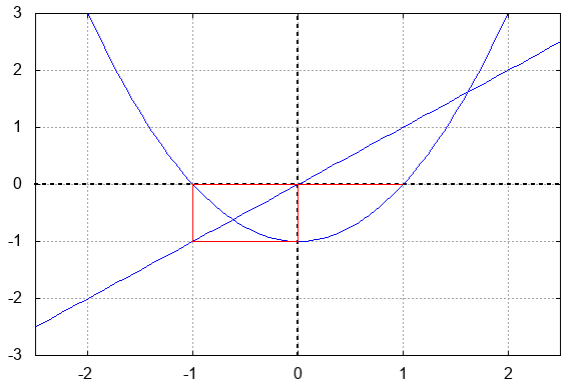
Rys. 7. $c=-2, x_0=1$



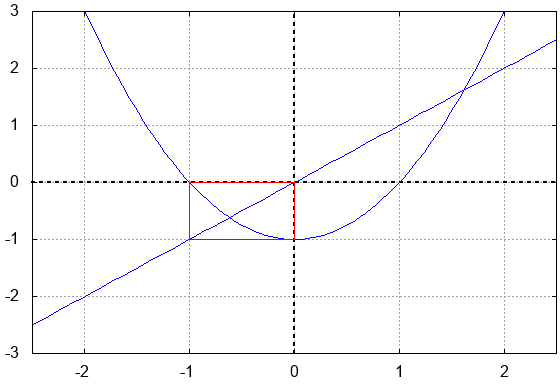
Rys. 8. $c=-2, x_0=2$



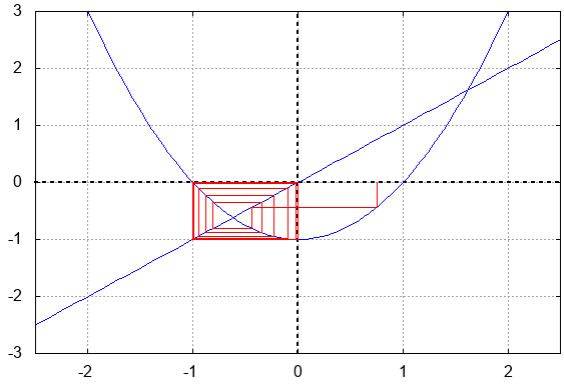
Rys. 9. $c = -2, x_0 = 1.99999999999999$



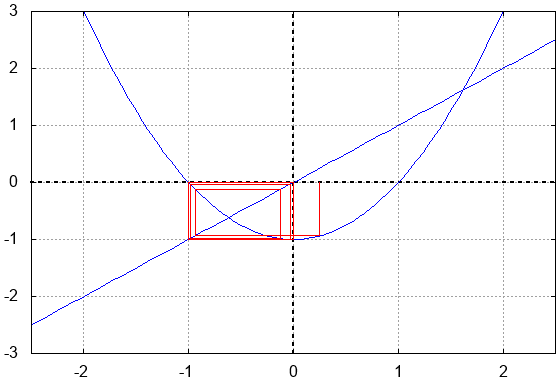
Rys. 10. $c = -1, x_0 = 1$



Rys. 11. $c = -1, x_0 = -1$



Rys. 12. $c = -1, x_0 = 0.75$



Rys. 13. $c = -1, x_0 = 0.25$

Powyższe wykresy wyraźnie pokaują wpływ odpowiednio dobranych danych wejściowych na stabilność algorytmu. W przypadku, gdy zarówno c, jak i x_0 mają wartości całkowite iteracje zachowują się w sposób stabilny. Największą chaotyczność wyników iteracyjnych można natomiast zaobserwować, gdy $c = -2$ (dzięki czemu funkcja posiada całkowite punkty stałe: -1 i 2), a $x_0 = 1.99999999999999$ - liczbie, która z pośród wszystkich użytych w doświadczeniu ma najwięcej cyfr po przecinku. Obserwacje te pozwalają

wywnioskować, że w zadanym przedziale [-2, 2] nie istnieje wiele wartości, które generują stabilne wyniki.

Ciekawe zjawisko zauważyć można dla $c = -1$. Punkty stałe w tym przypadku mają wartości $\frac{1-\sqrt{5}}{2}$ i $\frac{1+\sqrt{5}}{2}$. Dla wartości x_0 takich jak 0.25, czy 0.75 wyniki iteracji, choć początkowo chaotyczne, ustabilizowały się i naprzemiennie przyjmowały wartości 0 i -1. Fakt ten sugeruje, że do analizy tego zjawiska można posłużyć się arytmetyką o skończonej precyzji.