Sprawozdanie z listy nr 2 Obliczenia Naukowe

Marek Świergoń (261750)

14 listopada 2022, PWr, WIiT INA

1 Zadanie 1

1.1 Cele zadania

Powtórzyć zadanie 5. z listy 1., dla zaburzonego wektora x. Dane użyte w liście 1.:

$$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$$

$$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049].$$

Dane z zaburzonym wektorem x:

$$\tilde{x} = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995],$$

y jak wyżej.
Celem zadania 5. z listy 1. było obliczenie iloczynu skalarnego dwóch wektorów x i y w typach Float32 i Float64 z wykorzystaniem czterech algorytmów różniących się kolejnością sumowania:

1. "w przód", tj. $\sum_{i=1}^{n} x_i y_i$
2. "w tył", tj. $\sum_{i=n}^{1} x_i y_i$
3. dodając osobno dodatnie iloczyny składowe w porządku od największego do najmniejszego i ujemne iloczyny składowe w porządku od najmniejszego do największego, następnie dodając obliczone sumy częściowe
4. przeciwnie do metody 3.

1.2 Rozwiązanie zadania

Metody wyznaczające iteracyjnie powyższe liczby znajdują się w pliku *zadanie1.jl*, są one identyczne do tych z pliku *zadanie5.jl* z poprzedniej listy.

Nr alg.	Float32 dla $x \cdot y$	Float32 dla $\tilde{x} \cdot y$	Float64 dla $x \cdot y$	Float64 dla $\tilde{x} \cdot y$
1	-0.4999443	-0.4999443	$1.0251881368296672 * 10^{-10}$	-0.004296342739891585
2	-0.4543457	-0.4543457	$-1.5643308870494366 * 10^{-10}$	-0.004296342998713953
3	-0.5	-0.5	0.0	-0.004296342842280865
4	-0.5	-0.5	0.0	-0.004296342842280865

Tabela 1: Porównanie wyniku obliczania iloczynu skalarnego dwóch wektorów z użyciem różnych algorytmów dla danych prawidłowych oraz dla zaburzonego wektora x ($\tilde{x}$). Prawidłowy wynik dla $x \cdot y$ to $-1.00657107000000 * 10^{-11}$.

1.3 Interpretacja wyników i wnioski

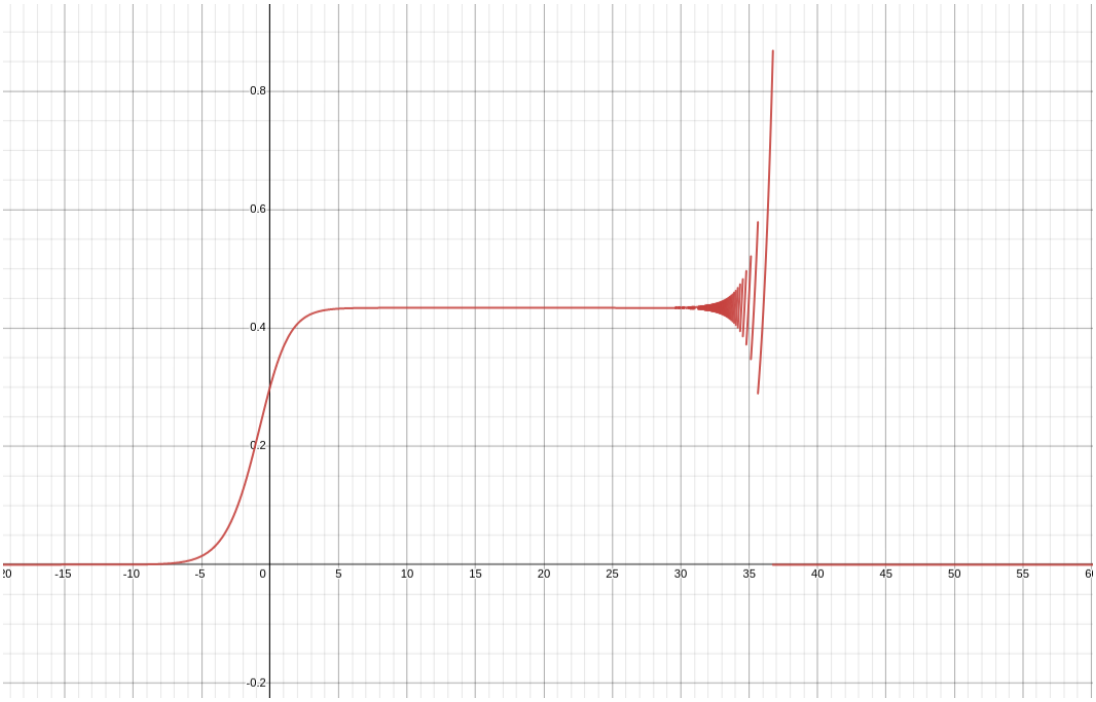
Zauważmy, że dla arytmetyki Float32 wyniki nie różnią się. Wynika to ze zbyt małej precyzji tej arytmetyki, zadanie obliczenia iloczynu skalarnego powyższych wektorów jest obarczone dużym błędem, wynikającym z wykonywania działań na liczbach o istotnie różnych rzędach wielkości.
Arytmetyka Float64 charakteryzuje się większą precyzją, w której już możemy zobaczyć duże rozbieżności wyników działań $x \cdot y$ i $\tilde{x} \cdot y$, pomimo tego, że zaburzenie wektora x jest bardzo niewielkie. Możemy zatem wywnioskować, iż algorytmy wyliczania iloczynu skalarnego, zastosowane dla zadanych powyżej wektorów x i y, są wrażliwe na nawet niewielkie zmiany danych. W związku z tym zadanie policzenia iloczynu skalarnego $x \cdot y$ jest źle uwarunkowane.

2 Zadanie 2

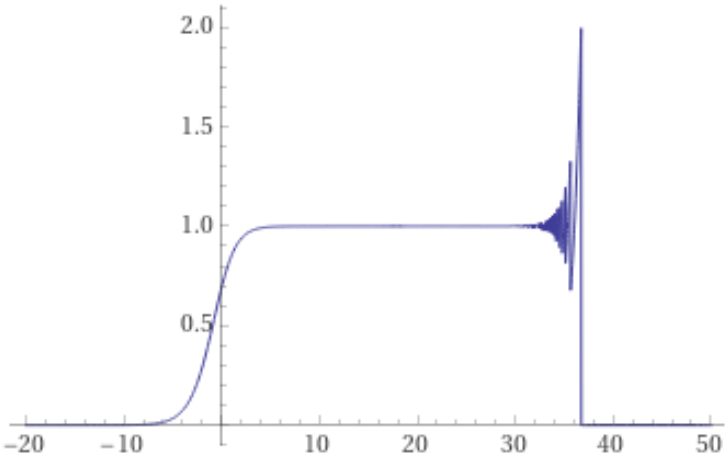
2.1 Cel zadania

Narysować wykres funkcji $f(x) = e^x \ln(1+e^{-x})$ w co najmniej dwóch dowolnych programach do wizualizacji, policzyć granicę funkcji $\lim_{x\to\infty} f(x)$ i porównać z wykresami funkcji.

2.2 Rozwiązanie i wyniki



Rysunek 1: Wykres funkcji $f(x)$ narysowany z użyciem kalkulatora graficznego Desmos.



Rysunek 2: Wykres funkcji $f(x)$ narysowany z użyciem programu Wolfram Alpha.

Obliczenie granicy funkcji:

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

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

2.3 Interpretacja wyników i wnioski

Zauważmy, że wyniki przedstawione przez programy graficzne nie pokrywają się z wyliczoną granicą funkcji *f(x)*: na wykresach wartość funkcji zdaje się być równa zero dla odpowiednio dużego x, a w rzeczywistości wartość ta powinna zbiegać do 1 przy $x \to \infty$. Już dla $x > 30$ funkcja zaczyna na wykresach zachowywać się nieprawidłowo. Oscylacje wynikają z faktu, że w zastosowanym wprost wzorze $f(x) = e^x \ln(1+e^{-x})$ czynnik e^x staje się bardzo duży, a czynnik $\ln(1+e^{-x})$ bardzo mały. Działanie mnożenia liczb różniących się wielkością rzędów jest obarczone dużym błędem (stąd oscylacje). Ponadto w pewnym momencie w ramach arytmetyki stosowanej w użytych kalkulatorach graficznych $\ln(1+e^{-x}) \approx 0$, przez co wynik iloczynu zostaje wyzerowany i nie odwzorowuje poprawnie oczekiwanego wyniku.

3 Zadanie 3

3.1 Cele zadania

Rozwiązać dwoma metodami układ równań liniowych postaci $\mathbf{Ax} = \mathbf{b}$ dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n\times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n$. Metody te to:

- metoda eliminacji Gausa, czyli $x = A\backslash b$,
- metoda korzystająca wprost ze wzoru $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ (w języku Julia `x = inv(A)*b`).

Macierz $\mathbf{A}$ ma być wygenerowana na dwa sposoby:

- $\mathbf{A} = \mathbf{H}^n$, gdzie $\mathbf{H}^n$ jest macierzą Hilberta stopnia n wygenerowaną w Julii za pomocą funkcji `hilb(n)` zdefiniowanej i zaimplementowanej w pliku załączonym do zadania (*hilb.jl*).
- $\mathbf{A} = \mathbf{R}^n$, gdzie $\mathbf{R}^n$ jest losową macierzą stopnia n z zadanym wskaźnikiem uwarunkowania c wygenerowaną w Julii za pomocą funkcji `matcond(n,c)` zdefiniowanej i zaimplementowanej w pliku załączonym do zadania (*matcond.jl*).

Opisane powyżej zadanie należało wykonać dla macierzy Hilberta z rosnącym stopniem $n > 1$ oraz dla macierzy losowej, dla której $n \in \{5, 10, 20\}$ oraz $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$. Wyniki ($\tilde{\mathbf{x}}$) porównać z rozwiązaniem dokładnym $\mathbf{x} = (1, \ldots, 1)^T$ poprzez wyliczenie błędu względnego $\frac{||\mathbf{x}-\tilde{\mathbf{x}}||}{||\mathbf{x}||}$.

3.2 Rozwiązanie

Metody służące do wyznaczenia błędu względnego (przy zastosowaniu wyżej wymienionych metod do rozwiązania układu równań liniowych) znajdują się w pliku *zadanie3.jl*. Zostały do niego dodane również funkcje `hilb(n)` (z pliku *hilb.jl*) oraz `matcond(n,c)` (z pliku *matcond.jl*). Poza wyznaczeniem błędu metody wyświetlają rozmiar, wskaźnik uwarunkowania i rząd danej macierzy.
Wszystkie obliczenia wykonano w arytmetyce Float64 aby zminimalizować błąd wynikający z samej precyzji arytmetyki.

3.3 Wyniki

n	cond($\mathbf{H}^n$)*	rank($\mathbf{H}^n$)	$\frac{\|\mathbf{x}-\mathbf{x}_{gauss}\|}{\|\mathbf{x}\|}$	$\frac{\|\mathbf{x}-\mathbf{x}_{inv}\|}{\|\mathbf{x}\|}$
1	1.0	1	0.0	0.0
2	$1.928147 * 10^{1}$	2	$5.661049 * 10^{-16}$	$1.404333 * 10^{-15}$
3	$5.240568 * 10^{2}$	3	$8.022594 * 10^{-15}$	0.0
4	$1.551374 * 10^{4}$	4	$4.137410 * 10^{-14}$	0.0
5	$4.766073 * 10^{5}$	5	$1.682843 * 10^{-12}$	$3.354436 * 10^{-12}$
6	$1.495106 * 10^{7}$	6	$2.618913 * 10^{-10}$	$2.016376 * 10^{-10}$
7	$4.753674 * 10^{8}$	7	$1.260687 * 10^{-8}$	$4.713280 * 10^{-9}$
8	$1.525758 * 10^{10}$	8	$6.124090 * 10^{-8}$	$3.077484 * 10^{-7}$
9	$4.931538 * 10^{11}$	9	$3.875163 * 10^{-6}$	$4.541268 * 10^{-6}$
10	$1.602442 * 10^{13}$	10	$8.670390 * 10^{-5}$	$2.501493 * 10^{-4}$
11	$5.222701 * 10^{14}$	10	$1.582781 * 10^{-4}$	$7.618304 * 10^{-3}$
12	$1.751595 * 10^{16}$	11	$1.339621 * 10^{-1}$	$2.589941 * 10^{-1}$
13	$3.188395 * 10^{18}$	11	$1.103970 * 10^{-1}$	5.331276

Tabela 2: Wartości wskaźnika uwarunkowania i rzędu macierzy Hilberta rozmiaru n oraz błędy względne rozwiązań $\mathbf{H}_n\tilde{\mathbf{x}} = \mathbf{b}$ dla metody Gaussa i metody korzystającej z odwrotności macierzy. Wyniki są podane dla $n < 14$, ponieważ precyzja Float64 jest niewystarczająca do poprawnych wyliczeń dla większych macierzy Hilberta.

c	n	rank($\mathbf{H}^n$)	$\frac{\|\mathbf{x}-\mathbf{x}_{gauss}\|}{\|\mathbf{x}\|}$	$\frac{\|\mathbf{x}-\mathbf{x}_{inv}\|}{\|\mathbf{x}\|}$
1	5	5	$2.937374 * 10^{-16}$	$2.627267 * 10^{-16}$
	10	10	$3.140185 * 10^{-16}$	$2.077037 * 10^{-16}$
	20	20	$4.475452 * 10^{-16}$	$3.901609 * 10^{-16}$
10	5	5	$6.734946 * 10^{-16}$	$8.656892 * 10^{-16}$
	10	10	$2.135557 * 10^{-16}$	$2.895107 * 10^{-16}$
	20	20	$8.458850 * 10^{-16}$	$8.455207 * 10^{-16}$
10^3	5	5	$8.945945 * 10^{-15}$	$1.002501 * 10^{-14}$
	10	10	$2.128376 * 10^{-14}$	$2.289365 * 10^{-14}$
	20	20	$1.595696 * 10^{-14}$	$1.504768 * 10^{-14}$
10^7	5	5	$1.200502 * 10^{-10}$	$1.723035 * 10^{-10}$
	10	10	$8.395776 * 10^{-11}$	$9.351799 * 10^{-11}$
	20	20	$3.112270 * 10^{-11}$	$3.494165 * 10^{-11}$
10^{12}	5	5	$4.302324 * 10^{-5}$	$3.768643 * 10^{-5}$
	10	10	$4.061037 * 10^{-5}$	$4.498302 * 10^{-5}$
	20	20	$1.212850 * 10^{-5}$	$1.919262 * 10^{-5}$
10^{16}	5	4	0.4387790	0.3773365
	10	9	0.07567409	0.05719689
	20	19	0.2478111	0.3564761

Tabela 3: Wartości wskaźnika rzędu macierzy losowej o rozmiarze n i wskaźniku uwarunkowania c oraz błędy względne rozwiązań $\mathbf{R}_n\tilde{\mathbf{x}} = \mathbf{b}$ dla metody Gaussa i metody korzystającej z odwrotności macierzy.

3.4 Interpretracja wyników i wnioski

- W przypadku macierzy Hilberta, zarówno wskaźnik uwarunkowania, jak i błędy względne dla poszczególnych metod, bardzo szybko rosną wraz ze wzrostem rozmiaru macierzy. Wywnioskować można, że zadanie obliczenia układu równań $\mathbf{H}_n\mathbf{x} = \mathbf{b}$ jest źle uwarunkowane, nawet dla macierzy Hilberta niewielkich rozmiarów.

- Dla macierzy losowych zauważyć możemy, że wielkość błędu względnego rozwiązania zadania podanymi wcześniej metodami zależy głównie od wartości wskaźnika uwarunkowania; nie zależy istotnie od jej rozmiaru.

- Gdy dowolna macierz $\mathbf{A}$ ma wysoki wskaźnik uwarunkowania, to zadanie rozwiązania układu równań liniowych $\mathbf{Ax} = \mathbf{b}$ staje się źle uwarunkowane, a otrzymane wyniki obarczone są bardzo dużym błędem, rzędowo znacznie większym od precyzji arytmetyki, w której wykonywane były obliczenia.

4 Zadanie 4

4.1 Cel zadania

Dany jest wielomian Wilkinsona w dwóch postaciach: naturalnej, tj.
$P(x) = x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} - 1672280820x^{15} + 40171771630x^{14} - 756111184500x^{13} + 11310276995381x^{12} - 135585182899530x^{11} + 1307535010540395x^{10} - 10142299865511450x^{9} + 63030812099294896x^{8} - 311333643161390640x^{7} + 1206647803780373360x^{6} - 3599979517947607200x^{5} + 8037811822645051776x^{4} - 12870931245150988800x^{3} + 13803759753640704000x^{2} - 8752948036761600000x + 2432902008176640000$,
oraz w postaci iloczynowej, tj.
$p(x) = (x-20)(x-19)(x-18)(x-17)(x-16)(x-15)(x-14)(x-13)(x-12)(x-11)(x-10)(x-9)(x-8)(x-7)(x-6)(x-5)(x-4)(x-3)(x-2)(x-1)$. Wyznaczyć w Julii pierwiastki tego wielomianu z_k oraz policzyć $|z_k - k|$, $|P(z_k)|$, $|p(z_k)|$, $k \in \{1, \ldots, 20\}$. Następnie zaburzyć wielomian poprzez zaburzenie współczynnika $\tilde{a_{19}} = a_{19} - 2^{-23}$ i ponownie wyznaczyć pierwiastki, wyjaśnić zachodzące zjawiska.

4.2 Rozwiązanie

Rozwiązanie tego zadania w całości znajduje się w pliku *zadanie4.jl*.
Wykorzystując metody z pakietu Polynomials zapisano wielomian Wilkinsona w postaci naturalnej i iloczynowej (użycie funkcji `fromroots()`). Następnie wyznaczone zostały pierwiastki tego wielomianu z_k z użyciem funkcji `roots()` zastosowanej na wielomianie w postaci naturalnej. Wyliczono błąd bezwzględny wyznaczonych pierwiastków oraz wartości obu form wielomianów dla tych pierwiastków.
W drugiej części zadania wielomian Wilkinsona został zaburzony, tj. $\tilde{a_{19}} = a_{19} - 2^{-23}$ i porównano pierwiastki wyznaczone po zaburzeniu z prawidłowymi.

4.3 Wyniki

k	z_k	$\lvert z_k - k\rvert$	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$
1	0.999999999999699	0.000000000000301	$3.569650964788257 * 10^4$	$5.518479490350445 * 10^6$
2	2.000000000028318	0.000000000028318	$1.762526002666841 * 10^5$	$7.378697629901740 * 10^{19}$
3	2.999999999592097	0.000000000407903	$2.791576968824087 * 10^5$	$3.320413931687579 * 10^{20}$
4	3.999999983737532	0.000000016262468	$3.027109298899109 * 10^6$	$8.854437035384718 * 10^{20}$
5	5.000000665769791	0.000000665769791	$2.291747375656708 * 10^7$	$1.844675205654569 * 10^{21}$
6	5.999989245824773	0.000010754175227	$1.290241728420510 * 10^8$	$3.320394888870117 * 10^{21}$
7	7.000102002793008	0.000102002793008	$4.805112754602064 * 10^8$	$5.423593016891273 * 10^{21}$
8	7.999355829607762	0.000644170392238	$1.637952021896114 * 10^9$	$8.262050140110275 * 10^{21}$
9	9.002915294362053	0.002915294362053	$4.877071372550003 * 10^9$	$1.196559421646277 * 10^{22}$
10	9.990413042481725	0.009586957518275	$1.363863819545813 * 10^{10}$	$1.655260133520688 * 10^{22}$
11	11.025022932909318	0.025022932909318	$3.585631295130865 * 10^{10}$	$2.247833297924790 * 10^{22}$
12	11.953283253846857	0.046716746153143	$7.533332360358197 * 10^{10}$	$2.886944688412679 * 10^{22}$
13	13.074314032447340	0.074314032447340	$1.960598812433082 * 10^{11}$	$3.807325552826988 * 10^{22}$
14	13.914755591802127	0.085244408197873	$3.575134782310432 * 10^{11}$	$4.612719853150334 * 10^{22}$
15	15.075493799699476	0.075493799699476	$8.216271236455970 * 10^{11}$	$5.901011420218566 * 10^{22}$
16	15.946286716607972	0.053713283392028	$1.551497888049407 * 10^{12}$	$7.010874106897764 * 10^{22}$
17	17.025427146237412	0.025427146237412	$3.694735918486229 * 10^{12}$	$8.568905825736165 * 10^{22}$
18	17.990921352716480	0.009078647283520	$7.650109016515867 * 10^{12}$	$1.014479936104443 * 10^{23}$
19	19.001909818299438	0.001909818299438	$1.143527374972120 * 10^{13}$	$1.199037620237126 * 10^{23}$
20	19.999809291236637	0.000190708763363	$2.792410639368073 * 10^{13}$	$1.401911741431813 * 10^{23}$

Tabela 4: Wartości pierwiastków wielomianu Wilkinsona rzeczywiste (k) i wyliczone z użyciem funkcji `roots()` z pakietu Polynomials (z_k). Błąd bezwzględny wyliczenia pierwiastka i wartości bezwzględne wielomianu Wilkinsona w wyliczonych pierwiastkach (wielomian zadany w postaci naturalnej ($P(z_k)$) i w postaci iloczynowej ($p(z_k)$)).

k	$\|P(k)\|$	$\|p(k)\|$
1	0.0	0.0
2	$8.192 * 10^3$	0.0
3	$2.7648 * 10^4$	0.0
4	$6.22592 * 10^5$	0.0
5	$2.176 * 10^6$	0.0
6	$8.84736 * 10^6$	0.0
7	$2.4410624 * 10^7$	0.0
8	$5.89824 * 10^7$	0.0
9	$1.45753344 * 10^8$	0.0
10	$2.27328 * 10^8$	0.0
11	$4.79074816 * 10^8$	0.0
12	$8.75003904 * 10^8$	0.0
13	$1.483133184 * 10^9$	0.0
14	$2.457219072 * 10^9$	0.0
15	$3.905712 * 10^9$	0.0
16	$6.029312 * 10^9$	0.0
17	$9.116641408 * 10^9$	0.0
18	$1.333988352 * 10^{10}$	0.0
19	$1.9213101568 * 10^{10}$	0.0
20	$2.7193344 * 10^{10}$	0.0

Tabela 5: Dodatkowe sprawdzenie jakie wartości zwróci wielomian Wilkinsona zadany w postaci naturalnej ($P(z_k)$) i w postaci iloczynowej ($p(z_k)$) dla argumentów będącymi dokładnymi pierwiastkami tego wielomianu (k).

k	$\tilde{z_k}$	$\|\tilde{z_k} - k\|$
1	0.9999999999998357 + 0.0im	1.6431300764452317 * 10−13
2	2.0000000000550373 + 0.0im	5.503730804434781 * 10−11
3	2.99999999660342 + 0.0im	3.3965799062229962 * 10−9
4	4.000000089724362 + 0.0im	8.972436216225788 * 10−8
5	4.99999857388791 + 0.0im	1.4261120897529622 * 10−6
6	6.000020476673031 + 0.0im	$2.0476673030955794 * 10^{-5}$
7	6.99960207042242 + 0.0im	0.00039792957757978087
8	8.007772029099446 + 0.0im	0.007772029099445632
9	8.915816367932559 + 0.0im	0.0841836320674414
10	10.095455630535774 - 0.6449328236240688im	0.6519586830380407
11	10.095455630535774 + 0.6449328236240688im	1.1109180272716561
12	11.793890586174369 - 1.6524771364075785im	1.665281290598479
13	11.793890586174369 + 1.6524771364075785im	2.0458202766784277
14	13.992406684487216 - 2.5188244257108443im	2.518835871190904
15	13.992406684487216 + 2.5188244257108443im	2.7128805312847097
16	16.73074487979267 - 2.812624896721978im	2.9060018735375106
17	16.73074487979267 + 2.812624896721978im	2.825483521349608
18	19.5024423688181 - 1.940331978642903im	2.4540214463129764
19	19.5024423688181 + 1.940331978642903im	2.0043294443099486
20	20.84691021519479 + 0.0im	0.8469102151947894

Tabela 6: Wartości pierwiastków wielomianu Wilkinsona rzeczywiste (k) i wyliczone z użyciem funkcji `roots()` z pakietu Polynomials na zaburzonym wielomianie Wilkinsona ($\tilde{z_k}$). Błąd bezwzględny wyliczenia pierwiastka.

4.4 Interpretacja wyników i wnioski

W Tabeli 4 zauważyć możemy, że wyliczone w Julii wartości pierwiastków nie pokrywają się dokładnie z rzeczywistymi pierwiastkami wielomianu Wilkinsona. Generalnie mniejsze pierwiastki zostały dokładniej wyznaczone niż większe, ale wydawać by się mogło, że błędy nie są bardzo duże ($|z_k - k| < 10^{-1}$). Pomimo tego, wartości wielomianu w niedokładnie wyznaczonych pierwiastkach, są bardzo duże, największe dla większych pierwiastków (nawet $> 10^{13}$). Sugeruje to, że błąd, jaki wystąpił przy błędnie wyznaczonym pierwiastku, kumuluje się bardzo silnie w trakcie kolejnych działań w ramach wykorzystanej arytmetyki.

Ponieważ zmuszeni jesteśmy pracować w arytmetyce o ograniczonej precyzji (Float64), to błąd związany z precyzją arytmetyki oraz działaniami wykonywanymi w tej arytmetyce jest nieunikniony, a **zadanie wyznaczenia pierwiastków wielomianu Wilkinsona jest bardzo źle uwarunkowane**. Potwierdza to również druga część eksperymentu Wilkinsona, gdzie w sposób bardzo marginalny został zaburzony jeden ze współczynników wielomianu. Tak mała zmiana wartości współczynnika spowodowała, że pierwiastki wielomianu otrzymały część urojoną, a ich odległość od faktycznych pierwiastków stała się duża, co pokazuje Tabela 6.

Z pomocą Tabeli 5 możemy dodatkowo zauważyć, że nie jesteśmy w stanie w sposób dokładny przechowywać w arytemtyce Float64 wielomianu Wilkinsona w postaci naturalnej. Wynika to z tego, że przy niskich potęgach x znajdują się bardzo duże współczynniki; są one na tyle duże, że nie możemy ich zapisać bez pominięcia kilku cyfr znaczących. Przez to dla całkowicie poprawnych pierwiastków k (za wyjątkiem $k = 1$) wartość $|P(k)| \gg 0$, natomiast $|p(k)| = 0$

5 Zadanie 5

5.1 Cel zadania

Dane jest równanie rekurencyjne pewnego modelu logistycznego: $p_{n+1} := p_n + r * p_n * (1 - p_n)$, dla $n = 0, 1, ...$ Wyliczyć 40 iteracji tego równania dla $p_0 = 0.01$ i $r = 3$, stosując różne metody i arytmetyki, tj. wykonać:

1. wszystkie 40 iteracji w arytmetyce Float32 bez modyfikacji danych,
2. najpierw 10 iteracji w arytmetyce Float32, następnie wynik uciąć do 3 liczb znaczących po przecinku i ten wynik zastosować do kolejnych 30 iteracji w arytmetyce Float32,
3. wszystkie 40 iteracji w arytmetyce Float64 bez modyfikacji danych.

5.2 Rozwiązanie

Kod źródłowy zawierający wykonanie wszystkich metod zgodnie z celem zadania znajduje się w pliku *zadanie5.jl*. Do realizacji zadania w tym pliku zaimplementowano funkcje `nextFloat32(p,r)` i `nextFloat64(p,r)`, wyliczające p_{n+1} w arytmetykach odpowiednio Float32 i Float64.

5.3 Wyniki

Numer iteracji n	p_n dla Float32	p_n dla Float32 z obcięciem po 10 iteracji	p_n dla Float64
0	0.01	0.01	0.01
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	0.7229306	0.722914301179573
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

Tabela 7: Wartości p_n równania rekurencyjnego modelu logistycznego dla różnych zastosowanych precyzji wyliczania tychże wartości.

5.4 Interpretacja wyników i wnioski

W początkowych kilku iteracjach wyniki są zbliżone do siebie, jednak wraz ze wzrostem numeru iteracji stają się w sposób nieprzewidywalny coraz bardziej rozbieżne. Ostatecznie metody dają wyniki zupełnie ze sobą nieskorelowane.

Możnaby myśleć, że wynik 40. iteracji zwrócony przez metodę 2. jest najmniej dokładny, a najbardziej godny zaufania jest wynik wygenerowany z użyciem arytmetyki Float64. Jest to myślenie błędne, gdyż żadna z metod nie jest w stanie dać jakkolwiek sensownego rezultatu. Wynika to między innymi z faktu, że do przechowania w sposób poprawny kwadratu danej liczby zastosowana precyzja musi być w stanie przechować aż do dwóch razy więcej cyfr znaczących niż liczba cyfr znaczących liczby podnoszonej do kwadratu. Gdy w układzie ze **sprzężeniem zwrotnym** nie jesteśmy w stanie zapewnić wystarczającej precyzji, to błędy przez nas popełniane kumulują się wraz z wyliczaniem kolejnych wartości. Proces iteracyjnego wyznaczania wartości modelu logistycznego jest numerycznie **niestabilny**, a wyniki otrzymane przy zastosowaniu rażąco niedostatecznej precyzji mogą być tak niedokładne, jak wyniki otrzymane w sposób losowy.

6 Zadanie 6

6.1 Cel zadania

Dane jest równanie rekurencyjne: $x_{n+1} := x_n^2 + c$ dla $n = 0, 1, ...$, gdzie c jest pewną stałą. Wykonać 40 iteracji tego wyrażenia dla:

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 $x_0 = 0.25$.

Zaobserwować i wyjaśnić zachowanie generowanych ciągów.

6.2 Rozwiązanie

Dla każdego z podpunktów wykonane zostało w pętli 40 iteracji w arytmetyce Float64, kod źródłowy znajduje się w pliku *zadanie6.jl*. Wyniki wywołania progamu zostały przedstawione w tabelach oraz dodatkowo zaprezentowane w postaci iteracji graficznej z użyciem bezpłatnego kalkulatora graficznego Desmos.

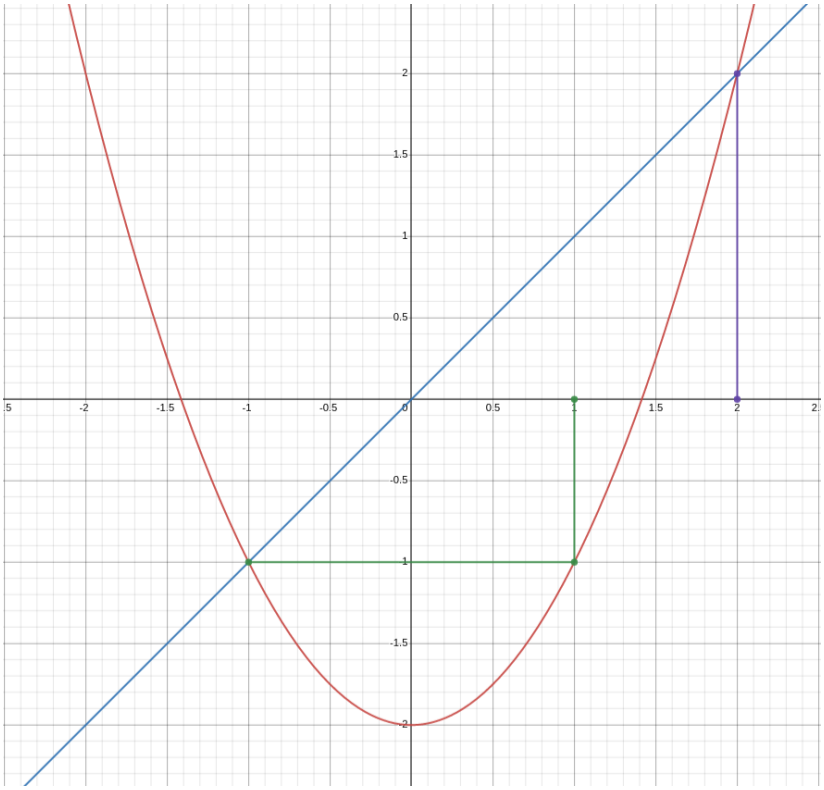
6.3 Wyniki

n	$x_0 = 1$	$x_0 = 2$	$x_0 = 1.99999999999999$
1	-1.0	2.0	1.99999999999996
2	-1.0	2.0	1.9999999999998401
3	-1.0	2.0	1.9999999999993605
4	-1.0	2.0	1.999999999997442
5	-1.0	2.0	1.9999999999897682
6	-1.0	2.0	1.9999999999590727
7	-1.0	2.0	1.999999999836291
8	-1.0	2.0	1.9999999993451638
9	-1.0	2.0	1.9999999973806553
10	-1.0	2.0	1.999999989522621
11	-1.0	2.0	1.9999999580904841
12	-1.0	2.0	1.9999998323619383
13	-1.0	2.0	1.9999993294477814
14	-1.0	2.0	1.9999973177915749
15	-1.0	2.0	1.9999892711734937
16	-1.0	2.0	1.9999570848090826
17	-1.0	2.0	1.999828341078044
18	-1.0	2.0	1.9993133937789613
19	-1.0	2.0	1.9972540465439481
20	-1.0	2.0	1.9890237264361752
21	-1.0	2.0	1.9562153843260486
22	-1.0	2.0	1.82677862987391
23	-1.0	2.0	1.3371201625639997
24	-1.0	2.0	-0.21210967086482313
25	-1.0	2.0	-1.9550094875256163
26	-1.0	2.0	1.822062096315173
27	-1.0	2.0	1.319910282828443
28	-1.0	2.0	-0.2578368452837396
29	-1.0	2.0	-1.9335201612141288
30	-1.0	2.0	1.7385002138215109
31	-1.0	2.0	1.0223829934574389
32	-1.0	2.0	-0.9547330146890065
33	-1.0	2.0	-1.0884848706628412
34	-1.0	2.0	-0.8152006863380978
35	-1.0	2.0	-1.3354478409938944
36	-1.0	2.0	-0.21657906398474625
37	-1.0	2.0	-1.953093509043491
38	-1.0	2.0	1.8145742550678174
39	-1.0	2.0	1.2926797271549244
40	-1.0	2.0	-0.3289791230026702

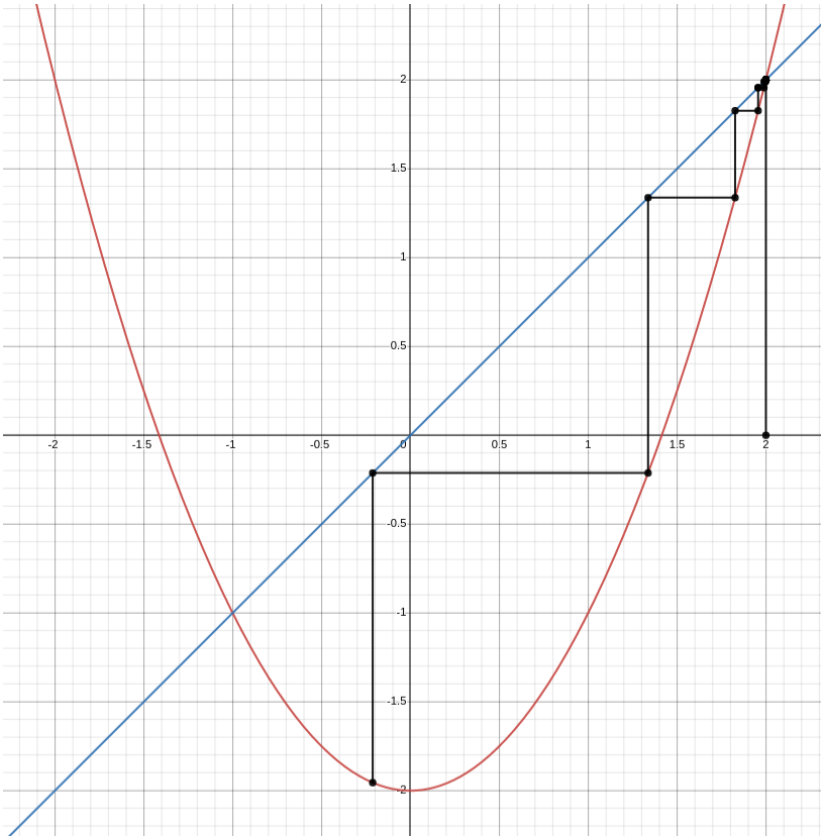
Tabela 8: Wartości x_n pierwszych 40 iteracji równania rekurencyjnego $x_{n+1} := x_n^2 + c$ dla $c = -2$ oraz różnych wartości początkowych x_0.

n	$x_0 = 1$	$x_0 = -1$	$x_0 = 0.75$	$x_0 = 0.25$
1	0.0	0.0	-0.4375	-0.9375
2	-1.0	-1.0	-0.80859375	-0.12109375
3	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	0.0	0.0	-0.2253147218564956	-0.9991524699951226
6	-1.0	-1.0	-0.9492332761147301	-0.0016943417026455965
7	0.0	0.0	-0.0989561875164966	-0.9999971292061947
8	-1.0	-1.0	-0.9902076729521999	-5.741579369278327e-6
9	0.0	0.0	-0.01948876442658909	-0.9999999999670343
10	-1.0	-1.0	-0.999620188061125	-6.593148249578462e-11
11	0.0	0.0	-0.0007594796206411569	-1.0
12	-1.0	-1.0	-0.9999994231907058	0.0
13	0.0	0.0	-1.1536182557003727e-6	-1.0
14	-1.0	-1.0	-0.9999999999986692	0.0
15	0.0	0.0	-2.6616486792363503e-12	-1.0
16	-1.0	-1.0	-1.0	0.0
17	0.0	0.0	0.0	-1.0
18	-1.0	-1.0	-1.0	0.0
19	0.0	0.0	0.0	-1.0
20	-1.0	-1.0	-1.0	0.0
21	0.0	0.0	0.0	-1.0
22	-1.0	-1.0	-1.0	0.0
23	0.0	0.0	0.0	-1.0
24	-1.0	-1.0	-1.0	0.0
25	0.0	0.0	0.0	-1.0
26	-1.0	-1.0	-1.0	0.0
27	0.0	0.0	0.0	-1.0
28	-1.0	-1.0	-1.0	0.0
29	0.0	0.0	0.0	-1.0
30	-1.0	-1.0	-1.0	0.0
31	0.0	0.0	0.0	-1.0
32	-1.0	-1.0	-1.0	0.0
33	0.0	0.0	0.0	-1.0
34	-1.0	-1.0	-1.0	0.0
35	0.0	0.0	0.0	-1.0
36	-1.0	-1.0	-1.0	0.0
37	0.0	0.0	0.0	-1.0
38	-1.0	-1.0	-1.0	0.0
39	0.0	0.0	0.0	-1.0
40	-1.0	-1.0	-1.0	0.0

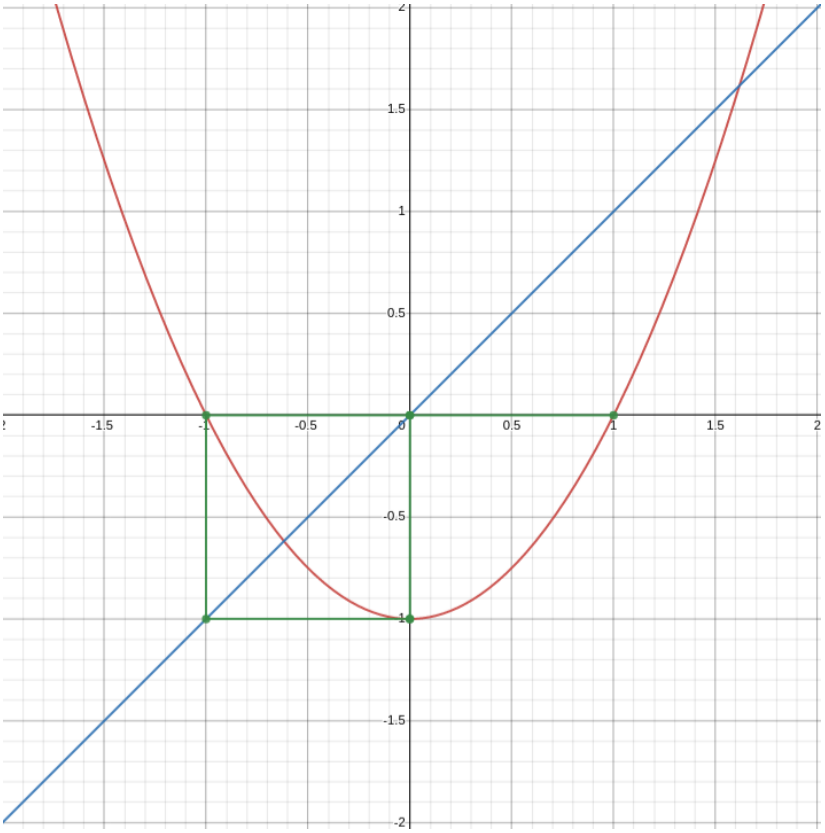
Tabela 9: Wartości x_n pierwszych 40 iteracji równania rekurencyjnego $x_{n+1} := x_n^2 + c$ dla $c = -1$ oraz różnych wartości początkowych x_0.



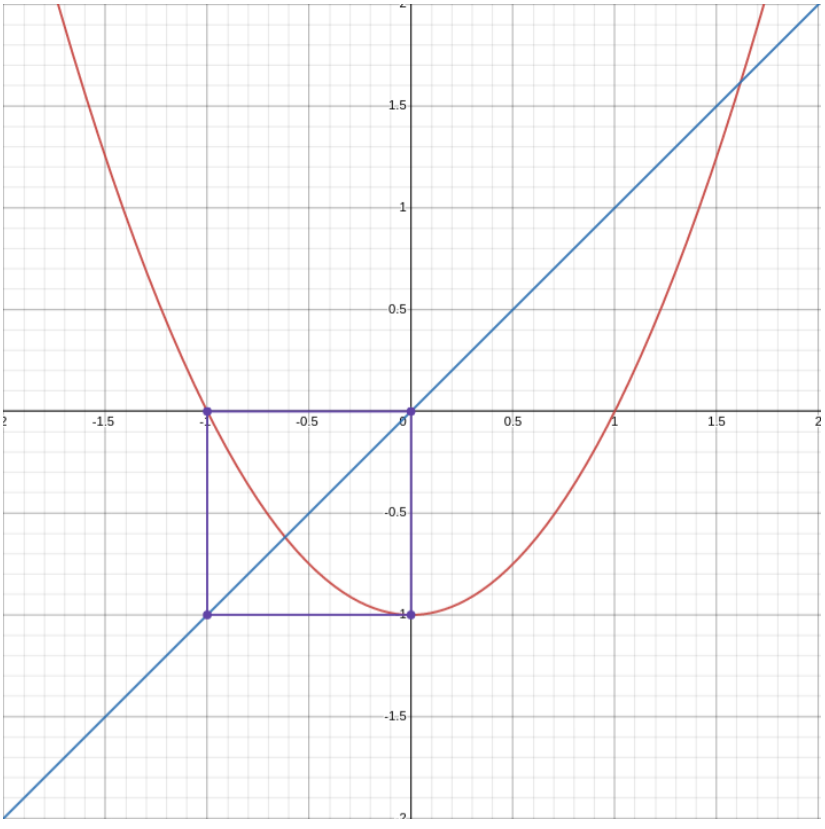
Rysunek 3: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 2$ dla $x_0 = 1$ (zielona łamana) i $x_0 = 2$ (fioletowa łamana).



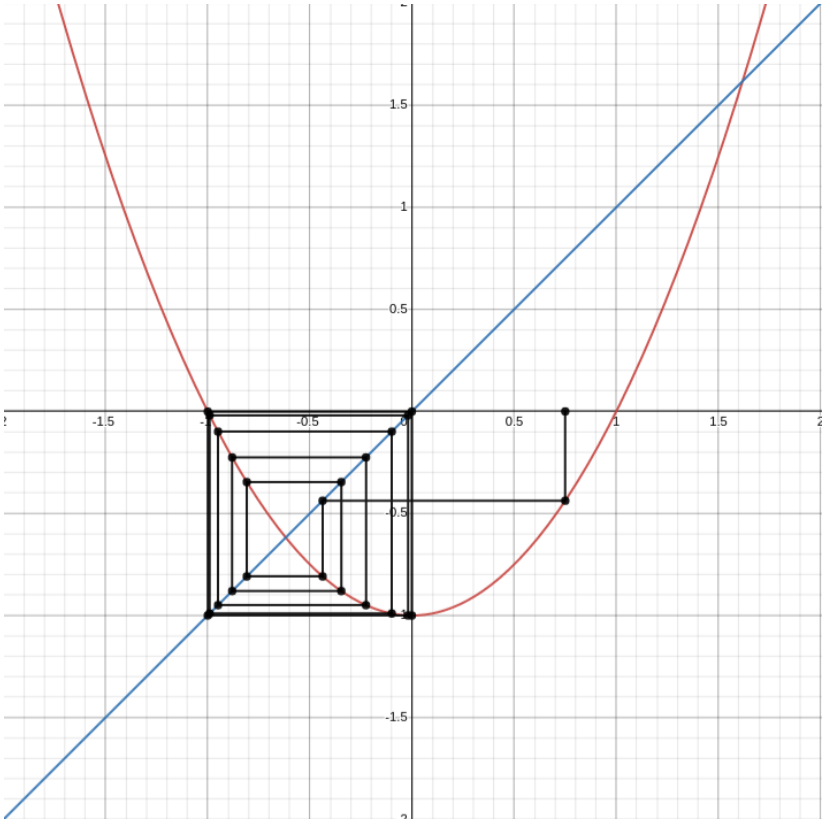
Rysunek 4: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 2$ dla $x_0 = 1.99999999999999$ (czarna łamana, 25 pierwszych iteracji).



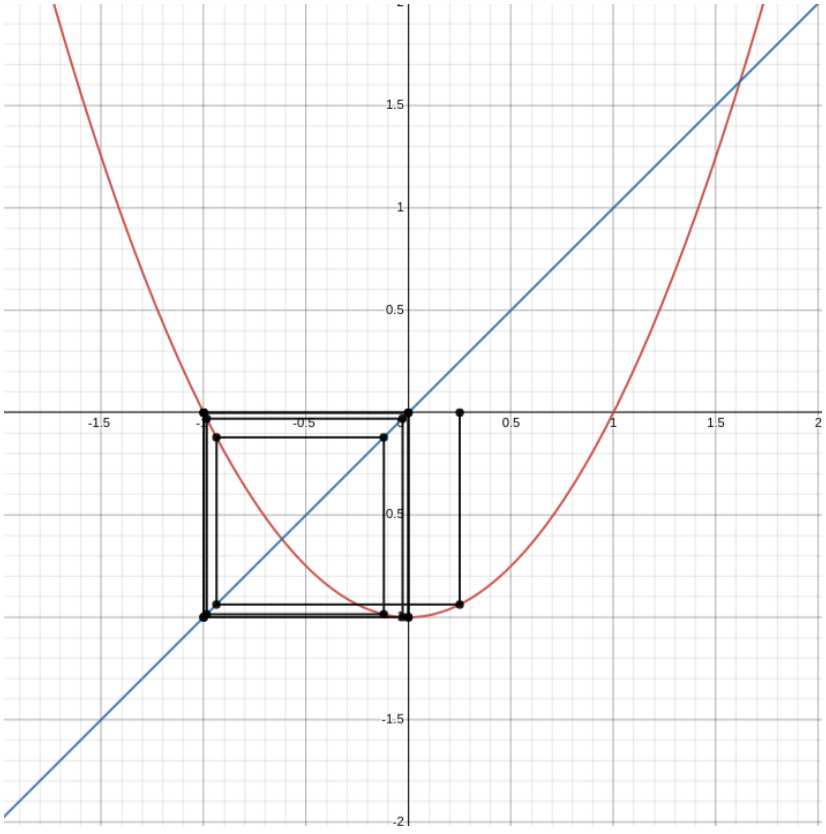
Rysunek 5: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 1$ dla $x_0 = 1$ (zielona łamana).



Rysunek 6: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 1$ dla $x_0 = -1$ (fioletowa łamana).



Rysunek 7: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 1$ dla $x_0 = 0.75$ (czarna łamana, 15 pierwszych iteracji).



Rysunek 8: Przedstawienie graficzne iteracyjnego wyznaczania x_n z równania rekurencyjnego $x_{n+1} := x_n^2 - 1$ dla $x_0 = 0.25$ (czarna łamana, 10 pierwszych iteracji).

6.4 Interpretacja wyników i wnioski

Na Rysunku 3. widzimy, jak dla równania rekurencyjnego $x_{n+1} := x_n^2 - 2$, gdzie $x_0 = 1$ i $x_0 = 2$, łamane iteracji graficznej kończą się dokładnie w punkcie stałym praktycznie na samym początku pracy programu. Punktu stałego nie można opuścić, zatem w Tabeli 8. widzimy, że kolejne iteracje nie mają wpływu na zmianę wartości x_i. Zatem iteracje graficzne dla tych wartości prowadzą do **zachowania stabilnego**.

Dla niewiele różniącego się od powyższego $x_0 = 1.99999999999999$ iteracja graficzna nie stabilizuje się dla zadanej w poleceniu liczby 40 iteracji (patrz Tabela 8. i pierwsze 25 iteracji na Rysunku 4.), zdaje się coraz bardziej odbiegać od punktu stałego $\alpha = 2$.

Następnie rozważamy równanie rekurencyjne $x_{n+1} := x_n^2 - 1$. Tutaj można zauważyć w Tabeli 9., że dla każdej z zastosowanych w doświadczeniu wartości x_0, iteracja graficzna ostatecznie stabilizuje się, wchodząc w cykl $(0, -1, 0, -1, ...)$. W przypadku wartości początkowych $x_0 = -1$ i $x_0 = 1$ cykl następuje natychmiastowo, wraz z pierwszą iteracją (Rysunki 5. i 6.). Dla $x_0 = 0.75$ i $x_0 = 0.25$ proces potrzebuje większej liczby iteracji do wejścia w cykl: w przypadku $x_0 = 0.75$ potrzeba 16 iteracji, a gdy $x_0 = 0.25$ potrzeba ich 11.

Równanie $x_{n+1} := x_n^2 + c$ wyliczane w sposób iteracyjny jest de facto **układem sprzężenia zwrotnego**. Analiza procesu opisanego takim układem jest trudna, ponieważ stany stabilne i niestabilne przeplatają się ze sobą w niełatwy do przewidzenia sposób. Zachowanie układu może się znacząco różnić w zależności jedynie od ustawienia parametrów początkowych, co możemy nazwać **czułą zależnością od warunków początkowych**. Jest to jedna z podstawowych własności składających się na pojęcie **chaosu deterministycznego**[1].

[1] Heinz-Otto Peitgen, Dietmar Saupe, *Granice chaosu. Fraktale. Część 1*