Sprawozdanie 2 Obliczenia Naukowe

Piotr Zapała

Novemaber 2022

Spis treści

1 Zadanie 1

1.1 Opis problemu

W zadaniu pierwszym jesteśmy proszeni, aby powtórz eksperyment z pierwszej listy laboratoryjnej. Różnica polega na małej modyfikacji danych, należało usuniąć ostatniej 9 z x_4 oraz ostatniej 7 z x_5. W ramach przypomnienia, w zadaniu piątym z pierwszej listy mieliśmy zaimplementować, na cztery różne sposoby, algorytm obliczania iloczynu skalarnego wektorów, a następnie należało obliczyć ten iloczyn dla następującej pary wektorów:

$$x = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$$

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

(a) "w przód" $\sum_{i=1}^{n} x_i y_i$, tj. algorytm

$S := 0$
for i:=1 **to** n **do**
    $S := S + x_i * y_i$
end for

(b) "w tył" $\sum_{i=n}^{1} x_i y_i$, tj. algorytm

$S := 0$
for i:=n **downto** 1 **do**
    $S := S + x_i * y_i$
end for

(c) dodatnie dodajemy w porządku od największego do najmniejszego, a ujemne w porządku od najmniejszego do największego, następnie dodajemy do siebie obliczone sumy częściowe.
(d) przeciwnie do metody (c).

1.2 Rozwiązanie

Rozwiązanie polega na zaimplementowaniu podanych algorytmów, a następnie obliczeniu iloczynu skalarnego dla każdego z osobna.

1.3 Wyniki

type	Float32	Float64
example1	-0.49999443f0	-5.0134667188046684e-5
example2	-0.4543457f0	-5.013464760850184e-5
example3	-0.5f0	-5.013449117541313e-5
example4	-0.5f0	-5.013495683670044e-5

Tabela 1: iloczyn skalarny.

type	Float32	Float64
example1	-0.49999443f0	-0.004346477509481865
example2	-0.4543457f0	-0.0043464774898893666
example3	-0.5f0	-0.004346477333456278
example4	-0.5f0	–0.004346477799117565

Tabela 2: iloczyn skalarny z modyfikacją.

1.4 Wnioski

Analizując powyższe przykłady możemy zauważyć, iż wartości wyliczonych iloczynów skalarnych dla arytmetyki `Float32` nie uległy zmianie. Natomiast sytuacja w przypadku `Float64` wygląda zgoła inaczej, na co jednoznacznie wskazują wyniki. Zatem w przypadku arytmetyki o większej precyzji, mamy widoczny wpływ naszej na pozór nieznaczącej modyfikacji. W arytmetyce `Float32` ze względu na mniejszy zakres mantysy, wprowadzone zmiany nie wpływają na wynik końcowy. `Float64` jest mniej pobłażliwy pod tym względem i początkowa modyfikacja pozostaje w trakcie obliczeń w zakresie jej mantysy. Możemy stąd wywnioskować, iż zadanie to jest źle uwarunkowane i bardzo ciężko oczekiwać poprawnych wyników w naszym przypadku.

2 Zadanie 2

2.1 Opis problemu

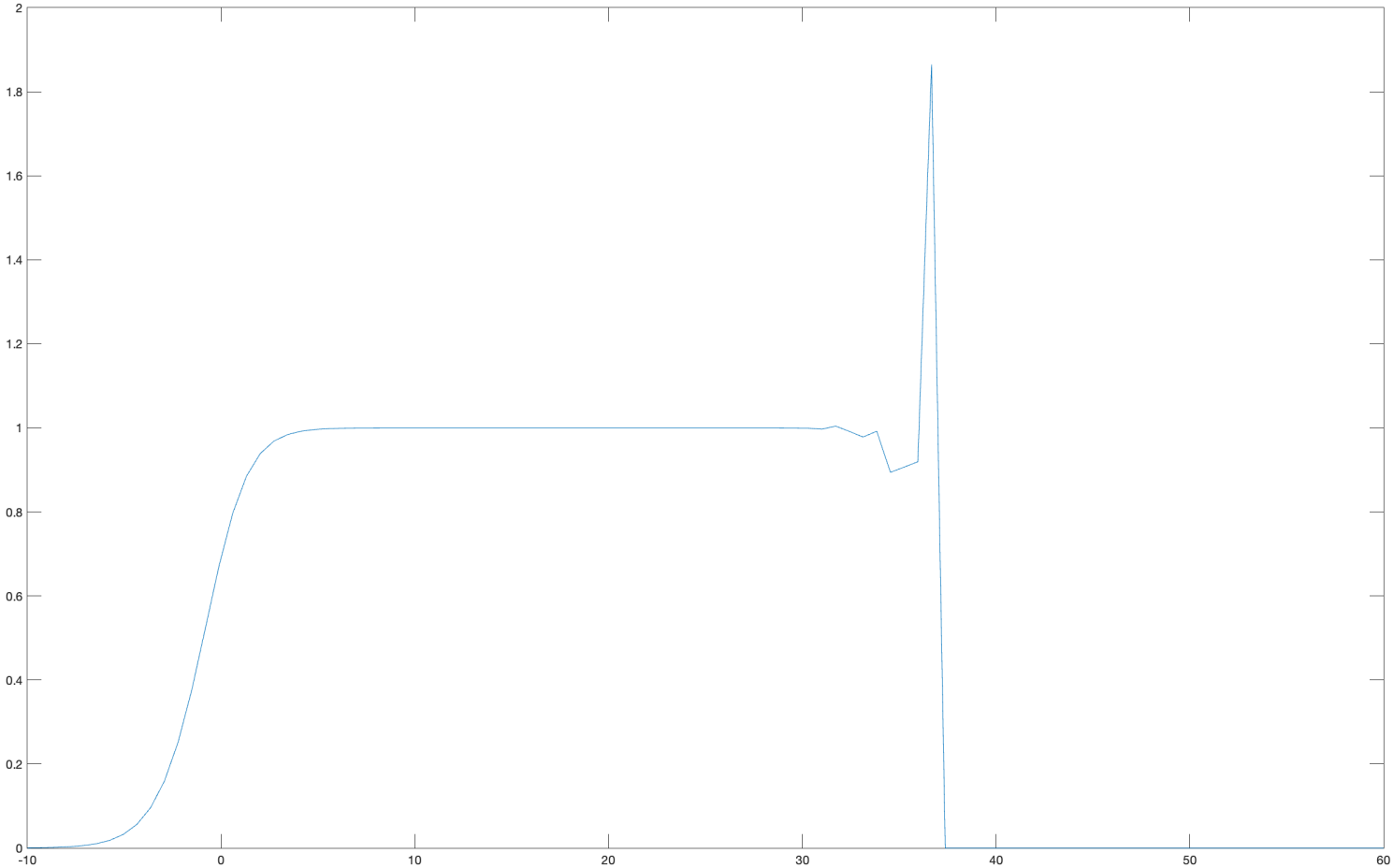
W zadaniu drugim jesteśmy proszeni o narysowanie wykresu funkcji $f(x) = e^x * ln(1 + e^{-x})$ w co najmniej dwóch programach. Następnie należy policzyć granicę $\lim_{x\to\infty} e^x * \ln(1 + e^{-x})$, a otrzymany wynik skonfrontować z naszymi wykresami.

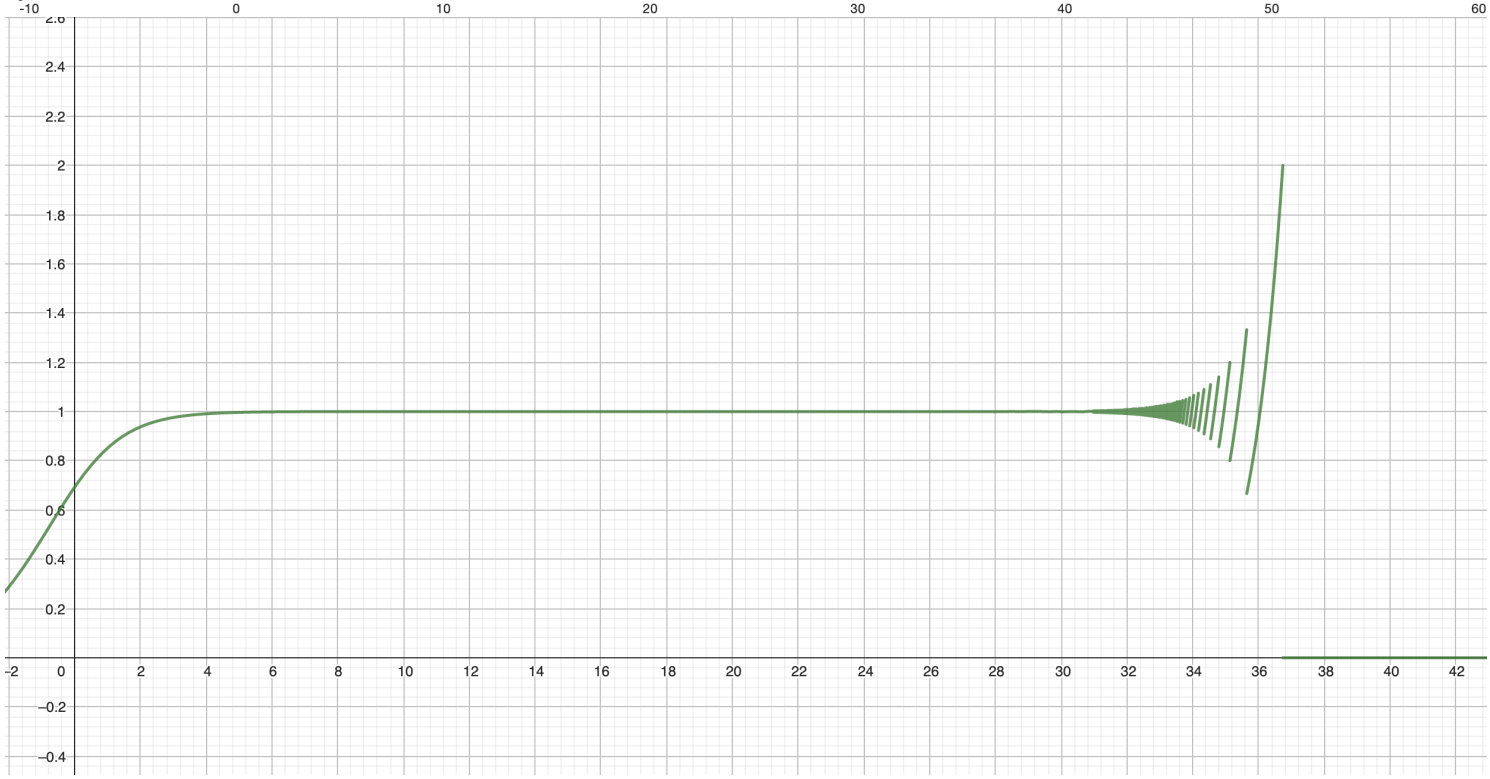
2.2 Rozwiązanie

Do narysowania wykresów posłużyłem się `MATLAB'em` oraz `GeoGebrą`, granice obliczyłem przy pomocy `Wolfram Alpha`.

2.3 Wyniki

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





2.4 Wnioski

Przyglądając się powyższym wykresom, możemy dostrzec, iż dla wartości przekraczających 30 nasze wykresy prezentują niespodziewane perturbacje. Zgodnie z wyznaczoną granicą oczekiwalibyśmy, że od

pewnego miejsca wartości zaczną zbiegać do 1. Natomiast w naszym przypadku, następuje nagły spadek do 0, poprzedzony chwilową oscylacją naszych wartości wokół 1. Problem najpewniej wynika ze zbyt małej wartości wykładnika, gdyż następuje wtedy dzielenie jedynki przez bardzo duży składnik znajdujący się w mianowniku.

3 Zadanie 3

3.1 Opis problemu

Zadanie trzecie polega na rozwiązaniu układu równań liniowych **Ax=b**, dla danej macierzy współczynników $\mathbf{A} \in R^{n \times n}$ i wektora prawych stron $\mathbf{b} \in R^n$. Do tego celu należy użyć następujących algorytmów: eliminacji Gaussa **x=A/b** oraz x = A^{-1}b (x=inv(A)*b). Eksperymenty wykonujemy dla dwóch typów macierzy, Hilberta H_n dla $n > 1$ w moim przypadku ($n = 2, 4, 6 \ldots, 36$) oraz macierzy losowej R_n, dla $n = 5, 10, 20$ z rosnącym wskaźnikiem uwarunkowania $c = 1, 10, 10^3, 10^7, 10^{12}, 10^{16}$. Następnie policzyć błędy względne $\frac{\|x-\tilde{x}\|}{\|x\|}$, gdzie $x = (1, \ldots, 1)^T$. Przykładowa macierz Hilberta 3×3.

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

3.2 Rozwiązanie

Podczas rozwiązywania zadania należało skorzystać z funkcji **A=hilb(n)**, która zwraca macierz Hilberta o zadanym **n** oraz **A=matcond(n,c)** zwracającą losową macierz z danym wskaźnikiem uwarunkowania. Algorytm eliminacji Gaussa jest częścią języka `Julia`, a jego wywołanie jest równoważne następującej operacji **x=A/b**. Aby rozwiązać układ równań drugim sposobem używamy funkcji **inv(x)**.

3.3 Wyniki

size	rank	cond	gaussian error	invert error
2	2	19.281470067903967	5.661048867003676e-16	1.1240151438116956e-15
4	4	15513.738738929662	3.587613383388527e-13	2.4396188501254767e-13
6	6	1.4951058641931808e7	2.6132877364524203e-10	2.556358881528184e-10
8	8	1.5257576052786306e10	3.7551147709171007e-7	4.029793620039031e-7
10	10	1.602489743907797e13	0.00020726510768035548	0.0002485713513251291
12	11	1.6360718665566702e16	0.04676893017350922	0.17615857301849733
14	11	2.4325396487256118e17	3.0194351169450253	3.4642402117856057
16	12	6.019326309924319e17	4.586582236304648	5.860603956215107
18	12	2.2287877143812815e18	4.480862554024397	5.339000330241911
20	13	1.0843011915344271e18	12.655226165432483	36.458539959836386
22	13	9.63324344620851e18	50.98332587273586	41.1754108459008
24	13	7.765483100170821e18	17.042652561190312	42.752046249600056
26	14	4.872032287682753e18	12.366008990359907	18.455425454123958
28	14	5.969866145859385e18	91.81593379606892	80.87151296965483
30	14	3.8417252904919854e18	22.757911310802164	31.231681742390187
32	14	7.226010642658811e18	91.18468026004645	97.01384267591234
34	14	7.877864065765033e18	16.318780330167282	18.52148018667523
36	15	1.1984500558524019e19	31.4344690401547	44.67185207319411

Tabela 3: Hilbert

size	rank	cond	gaussian error	invert error
5	5	1.0000000000000007	1.7901808365247238e-16	1.4043333874306804e-16
5	5	9.999999999999995	4.864753555590494e-16	5.438959822042073e-16
5	5	1000.0000000000095	4.0296660774695415e-15	3.7224769761911e-15
5	5	9.999999988980627e6	8.733377064403182e-11	4.8241189717112e-11
5	5	9.999816990527867e11	1.1133476564496114e-5	1.849016181079592e-5
5	4	1.5951433239650136e16	0.21622357302232126	0.2711884395501898
10	10	1.0000000000000007	3.3857251850959236e-16	3.1597501217190306e-16
10	10	9.999999999999998	2.3551386880256624e-16	2.2752801345137457e-16
10	10	1000.0000000000496	3.321272584248899e-14	3.348876922871696e-14
10	10	1.0000000001124866e7	2.706753549009248e-10	2.3291364309905156e-10
10	10	1.0001159201077572e12	1.3351437040014653e-5	2.0643407181520504e-5
10	9	6.379387724567673e15	0.059063649540189334	0.029978697058601657
20	20	1.0000000000000018	5.002167713849564e-16	5.534436722516086e-16
20	20	10.000000000000007	7.69985892130607e-16	4.1910000110727263e-16
20	20	999.9999999999699	2.553266764881752e-14	2.6184651813276296e-14
20	20	1.0000000002053203e7	2.2513786794432707e-10	2.0531536109299356e-10
20	20	1.0000836964768766e12	1.9390807211897133e-5	1.5496514406250566e-5
20	19	5.81750942337733e15	0.04332188790837964	0.17286837675040306

Tabela 4: Random

3.4 Wnioski

Pierwszym nasuwającym się wnioskiem jest fakt, że nawet dla małych rozmiarów macierze Hilberta, mają bardzo duże wskaźniki uwarunkowania. Kolejną rzeczą jest to, iż w przypadku macierzy Hilberta widzimy znaczącą różnice w wartościach błędów względnych, względem macierzy generowanej losowo. W przypadku macierzy generowanej losowo, nie jesteśmy w stanie stwierdzić wyższości jednego algorytmu nad drugim. Dla Hilberta sytuacja wygląda zgoła inaczej, gdyż algorytm eliminacji Gaussa daje wyniki bliższe temu rzeczywistemu. Zatem nasuwającym się wnioskiem jest to, że rozwiązywanie układu równań dla macierzy Hilberta jest zadaniem źle uwarunkowanym.

4 Zadanie 4

4.1 Opis problemu

W zadaniu czwartym należy przy użyciu funkcji `roots` obliczyć 20 miejsc zerowym wielomianu P.

$$P(x) = x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16}$$
$$-1672280820x^{15} + 40171771630x^{14} - 756111184500x^{13}$$
$$+11310276995381x^{12} - 135585182899530x^{11}$$
$$+1307535010540395x^{10} - 10142299865511450x^{9}$$
$$+63030812099294896x^{8} - 311333643161390640x^{7}$$
$$+1206647803780373360x^{6} - 3599979517947607200x^{5}$$
$$+8037811822645051776x^{4} - 12870931245150988800x^{3}$$
$$+13803759753640704000x^{2} - 8752948036761600000x$$
$$+2432902008176640000$$

P jest postacią naturalną wielomianu Wilkinsona p.

$$p(x) = (x-20)(x-19)(x-18)(x-17)(x-16)$$
$$(x-15)(x-14)(x-13)(x-12)(x-11)$$
$$(x-10)(x-9)(x-8)(x-7)(x-6)$$
$$(x-5)(x-4)(x-3)(x-2)(x-1)$$

Następnie należy sprawdzić wyznaczone pierwiastki z_k, obliczając $|P(z_k)|$, $|p(z_k)|$ i $|z_k - k|$, gdzie $k \in [1, \ldots, 20]$. Drugim poleceniem jest to, aby powtórz eksperyment, ale z modyfikacją współczynnika -210 na $-210 - 2^{-23}$.

4.2 Rozwiązanie

Do rozwiązania zadania posłużyłem się trzema funkcjami z pakietu `Polynomials`. Pierwszą jest **polynomials(coefficients)** tworząca równanie w postaci ogólnej z podanych współczynników, **fromroots(1:20)** zwracająca wielomian na podstawie jego miejsc zerowych oraz **roots(polynomial)**, która oblicza pierwiastki podanego wielomianu.

4.3 Wyniki

$P(z_k)$	$p(z_k)$	$\|z_k - k\|$
1603.9573920179469	2627.95739201793	1.6653345369377348e-14
7032.126244053075	9351.873755953971	8.602007994795713e-13
363020.53660426877	280076.5365949661	3.3646685437815904e-10
4.1297007568024816e6	3.867556758775393e6	3.0104239989725556e-8
3.075695036659063e7	3.0116950254288312e7	8.773618791479976e-7
1.4257935012621844e8	1.4125224900969258e8	1.3036540314814715e-5
5.24632388586803e8	5.2217372324751204e8	0.00011769796309923919
1.7358866242071867e9	1.7316932035664785e9	0.0007083981118194416
5.1002248626897955e9	5.093502113334706e9	0.003039097772564503
1.166611991635824e10	1.165589915888346 3e10	0.009425364817383652
3.127208130563727e10	3.125699514215297e10	0.02299817354506395
6.1418141120587135e10	6.1397123899342445e10	0.040566592069646745
1.5300549815452048e11	1.5297585245239474e11	0.05950452401079254
3.0985557081238544e11	3.0981676989206445e11	0.06480991092200128
5.010255098229966e11	5.009726725103758e11	0.05398301833971253
9.394123675655612e11	9.393457079007137e11	0.03609096183973115
2.3971581253245e12	2.39707235924346e12	0.0165393639009217
5.670957056381105e12	5.67084966955888e12	0.005602369314754441
7.741019369389632e12	7.740885904434366e12	0.0011451039164107613
1.2178388919649346e13	1.2178225081520236e13	0.00011119342792653697

Tabela 5: -210

$P(z_k)$	$p(z_k)$	$\|z_k - k\|$
10130.71275321466	10130.712753214342	7.749356711883593e-14
56308.28771988236	72692.28772015421	8.29603052920902e-12
807092.163108728	631988.1628406073	8.906004822506475e-10
8.084860855411538e6	7.298429054000744e6	5.614410936161107e-8
3.879198510577002e7	3.4091787996578604e7	1.0868296920207854e-6
1.4732257627258718e8	6.416108188442689e7	5.193150526494605e-6
4.1411759028405327e8	9.853716087511169e8	0.0002283049351108346
5.02184047848722e8	1.676272204696169e10	0.006980931819212444
1.2260299344962182e9	1.3459810989298729e11	0.08227581314042176
1.711412152637466e9	1.483751062525329e12	0.6505270479399019
1.711412152637466e9	1.483751062525329e12	1.1105047852610384
1.922343284948516e10	3.2961283180123582e13	1.665389446835571
1.922343284948516e10	3.2961283180123582e13	2.0460081179778995
2.78104055609644e11	9.547548928471629e14	2.518885830105916
2.78104055609644e11	9.547548928471629e14	2.712916579182928
1.0504983293359739e12	2.7422169135681444e16	2.9060058491454366
1.0504983293359739e12	2.7422169135681444e16	2.8254814771679904
6.884949792026641e12	4.252502050102871e17	2.454019284846941
6.884949792026641e12	4.252502050102871e17	2.004325976483215
9.72908249032009e11	1.3743613755543311e18	0.8469082068719089

Tabela 6: $-210 - 2^{-23}$

4.4 Wnioski

Spoglądając na otrzymane wyniki widzimy, iż zarówno wyliczając wartość z postaci normalnej jak i iloczynowej, nie byliśmy w stanie uzyskać spodziewanego wyniku. Jednoznacznie wskazuje nam to, iż mamy do czynienia z bardzo "patologicznym" przypadkiem, ponieważ miejsca zerowe nie zerują naszych funkcji. Wartości naszych wielomianów przyjmują wręcz niebotyczne wartości, a fakt iż, nawet tak pozornie mała modyfikacja współczynnika z -210 na $-210 - 2^{-23}$, ma znaczący wpływ na wyniki końcowe. Sprawia, że nasuwa się oczywisty wniosek, mianowice to zadanie z pewnością jest źle uwarunkowane.

5 Zadanie 5

5.1 Opis problemu

W zadaniu piątym mamy przedstawiony model wzrostu populacji,

$$p_n := p_n + rp_n(1 - pn), n = 0, 1, \ldots,$$

r jest pewną stałą, $r(1 - p_n)$ jest czynnikiem wzrostu populacji, a p_0 jest wielkością populacji stanowiącą procent maksymalnej wielkości populacji dla danego stanu środowiska. Dla danych $p_0 = 0.01$ i $r = 3$ należy w arytmetyce `Float32` wykonać 40 iteracji wyrażenia, a następnie ponownie wykonać 40 iteracji z tą różnicą, iż po 10 iteracjach odrzucamy cyfry po trzecim miejscu po przecinku i dopiero po obcięciu wyniku prowadzimy dalsze obliczenia. Następnie należy wykonać po 40 iteracji wyrażenia w arytmetyce `Float32` i `Float64`, a następnie porównać wyniki.

5.2 Rozwiązanie

Rozwiązanie polega na obliczaniu kolejnych wartości p_{n+1} w pętli.

5.3 Wyniki

type	result
$iterations = 40$	0.25860548
$iterations = 10 + 30$	1.093568

Tabela 7: przykład a

type	result
Float32	0.25860548
Float64	0.011611238029748606

Tabela 8: przykład b

5.4 Wnioski

Wyniki zaprezentowane powyżej, jednoznacznie ukazują wpływ zastosowanej przez nas redukcji cyfr po przecinku. W naszym przypadku nastąpiło obcięcie do trzech cyfr znaczących, pomimo tego różnica w uzyskanych wynikach jest duża. Podobna sytuacja ma miejsce w przypadku wykorzystania innej arytmetyki, tylko zamiast zmniejszać precyzje, zwiększamy ją. Dzięki temu prowadząc obliczenia jesteśmy mniej narażeni na utratę cyfr, które mogą mieć realny wpływ na wynik końcowy.

6 Zadanie 6

6.1 Opis problemu

W zadaniu szóstym ponownie mamy do rozważenia równanie rekurencyjne

$$x_{n+1} := x_n^2 + c, n = 0, 1, \ldots,$$

gdzie c jest pewną daną stałą. Należy przeprowadzić szereg eksperymentów, polegających na wykonywaniu po 40 iteracji powyższego wyrażenia w arytmetyce `Float64` dla różnych wartości c oraz x_0.

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, $x_0 = 0.25$

6.2 Rozwiązanie

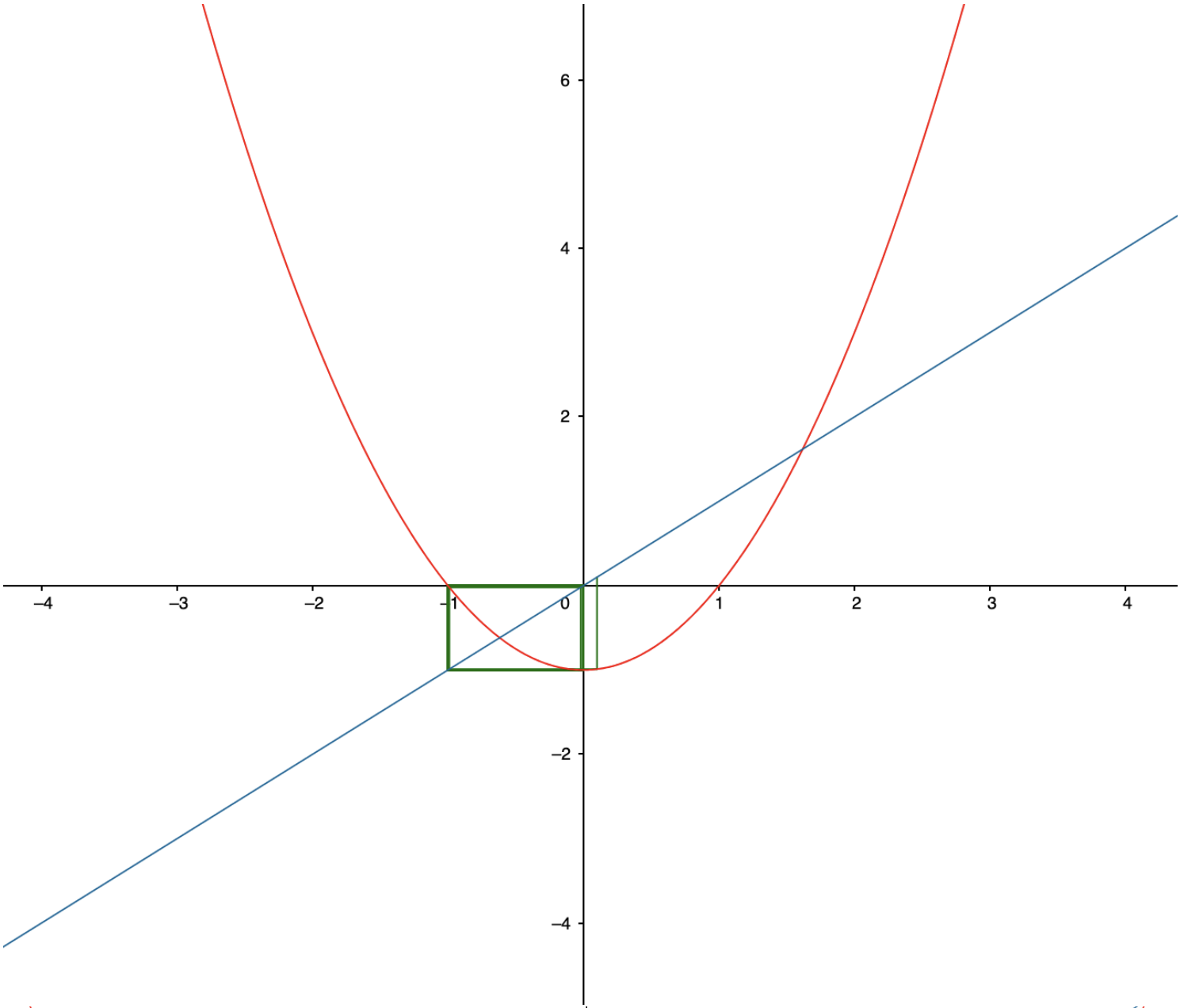
Rozwiązanie polega na obliczaniu kolejnych wartości x_{n+1} w pętli.

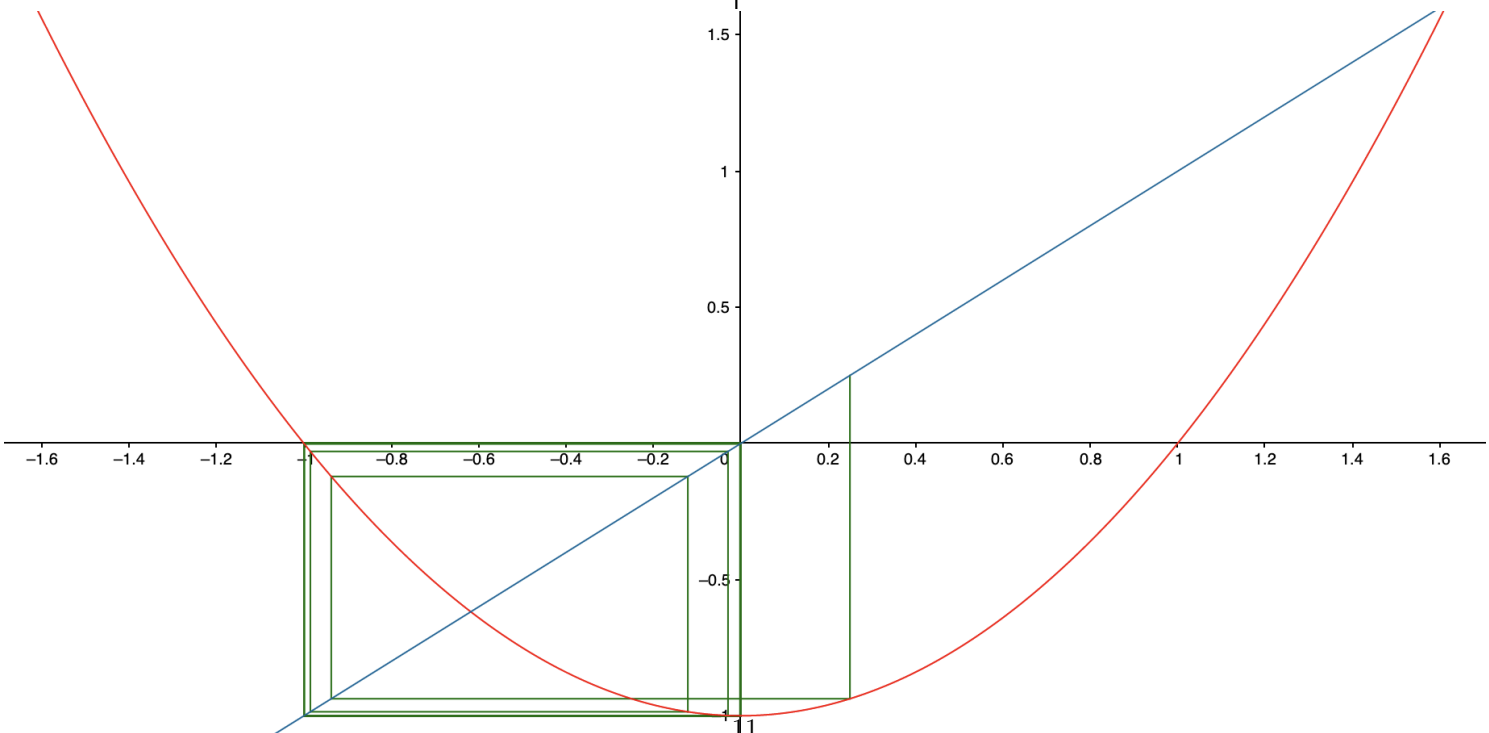
6.3 Wyniki

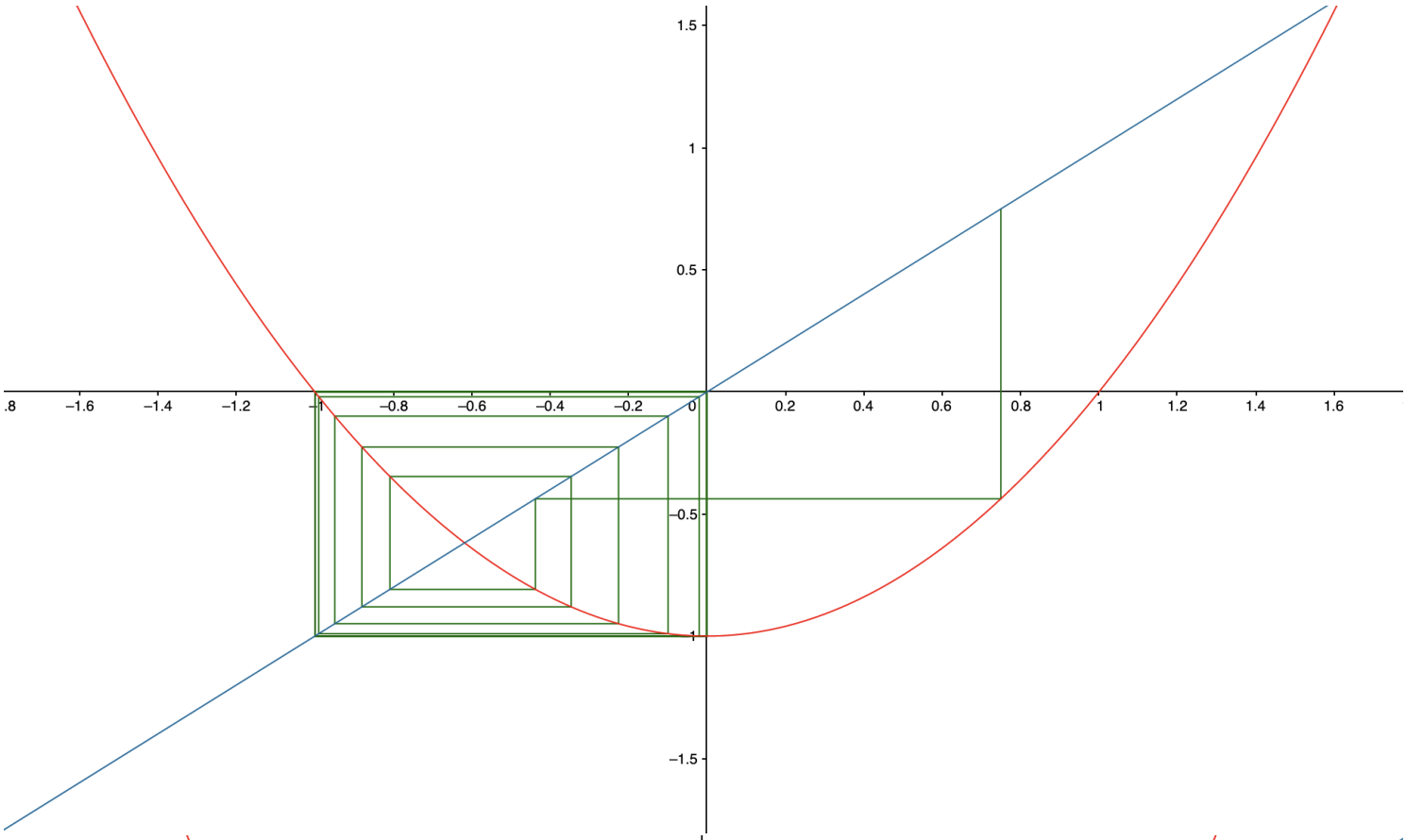
c	x_0	**result**
-2	1	-1.0
-2	2	2.0
-2	1.99999999999999	-0.3289791230026702
-1	1	-1.0
-1	1	-1.0
-1	0.75	-1.0
-1	0.25	0.0

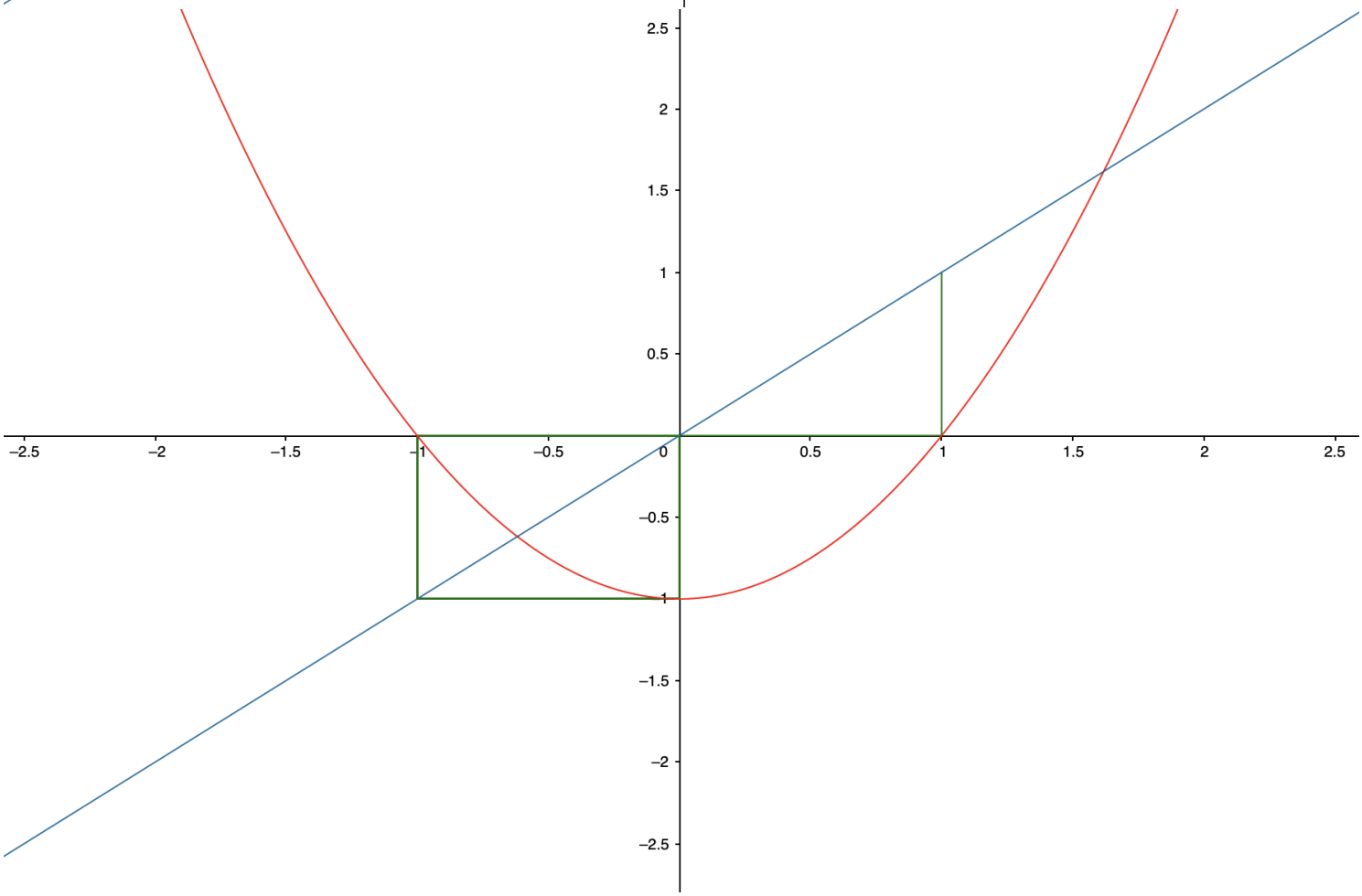
Tabela 9:

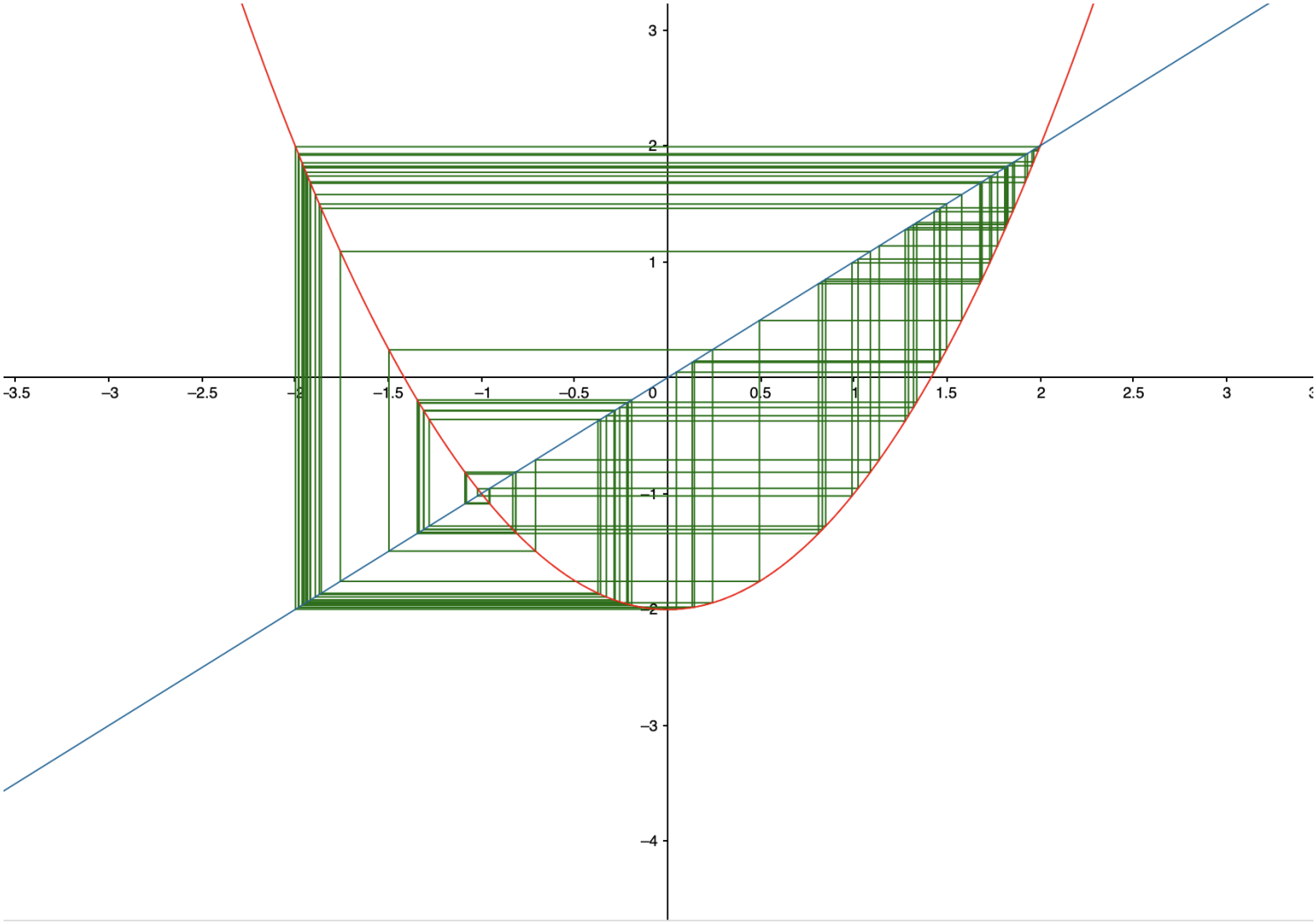
6.4 Wnioski

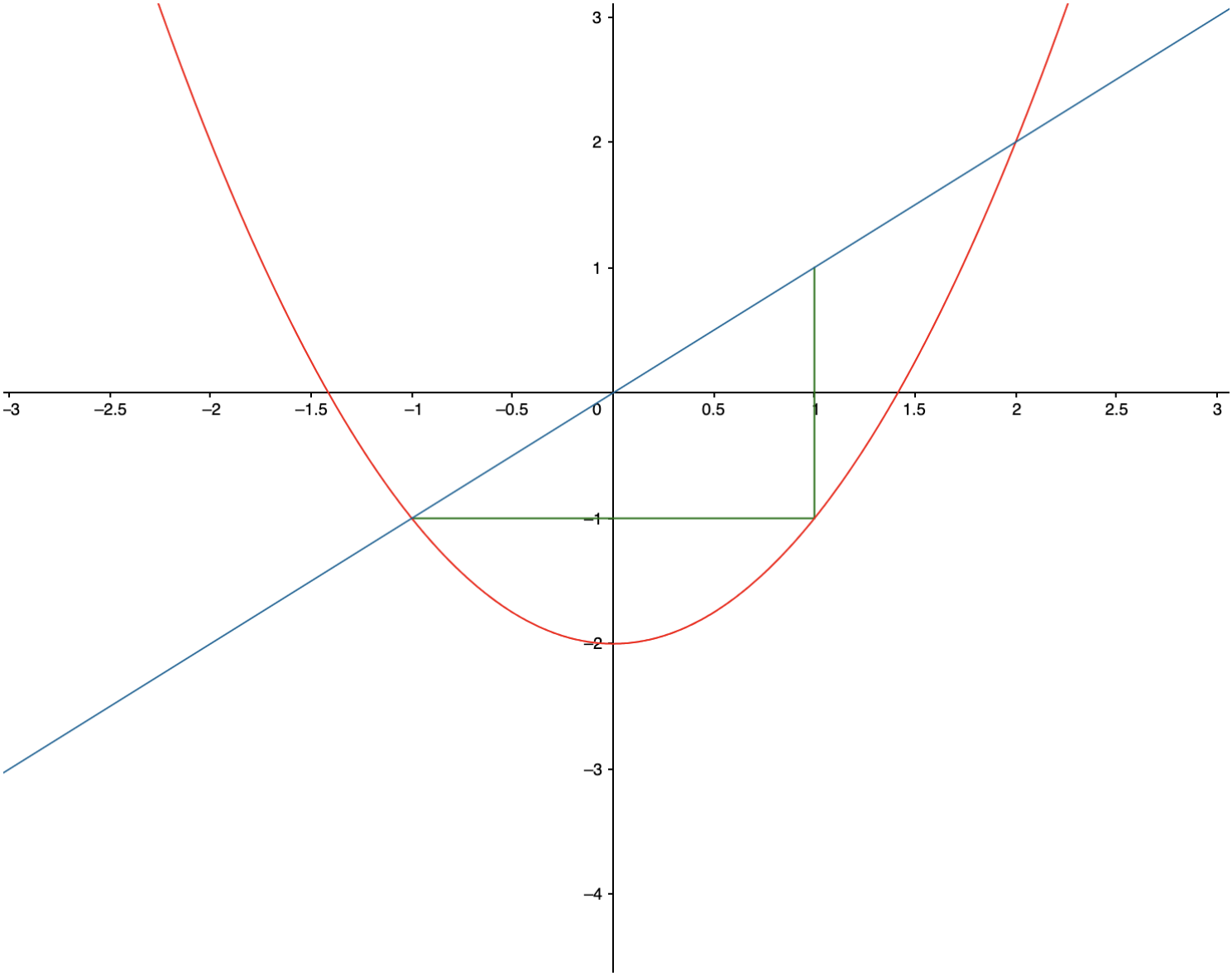

6
4
2
−2
−4
−4
−3
−2
−1
0
1
2
3
4

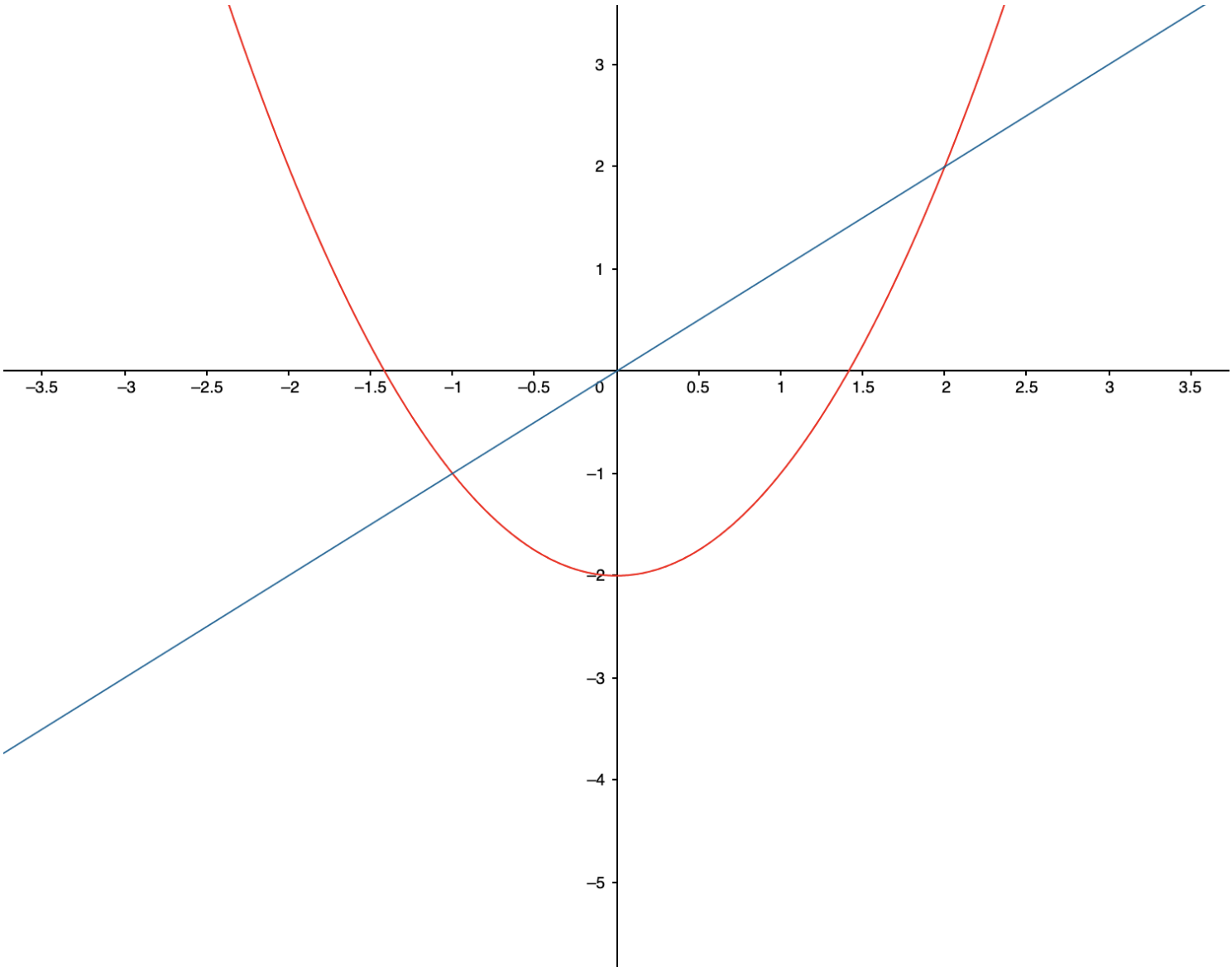

1.5
1
0.5
−0.5
−1
−1.6
−1.4
−1.2
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


1.5
1
0.5
−0.5
−1
−1.5
.8
−1.6
−1.4
−1.2
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


2.5
2
1.5
1
0.5
−0.5
−1
−1.5
−2
−2.5
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5


3
2
1
−1
−2
−3
−4
-3.5
−3
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3


3
2
1
−1
−2
−3
−4
−3
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3


3
2
1
−1
−2
−3
−4
−5
−3.5
−3
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3
3.5