Obliczenia naukowe

Lista 2

Aleksandra Czarniecka (272385)

listopad 2024

1 Analiza wpływu na wynik niewielkiej zmiany danych wejściowych: Obliczanie iloczynu skalarnego dwóch wektorów

1.1 Opis problemu

Problemem zadania jest przeanalizowanie jak niewielka zmiana danych wejściowych wpływa na otrzymane wyniki. W tym celu należało powtórzyć zadanie 5 z listy 1, ale w wektorze x w x_4 i x_5 usunąć ostatnią cyfrę (tj. 9 i 7).
Zadanie 5 z listy 1 polegało na eksperymentalnym obliczeniu iloczynu skalarnego dwóch wektorów:

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

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

za pomocą 4 różnych algorytmów używając pojedynczej i podwójnej precyzji (typy Float32 i Float64 w języku Julia). Prawidłowa wartość wynosi (dokładność do 15 cyfr) $-1.00657107000000 \cdot 2.0^{-52}$.

1.2 Rozwiązanie

Rozwiązanie polega na przeanalizowaniu wyników dla wektorów po niewielkiej zmianie:

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

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

Prawidłowa wartość, w tym przypadku wynosi $-1.00657107 \cdot 10^{-11} - 9 \cdot 10^{-10} \cdot 4773714.647 - 7 \cdot 10^{-10} \cdot 0.000185049 = -0.004296343192495245$.

1.3 Wyniki oraz ich interpretacja

W celu analizy wyników przedstawione zostaną najpierw wyniki z zadania 5 z listy 1.

Typ	"w przód"	"w tył"	od najw. do najmn.	od najmn. do najw.
Float32	-0.4999443	-0.4543457	-0.5	-0.5
Float64	1.0251881368296672e-10	-1.5643308870494366e-10	0.0	0.0

Tabela 1: Wyniki iloczynu skalarnego wektorów z zadania 5 z listy 1.

Po zmianie danych wejściowych wyniki przedstawiają się następująco.

Typ	"w przód"	"w tył"	od najw. do najmn.	od najmn. do najw.
Float32	-0.4999443	-0.4543457	-0.5	-0.5
Float64	-0.00429634273	-0.00429634299	-0.00429634284	-0.00429634284

Tabela 2: Wyniki iloczynu skalarnego wektorów po niewielkiej zmianie.

Można zauważyć, że wyniki dla typu Float32 są jednakowe, natomiast dla Float64 znacząco się różnią. W tym przypadku (Float64) wyniki po zmianie są zdecydowanie dokładniejsze niż te z Tabeli 1, i jednocześnie wszystkie bardzo zbliżone do prawidłowego wyniku.

1.4 Wnioski

Ta niewielka zmiana nie wpłynęła na wyniki w arytmetyce Float32, ponieważ usuwane cyfry z liczb znajdowały się na granicy precyzji, były to najmniej znaczace cyfry. W dalszym ciągu wyniki obliczeń obarczone są dużym błędem, ze względu na działania na liczbach o różniących się rzędach wielkości. Natomiast w arytmetyce Float64 dzięki jej większej precyzji błąd zdecydowanie zmalał. Można z tego wywnioskować, że zadanie jest źle uwarunkowane, to znaczy, że nawet najmniejszy wpływ na dane wejściowe może mieć kolosalne znaczenie w wyniku.

2 Porównanie wykresu funkcji z jej obliczoną granicą

2.1 Opis problemu

Problemem zadania jest analiza wykresów funkcji $f(x) = e^x ln(1 + e^{-x})$ narysowanych w różnych programach do wizualizacji i porównanie ich z obliczoną granicą $\lim_{x\to\infty} f(x)$.

2.2 Rozwiązanie

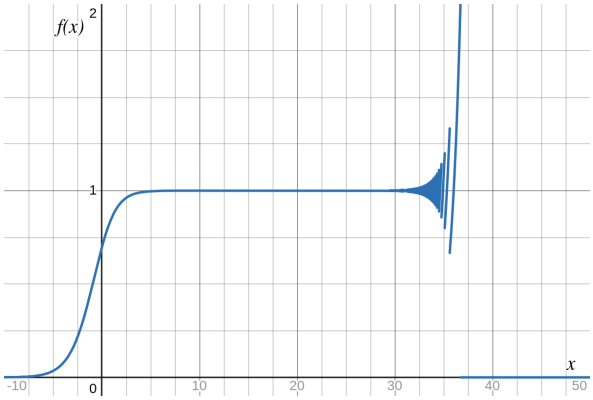
Do narysowania wykresu funkcji $f(x) = e^x ln(1 + e^{-x})$ użyte zostały dwa programy do wizualizacji, tj. Desmos i Wolfram Alpha.

Obliczenie granicy funkcji w nieskończoności:

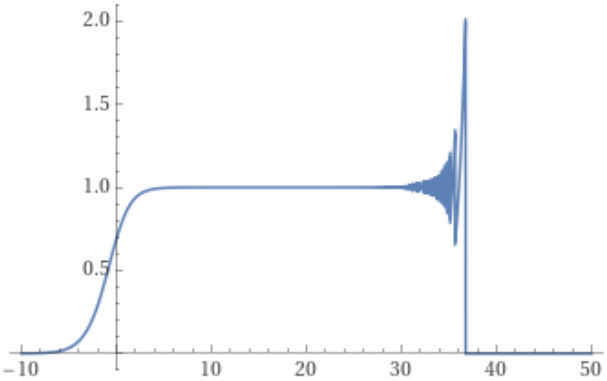
$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^x \ln(1 + e^{-x}) = \lim_{x\to\infty} \frac{\ln(1+e^{-x})}{e^{-x}} =_{\text{Hospital}} = \lim_{x\to\infty} -\frac{1}{e^x+1} \cdot \frac{1}{-e^{-x}} = \lim_{x\to\infty} \frac{e^x}{e^x+1} =$
$= \lim_{x\to\infty} \frac{e^x+1-1}{e^x+1} = \lim_{x\to\infty} \left(1 - \frac{1}{e^x+1}\right) = 1$

Zatem oczekiwane jest, aby funkcja w nieskończoności zbiegała do wartości 1.

2.3 Wyniki oraz ich interpretacja



Rysunek 1:
Wykres funkcji otrzymany za pomocą Desmos.



Rysunek 2:
Wykres funkcji otrzymany za pomocą Wolfram Alpha.

Z wykresów widać, że już dla x > 30 zaczyna się dziać coś dziwnego, funkcja zachowuje się nieprawidłowo. Najpierw widoczne są oscylacje, aż w końcu funkcja przyjmuje wartość 0, co nie pokrywa się z obliczoną granicą.

2.4 Wnioski

Wzór funkcji $f(x) = e^x ln(1 + e^{-x})$ zapisany w takiej formie sprawia, że dla odpowiednio dużego argumentu funkcji czynnik e^x staje się bardzo duży, natomiast czynnik $ln(1+e^{-x})$ wprost przeciwnie- staje się bardzo mały. Z tego powodu zaczynają pojawiać się oscylacje, ponieważ mnożenie liczb o tak różnym rzędzie generuje duży błąd. Dodatkowo ze względu arytmetykę stosowaną w użytych programach do wizualizacji od pewnego momentu $ln(1 + e^{-x}) \approx 0$, dlatego funkcja się zeruje. Problem jest źle uwarunkowany, a programy do wizualizacji niedokładne przez precyzję arytmetyki.

3 Rozwiązywanie układu równań liniowych $Ax = b$

3.1 Opis problemu

Problemem zadania jest porównanie dwóch metod rozwiązywania układów liniowych $Ax = b$. W tym celu eksperymenty zostały przeprowadzone na następujących macierzach A:

- $A = H^n$, gdzie H^n jest macierzą Hilberta z rosnącym stopniem $n > 1$;
- $A = R^n$, gdzie R^n jest losową macierzą stopnia $n \in \{5, 10, 20\}$, o współczynniku uwarunkowania $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$.

Układy równań będą rozwiązywane za pomocą następujących metod:

- metody eliminacji Gaussa, tj. $x = A\backslash b$;
- metody inwersji (z macierzą odwrotną), tj. $x = A^{-1}b$.

Rozwiązanie dokładne to $x = \{1, ..., 1\}^T$. Na jego podstawie należy obliczyć błąd względny wyników powyższych metod.

3.2 Rozwiązanie

Rozwiązanie polega na implementacji problemu w języku Julia. Do wygenerowania macierzy Hilberta użyto funkcji hilb(n) załączonej do zadania w pliku hilb.jl, natomiast do macierzy losowej użyto funkcji matcond(n, c) z pliku matcond.jl. Zaimplementowano funkcję do rozwiązywania układu równań Ax = b, gdzie prawdziwym rozwiązaniem jest wektor jedynek. Funkcja ta wyznacza $b = A \cdot \{1, ..., 1\}^T$, a następnie oblicza x metodą Gaussa i metodą inwersji. Na koniec obliczany jest błąd względny i drukowane są otrzymane wyniki. Użyte zostają tam takie funkcje wbudowane jak: cond(A) (wskaźnik uwarunkowania macierzy) i rank(A) (rząd macierzy) z pakietu LinearAlgebra.

3.3 Wyniki oraz ich interpretacja

n	cond(A)	rank(A)	błąd metody Gaussa	błąd metody inwersji
1	1.000000e+00	1	0.000000e+00	0.000000e+00
2	1.928147e+01	2	5.661049e-16	1.404333e-15
3	5.240568e+02	3	8.022594e-15	0.000000e+00
4	1.551374e+04	4	4.137410e-14	0.000000e+00
5	4.766073e+05	5	1.682843e-12	3.354436e-12
6	1.495106e+07	6	2.618913e-10	2.016376e-10
7	4.753674e+08	7	1.260687e-08	4.713280e-09
8	1.525758e+10	8	6.124090e-08	3.077484e-07
9	4.931538e+11	9	3.875163e-06	4.541268e-06
10	1.602442e+13	10	8.670390e-05	2.501493e-04
11	5.222701e+14	10	1.582781e-04	7.618304e-03
12	1.751595e+16	11	1.339621e-01	2.589941e-01
13	3.188395e+18	11	1.103970e-01	5.331276e+00

Tabela 3: Porównanie wyników dla macierzy Hilberta (dla n > 13 precyzja Float64 jest niewystarczająca).

Wskaźnik uwarunkowania macierzy dla macierzy Hilberta rośnie szybko, jest spory nawet dla małych wartości n. Obie metody obliczania rozwiązania generują duże błędy względne, które są podobnych rzędów.

c	n	rank(A)	błąd metody Gaussa	błąd metody inwersji
1	5	5	1.790181e-16	1.719950e-16
	10	10	1.755417e-16	2.482534e-16
	20	20	4.227603e-16	5.165819e-16
10	5	5	1.719950e-16	1.489520e-16
	10	10	3.349122e-16	2.719480e-16
	20	20	3.376612e-16	3.475548e-16
10^3	5	5	1.719950e-16	1.489520e-16
	10	10	2.984828e-15	5.055600e-15
	20	20	2.322274e-14	2.478877e-14
10^7	5	5	9.668178e-11	9.955042e-11
	10	10	1.908004e-10	2.438477e-10
	20	20	9.043490e-11	1.129620e-10
10^{12}	5	5	3.368904e-05	3.936900e-05
	10	10	3.217732e-16	1.651812e-06
	20	20	1.845942e-06	4.212324e-06
10^{16}	5	4	4.545471e-02	5.517881e-02
	10	9	2.268347e-01	2.293870e-01
	20	19	2.388129e-01	2.254282e-01

Tabela 4: Porównanie wyników dla macierzy losowej.

Błędy względne dla macierzy losowej także są spore w obu metodach. Zależą jednak bardziej od współczynnika uwarunkowania, niż od rozmiaru macierzy. Dla różnych wielkości macierzy, ale tych samych uwarunkowaniach, rzędy błędów są zbliżone.

3.4 Wnioski

Czym wyższa wartość wskaźnika uwarunkowania macierzy A, tym rozwiązanie równania jest obarczone większym błędem. Zadanie staje się źle uwarunkowane dla dużych wartości wskaźnika uwarunkowania. Obie metody w obu przypadkach generują zbliżone błędy, dlatego nie można ocenić, która jest lepsza.

4 Wielomiany i eksperyment Wilkinsona

4.1 Opis problemu

Problemem zadania jest złośliwy wielomian Wilkinsona $P(x) = \prod_{k=1}^{n}(x-k)$, który można zapisać w postaci:

- iloczynowej, tj. $P(x) = (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7)(x-8)(x-9)(x-10)(x-11)(x-12)(x-13)(x-14)(x-15)(x-16)(x-17)(x-18)(x-19)(x-20)$;
- normalnej, tj. $P(x) = x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} - 1672280820x^{15} + 40171771630x^{14} - 756111184500x^{13} + 11310276995381x^{12} - 135585182899530x^{11} + 1307535010540395x^{10} - 10142299865511450x^9 + 63030812099294896x^8 - 311333643161390640x^7 + 1206647803780373360x^6 - 3599979517947607200x^5 + 8037811822645051776x^4 - 12870931245150988800x^3 + 13803759753640704000x^2 - 8752948036761600000x + 2432902008176640000$.

Celem jest wyznaczenie w Julii pierwiastków z_k, gdzie $k \in [1, 20]$. Następnie sprawdzenie ich przez policzenie $|P(z_k)|, |p(z_k)|i|z_k - k|$ oraz wyjaśnienie rozbieżności.

Kolejno należy powtórzyć eksperyment Wilkinsona, tj. zmienić współczynnik -220 na $-210 - 2^{-23}$.

4.2 Rozwiązanie

Rozwiązanie opiera się na użyciu pakietu Polynomials języka Julia. Należy zapisać wielomian Wilkinsona w postaci naturalnej oraz iloczynowej (za pomocą wbudowanej funkcji fromroots()). Następnie z użyciem funkcji roots() obliczono pierwiastki z_k tego wielomianu z jego postaci normalnej. Po tym, wyznaczono błąd bezwzględny pierwiastków i wyników wielomianu dla nich. Na koniec powtórzono ekperyment przy zmianie współczynnika -210 na $-210 - 2^{-23}$.

4.3 Wyniki oraz ich interpretacja

k	$\|P(k)\|$	$\|p(k)\|$
1	0.0	0
2	8192.0	0
3	27648.0	0
4	622592.0	0
5	2.176e6	0
6	8.84736e6	0
7	2.4410624e7	0
8	5.89824e7	0
9	1.45753344e8	0
10	2.27328e8	0
11	4.79074816e8	0
12	8.75003904e8	0
13	1.483133184e9	0
14	2.457219072e9	0
15	3.905712e9	0
16	6.029312e9	0
17	9.116641408e9	0
18	1.333988352e10	0
19	1.9213101568e10	0
20	2.7193344e10	0

Tabela 5: Wyniki wielomianu Wilkinsona w postaci normalnej (P(k)) i iloczynowej (p(k)) dla dokładnych wartości pierwiastków (wartość k).

Dla dokładnych wartości pierwiastków wielomian w postaci iloczynowej jest równy zero dla każdego pierwiastka, natomiast w postaci normalnej wychodzą różne wyniki.

k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999996989	35696.50964788257	5.518479490350445e6	3.0109248427834245e-13
2	2.0000000000283182	176252.60026668405	7.37869762990174e19	2.8318236644508943e-11
3	2.9999999995920965	279157.6968824087	3.3204139316875795e20	4.0790348876384996e-10
4	3.9999999837375317	3.0271092988991085e6	8.854437035384718e20	1.626246826091915e-8
5	5.000000665769791	2.2917473756567076e7	1.8446752056545688e21	6.657697912970661e-7
6	5.999989245824773	1.2902417284205095e8	3.320394888870117e21	1.0754175226779239e-5
7	7.000102002793008	4.805112754602064e8	5.423593016891273e21	0.00010200279300764947
8	7.999355829607762	1.6379520218961136e9	8.262050140110275e21	0.0006441703922384079
9	9.002915294362053	4.877071372550003e9	1.196559421646277e22	0.002915294362052734
10	9.990413042481725	1.3638638195458128e10	1.655260133520688e22	0.009586957518274986
11	11.025022932909318	3.585631295130865e10	2.24783329792479e22	0.025022932909317674
12	11.953283253846857	7.533332360358197e10	2.886944688412679e22	0.04671674615314281
13	13.07431403244734	1.9605988124330817e11	3.807325552826988e22	0.07431403244734014
14	13.914755591802127	3.5751347823104315e11	4.612719853150334e22	0.08524440819787316
15	15.075493799699476	8.21627123645597e11	5.901011420218566e22	0.07549379969947623
16	15.946286716607972	1.5514978880494067e12	7.010874106897764e22	0.05371328339202819
17	17.025427146237412	3.694735918486229e12	8.568905825736165e22	0.025427146237412046
18	17.99092135271648	7.650109016515867e12	1.0144799361044434e23	0.009078647283519814
19	19.00190981829944	1.1435273749721195e13	1.1990376202371257e23	0.0019098182994383706
20	19.999809291236637	2.792410639368072 7e13	1.4019117414318134e23	0.00019070876336257925

Tabela 6: Porównania wartości dla wielomianu Wilkinsona (k- dokładna wartość pierwiastka, z_k- zwrócona wartość pierwiastka z funkcji roots(), $|P(z_k)|$- wartość bezwzględna wyniku wielomianu zapisanego w postaci normalnej, $|p(z_k)|$- wartość bezwzględna wyniku wielomianu zapisanego w postaci iloczynowej, $|z_k - k|$- błąd bezwzględny znalezionych pierwiastków).

Wszystkie wartośći z_k różnią się od dokładnych wartości pierwiastków wielomianu Wilkinsona (k). Można także zauważyć, że błędy bezwzględne zwiększają się wraz ze wzrostem wartości pierwiastka. Skutkują one dość niespodziewanymi wynikami wartości wielomianu zapisanego w postaci normalnej, jak i iloczynowej. Obie te wartości są od siebie różne i osiągają rzędy nawet 10^{13} i 10^{23}.

k	z_k	$\|P(z_k)\|$
1	0.9999999999998357 + 0.0im	20259.872313418207
2	2.0000000000550373 + 0.0im	346541.4137593836
3	2.99999999660342 + 0.0im	2.2580597001197007e6
4	4.000000089724362 + 0.0im	1.0542631790395478e7
5	4.99999857388791 + 0.0im	3.757830916585153e7
6	6.000020476673031 + 0.0im	1.3140943325569446e8
7	6.99960207042242 + 0.0im	3.939355874647618e8
8	8.007772029099446 + 0.0im	1.184986961371896e9
9	8.915816367932559 + 0.0im	2.2255221233077707e9
10	10.095455630535774 - 0.6449328236240688im	1.0677921232930157e10
11	10.095455630535774 + 0.6449328236240688im	1.0677921232930157e10
12	11.793890586174369 - 1.6524771364075785im	3.1401962344429485e10
13	11.793890586174369 + 1.6524771364075785im	3.1401962344429485e10
14	13.992406684487216 - 2.5188244257108443im	2.157665405951858e11
15	13.992406684487216 + 2.5188244257108443im	2.157665405951858e11
16	16.73074487979267 - 2.812624896721978im	4.850110893921027e11
17	16.73074487979267 + 2.812624896721978im	4.850110893921027e11
18	19.5024423688181 - 1.940331978642903im	4.557199223869993e12
19	19.5024423688181 + 1.940331978642903im	4.557199223869993e12
20	20.84691021519479 + 0.0im	8.756386551865696e12

Tabela 7: Porównania wartości dla wielomianu Wilkinsona po zmianie (k- dokładna wartość pierwiastka, z_k- zwrócona wartość pierwiastka z funkcji roots(), $|P(z_k)|$- wartość bezwzględna wyniku wielomianu zapisanego w postaci normalnej).

Okazało się, że po zmianie jednego z współczynników wielomianu, jego obliczone pierwiastki zyskały część urojoną. Tutaj również wartości $|P(z_k)|$ są różne od zera.

4.4 Wnioski

Zadanie jest źle uwarunkowane. Już sam problem wyznaczania pierwiastków wielomianu Wilkinsona to pokazuje, gdyż małe niedokładności ich wartości generują ogromne zmiany w wartości wielomianu, rzędu nawet 10^{13} dla postaci normalnej i 10^{23} dla postaci iloczynowej. Widać to również przy powtórzeniu eksperymentu ze zmianą jednego ze współczynników. Niewielka zmiana skutkowała się pojawieniem części urojonej i w dalszym ciągu absolutnie niedokładymi wartościami wielomianu.

Oprócz tego można wywnioskować, że przechowywanie wielomianu Wilkinsona w postaci naturalnej jest bardzo niedokładne, co widać w Tabeli 5, gdzie nawet dla dokładnych pierwiastków, wielomian się nie zeruje. Dzieje się tak przez to, że arytmetyka Float64 ma w systemie dziesiętnym w języku Julia od 15 do 17 miejsc znaczących, a część ze współczynników potrzebuje więcej. W ten sposób tracą one kilka miejsc znaczących.

5 Równanie rekurencyjne modelu logistycznego

5.1 Opis problemu

Problemem zadania jest pewnien model logistyczny, model wzrostu populacji zadany równaniem rekurencyjnym: $p_{n+1} := p_n + rp_n(1 - p_n)$, dla $n = 0, 1, ...$, gdzie r jest pewną daną stałą, $r(1 - p_n)$ jest czynnikiem wzrostu populacji, a p_0 jest wielkośćią populacji stanowiącą procent maksymalnej wielkości populacji dla danego stanu środowiska.

Celem jest przeprowadzenie trzech eksperymentów (dla każdego po 40 iteracji) dla $p_0 = 0.001$ i $r = 3$:

- w arytmetyce Float32 bez modyfikacji danych wejściowych;

- w arytmetyce Float64 bez modyfikacji danych wejściowych;
- w arytmetyce Float32, gdzie wartość p_{10} zostanie obcięta do trzech miejsc po przecinku i kolejne liczone już na tej podstawie.

5.2 Rozwiązanie

Rozwiązanie polega na implementacji funkcji obliczającej rekurencyjnie następne czterdzieści wyników równania w arytmetykach Float32 oraz Float64 bez modyfikacji, a także z ucięciem wyrazu p_{10} do trzech miejsc po przecinku w arytmetyce Float32.

5.3 Wyniki oraz ich interpretacja

Iteracja	Float32 (bez modyfikacji)	Float32 (obcięcie po 10 iteracji)	Float64 (bez modyfikacji)
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	0.722	0.722914301179573
11	1.3238364	1.3241479	1.3238419441684408
12	0.037716985	0.036488228	0.03769529725473175
13	0.14660022	0.14195873	0.14651838271355924
14	0.521926	0.5073781	0.521670621435246
15	1.2704837	1.2572148	1.2702617739350768
16	0.2395482	0.28709212	0.24035217277824272
17	0.7860428	0.90110284	0.7881011902353041
18	1.2905813	1.1684524	1.2890943027903075
19	0.16552472	0.57796663	0.17108484670194324
20	0.5799036	1.3097303	0.5965293124946907
21	1.3107498	0.09274103	1.3185755879825978
22	0.088804245	0.3451614	0.058377608259430724
23	0.3315584	1.0232364	0.22328659759944824
24	0.9964407	0.9519073	0.7435756763951792
25	1.0070806	1.0892466	1.315588346001072
26	0.9856885	0.7976118	0.07003529560277899
27	1.0280086	1.2818935	0.26542635452061003
28	0.9416294	0.1978212	0.8503519690601384
29	1.1065198	0.6738851	1.2321124623871897
30	0.7529209	1.333177	0.37414648963928676
31	1.3110139	0.0006252018	1.0766291714289444
32	0.0877831	0.0024996346	0.8291255674004515
33	0.3280148	0.009979794	1.2541546500504441
34	0.9892781	0.039620385	0.29790694147232066
35	1.021099	0.15377222	0.9253821285571046
36	0.95646656	0.5441512	1.1325322626697856
37	1.0813814	1.2883033	0.6822410727153098
38	0.81736827	0.17403738	1.3326056469620293
39	1.2652004	0.6052825	0.0029091569028512065
40	0.25860548	1.3220292	0.011611238029748606

Tabela 8: Porównanie wartości p_n wyznaczone z rekurencyjnego równania w kolejnych iteracjach (wartość n).

Początkowe wartości p_n wyznaczone z rekurencyjnego równania są do siebie zbliżone we wszystkich esperymentach, jednak czym dalsza iteracja, tym bardziej się od siebie różnią. Obcięcie p_{10} znacząco wpływa na kolejne obliczenia. Od około 19-tej iteracji wyniki Float32 z obcięciem zaczynają odstawać od pozostałych wyników. Natomiast Float32 i Float64 zaczynają się rozbiegać około iteracji 22.

5.4 Wnioski

Wszystkie uzyskane wyniki są od siebie znacząco różne, a każdy z nich prawdopodobnie zawiera istotny błąd. Przyczyną tego zjawiska jest fakt, że w obliczeniach podnosimy poprzednią wartość do kwadratu, co powoduje, że do jej zapisania wymagane jest aż dwukrotnie więcej cyfr znaczących. W efekcie błędy obliczeniowe zaczynają się kumulować. Każdy kolejny krok iteracyjny wiąże się z zaokrągleniem wartości do określonej precyzji arytmetyki, co dodatkowo pogłębia narastający błąd. W związku z tym proces iteracyjnego rozwiązywania modelu logistycznego okazuje się numerycznie niestabilny, co prowadzi do stopniowego wzrostu nieścisłości i odchylenia wyników od ich rzeczywistych wartości.

6 Eksperymenty na podstawie równania rekurencyjnego

6.1 Opis problemu

Problemem zadania jest przeprowadzenie siedmiu eksperymentów na podstawie równania rekurencyjnego:
$x_{n+1} := x_n^2 + c$ dla $n = 0, 1, ...$

W tym celu należy wykonać w języku Julia w arytmetyce Float64 po 40 iteracji wyrażenia dla:

- $c = -2$ i $x_0 \in \{1, 2, 1.99999999999999\}$;
- $c = -1$ i $x_0 \in \{1, -1, 0.75, 0.25\}$.

6.2 Rozwiązanie

Rozwiązanie polega na implementacji funkcji obliczającej rekurencyjnie następne czterdzieści wyników równania w arytmetyce Float64. Pomocne w rozwiązaniu jest przeprowadzenie iteracji graficznych $x_{n+1} := x_n^2 + c$.

6.3 Wyniki oraz ich interpretacja

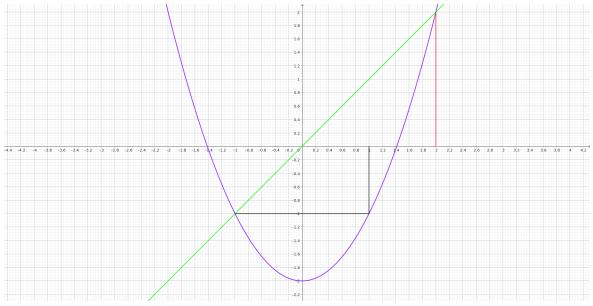
c = -2.0	1.0	2.0	1.99999999999999
1	-1.0	2.0	1.99999999999996
2	-1.0	2.0	1.9999999999998401
3	-1.0	2.0	1.9999999999993605
4	-1.0	2.0	1.999999999997442
5	-1.0	2.0	1.9999999999897682
6	-1.0	2.0	1.9999999999590727
7	-1.0	2.0	1.999999999836291
8	-1.0	2.0	1.9999999993451638
9	-1.0	2.0	1.9999999973806553
10	-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 9: Wyniki iteracji dla c = -2.0

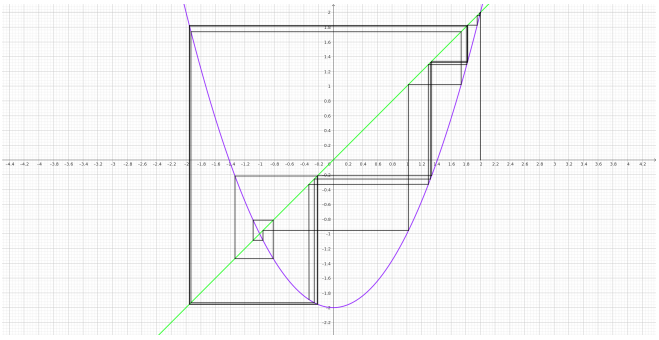
c = -1.0	1.0	-1.0	0.75	0.25
1	0.0	0.0	-0.4375	-0.9375
2	-1.0	-1.0	-0.80859375	-0.12109375
3	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	0.0	0.0	-0.2253147218564956	-0.9991524699951226
6	-1.0	-1.0	-0.9492332761147301	-0.0016943417026455965
7	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 10: Wyniki iteracji dla c = -1.0

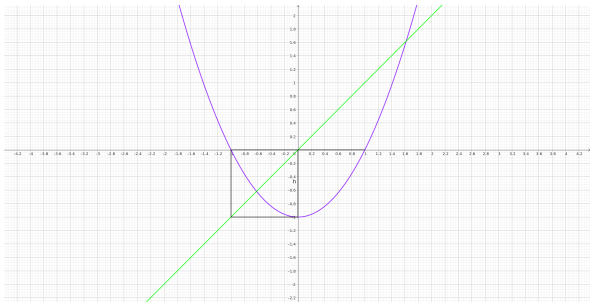
Do pomocy analizy wyników zastała przeprowadzona iteracja graficzna $x_{n+1} := x_n^2 + c$.



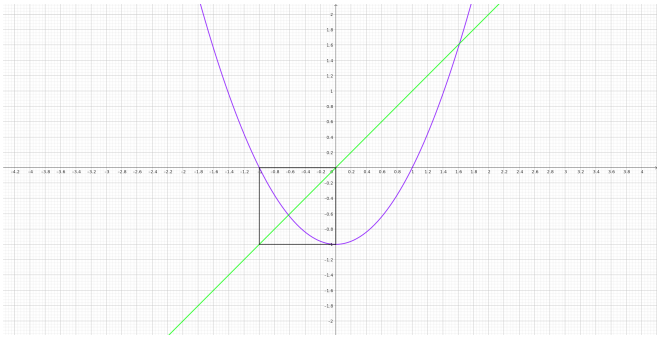
Rysunek 3:
Wykres dla c = -2 i x_0 = 1 (czarny) lub x_0 = 2 (czerwony). W obu przypadkach ciągi wędrują do punktów stałych, tj. odpowiednio -1 i 2, z których nie można wyjść, co obrazuje nam brak dalszych zmian w Tabeli 9.



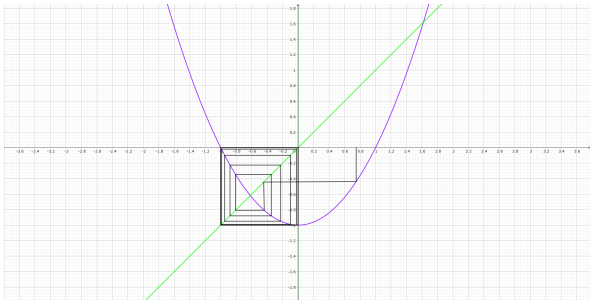
Rysunek 4:
Wykres dla c = -2 i x_0 = 1.99999999999999 (czarny). W tym przypadku ciąg raczej się nie stabilizuje- wędruje w okolicach wartości 1.9, 1.7, -0.2, -1.9, co zgadza się z danymi Tabeli 9. Cykl się niestabilizuje.



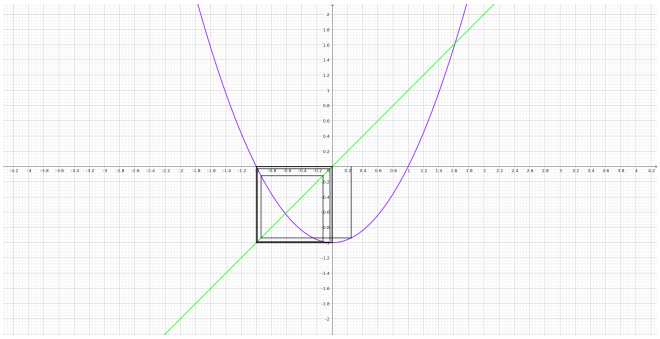
Rysunek 5:
Wykres dla c = -1 i x_0 = 1 (czarny). W tym przypadku widać zacyklenie się procesu, co odpowiada stabilizującym się wynikom zawartym w Tabeli 10, gdzie w kółko powtarzają się wartości 0 i -1.



Rysunek 6:
Wykres dla c = -1 i x_0 = -1 (czarny). W tym przypadku również widać zacyklenie się procesu, co odpowiada stabilizującym się wynikom zawartym w Tabeli 10, gdzie w kółko powtarzają się wartości 0 i -1.



Rysunek 7:
Wykres dla c = -1 i x_0 = 0.75 (czarny). Podobnie jak w poprzednim przypadku ciąg osiąga stabilizacje dopiero po pewnym czasie (z Tabeli 10 można odczytać, że dzieje się to szybciej, już po 10 iteracji. W kółko powtarzają się wartości 0 i -1.



Rysunek 8:
Wykres dla c = -1 i x_0 = 0.25 (czarny). W tym przypadku ciąg osiąga stabilizacje dopiero po pewnym czasie (z Tabeli 10 można odczytać, że dzieje się to po 15 iteracji. W kółko powtarzają się wartości 0 i -1.

6.4 Wnioski

Rekurencyjnie wyznaczanie ciągu może mieć różną stabilność w zależności od danych wejściowych, od całkowicie stabilnego, przez stabilizującego się (w różnych tempach w zależności od parametrów), aż do całkowicie niestabilnego. Do uzyskania wiarygodnych wyników trzeba dobrać odpowiednie wartości parametrów i ilość iteracji.