Obliczenia naukowe

Prowadzący: dr hab. Paweł Zieliński

wtorek TN 7^{30}

Sprawozdanie 2

Weronika Jasiak

236733

1. Zadanie 1

1.1. Opis problemu

Ponowne rozwiązanie zadania 5 z listy 1 z uwzględnieniem zmiany danych w x_4 oraz x_5, która prezentuje się następująco:

$$x_4 = 0.5772156649 \rightarrow x_4 = 0.577215664$$
$$x_5 = 0.3010299957 \rightarrow x_5 = 0.301029995$$

1.2. Rozwiązanie

Obliczając iloczyny skalarne skorzystano z funkcji wykonanej na potrzeby zadania 5 z listy 1, w których to zmieniono dane wejściowe zgodnie z powyższym schematem.

1.3. Wyniki

Dla poszczególnych typów zmiennopozycyjnych uzyskano następujące wyniki (Tabela 1, Tabela 2, Tabela3, Tabela4).

	Lista 1	Lista 2
x_4	00111111000100111100010001101000	00111111000100111100010001101000
x_5	00111110100110100010000010011011	00111110100110100010000010011010

Tabela 1. Zapis bitowy danych dla arytmetyki `Float32`

	Lista 1	Lista 2
x_4	0011…1000110111110000000000101010	0011…0111111001111001110001111011
x_5	00…1000010101001011000010011000	00…0111111010001111011001111001

Tabela 2. Zapis bitowy danych dla arytmetyki `Float64`

	Lista 1	Lista 2
1	-0.4999443	-0.4999443
2	-0.4543457	-0.4543457
3	-0.5	-0.5
4	-0.5	-0.5

Tabela 3. Produkt iloczynu skalarnego wektorów na cztery różne sposoby dla arytmetyki `Float32`

	Lista 1	Lista 2
1	1.0251881368296672e-10	-0.004296342739891585
2	-1.5643308870494366e-10	-0.004296342998713953
3	0.0	-0.004296342842280865
4	0.0	-0.004296342842280865

Tabela 4. Produkt iloczynu skalarnego wektorów na cztery różne sposoby dla arytmetyki `Float64`

1.4. Wnioski

Otrzymane wyniki dla arytmetyki `Float32` (Tabela 3.) są identyczne dla obu zestawów danych, co wiąże się z dość niską (pojedynczą) precyzją tej arytmetyki. W Tabeli 1. został przedstawiony 32 bitowy zapis liczb zarówno przed zmianą danych jak i po. Można jednoznacznie stwierdzić, że w tej arytmetyce zapis x_4 jest identyczny, a dla x_5 różni się jedynie ostatnim bitem. Z kolei w arytmetyce `Float64` (Tabela 2. I Tabela 4.) różnice są widoczne gołym okiem, co wiąże się z podwójną precyzją arytmetyki, dzięki której można przechowywać dokładniejsze dane. Nieznaczne zmiany danych mają duży wpływ na otrzymane wyniki, które mogą prowadzić do uzyskania błędnych wartości.

2. Zadanie 2

2.1. Opis problemu

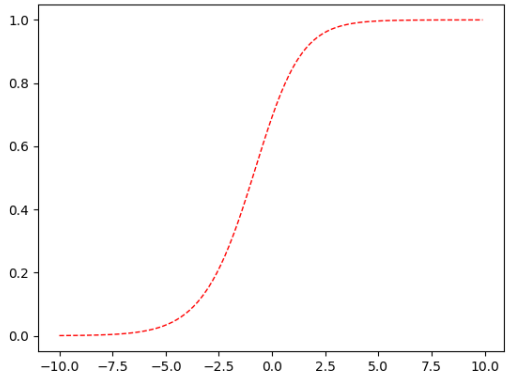
Narysowanie wykresu funkcji $f(x) = e^x \ln(1 + e^{-x})$ przy wykorzystaniu co najmniej dwóch dowolnych programów do wizualizacji oraz obliczenie granicy funkcji $f(x) = \lim_{x \to \infty} f(x)$.

2.2. Rozwiązanie

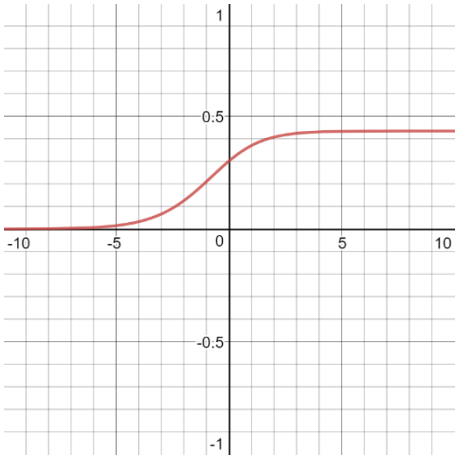
Wykresy narysowano przy użyciu pakietu `PyPlot` rozszerzającego język `Julia` oraz dwóch zewnętrznych programów `Desmos` i `Fooplot`. Granicę obliczono przy pomocy funkcji `limit` dostępnej w pakiecie `SymPy` rozszerzającej język `Julia`.

2.3. Wyniki

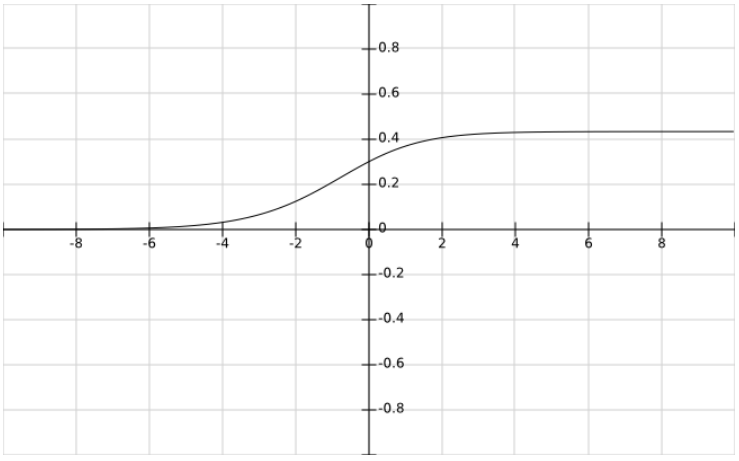
Dla funkcji $f(x) = e^x \ln(1 + e^{-x})$ otrzymano następującą interpretację graficzną (Wykres 1., Wykres 2, Wykres 3.).



a) `Pyplot`

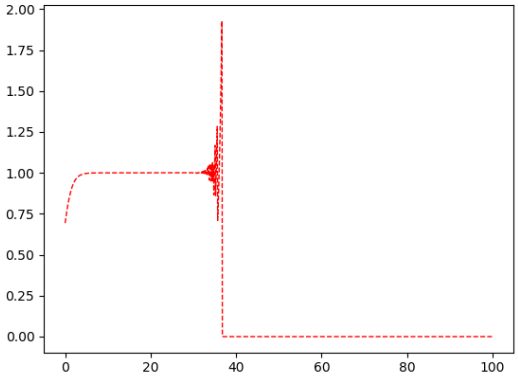


b) `Desmos`

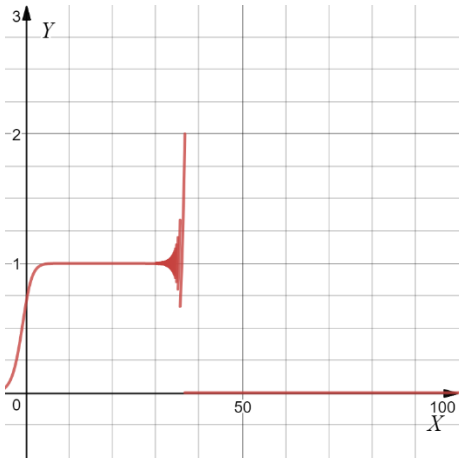


c) `Fooplot`

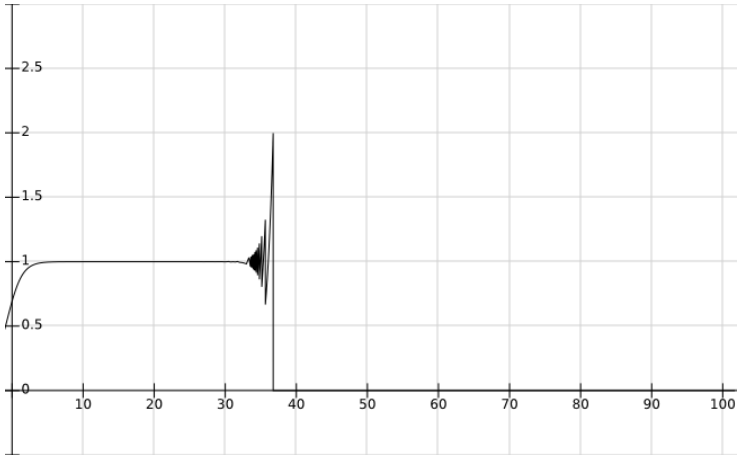
Wykres 1. Wykres funkcji dla $x \in (-10; 10)$



a) `Pyplot`

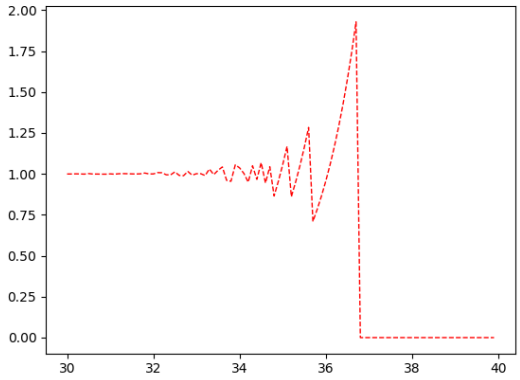


b) `Desmos`

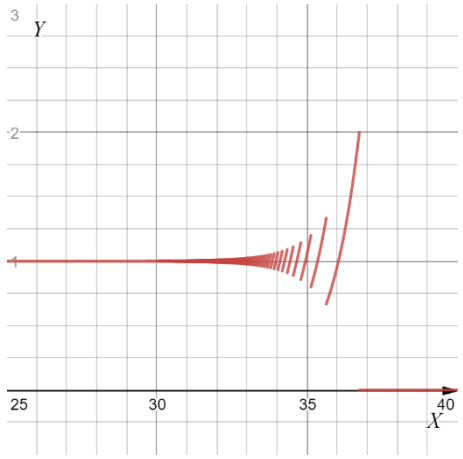


c) `Fooplot`

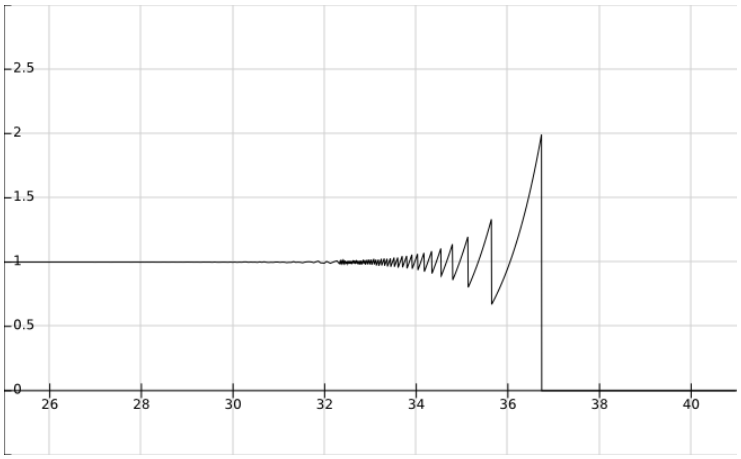
Wykres 2. Wykres funkcji dla $x \in (0; 100)$



a) `Pyplot`



b) `Desmos`



c) `Fooplot`

Wykres 3. Wykres funkcji dla $x \in (30; 40)$

Obliczono granicę funkcji $f(x) = e^x \ln(1 + e^{-x})$.

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

2.4. Wnioski

Granica funkcji $f(x)$ przy $x \to \infty$ wynosi 1, jednak gdy przyjrzymy się bliżej wykresom dla wartości argumentów przekraczających 31 funkcja zaczyna oscylować, co nie jest zgodne z wcześniej obliczoną granicą. Mnożenie logarytmu, który przyjmuje bardzo małe wartości z potęgowaną liczbą e, która z kolei przyjmuje bardzo duże wartości prowadzi do uzyskania błędnych wyników. Gdy argumenty są już większe od 36 funkcja przyjmuje stałą wartość równą 0, czego przyczyną jest malejąca wartość e^{-x}. Nieznaczne zmiany danych mają duży wpływ na otrzymane wyniki, które mogą prowadzić do uzyskania błędnych wartości – dużych odchyleń.

3. Zadanie 3

3.1. Opis problemu

Rozwiązanie układu równań liniowych $Ax = b$ dla danej macierzy współczynników $A \in \mathbb{R}^{n \times n}$ i wektora prawych stron $b \in \mathbb{R}^n$ za pomocą dwóch algorytmów: eliminacji Gaussa (`x=A\b`) oraz $x = A^{-1}b$ (`x=inv(A)*b`). Macierz A generowana jest w następujący sposób:

a) $A = H_n$, gdzie H_n jest macierzą Hilberta stopnia n wygenerowaną za pomocą funkcji `A=hilb(n)`.

b) $A = R_n$, gdzie R_n jest losową macierzą stopnia n z zadanym wskaźnikiem uwarunkowania c wygenerowaną za pomocą funkcji `A=matcond(n,c)`.

Następnie wykonanie eksperymentów dla macierzy Hilberta H_n z rosnącym stopniem $n > 1$ oraz dla macierzy losowej R_n, $n = 5, 10, 20$ z rosnącym wskaźnikiem uwarunkowania $c = 1, 10, 10^3, 10^7, 10^{12}, 10^{16}$ oraz porównanie obliczonych $\tilde{x}$ z rozwiązaniem dokładnym $x = (1, \ldots, 1)^T$ i policzenie błędów względnych.

3.2. Rozwiązanie

Z wykorzystaniem języka `Julia` zostały stworzone dwie funkcje pozwalające generować macierze (macierz Hilberta stopnia n wygenerowano przy pomocy funkcji `A=hilb(n)`, a losową macierz stopnia n przy pomocy funkcji `A=matcond(n,c)`. W następnym kroku zostały obliczone błędy względne dla dwóch podanych powyżej metod.

3.3. Wyniki

Dla macierzy Hilberta uzyskano następujące wyniki (Tabela 5).

rząd	wskaźnik uwarunkowania	eliminacja Gaussa	odwrotność macierzy
1	1.0	0.0	0.0
2	$1.928147006790397 * 10^{1}$	$5.661048867003676 * 10^{-16}$	$1.4043333874306803 * 10^{-15}$
3	$5.240567775860644 * 10^{2}$	$8.022593772267726 * 10^{-15}$	0.0
4	$1.551373873892924 * 10^{4}$	$4.137409622430382 * 10^{-14}$	0.0
5	$47660725024259434 * 10^{5}$	$1.6828426299227195 * 10^{-12}$	$3.3544360584359632 * 10^{-12}$
6	$1.4951058642254665 * 10^{7}$	$2.618913302311624 * 10^{-10}$	$2.0163759404347654 * 10^{-10}$
7	$4.75367356583129 * 10^{8}$	$1.2606867224171548 * 10^{-8}$	$4.713280397232037 * 10^{-9}$
8	$1.5257575538060041 * 10^{10}$	$6.124089555723088 * 10^{-8}$	$3.07748390309622 * 10^{-7}$
9	$4.931537564468762 * 10^{11}$	$3.8751634185032475 * 10^{-6}$	$4.541268303176643 * 10^{-6}$
10	$1.6024416992541715 * 10^{13}$	$8.67039023709691 * 10^{-5}$	$2.501493411824886 * 10^{-4}$
11	$5.222677939280335 * 10^{14}$	$1.5827808158590435 * 10^{-4}$	$7.618304284315809 * 10^{-3}$
11	$1.7514731907091464 * 10^{16}$	$1.3396208372085344 * 10^{-1}$	$2.58994120804705 * 10^{-1}$
11	$3.344143497338461e18 * 10^{18}$	$1.1039701117868264 * 10^{-1}$	5.331275639426837
12	$6.200786263161444 * 10^{17}$	1.4554087127659643	8.71499275104814
12	$3.674392953467974 * 10^{17}$	4.696668350857427	7.344641453111494
12	$7.865467778431645 * 10^{17}$	$5.415518954564602 * 10^{1}$	$2.984884207073541 * 10^{1}$
12	$1.263684342666052 * 10^{18}$	$1.3707236683836307 * 10^{1}$	$1.0516942378369349 * 10^{1}$
12	$2.2446309929189128 * 10^{18}$	9.134134521198485	7.575475905055309
13	$6.471953976541591 * 10^{18}$	9.720589712655698	$1.2233761393757726 * 10^{1}$
13	$1.3553657908688225 * 10^{18}$	7.549915039472976	$2.2062697257870493 * 10^{1}$

Tabela 5. Wyniki eksperymentu z macierzą Hilberta H_n

Dla losowej macierzy uzyskano następujące wyniki (Tabela 6).

rząd	wskaźnik uwarunkowania	eliminacja Gaussa	odwrotność macierzy
5	1.0000000000000002	$2.432376777795247 * 10^{-16}$	$2.220446049250313 * 10^{-16}$
5	10.00000000000001	$1.9860273225978183 * 10^{-16}$	$2.2752801345137457 * 10^{-16}$
5	$1.0000000000000764 * 10^{3}$	$8.43696454564983 * 10^{-15}$	$1.1234667099445442 * 10^{-14}$
5	$9.999999990532476 * 10^{6}$	$2.8587566296606945 * 10^{-10}$	$2.8752463486111883 * 10^{-10}$
5	$9.999548901019196 * 10^{11}$	$2.1545811788938093 * 10^{-6}$	$2.0454035367514813 * 10^{-6}$
4	$6.45531203719893 * 10^{15}$	$2.811599519368878 * 10^{-1}$	$3.5603063421565284 * 10^{-1}$
10	1.0000000000000016	$3.4932351950072765 * 10^{-16}$	$2.764433037591714 * 10^{-16}$
10	10.0	$1.447553722489536 * 10^{-16}$	$1.954749347017227 * 10^{-16}$
10	$1.0000000000000202 * 10^{3}$	$2.4012766360639104 * 10^{-14}$	$2.5271758721286715 * 10^{-14}$
10	$1.000000000660034 * 10^{7}$	$2.2218498218956288 * 10^{-10}$	$2.1178634310403879 * 10^{-10}$
10	$9.999883675840302 * 10^{11}$	$1.952188740359763 * 10^{-6}$	$4.228318379355514 * 10^{-6}$
9	$6.987835713607105 * 10^{15}$	$3.867870129935218 * 10^{-3}$	$1.8936249443032533 * 10^{-2}$
20	1.000000000000001	$6.250861277594332 * 10^{-16}$	$3.797547027170516 * 10^{-16}$
20	$1.0000000000000012 * 10^{1}$	$6.014704321524861 * 10^{-16}$	$5.832634613762131 * 10^{-16}$
20	$9.999999999999862 * 10^{2}$	$4.961653647841151 * 10^{-15}$	$7.09153599073145 * 10^{-15}$
20	$9.99999999156604 * 10^{6}$	$2.6083916022025507 * 10^{-10}$	$2.546273319115533 * 10^{-10}$
20	$1.0000931293678986 * 10^{12}$	$4.207217722647996 * 10^{-5}$	$3.802294237196825 * 10^{-5}$
19	$7.699855054655139 * 10^{15}$	$17223888214551697 * 10^{-1}$	$19782044598136614 * 10^{-1}$

Tabela 6. Wyniki eksperymentu z losową macierzą R_n

3.4. Wnioski

Wyniki znajdujące się w Tabeli 5. jednoznacznie wskazują, że dla macierzy Hilberta błąd względny rośnie, gdy zwiększa się jej rząd oraz wartość wskaźnika uwarunkowania (`cond(A)`), ponieważ macierz ta jest źle uwarunkowana. Również w przypadku losowej macierzy R_n (Tabela 6.) błąd rośnie wraz ze zwiększającą się wartością `cond(A)`. Zatem wraz ze wzrostem wskaźnika uwarunkowania rośnie błąd względny.

4. Zadanie 4

4.1. Opis problemu

Obliczenie dwudziestu zer wielomianu P w postaci naturalnej:

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

gdzie P jest postacią naturalną wielomianu Wilkinsona p

$$\begin{aligned} p(x) = {} & (x - 20)(x - 19)(x - 18)(x - 17)(x - 16) \\ & (x - 15)(x - 14)(x - 13)(x - 12)(x - 11) \\ & (x - 10)(x - 9)(x - 8)(x - 7)(x - 6) \\ & (x - 5)(x - 4)(x - 3)(x - 2)(x - 1), \end{aligned}$$

oraz sprawdzenie obliczonych pierwiastków z_k, $1 \leq k \leq 20$, obliczając $|P(z_k)|$, $|p(z_k)|$, $|z_k - k|$. Następnie powtórzenie testu dla zmienionego współczynnika stojącego przy x^{19}, który wygląda następująco:

$$-210x^{19} \rightarrow -210 - 2^{-23}.$$

4.2. Rozwiązanie

Korzystając z pakietu `Polynomials` będącego rozszerzeniem języka `Julia` stworzono kompleksowy sposób rozwiązania zadania. W pierwszym kroku zostały stworzone dwa wielomiany, jeden w postaci ogólnej z wykorzystaniem funkcji `Poly`, a drugi w postaci iloczynowej z wykorzystaniem funkcji `poly`. Następnie zostały obliczone pierwiastki wielomianu $z_k, 1 \leq k \leq 20$ z wykorzystaniem funkcji `roots`. Na samym końcu zostały obliczone błędy względne wielomianów oraz ich miejsc zerowych.

4.3. Wyniki

Dla wielomianu Wilkinsona przed zmianami uzyskano następujące wyniki (Tabela 7).

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
20	19.999809291236637	$2.7462952745472 * 10^{13}$	$2.7462952745472 * 10^{13}$	$1.9070876336257925 * 10^{-4}$
19	19.00190981829944	$1.0278376162816 * 10^{13}$	$1.0278376162816 * 10^{13}$	$1.9098182994383706 * 10^{-3}$
18	17.99092135271648	$7.199554861056 * 10^{12}$	$7.199554861056 * 10^{12}$	$9.078647283519814 * 10^{-3}$
17	17.025427146237412	$3.777623778304 * 10^{12}$	$3.777623778304 * 10^{12}$	$2.5427146237412046 * 10^{-2}$
16	15.946286716607972	$1.555027751936 * 10^{12}$	$1.555027751936 * 10^{12}$	$5.371328339202819 * 10^{-2}$
15	15.075493799699476	$6.13987753472 * 10^{11}$	$6.13987753472 * 10^{11}$	$7.549379969947623 * 10^{-2}$
14	13.914755591802127	$3.65383250944 * 10^{11}$	$3.65383250944 * 10^{11}$	$8.524440819787316 * 10^{-2}$
13	13.07431403244734	$2.15723629056 * 10^{11}$	$2.15723629056 * 10^{11}$	$7.431403244734014 * 10^{-2}$
12	11.953283253846857	$7.216771584 * 10^{10}$	$7.216771584 * 10^{10}$	$4.671674615314281 * 10^{-2}$
11	11.025022932909318	$3.5759895552 * 10^{10}$	$3.5759895552 * 10^{10}$	$2.5022932909317674 * 10^{-2}$
10	9.990413042481725	$1.2707126784 * 10^{10}$	$1.2707126784 * 10^{10}$	$9.586957518274986 * 10^{-3}$
9	9.002915294362053	$4.465326592 * 10^{9}$	$4.465326592 * 10^{9}$	$2.915294362052734 * 10^{-3}$
8	7.999355829607762	$1.682691072 * 10^{9}$	$1.682691072 * 10^{9}$	$6.441703922384079 * 10^{-4}$
7	7.000102002793008	$4.80398336 * 10^{8}$	$4.80398336 * 10^{8}$	$1.0200279300764947 * 10^{-4}$
6	5.999989245824773	$1.20152064 * 10^{8}$	$1.20152064 * 10^{8}$	$1.0754175226779239 * 10^{-5}$
5	5.000000665769791	$2.4114688 * 10^{7}$	$2.4114688 * 10^{7}$	$6.657697912970661 * 10^{-7}$
4	3.9999999837375317	$3.106816 * 10^{6}$	$3.106816 * 10^{6}$	$1.626246826091915 * 10^{-8}$
3	2.9999999995920965	209408.0	209408.0	$4.0790348876384996 * 10^{-10}$
2	2.0000000000283182	181760.0	181760.0	$2.8318236644508943 * 10^{-11}$
1	0.9999999999996989	36352.0	36352.0	$3.0109248427834245 * 10^{-13}$

Tabela 7. Pierwiastki wielomianu i błędy uzyskane przy ich obliczaniu

Dla wielomianu Wilkinsona po wykonanej zmianie uzyskano następujące wyniki (Tabela 8).

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
20	$20.84691021519479 + 0.0im$	$1.114453504512 * 10^{13}$	$1.3743733195398482 * 10^{18}$	$8.469102151947894 * 10^{-1}$
19	$19.5024423688181 + 1.940331978642903im$	$9.539424609817828 * 10^{12}$	$4.252502487879955 * 10^{17}$	2.004329444309949
18	$19.5024423688181 - 1.940331978642903im$	$9.539424609817828 * 10^{12}$	$4.252502487879955 * 10^{17}$	2.454021446312976
17	$16.73074487979267 + 2.812624896721978im$	$3.315103475981763 * 10^{11}$	$2.7420894080997828 * 10^{16}$	2.825483521349608
16	$16.73074487979267 - 2.812624896721978im$	$3.315103475981763 * 10^{11}$	$2.7420894080997828 * 10^{16}$	2.9060018735375106
15	$13.992406684487216 + 2.5188244257108443im$	$1.0612064533081976 * 10^{11}$	$9.545941965367332 * 10^{14}$	2.7128805312847097
14	$13.992406684487216 + 2.5188244257108443im$	$1.0612064533081976 * 10^{11}$	$9.545941965367332 * 10^{14}$	2.5188358711909045
13	$11.793890586174369 + 1.6524771364075785im$	$3.357756113171857 * 10^{10}$	$3.2960224849741504 * 10^{13}$	2.045820276678428
12	$11.793890586174369 - 1.6524771364075785im$	$3.357756113171857 * 10^{10}$	$3.2960224849741504 * 10^{13}$	1.665281290598479
11	$10.095455630535774 + 0.6449328236240688im$	$7.143113638035824 * 10^{9}$	$1.4912572850824043 * 10^{12}$	1.1109180272716561
10	$10.095455630535774 - 0.6449328236240688im$	$7.143113638035824 * 10^{9}$	$1.4912572850824043 * 10^{12}$	$6.519586830380406 * 10^{-1}$
9	$8.915816367932559 + 0.0im$	$3.065575424 * 10^{9}$	$1.37168464896 * 10^{11}$	$8.41836320674414 * 10^{-2}$
8	$8.007772029099446 + 0.0im$	$1.072547328 * 10^{9}$	$1.852128 * 10^{10}$	$7.772029099445632 * 10^{-3}$
7	$6.99960207042242 + 0.0im$	$3.88123136 * 10^{8}$	$1.754868736 * 10^{9}$	$3.9792957757978087 * 10^{-4}$
6	$6.000020476673031 + 0.0im$	$1.29148416 * 10^{8}$	$2.04793344 * 10^{8}$	$2.0476673030955794 * 10^{-5}$
5	$4.99999857388791 + 0.0im$	$3.9463936 * 10^{7}$	$4.2535936 * 10^{7}$	$1.4261120897529622 * 10^{-7}$
4	$4.000000089724362 + 0.0im$	$1.046784 * 10^{7}$	$1.046784 * 10^{7}$	$8.972436216225788 * 10^{-8}$
3	$2.99999999660342 + 0.0im$	$2.221568 * 10^{6}$	$2.221568 * 10^{6}$	$3.3965799062229962 * 10^{-10}$
2	$2.0000000000550373 + 0.0im$	349184.0	349184.0	$5.503730804434781 * 10^{-11}$
1	$0.9999999999998357 + 0.0im$	20992.0	20992.0	$1.6431300764452317 * 10^{-13}$

Tabela 8. Pierwiastki wielomianu i błędy uzyskane przy ich obliczaniu gdy dane zostały zmienione

4.4. Wnioski

Otrzymane wyniki różnią się znacznie od oczekiwanych. Dane w powyższych tabelach (Tabela 7. oraz Tabela 8.) wskazują jednoznacznie, że wraz ze wzrostem wartości pierwiastków wzrasta również wartość błędu. Dokładna reprezentacja współczynników nie jest możliwa, ponieważ w arytmetyce `Float64` dostępne jest tylko od 15 do 17 miejsc na cyfry znaczące, stąd wynikają zaburzenia. Obliczenia wykonane dla drugiego wielomianu prowadzą do analogicznych wniosków. Natomiast zmienione dane uwydatniają różnice pomiędzy błędami bezwzględnymi, powodując ich zwiększenie względem poprzedniego wielomianu. Pierwiastki uzyskane po tej małej zmianie należą już do zbioru liczb zespolonych. Zatem niewielkie zmiany mają znaczący wpływ na otrzymane wyniki, mogą powodować błędy oraz zaburzenia utrudniające ich interpretację.

5. Zadanie 5

5.1. Opis problemu

Przeprowadzenie dwóch eksperymentów z wykorzystaniem modelu wzrostu populacji, który reprezentuje poniższe równanie rekurencyjne

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

gdzie 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. Przeprowadzić eksperymenty dla $p_0 = 0.01, r = 3$, gdzie:

1. obliczenia są wykonywane w arytmetyce `Float32` dla 40 kolejnych iteracji pętli,
2. obliczenia są wykonywane w arytmetyce `Float32` dla 10 kolejnych iteracji pętli, następnie wstrzymać działanie, zastosować obcięcie wyniku odrzucając cyfry po trzecim miejscu po przecinku i kontynuować dalej obliczenia.
3. obliczenia są wykonywane w arytmetyce `Float64` dla 40 kolejnych iteracji pętli.

5.2. Rozwiązanie

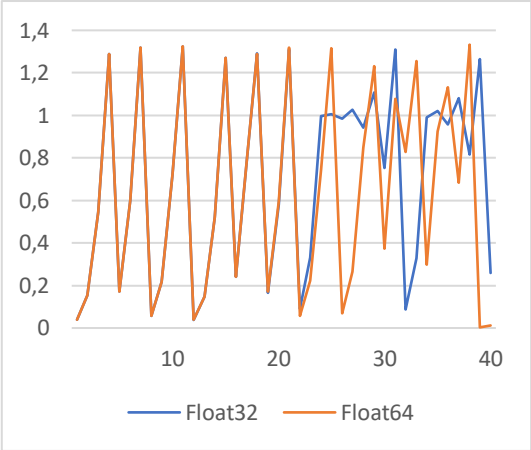
Z wykorzystaniem języka `Julia` zostały stworzone dwie proste funkcje pozwalające wykonać 40 iteracji pętli, wyznaczając w ten sposób wartość p_n, przy czym jedna z nich po 10 kolejnych wywołaniach dokonywała obcięcia wyniku odrzucając cyfry po trzecim miejscu.

5.3. Wyniki

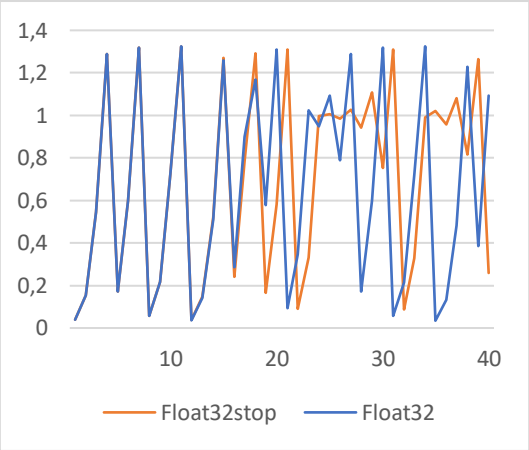
Dla powyższych danych otrzymano następujące wyniki (Tabela 9).

	Float64	Float32	Float32 stop
1	0.0397	0.0397	0.0397
2	0.15407173000000002	0.15407173	0.15407173
3	0.5450726260444213	0.5450726	0.5450726
4	1.2889780011888006	1.2889781	1.2889781
	⋮	⋮	⋮
11	0.722914301179573	0.7229306	0.722
12	1.3238419441684408	1.3238364	1.3241479
13	0.03769529725473175	0.037716985	0.036488414
14	0.14651838271355924	0.14660022	0.14195944
	⋮	⋮	⋮
21	0.5965293124946907	0.5799036	1.3096911
22	1.3185755879825978	1.3107498	0.09289217
23	0.058377608259430724	0.088804245	0.34568182
24	0.22328659759944824	0.3315584	1.0242395
	⋮	⋮	⋮
31	0.37414648963928676	0.7529209	1.3191822
32	1.0766291714289444	1.3110139	0.05600393
33	0.8291255674004515	0.0877831	0.21460639
34	1.2541546500504441	0.3280148	0.7202578
	⋮	⋮	⋮
37	0.6822410727153098	1.0813814	0.48036796
38	1.3326056469620293	0.81736827	1.2292118
39	0.0029091569028512065	1.2652004	0.3839622
40	0.011611238029748606	0.25860548	1.093568

Tabela 9. Wyniki otrzymane dla eksperymentów w różnych arytmetykach



a) porównanie dwóch arytmetyk



b) porównanie w obrębie jednej arytmetyki

Wykres 4. Wykresy funkcji reprezentują przebieg kolejnych 40 iteracji dla powyższego równania rekurencyjnego.

5.4. Wnioski

Pojedyncza precyzja nie jest wystarczająca, aby otrzymać prawidłowe wyniki. W kolejnych przejściach pętli można zauważyć coraz większą różnicę pomiędzy wartościami dla arytmetyki `Float32` i `Float64`. Jest to spowodowane zbyt małą ilością dostępnych miejsc dla cyfr znaczących w arytmetyce `Float32`. Z kolei wykonując w arytmetyce `Float32` 40 iteracji bez przerwy otrzymujemy inne wyniki niż gdy po 10 iteracji obetniemy wynik do 3 cyfr znaczących. Błąd bezwzględny dla początkowych wartości był równy 0.0, jednak po 10 iteracji zaczął powoli narastać i spowodował znaczne zaburzenia wyniku. Małe zmiany w danych mogą powodować znaczące różnice w wynikach.

6. Zadanie 6

6.1. Opis problemu

Przeprowadzenie eksperymentów polegających na wykonaniu 40 iteracji pętli w arytmetyce `Float64` z wykorzystaniem następującego równania rekurencyjnego:

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

gdzie c jest pewną stałą, dla poniższych danych:

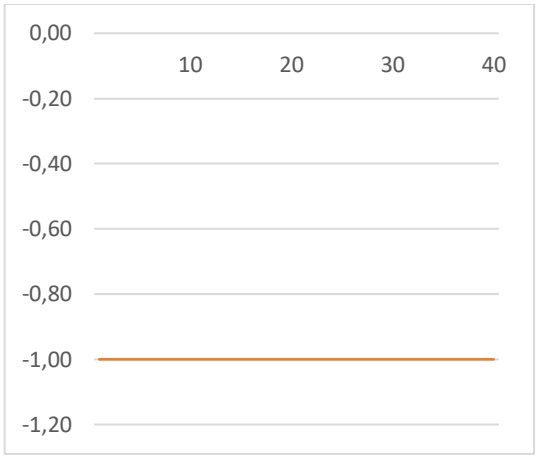
1. $c = -2$ i $x_0 = 1$,
2. $c = -2$ i $x_0 = 2$,
3. $c = -2$ i $x_0 = 1.99999999999999$,
4. $c = -1$ i $x_0 = 1$,
5. $c = -1$ i $x_0 = -1$,
6. $c = -1$ i $x_0 = 0.75$,
7. $c = -1$ i $x_0 = 0.25$.

6.2. Rozwiązanie

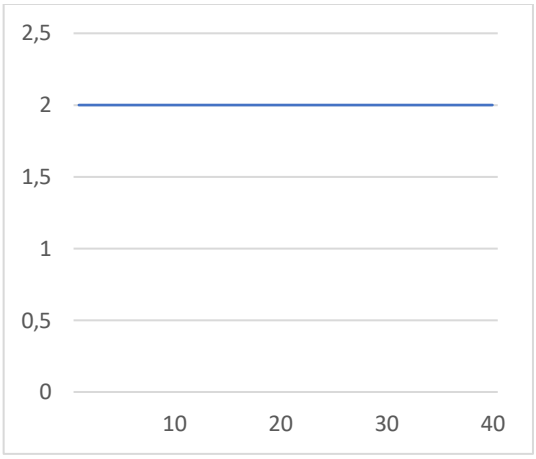
Przy pomocy języka `Julia` zostały stworzone dwie proste funkcje wykonujące 40 kolejnych iteracji pętli z wykorzystaniem powyższego wzoru rekurencyjnego, które na wejściu przyjmują dwa argumenty będące wartościami c i x_0.

6.3. Wyniki

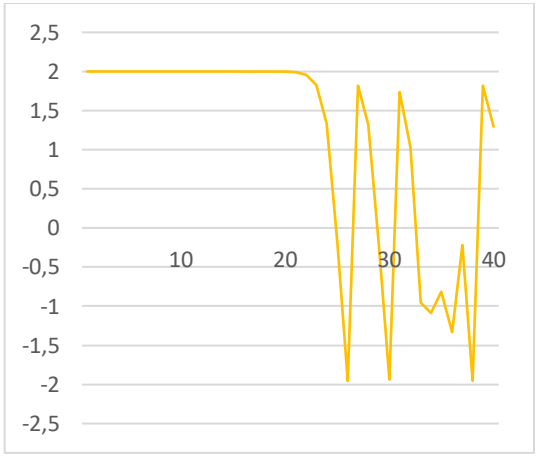
Dla powyższego równania rekurencyjnego w zależności od parametrów wywołania funkcji otrzymano następujące wyniki w postaci graficznej.



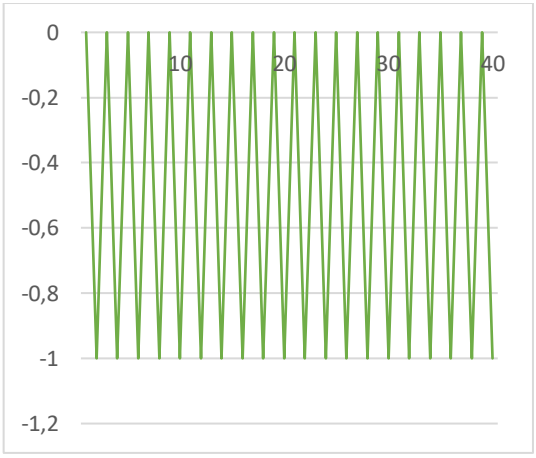
a) $c = -2$ i $x_0 = 1$



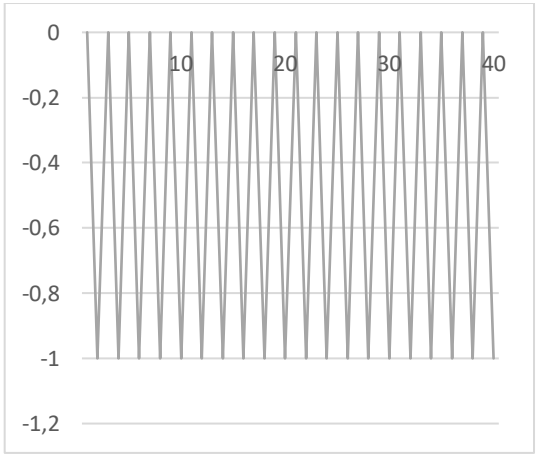
b) $c = -2$ i $x_0 = 2$



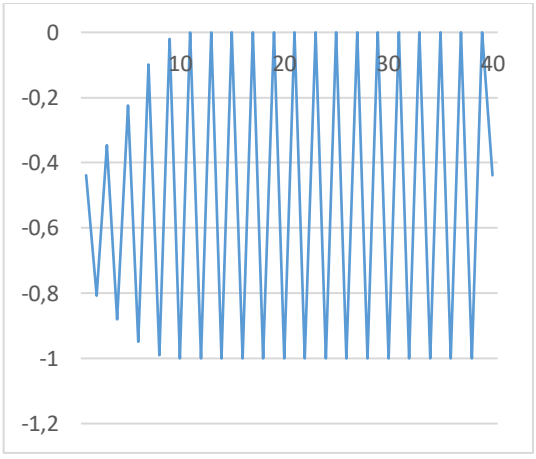
c) $c = -2$ i $x_0 = 1.99999999999999$



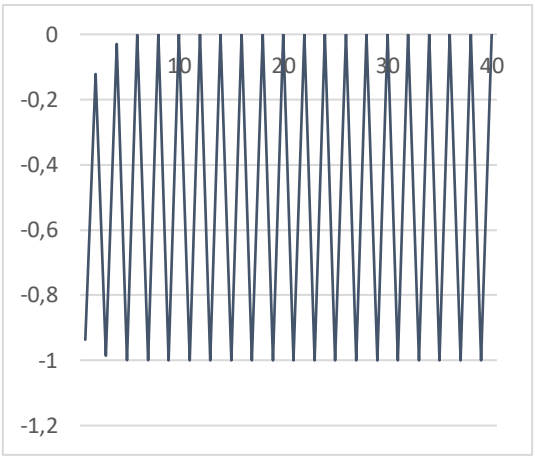
d) $c = -1$ i $x_0 = 1$



e) $c = -1$ i $x_0 = -1$



f) $c = -1$ i $x_0 = 0.75$



g) $c = -1$ i $x_0 = 0.25$

Wykres 5. Wykresy funkcji reprezentują przebieg kolejnych 40 iteracji powyższego równania rekurencyjnego dla różnych danych.

6.4. Wnioski

Wykresy c), f), g) dla zadanych zestawów danych przedstawiają chaos deterministyczny, gdzie błędy zaczynają się kumulować i z każda kolejną iteracją narastać. Błąd znajdujący się na wyjściu zostaje przeniesiony na wejście kolejnej iteracji. Z kolei wykresy a), b), d), e) przedstawiają odwrotna sytuację, gdzie uzyskane wyniki są przewidywalne i zgodne z oczekiwaniami. Niestabilność danych nie wynika więc z podnoszonego do kwadratu x_n, tylko z danych podanych podczas wywołania funkcji. Jeśli są one liczbami całkowitymi, to wykres zaczyna oscylować, jednak gdy dane są już bardziej precyzyjne takie zachowanie następuje dopiero po większej ilości iteracji.