03.11.2019

Obliczenia Naukowe

Lista Nr 2(Laboratorium)

Piotr Popis

- Zadanie 1

1. Opis problemu.

Celem jest ponowne eksperymentalne zbadanie problemu różności rozwiązań iloczynu skalarnego dwóch wektorów. Tym razem usuwamy osatnie znaczące cyfry współrzędnych x_4 oraz x_5 odpowiednio 9 i 7. Problem należy sprawdzić dla pojedynczej i podwójnej precyzji.

@Zadanie 5 Lista 1
x=[2.718281828, (-3.141592654), 1.414213562,**0.5772156649**,**0.3010299957**]
y=[1486.2497,878366.9879, (-22.37492),4773714.647,0.000185049]

@Zadanie 1 Lista 2
a=[2.718281828, (-3.141592654), 1.414213562,**0.577215664**,**0.301029995**]
b=[1486.2497,878366.9879, (-22.37492),4773714.647,0.000185049]

2. Rozwiązanie problemu.

Ponowne przepuszczenie zmienionych danych przez algorytmy :

- (a) "w przód"
- (b) "w tył"
- (c) od największego do najmniejszego (dodaj dodatnie liczby w porządku od największegodo najmniejszego, dodaj ujemne liczby w porządku od najmniejszego do największego,a następnie daj do siebie obliczone sumy częściowe)
- (d) od najmniejszego do największego (przeciwnie do metody (c))

zaimplementowane w zadaniu nr 5 listy nr 1 oraz porównanie i analiza otrzymanych wyników.

3. Wyniki

@Zadanie 5 Lista 1

Algorytm	a	b	c	d
Float32	-0.4999443	-0.4543457	-0.5	-0.5
Float64	1.0251881368296672e-10	-1.5643308870494366e-10	0.0	0.0

@Zadanie 1 Lista 2

Algorytm	a	b	c	d
Float32	-0.4999443	-0.4543457	-0.5	-0.5
Float64	-0.004296342739891585	-0.004296342998713953	-0.004296342842280865	-0.004296342842280865

4. Wnioski.

- *Jaki wpływ na wyniki mają niewielkie zmiany danych?*

W przypadku precyzji **Float32** wyniki nie zmieniły się. Dzieję się tak z przyczny zbyt słabej precyzji(rząd 10^{-7}) arytmetyki względem precyzji zapisu liczby(rząd 10^{-9}).

W przypadku precyzji **Float64** zauważalna jest ogromna różnica między wynikami poprzednimi, (przed ucięciem), a nowymi, (po ucięciu). Możemy śmiało stwierdzić, że jest to przykład zadania **źle uwarunkowanego.** Minimalna zmiana danych wejściowych powoduje ogromne zmiany wyjściowego wyniku.

• Zadanie 2

1. Opis problemu.

Celem jest narysowanie wykresu funkcji f(x) w co najmniej dwóch dowolnych programach do wizualizacji. Oraz policzenie limes f(x) dla $x \to \infty$, gdzie

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

Porównać otrzymany wynik limes z wykresami oraz wyjaśnić zjawisko.

2. Rozwiązanie problemu.

W celu obliczenia wyznaczenia granicy wykorzystuję SymPy oraz jego podrzędna funkcję limit(func,arg,aim). Do narysowania wykresu wykorzstuję Plots- biblioteka

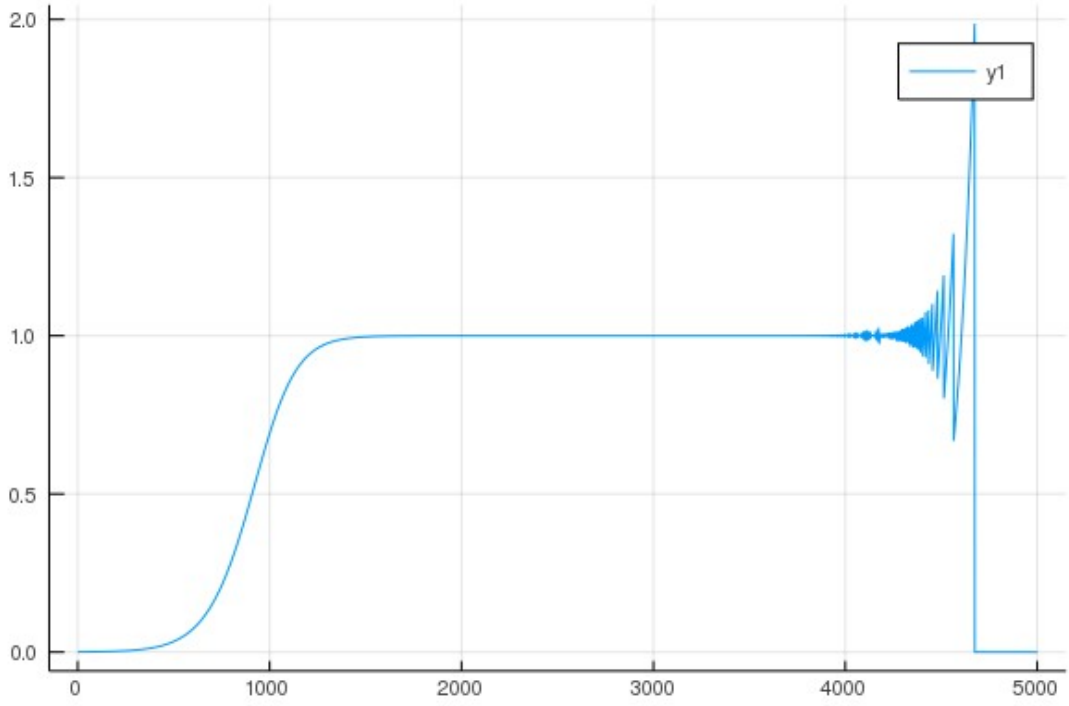
do tworzenia wykresów , Desmos – dostępny w każdej nowoczesnej przeglądarce, Geogebra oraz Footplot.

3. Wyniki.

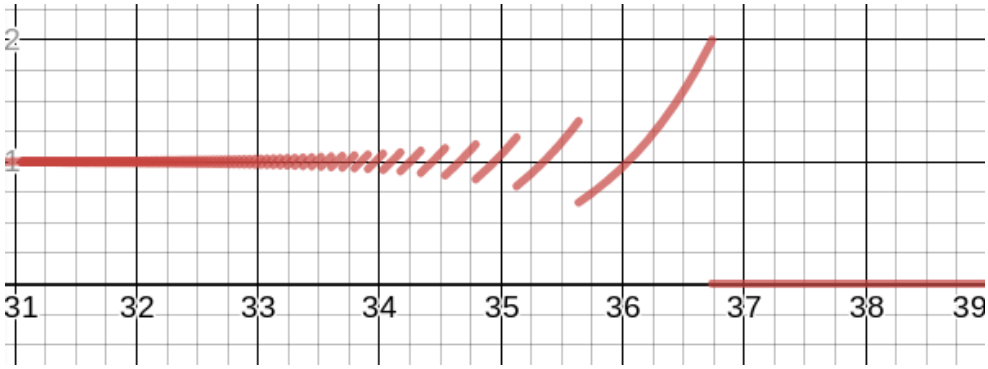
Szukana granica funkcji $f(x) = e^{x} \ln(1 + e^{-x})$ wynosi 1.

Plots – Julia Library

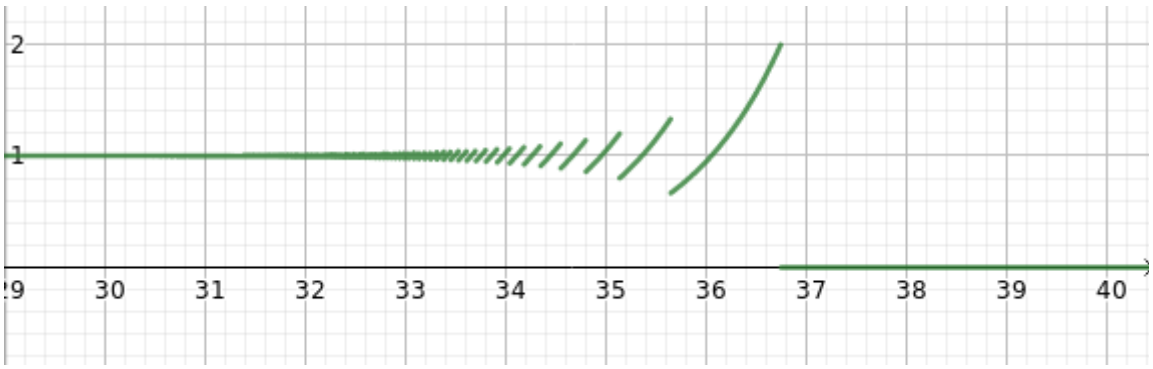
Osie są przesunięte ze względu na komendę RANGE()



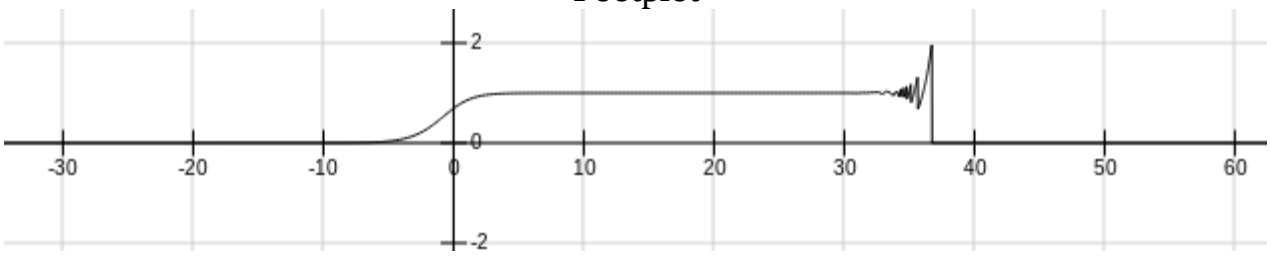
Desmos



Geogebra



Footplot



4. Wnioski.

Jak widzimy po przejściu przez x=32 otrzymujemy bardzo dużą wartość $\mathbf{e^{32}}$ **,** która jest przemnażana przez liczbę bardzo małą $\mathbf{\ln(1+e^{-x})}$. Przez co otrzymujemy wyniki oscylujące na przemian powyżej i poniżej wyliczonej granicy. Otrzymane wyniki są oczywiście błędne. Następnie dla x=37 funkcja osiąga wartość 0 ze względu na fakt, iż wartość $\mathbf{\ln(1+e^{-x})}$ jest już tak mała, że zostaje przybliżona do 0. Więc przemnożona z pozostałym członem zawsze daje zero. W niektórych programach dla dostatecznie dużych x program jest na tyle inteligentny, że wylicza granicę podanej funkcji i od tego miejsca podaje jej wartość zamiast błędnych wyników.

• Zadanie 3

1. Opis problemu.

Głównym problemem jest rozwiązanie układu równań liniowych

$$\mathbf{Ax=b}$$

zadana jest macierz współczynników A i wektor prawych stron b. Macierz generujemy na dwa sposoby. Pierwszy to wykorzystanie macierzy Hilbera stopnia n. Druga natomiast jest macierzą losową stopnia n z zadanym wskaźnikiem uwarunkowania **c**.

Wektor **b** jest zadany jako **Ax**, gdzie a to wygenerowana przez nas macierz, a

$$\mathbf{x=(1,\ldots,1)^T}$$

Rozwiązać zadane równanie na dwa sposoby, przy użyciu eliminacji Gausa i inversji. Eksperymenty wykonać dla rosnącego n>1 w przypadku macierzy Hilberta, dla n=5,10,20 z rosnącym c=1,10,10^3,10^7,10^{12},10^{16}. Policzyć błędy względne.

2. Rozwiązanie problemu.

Do rozwiązania wykorzystamy pakiet LinearAlgebra w celu pobrania cond() oraz rank() odpowiednio wskaźnik uwarunkowania wygenerowanej macierzy oraz rząd macierzy. Zostały zaimplementowane metody do generowania macirzy hilberta oraz losowej z zadanym wskaźnikiem c oraz metoda obliczająca błąd względny wykorzystujący funkcję norm(). Następnie zostały przeprowadzone eksperymenty a ich wyniki możemy sprawdzić w sekcji wyniki.

3. Wyniki.

Macierz Hilberta M_n				
cond()	Rozmiar	rank()	Błąd względny eliminacji Gausaa	Błąd względny inwersji
19.28147006790397	2	2	5.661048867003676e-16	1.4043333874306803e-15
524.0567775860644	3	3	8.022593772267726e-15	0.0
15513.73873892924	4	4	4.137409622430382e-14	0.0
476607.25024259434	5	5	1.6828426299227195e-12	3.3544360584359632e-12
1.4951058642254665e7	6	6	2.618913302311624e-10	2.0163759404347654e-10
4.75367356583129e8	7	7	1.2606867224171548e-8	4.713280397232037e-9
1.5257575538060041e10	8	8	6.124089555723088e-8	3.07748390309622e-7
4.931537564468762e11	9	9	3.8751634185032475e-6	4.541268303176643e-6
1.6024416992541715e13	10	10	8.67039023709691e-5	0.0002501493411824886
5.222677939280335e14	11	10	0.00015827808158590435	0.007618304284315809
1.7514731907091464e16	12	11	0.13396208372085344	0.258994120804705
3.344143497338461e18	13	11	0.11039701117868264	5.331275639426837
6.200786263161444e17	14	11	1.4554087127659643	8.71499275104814
3.674392953467974e17	15	12	4.696668350857427	7.344641453111494
7.865467778431645e17	16	12	54.15518954564602	29.84884207073541
1.263684342666052e18	17	12	13.707236683836307	10.516942378369349
2.2446309929189128e18	18	12	9.134134521198485	7.575475905055309
6.471953976541591e18	19	13	9.720589712655698	12.233761393757726

1.3553657908688225e18	20	13	7.549915039472976	22.062697257870493

Macierz Losowa R_n				
cond()	Rozmiar	rank()	Błąd względny eliminacji Gausaa	Błąd względny inwersji
1.0	5	5	1.5700924586837752e-16	1.4043333874306804e-16
10.0	5	5	1.719950113979703e-16	1.719950113979703e-16
1000.0	5	5	1.4148266844381742e-15	9.79415571016078e-15
1.0e7	5	5	3.6204531690624907e-10	3.1588048560580086e-10
1.0e12	5	5	1.6541515455114935e-5	1.619562508800795e-5
1.0e16	5	4	0.11443581117792277	0.0739509972887452
1.0	10	10	2.1642230995786354e-16	1.9229626863835638e-16
10.0	10	10	1.2658490090568384e-16	2.016820280180126e-16
1000.0	10	10	2.920810890748086e-14	3.0974784661609553e-14
1.0e7	10	10	1.711913162282261e-10	1.7144107812809687e-10
1.0e12	10	10	1.423255114178321e-6	3.428588488053808e-6
1.0e16	10	9	0.032216726513605014	0.04332240822816178
1.0	20	20	5.23691153334427e-16	5.09978018830275e-16
10.0	20	20	3.7485443673843946e-16	4.37799666588975e-16
1000.0	20	20	8.177680028281057e-15	5.9090571005317786e-15
1.0e7	20	20	3.2123824321972287e-10	3.55640107589538e-10
1.0e12	20	20	3.1645714988939825e-5	3.2600983724545365e-5
1.0e16	20	19	0.14898201945476774	0.1196613874909112

4. Wnioski.

Dla macierzy Hilberta wraz ze wzrostem rozmiaru macierzy błąd względny obu metod oraz wskaźnik uwaronkowania cond() również rośnie.

Dla macierzy Losowej przy zwiększeniu stopnia macierzy zauważalny jest wzrost błędów względnych obu metod. Jednak w tym przypadku rosnący błąd powiązany jest również z wskaźnikiem uwarunkowania cond() danej macierzy.

Metoda inwersji jest w tym przypadku widocznie bardziej precyzyjna niż metoda Eliminacji Gauss’a.
Zwłaszcza dla macierzy losowej zauważalne jest, że większy wpływ ma cond() niż sam rozmiar, co dla macierzy Hilberta nie jest tak widoczne. Jednak nie zmienia to faktu, iż tym większa macierz tym większy błąd. Błędy dla macierzy losowej rosną jednakże dużo wolniej niż dla macierzy Hilbera.

Na podstawie wyników eksperymentu macierz Hilberta jest zadaniem **źle uwarunkowaną,** bo kolejna inkrementacja stpnia powoduje gwałtowny wzrost jej wskaźnika uwarunkowania cond(), a za tym wzrost błędu.

- Zadanie 4

1. Opis problemu.

Celem zadania jest znalezienie 20 miejsc zerowych wielomianu P o zadanych współczynnikach:

$$p = [1, -210.0, 20615.0, -1256850.0, 53327946.0, -1672280820.0, 40171771630.0, -756111184500.0, 11310276995381.0, -135585182899530.0, 1307535010540395.0, -10142299865511450.0, 63030812099294896.0, -311333643161390640.0, 1206647803780373360.0, -3599979517947607200.0, 8037811822645051776.0, -12870931245150988800.0, 13803759753640704000.0, -8752948036761600000.0, 2432902008176640000.0]$$

Wielomian skonstruowany z powyższych współczynników jest postacią naturalna wielomianu Wilkinsona p

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

W następnym etapie zadania należy koniecznie sprawdzić wszystkie pierwiastki obliczając $|P(z_k)|$, $|p(z_k)|$ oraz $|z_k-k|$ i wyjaśnić rozbieżności. Sprawdzić eksperyment Wilkinsona.

2. Rozwiązanie problemu.

W pierwszej części zadania skorzystamy z funkcji roots z pakietu Polynomails do obliczenia szukanych miejsc. **Poly** do utworzenia wielomianu z tablicy zaiwerającej współczynniki. Natomiast **poly** do tworzenia wielomianu z tablicy zawierającej miejsca zerowe. Kolejno **polyval(P,x)** do obliczenia wartości wielomianu P w x z wykorzystaniem metody Hornera.
W drugiej części przeprowadzamy ekspryment Wilkinsona tzn, zamieniamy współczynnik przy -210 na $-210\ -2^{-23}$.

3. Wyniki.

3.1.a

k	z_k	$\lvert z_k - k\rvert$	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$
1	0.9999999999996989	3.0109248427834245e-13	36352.0	38400.0
2	2.0000000000283182	2.8318236644508943e-11	181760.0	198144.0
3	2.9999999995920965	4.0790348876384996e-10	209408.0	301568.0
4	3.9999999837375317	1.626246826091915e-8	3.106816e6	2.844672e6
5	5.000000665769791	6.657697912970661e-7	2.4114688e7	2.3346688e7
6	5.999989245824773	1.0754175226779239e-5	1.20152064e8	1.1882496e8
7	7.000102002793008	0.00010200279300764947	4.80398336e8	4.78290944e8
8	7.999355829607762	0.0006441703922384079	1.682691072e9	1.67849728e9
9	9.002915294362053	0.002915294362052734	4.465326592e9	4.457859584e9
10	9.990413042481725	0.009586957518274986	1.2707126784e10	1.2696907264e10
11	11.025022932909318	0.025022932909317674	3.5759895552e10	3.5743469056e10
12	11.953283253846857	0.04671674615314281	7.216771584e10	7.2146650624e10
13	13.07431403244734	0.07431403244734014	2.15723629056e11	2.15696330752e11
14	13.914755591802127	0.08524440819787316	3.65383250944e11	3.653447936e11
15	15.075493799699476	0.07549379969947623	6.13987753472e11	6.13938415616e11
16	15.946286716607972	0.05371328339202819	1.555027751936e12	1.554961097216e12
17	17.025427146237412	0.025427146237412046	3.777623778304e12	3.777532946944e12
18	17.99092135271648	0.009078647283519814	7.199554861056e12	7.1994474752e12
19	19.00190981829944	0.0019098182994383706	1.0278376162816e13	1.0278235656704e13
20	19.999809291236637	0.00019070876336257925	2.7462952745472e13	2.7462788907008e

				13

3.1.b

k	z_k	$\|z_k - k\|$	$\|P(z_k)\|$	$\|p(z_k)\|$
1	0.9999999999998357 + 0.0im	1.6431300764452317e-13	20992.0	22016.0
2	2.0000000000550373 + 0.0im	5.503730804434781e-11	349184.0	365568.0
3	2.99999999660342 + 0.0im	3.3965799062229962e-9	2.221568e6	2.295296e6
4	4.000000089724362 + 0.0im	8.972436216225788e-8	1.046784e7	1.0729984e7
5	4.99999857388791 + 0.0im	1.4261120897529622e-6	3.9463936e7	4.3303936e7
6	6.000020476673031 + 0.0im	2.0476673030955794e-5	1.29148416e8	2.06120448e8
7	6.99960207042242 + 0.0im	0.00039792957757978087	3.88123136e8	1.757670912e9
8	8.007772029099446 + 0.0im	0.007772029099445632	1.072547328e9	1.8525486592e10
9	8.915816367932559 + 0.0im	0.0841836320674414	3.065575424e9	1.37174317056e11
10	10.095455630535774 - 0.6449328236240688im	0.6519586830380406	7.143113638035824e9	1.4912633816754019e12
11	10.095455630535774 + 0.6449328236240688im	1.1109180272716561	7.143113638035824e9	1.4912633816754019e12
12	11.793890586174369 - 1.6524771364075785im	1.665281290598479	3.357756113171857e10	3.2960214141301664e13
13	11.793890586174369 + 1.6524771364075785im	2.045820276678428	3.357756113171857e10	3.2960214141301664e13
14	13.992406684487216 - 2.5188244257108443im	2.5188358711909045	1.0612064533081976e11	9.545941595183662e14
15	13.992406684487216 + 2.5188244257108443im	2.7128805312847097	1.0612064533081976e11	9.545941595183662e14
16	16.73074487979267 - 2.812624896721978im	2.9060018735375106	3.315103475981763e11	2.7420894016764064e16
17	16.73074487979267 + 2.812624896721978im	2.825483521349608	3.315103475981763e11	2.7420894016764064e16

18	19.5024423688181 - 1.940331978642903im	2.454021446312976	9.539424609817828e12	4.2525024879934694e17
19	19.5024423688181 + 1.940331978642903im	2.004329444309949	9.539424609817828e12	4.2525024879934694e17
20	20.84691021519479 + 0.0im	0.8469102151947894	1.114453504512e13	1.3743733197249713e18

4. Wnioski.

4.1.a

Wilkinson swoje odkrycie uznał za najbardziej traumatyczne w całej jego karierze naukowej. Zauważmy, że stosunkowe małe błędy przybliżenia powstałe podczas obliczania pierwiastków przyczyniają się do otrzymania wyników odlegle błędnych. Zadanie obliczenia pierwiastków z_k wielomianu Wilkinsona jest zadaniem skrajnie źle uwarunkowanym. Powodem tych sprzecznych wyników są prawdopodbnie duże wartości współczynników, błąd zatem jest powielany. Drugą sprawą jest rozwikłanie zadanej wskazówki. Język Julia (float64) ma od 15 do 17 cyfr znaczących, więc nie wszystkie współczynniki wielomianu są dokładnie rezprezentowane.

4.1.b Eksperyment Wilkinsona (- 2^{-23})

Dla zoobrazowania różnicy:

$$\overset{-210}{1100000001101010010000000000000000000000\mathbf{0}00000000000000000000}$$

$$\overset{-210-2^{-23}}{1100000001101010010000000000000000000000\mathbf{1}00000000000000000000}$$

Po odjęciu względnie małej liczby 2^{-23} pierwiastki przyjmują zwyczajowo wartości większe niż prawidłowe dla x= 20 otrzymujemy 20,8. Pojawiają się pierwiastki zespolone. Niewielka zmiana współczynnika doprowadza do znacznie innego rozmieszczenia miejsc zerowych. Trzeba zauważyć, że w wejściowym stanie pierwiastki rozłożone są równomiernie. Wilkinson wykazał, iż zachowanie to wynika ze względu na to, że niektóre pierwiastki **a** będą miały wiele bliskich pierwiastków **b**

takich, że |**a-b**| jest mniejszy od $\mathbf{a}^2$ (np. 15). Powyższy eksperyment arguemtnuje, iż zadanie jest zadaniem skrajnie źle uwarunkowanym.

- Zadanie 5

1. Opis problemu.

Rozważanym problem jest model logistyczny, model wzrostu populacji Ver-huste

$$p_{n+1}=p_n+r\,p_n(1-\,p_n)$$

gdzie r jest zadaną stałą, $r(1-p_n)$ jest czynnikiem wzrostu populacji natomiast p_0 jest wielkością poppulacji stanowiącą procent maksymalenj wielkości populacji dla danego stanu środowiska.

W zadaniu mamy zmierzyć się z dwoma eksperymentami. Dla narzuconego p=0.01 oraz 3=3 przeprowadzić 40 iteracji wyrażenia w arytmetyce Float32 z zaburzeniem tzn ucięciem do 3 cyfr po przecinku dla wartości w 10 iteracji . Następnie to samo doświadczenie wykonać bez zbaurzenia dla arytmetki Floaat64.

2. Rozwiązanie problemu.

Zaimplementowanie algorytmu iterującego 40 razy dla aż trzech wartości p_1,p_2 oraz p_3 jest słusznym podejściem. Wykonanie zadanego działania 40 razy dla p_1 oraz p_3 natomiast z zatrzymaniem w 10 iteracji dla p_2. Zapisanie otrzymanych wyników

kolejnych kolumnach w celu ułatwienia interpretacji otrzymanych wartości dla kolejnych problemów.

3. Wyniki.

i	Float32	Float32 z ucięciem	Float64
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	0.036488414	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

4. Wnioski.

Pozornie drobne, niewielkie zmiany w danych takie jak obcięcie kilku miejsc po przecinku doprowadza wyniki końcowe do dużych błędów. Takie ucięcie może być np. spowodowane ograniczeniami ze strony bazy danych.
Zauważalne jest, że różnice są coraz większe w kolejnych iteracjach.
Zadanie jest zadaniem źle uwarunkowanym. Zwiększenie precyzji dla małych n może przynić się do zbliżenia do bardziej poprawnych wyników.

- Zadanie 6

1. Opis problemu.

Rozważanym problemem jest równanie rekurencyjne postaci:

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

Celem jest przeprowadzenie eksperymentów dla 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

2. Rozwiązanie problemu.

W celu rozwiązania problemu wykonujemy 40 iretacji wyżej podanego wyrażenia. Konluzję i wnioski wyciągniemy z iteracji graficznej zadanego równania.

3. Wyniki.

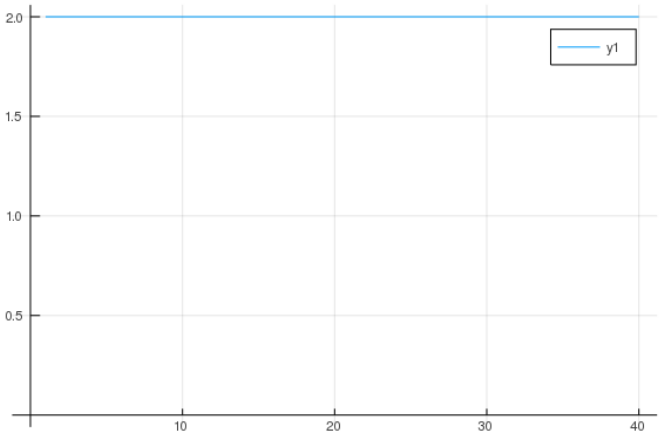
i	C= -2 x_0=1	C= -2 x_0=2	C= -2 x_0=1.9999999999999999	C= -1 x_0=1	C= -1 x_0=2	C= -1 x_0=0.75	C= -1 x_0=0.25
1	-1.0	2.0	1.99999999999996	0.0	0.0	-0.4375	-0.9375
2	-1.0	2.0	1.99999999999998401	-1.0	-1.0	-0.80859375	-0.12109375
3	-1.0	2.0	1.99999999999993605	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	2.0	1.9999999999974 42	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	-1.0	2.0	1.9999999999897682	0.0	0.0	-0.2253147218564956	-0.9991524699951226
6	-1.0	2.0	1.9999999999590727	-1.0	-1.0	-0.9492332761147301	-0.0016943417026455965
7	-1.0	2.0	1.999999999836291	0.0	0.0	-0.0989561875164966	-0.9999971292061947
8	-1.0	2.0	1.9999999993451638	-1.0	-1.0	-0.9902076729521999	-5.741579369278327e-6
9	-1.0	2.0	1.9999999973806553	0.0	0.0	-0.01948876442658909	-0.9999999996703423
10	-1.0	2.0	1.999999989522621	-1.0	-1.0	-0.999620188061125	-6.593148249578462e-11
11	-1.0	2.0	1.9999999580904841	0.0	0.0	-0.0007594796206411569	-1.0
12	-1.0	2.0	1.9999998323619383	-1.0	-1.0	-0.9999994231907058	0.0
13	-1.0	2.0	1.9999993294477814	0.0	0.0	-1.1536182557003727e-6	-1.0
14	-1.0	2.0	1.9999973177915749	-1.0	-1.0	-0.9999999999986692	0.0
15	-1.0	2.0	1.9999892711734937	0.0	0.0	-2.6616486792363503e-12	-1.0
16	-1.0	2.0	1.9999570848090	-1.0	-1.0	-1.0	0.0

			826				
17	-1.0	2.0	1.999828341078044	0.0	0.0	0.0	-1.0
18	-1.0	2.0	1.9993133937789613	-1.0	-1.0	-1.0	0.0
19	-1.0	2.0	1.9972540465439481	0.0	0.0	0.0	-1.0
20	-1.0	2.0	1.9890237264361752	-1.0	-1.0	-1.0	0.0
21	-1.0	2.0	1.9562153843260486	0.0	0.0	0.0	-1.0
22	-1.0	2.0	1.82677862987391	-1.0	-1.0	-1.0	0.0
23	-1.0	2.0	1.3371201625639997	0.0	0.0	0.0	-1.0
24	-1.0	2.0	-0.21210967086482313	-1.0	-1.0	-1.0	0.0
25	-1.0	2.0	-1.9550094875256163	0.0	0.0	0.0	-1.0
26	-1.0	2.0	1.822062096315173	-1.0	-1.0	-1.0	0.0
27	-1.0	2.0	1.319910282828443	0.0	0.0	0.0	-1.0
28	-1.0	2.0	-0.2578368452837396	-1.0	-1.0	-1.0	0.0
29	-1.0	2.0	-1.9335201612141288	0.0	0.0	0.0	-1.0
30	-1.0	2.0	1.7385002138215109	-1.0	-1.0	-1.0	0.0
31	-1.0	2.0	1.0223829934574389	0.0	0.0	0.0	-1.0
32	-1.0	2.0	-0.9547330146890065	-1.0	-1.0	-1.0	0.0
33	-1.0	2.0	-1.0884848706628412	0.0	0.0	0.0	-1.0
34	-1.0	2.0	-0.8152006863380978	-1.0	-1.0	-1.0	0.0
35	-1.0	2.0	-1.3354478409938944	0.0	0.0	0.0	-1.0
36	-1.0	2.0	-0.21657906398474625	-1.0	-1.0	-1.0	0.0
37	-1.0	2.0	-1.953093509043491	0.0	0.0	0.0	-1.0
38	-1.0	2.0	1.8145742550678174	-1.0	-1.0	-1.0	0.0
39	-1.0	2.0	1.2926797271549	0.0	0.0	0.0	-1.0

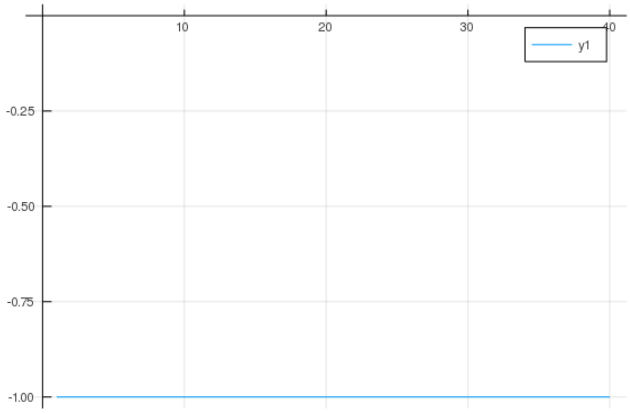
			244				
40	-1.0	2.0	-0.3289791230026702	-1.0	-1.0	-1.0	0.0

4. Wnioski.

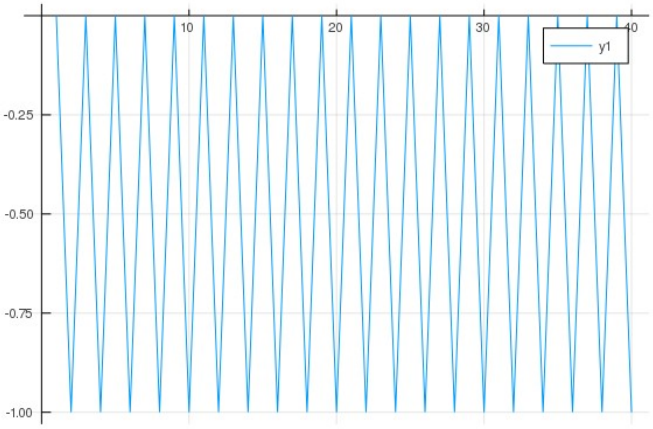
$X_0 = 1$ c = -2



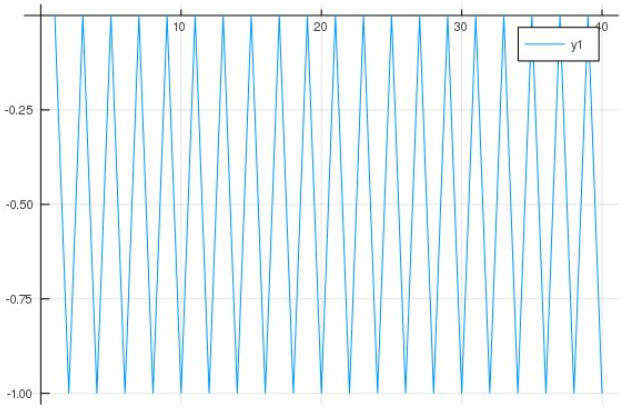
$x_0 = 2$ c = -2



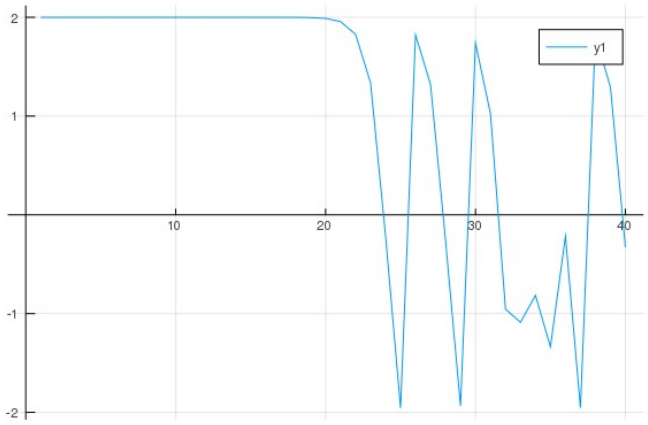
c = -1 x_0= -1



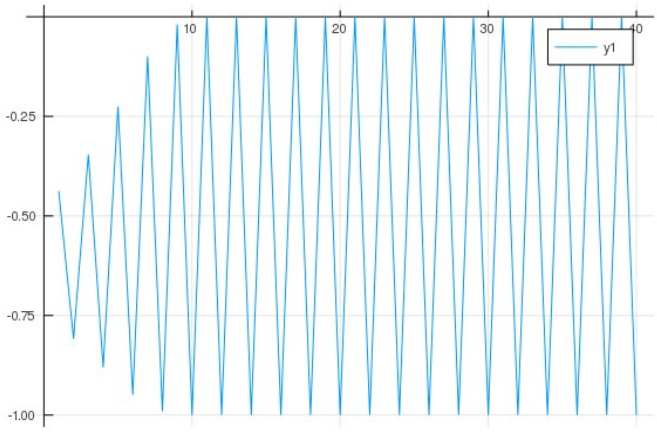
$x_0 = 1$ c = -1



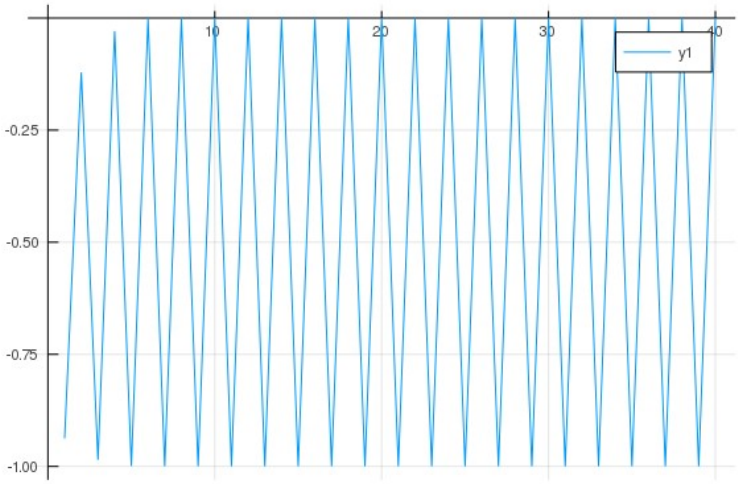
x_0= 1.99999999999999 c = -2



c = -1 x_0= 0.75



c = -1 x_0= 0.25



Dla pierwszych 4 wykresów wyniki są dość przewidywalne, zgodne.
W kolejnym przykładzie tj. dla x_o=1. 99999999999999 zauważalny jest fakt, iż z kolejną iteracją wyniki są coraz bardziej rozrzucone, niezgodne. Intuicja podpowiada, że x_0 zostanie zaokrąglony do 2 i otrzymamy rezultaty zbliżone to [2.0, 2.0, …, 2.0].
Dla x_o=0.75 v x_o=0.25 wartości po pewnej iteracji stabilizują się i oscylują po wartościach 0 oraz – 1.0 .
Sprawcą chaosu potencjalnie jest podnoszenie do kwadratu, ponieważ w wyniku np. błędu przybliżenia w pewnej iteracji w kolejnych błąd będzie wielokrotnie powtarzany. Jednak nasza analiza dowodzi, że wyniki są bezpośrednio i w dużym stopniu zależne od danych wejściowych $\mathbf{x_0}$ i **c**. Jest to również zadanie źle uwarunkowane, ponieważ dorbna zmiana może doprowadzić do otrzymania skrajnie błędnych wyników.