Obliczenia Naukowe

Lista 2

Jakub Gogola
236412

7 listopada 2018

1 Zadanie I

1.1 Problem

Zgodnie z poleceniem zadania należało ponownie wykonać zadanie 5. z listy 1., ale tym razem dokonując niewielkich zmian danych dla jednego z wektorów. Wektory X oraz Y z poprzedniej listy były następującej postaci:

$$X = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$$
$$Y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$$

Zmiana, której dokonano w przypadku tej listy, polegała na usunięciu ostatnich cyfr ze składowych x_4 i x_5 wektora X. Po dokonaniu tej zmiany, wektor X prezentował się następująco:

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

Wektor Y pozostał bez zmian.

Następnym krokiem było policzenie iloczynu skalarnego nowego wektora X oraz Y wykorzystując algorytmy z poprzedniej listy i zaobserwowanie, jak na otrzymane wyniki wpłynęły wprowadzone zmiany.

1.2 Algorytm

W zadaniu zostały użyte te same algorytmy, co w zadaniu 5. z poprzedniej listy. Dla przypomnienia:

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

1: $S \leftarrow 0$
2: **for** $i \leftarrow 1$ **to** n do **do**
3: $\quad S \leftarrow S + x_i \cdot y_i$
4: **end for**

(b) "w tył" - stosując algorytm $\sum_{i=n}^{1} + x_i y_i$

1: $S \leftarrow 0$
2: **for** $i \leftarrow 1$ **downto** n **do**
3: $\quad S \leftarrow S + x_i y_i$
4: **end for**

(c) od największego do najmniejszego (dodając liczby dodatnie w porządku malejącym, dodając liczby ujemne w porządku rosnącym i na koniec dodając do siebie obliczone sumy częściowe

(d) odwrotnie niż w punkcie **(c)**.

1.3 Wyniki

W rezultacie działania opisanych algorytmów dla zmienionych danych, otrzymano następujące wyniki:

(a) Arytmetyka `Float32`: -0.4999443
Arytmetyka `Float64`: -0.004296342739891585

(b) Arytmetyka `Float32`: -0.4543457
Arytmetyka `Float64`: -0.004296342998713953

(c) Arytmetyka `Float32`: -0.5
Arytmetyka `Float64`: -0.004296342842280865

(d) Arytmetyka `Float32`: -0.5
Arytmetyka `Float64`: -0.004296342842280865

W celu zauważenia analizy różnic wynikających ze zmian w danych wejściowych, poniżej znajdują się wyniki uzyskane w zadaniu 5. na poprzedniej liście:

(a) Arytmetyka `Float32`: -0.4999443
Arytmetyka `Float64`: $1.0251881368296672 \cdot 10^{-10}$

(b) Arytmetyka `Float32`: -0.4543457
Arytmetyka `Float64`: $-1.5643308870494366 \cdot 10^{-10}$

(c) Arytmetyka `Float32`: -0.5
Arytmetyka `Float64`: 0.0

(d) Arytmetyka `Float32`: -0.5
Arytmetyka `Float64`: 0.0

1.4 Obserwacje

Wyniki uzyskane dla arytmetyki `Float32` są takie same zarówno dla danych z poprzedniej listy, jak i dla tych z listy bieżącej. Dla arytmetyki `Float64` występują znaczące różnice w wynikach dla każdego z użytych algorytmów. Dla algorytmów **(a)** oraz **(b)** różnice są znaczące, ponieważ dla zmienionych danych wyniki różnią się od siebie aż o 10 rzędów wielkości.

1.5 Wnioski

Brak zmian w otrzymanych wynikach dla arytmetyki `Float32` wynika z względnie niewielkiej precyzji tejże. Powodem takiego stanu rzeczy jest fakt, że poszczególne składowe każdego z wektorów nie są przechowywane w zbyt dokładny sposób, a w konsekwencji - wyniki częściowe otrzymane podczas działania każdego z algorytmów **(a)** - **(d)** nie były zbyt dokładne, co finalnie spowodowało, że początkowe zmiany nie wpłynęły na wynik końcowy, ponieważ dokonano ich na pozycjach, które w przypadku obliczeń w arytmetyce `Float32` nie mają wielkiego znaczenia.

Sytuacja wygląda zupełnie inaczej w przypadku zastosowania do obliczeń arytmetyki `Float64`. Tutaj dokonanie zmian (usunięcie jednej z cyfr po przecinku w danych wejściowych) ma już wpływ na końcowy wynik. Dla przypadków **(a)** i **(b)** wyniki te różnią się o aż 10 rzędów wielkości od tych otrzymanych w zadaniu 5. na poprzedniej liście. Widać zatem, że algorytmy zastosowane do obliczania iloczynu skalarnego są bardzo wrażliwe na niewielkie zmiany w danych. Mamy zatem do czynienia z zadaniem źle uwarunkowanym. Na błędy w otrzymanych wynikach wpływa również fakt, iż wektory X i Y są prawie ortogonalne(prostopadłe).

2 Zadanie II

2.1 Problem

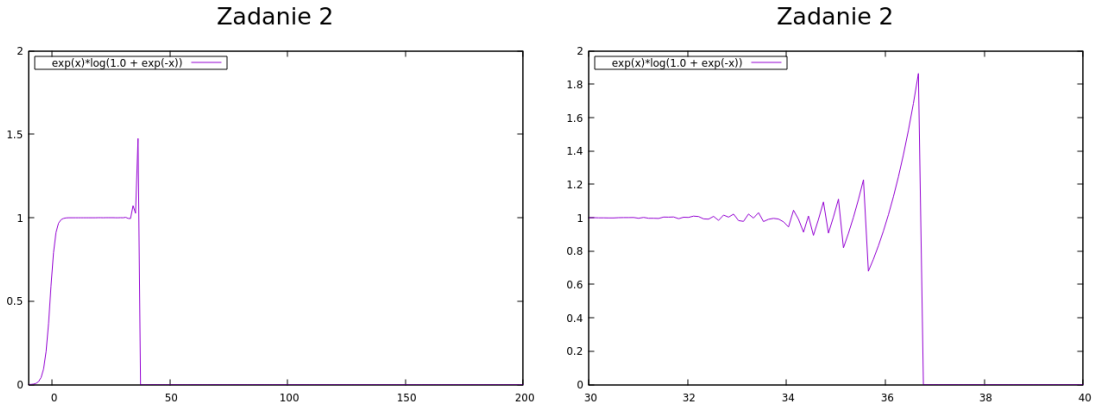
W zadaniu 2. należało, używając co najmniej dwóch różnych narzędzi do wizualizacji, wygenerować wykres funkcji $f(x) = e^x \ln(1 + e^{-x})$, a następnie, używając języka `Julia`, policzyć granicę $\lim_{x\to\infty} f(x)$ oraz porównać na końcu otrzymany wynik z wyrysowanymi przez programy wykresami funkcji f.

2.2 Algorytm

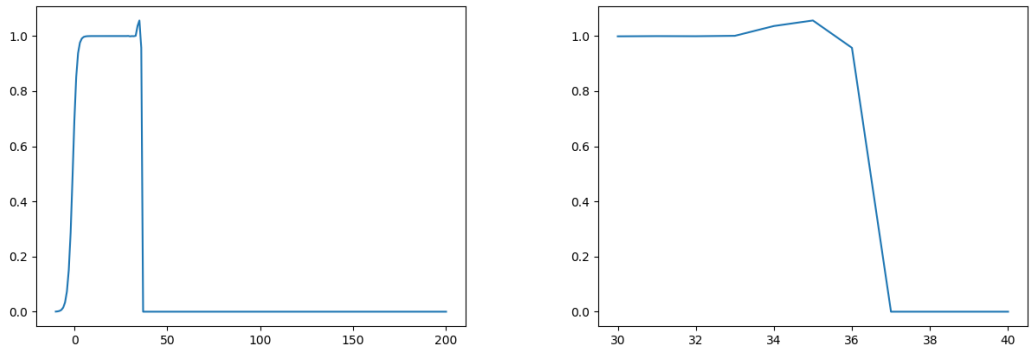
Do obliczenia granicy funkcji f została użyta bibliotek `SymPy` dla języka `Julia`.

2.3 Wyniki

W wyniku działania programu została obliczona granica dla funkcji f i wyniosła ona: $\lim_{x\to\infty} f(x) = 1$. Poniżej znajdują się również wygenerowane wykresy:



Rysunek 1: Wykresy wygenerowane za pomocą programu `GNUPlot`



Rysunek 2: Wykresy wygenerowane za pomocą pakietu `PyPlot`

2.4 Obserwacje

Na podstawie wygenerowanych wykresów można stwierdzić, że funkcja dla na odcinku $[10; 0]$ rośnie, następnie na odcinku $(0; 30]$ jest stała. Na przedziale $(30; 40]$ widać wyraźne oscylacje wykresu funkcji pomiędzy wartościami około $(0.6; 2)$, a dla $x > 40$ funkcja zaczyna zbiegać do wartości

0. Granica wynikająca z wykresów, do której zbiega funkcja f, jest różna od tej, która została obliczona.

2.5 Wnioski

Oscylacje wykresu na pewnym odcinku i grube odchylenia wartości od liczby 1.0 (do której funkcja f powinna zbiegać) wynikają z błędów, które powstają w wyniku mnożenia logarytmu naturalnego, którego wartość w tym przypadku jest bardzo mała przez bardzo dużą liczbę e^x względem niego. Powstają wtedy spore niedokładności w otrzymywanych wynikach. Zjawisko zbiegania f do 0 (dla $x > 40$) wynika natomiast z faktu, iż $\ln(1 + e^{-x}) \approx 0$. Spowodowane jest to tym, że e^{-x} jest bardzo małą wartością w stosunku do 1, więc $1 - e^{-x} \approx 1$, zatem wartość całej funkcji zbiega do 0. Zauważamy zatem, iż jest to kolejny przykład zadania źle uwarunkowanego.

3 Zadanie III

3.1 Problem

W zadaniu 3. należało rozwiązać układ równań liniowych w postaci $\mathbf{Ax} = \mathbf{b}$ dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n$. Macierz $\mathbf{A}$ była generowana w dwojaki sposób:

(a) $\mathbf{A} = \mathbf{H}_n$, gdzie $\mathbf{H}_n$ jest macierzą Hilberta stopnia n wygenerowaną za pomocą funkcji `A = hilb(n)` zdefiniowanej w dołączonym do zadania pliku,

(b) $\mathbf{A} = \mathbf{R}_n$, gdzie $\mathbf{R}_n$ jest macierzą stopnia n z zadanym wskaźnikiem uwarunkowania c wygenerowaną za pomocą funkcji `A = matcond(n,c)` zdefiniowanej w dołączonym do zadania pliku.

Zgodnie z poleceniem zadania, zadany układ równań należało rozwiązać za pomocą dwóch różnych metod:

(1) eliminacji Gaussa, czyli `x = A \ b`

(2) używając wzoru $x = \mathbf{A}^{-1}\mathbf{b}$, czyli, w języku Julia: `x = inv(A)*b`.

Ponadto, opisane wyżej eksperymenty należało przeprowadzić: dla macierzy Hilberta $\mathbf{H}_n$ z rosnącym stopniem $n > 1$ oraz dla macierzy losowej $\mathbf{R}_n$ dla $n \in \{5, 10, 20\}$ z rosnącym wskaźnikiem uwarunkowania $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$.

3.2 Algorytm

W języku `Julia` zaimplementowano funkcje `hilb_matrix()` oraz `random_matrix()`, z której każda generowała macierz determinowaną przez nazwę każdej z tych funkcji, rozwiązywała układ równań metodami **(1)** oraz **(2)** zgodnie z zadanymi warunkami i na końcu obliczała błędy powstałe podczas wykonywania obliczeń. Należy tutaj zaznaczyć, iż wszystkie obliczenia w tym zadaniu były wykonywane w arytmetyce `Float64`.

3.3 Wyniki

W wyniku działania algorytmów otrzymano następujące wyniki, które zaprezentowano w poniższych tabelach (są to błędy obliczone dla każdej z metod).

Wyniki dla zastosowania metody **(1)**:

n	`cond(A)`	Błąd dla metody eliminacji Gaussa	Błąd dla metody z odwrotnością macierzy
1	1.0	0.0	0.0
2	19.28147006790397	$5.66104886700367 \cdot 10^{-16}$	$1.404333387430680 \cdot 10^{-15}$
3	524.0567775860644	$8.02259377226772 \cdot 10^{-15}$	0.0
4	15513.73873892924	$4.13740962243038 \cdot 10^{-14}$	0.0
5	476607.25024259434	$1.682842629922719 \cdot 10^{-12}$	$3.354436058435963 \cdot 10^{-12}$
6	$1.495105864225466 \cdot 10^{7}$	$2.61891330231162 \cdot 10^{-10}$	$2.016375940434765 \cdot 10^{-10}$
7	$4.7536735658312 \cdot 10^{8}$	$1.260686722417154 \cdot 10^{-8}$	$4.71328039723203 \cdot 10^{-9}$
8	$1.525757553806004 \cdot 10^{10}$	$6.12408955572308 \cdot 10^{-8}$	$3.0774839030962 \cdot 10^{-7}$
9	$4.93153756446876 \cdot 10^{11}$	$3.875163418503247 \cdot 10^{-6}$	$4.54126830317664 \cdot 10^{-6}$
10	$1.602441699254171 \cdot 10^{13}$	$8.6703902370969 \cdot 10^{-5}$	0.0002501493411824886
11	$5.22267793928033 \cdot 10^{14}$	0.00015827808158590435	0.007618304284315809
12	$1.751473190709146 \cdot 10^{16}$	0.13396208372085344	0.258994120804705
13	$3.34414349733846 \cdot 10^{18}$	0.11039701117868264	5.331275639426837
14	$6.20078626316144 \cdot 10^{17}$	1.4554087127659643	8.71499275104814
15	$3.67439295346797 \cdot 10^{17}$	4.696668350857427	7.344641453111494
16	$7.86546777843164 \cdot 10^{17}$	54.15518954564602	29.84884207073541
17	$1.26368434266605 \cdot 10^{18}$	13.707236683836307	10.516942378369349
18	$2.244630992918912 \cdot 10^{18}$	9.134134521198485	7.575475905055309
19	$6.47195397654159 \cdot 10^{18}$	9.720589712655698	12.233761393757726
20	$1.355365790868822 \cdot 10^{18}$	7.549915039472976	22.062697257870493

Wyniki dla zastosowania metody **(2)**:

n	c	Błąd dla metody eliminacji Gaussa	Błąd dla metody Błąd dla metody z odwrotnością macierzy
5	1.0	$1.489520491948363 \cdot 10^{-16}$	$1.216188388897623 \cdot 10^{-16}$
5	10.0	$2.80866677486136 \cdot 10^{-16}$	0.0
5	1000.0	$8.90562200290678 \cdot 10^{-15}$	$1.32216329020653 \cdot 10^{-14}$
5	$1. \cdot 10^{7}$	$1.897153882444540 \cdot 10^{-10}$	$2.702114728380730 \cdot 10^{-10}$
5	$1. \cdot 10^{12}$	$3.149537434670237 \cdot 10^{-6}$	$1.092363057195766 \cdot 10^{-5}$
5	$1. \cdot 10^{16}$	0.08627458226764755	0.06987712429686843
10	1.0	$3.43990022795940 \cdot 10^{-16}$	$2.457583428003690 \cdot 10^{-16}$
10	10.0	$2.74204859601134 \cdot 10^{-16}$	$3.748544367384394 \cdot 10^{-16}$
10	1000.0	$9.55146748377680 \cdot 10^{-15}$	$1.456179286088797 \cdot 10^{-14}$
10	$1. \cdot 10^{7}$	$2.86819506455097 \cdot 10^{-10}$	$3.20919429481563 \cdot 10^{-10}$
10	$1. \cdot 10^{12}$	$7.8323491039955 \cdot 10^{-6}$	$6.008905030068726 \cdot 10^{-6}$
10	$1. \cdot 10^{16}$	0.06459023625268558	0.1128088209233879
20	1.0	$6.04536571471835 \cdot 10^{-16}$	$6.64280865943016 \cdot 10^{-16}$
20	10.0	$4.263892432392672 \cdot 10^{-16}$	$4.38502959679432 \cdot 10^{-16}$
20	1000.0	$4.28799551822986 \cdot 10^{-14}$	$4.01745365666494 \cdot 10^{-14}$
20	$1. \cdot 10^{7}$	$8.45568212845224 \cdot 10^{-12}$	$5.07788086257475 \cdot 10^{-11}$
20	$1. \cdot 10^{12}$	$2.423340464052032 \cdot 10^{-5}$	$2.17969222908066 \cdot 10^{-5}$
20	$1. \cdot 10^{16}$	0.21774410225511787	0.20868480280995097

3.4 Obserwacje

Zauważamy, że w przypadku macierzy Hilberta $\mathbf{H}_n$ błąd dla każdej z zastosowanych metod rośnie wprost proporcjonalnie do rozmiaru macierzy. Podobnie zachowuje się wskaźnik uwarunkowania. W przypadku macierzy losowej $\mathbf{R}_n$ Również mamy do czynienia ze wzrostem wskaźnika uwarunkowania i błędów w zależności od rozmiaru tej macierzy, jednak w tym przypadku wzrost ten jest znacznie wolniejszy niż w przypadku macierzy $\mathbf{H}_n$.

3.5 Wnioski

Wnioskiem z otrzymanych wyników jest fakt, iż to zadanie jest źle uwarunkowane dla macierzy Hilberta $\mathbf{H}_n$. W przypadku macierzy $\mathbf{R}_n$ też pojawiają się błędy wpływające na ostateczny wynik, lecz są one w większości pomijalnie małe.

4 Zadanie IV

4.1 Problem

W zadaniu 4. należało zbadać problem obliczania miejsc zerowych wielomianu zaproponowanego przez Wilkinsona. Wielomian dany był w dwóch postaciach - kanonicznej (naturalnej) **(1)** oraz iloczynowej **(2)**:

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

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

Zgodnie z poleceniem zadania (podpunkt **(a)**), wykorzystując bibliotekę `Polynomials` języka `Julia`, należało policzyć miejsca zerowa wielomianu za pomocą funkcji `roots`, a następnie sprawdzić otrzymane pierwiastki z_k, gdzie $(k \in [1; 20])$, obliczając $|P(z_k)|$, $|p(z_k)|$ i $|z_k - k|$ i wyjaśnić rozbieżności pomiędzy otrzymanymi wynikami. W tym celu należało zapoznać się z funkcjami `Poly`, `poly` i `polyval` z pakietu `Polynomials`. Następnie, stosując się do podpunktu **(b)**, należało powtórzyć eksperyment przeprowadzony przez Wilkinsona zmieniając współczynnik -210 na $-210 - 2^{-23}$ i wyjaśnić zjawisko.

4.2 Algorytm

Wykorzystując wymienione w poleceniu zadania funkcje z biblioteki `Polynomials` zostały wygenerowane wielomiany $P(x)$ (funkcja `Poly` - postać kanoniczna) oraz $p(x)$ (funkcja `poly` - postać iloczynowa). Następnie, za pomocą funkcji `roots`, policzono pierwiastki wielomianu $P(x)$, a dalszej kolejności, wykorzystując funkcję `polyval`, obliczono: $|P(z_k)|$, $|p(z_k)|$, $|z_k - k|$.

4.3 Wyniki

Wyniki dla danych z podpunktu **(a)** zadania:

k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	36352.0	38400.0	$3.010924842783424 \cdot 10^{-13}$
2	181760.0	198144.0	$2.831823664450894 \cdot 10^{-11}$
3	209408.0	301568.0	$4.079034887638499 \cdot 10^{-10}$
4	$3.10681 \cdot 10^{6}$	$2.84467 \cdot 10^{6}$	$1.62624682609191 \cdot 10^{-8}$
5	$2.411468 \cdot 10^{7}$	$2.334668 \cdot 10^{7}$	$6.65769791297066 \cdot 10^{-7}$
6	$1.2015206 \cdot 10^{8}$	$1.188249 \cdot 10^{8}$	$1.075417522677923 \cdot 10^{-5}$
7	$4.8039833 \cdot 10^{8}$	$4.7829094 \cdot 10^{8}$	0.00010200279300764947
8	$1.68269107 \cdot 10^{9}$	$1.6784972 \cdot 10^{9}$	0.0006441703922384079
9	$4.46532659 \cdot 10^{9}$	$4.45785958 \cdot 10^{9}$	0.002915294362052734
10	$1.270712678 \cdot 10^{10}$	$1.269690726 \cdot 10^{10}$	0.009586957518274986
11	$3.575989555 \cdot 10^{10}$	$3.574346905 \cdot 10^{10}$	0.025022932909317674
12	$7.21677158 \cdot 10^{10}$	$7.214665062 \cdot 10^{10}$	0.04671674615314281
13	$2.1572362905 \cdot 10^{11}$	$2.1569633075 \cdot 10^{11}$	0.07431403244734014
14	$3.6538325094 \cdot 10^{11}$	$3.65344793 \cdot 10^{11}$	0.08524440819787316
15	$6.1398775347 \cdot 10^{11}$	$6.1393841561 \cdot 10^{11}$	0.07549379969947623
16	$1.55502775193 \cdot 10^{12}$	$1.55496109721 \cdot 10^{12}$	0.05371328339202819
17	$3.77762377830 \cdot 10^{12}$	$3.77753294694 \cdot 10^{12}$	0.025427146237412046
18	$7.19955486105 \cdot 10^{12}$	$7.199447475 \cdot 10^{12}$	0.009078647283519814
19	$1.027837616281 \cdot 10^{13}$	$1.027823565670 \cdot 10^{13}$	0.0019098182994383706
20	$2.746295274547 \cdot 10^{13}$	$2.746278890700 \cdot 10^{13}$	0.00019070876336257925

Wyniki dla danych z podpunktu **(b)** zadania (wielomian z zaburzonym współczynnikiem):

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	$0.9999999999998357 + 0.0im$	20992.0	22016.0	$1.643130076445231 \cdot 10^{-13}$
2	$2.0000000000550373 + 0.0im$	349184.0	365568.0	$5.50373080443478 \cdot 10^{-11}$
3	$2.99999999660342 + 0.0im$	$2.22156 \cdot 10^{6}$	$2.29529 \cdot 10^{6}$	$3.396579906222996 \cdot 10^{-9}$
4	$4.000000089724362 + 0.0im$	$1.04678 \cdot 10^{7}$	$1.072998 \cdot 10^{7}$	$8.97243621622578 \cdot 10^{-8}$
5	$4.99999857388791 + 0.0im$	$3.946393 \cdot 10^{7}$	$4.330393 \cdot 10^{7}$	$1.426112089752962 \cdot 10^{-6}$
6	$6.000020476673031 + 0.0im$	$1.2914841 \cdot 10^{8}$	$2.0612044 \cdot 10^{8}$	$2.047667303095579 \cdot 10^{-5}$
7	$6.99960207042242 + 0.0im$	$3.8812313 \cdot 10^{8}$	$1.75767091 \cdot 10^{9}$	0.00039792957757978087
8	$8.007772029099446 + 0.0im$	$1.07254732 \cdot 10^{9}$	$1.852548659 \cdot 10^{10}$	0.007772029099445632
9	$8.915816367932559 + 0.0im$	$3.06557542 \cdot 10^{9}$	$1.3717431705 \cdot 10^{11}$	0.0841836320674414
10	$10.095455630535774 - 0.6449328236240688im$	$7.14311363803582 \cdot 10^{9}$	$1.491263381675401 \cdot 10^{12}$	0.6519586830380406
11	$10.095455630535774 + 0.6449328236240688im$	$7.14311363803582 \cdot 10^{9}$	$1.491263381675401 \cdot 10^{12}$	1.1109180272716561
12	$11.793890586174369 - 1.6524771364075785im$	$3.35775611317185 \cdot 10^{10}$	$3.296021414130166 \cdot 10^{13}$	1.665281290598479
13	$11.793890586174369 + 1.6524771364075785im$	$3.35775611317185 \cdot 10^{10}$	$3.296021414130166 \cdot 10^{13}$	2.045820276678428
14	$13.992406684487216 - 2.5188244257108443im$	$1.061206453308197 \cdot 10^{11}$	$9.54594159518366 \cdot 10^{14}$	2.5188358711909045
15	$13.992406684487216 + 2.5188244257108443im$	$1.061206453308197 \cdot 10^{11}$	$9.54594159518366 \cdot 10^{14}$	2.7128805312847097
16	$16.73074487979267 - 2.812624896721978im$	$3.31510347598176 \cdot 10^{11}$	$2.742089401676406 \cdot 10^{16}$	2.9060018735375106
17	$16.73074487979267 + 2.812624896721978im$	$3.31510347598176 \cdot 10^{11}$	$2.742089401676406 \cdot 10^{16}$	2.825483521349608
18	$19.5024423688181 - 1.940331978642903im$	$9.53942460981782 \cdot 10^{12}$	$4.252502487993469 \cdot 10^{17}$	2.454021446312976
19	$19.5024423688181 + 1.940331978642903im$	$9.53942460981782 \cdot 10^{12}$	$4.252502487993469 \cdot 10^{17}$	2.004329444309949
20	$20.84691021519479 + 0.0im$	$1.11445350451 \cdot 10^{13}$	$1.374373319724971 \cdot 10^{18}$	0.8469102151947894

4.4 Obserwacje

Wartości wielomianów $P(x)$ i $p(x)$ dla otrzymanych miejsc zerowych (zarówno dla **(a)**, jak i dla **(b)**), pomimo że te niewiele różnią się od tych, które powinny zostać otrzymane w rzeczywistości, są bardzo dalekie od wartości 0 (dla ostatniego pierwiastka wartość wielomianu jest rzędu 10^{13}). Wartości obliczonych błędów są niewielkie, rosną one jednak wprost proporcjonalnie do wartości dla danego pierwiastka. W przypadku zaburzenia jednego ze współczynników wielomianu, funkcja `roots` zwróciła pierwiastki zespolone

4.5 Wnioski

Wnioskiem z otrzymanych wyników w podpunkcie **(a)** jest stwierdzenie, że błędne wartości wynikają z ograniczeń zastosowanej arytmetyki. Wartości poszczególnych współczynników nie mogą być dokładnie przechowywane. Dla arytmetyki `Float64` bowiem możliwym jest zapisanie jedynie 15-17 cyfr znaczących w systemie dziesiętnym, co znacząco wpływa na jakoś otrzymanych wyników. Drugim wnioskiem, tym razem z podpunktu **(b)**, jest to, że jest to zadanie bardzo źle uwarunkowane pod względem szukania pierwiastka zadanego wielomianu. Zaburzenie jednego z jego współczynników o jedynie 2^{-23} spowodowało, że w rezultacie otrzymano pierwiastki zespolone

5 Zadanie V

5.1 Problem

W zadaniu 5. rozważaliśmy równanie rekurencyjne występujące w modelu logistycznym (modelu wzrostu populacji zaproponowanym przez Verhulsta) o następującej postaci:

$$p_{n+1} = p_n + rp_n(1 - p_n)$$

dla $n \in \mathbb{N}$, gdzie r jest pewną daną stałą, $r(1 - p_n$ jest czynnikiem wzrostu populacji, a p_0 jest wielkością populacji stanowiącą procent maksymalnej wielkości populacji dla danego stanu środowiska.

Należało przeprowadzić następujące eksperymenty:

1. Dla $p_0 = 0.01$ oraz $r = 3$ wykonać 40 iteracji danego wyżej wyrażenia, a następnie ponownie wykonać 40 iteracji tego wyrażenia, ale tym razem, po 10 pierwszych iteracjach, zastosować obcięcie 10. otrzymanego wyniku do 3 cyfr znaczących i kontynuować wykonywanie iteracji. Na końcu należało porównać otrzymane wyniki. Wszystkie obliczenia należało wykonać w arytmetyce `Float32` dla języka `Julia`.

2. Dla p_0 i r takich, jak w poprzednim podpunkcie, wykonać 40 iteracji danego wyrażenia dla arytmetyk `Float32` i `Float64`.

5.2 Algorytm

Dla pierwszego rodzaju iteracji zastosowano następujący algorytm:

```
1: function NORMAL-ITERATION
2:     p ← 0.01
3:     r ← 3
4:     for i ← 1 to 40 do
5:         p ← p + r · p · (1 − p)
6:         P[i] ← p
7:     end for
8:     return P
9: end function
```

Algorytm dla iteracji, gdzie wykonano obcięcie wyglądał następująco:

Zakłada się, że wszystkie obliczenia w powyższych algorytmach są wykonywane w zadanej wcześniej arytmetyce.

5.3 Wyniki

Przyjmijmy oznaczenia:

I - 40 iteracji bez żadnych zmian w danych

II - 40 iteracji z obcięciem p_{n+1} do 3 cyfr znaczących po wykonaniu 10 pierwszych iteracji

```
1: function ITERATION-WITH-TRUNCATION
2:     p ← 0.01
3:     r ← 3
4:     for i ← 1 to 10 do
5:         p ← p + r · p · (1 − p)
6:         if i = 10 then
7:             p ← TRUNC(p, 3)
8:         end if
9:         P[i] ← p
10:    end for
11:    for i ← 11 to 40 do
12:        p ← p + r · p · (1 − p)
13:        P[i] ← p
14:    end for
15:    return P
16: end function
```

Dla podpunktu 1. otrzymano następujące wyniki:

n	**I**	**II**
1	0.0397	0.0397
2	0.15407173	0.15407173
3	0.5450726	0.5450726
4	1.2889781	1.2889781
5	0.1715188	0.1715188
6	0.5978191	0.5978191
7	1.3191134	1.3191134
8	0.056273222	0.056273222
9	0.21559286	0.21559286
10	0.7229306	0.722
11	1.3238364	1.3241479
12	0.037716985	0.036488414
13	0.14660022	0.14195944
14	0.521926	0.50738037
15	1.2704837	1.2572169
16	0.2395482	0.28708452
17	0.7860428	0.9010855
18	1.2905813	1.1684768
19	0.16552472	0.577893
20	0.5799036	1.3096911
21	1.3107498	0.09289217
22	0.088804245	0.34568182
23	0.3315584	1.0242395
24	0.9964407	0.94975823
25	1.0070806	1.0929108
26	0.9856885	0.7882812
27	1.0280086	1.2889631
28	0.9416294	0.17157483
29	1.1065198	0.59798557
30	0.7529209	1.3191822
31	1.3110139	0.05600393
32	0.0877831	0.21460639
33	0.3280148	0.7202578
34	0.9892781	1.3247173
35	1.021099	0.034241438
36	0.95646656	0.13344833
37	1.0813814	0.48036796
38	0.81736827	1.2292118
39	1.2652004	0.3839622
40	0.25860548	1.093568

Wyniki dla podpunktu 2. zadania prezentują się następująco:

n	Float32	Float64
1	0.0397	0.0397
2	0.15407173	0.15407173000000002
3	0.5450726	0.5450726260444213
4	1.2889781	1.2889780011888006
5	0.1715188	0.17151914210917552
6	0.5978191	0.5978201201070994
7	1.3191134	1.3191137924137974
8	0.056273222	0.056271577646256565
9	0.21559286	0.21558683923263022
10	0.7229306	0.722914301179573
11	1.3238364	1.3238419441684408
12	0.037716985	0.03769529725473175
13	0.14660022	0.14651838271355924
14	0.521926	0.521670621435246
15	1.2704837	1.2702617739350768
16	0.2395482	0.24035217277824272
17	0.7860428	0.7881011902353041
18	1.2905813	1.2890943027903075
19	0.16552472	0.17108484670194324
20	0.5799036	0.5965293124946907
21	1.3107498	1.3185755879825978
22	0.088804245	0.058377608259430724
23	0.3315584	0.22328659759944824
24	0.9964407	0.7435756763951792
25	1.0070806	1.315588346001072
26	0.9856885	0.07003529560277899
27	1.0280086	0.26542635452061003
28	0.9416294	0.8503519690601384
29	1.1065198	1.2321124623871897
30	0.7529209	0.37414648963928676
31	1.3110139	1.0766291714289444
32	0.0877831	0.8291255674004515
33	0.3280148	1.2541546500504441
34	0.9892781	0.29790694147232066
35	1.021099	0.9253821285571046
36	0.95646656	1.1325322626697856
37	1.0813814	0.6822410727153098
38	0.81736827	1.3326056469620293
39	1.2652004	0.0029091569028512065
40	0.25860548	0.011611238029748606

5.4 Obserwacje

Zauważamy, że w przypadku pierwszej części zadania, wyniki znacznie się od siebie różnią, w zależności od tego, czy dokonano obcięcia p_{n+1} po pierwszych 10 iteracjach algorytmu. W drugiej części ponownie widzimy różnicę w otrzymywanych wynikach w zależności od zastosowanej arytmetyki.

5.5 Wnioski

Analizując otrzymane wyniki można wnioskować, że obcięcie pewnej liczby cyfr znaczących znacząco wpływa na dalsze wyniki. Raz popełniony błąd niedokładności nawarstwia się i propaguje na kolejne wartości dla ciągu, bowiem każda kolejna wartość zależy od poprzedniej. Takie zjawisko nazywamy **sprzężeniem zwrotnym**. Otrzymywane wartości są tak samo pewne, jak te zwracane przez generator losowy. Druga część zadania miała na celu pokazanie, jak precyzja zastosowanej

arytmetyki wpływa na uzyskiwane wyniki. Zauważamy, że od pewnego momentu znacznie się one różnią, w zależności od tego, czy obliczenia wykonywane są w arytmetyce `Float32`, czy `Float64` (zmiana jest już dobrze widoczna od około 18-20 iteracji algorytmu). Brak korelacji pomiędzy wynikami dla każdej z wybranych arytmetyk obrazuje tzw. pojęcie *chaosu deterministycznego*. Mamy tutaj do czynienia ze zjawiskiem *czułej zależności od warunków początkowych*.

6 Zadanie VI

6.1 Problem

W zadaniu należało rozważyć następujące równanie zadane w postaci rekurencyjnej:

$$x_{n+1} = x_n^2 + c$$

dla $n \in \mathbb{N}$, gdzie c jest pewną stałą. Należało przeprowadzić, używając języka `Julia` i arytmetyki `Float64`, następujące eksperymenty poprzez wykonanie 40 iteracji dla następujących 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 $x_0 = 0.25$

Następnie należało przeanalizować zachowanie wygenerowanych ciągów.

6.2 Algorytm

W zadaniu użyto następującej funkcji do policzenia wartości dla kolejnych x_{n+1}:

```
1: function CALC-FUNCTION(c, x_0)
2:     prev ← x_0
3:     for i ← 1 to 40 do
4:         x ← prev^2 + c
5:         Y[i] ← x
6:         prev ← x
7:     end for
8:     return Y
9: end function
```

6.3 Wyniki

Na podstawie danych otrzymanych w wyniku działania powyższego algorytmu, wygenerowano następujące wykresy przedstawione na rysunku 3.

Ponadto, w poniższej tabeli zaprezentowane są wyniki dla każdej z 40 iteracji w zależności od wartości x_0 i c. Numery kolumn, podane w 1. wierszu tabeli, oznaczają konkretny z zestawów danych, które zostały opisane w podpunkcie **6.1**.

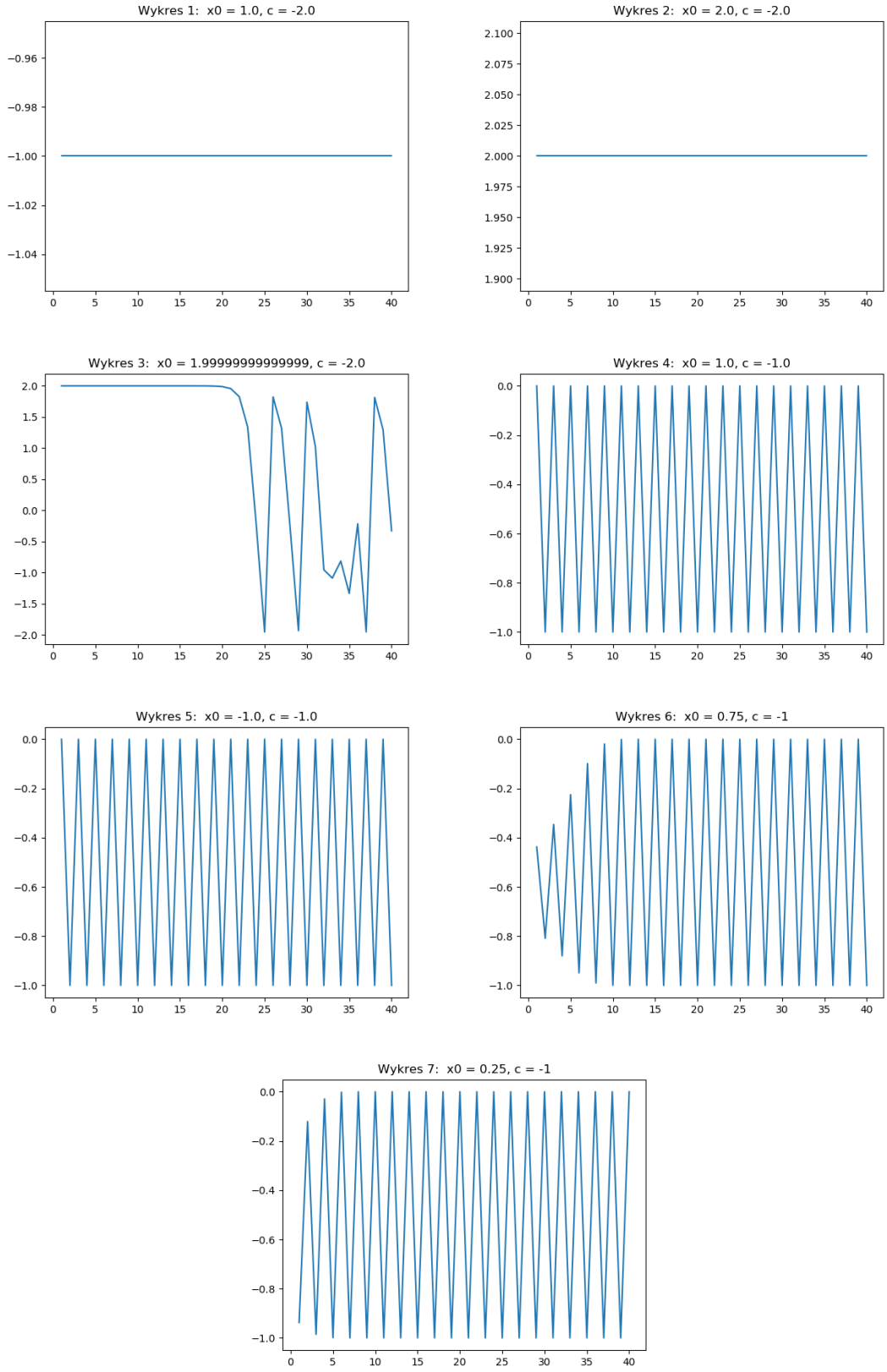
n	1	2	3	4	5	6	7
1	-1.0	2.0	1.99999999999996	0.0	0.0	-0.4375	-0.9375
2	-1.0	2.0	1.9999999999998401	-1.0	-1.0	-0.80859375	-0.12109375
3	-1.0	2.0	1.9999999999993605	0.0	0.0	-0.3461761474609375	-0.9853363037109375
4	-1.0	2.0	1.999999999997442	-1.0	-1.0	-0.8801620749291033	-0.029112368589267135
5	-1.0	2.0	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	-0.9999999999670343
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	1.9999570848090826	-1.0	-1.0	-1.0	0.0
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	1.2926797271549244	0.0	0.0	0.0	-1.0
40	-1.0	2.0	-0.3289791230026702	-1.0	-1.0	-1.0	0.0

6.4 Obserwacje

Zauważmy, że dla wykresów nr 4, 5, 6 i 7 widzimy regularne wahania wartości funkcji na przedziale wartości $[-1; 0]$. W przypadku wykresów 1 i 2 mamy linię prostą, a na wykresie 3 widoczne są losowe oscylacje. Na podstawie przedstawione tabeli widzimy, że dla danych 4. i 5. mamy do czynienia z ciągami, których podciągi zbiegają odpowiednio: do 0 dla n nieparzystych i do -1 dla n parzystych. W obu przypadkach $c = -1$. W przypadku zestawu danych 6. mamy podobną sytuację, aczkolwiek wartości równe dokładnie 0 i 1 otrzymujemy dopiero dla $n = 16$. Zauważamy, że i w tym przypadku $c = -1$. Dla ciągu 7. zauważamy, że mamy sytuację odwrotną, to znaczy, że dla n nieparzystych wartości zbiegają do -1, a dla parzystych do 0 (c ponownie wynosi -1). W przypadku wykresu 3. obserwujemy mocno losowe zachowanie wartości ($x_0 = 1.99999999999999$). W przypadkach 1. i 2. otrzymujemy ciągi stałe o wartościach równych odpowiednio -1 i 2 (tutaj $c = -1$).

6.5 Wnioski

Zauważamy, że zadanie 6. jest dosyć podobne do zadania 5., ponieważ i tutaj mamy do czynienia ze sprzężeniem zwrotnym. Analizując otrzymane wyniki zauważamy, że można posłużyć się w tym przypadku pojęciem *układu stabilnego*. Stabilność układu określa to, jak pewien układ matematyczny jest wrażliwy na zmiany w danych. Zauważamy, że dla danych 6. i 7. mamy do czynienia z układem stabilnym (x_0 wyniosło odpowiednio 0.75 i 0.25), ponieważ chociaż na początku mieliśmy do czynienia z dosyć losowymi wartościami, to jednak odpowiednie podciągi zbiegają do 0 i -1 i dla pewnego n przyjmują już stale jedną z tych wartości. Dla 3. zestawu danych (przypomnijmy: $c = -2$ i $x_0 = 1.99999999999999$) mamy układ niestabilny. Powyżej pewnego n wartości ciągu są kompletnie losowe i "rozrzucone" w losowe miejsca danej przestrzeni liczbowej.



Rysunek 3: Wykresy wykonano za pomocą pakietu `PyPlot`