Obliczenia naukowe **2017/2018**

Prowadzący: dr hab. Paweł Zieliński czwartek TN, 11:15

Agata Jasionowska 229726

Laboratorium – Lista 2

1. Zadanie 1

1.1. Opis problemu

Ponowne rozwiązanie zadania 5 z listy 1, jednak z wykorzystaniem nieznacznie zmienionych danych (usunięcie ostatnich cyfr w x_4 oraz x_5). Ta modyfikacja prezentuje się następująco:

$$x_4 = 0.5772156649 \quad x'_4 = 0.577215664$$

$$x_5 = 0.3010299957 \quad x'_5 = 0.301029995$$

W dalszej części sprawozdania wartości iloczynu skalarnego dla niezmienionych danych oznaczane będą jako I, zaś dla danych po modyfikacji – I'.

1.2. Opis rozwiązania

W celu obliczenia iloczynów skalarnych użyto kodu zadania 5 listy 1 na zmodyfikowanych zgodnie z treścią zadania danych wejściowych.

1.3. Wyniki

Tabela 1 prezentuje uzyskane wyniki dla czterech algorytmów, zestawiając rozwiązania dla I oraz I':

1.4. Wnioski

Uzyskane wyniki pokazują, że usunięcie cyfr zgodnie z poleceniem nie wpłynęło na rezultaty dla obliczeń w arytmetyce `Float32`. Wynika to ze stosunkowo niskiej precyzji obliczeń w przypadku tej arytmetyki. Za poparcie tego stwierdzenia może posłużyć analiza zapisu bitowego zmienionych wartości – reprezentacja x_4 oraz x'_4 wygląda tu identycznie, zaś dla x_5 różnica pojawia się dopiero na najmniej znaczącym bicie.Zupełnie odmienna sytuacja ma miejsce w przypadku arytmetyki `Float64` – rozbieżności są bardzo wyraźne, co wynika ze zwiększonej dokładności. Ma tu zastosowanie podwójna precyzją, czyli 15-17 cyfr znaczących w zapisie dziesiętnym, zaś usunięcie choćby jednej z nich umożliwia przechowanie dokładniejszego wyniku. Ostatecznie rezultaty nie pokrywają się z oczekiwanymi; przyczynia się do tego również fakt,

podpunkt	I	I'
	Float32	
1	$-0.499\,944\,3$	$-0.499\,944\,3$
2	$-0.454\,345\,7$	$-0.454\,345\,7$
3	-0.5	-0.5
4	-0.5	-0.5
	Float64	
1	$-0.499\,944\,3$	$-0.004\,296\,342\,739\,891\,585$
2	$-0.454\,345\,7$	$-0.004\,296\,342\,998\,713\,953$
3	-0.5	$-0.004\,296\,342\,842\,280\,865$
4	-0.5	$-0.004\,296\,342\,842\,280\,865$

Tabela 1: Iloczyn skalarny wektorów.

iż dane wektory są prawie prostopadłe(ortogonalne). Głównym problemem w przypadku tego zadania jest niestabilność zastosowanych algorytmów, przy których wskazane jest zastosowanie jak największej precyzji obliczeń. Nieznaczne zmiany wpływają na duże błędy końcowe, co potwierdza, że zadanie jest źle uwarunkowane.

2. Zadanie 2

2.1. Opis problemu

W co najmniej dwóch wybranych programach do wizualizacji narysować wykres funkcji $f(x) = e^x \ln(1 + e^{-x})$ oraz policzyć granicę $\lim_{x\to\infty} f(x)$

2.2. Opis rozwiązania

Wykresy zostały narysowane z użyciem programu `GNUPlot`, pakietu `Gadfly` oraz `Wolfram Alpha Cloud`, zaś granicę policzono przy pomocy funkcji `limit` języka `Julia` (`SymPy`).

2.3. Wyniki

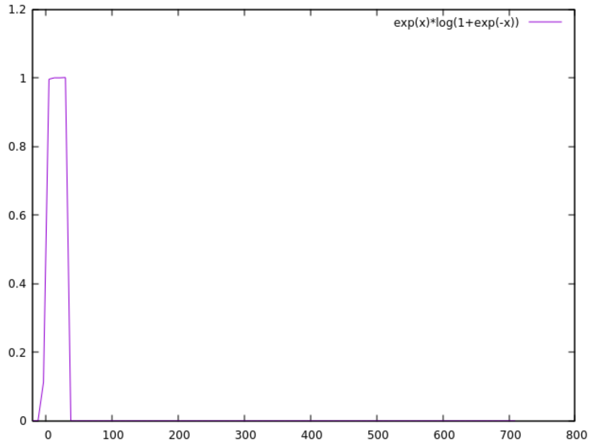
Obliczona granica równa jest:

$$\lim_{x\to\infty} f(x) = 1$$

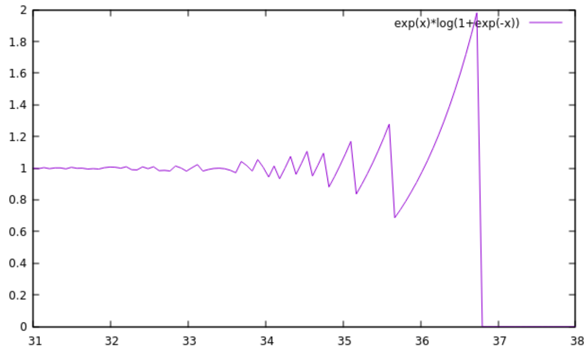
Uzyskane wykresy przedstawione są na Rysunkach 1, 2 oraz 3.

2.4. Wnioski

Wykresy obrazują dość silną oscylację funkcji $f(x)$ dla pewnych argumentów $x > 31$ - można zauważyć przyjmowanie wartości większych niż 1 (niezgodność z wyliczoną granicą). Przyczyną takiego zachowania jest wykonywanie mnożenia bardzo małego logarytmu oraz dużej liczby $e(x)$, w efekcie czego generowane są znaczne błędy. Od pewnego x funkcja osiąga stałą wartość równą 0. Powód stanowi bardzo już niewielka wartość $e(-x)$, co przekłada się na $\ln(1+e(-x)) = \ln(1) = 0$. Również ten problem jest źle uwarunkowany – małe zmiany danych wpłynęły na duże odchylenia.

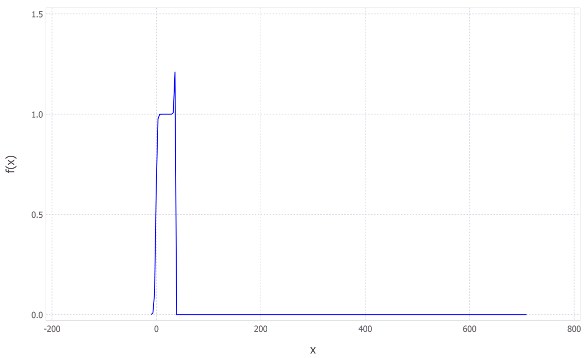


(a) $x \in (-10, 800)$

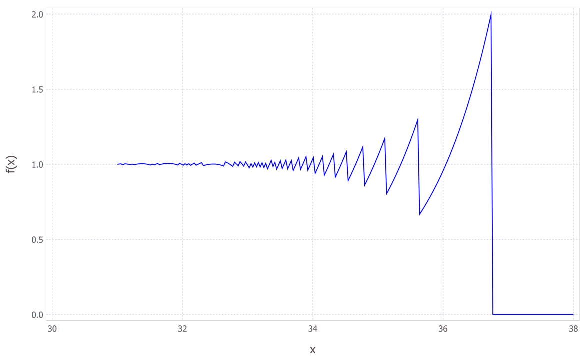


(b) $x \in (31, 38)$

Rysunek 1: Wykresy w `GNUPlot`.

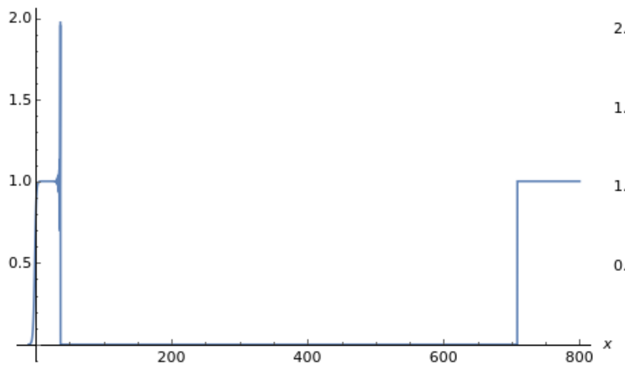


(a) $x \in (-10, 800)$

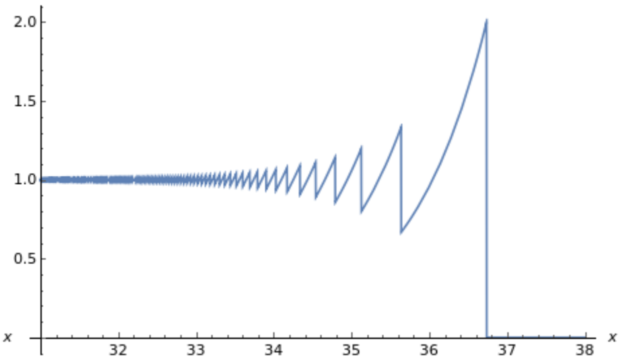


(b) $x \in (31, 38)$

Rysunek 2: Wykresy w `Gadfly`.



(a) $x \in (-10, 800)$



(b) $x \in (31, 38)$

Rysunek 3: Wykresy w `Wolfram Alpha Cloud`.

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 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;
(b) $A = R_n$, gdzie R_n jest losową macierzą stopnia n o zadanym wskaźniku uwarunkowania.

Przeprowadzenie 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 $c = 1, 10, 10^3, 10^7, 10^{12}, 10^{16}$, a także obliczenie błędów względnych.

3.2. Opis rozwiązania

W celu rozwiązania zadanego problemu utworzono w języku `Julia` funkcje konstruujące macierze (do wygenerowania macierzy Hilberta n–stopnia użyto `hilb(`n`)`, zaś do losowej – `matcond(c,`n`)` z dołączonych plików) i obliczające błędy względne dla dwóch podanych algorytmów.

3.3. Wyniki

Uzyskane rezultaty prezentują: Tabela 2 oraz Tabela 3.

n	`cond(`n`)`	Eliminacja Gaussa	Odwrotność macierzy
1	1.0	0.0	0.0
2	$1.928\,147\,006\,790\,397 \times 10^{1}$	$5.661\,048\,867\,003\,676 \times 10^{-16}$	$1.404\,333\,387\,430\,680\,3 \times 10^{-15}$
3	$5.240\,567\,775\,860\,644 \times 10^{2}$	$8.022\,593\,772\,267\,726 \times 10^{-15}$	0.0
4	$1.551\,373\,873\,892\,813\,8 \times 10^{4}$	$3.861\,787\,750\,251\,888 \times 10^{-13}$	$1.156\,679\,772\,130\,055 \times 10^{-13}$
5	$4.766\,072\,502\,417\,229\,7 \times 10^{5}$	$3.493\,384\,412\,947\,934 \times 10^{-13}$	$1.344\,084\,245\,058\,996\,3 \times 10^{-11}$
6	$1.495\,105\,864\,074\,776\,7 \times 10^{7}$	$3.106\,906\,352\,528\,99 \times 10^{-10}$	$2.191\,058\,908\,461\,152\,7 \times 10^{-10}$
7	$4.753\,673\,563\,686\,169 \times 10^{8}$	$1.408\,888\,818\,524\,629\,6 \times 10^{-8}$	$1.226\,103\,013\,648\,726\,2 \times 10^{-8}$
8	$1.525\,757\,525\,890\,056\,8 \times 10^{10}$	$2.256\,332\,083\,631\,954\,3 \times 10^{-7}$	$3.317\,498\,547\,889\,283 \times 10^{-7}$
9	$4.931\,544\,721\,936\,073\,6 \times 10^{11}$	$1.012\,295\,110\,413\,279 \times 10^{-5}$	$1.024\,503\,299\,281\,208\,5 \times 10^{-5}$
10	$1.602\,528\,535\,225\,018 \times 10^{13}$	$1.658\,285\,788\,577\,287 \times 10^{-4}$	$3.337\,389\,467\,051\,355\,4 \times 10^{-4}$
11	$5.225\,191\,290\,892\,983 \times 10^{14}$	$5.615\,377\,523\,997\,018 \times 10^{-3}$	$7.421\,014\,394\,851\,9 \times 10^{-3}$
12	$1.802\,560\,972\,707\,573\,2 \times 10^{16}$	$1.811\,305\,915\,033\,683\,8 \times 10^{-1}$	$2.066\,941\,820\,551\,894\,7 \times 10^{-1}$
13	$1.172\,202\,398\,838\,290\,2 \times 10^{18}$	$2.114\,212\,205\,460\,056$	$6.279\,218\,791\,835\,776 \times 10^{-1}$
14	$4.892\,374\,250\,062\,484 \times 10^{17}$	$1.101\,747\,227\,182\,980\,6 \times 10^{1}$	$4.415\,221\,385\,894\,176\,4 \times 10^{1}$
15	$1.105\,429\,325\,591\,602\,2 \times 10^{18}$	$2.936\,476\,333\,847\,876\,3$	$4.600\,555\,824\,279\,512$
16	$4.332\,109\,783\,660\,183\,7 \times 10^{17}$	$2.104\,258\,224\,602\,164$	$1.551\,741\,111\,529\,332 \times 10^{1}$
17	$6.912\,464\,846\,448\,458 \times 10^{17}$	$4.618\,693\,259\,670\,767$	$5.902\,966\,643\,537\,722$
18	$2.138\,620\,578\,491\,885\,8 \times 10^{18}$	$5.835\,620\,352\,091\,348\,6$	$2.736\,241\,550\,426\,244 \times 10^{1}$
19	$5.762\,759\,578\,133\,561\,6 \times 10^{17}$	$7.775\,664\,924\,181\,054$	$2.853\,711\,780\,136\,981 \times 10^{1}$
20	$2.638\,328\,071\,362\,124 \times 10^{18}$	$1.843\,691\,894\,440\,051 \times 10^{1}$	$1.827\,476\,266\,412\,702\,8 \times 10^{1}$

Tabela 2: Wskaźnik uwarunkowania oraz błędy względne dla macierzy Hilberta.

3.4. Wnioski

W przypadku macierzy Hilberta błąd rośnie wraz ze zwiększającą się wartością `cond` (co bezpośrednio związane jest z rosnącym stopniem). Błędy te są jednak większe dla algorytmu odwrotności macierzy (nie jest on zalecany z numerycznego punktu widzenia). Z kolei dla macierzy generowanej w sposób losowy ewidentnie występuje wzrost błędów w miarę zwiększania

n	c	Eliminacja Gaussa	Odwrotność macierzy
5	1.0	$9.930\,136\,612\,989\,092 \times 10^{-17}$	$1.790\,180\,836\,524\,723\,8 \times 10^{-16}$
5	10	$3.140\,184\,917\,367\,550\,3 \times 10^{-16}$	$3.439\,900\,227\,959\,406 \times 10^{-16}$
5	10^3	$1.912\,137\,988\,767\,086 \times 10^{-14}$	$2.383\,232\,787\,117\,382 \times 10^{-14}$
5	10^7	$3.398\,104\,980\,251\,079\,3 \times 10^{-10}$	$3.389\,065\,765\,777\,984 \times 10^{-10}$
5	10^{12}	$7.442\,503\,412\,124\,822\,6 \times 10^{-6}$	$6.716\,467\,613\,853\,077 \times 10^{-6}$
5	10^{16}	$2.554\,444\,890\,513\,606\,3 \times 10^{-1}$	$1.767\,766\,952\,966\,369 \times 10^{-1}$
10	1	$3.764\,949\,453\,935\,610\,6 \times 10^{-16}$	$2.673\,771\,110\,915\,334 \times 10^{-16}$
10	10	$3.845\,925\,372\,767\,127\,6 \times 10^{-16}$	$3.255\,813\,018\,879\,823 \times 10^{-16}$
10	10^3	$1.327\,401\,517\,070\,473\,3 \times 10^{-14}$	$1.331\,675\,378\,963\,219\,5 \times 10^{-14}$
10	10^7	$2.156\,241\,790\,236\,639\,5 \times 10^{-10}$	$1.909\,575\,041\,188\,421\,6 \times 10^{-10}$
10	10^{12}	$2.305\,968\,709\,493\,362\,3 \times 10^{-5}$	$1.899\,703\,918\,150\,136 \times 10^{-5}$
10	10^{16}	$1.657\,892\,355\,632\,386 \times 10^{-2}$	$4.004\,278\,398\,291\,985\,5 \times 10^{-2}$
20	1	$5.032\,874\,986\,385\,111 \times 10^{-16}$	$5.135\,906\,591\,666\,907 \times 10^{-16}$
20	10	$3.698\,892\,580\,885\,111\,6 \times 10^{-16}$	$3.179\,194\,921\,382\,489\,4 \times 10^{-16}$
20	10^3	$2.077\,097\,917\,151\,743\,5 \times 10^{-14}$	$1.903\,687\,736\,627\,341 \times 10^{-14}$
20	10^7	$2.649\,928\,100\,878\,158\,6 \times 10^{-10}$	$2.189\,265\,400\,192\,323\,7 \times 10^{-10}$
20	10^{12}	$4.575\,827\,403\,624\,591 \times 10^{-5}$	$4.661\,848\,415\,190\,589 \times 10^{-5}$
20	10^{16}	$7.272\,214\,577\,202\,413 \times 10^{-2}$	$6.411\,153\,265\,985\,08 \times 10^{-2}$

Tabela 3: Wskaźnik uwarunkowania oraz błędy względne dla macierzy losowej.

się stopnia oraz wskaźnika uwarunkowania. Prowadzi to do konkluzji, iż wykonywanie obliczeń na macierzach o dużym wskaźniku uwarunkowania doprowadza do generowania dużych błędów obliczeniowych. Można by rzec, iż "złośliwość" macierzy jest wprost proporcjonalna do wskaźnika uwarunkowania, co prowadzi do wniosku, iż zadanie to jest źle uwarunkowane.

4. Zadanie 4

4.1. Opis problemu

Przeprowadzenie eksperymentu obliczania zer wielomianu Wilkinsona, zapisanego pod dwiema następującymi postaciami:

$$P(x) = x^{20} - 210x^{19} + 20615x^{18} \dots$$

$$p(x) = (x-20)(x-19)(x-18)\dots(x-2)(x-1)$$

oraz określenie błędu bezwzględnego wartości tego wielomianu dla otrzymanych pierwiastków. Następnie powtórzenie testu dla współczynnika przy x^{19} zmienionego na $-210 - 2^{-23}$.

4.2. Opis rozwiązania

Rozwiązanie zadania przygotowano w języku `Julia` z użyciem trzech funkcji dostępnych w pakiecie `Polynomials`. Przebiega ono zgodnie z poniższymi krokami:

1. Utworzenie wielomianu na podstawie podanych współczynników w postaci kanonicznej (funkcja `Poly`), a także iloczynowej (funkcja `poly`);
2. Obliczenie pierwiastków wielomianu z_k, $1 \le k \le 20$ przy użyciu funkcji `roots`;
3. Prezentacja błędu bezwzględnego obu postaci wielomianu oraz miejsc zerowych.

4.3. Wyniki

Otrzymane wyniki zaprezentowano w Tabelach 4 oraz 5.

k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	2.2016×10^{4}	2.2016×10^{4}	$1.7530421558831222 \times 10^{-13}$
2	1.11104×10^{5}	1.11104×10^{5}	$1.8059331807762646 \times 10^{-11}$
3	2.33472×10^{5}	2.33472×10^{5}	$1.301203589321176 \times 10^{-10}$
4	3.204096×10^{6}	3.204096×10^{6}	$1.9510343118867013 \times 10^{-8}$
5	1.1813888×10^{7}	1.1813888×10^{7}	$2.7114605316569396 \times 10^{-7}$
6	1.39264×10^{7}	1.39264×10^{7}	$3.6291786109643454 \times 10^{-7}$
7	1.32373504×10^{8}	1.32373504×10^{8}	$2.1406434954407416 \times 10^{-5}$
8	4.73758208×10^{8}	4.73758208×10^{8}	$2.4147412876196483 \times 10^{-4}$
9	$1.653683712 \times 10^{9}$	$1.653683712 \times 10^{9}$	$1.4020327864088244 \times 10^{-3}$
10	$8.700433408 \times 10^{9}$	$8.700433408 \times 10^{9}$	$5.402181223315594 \times 10^{-3}$
11	$1.7547646464 \times 10^{10}$	$1.7547646464 \times 10^{10}$	$1.4432632050114691 \times 10^{-2}$
12	$5.0825762816 \times 10^{10}$	$5.0825762816 \times 10^{10}$	$2.990002384475332 \times 10^{-2}$
13	$9.3423894528 \times 10^{10}$	$9.3423894528 \times 10^{10}$	$4.396429964597992 \times 10^{-2}$
14	$2.41390764032 \times 10^{11}$	$2.41390764032 \times 10^{11}$	$5.062781545541739 \times 10^{-2}$
15	$3.71354137088 \times 10^{11}$	$3.71354137088 \times 10^{11}$	$4.358785452156155 \times 10^{-2}$
16	$1.139728016896 \times 10^{12}$	$1.139728016896 \times 10^{12}$	$2.6199925603251017 \times 10^{-2}$
17	$1.191399366656 \times 10^{12}$	$1.191399366656 \times 10^{12}$	$1.1769125575376904 \times 10^{-2}$
18	$2.064536443392 \times 10^{12}$	$2.064536443392 \times 10^{12}$	$3.3212325988323244 \times 10^{-3}$
19	$4.078909789184 \times 10^{12}$	$4.078909789184 \times 10^{12}$	$5.534781953926426 \times 10^{-4}$
20	$6.50945294848 \times 10^{12}$	$6.50945294848 \times 10^{12}$	$3.82874895308305 \times 10^{-5}$

Tabela 4: Błędy bezwzględne uzyskanych pierwiastków wielomianu Wilkinsona.

k	z_k	$\lvert P(z_k)\rvert$	$\lvert z_k - k\rvert$
1	$0.9999999999998185 + 0.0im$	2.3040×10^{4}	$1.815214645262131 \times 10^{-13}$
2	$2.0000000000367115 + 0.0im$	2.26304×10^{5}	$3.6711522710675126 \times 10^{-11}$
3	$2.9999999982108 + 0.0im$	1.020416×10^{6}	$1.7891998993491143 \times 10^{-9}$
4	$4.000000046165133 + 0.0im$	5.408768×10^{6}	$4.6165133049669294 \times 10^{-8}$
5	$4.999999028612442 + 0.0im$	2.4512×10^{7}	$9.71387557946457 \times 10^{-7}$
6	$6.000019095243171 + 0.0im$	1.18212096×10^{8}	$1.909524317067479 \times 10^{-5}$
7	$6.999593471115021 + 0.0im$	4.35741184×10^{8}	$4.0652888497927364 \times 10^{-4}$
8	$8.007845735396307 + 0.0im$	$1.326422528 \times 10^{9}$	$7.845735396307063 \times 10^{-3}$
9	$8.91577894554576 + 0.0im$	$2.641426944 \times 10^{9}$	$8.422105445423966 \times 10^{-2}$
10	$10.095898499560946 - 0.6443373996274354im$	$3.933353845712834 \times 10^{9}$	$6.514347294830742 \times 10^{-1}$
11	$10.095898499560946 + 0.6443373996274354im$	$3.933353845712834 \times 10^{9}$	1.1102117850458961
12	$11.793856811906817 - 1.6517914101583042im$	$5.607254943502496 \times 10^{10}$	1.6646050212197427
13	$11.793856811906817 + 1.6517914101583042im$	$5.607254943502496 \times 10^{10}$	2.0452863498435483
14	$13.99219600562902 - 2.5185303274174613im$	$7.277997935632854 \times 10^{11}$	2.518542418235128
15	$13.99219600562902 + 2.5185303274174613im$	$7.277997935632854 \times 10^{11}$	2.712685735796096
16	$16.73061534277457 - 2.8125332877614704im$	$9.452080179531715 \times 10^{12}$	2.905880636547886
17	$16.73061534277457 + 2.8125332877614704im$	$9.452080179531715 \times 10^{12}$	2.825404676911753
18	$19.50237924554758 - 1.9403001967481368im$	$6.223348199767701 \times 10^{13}$	2.4539576709782454
19	$19.50237924554758 + 1.9403001967481368im$	$6.223348199767701 \times 10^{13}$	2.004282854254313
20	$20.84687198804629 + 0.0im$	$1.48081940965376 \times 10^{14}$	$8.468719880462885 \times 10^{-1}$

Tabela 5: Błędy bezwzględne uzyskanych pierwiastków zmienionego wielomianu Wilkinsona.

4.4. Wnioski

Rzut oka pozwala dostrzec, że uzyskane wyniki różnią się znacznie od oczekiwanych. Miast zer, otrzymano liczby rzędu bilionów. Wyraźnie widoczny jest wzrost błędu wraz ze wzrostem wartości pierwiastka. Różnice początkowo nie są zbyt duże, jednak wraz z kolejnymi etapami obliczeń kumulują się, doprowadzając do tak diametralnych rozbieżności (np. dla $x_0 = 1,\ z_0 = 1-\epsilon,\ |p(x_0)| = \epsilon \cdot 19!$). Współczynniki wielomianu przekazywane jako argumenty funkcji `Poly(W)` nie mogą być dokładnie reprezentowane (dostępnych jest od 15 do 17 miejsc na cyfry znaczące systemu dziesiętnego), stąd wynikłe zaburzenia. Przebieg drugiego eksperymentu był analogiczny, poczyniono jednak nieznaczne zmiany w jednym ze współczynników danego wielomianu. W rezultacie otrzymano zauważalne różnice w błędach bezwzględnych obliczonych pierwiastków – są zdecydowanie wyższe niż w przypadku niezmienionego wielomianu Wilkinsona. Dodatkowo istotnym spostrzeżeniem jest uzyskanie pierwiastków należących do liczb zespolonych – część urojona pojawia się od z_{10}. Niewielkie zaburzenia wpłynęły znacząco na odkształcenie wyników, zatem zadanie to jest źle uwarunkowane.

5. Zadanie 5

5.1. Opis problemu

Przeprowadzenie dwóch eksperymentów przy wykorzystaniu następującego modelu wzrostu populacji:

$$p_{n+1} := p_n + rp_n(1 - p_n), \text{ dla } n = 1, 2, \dots,$$

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. Testy przeprowadzić dla danych wejściowych we wskazanych arytmetykach:

(a) $p_0 = 0.01$, $r = 3$, $n = 40$ {`Float32`}
(b) $p_0 = 0.01$, $r = 3$, $n = 40$ {`Float64`}
(c) $p_0 = 0.01$, $r = 3$, $n = 40$, po 10 iteracjach zastosowanie obcięcia wyniku do trzeciego miejsca po przecinku {`Float32`}.

5.2. Opis rozwiązania

Rozwiązanie problemu zrealizowano w języku `Julia`. Utworzono dwie funkcje, które wykonują 40 iteracji, wyliczając wartość p_n, a następnie zapisują ją w tablicy (przy czym druga z funkcji po 10 przebiegach pętli dokonuje obcięcia wyniku). Algorytm pierwszej z nich przedstawiony został na poniższym listingu:

Algorytm 1

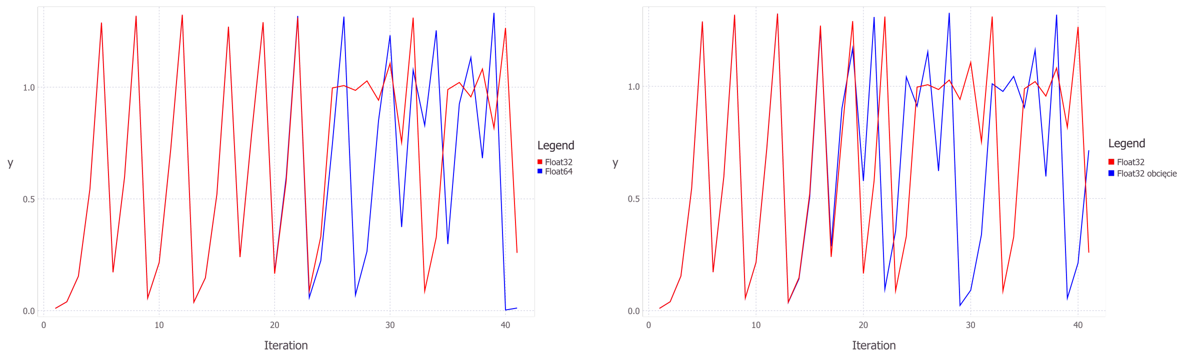
```
A ← [0, 0, ..., 0]
p ← 0.01
A[1] ← p
i ← 1
while i < n do
    p ← p + r * p * (1.0 − p)
    A[i + 1] ← p
    i ← i + 1
end while
return A
```

5.3. Wyniki

Uzyskane rezultaty dla przeprowadzonych eksperymentów zaprezentowano na Rysunku 4.

5.4. Wnioski

W przypadku pierwszego eksperymentu doskonale można zaobserwować, jak, początkowo zdawałoby się znikomy, błąd staje się wraz z kolejnymi przebiegami tak duży, iż uzyskujemy



(a) (a.) oraz (c.).

(b) (a.) oraz (b.).

Rysunek 4: Zestawienie wyników dla konkretnych danych

niemal nową, oddzielną iterację (wynik zgodny jest co do części tysięcznej, czyli w każdej kolejnej iteracji pojawia się różnica pomiędzy rozwiązaniami). W przypadku arytmetyki `Float32` dostępnych jest tylko 6-9 cyfr znaczących, których, w miarę postępowania iteracji, potrzeba coraz więcej by móc zapisać prawidłowy wynik.

Analiza drugiego wykresu pozwala dostrzec różnice w regularności przebiegu. Niższa precyzja arytmetyki `Float32` wpłynęła na stale powiększające się różnice, by osiągnąć swoistą kulminacyjną wartość równą 1.0. Każda kolejna iteracja oscylowała w pobliżu właśnie tego wyniku. Ciekawym spostrzeżeniem jest fakt, iż dla `Float64` ma miejsce podobna sytuacja, jednak w znacznie dalszej iteracji.

Powyższe eksperymenty pozwalają dostrzec, że błąd obliczeniowy wyniku, przeniesiony jako wejście kolejnej operacji, potęguje powstały błąd. Jest to typowy przykład sprzężenia zwrotnego - procesu, w którym dane wyjściowe jednego problemu są wejściem kolejnych obliczeń. Zaskakującym jest natomiast fakt nieskorelowania, co wskazuje na istnienie chaosu w systemie czy też może zjawisko niemożności przewidzenia (zgodnie ze sformułowaniem Lorenza). W uniknięciu tego rodzaju sytuacji pomóc może zwiększenie precyzji obliczeń, choć jednak w daleko wybiegających symulacjach powstałe błędy mogą zniekształcić obliczenia w tak dużym stopniu, iż również te wyniki staną się dla badającego bezużyteczne. Biorąc pod uwagę powyższe rozważania, niemożliwym jest uniknięcie błędów zaokrągleń, które są w pewien sposób integralną częścią obliczeń w arytmetyce zmiennopozycyjnej. Taki efekt będzie miał miejsce na każdym urządzeniu, niezależnie od jego mocy obliczeniowej. Stąd prosty wniosek – zadanie jest źle uwarunkowane.

6. Zadanie 6

6.1. Opis problemu

Przeprowadzenie eksperymentów w języku `Julia` w arytmetyce `Float64` dla następującego równania rekurencyjnego:

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

gdzie c jest pewną stałą, dla danych wejściowych:

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 $x_0 = 0.25$;

oraz ilości iteracji $n = 40$.

6.2. Opis rozwiązania

W języku `Julia` skonstruowano funkcję `calculate(`x_0`,`c`)`, która wykonuje n iteracji, obliczając aktualną wartość zgodnie z podanym w zadaniu wzorem. Uzyskane w ten sposób rozwiązania przechowywane są w tablicy, a następnie generowane są odpowiednie wykresy.

6.3. Wyniki

Tabela 6 przedstawia wyniki eksperymentów uzyskane dla kolejnych iteracji $1 \leq k \leq n$.

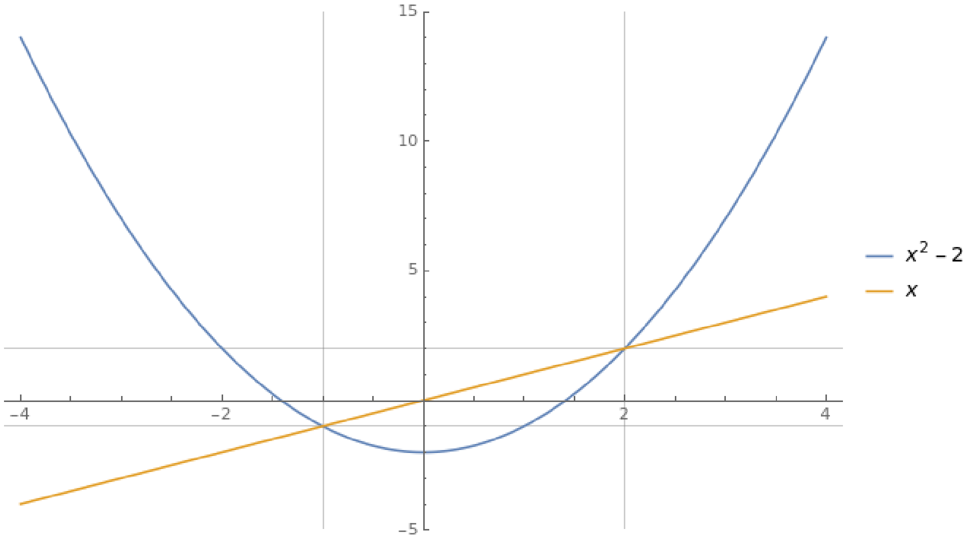
k	Eksperyment						
	1.	2.	3.	4.	5.	6.	7.
1	−1.0	2.0	1.999 999 999 999 96	0.0	0.0	−0.4375	−0.9375
2	−1.0	2.0	1.999 999 999 999 840 1	−1.0	−1.0	−0.808 593 75	−0.121 093 75
3	−1.0	2.0	1.999 999 999 999 360 5	0.0	0.0	−0.346 176 147 460 937 5	−0.985 336 303 710 937 5
4	−1.0	2.0	1.999 999 999 997 442	−1.0	−1.0	−0.880 162 074 929 103 3	−0.029 112 368 589 267 135
5	−1.0	2.0	1.999 999 999 989 768 2	0.0	0.0	−0.225 314 721 856 495 6	−0.999 152 469 995 122 6
6	−1.0	2.0	1.999 999 999 959 072 7	−1.0	−1.0	−0.949 233 276 114 730 1	−0.001 694 341 702 645 596 5
7	−1.0	2.0	1.999 999 999 836 291	0.0	0.0	−0.098 956 187 516 496 6	−0.999 997 129 206 194 7
8	−1.0	2.0	1.999 999 999 345 163 8	−1.0	−1.0	−0.990 207 672 952 199 9	$-5.741\,579\,369\,278\,327 \times 10^{-6}$
9	−1.0	2.0	1.999 999 997 380 655 3	0.0	0.0	−0.019 488 764 426 589 09	−0.999 999 999 967 034 3
10	−1.0	2.0	1.999 999 989 522 621	−1.0	−1.0	−0.999 620 188 061 125	$-6.593\,148\,249\,578\,462 \times 10^{-11}$
11	−1.0	2.0	1.999 999 958 090 484 1	0.0	0.0	−0.000 759 479 620 641 156 9	−1.0
12	−1.0	2.0	1.999 999 832 361 938 3	−1.0	−1.0	−0.999 999 423 190 705 8	0.0
13	−1.0	2.0	1.999 999 329 447 781 4	0.0	0.0	$-1.153\,618\,255\,700\,372\,7 \times 10^{-6}$	−1.0
14	−1.0	2.0	1.999 997 317 791 574 9	−1.0	−1.0	−0.999 999 999 998 669 2	0.0
15	−1.0	2.0	1.999 989 271 173 493 7	0.0	0.0	$-2.661\,648\,679\,236\,350\,3 \times 10^{-12}$	−1.0
16	−1.0	2.0	1.999 957 084 809 082 6	−1.0	−1.0	−1.0	0.0
17	−1.0	2.0	1.999 828 341 078 044	0.0	0.0	0.0	−1.0
18	−1.0	2.0	1.999 313 393 778 961 3	−1.0	−1.0	−1.0	0.0
19	−1.0	2.0	1.997 254 046 543 948 1	0.0	0.0	0.0	−1.0
20	−1.0	2.0	1.989 023 726 436 175 2	−1.0	−1.0	−1.0	0.0
21	−1.0	2.0	1.956 215 384 326 048 6	0.0	0.0	0.0	−1.0
22	−1.0	2.0	1.826 778 629 873 91	−1.0	−1.0	−1.0	0.0
23	−1.0	2.0	1.337 120 162 563 999 7	0.0	0.0	0.0	−1.0
24	−1.0	2.0	−0.212 109 670 864 823 13	−1.0	−1.0	−1.0	0.0
25	−1.0	2.0	−1.955 009 487 525 616 3	0.0	0.0	0.0	−1.0
26	−1.0	2.0	1.822 062 096 315 173	−1.0	−1.0	−1.0	0.0
27	−1.0	2.0	1.319 910 282 828 443	0.0	0.0	0.0	−1.0
28	−1.0	2.0	−0.257 836 845 283 739 6	−1.0	−1.0	−1.0	0.0
29	−1.0	2.0	−1.933 520 161 214 128 8	0.0	0.0	0.0	−1.0
30	−1.0	2.0	1.738 500 213 821 510 9	−1.0	−1.0	−1.0	0.0
31	−1.0	2.0	1.022 382 993 457 438 9	0.0	0.0	0.0	−1.0
32	−1.0	2.0	−0.954 733 014 689 006 5	−1.0	−1.0	−1.0	0.0
33	−1.0	2.0	−1.088 484 870 662 841 2	0.0	0.0	0.0	−1.0
34	−1.0	2.0	−0.815 200 686 338 097 8	−1.0	−1.0	−1.0	0.0
35	−1.0	2.0	−1.335 447 840 993 894 4	0.0	0.0	0.0	−1.0
36	−1.0	2.0	−0.216 579 063 984 746 25	−1.0	−1.0	−1.0	0.0
37	−1.0	2.0	−1.953 093 509 043 491	0.0	0.0	0.0	−1.0
38	−1.0	2.0	1.814 574 255 067 817 4	−1.0	−1.0	−1.0	0.0
39	−1.0	2.0	1.292 679 727 154 924 4	0.0	0.0	0.0	−1.0
40	−1.0	2.0	−0.328 979 123 002 670 2	−1.0	−1.0	−1.0	0.0

Tabela 6: Iteracje w arytmetykach `Float32` oraz `Float64`

6.4. Wnioski

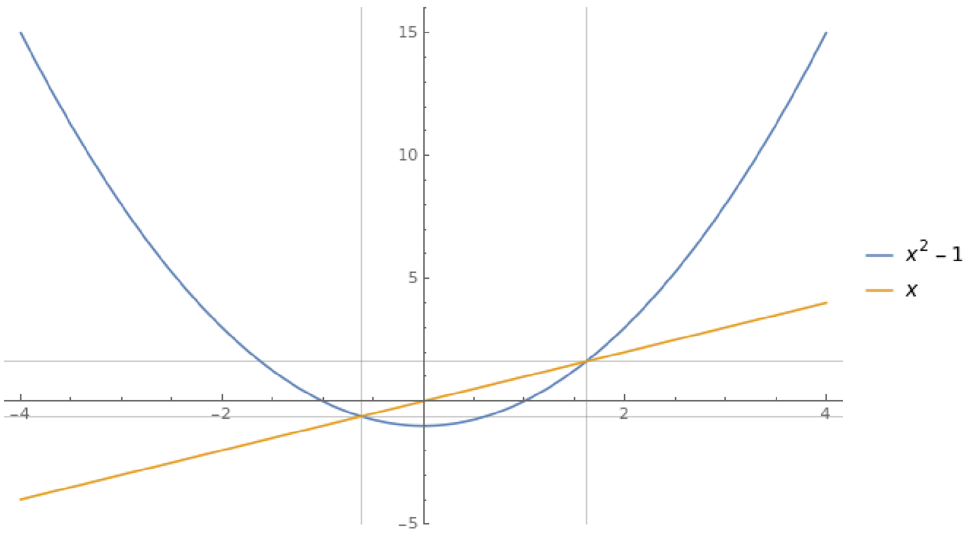
Wizualizacje obu funkcji (Rysunek 5, 6) pozwalają wyciągnąć kolejne wnioski.

1. $x_0 = 1 \;\Rightarrow\; \phi(x)$ zbieżna do -1.



Rysunek 5: $\phi = x^2 - 2,\ f(x) = x$

2. $x_0 = 2 \ \Rightarrow\ \phi(x)$ zbieżna do 2.
3. $x_0 = 1.99999999999999 \ \Rightarrow\ \phi(x)$ rozbieżna.



Rysunek 6: $\phi = x^2 - 1,\ f(x) = x$

1. $x_0 = 1 \ \Rightarrow$ podciąg wyrazów nieparzystych $\phi(x)$ zbieżny do 0, zaś parzystych: -1.
2. $x_0 = -1 \ \Rightarrow$ podciąg wyrazów nieparzystych $\phi(x)$ zbieżny do 0, zaś parzystych: -1.
3. $x_0 = 0.75 \ \Rightarrow$ podciąg wyrazów nieparzystych $\phi(x)$ (od pewnego x) zbieżny do 0, zaś parzystych: -1.
4. $x_0 = 0.25 \ \Rightarrow$ podciąg wyrazów nieparzystych $\phi(x)$ (od pewnego x) zbieżny do 0, zaś parzystych: -1.