Obliczenia naukowe lista nr 2

Sebastian Woźniak 268491

October 2023

1 Iloczyn skalarny v2

W tym programie obliczany jest iloczyn skalarny dwóch wektorów z liczbami zmiennoprzecinkowymi na różne sposoby tak jak w zadaniu 5 na liście nr 1 jednak po wprowadzeniu małych zmian w wektorze x.

$$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$$
$$x' = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$$

Z x_4 oraz x_5 zostały usunięte najmniej znaczące liczby po przecinku.

1.1 $x * y$ - poprzednie wyniki

```
[Prawdziwy iloczyn skalarny]: -1.00657107000000e-11

[Pojedyńcza precyzja - Float32]:
Iloczyn 'w przód':      -0.4999443
Iloczyn 'w tył':        -0.4543457
Iloczyn malejąco:       -0.5
Iloczyn rosnąco:        -0.5

[Podwójna precyzja - Float64]:
Iloczyn 'w przód':      1.0251881368296672e-10
Iloczyn 'w tył':        -1.5643308870494366e-10
Iloczyn malejąco:       0.0
Iloczyn rosnąco:        0.0
```

1.2 $x' * y$ - nowe wyniki

```
[Prawdziwy iloczyn skalarny]: -0.004296343192495245

[Pojedyńcza precyzja - Float32]:
Iloczyn 'w przód':      -0.4999443
Iloczyn 'w tył':        -0.4543457
```

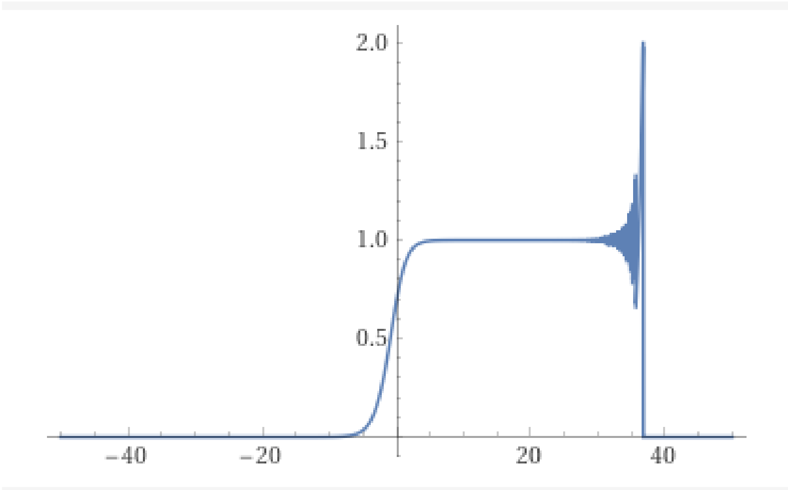
```
Iloczyn malejąco:       -0.5
Iloczyn rosnąco:        -0.5

[Podwójna precyzja - Float64]:
Iloczyn 'w przód':      -0.004296342739891585
Iloczyn 'w tył':        -0.004296342998713953
Iloczyn malejąco:       -0.004296342842280865
Iloczyn rosnąco:        -0.004296342842280865
```

1.3 Obserwacje i wnioski

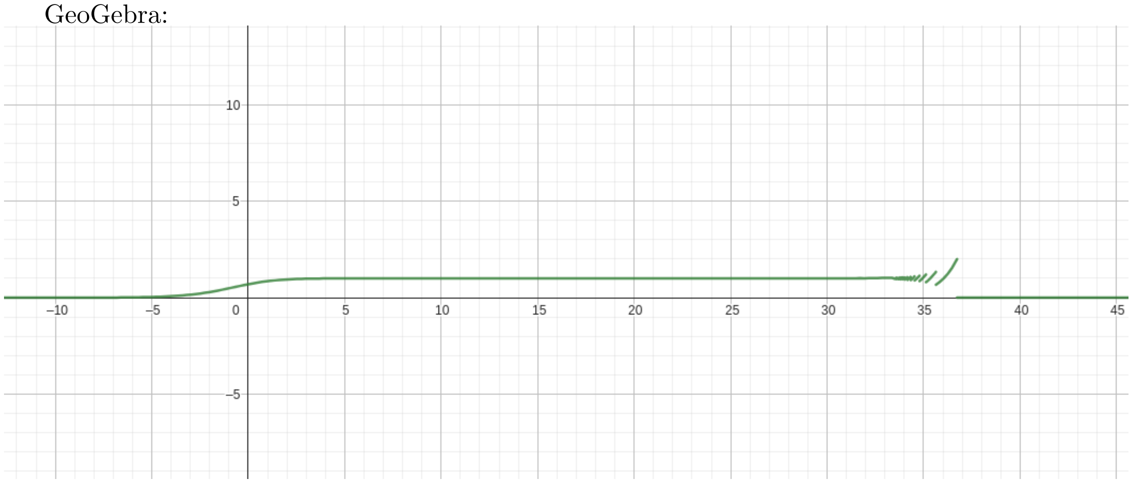
Dla Float32 nie otrzymaliśmy żadnej różnicy w wynikach co jest spowodowane tym, że usunięcie tak mało znaczących liczb po przecinku zostało niezuważone w 32 bitowej dokładności i zapis liczb x_4 i x_5 w standardzie IEEE-754 nadal jest taki sam. Dla Float64 wyniki znacznie się zmnieniły i widzimy, że otrzymany błąd jest dużo mniejszy. Najlepiej wypadała technika liczenia iloczynu "w tył", gdzie błąd względny wyniósł $1.93781292 * 10^{-10}$.
Można powiedzieć, że zadanie jest źle uwarunkowane, ponieważ względnie nieduże zmiany danych spowodowały duże zmiany wyników.

2 Wykres funkcji $f(x) = e^x ln(1 + e^{-x})$

Wolfram Alpha:



GeoGebra:



2.1 granica funkcji $f(x)$

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

Tutaj można zastosować regułę de l'Hospitala, ponieważ funkcja w liczniku oraz funkcja w mianowniku dążą do 0.

$$\lim_{x\to\infty} \frac{-\frac{1}{1+e^x}}{-e^{-x}} = \lim_{x\to\infty} \frac{e^x}{1+e^x} = 1$$

2.2 Obserwacje i wnioski

Z obliczonej granicy wynika, że funkcja powinna dążyć do 1 jednak na wykresach widać, że między wartościami argumentu $x[32, 37]$ wartość funkcji zaczyna odbiegać od tej wartości, funkcja zdaje się tracić ciągłość, aż w konću wartość zatrzymuje się na 0. Dzieje się tak, ponieważ coraz większe wartości e^x są mnożone ze zbiegającymi do 0 wartościami $\ln(1+e^{-x})$ przez co w pewnym momencie tracimy dokładność spowodowaną ograniczoną liczbą bitów na zapis liczby i całe wyrażenie jest mnożone przez 0.

3 Rozwiązywanie układu równań liniowych

W tym zadaniu należy rozwiązać rownanie $Ax = b$ gdzie A należy do $\mathbb{R}^n$. Przyjmujemy, że $x = (1, ..., 1)^T$ i na tej podstawie wyznaczamy b. Następnie równanie jest rozwiązywane na dwa różne sposoby.

Metoda Gaussa: $x = A/b$
Metoda inwersji: $x = A^{-1} * b$

Oto otrzymane wyniki dla macierzy Hilberta:

[Macierz Hilberta]

n	Błąd metodą Gaussa	Błąd metodą inwersji	Rząd macierzy	Wskaznik uwarunkowania
2	5.661048867003676e-16	1.1240151438116956e-15	2	19.28147006790397
4	4.4515459601812086e-13	2.950477637286781e-13	4	15513.73873892924
6	2.618913302311624e-10	3.3474135070361745e-10	6	1.4951058642254734e7
8	1.026543065687064e-7	2.698715074276819e-7	8	1.5257575538060041e10
10	0.0006329153722983848	0.00045521422517408853	10	1.602441698742836e13
12	0.2975640310734787	0.34392937091205217	11	1.7515952300879806e16
14	5.281004646755168	4.800641929017436	11	6.200786281355982e17
16	20.564655823804095	31.736467496266126	12	7.046389953630175e17
18	41.479969575058966	33.878120608181	12	2.2477642911280653e18
20	6.505361867134064	71.95104681106297	13	1.1484020388436145e18

Dla macierzy Hilberta rośnie wkażnik uwarunkowania macierzy i z tym rośnie błąd przybliżenia. W wiekszości przypadków mniejszy błąd otrzymaliśmy metodą Gaussa. Układ równań $Ax = b$ jest źle uwarunkowany dla macierzy Hilberta.

Otrzymane wyniki dla losowej macierzy o wskaźniku uwarunkowania c:

[Macierz Losowa]

n	c	Błąd metodą Gaussa	Błąd metodą inwersji	Rząd macierzy
5	1.0	1.1102230246251565e-16	2.0471501066083611e-16	5
5	10.0	3.020133145511626e-16	4.684036072284178e-16	5
5	1000.0	2.967017806518985e-14	6.635911773455463e-14	5
5	1.0e7	3.060673659425244e-16	1.2660649847187252e-10	5
5	1.0e12	2.5889664591538383e-5	1.7629517987018604e-5	5
5	1.0e16	0.21763384024982552	0.15622316584947313	4
10	1.0	3.274687455368547e-16	4.2855183991760757e-16	10
10	10.0	6.924445963069148e-16	5.517707908259862e-16	10
10	1000.0	4.1483138272472355e-14	2.3528897652403893e-14	10
10	1.0e7	6.581472217449465e-10	2.111318872536243e-10	10
10	1.0e12	5.5509172846143316e-5	2.182703768967815e-5	10
10	1.0e16	0.13626909981072616	0.09231534972839286	9
20	1.0	5.376277206893598e-16	3.339908118180928e-16	20
20	10.0	7.947987303456223e-16	4.890025172686938e-16	20
20	1000.0	2.701170565478764e-14	1.8776100195282036e-14	20
20	1.0e7	5.6220558053228445e-11	2.9598484682114417e-10	20
20	1.0e12	5.346874852734693e-6	1.7803743765238382e-5	20
20	1.0e16	0.42704313687686574	0.480605005312194	19

Dla losowej macierzy błąd przybliżenia rośnie wraz z rozmiarem macierzy oraz wskaźnikiem uwarunkowania. Tempo wzorstu jest znaczenie mniejsze od zobserwowanego dla macierzy Hilberta. Rożnica między błedem w obliczeniach metodą Guassa oraz metodą inwersji nie jest też tak mocno zauważalna. Dla niektórych przypadków lepsza jest ta pierwsza, a innych druga.

4 Złośliwy wielomian Wilkinsona

Używając funkcji roots obliczyliśmy miejsca zerowe wielomianu p:

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

Sprawdzimy je podstawiając do funkcji w postaci naturalnej oraz postaci iloczynowej:

```
k | z_k | |p(z_k)| | |P(z_k)| | |z_k - k|
1 | 0.9999999999996989 | 36626.4254824228 | 35696.50964788257 | 3.0109248427834245e-13
2 | 2.0000000000283182 | 181303.9336725767 | 176252.60026668405 | 2.8318236644508943e-11
3 | 2.9999999995920965 | 290172.2858891687 | 279157.6968824087 | 4.0790348876384996e-10
4 | 3.9999999837375317 | 2.04153729027509e6 | 3.0271092988991085e6 | 1.626246826091915e-8
5 | 5.000000665769791 | 2.0894625006962176e7 | 2.2917473756567076e7 | 6.657697912970661e-7
6 | 5.999989245824773 | 1.1250484577562997e8 | 1.2902417284205095e8 | 1.0754175226779239e-5
7 | 7.000102002793008 | 4.5729086427309465e8 | 4.805112754602064e8 | 0.00010200279300764947
8 | 7.999355829607762 | 1.5556459377357383e9 | 1.6379520218961136e9 | 0.0006441703922384079
9 | 9.002915294362053 | 4.68781617564839e9 | 4.877071372550003e9 | 0.002915294362052734
10 | 9.990413042481725 | 1.2634601896949207e10 | 1.3638638195458128e10 | 0.009586957518274986
11 | 11.025022932909318 | 3.3001284744984142e10 | 3.585631295130865e10 | 0.025022932909317674
12 | 11.953283253846857 | 7.388525665404987e10 | 7.533332360358197e10 | 0.04671674615314281
13 | 13.07431403244734 | 1.84762150931442e11 | 1.9605988124330817e11 | 0.07431403244734014
14 | 13.914755591802127 | 3.551427752842085e11 | 3.5751347823104315e11 | 0.08524440819787316
15 | 15.075493799699476 | 8.423201558964255e11 | 8.21627123645597e11 | 0.07549379969947623
16 | 15.946286716607972 | 1.5707287366258018e12 | 1.5514978880494067e12 | 0.05371328339202819
17 | 17.025427146237412 | 3.3169782238892354e12 | 3.6947359184862 29e12 | 0.025427146237412046
18 | 17.99092135271648 | 6.344853141791281e12 | 7.650109016515867e12 | 0.009078647283519814
19 | 19.00190981829944 | 1.2285717366719662e13 | 1.1435273749721195e13 | 0.0019098182994383706
20 | 19.999809291236637 | 2.318309535271639e13 | 2.7924106393680727e13 | 0.00019070876336257925
```

Obliczone pierwiastki nie są dokładne, a błędy przybliżenia znajdują w się przedziale od $3.0109248427834245e-13$ do 0.07549379969947623. Wartości są zbliżone do właściwych jednak przez ograniczenia precyzji Float64 oraz duże współczynniki otrzymujemy wyniki różne od 0 po podstawieniu zarówno do funkcji w postaci naturalnej jak i postaci iloczynowej. Dla obu postaci wyniki są podobne.

4.1 Zmiana jednego ze współczynników

Współczynnik -210 zostaje zmniejszony o małą liczbę 2^{-23}. Tak prezentują się otrzymane wyniki:

```
k | z_k | |p(z_k)| | |P(z_k)| | |z_k - k|
1 | 0.9999999999998357 + 0.0im | 19987.872313406842 | 20259.872313418207 | 1.6431300764452317e-13
2 | 2.0000000000550373 + 0.0im | 352369.413808796 | 346541.4137593836 | 5.503730804434781e-11
3 | 2.99999999660342 + 0.0im | 2.416241558251844e6 | 2.2580597001197007e6 | 3.3965799062229962e-9
4 | 4.000000089724362 + 0.0im | 1.126370230029202e7 | 1.0542631790395478e7 | 8.972436216225788e-8
5 | 4.99999857388791 + 0.0im | 4.475744423806907e7 | 3.757830916585153e7 | 1.4261120897529622e-6
6 | 6.000020476673031 + 0.0im | 2.1421031658039317e8 | 1.3140943325569446e8 | 2.0476673030955794e-5
7 | 6.99960207042242 + 0.0im | 1.7846173427860644e9 | 3.939355874647618e8 | 0.00039792957757978087
8 | 8.007772029099446 + 0.0im | 1.868697217000985e10 | 1.184986961371896e9 | 0.007772029099445632
9 | 8.915816367932559 + 0.0im | 1.3746309775142996e11 | 2.2255221233077707e9 | 0.0841836320674414
10 | 10.095455630535774 - 0.6449328236240688im | 1.4900695352000583e12 | 1.0677921232930157e10 | 0.6519586830380407
11 | 10.095455630535774 + 0.6449328236240688im | 1.4900695352000583e12 | 1.0677921232930157e10 | 1.1109180272716561
12 | 11.793890586174369 - 1.6524771364075785im | 3.2962792355717137e13 | 3.1401962344429485e10 | 1.665281290598479
13 | 11.793890586174369 + 1.6524771364075785im | 3.2962792355717137e13 | 3.1401962344429485e10 | 2.0458202766784277
14 | 13.992406684487216 - 2.5188244257108443im | 9.546022365750218e14 | 2.157665405951858e11 | 2.518835871190904
15 | 13.992406684487216 + 2.5188244257108443im | 9.546022365750218e14 | 2.157665405951858e11 | 2.7128805312847097
16 | 16.73074487979267 - 2.812624896721978im | 2.742106076928478e16 | 4.850110893921027e11 | 2.9060018735375106
17 | 16.73074487979267 + 2.812624896721978im | 2.742106076928478e16 | 4.850110893921027e11 | 2.825483521349608
18 | 19.5024423688181 - 1.940331978642903im | 4.25248587652037e17 | 4.557199223869993e12 | 2.4540214463129764
19 | 19.5024423688181 + 1.940331978642903im | 4.25248587652037e17 | 4.557199223869993e12 | 2.0043294443099486
20 | 20.84691021519479 + 0.0im | 1.374374355999760 1e18 | 8.756386551865696e12 | 0.8469102151947894
```

Ta mała zmiana sprawiła, że obliczone pierwiastki są liczbami zespolonymi i zaczęły znacznie odbiegać od rzeczywistych wartości. Na podstawie tego eksperymentu możemy stwierdzić, że zadanie jest źle uwarunkowane.

5 Równanie rekurencyjne

W zadaniu obliczne jest równanie:

$$p_{n+1} := p_n + r * p_n(1 - p_n)$$

Dla danych:

$$n = 40, r = 3, p_0 = 0.01$$

Przeprowadzono obliczenia dla Float32 oraz Float64, a także z zaokrągleniem wyniku do 3 cyfr po przecinku po 10 iteracji:

```
Bez obcięcia Float32: 0.25860548
Z obcięciem Float32: 1.093568
Bez obcięcia Float64: 0.011611238029748606
Z obcięciem Float64: 0.7305550338104317
```

Wnioskiem jest, że zadaniu jest źle uwarunkowane, ponieważ zmiany dokładoności czy ucięcie liczby w jednym z kroków proawdzi do uzyskania kompletnie różnych wyników. Błędy zaokrągleń bardzo szybko się kumulują:

```
n|      Float32 |       Float64
2       0.15407173      0.15407173000000002
7       1.3191134       1.3191137924137974
12      0.037716985     0.03769529725473175
17      0.7860428       0.7881011902353041
22      0.088804245     0.058377608259430724
27      1.0280086       0.26542635452061003
32      0.0877831       0.8291255674004515
37      1.0813814       0.6822410727153098
```

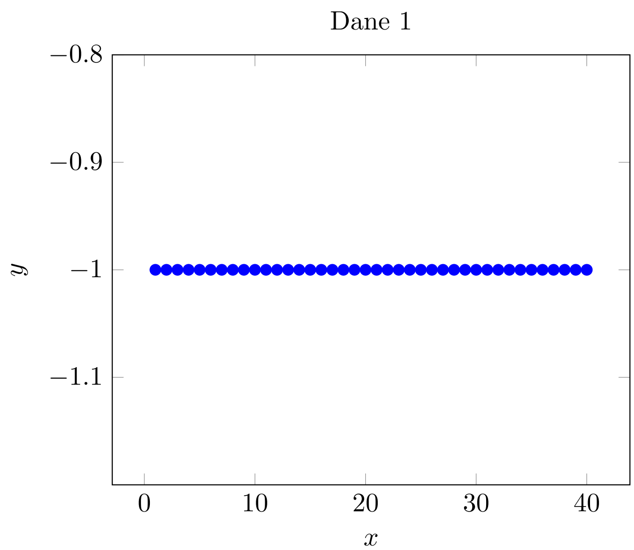
6 Równanie rekurencyjne 2

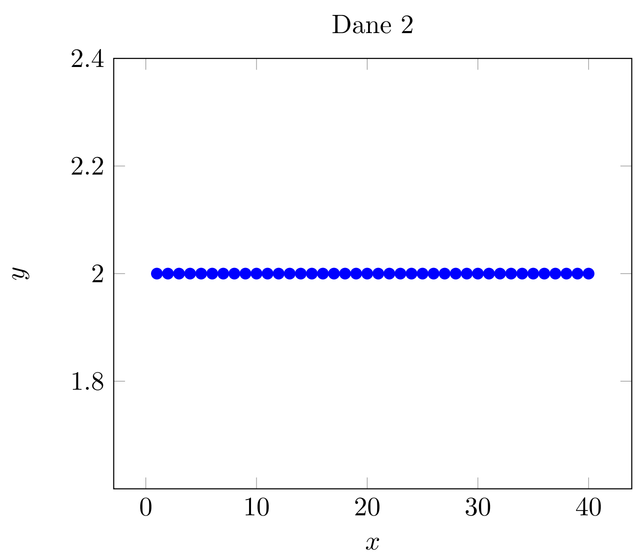
W zadaniu obliczane jest równanie:

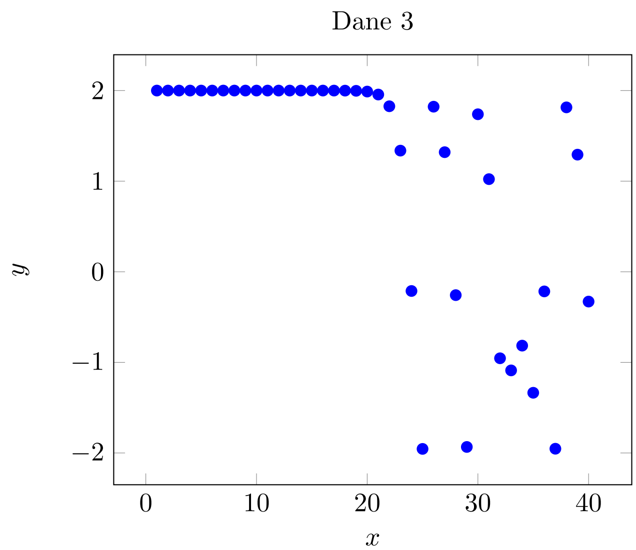
$$x_{n+1} := x_n^2 + c$$

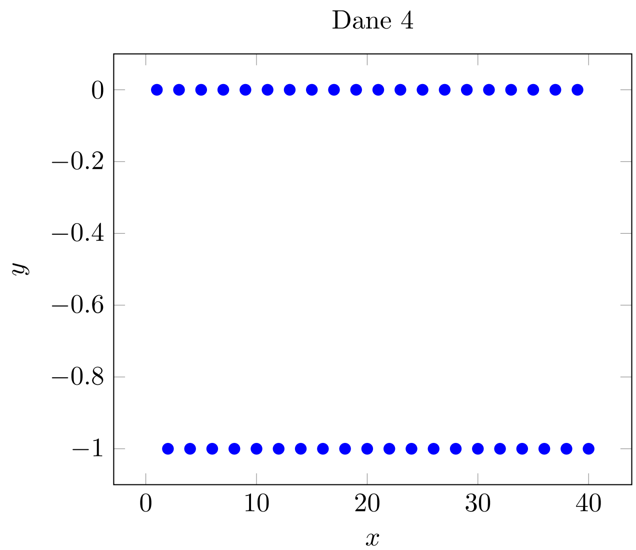
Dla 7 różnych danych:
1. $c = -2, x0 = 1$
2. $c = -2, x0 = 2$
3. $c = -2, x0 = 1.99999999999999$
4. $c = -1, x0 = 1$
5. $c = -1, x0 = -1$
6. $c = -1, x0 = 0.75$
7. $c = -1, x0 = 0.25$

6.1 Graficzna iteracja

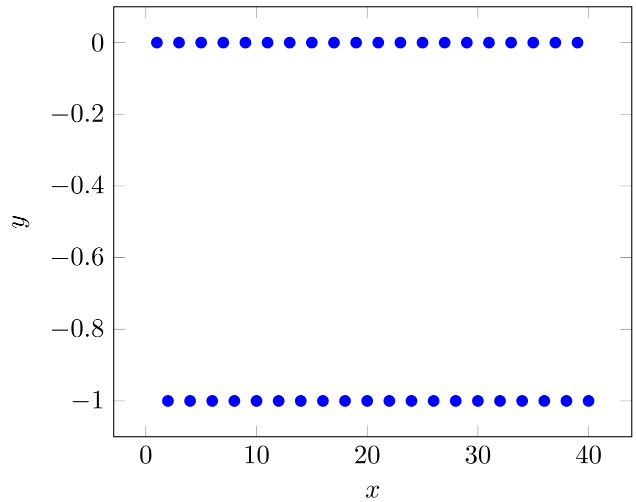




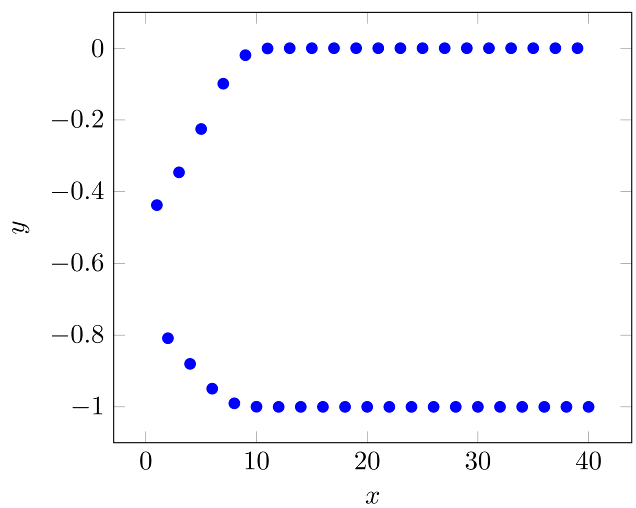

Dane 3
y
2
1
0
−1
−2
0
10
20
30
40
x

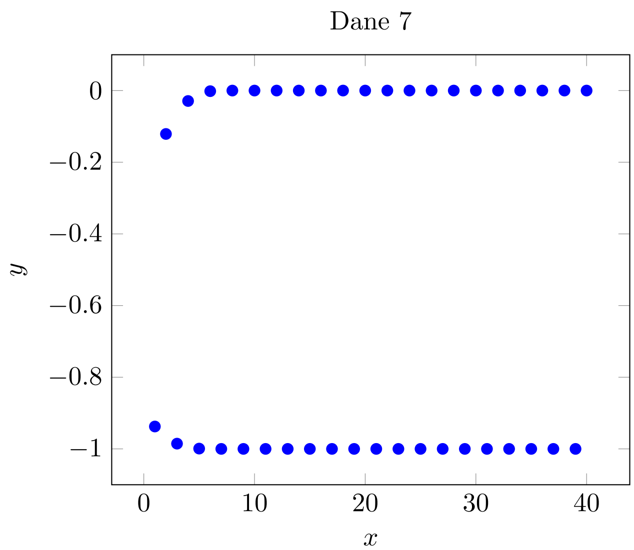

Dane 4
y
0
−0.2
−0.4
−0.6
−0.8
−1
0
10
20
30
40
x

Dane 5



Dane 6





6.2 Wnioski

Z zaobserwowanych wyników możemy wywnioskować, że funkcja zachowuje się tak jak można było przewidzieć dla wartości całkowitych. W przypadku ułamków z wieloma cyframi po przecinku wyniki zaczęły odbiegać od rzeczywistych z powodu utraty precyzji. W przypadku ułamków 0.25 oraz 0.75 wyniki zbiegły do wartości całkowitych również z powodu tracenia bitów.