Sprawozdanie z listy 2. - Obliczenia naukowe

Jakub Bachanek

8 listopada 2020

1 Zadanie 1.

1.1 Opis problemu

Celem zadania jest sprawdzenie jaki wpływ na wynik ma niewielka zmiana danych przy obliczeniu iloczynu skalarnego dwóch wektorów w arytmetyce `Float32` oraz `Float64` za pomocą czterech różnych algorytmów sumowania:

- "w przód"
- "w tył"
- od największego do najmniejszego
- od najmniejszego do największego

1.2 Rozwiązanie

Uruchamiamy ponownie zadanie 5 z listy 1, ale teraz z lekko zaburzonymi danymi przy x_4 i x_5.

1.3 Wyniki i interpretacja

Poniższe tabele przedstawiają uzyskane wyniki.

Dla `Float32`:

Punkt	Result OLD	Result NEW
a	-0.4999443	-0.4999443
b	-0.4543457	-0.4543457
c	-0.5	-0.5
d	-0.5	-0.5

Dla `Float64`:

Punkt	Result OLD	Result NEW
a	1.0251881368296672e-10	-0.004296342739891585
b	-1.5643308870494366e-10	-0.004296342998713953
c	0.0	-0.004296342842280865
d	0.0	-0.004296342842280865

Prawidłowa wartość = -1.00657107000000e-11.

W arytmetyce `Float32` wyniki nie różnią się, natomiast w `Float64` zauważamy znaczne różnice, które wynikają z innej reprezentacji bitowej zaburzonych liczb i uwarunkowania zadania.

1.4 Wnioski

Z powodu dużych zmian wyników w `Float64` możemy wnioskować, że zadanie było źle uwarunkowane.

2 Zadanie 2.

2.1 Opis problemu

Celem zadania jest sprawdzenie, czy programy do wizualizacji poprawnie rysują wykres funckji $f(x) = e^x \ln(1 + e^{-x})$.

2.2 Rozwiązanie

Najpierw obliczamy granicę funkcji:
$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^x \ln(1 + e^{-x}) = \lim_{x\to\infty} \frac{\ln(1+e^{-x})}{e^{-x}} = \lim_{x\to\infty} \frac{1}{1+e^{-x}} = 1$

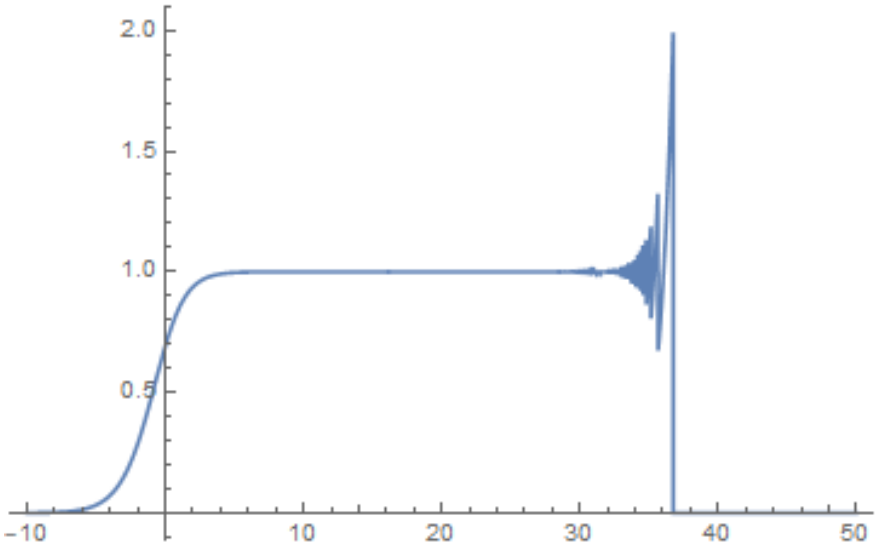
Wykresy narysowano przy użyciu programów Mathematica oraz Matplotlib Pyplot.

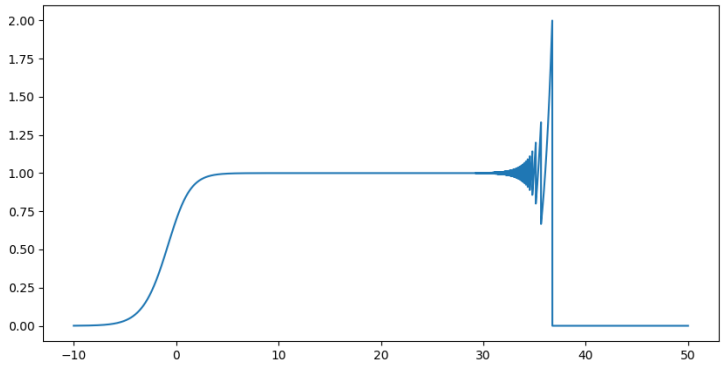
2.3 Wyniki i interpretacja

Widzimy, że oba programy zachowały się podobnie. Dla argumentów mniejszych niż ~ 30 wartości funkcji zbliżają się do obliczonej granicy, jednak później następuje narastające odchylenie od niej. Dla argumentów większych od ~ 37 zauważamy, że wartość funkcji przyjmuje 0 i dalej się już nie zmienia.
Problem ten jest spowodowany tym, że wartość logarytmu naturalnego przyjmuje coraz mniejsze wartości, nieskończenie bliskie zeru, z kolei wartość e^x nieskończenie rośnie, a wiemy, że z powodu skończonej liczby liczb maszynowych, w pewnym momencie wartość logarytmu będzie reprezentowana przez 0, zatem wynikiem mnożenia również będzie 0.

2.4 Wnioski

Z uwagi na skończoną dokładność arytmetyk zmiennoprzecinkowych, czasami nie jest możliwe stworzenie wiarygodnych wykresów funkcji bazując jedynie na obliczeniach w tych arytmetykach.





3 Zadanie 3.

3.1 Opis problemu

Celem zadania jest porównanie rozwiązań układu równań liniowych $Ax = b$ dla danej macierzy współczynników A i wektora prawych stron b. Testować będziemy dla macierzy Hilberta H_n z rosnącym stopniem n oraz losowej macierzy R_n stopnia n z zadanym wskaźnikiem uwarunkowania. Wykorzystamy dwa algorytmy: eliminacji Gaussa ($x = A\backslash b$) oraz macierzy odwrotnej ($x = A^{-1}b$).

3.2 Rozwiązanie

Obliczamy rozwiązania dla wygenerowanych macierzy w `Julii` za pomocą dwóch algorytmów dla kolejnych wartości n.

```
function solve_hilbert(n::Int)
    A = hilb(n)
    x = ones(n)
    b = A * x
    x_gauss = A \ b
    x_inv = inv(A) * b
    println("Order = ", n, "  rank = ", rank(A),"  cond = ", cond(A), "  gauss error = ", no
end

function solve_random(n::Int, c::Float64)
    A = matcond(n, c)
    x = ones(n)
    b = A * x
    x_gauss = A \ b
    x_inv = inv(A) * b
    println("Order = ", n, "  rank = ", rank(A), "  cond = ", cond(A), "  gauss error = ", r
end
```

3.3 Wyniki i interpretacja

Poniższe tabele przedstawiają uzyskane wyniki kolejno dla macierzy H_n oraz R_n.

n	rank	cond	Gauss error	inv error
1	1	1.0	0.0	0.0
2	2	19.28147006790397	5.661048867003676e-16	1.4043333874306803e-15
3	3	524.0567775860644	8.022593772267726e-15	0.0
4	4	15513.738738928929	4.637277712035294e-13	7.542470546988852e-13
5	5	476607.25024224253	1.7697056701418277e-13	7.45602798259539e-12
6	6	1.495105864125091e7	3.496491467713994e-10	3.533151828962887e-10
7	7	4.7536735637688667e8	1.3175049864850338e-8	6.190844397992631e-9
8	8	1.5257575516147259e10	2.487433466002445e-7	3.775275483015941e-7
9	9	4.9315408927806335e11	9.643625435772316e-6	1.1659486044133412e-5
10	10	1.6024859712306152e13	0.00022035288727930986	0.0003357158826776558
11	10	5.2210348947688544e14	0.006022512934347414	0.01113776822564549
12	11	1.7255427417341868e16	0.19509235225028912	0.16218620232347905
13	11	7.126491965424366e17	7.894191771622431	5.511855154155295
14	11	6.101307732044041e17	0.8270688593203056	3.3522039875276723
15	12	4.223311222761075e17	3.10349386243609	4.354299435453685
16	12	3.535827507735838e17	9.083139658689422	54.189834405860445
17	12	3.1182808742153696e17	4.24328971542452	5.786281231941037
18	12	1.5639169583348145e18	4.7860299021083	5.7599951815224495
19	13	1.3274441976880407e18	6.114994252530053	12.309212980457932
20	13	2.2777635596453635e18	19.122235961045973	17.030822563878868

n	rank	cond	Gauss error	inv error
5	5	1.000000000000001	2.6272671962866383e-16	1.5700924586837752e-16
5	5	10.000000000000002	1.1102230246251565e-16	1.7901808365247238e-16
5	5	999.9999999998864	5.634676625616551e-14	5.1220256856965476e-14
5	5	1.000000000442854e7	1.172346560482253e-10	1.7048113679425996e-10
5	5	9.999338663012544e11	3.249074318686922e-5	3.622958395563893e-5
5	4	5.457174077444961e15	0.02878417633256983	0.09514345280327031
10	10	1.0000000000000016	3.528343968566428e-16	2.6506211417561425e-16
10	10	10.000000000000014	6.319500243438061e-16	6.435464048854839e-16
10	10	999.9999999999837	2.935737029873833e-14	2.923780292605362e-14
10	10	9.99999999308061e6	1.0457293971517006e-10	1.1260117419109273e-10
10	10	1.0000594421795704e12	1.0738718999068445e-5	1.2419067935840853e-5
10	9	7.068496140187692e15	0.2717050897386556	0.3053078620425619
15	15	1.0000000000000013	4.459357425957953e-16	3.4279353213493766e-16
15	15	10.000000000000009	5.867750130370136e-16	3.3183103333940837e-16
15	15	999.9999999999902	5.4920329025934675e-15	9.973074373863652e-15
15	15	9.999999994809046e6	1.149767679852658e-10	1.415231426255696e-10
15	15	1.0000082780879507e12	1.3968369305910557e-5	1.8357095229145035e-5
15	14	6.283649419112325e15	0.32277751345850186	0.3230446756604018

Można zauważyć, że zarówno dla macierzy Hilberta, jak i dla losowej wartość błędu względnego rośnie proporcjonalnie do wartości wskaźnika uwarunkowania.

3.4 Wnioski

Ogromny wpływ na dokładność obliczeń ma wskaźnik uwarunkowania macierzy. Rozmiar macierzy i rodzaj algorytmu również oddziałuje na wyniki, jednak ich znaczenie jest pomijalne przy tak dużych zmianach *cond*.

4 Zadanie 4.

4.1 Opis problemu

Celem zadania jest sprawdzenie wartości wielomianu Wilkinsona (w postaci naturalnej i iloczynowej) dla 20 zer tego wielomianu obliczonych przy pomocy funkcji `roots` z pakietu `Polynomials`. Następnie należy powtórzyć eksperyment dla zaburzonego współczynnika przy x^{19} (zmiana z -210 na $-210 - 2^{-23}$).

4.2 Rozwiązanie

W `Julii` tworzymy wielomiany ze współczynników, obliczamy zera przy pomocy funkcji `roots`, a następnie iterujemy po tych wartościach i sprawdzamy wyniki.

```
poly_a = Polynomial(reverse(P_a))
poly_b = Polynomial(reverse(P_b))

roots_a = roots(poly_a)
roots_b = roots(poly_b)
```

4.3 Wyniki i interpretacja

Poniższe tabele przedstawiają wyniki kolejno dla podpunktów *a*) oraz *b*).

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	0.9999999999996989	36352.0	36626.425482422805	3.0109248427834245e-13
2	2.0000000000283182	181760.0	181303.93367257662	2.8318236644508943e-11
3	2.9999999995920965	209408.0	290172.2858891686	4.0790348876384996e-10
4	3.9999999837375317	3.106816e6	2.0415372902750901e6	1.626246826091915e-8
5	5.000000665769791	2.4114688e7	2.0894625006962188e7	6.657697912970661e-7
6	5.999989245824773	1.20152064e8	1.1250484577562995e8	1.0754175226779239e-5
7	7.000102002793008	4.80398336e8	4.572908642730946e8	0.00010200279300764947
8	7.999355829607762	1.682691072e9	1.5556459377357383e9	0.0006441703922384079
9	9.002915294362053	4.465326592e9	4.687816175648389e9	0.002915294362052734
10	9.990413042481725	1.2707126784e10	1.2634601896949205e10	0.009586957518274986
11	11.025022932909318	3.5759895552e10	3.300128474498415e10	0.025022932909317674
12	11.953283253846857	7.216771584e10	7.388525665404988e10	0.04671674615314281
13	13.07431403244734	2.15723629056e11	1.8476215093144193e11	0.07431403244734014
14	13.914755591802127	3.65383250944e11	3.5514277528420844e11	0.08524440819787316
15	15.075493799699476	6.13987753472e11	8.423201558964254e11	0.07549379969947623
16	15.946286716607972	1.555027751936e12	1.570728736625802e12	0.05371328339202819
17	17.025427146237412	3.777623778304e12	3.3169782238892363e12	0.025427146237412046
18	17.99092135271648	7.199554861056e12	6.34485314179128e12	0.009078647283519814
19	19.00190981829944	1.0278376162816e13	1.228571736671966e13	0.0019098182994383706
20	19.999809291236637	2.7462952745472e13	2.318309535271638e13	0.00019070876336257925

k	z_k	$\lvert P(z_k)\rvert$	$\lvert p(z_k)\rvert$	$\lvert z_k - k\rvert$
1	0.9999999999998357 + 0.0im	20496.0	19987.872313406835	1.6431300764452317e-13
2	2.0000000000550373 + 0.0im	339570.0	352369.4138087958	5.503730804434781e-11
3	2.99999999660342 + 0.0im	2.2777455e6	2.4162415582518433e6	3.3965799062229962e-9
4	4.000000089724362 + 0.0im	1.0488020625e7	1.1263702300292023e7	8.972436216225788e-8
5	4.99999857388791 + 0.0im	4.1239073125e7	4.475744423806908e7	1.4261120897529622e-6
6	6.000020476673031 + 0.0im	1.406328934140625e8	2.1421031658039317e8	2.0476673030955794e-5
7	6.99960207042242 + 0.0im	4.122812662421875e8	1.7846173427860644e9	0.00039792957757978087
8	8.007772029099446 + 0.0im	1.0307901272578125e9	1.8686972170009857e10	0.007772029099445632
9	8.915816367932559 + 0.0im	2.1574055781816406e9	1.3746309775142993e11	0.0841836320674414
10	10.095455630535774 - 0.6449328236240688im	9.384147605647182e9	1.490069535200058e12	0.6519586830380407
11	10.095455630535774 + 0.6449328236240688im	9.384147605647182e9	1.490069535200058e12	1.1109180272716561
12	11.793890586174369 - 1.6524771364075785im	3.0012060598372482e10	3.2962792355717145e13	1.665281290598479
13	11.793890586174369 + 1.6524771364075785im	3.0012060598372482e10	3.2962792355717145e13	2.0458202766784277
14	13.992406684487216 - 2.5188244257108443im	2.0030917431984006e11	9.546022365750216e14	2.518835871190904
15	13.992406684487216 + 2.5188244257108443im	2.0030917431984006e11	9.546022365750216e14	2.7128805312847097
16	16.73074487979267 - 2.812624896721978im	1.1583329328642004e12	2.742106076928478e16	2.9060018735375106
17	16.73074487979267 + 2.812624896721978im	1.1583329328642004e12	2.742106076928478e16	2.825483521349608
18	19.5024423688181 - 1.940331978642903im	5.867381806750561e12	4.2524858765203725e17	2.4540214463129764
19	19.5024423688181 + 1.940331978642903im	5.867381806750561e12	4.2524858765203725e17	2.0043294443099486
20	20.84691021519479 + 0.0im	9.550552334336e12	1.37437435599976e18	0.8469102151947894

Wyniki znacznie odbiegają od prawidłowych. Wartość każdego obliczonego przy pomocy `roots` zera mija się z wartością dokładną z powodu trudności problemu znajdowania zer wielomianów oraz błędów reprezentacji. Z kolei w trakcie

obliczania wartości wielomianów w tych zerach dochodzi do utraty cyfr znaczących, a w przypadku postaci iloczynowej wykonujemy mnożenie przypominające działanie silni.

Oceniając wpływ małego zaburzenia współczynnika przy x^{19} możemy wnioskować, że znacznie wpłynął na znalezione zera, przy czym różnica $|z_k - k|$ rośnie dużo gwałtowniej niż w podpunkcie a).

4.4 Wnioski

Zadanie wyznaczania pierwiastków wielomianów jest źle uwarunkowane, bardzo niewielkie zaburzenia powodują narastanie błędów i w konsekwencji bardzo niedokładne wyniki.

5 Zadanie 5.

5.1 Opis problemu

Celem zadania jest zbadanie równania rekurencyjnego $p_{n+1} = p_n + rp_n(1 - p_n)$, które reprezentuje model logistyczny, 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.
Dla danych $p_0 = 0.01$ i $r = 3$ należy wykonać 40 iteracji tego równania, a następnie ponownie 40 iteracji, tylko po 10 iteracji obciąć wynik do trzech miejsc po przecinku. Trzeba porównać również arytmetyki `Float32` i `Float64`.

5.2 Rozwiązanie

Obliczamy w pętli wartości kolejnych wyrazów dla danej liczby iteracji oraz typu.

```
function calc(p0, r, type, iterations)
    p::type = p0
    r::type  = r

    println(type)
    for i in 1:iterations
        p = p + r*p*(type(1.0) - p)
        println(p)
    end
    println()
    return p
end
```

5.3 Wyniki i interpretacja

Poniższa tabela przedstawia wyniki obliczeń.

`Float32`	`Float32` truncated	`Float64`
0.0397	0.0397	0.0397
0.15407173	0.15407173	0.15407173000000002
0.5450726	0.5450726	0.5450726260444213
1.2889781	1.2889781	1.2889780011888006
0.1715188	0.1715188	0.17151914210917552
0.5978191	0.5978191	0.5978201201070994
1.3191134	1.3191134	1.3191137924137974
0.056273222	0.056273222	0.056271577646256565
0.21559286	0.21559286	0.21558683923263022
0.7229306	0.722	0.722914301179573
1.3238364	1.3241479	1.3238419441684408
0.037716985	0.036488414	0.03769529725473175
0.14660022	0.14195944	0.14651838271355924
0.521926	0.50738037	0.521670621435246
1.2704837	1.2572169	1.2702617739350768
0.2395482	0.28708452	0.24035217277824272
0.7860428	0.9010855	0.7881011902353041
1.2905813	1.1684768	1.2890943027903075
0.16552472	0.577893	0.17108484670194324
0.5799036	1.3096911	0.5965293124946907
1.3107498	0.09289217	1.3185755879825978
0.088804245	0.34568182	0.058377608259430724
0.3315584	1.0242395	0.22328659759944824
0.9964407	0.94975823	0.7435756763951792
1.0070806	1.0929108	1.315588346001072
0.9856885	0.7882812	0.07003529560277899
1.0280086	1.2889631	0.26542635452061003
0.9416294	0.17157483	0.8503519690601384
1.1065198	0.59798557	1.2321124623871897
0.7529209	1.3191822	0.37414648963928676
1.3110139	0.05600393	1.0766291714289444
0.0877831	0.21460639	0.8291255674004515
0.3280148	0.7202578	1.2541546500504441
0.9892781	1.3247173	0.29790694147232066
1.021099	0.034241438	0.9253821285571046
0.95646656	0.13344833	1.1325322626697856
1.0813814	0.48036796	0.6822410727153098
0.81736827	1.2292118	1.3326056469620293
1.2652004	0.3839622	0.0029091569028512065
0.25860548	1.093568	0.011611238029748606

W arytmetyce `Float32` różnicę zauważamy w pierwszej iteracji po obcięciu,

a w każdych kolejnych wyniki coraz bardziej się różnią, aż w końcu utracona zostaje jakakolwiek korelacja. To samo dzieje się dla `Float64`, chociaż tutaj wymagane jest więcej iteracji żeby ten efekt nastąpił. Zjawisko to jest spowodowane narastaniem błędów zaokrągleń oraz utratą cyfr znaczących w naszym układzie sprzężenia zwrotnego.

5.4 Wnioski

Obliczany przez nas model logistyczny jest numerycznie niestabilny, ponieważ niewielkie błędy w początkowym stadium procesu kumulują się w kolejnych stadiach. powodując znaczną utratę dokładności.

6 Zadanie 6.

6.1 Opis problemu

Celem zadania jest zbadanie równania rekurencyjnego $x_{n+1} = x_n^2 + c$, gdzie c jest pewną stałą.

Dla danych:

- $c = -2$ i $x_0 = 1$
- $c = -2$ i $x_0 = 2$
- $c = -2$ i $x_0 = 1.99999999999999$
- $c = -1$ i $x_0 = 1$
- $c = -1$ i $x_0 = -1$
- $c = -1$ i $x_0 = 0.75$
- $c = -1$ i $x_0 = 0.25$

należy wykonać 40 iteracji tego równania w `Float64`, a następnie opisać zachowanie generowanych ciągów.

6.2 Rozwiązanie

Obliczamy w pętli wartości kolejnych wyrazów dla zadanej liczby iteracji, w typie `Float64`.

```
function calc(x0, c, iterations)
    x::Float64 = x0
    c::Float64  = c
```

```
    seq = zeros(0)
    for i in 1:iterations
        x = x^2 + c
        append!(seq, x)
    end
    return seq
end
```

6.3 Wyniki i interpretacja

Poniższa tabela przedstawia wyniki obliczeń.

n	a	b	c	d	e	f	g
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

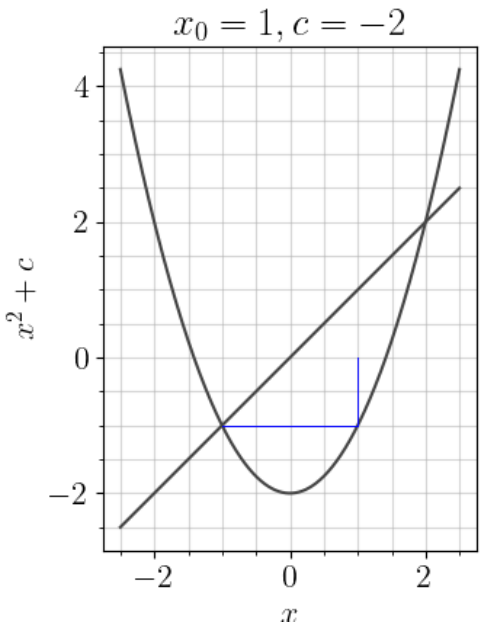
Dla przykładów *a*) *b*) *d*) i *e*) wartości są stałe lub cyklicznie się zmieniają, są zatem stabilne. Dla *f*) i *g*) stan stabilny uzyskuje się dopiero po skończonej liczbie iteracji. Natomiast dla *c*) wartości nie zbiegają do stałych punktów, od pewnej iteracji są chaotyczne. Powodem takiego zachowania jest kumulacja

błędów.

To samo zachowanie w formie iteracji graficznej:

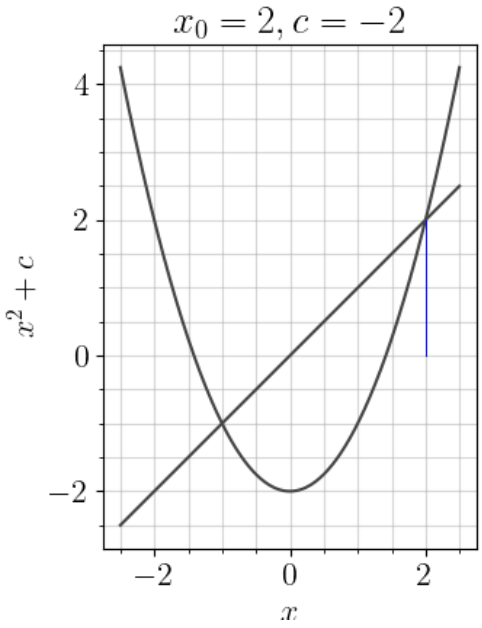
6.4 Wnioski

Niestabilny numerycznie proces powoduje poważną utratę dokładności. Analiza błędu nie jest łatwa do przeprowadzenia, ponieważ niestabilność nie wynika z własności operacji matematycznych, ale z samych danych jakie podamy do algorytmu.



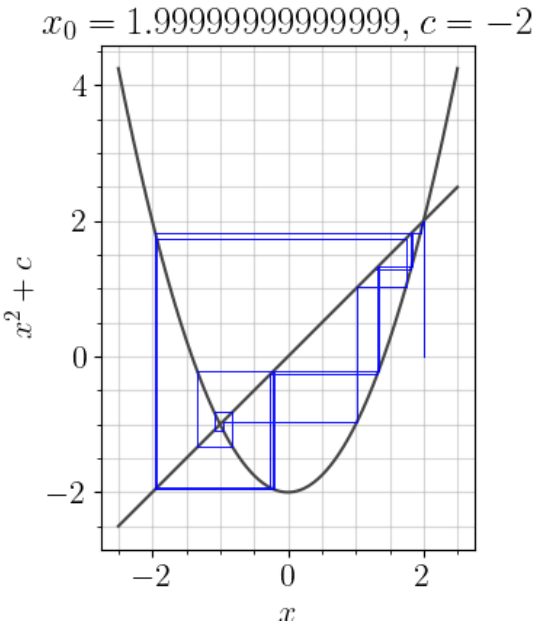
.5

Float32, zakres [-10,25]



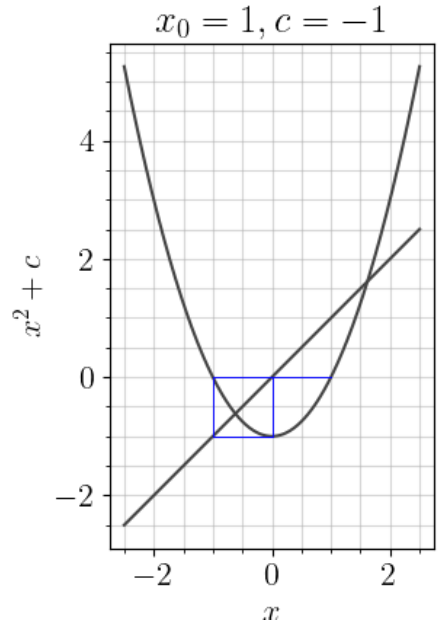
.5

Float32, zakres [-10,50]



.5

Float64, zakres [30,40]



.5

Float64, zakres [-10,50]