on-lista2

Marcin Zubrzycki

November 2024

1 Zadanie 1

1.1 Krótki opis problemu

Nieznacznie zmieniliśmy wartości zmiennych z ostatniej listy. Jaki to ma wpływ na wynik algorytmów obliczenia iloczynu skalarnego dwóch wektorów?

1.2 Rozwiazanie

Kod źródłowy zawarłem w pliku `1.jl`. Uruchomiłem algorytmy w przód, w tył, cześciowy w przód i cześciowy w tył dla nowych wartości

$$x2 = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$$

i porównałem z wynikami dla starych wartości

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

1.3 Wyniki oraz ich interpretacja

	Float32	Float64
algo1	3.6379788070917129516601562 5e-12	0.0042963428424103990111923567773999366181669756770133
algo2	3.6379788070917129516601562 5e-12	0.0042963428424103990111923567773999366181669756770133
algo3	0.0	0.004296342842280865
algo4	0.0	0.004296342842280865

Wartości poszczególnych komórek sa wynikiem odjecia $\text{algo}(x, y) - \text{algo}(x2, y)$

1.4 Wnioski

Mała zmiana prowadzi do dużej różnicy w wynikach - zadanie jest źle uwarunkowane.

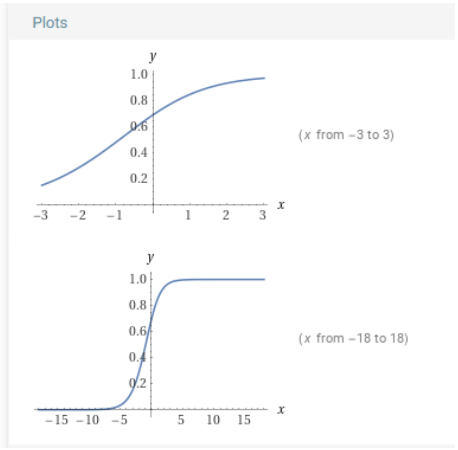

Figure 1: Wykres z WolframAlpha

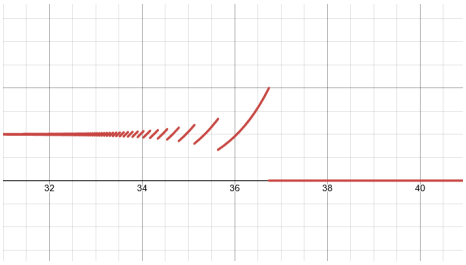

Figure 2: Wykres z Desmosa

2 Zadanie 2

2.1 Krótki opis problemu

Należało sprawdzić, jak wyglada wykres funkcji w różnych programach do wizualizacji i porównać z analitycznie wyliczona granica funkcji kiedy $x \to \infty$

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

2.2 Rozwiazanie

Analitycznie wyliczona granica funkcji:

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

Wykresy wyżej.

2.3 Wyniki oraz ich interpretacja

Dla x około 36 wykres przestaje odwzorowywać prawdziwa wartość funkcji

2.4 Wnioski

Zadanie jest źle uwarunkowane. Użyta jest zbyt mała precyzja, aby dokładnie policzyć wartość funkcji. Wynika to z faktu, że dla dużych x zachodzi $1+x \approx 1$, wiec $\ln(1+x) \approx 0$

3 Zadanie 3

3.1 Krótki opis problemu

Należy porównać wyniki dwóch algorytmów na obliczanie układów równań liniowych

$$Ax = b$$

dla danej macierzy współczynników i wektora prawych stron. Eksperymentowi poddane beda dwa rodzaje macierzy, macierz Hilberta n-tego stopnia oraz Macierz losowa o zadanym wskaźniku uwarunkowania. Metody obliczeń to eliminacja Gaussa oraz metoda odwrotności.

3.2 Rozwiazanie

Wygenerowałem Macierze Hilberta dla $n \in [2, 15]$ i policzyłem układ równań na dwa sposoby. Potem policzyłem bład wzgledny, czyli jak daleko jest wynikom od wektora jedynek. Zadanie powtórzyłem dla macierzy losowych dla wskaźników uwarunkowania $c = 1, 10, 10^3, 10^7, 10^{12}, 10^{16}$ i $n = 5, 10, 20$.

3.3 Wyniki oraz ich interpretacja

Wyniki dla macierzy Hilberta n-tego stopnia

n	Blad wzgledny Gauss	Blad wzgledny Inv
2	5.66e-16	1.4e-15
3	8.02e-15	0.0
4	4.14e-14	0.0
5	1.68e-12	3.35e-12
6	2.62e-10	2.02e-10
7	1.26e-8	4.71e-9
8	6.12e-8	3.08e-7
9	3.88e-6	4.54e-6
10	8.67e-5	0.00025
11	0.000158	0.00762
12	0.134	0.259
13	0.11	5.33
14	1.46	8.71
15	4.7	7.34

Wyniki dla macierzy losowej n-tego stopnia ze wskaźnikiem uwarunkowania c

n	c	Blad wzgledny Gauss	Blad wzgledny Inv
5	1	3.44e-16	2.43e-16
5	10	1.99e-16	3.29e-16
5	1000	3.34e-14	3.19e-14
5	1.0e7	9.38e-11	6.89e-11
5	1.0e12	2.37e-5	2.64e-5
5	1.0e16	0.0276	0.0736
10	1	3.67e-16	2.25e-16
10	10	3.06e-16	4.02e-16
10	1000	9.07e-15	6.5e-15
10	1.0e7	9.99e-11	8.09e-11
10	1.0e12	4.35e-5	4.18e-5
10	1.0e16	0.197	0.404
20	1	3.5e-16	4.23e-16
20	10	4.44e-16	3.22e-16
20	1000	2.79e-14	2.63e-14
20	1.0e7	9.58e-11	1.1e-10
20	1.0e12	3.61e-5	3.78e-5
20	1.0e16	0.122	0.106

3.4 Wnioski

Macierz losowa nawet z wysokim wskaźnikiem uwarunkowania jest bardziej stabilna w obliczeniach od macierzy Hilberta. Im wiekszy rzad macierzy tym wiekszy blad wzgledny w obliczeniach. Metoda Gaussa sprawdza sie dużo lepiej niż metoda z użyciem Inv dla macierzy Hilberta.

4 Zadanie 4

4.1 Krótki opis problemu

Rozpatrujemy jak Julia radzi sobie z obliczaniem miejsc zerowych złośliwego wielomianu Wilkinsona:

$$\prod_{i=1}^{20}(x-i)$$

Nastepnie lekko zaburzyć jeden ze współczyników o 2^{-23}

4.2 Rozwiazanie

Obliczenia zostały wykonane w pliku `4.jl`. $|P(z_k)|$, $|p(z_k)|$ i $|z_k - k|$ gdzie, $P()$ to Wielomian Wilkinsona w postaci naturalnej a $p()$ to wielomian w postaci iloczynowej.

4.3 Wyniki oraz ich interpretacja

Przed naniesieniem zaburzenia obliczanie zer wielomianu:

k	z_k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999996989	35696.50964788257	5.518479490350445e6	3.0109248427834245e-13
2	2.0000000000283182	176252.60026668405	7.37869762990174e19	2.8318236644508943e-11
3	2.9999999995920965	279157.6968824087	3.3204139316875795e20	4.0790348876384996e-10
4	3.9999999837375317	3.0271092988991085e6	8.854437035384718e20	1.626246826091915e-8
5	5.000000665769791	2.2917473756567076e7	1.8446752056545688e21	6.657697912970661e-7
6	5.999989245824773	1.2902417284205095e8	3.320394888870117e21	1.0754175226779239e-5
7	7.000102002793008	4.805112754602064e8	5.423593016891273e21	0.00010200279300764947
8	7.999355829607762	1.6379520218961136e9	8.262050140110275e21	0.0006441703922384079
9	9.002915294362053	4.877071372550003e9	1.196559421646277e22	0.002915294362052734
10	9.990413042481725	1.3638638195458128e10	1.655260133520688e22	0.009586957518274986
11	11.025022932909318	3.585631295130865e10	2.24783329792479e22	0.025022932909317674
12	11.953283253846857	7.533332360358197e10	2.886944688412679e22	0.04671674615314281
13	13.07431403244734	1.9605988124330817e11	3.807325552826988e22	0.07431403244734014
14	13.914755591802127	3.5751347823104315e11	4.612719853150334e22	0.08524440819787316
15	15.075493799699476	8.21627123645597e11	5.901011420218566e22	0.07549379969947623
16	15.946286716607972	1.5514978880494067e12	7.010874106897764e22	0.05371328339202819
17	17.025427146237412	3.694735918486229e12	8.568905825736165e22	0.025427146237412046
18	17.99092135271648	7.650109016515867e12	1.0144799361044434e23	0.009078647283519814
19	19.00190981829944	1.1435273749721195e13	1.1990376202371257e23	0.0019098182994383706
20	19.999809291236637	2.7924106393680727e13	1.4019117414318134e23	0.00019070876336257925

Prawidłowe wartości dla dokładnych wartości:

k	$\|P(k)\|$	$\|p(k)\|$
1	0.0	0
2	8192.0	0
3	27648.0	0
4	622592.0	0
5	2.176e6	0
6	8.84736e6	0
7	2.4410624e7	0
8	5.89824e7	0
9	1.45753344e8	0
10	2.27328e8	0
11	4.79074816e8	0
12	8.75003904e8	0
13	1.483133184e9	0
14	2.457219072e9	0
15	3.905712e9	0
16	6.029312e9	0
17	9.116641408e9	0
18	1.333988352e10	0
19	1.9213101568e10	0
20	2.7193344e10	0

Ponowienie eksperymentu Wilkinsona, czyli zaburzamy jeden współczynnik o 2^{-23}:

k	z_k	$\|P(z_k)\|$
1	0.9999999999998357 + 0.0i	20259.872313418207
2	2.0000000000550373 + 0.0i	346541.4137593836
3	2.99999999660342 + 0.0i	2.2580597001197007e6
4	4.000000089724362 + 0.0i	1.0542631790395478e7
5	4.99999857388791 + 0.0i	3.757830916585153e7
6	6.000020476673031 + 0.0i	1.3140943325569446e8
7	6.99960207042242 + 0.0i	3.939355874647618e8
8	8.007772029099446 + 0.0i	1.184986961371896e9
9	8.915816367932559 + 0.0i	2.2255221233077707e9
10	10.095455630535774 - 0.6449328236240688i	1.0677921232930157e10
11	10.095455630535774 + 0.6449328236240688i	1.0677921232930157e10
12	11.793890586174369 - 1.6524771364075785i	3.1401962344429485e10
13	11.793890586174369 + 1.6524771364075785i	3.1401962344429485e10
14	13.992406684487216 - 2.5188244257108443i	2.157665405951858e11
15	13.992406684487216 + 2.5188244257108443i	2.157665405951858e11
16	16.73074487979267 - 2.812624896721978i	4.850110893921027e11
17	16.73074487979267 + 2.812624896721978i	4.850110893921027e11
18	19.5024423688181 - 1.940331978642903i	4.557199223869993e12
19	19.5024423688181 + 1.940331978642903i	4.557199223869993e12
20	20.84691021519479 + 0.0i	8.756386551865696e12

4.4 Wnioski

Wyznaczone pierwiastki nie sa dokładne, ale sa bliskie do prawdziwych. Wielomian w postaci naturalnej nie jest zapisany dokładnie z uwagi na ograniczona precyzje arytmetyki Float64. Bardzo niewielkie oddalenie od faktycznych pierwiastków powoduje, że wartości funkcji odbiegaja bardzo od prawdziwych wartości w góre.

5 Zadanie 5

5.1 Krótki opis problemu

Sprawdzić jak beda wygladały wyniki kolejnych iteracji równania rekurencyjnego

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

gdzie r jest stała 3 a p_0 równe jest 0.01. Oraz jak beda wygladały po obcieciu liczb do trzeciej po przecinku po dziesieciu iteracjach we Float32. Potem porównać z wynikami dla Float64

5.2 Rozwiazanie

Wyliczamy kolejne wartości korzystajac z kodu zawartego w pliku `5.jl`

5.3 Wyniki oraz ich interpretacja

Iteracja	Float32	Float32 z obcieciem	Float64
0	0.01	0.01	0.01
1	0.0397	0.0397	0.039699998
2	0.15407173	0.15407173	0.15407172
3	0.5450726	0.5450726	0.5450726
4	1.2889781	1.2889781	1.288978
5	0.1715188	0.1715188	0.17151922
6	0.5978191	0.5978191	0.59782034
7	1.3191134	1.3191134	1.3191139
8	0.056273222	0.056273222	0.05627135
9	0.21559286	0.21559286	0.21558599
10	0.7229306	0.722	0.722912
11	1.3238364	1.3241479	1.3238428
12	0.037716985	0.036488414	0.037692066
13	0.14660022	0.14195944	0.14650619
14	0.521926	0.50738037	0.52163255
15	1.2704837	1.2572169	1.2702286
16	0.2395482	0.28708452	0.24047223
17	0.7860428	0.9010855	0.7884083
18	1.2905813	1.1684768	1.2888702
19	0.16552472	0.577893	0.17192157
20	0.5799036	1.3096911	0.5990152
21	1.3107498	0.09289217	1.3196032
22	0.088804245	0.34568182	0.054354966
23	0.3315584	1.0242395	0.20855647
24	0.9964407	0.94975823	0.7037385
25	1.0070806	1.0929108	1.3292104
26	0.9856885	0.7882812	0.016440736
27	1.0280086	1.2889631	0.06495205
28	0.9416294	0.17157483	0.2471519
29	1.1065198	0.59798557	0.80535537
30	0.7529209	1.3191822	1.2756296
31	1.3110139	0.05600393	0.22082563
32	0.0877831	0.21460639	0.73701066
33	0.3280148	0.7202578	1.3184885
34	0.9892781	1.3247173	0.05871831
35	1.021099	0.034241438	0.22452971
36	0.95646656	0.13344833	0.746878
37	1.0813814	0.48036796	1.3140317
38	0.81736827	1.2292118	0.07608879
39	1.2652004	0.3839622	0.28698665
40	0.25860548	1.093568	0.9008626

5.4 Wnioski

Obciecie liczb po trzecim miejscu po przecinku sprawiło że wynik po 40 iteracjach jest kompletnie inny od tego, który osiagneliśmy bez odrzucania precyzji. Podobnie, jeśli przeznaczymy wiecej pamieci na zapamietanie każdej z liczb wyniki również sa inne niż we Float32 - Można spodziewać sie, że sa dokładniejsze.

6 Zadanie 6

6.1 Krótki opis problemu

Jak zachowuje sie ciag wyrażony wzorem rekurencyjnym

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

gdy $c = -2, x_0 \in \{1, 2, 1.99999999999999\}$ i $c = -1, x_0 \in \{1, -1, 0.75, 0.25\}$

6.2 Rozwiazanie

Oblicznia wykonalem w pliku `6.jl`

6.3 Wyniki oraz ich interpretacja

Case 1: $c = -2.0, x_0 = 1.0$	
0	1.0
1	-1.0
2	-1.0
3	-1.0
4	-1.0
5	-1.0
6	-1.0
7	-1.0
8	-1.0
9	-1.0
10	-1.0
11	-1.0
12	-1.0
13	-1.0
14	-1.0
15	-1.0
16	-1.0
17	-1.0
18	-1.0
19	-1.0
20	-1.0
21	-1.0
22	-1.0
23	-1.0
24	-1.0
25	-1.0
26	-1.0
27	-1.0
28	-1.0
29	-1.0
30	-1.0
31	-1.0
32	-1.0
33	-1.0
34	-1.0
35	-1.0
36	-1.0
37	-1.0
38	-1.0
39	-1.0
40	-1.0

Case 2: $c = -2.0, x_0 = 2.0$	
0	2.0
1	2.0
2	2.0
3	2.0
4	2.0
5	2.0
6	2.0
7	2.0
8	2.0
9	2.0
10	2.0
11	2.0
12	2.0
13	2.0
14	2.0
15	2.0
16	2.0
17	2.0
18	2.0
19	2.0
20	2.0
21	2.0
22	2.0
23	2.0
24	2.0
25	2.0
26	2.0
27	2.0
28	2.0
29	2.0
30	2.0
31	2.0
32	2.0
33	2.0
34	2.0
35	2.0
36	2.0
37	2.0
38	2.0
39	2.0
40	2.0

Case 3: $c = -2.0, x_0 = 1.99999999999999$	
0	1.99999999999999
1	1.99999999999996
2	1.9999999999998401
3	1.9999999999993605
4	1.999999999997442
5	1.9999999999897682
6	1.9999999999590727
7	1.999999999836291
8	1.9999999993451638
9	1.9999999973806553
10	1.999999989522621
11	1.9999999580904841
12	1.9999998323619383
13	1.9999993294477814
14	1.9999973177915749
15	1.9999892711734937
16	1.9999570848090826
17	1.999828341078044
18	1.9993133937789613
19	1.9972540465439481
20	1.9890237264361752
21	1.9562153843260486
22	1.82677862987391
23	1.3371201625639997
24	-0.21210967086482313
25	-1.9550094875256163
26	1.822062096315173
27	1.319910282828443
28	-0.2578368452837396
29	-1.9335201612141288
30	1.7385002138215109
31	1.0223829934574389
32	-0.9547330146890065
33	-1.0884848706628412
34	-0.8152006863380978
35	-1.3354478409938944
36	-0.21657906398474625
37	-1.953093509043491
38	1.8145742550678174
39	1.2926797271549244
40	-0.3289791230026702

Case 4: $c = -1.0, x_0 = 1.0$	
0	1.0
1	0.0
2	-1.0
3	0.0
4	-1.0
5	0.0
6	-1.0
7	0.0
8	-1.0
9	0.0
10	-1.0
11	0.0
12	-1.0
13	0.0
14	-1.0
15	0.0
16	-1.0
17	0.0
18	-1.0
19	0.0
20	-1.0
21	0.0
22	-1.0
23	0.0
24	-1.0
25	0.0
26	-1.0
27	0.0
28	-1.0
29	0.0
30	-1.0
31	0.0
32	-1.0
33	0.0
34	-1.0
35	0.0
36	-1.0
37	0.0
38	-1.0
39	0.0
40	-1.0

Case 5: $c = -1.0, x_0 = -1.0$	
0	-1.0
1	0.0
2	-1.0
3	0.0
4	-1.0
5	0.0
6	-1.0
7	0.0
8	-1.0
9	0.0
10	-1.0
11	0.0
12	-1.0
13	0.0
14	-1.0
15	0.0
16	-1.0
17	0.0
18	-1.0
19	0.0
20	-1.0
21	0.0
22	-1.0
23	0.0
24	-1.0
25	0.0
26	-1.0
27	0.0
28	-1.0
29	0.0
30	-1.0
31	0.0
32	-1.0
33	0.0
34	-1.0
35	0.0
36	-1.0
37	0.0
38	-1.0
39	0.0
40	-1.0

Case 6: $c = -1.0, x_0 = 0.75$	
0	0.75
1	-0.4375
2	-0.80859375
3	-0.3461761474609375
4	-0.8801620749291033
5	-0.2253147218564956
6	-0.9492332761147301
7	-0.0989561875164966
8	-0.9902076729521999
9	-0.01948876442658909
10	-0.999620188061125
11	-0.0007594796206411569
12	-0.9999994231907058
13	-1.1536182557003727e-6
14	-0.9999999999986692
15	-2.6616486792363503e-12
16	-1.0
17	0.0
18	-1.0
19	0.0
20	-1.0
21	0.0
22	-1.0
23	0.0
24	-1.0
25	0.0
26	-1.0
27	0.0
28	-1.0
29	0.0
30	-1.0
31	0.0
32	-1.0
33	0.0
34	-1.0
35	0.0
36	-1.0
37	0.0
38	-1.0
39	0.0
40	-1.0

Case 7: $c = -1.0, x_0 = 0.25$	
0	0.25
1	-0.9375
2	-0.12109375
3	-0.9853363037109375
4	-0.029112368589267135
5	-0.9991524699951226
6	-0.0016943417026455965
7	-0.9999971292061947
8	-5.741579369278327e-6
9	-0.9999999999670343
10	-6.593148249578462e-11
11	-1.0
12	0.0
13	-1.0
14	0.0
15	-1.0
16	0.0
17	-1.0
18	0.0
19	-1.0
20	0.0
21	-1.0
22	0.0
23	-1.0
24	0.0
25	-1.0
26	0.0
27	-1.0
28	0.0
29	-1.0
30	0.0
31	-1.0
32	0.0
33	-1.0
34	0.0
35	-1.0
36	0.0
37	-1.0
38	0.0
39	-1.0
40	0.0

6.4 Wnioski

Wyniki sa zgodne z oczekiwaniami w przypadku c i x_0 całkowitych, jednak jeśli korzystamy z wartości z cyframi po przecinku wartości powoli zbiegaja sie do całkowitych. Dla liczb bardzo bliskim całkowitym algorytm jest niestabilny. Skończona dokładność arytmetyki sprawia że poczatkowo zapamietane wartości powoduja wyniki którym nie można ufać z uwagi na kumulacje bledów.