Sprawozdanie

OBLICZENIA NAUKOWE, LISTA NR 2

ŁUKASZ KLEKOWSKI 229738

Wrocław 12.11.2017

1. Zadanie pierwsze

1.1.Opis problemu

Polecenie polegało na powtórzeniu zadania piątego z listy pierwszej, które mówiło aby obliczyć iloczyn skalarny dwóch wektorów za pomocą czterech algorytmów używając precyzji ***Float32*** oraz ***Float64***.

Tym razem należało usunąć z końca czwartej współrzędnej x cyfrę 9 oraz z końca 5. Współrzędnej x cyfrę 7.

1.2.Rozwiązanie problemu

Pierwszy algorytm polegał na zwykłym obliczeniu produktu skalarnego sumując po kolei iloczyny.

Drugi algorytm robił to w odwrotnej kolejności.

Trzeci algorytm polegał na dodawaniu wyników mnożenia składowych wektora od najmniejszej wartości do największej, a czwarty algorytm robił to na odwrót.

1.3.Wyniki

Wyniki z 1. listy:

Algorytm	Float32	Float64	Błąd względny F32	Błąd względny F64
I	-4.9994429945e-1	1.02518813e-10	4.996159e10	1.1245e1
II	-4.5434570312500e-1	-1.56433088e-10	4.540473e10	1.463e1
III	-5.000e-1	0	4.996716e10	-
IV	-5.000e-1	0	4.9967165e10	-

Wyniki z listy 2. :

Algorytm	Float32	Float64	Błąd względny F32	Błąd względny F64
I	-4.9994429946e-01	-4.2963427399e-03	4.9961598976e+10	4.2935212800e+08
II	-4.5434570313e-01	-4.2963429987e-03	4.5404733440e+10	4.2935216000e+08
III	-5.0000000000e-01	4.2963428423e-03	4.9967165440e+10	4.2935216000e+08
IV	-5.0000000000e-01	4.2963428423e-03	4.9967165440e+10	4.2935216000e+08

1.4.Wnioski

W arytmetyce Float32 usunięcie ostatnich cyfr nie wpłynęło na otrzymane wyniki. Stało się tak ponieważ arytmetyka o pojedynczej precyzji pamięta jedynie około siedmiu cyfr znaczących więc gdy usunęliśmy dziesiątą cyfrę te zmiany nie mogły wpłynąć na wyniki.

W arytmetyce Float64 usunięcie cyfr spowodowało już zmiany. Błąd w I i II algorytmie jest teraz większy lecz w III i IV algorytmie otrzymaliśmy jakieś wyniki, a w pierwszej liście otrzymywaliśmy 0.

2. Zadanie drugie

2.1.Opis zadania

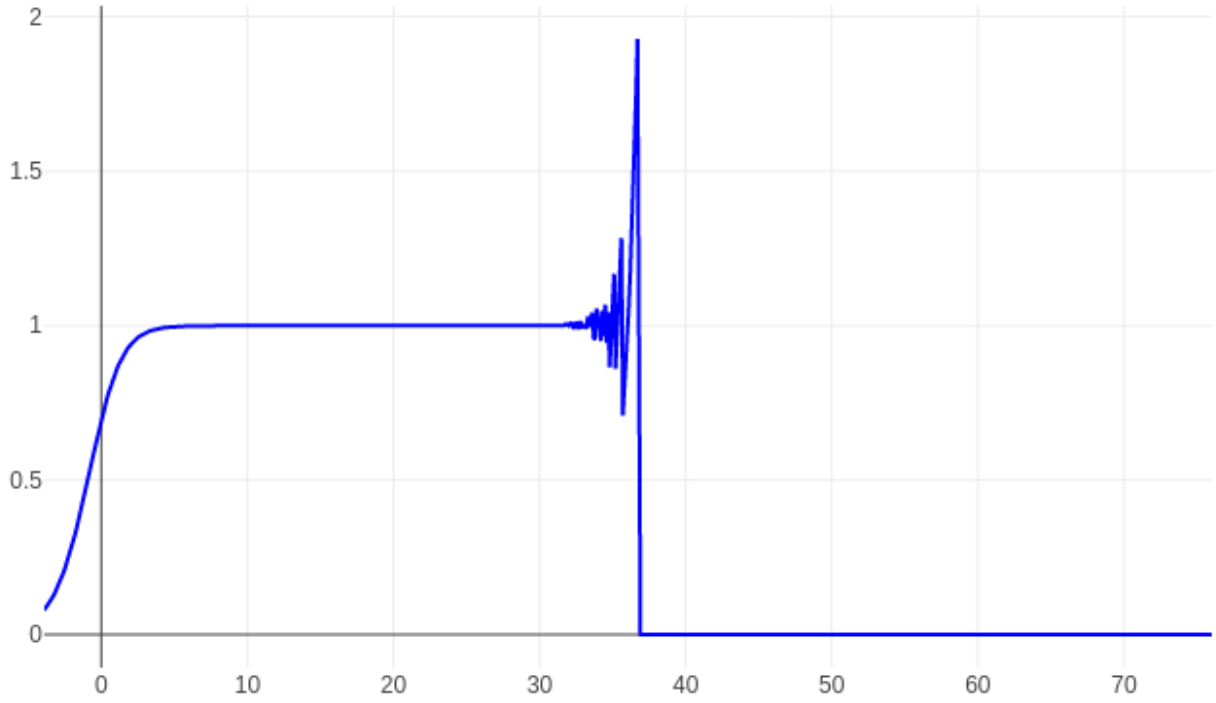
Polecenie polegało na narysowaniu wykresu funkcji $f(x) = e^x \ln(1 + e^{-x})$ w co najmniej dwóch dowolnych programach do wizualizacji. Następnie należało policzyć granicę tej funkcji $lim_{x\to\infty} f(x)$ i wynik porównać z otrzymanym wykresem.

2.2.Rozwiązanie

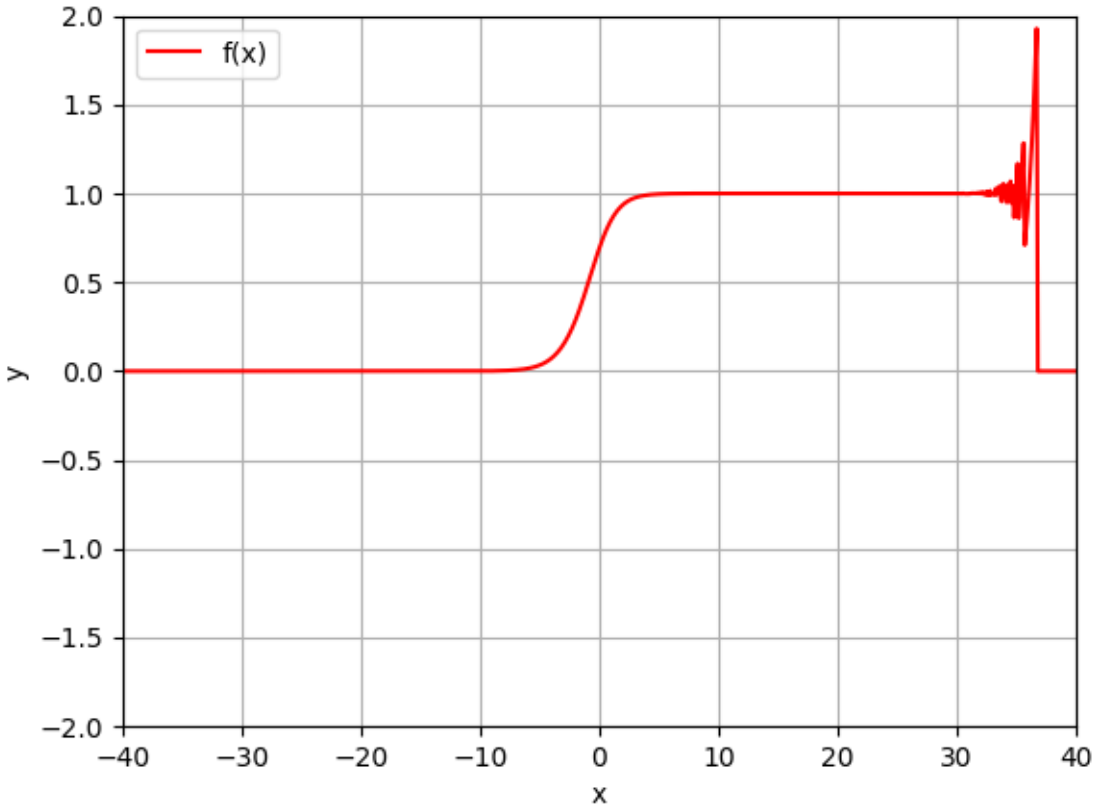
Napisałem w języku Julia oraz ***Python*** programy które obliczały daną funkcję a następnie narysowałem wykresy funkcji za pomocą biblioteki ***PlotlyJS*** w ***Julii*** i ***Matlibplot*** w ***Pythonie***. Granicę funkcji obliczyłem w ***Wolphramie Alpha***.

2.3.Wyniki

Wykres otrzymany w Julii:



Wykres otrzymany w Pythonie:



Wynik granicy otrzymany w Wolphramie Alpha:

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

2.4.Wnioski

Granica dążąca do nieskończoności dla tej funkcji wynosi 0. Na wykresach widać jednak co innego. W okolicach argumentu x = 34 wykres zaczyna się zachowywać inaczej, aż f(x) zaczyna być ciągle równe 0. Dzieje się tak ponieważ w logarytmie naturalnym mamy $(1 + e^{-x})$. Przy dużym x do 1 jest dodawana liczba mniejsza niż macheps i obliczany jest logarytm z 1 co daje 0.

3. Zadanie trzecie

3.1.Opis zadania

Polecenie polegało na rozwiązaniu układu równań liniowych $Ax = b$ gdzie A jest macierzą Hilberta lub macierzą wygenerowaną automatycznie. Należało użyć dwóch algorytmów: eliminacji Gaussa oraz mnożenia odwrotności macierzy przez werktor b, a następnie porównać obliczony x z dokładnym rozwiązaniem.

3.2.Opis rozwiązania

Macierze wygenerowałem za pomocą funkcji udostępnionych na stronie domowej. Następnie obliczałem wektor b mnożąc macierz A przez wektor x. Następnie posiadając wektor b liczyłem wektor x za pomocą eliminacji Gaussa czyli $x = A/b$ oraz $x = A^{-1} * b$.

3.3.Wyniki

Macierz Hilberta:

Stopień	Rząd	Wskaźnik uwarunkowania	Błąd wzg. Met. Gaussa	Błąd wzg. odwrotność
1	1	1.0	0.0	0.0
2	2	19.28147006790397	1.1102230246251565e-16	4.710277376051325e-16
3	3	524.0567775860644	3.7007434154171886e-16	0.0
4	4	15513.738738928929	2.55351295663786e-15	1.2079226507921703e-13
5	5	476607.25024224253	1.5099033134902129e-15	3.672164519483366e-12
6	6	1.495105864125091e7	5.6621374255882984e-14	2.402580729976363e-10
7	7	4.7536735637688667e8	3.391572736940621e-13	1.495728980293402e-9
8	8	1.5257575516147259e10	1.0985101717153611e-12	6.27831317062347e-9
9	9	4.9315408927806335e11	8.69423052070791e-12	3.0443575127492295e-7
10	10	1.6024859712306152e13	7.70294890095613e-9	3.7978201886807816e-5
11	11	5.2210348947688544e14	5.467844571701877e-6	0.002101464446245883
12	11	1.7255427417341868e16	0.005442334926016572	0.09768983047743608
13	11	7.126491965424366e17	1.9296015720313322	4.671860899891572
14	12	6.101307732044041e17	0.0795654857002479	2.771871454140025
15	12	4.223311222761075e17	0.5836908157447972	3.360075193492441
16	12	3.535827507735838e17	2.0345052265681667	53.392421220175216
17	12	3.1182808742153696e17	0.8148058696009675	4.808304710664456
18	12	1.5639169583348145e18	0.916736752762865	5.878323826631739

Macierze losowe

Stopień	Rząd	Wskaźnik uwarunkowania	Błąd wzg. Met. Gaussa	Błąd wzg. odwrotność
5	5	1.0000000000000007	0.0	0.0
5	5	10.000000000000002	1.9860273225978183e-16	1.9860273225978183e-16
5	5	999.9999999999527	2.303791694213469e-14	1.410079399044451e-14
5	5	1.0000000006169809e7	1.9860273225978183e-16	6.402852786689236e-11
5	5	1.0000135666477858e12	1.30433539424112e-6	3.814705996600516e-6
5	4	7.781970534309602e16	0.11836579522152672	0.06918850653193977
10	10	1.0000000000000016	0.0	1.4043333874306804e-16

10	10	10.000000000000002	2.808666774861361e-16	1.4043333874306804e-16
10	10	1000.0000000000108	7.021666937153401e-16	9.830333712014763e-16
10	10	1.000000000980536e7	1.3312027262802277e-10	1.2760208501545265e-10
10	10	1.000049055101323e12	6.3965479194835e-5	6.332412914081937e-5
10	9	4.94902951089104e15	0.3120335352038824	0.2980435056772572
20	20	1.0000000000000016	1.9860273225978183e-16	1.9860273225978183e-16
20	20	10.0	1.9860273225978183e-16	0.0
20	20	1000.0000000000105	2.383232787117382e-15	1.9860273225978185e-15
20	20	9.999999999766815e6	3.4448199998109484e-10	3.590685762546468e-10
20	20	9.999457901478606e11	5.280767244767937e-5	5.741117126367896e-5
20	19	2.24967985905202e16	0.1083361689066462	0.10003780868261967

3.4.Wnioski

W wykonanym eksperymencie wyraźnie widać że macierz Hilberta jest źle uwarunkowana. Im większy stopień macierzy tym większy wskaźnik uwarunkowania i większy błąd. Macierz ta składa się z wartości których nie da się zapisać dokładnie co powoduje widoczne tutaj błędy.
W macierzy losowej też wyraźnie widać, że im większy wskaźnik uwarunkowania tym większe powstają błędy.

4. Zadanie czwarte

4.1.Opis zadania

Polecenie polegało na obliczeniu pierwiastków wielomianu Wilkinsona w postaci naturalnej za pomocą biblioteki Polynomials. Następnie należało obliczyć wartość wielomianu w obliczonych miejscach zerowych (także dla wielomianu w postaci iloczynowej). Później polecenie mówiło aby zmienić jeden współczynnik z -210 na -210-2^-23 i powtórzyć eksperyment dla iloczynu w postaci naturalnej.

4.2.Rozwiązanie

Do obliczenia użyłem biblioteki Polynomials. Za pomocą funkcji Poly() utworzyłem wielomian naturalny. Funkcja poly() posłużyła mi do utworzenia wielomianu w postaci iloczynowej. Dzięki funkcji roots() obliczyłem zera wielomianu. polyval() umożliwiła mi policzenie wielomianu dla danej wartości.

4.3.Wyniki

Zera przed zmianą:

L.p.	Zera wielomianu	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	0.9999999999996989	5.517824e6	36352.0	3.010924842783424e-13
2	2.0000000000283182	7.378697629901744e19	181760.0	2.831823664450894e-11
3	2.9999999995920965	3.320413931687578e20	209408.0	4.079034887638499e-10
4	3.9999999837375317	8.854437035384718e20	3.106816e6	1.626246826091915e-8
5	5.000000665769791	1.8446752056545675e21	2.4114688e7	6.657697912970661e-7
6	5.999989245824773	3.320394888870126e21	1.20152064e8	1.0754175226779239e-5
7	7.000102002793008	5.423593016891272e21	4.80398336e8	0.0001020027930076494
8	7.999355829607762	8.26205014011023e21	1.682691072e9	0.0006441703922384079

9	9.002915294362053	1.196559421646318e22	4.465326592e9	0.002915294362052734
10	9.990413042481725	1.6552601335207813e22	1.2707126784e10	0.009586957518274986
11	11.025022932909318	2.2478332979247994e22	3.5759895552e10	0.025022932909317674
12	11.953283253846857	2.8869446884129956e22	7.216771584e10	0.04671674615314281
13	13.07431403244734	3.807325552825022e22	2.15723629056e11	0.07431403244734014
14	13.914755591802127	4.612719853149547e22	3.65383250944e11	0.08524440819787316
15	15.075493799699476	5.901011420239329e22	6.13987753472e11	0.07549379969947623
16	15.946286716607972	7.01087410689741e22	1.555027751936e12	0.05371328339202819
17	17.025427146237412	8.568905825727875e22	3.777623778304e12	0.025427146237412046
18	17.99092135271648	1.0144799361089491e23	7.199554861056e12	0.009078647283519814
19	19.00190981829944	1.1990376202486947e23	1.027837616281 6e13	0.0019098182994383706
20	19.999809291236637	1.4019117414364248e23	2.7462952745472e13	0.00019070876336257925

Zera po zmianie:

L.p.	Zera wielomianu	$\|P(z_k)\|$
1	0.9999999999998357 + 0.0im	20992.0
2	2.0000000000550373 + 0.0im	349184.0
3	2.99999999660342 + 0.0im	2.221568e6
4	4.000000089724362 + 0.0im	1.046784e7
5	4.99999857388791 + 0.0im	3.9463936e7
6	6.000020476673031 + 0.0im	1.29148416e8
7	6.99960207042242 + 0.0im	3.88123136e8
8	8.007772029099446 + 0.0im	1.072547328e9
9	8.915816367932559 + 0.0im	3.065575424e9
10	10.095455630535774 - 0.6449328236240688im	7.143113638035824e9
11	10.095455630535774 + 0.6449328236240688im	7.143113638035824e9
12	11.793890586174369 - 1.6524771364075785im	3.357756113171857e10
13	11.793890586174369 + 1.6524771364075785im	3.357756113171857e10
14	13.992406684487216 - 2.5188244257108443im	1.0612064533081976e11
15	13.992406684487216 + 2.5188244257108443im	1.0612064533081976e11
16	16.73074487979267 - 2.812624896721978im	3.315103475981763e11
17	16.73074487979267 + 2.812624896721978im	3.315103475981763e11
18	19.5024423688181 - 1.940331978642903im	9.539424609817828e12
19	19.5024423688181 + 1.940331978642903im	9.539424609817828e12
20	20.84691021519479 + 0.0im	1.114453504512e13

4.4.Wnioski

W podpunkcie a) nie otrzymaliśmy dokładnych pierwiastków ponieważ arytmetyka Float64 jest niewystarczająca. Posiada ona od 15 do 17 cyfr znaczących przez co niektóre współczynniki są ucinane co powoduje błąd w obliczeniach.

W podpunkcie b) w pierwiastkach otrzymaliśmy liczby zespolone. Dzieje się tak ponieważ wielomian Wilkinsona jest wrażliwy na nawet niewielkie zaburzenia współczynników przez co powstają duże zmiany wartości zer. Takie wielomiany nazywa się źle uwarunkowanymi.

5. Zadanie piąte

5.1.Opis zadania

Polecenie polegało na obliczeniu wartości wyrażenia $p_{n+1} = p_n + rp_n(1 - p_n)$ dla $n = 0,1, \ldots, 40$ r =3 i p_0 =0,01 w arytmetyce Float32, natępnie ponowić obliczenia tylko tym razem po 10 iteracji uciąć wynik do 3. Miejsca po przecinku. Ostatnie zadanie polegało na porównianiu 40 iteracji bez obcinania w arytmetyce Float32 oraz arytmetyce Float64.

5.2.Rozwiązanie

Napisałem w języku Julia program który obliczał dane wyrażenie

5.3.Wyniki

Pierwsza część zadania:

n	Wrażenie bez obcięcia	Wyrażenie z obcięciem
1	0.0397	0.0397
2	0.15407173	0.15407173
6	0.5978191	0.5978191
9	0.21559286	0.21559286
10	0.7229306	0.722
11	1.3238364	1.3241479
25	1.0070806	1.0929108
26	0.9856885	0.7882812
39	1.2652004	0.3839622
40	0.25860548	1.093568

Druga część zadania:

n	Wrażenie z Float32	Wyrażenie z Float64
1	0.0397	0.0397
2	0.15407173	0.15407173000000002
6	0.5978191	0.5978201201070994
9	0.21559286	0.21558683923263022
10	0.7229306	0.722914301179573
11	1.3238364	1.3238419441684408
25	1.0070806	1.315588346001072
26	0.9856885	0.07003529560277899
39	1.2652004	0.0029091569028512065
40	0.25860548	0.011611238029748606

5.4.Wnioski

Patrząc na wyniki końcowe z pierwszej części zadania widzimy że są zupełnie różne. Jest to wynik tak zwanej niestabilności numerycznej. Algorytm polegał na podnoszeniu do kwadratu liczby zmiennoprzecinkowej co powodowało podwojenie się ilości cyfr znaczących. Arytmetyka ma ograniczoną ich ilość więc komputer jest zmuszony je zaokrąglić. Tak zaokrąglona liczba jest podawana dalej w rekursji co powoduje skumulowanie się błędu.
To co napisałem wyżej widać także w drugiej części zadania gdzie użyliśmy arytmetyki Float64 która ma więcej cyfr znaczących przez co wyniki końcowe różnią się bardzo od tych z arytmetyki Float32. Przez

pewien czas otrzymywaliśmy zbliżone wyniki lecz od pewnego momentu ta arytmetyka też nie dawała już sobie rady przez niewystarczającą ilość cyfr znaczących.

6. Zadanie szóste

6.1.Opis zadania

Polecenie polegało na wykonaniu 40 iteracji wyrażenia $x_{n+1} = x_n^2 + c$ dla siedmiu różnych c oraz x_0

6.2.Wyniki i wnioski

Dla zestawu pierwszego oraz drugiego zgodnie z oczekiwaniami otrzymujemy za każdym razem odpowiednio -1 oraz 2.

Problem zaczyna się w zestawie trzecim. Błąd powstaje w taki sam sposób jak opisałem to w zadaniu 5. Oznacza to że już po pierwszej iteracji powstanie nam błąd i kolejne iteracje będą go powielać.

Zestawy 4. oraz 5. Zwracały odpowiednie wyniki (na przemian -1 oraz 0)

W dwóch ostatnich zestawach algorytm od pewnego momentu zaczął zwracać 0 i -1. To wyrażenie dla tych zestawów powinno dążyć do tych cyfr lecz dopiero w nieskończoności. To zjawisko nazywa się zjawiskiem stabilności, które jest pożądane przy wykonywaniu obliczeń. Jesteśmy w stanie przewidzieć wyniki dzięki temu że pomijane są małe błędy

L.p.	$x_0 = 1.9999999\ c = -2$	$x_0 = 0.75\ c = -1$	$x_0 = 0.25\ c = -1$
1	1.99999999999996	-0.4375	-0.9375
2	1.9999999999998401	-0.80859375	-0.12109375
3	1.9999999999993605	-0.3461761474609375	-0.9853363037109375
4	1.999999999997442	-0.8801620749291033	-0.029112368589267135
5	1.9999999999897682	-0.2253147218564956	-0.9991524699951226
6	1.9999999999590727	-0.9492332761147301	-0.0016943417026455965
10	1.999999989522621	-0.999620188061125	-6.593148249578462e-11
11	1.9999999580904841	-0.0007594796206411569	-1.0
12	1.9999998323619383	-0.9999994231907058	0.0
13	1.9999993294477814	-1.1536182557003727e-6	-1.0
14	1.9999973177915749	-0.9999999999986692	0.0
15	1.9999892711734937	-2.6616486792363503e-12	-1.0
16	1.9999570848090826	-1.0	0.0
17	1.999828341078044	0.0	-1.0
18	1.9993133937789613	-1.0	0.0
19	1.9972540465439481	0.0	-1.0
34	0.8152006863380978	-1.0	0.0
35	-1.3354478409938944	0.0	-1.0
39	1.2926797271549244	0.0	-1.0
40	-0.3289791230026702	-1.0	0.0