Obliczenia Naukowe - Lista 2

Paweł Dychus (244941)

Listopad 2019

Zadanie 1

Opis problemu

Obliczyć iloczyn skalarny wektorów, ze zmienionymi wartościami(względem zadania poprzedniego)
$x = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]$
$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$.
czterema sposobami i określić wpływ zmiany danych na wartości.

Rozwiązanie

Wystarczy podstawić nowe wartości do implementacji zadania poprzedniego.

Wyniki

Poniższe tabele przedstawiają, kolejne typy, wartości i różnice wartości(odległość) względem zadania z listy poprzedniej, dla czterech kolejnych algorytmów.

Algorytm 1		
Typ	Wynik	Różnica
Float64	-0.004296342739891585	0.004296342842410399
Float32	-0.4999443	0.0

Algorytm 2		
Typ	Wynik	Różnica
Float64	-0.004296342998713953	0.004296342842280865
Float32	-0.4543457	0.0

Algorytm 3		
Typ	Wynik	Różnica
Float64	-0.004296342842280865	0.004296342842280865
Float32	-0.5	0.0

Algorytm 4		
Typ	Wynik	Różnica
Float64	-0.004296342842280865	0.004296342842280865
Float32	-0.5	0.0

Jak można zauważyć, dla Float32 nie ma żadnej zmiany. Dla Float64, różnice są stosunkowo spore, już dla 3 miejsca po przecinku mamy zmianę wartości.

Wniosek

Niewielka zmiana wartości przyniosła znaczne, aczkolwiek nie olbrzymie zmiany w wyniku, rzędu 10^{-3}, dla zaburzeń na poziomie 10^{-10}. Na tej podstawie, można stwierdzić, że zadanie jest raczej źle uwarunkowane.

Zadanie 2

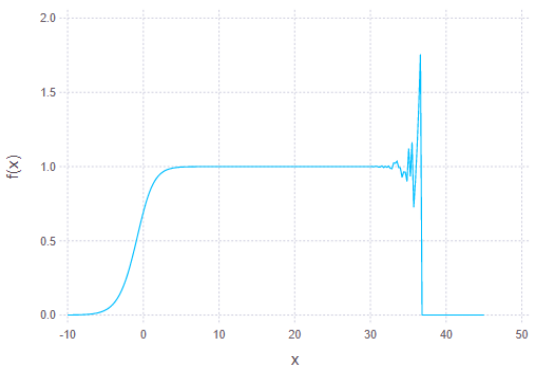
Problem

Wyznaczenie graficzne i matematyczne prawej granicy wykresu funkcji $f(x) = e^x ln(1 + e^{-x})$ oraz poprawność.

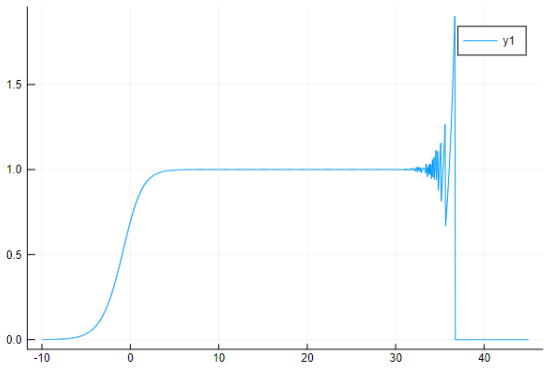
Rozwiązanie

Matematyczne rozwiązanie: $\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^x ln(1 + e^{-x}) = \lim_{x\to\infty} ln(1 + \frac{1}{e^x})^{e^x}, (e^x = y)$, $\lim_{y\to\infty} ln(1 + \frac{1}{y})^y = 1$

Graficznie poniższe wykresy funkcji:



(a) Wykres Gadfly



(b) Wykres Plot.jl

Rysunek 1 – Wykresy $f(x) = e^x ln(1 + e^{-x})$ na $x \in [-10, 50]$

Wyniki

Jak widać na wykresach, dla dostatecznie dużych x, wartości funkcji zaczynają fluktuować. Po czym, ostatecznie spada do wartości równej 0. Jest to błędne, ponieważ wiemy, że granica w rzeczywistości wynosi 1.

Wniosek

Wniosek jest prosty: e^{-x} zaczyna tracić precyzje, aż do osiągnięcia epsilona maszynowego. Po osiągnięciu wartości macheps, sumowanie z 1 wewnątrz logarytmu, zwraca zawsze wartość równą 1, a $ln(1) = 0$. Ponadto jest to dowód na to, że programy do wizualizacji, mogą wprowadzać w błąd.

Zadanie 3

Problem

Błędy przy obliczaniu układu równań metodą Gaussa oraz inwersji dla określonych macierzy: Macierz Hilberta i macierz losowa.

Rozwiązanie

Rozwiązanie polega na implementacji załączonych algorytmów, wypisaniu wyniku i poddaniu ich analizie.

Wyniki

Poniższe tabele przedstawiają wartości zastosowanych eskperymentów.

Hilbert				
n	rank	cond	err-Gauss	err-inverse
2	2	19.28147006790397	5.661048867003676e-16	1.4043333874306803e-15
3	3	524.0567775860644	8.022593772267726e-15	0.0
4	4	15513.73873892924	4.137409622430382e-14	0.0
5	5	476607.25024259434	1.68284262992271e-12	3.3544360584359632e-12
6	6	1.4951058642254665e7	2.6189133023116e-10	2.0163759404347654e-10
7	7	4.75367356583129e8	1.2606867224171e-8	4.713280397232037e-9
8	8	1.5257575538060041e10	6.124089555723e-8	3.07748390309622e-7
9	9	4.931537564468762e11	3.8751634185032e-6	4.541268303176643e-6
10	10	1.6024416992541715e13	8.67039023709e-5	0.0002501493411824886
11	10	5.222677939280335e14	0.00015827808158	0.007618304284315809
12	11	1.7514731907091464e16	0.13396208372	0.258994120804705
13	11	3.344143497338461e18	0.11039701117868264	**5.33127563942683**
14	11	6.200786263161444e17	**1.4554087127659643**	8.71499275104814
15	12	3.674392953467974e17	4.696668350857427	7.344641453111494
16	12	7.865467778431645e17	54.15518954564602	29.84884207073541
17	12	1.263684342666052e18	13.707236683836307	10.51694237836934
18	12	2.2446309929189128e18	9.134134521198485	7.575475905055309
19	13	6.471953976541591e18	9.720589712655698	12.233761393757726
20	13	1.3553657908688225e18	7.549915039472976	22.06269725787049

Dla powyższych wyników, można dostrzec ogromny wzrost wskaźnika uwarunkowania dla kolejnych wartości n. Oznacza, to że macierz Hilberta jest bardzo źle uwarunkowana. Przekłada się to na błąd wyniku, który już przy 13 iteracji przekracza 100% błędu względnego dla metody z inwersją i przy 14 iteracji dla metody Gaussa. Ponadto dla n = 11, tracimy prawidłowy stopień macierzy.

Random				
n	rank	cond	err-Gauss	err-inverse
5	5	1.0	2.1065000811460203e-16	1.5700924586837752e-16
5	5	10.0	2.808666774861361e-16	4.328446199157272e-16
5	5	1000.0	2.7741005665387162e-14	2.5483137506553956e-14
5	5	1.0e7	2.15133323313752e-10	1.478291244618629e-10
5	5	1.0e12	3.5803616730494477e-16	1.6523628822756612e-5
5	4	1.0e16	0.13486827020540584	0.19008632907181935
10	10	1.0	3.1597501217190306e-16	1.9229626863835638e-16
10	10	10.0	3.9720546451956367e-16	2.531698018113677e-16
10	10	1000.0	2.9930756179749472e-15	4.250284076336683e-15
10	10	1.0e7	2.381523200785395e-10	1.3177319689038872e-10
10	10	1.0e12	2.9195243512259457e-5	2.617332187665795e-5
10	9	1.0e16	0.0715705530062106	0.07407469572583474
20	20	1.0	5.994176260545586e-16	3.3583099550713974e-16
20	20	10.0	1.3414875847488985e-15	1.0441409976237532e-15
20	20	1000.0	1.7888859297457136e-14	1.48075618724414e-14
20	20	1.0e7	4.70165384989995e-10	4.4579493045128954e-10
20	20	1.0e12	5.757097268849091e-6	1.9804534585280597e-6
20	19	1.0e16	0.10537882939428717	0.08618572106576007

Zgodnie z przewidywaniami, macierz losowa o wskaźniku uwarunkowania równego 1.0e16, osiąga błędy względne bliskie 100%, a dla niskiego uwarunkowania, błędy rzędu 10^{-16}.

Wniosek

Wskaźnik uwarunkowania bezpośrednio wpływa na błędy w wynikach. Macierz Hilberta jest bardzo źle uwarunkowana, a macierz losowa z niskim wskaźnikiem uwarunkowania, generuje stosunkowo dokładne wyniki.

Zadanie 4

Problem

Problem polega na obliczeniu miejsc zerowych wielomianu Wilkinsona. Następnie do podstawienia znalezionych miejsc zerowych, oraz sprawdzenia poprawności miejsc zerowych. Ponadto podpunkt B. wymaga niewielkiej zmiany wartości jednego ze współczynników.

Rozwiązanie

Rozwiązanie polega na implementacji załączonych algorytmów: Poly, poly i użycia roots w celu znalezienia miejsc zerowych.

Wyniki

Poniższe tabele przedstawiają otrzymane wartości dla kolejno podpunktu A i B. Gdzie $P(z_k)$ to wielomian w postaci naturalnej. $p(z_k)$ to iloczyn kolejnych rzeczywistych miejsc zerowych, a $|z_k - k|$ to odległość do rzeczywistej pozycji miejsca zerowego.

Podpunkt A			
k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	36352.0	38400.0	3.0109248427834245e-13
2	181760.0	198144.0	2.8318236644508943e-11
3	209408.0	301568.0	4.0790348876384996e-10
4	3.106816e6	2.844672e6	1.626246826091915e-8
5	2.4114688e7	2.3346688e7	6.657697912970661e-7
6	1.20152064e8	1.1882496e8	1.0754175226779239e-5
7	4.80398336e8	4.78290944e8	0.00010200279300764947
8	1.682691072e9	1.67849728e9	0.0006441703922384079
9	4.465326592e9	4.457859584e9	0.002915294362052734
10	1.2707126784e10	1.2696907264e10	0.009586957518274986
11	3.5759895552e10	3.5743469056e10	0.025022932909317674
12	7.216771584e10	7.2146650624e10	0.04671674615314281
13	2.15723629056e11	2.15696330752e11	0.07431403244734014
14	3.65383250944e11	3.653447936e11	0.08524440819787316
15	6.13987753472e11	6.13938415616e11	0.07549379969947623
16	1.555027751936e12	1.554961097216e12	0.05371328339202819
17	3.777623778304e12	3.777532946944e12	0.025427146237412046
18	7.199554861056e12	7.1994474752e12	0.009078647283519814
19	1.0278376162816e13	1.0278235656704e13	0.0019098182994383706
20	2.7462952745472e13	2.7462788907008e13	0.0001907087633625792 5

Można zauważyć, że dla każdego wyznaczonego miejsca zerowego, nie udało się otrzymać wartości 0.

Podpunkt B			
k	$\|P(z_k)\|$	$\|p(z_k)\|$	$\|z_k - k\|$
1	20992.0	22016.0	1.6431300764452317e-13
2	349184.0	365568.0	5.503730804434781e-11
3	2.221568e6	2.295296e6	3.3965799062229962e-9
4	1.046784e7	1.0729984e7	8.972436216225788e-8
5	3.9463936e7	4.3303936e7	1.4261120897529622e-6
6	1.29148416e8	2.06120448e8	2.0476673030955794e-5
7	3.88123136e8	1.757670912e9	0.00039792957757978087
8	1.072547328e9	1.8525486592e10	0.007772029099445632
9	3.065575424e9	1.37174317056e11	0.0841836320674414
10	7.143113638035824e9	1.4912633816754019e12	0.6519586830380406
11	7.143113638035824e9	1.4912633816754019e12	1.1109180272716561
12	3.357756113171857e10	3.2960214141301664e13	1.665281290598479
13	3.357756113171857e10	3.2960214141301664e13	2.045820276678428
14	1.0612064533081976e11	9.545941595183662e14	2.5188358711909045
15	1.0612064533081976e11	9.545941595183662e14	2.7128805312847097
16	3.315103475981763e11	2.7420894016764064e16	2.9060018735375106
17	3.315103475981763e11	2.7420894016764064e16	2.825483521349608
18	9.539424609817828e12	4.2525024879934694e17	2.454021446312976
19	9.539424609817828e12	4.2525024879934694e17	2.004329444309949
20	1.114453504512e13	1.3743733197249713e18	0.8469102151947894

Dla podpunktu B, po zaburzeniu wartości, otrzymaliśmy skrajnie inne wyniki względem A.

Wnioski

Nie udało się nam otrzymać wartości zerowych, co spowodowane jest tym, że w arytmetyce brakuje precyzji, a dokładniej odpowiedniej ilości cyfr znaczących. Ponadto na podstawie podpunktu B, możemy wywnioskować, że zadanie jest źle uwarunkowane, ponieważ niewielka zmiana wartości, wpłynęła znacznie na wyniki końcowe.

Zadanie 5

Problem

Problem polega na testowaniu równania rekurencyjnego danego wzorem: $p_{n+1} := p_n + rp_n(1 - p_n)$, w 40 iteracjach, na

- Float 32
- Float 32, gdzie co 10 iteracji ucianamy do 3 miejsca po przecinku
- Float 64

Gdzie r jest ustalone i wynosi 3.

Rozwiązanie

Rozwiązanie polega na implementacji funkcji rekurencyjnej symulującej równanie rekurencyjne. W przypadku z ucinaniem części ułamkowej, co 10 iteracji, przerywamy funkcję, aby następnie ją wznowić po wykonaniu operacji trunc.

Wyniki

Poniższa tabela przedstawia wartości kolejnych iteracji funkcji rekurencyjnych. Gdzie n+1 oznacza aktualny wyraz, a FLOAT32+TRUNC, to wyniki rekurencji z obcinaniem części ułamkowej.

Tabela kolejnych wartości rekurencji			
n+1	FLOAT32	FLOAT32+TRUNC	FLOAT64
1	0.01	0.01	0.01
2	0.0397	0.0397	0.0397
3	0.15407173	0.15407173	0.15407173000000002
4	0.5450726	0.5450726	0.5450726260444213
5	1.2889781	1.2889781	1.2889780011888006
6	0.1715188	0.1715188	0.17151914210917552
7	0.5978191	0.5978191	0.5978201201070994
8	1.3191134	1.3191134	1.3191137924137974
9	0.056273222	0.056273222	0.056271577646256565
10	0.21559286	0.21559286	0.21558683923263022
11	0.7229306	0.722	0.722914301179573
12	1.3238364	1.3241479	1.3238419441684408
13	0.037716985	0.036488414	0.03769529725473175
14	0.14660022	0.14195944	0.14651838271355924
15	0.521926	0.50738037	0.521670621435246
16	1.2704837	1.2572169	1.2702617739350768
17	0.2395482	0.28708452	0.24035217277824272
18	0.7860428	0.9010855	0.7881011902353041
19	1.2905813	1.1684768	1.2890943027903075
20	0.16552472	0.577893	0.17108484670194324
21	0.5799036	1.309	0.5965293124946907
22	1.3107498	0.095556974	1.3185755879825978
23	0.088804245	0.3548345	0.058377608259430724
24	0.3315584	1.0416154	0.22328659759944824
25	0.9964407	0.91157377	0.7435756763951792
26	1.0070806	1.1533948	1.315588346001072
27	0.9856885	0.62262046	0.07003529560277899
28	1.0280086	1.3275132	0.26542635452061003
29	0.9416294	0.023178816	0.8503519690601384
30	1.1065198	0.091103494	1.2321124623871897
31	0.7529209	0.339	0.37414648963928676
32	1.3110139	1.0112369	1.0766291714289444
33	0.0877831	0.9771474	0.8291255674004515
34	0.3280148	1.0441384	1.2541546500504441
35	0.9892781	0.90587854	0.29790694147232066
36	1.021099	1.1616664	0.9253821285571046
37	0.95646656	0.59825915	1.1325322626697856
38	1.0813814	1.3192946	0.6822410727153098
39	0.81736827	0.05556369	1.3326056469620293
40	1.2652004	0.21299279	0.0029091569028512065
41	0.25860548	0.71587336	0.011611238029748606

Niedokładności na pierwszej pozycji po ułamku, dla wszystkich 3 typów, występują już dla 17 wyrazu. A do czasu ostatniej iteracji, wyniki są już całkowicie inne.

Wniosek

Wartość uwarunkowania równań rekurencyjnych rośnie proporcjonalnie wraz z ilością iteracji. Jak widać na naszym przykładzie, dla 40 iteracji uwarunkowanie jest beznadziejne.

Zadanie 6

Problem

Problem polega na testowaniu równania rekurencyjnego danego wzorem: $x_{n+1} := x_n^2 + c$, w 40 iteracjach, dla ustalonych danych.

Rozwiązanie

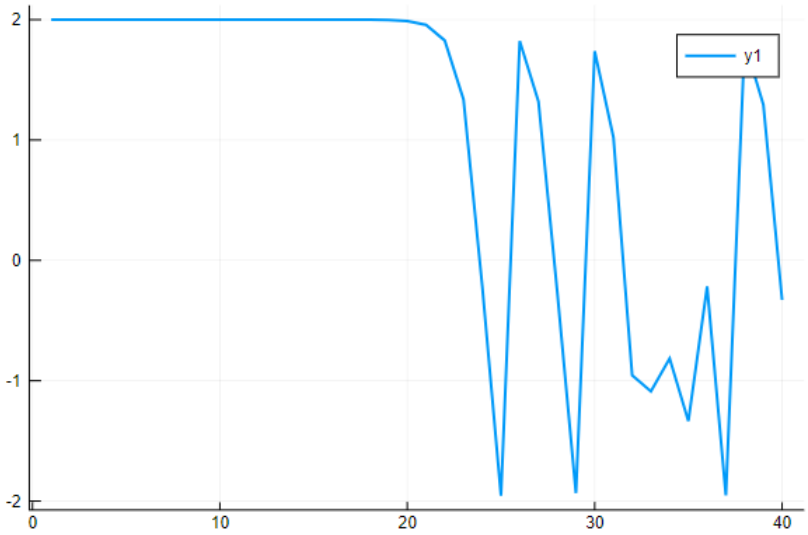
Rozwiązanie polega na implementacji funkcji rekurencyjnej symulującej równanie rekurencyjne.

Wyniki

Otrzymaliśmy kolejno dla ustalonych danych:

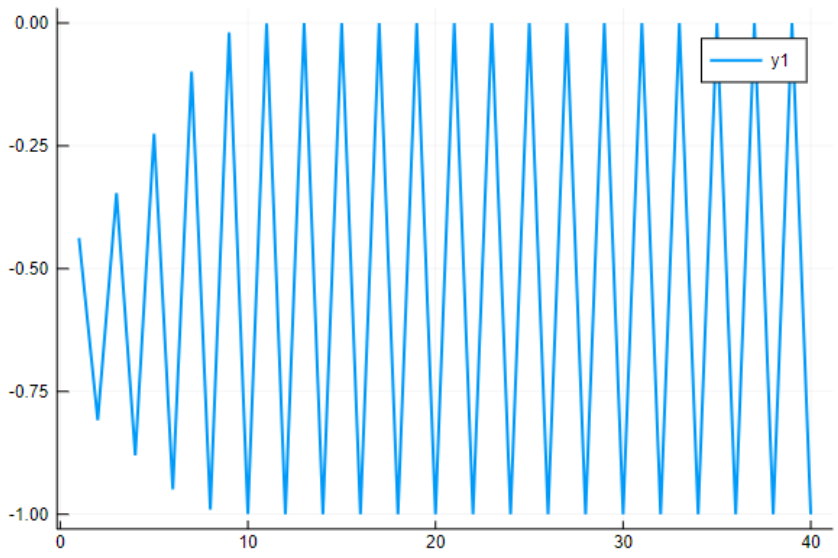
1. $c = -2$ i $x_0 = 1$ wynik: -1
2. $c = -2$ i $x_0 = 2$ wynik: 2
3. $c = -2$ i $x_0 = 1.99999999999999$ wynik: -0.3289791230026702
4. $c = -1$ i $x_0 = 1$ wynik: -1
5. $c = -1$ i $x_0 = -1$ wynik: -1
6. $c = -1$ i $x_0 = 0.75$ wynik: -1.0
7. $c = -1$ i $x_0 = 0.25$ wynik: 0.0

Przypadki całkowite są trywialne. Ale dla przykładów na liczbach zmiennoprzecinkowych mamy poniższe iteracje graficzne:



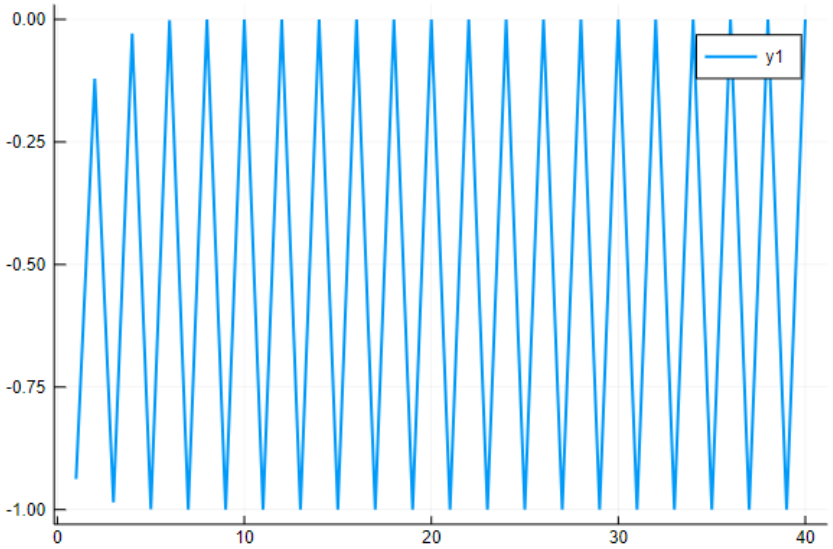
Rysunek 2 – $c = -2$ i $x_0 = 1.99999999999999$

Patrząc na rysunek 2, wartości funkcji zaczynają znacznie odchodzić od prawdziwych wyników, już po około 20 iteracjach.



Rysunek 3 – $c = -1$ i $x_0 = 0.75$

Dla rysunku 3, wartości dążą do liczb całkowitych i po 16 iteracjach je osiągają.



Rysunek 4 – $c = -1$ i $x_0 = 0.25$

Dla rysunku 4, wartości dążą do liczb całkowitych i po 11 iteracjach je osiągają.

Wniosek

Zadanie znowu dowodzi o błędach równań rekurencyjnych zaimplementowanych na liczbach zmiennoprzecinkowych. Widać na przykładzie z 1.9999999999999, że niewielkie zaburzenie danych spowodowało znaczną zmianę wyników, przez co można stwierdzić, że zadanie jest źle uwarunkowane.