Obliczenia naukowe

Felix Zieliński 272336

Lista 2

TODO OPIS, czy zadanie 2? zadanie 3

Zadanie 1. Niewielkie zmiany danych oraz ich wpływ na wyniki obliczeń.

W ramach przypomnienia zadania: na poprzedniej liście należało obliczyć iloczyny skalarne dwóch wektorów na cztery rózne sposoby.

Zaimplementowałem każdy z podanych w poleceniu sposobów, tak więc funkcja `a` liczy "w przód", od pierwszych indeksów, funkcja `b` "w tył", analogicznie, a `c` oraz `d` liczą, odpowiednio, od największego do najmniejszego oraz od najmniejszego do największego względem ich wartości absolutnej.

Różnica w tym zadaniu, a zadaniu 5. z poprzedniej listy polegała na dokonaniu drobnej zmiany w niektórych wartościach wektora. Poniżej prezentuję wyniki otrzymane po, jak i przed tej zmianie:

Sposób	Float32 stare	Float32 nowe	Float64 stare	Float64 nowe
a	-0.4999443	-0.4999443	1.0251881368296672e-10	-0.004296342739891585
b	-0.4543457	-0.4543457	-1.5643308870494366e-10	-0.004296342998713953
c	-0.5	-0.5	0.0	-0.004296342842280865
d	-0.5	-0.5	0.0	-0.004296342842280865

Tabela 1: Porównanie nowych i starych danych

gdzie wartość prawidłowa wynosi:

`-1.00657107000000e-11`

Jak widać, wyniki dla typu `Float32` nie zmieniły się. Jest to spowodowane niewystarczającą do zauważenia różnicy precyzją zapisu liczby zmiennopozycyjnej w tym typie.

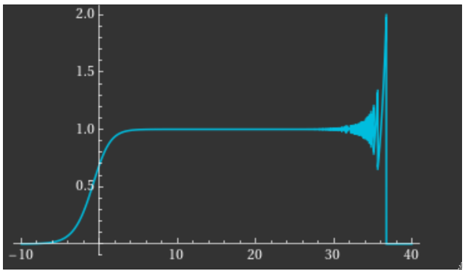
Natomiast w typie `Float64` różnica jest znaczna mimo tak niewielkiej zmiany danych. Mimo że wyniki nadal odbiegają od prawidłowego, są one mu znacznie bliższe.

Można więc stwierdzić, że zadanie to było **źle uwarunkowane** - o wysokim wskaźniku uwarunkowania. Wskaźnik ten określa, w jakim stopniu błąd reprezentacji numerycznej danych wejściowych dla danego problemu będzie wpływać na błąd wyniku. Małe zmany danych w tym zadaniu spowodowały znaczną zmianę wyników.

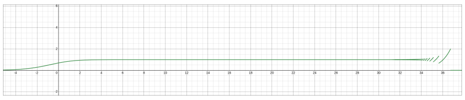
Zadanie 2. W tym zadaniu należało narysować wykres funkcji

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

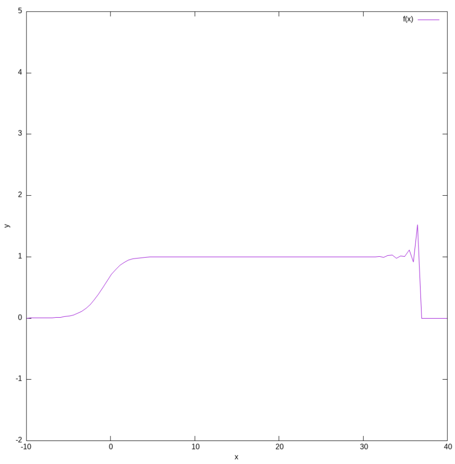
w dwóch różnych programach do wizualizacji danych. Zdecydowałem się na użycie WolframaAlpha, Desmosa oraz Gnuplota



Rysunek 1: WolframAlpha: `plot` $e^x * \ln(1 + e^{-x})$ `from` $x = -10$ `to` $x = 40$



Rysunek 2: Desmos



Rysunek 3: Gnuplot

Granica tej funkcji dla x zmierzającego do nieskończoności wynosi

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

Jak można zauważyć, dla wartości `x >= 32` wykresy zaczynają wskazywać błędne wartości, każdy na trochę inny sposób. Oscylują one wokół 1, coraz bardziej odbiegając od jej wartości, a następnie spadają do 0.

Dzieje się tak, gdyż dla `x > 30` wartości e^x są już tak duże, że mnożenie jej z niewielką wartością $ln(1 + e^{-x})$ skutkuje znacznymi błędami przybliżenia, które dla ok `x = 38` powodują zwracanie wartości 0. Jest to spowodowane przybliżeniem $1 + e^{-x} \approx 1$, i tym samym całej wartości logarytmu do 0. Tak więc algorytm obliczający wartości `f(x)` nie jest stabilny numerycznie.

Zadanie 3. W zadaniu tym należało rozwiązać liniowy układ równań `A * x = b` dla danej macierzy współczynników $A \in \mathbb{R}^{n \times n}$ i wektora prawych stron $b \in \mathbb{R}^n$. Macierze były generowane poprzez dostarczone przez prowadzącego funkcje: generującą macierz Hilberta `n`-tego stopnia oraz generującą losową macierz `n`-tego stopnia z zadanym wskaźnikiem uwarunkowania.

W tabelach przedstawiłem błędy względne dla zadanych algorytmów: eliminacji Gaussa oraz inwersji. Uwarunkowanie oraz rząd macierzy są obliczanie poprzez wbudowane w paczkę LinearAlgebra funkcje `cond()` oraz `rank()`.

n	uwarunkowanie	rząd	błąd metody Gaussa	błąd metody inwersji
1	1.0	1	0.0	0.0
2	19.281470067903967	2	5.661048867003676e-16	1.1240151438116956e-15
3	524.0567775860627	3	8.351061872731819e-15	9.825526038180824e-15
4	15513.738738929662	4	4.2267316576255873e-13	3.9600008750140806e-13
5	476607.2502419338	5	1.256825919192874e-12	8.128168770215688e-12
6	1.495105864177819e7	6	1.5435074657413347e-10	1.0423794065751672e-10
7	4.753673568766496e8	7	6.520804933066021e-9	4.3299229851434615e-9
8	1.5257575563722723e10	8	3.6010489197068436e-7	4.0236799996435915e-7
9	4.9315332284138226e11	9	1.3216991540025553e-5	1.4626798972086921e-5
10	1.6024980732174455e13	10	0.0004194170177181955	0.00040714905218460087
11	5.224780779168285e14	10	0.01004906783345069	0.010645959401385671
12	1.6425917529444498e16	11	0.5502106922296848	0.6697890564301745
13	4.4936679531246986e18	11	70.1556197115221	82.66675811171989
14	3.2198422552156205e17	11	9.649642437452474	10.094732062453225
15	3.3660126672602944e17	12	692.4295360390742	715.740988667373
16	2.249940193352714e18	12	10.414656083840297	8.442143351389534
17	6.26204622473199e17	12	18.67581817300634	17.157982115668773
18	3.266632306940269e18	12	5.40548300394664	3.742412802776696
19	3.462302955915255e18	13	15.073941146224387	16.84769281513296
20	6.806966421072721e18	13	28.79267493699834	30.751202239608727

Tabela 2: Macierze Hilberta

Im wyższe uwarunkowanie, tym wyższy błąd względny dla obu metod.

n	c	uwarunkowanie	rząd	błąd metody Gaussa	błąd metody inwersji
5	10^0	1.0000000000000007	5	2.0471501066083611e-16	1.7901808365247238e-16
5	10^1	10.00000000000001	5	2.579925170969555e-16	1.4895204919483638e-16
5	10^3	999.9999999999956	5	4.154180998732242e-14	3.6649738390350505e-14
5	10^7	9.999999992624711e6	5	1.2808136131610903e-10	1.2922561774440224e-10
5	10^{12}	1.0000402198324714e12	5	2.4114718692896424e-5	2.1083207553058387e-5
5	10^{16}	6.457380316295465e15	4	0.279866038431425	0.23511506127614903
10	10^0	1.0000000000000009	10	3.1006841635969763e-16	2.6506211417561425e-16
10	10^1	9.999999999999991	10	2.432376777795247e-16	3.255813018879823e-16
10	10^3	999.9999999999854	10	5.123291327463699e-15	4.80612456985904e-15
10	10^7	9.99999999300524e6	10	2.613887510795298e-10	2.9367939862640613e-10
10	10^{12}	9.999916908430352e11	10	2.5298084239916387e-5	2.7222350440303e-5
10	10^{16}	5.260556228219448e16	9	0.2349713562983511	0.17164964730117435
20	10^0	1.0000000000000009	20	5.450279209566124e-16	4.557326905135503e-16
20	10^1	9.99999999999999	20	5.318651993048588e-16	3.430930459816227e-16
20	10^3	1000.0000000000724	20	4.374127960791212e-14	3.7395403352225206e-14
20	10^7	1.0000000008586796e7	20	4.8148086460727405e-12	7.333934288678081e-11
20	10^{12}	1.000058179546536e12	20	5.063442743791289e-5	5.134232142086649e-5
20	10^{16}	9.35903000507404e15	19	0.13171344480426542	0.16281145033250885

Tabela 3: Macierze losowe

Tak jak przy macierzach Hilberta, w wynikach dla macierzy losowych można zaobserwować rosnący błąd względny. Ponadto widać, że funkcja cond() nie oblicza dokładnie uwarunkowania, tylko je przybliża. Ponadto wartości tych przybliżeń są różne dla różnych procesorów.

Powyższe tabele pozwalają na wyciągnięcie wniosku, że zadanie to było źle uwarunkowane, zwłaszcza dla macierzy Hilberta - wraz ze wzrostem stopnia macierzy ostro rośnie też wskaźnik uwarunkowania i, jednocześnie, błąd względny. Dla macierzy losowych dzieje się podobnie, jednakże ten wzrost jest wolniejszy (błędy są mniejsze).

Zadanie 4. W zadaniu tym należało obliczyć miejsca zerowe zadanego wielomianu `P(x)` i przedstawić wyniki dla z_k, $1 \le k \le 20$, obliczając $|P(z_k)|$, $|p(z_k)|$ i $|z_k - k|$. Następnie należało powtórzyć eksperyment Wilkinsona, czyli lekko zmodyfikować wielomian i przeprowadzić ponownie obliczenia.

Jak można zauważyć, obliczone miejsca zerowe przy pomocy `roots()` są zbliżone, ale nie równe wartościom prawidłowym.

k	z_k	$\lvert Pz_k \rvert$	$\lvert pz_k \rvert$	$\lvert z_k - k \rvert$
1	0.9999999999996989	35696.50964788257	368.50964789367345	3.0109248427834245e-13
2	2.0000000000283182	176252.60026668405	15996.60026439321	2.8318236644508943e-11
3	2.9999999995920965	279157.6968824087	100981.69690684375	4.0790348876384996e-10
4	3.9999999837375317	3.0271092988991085e6	252581.31017303217	1.626246826091915e-8
5	5.000000665769791	2.2917473756567076e7	152225.31504973918	6.657697912970661e-7
6	5.999989245824773	1.2902417284205095e8	4.441356137271629e6	1.0754175226779239e-5
7	7.000102002793008	4.805112754602064e8	4.510025884078405e7	0.00010200279300764947
8	7.999355829607762	1.6379520218961136e9	2.309311180467704e8	0.0006441703922384079
9	9.002915294362053	4.877071372550003e9	5.2519713947457147e8	0.002915294362052734
10	9.990413042481725	1.3638638195458128e10	1.4705597374927537e9	0.009586957518274986
11	11.025022932909318	3.585631295130865e10	2.257814747009822e9	0.025022932909317674
12	11.953283253846857	7.533332360358197e10	5.776283226748977e9	0.04671674615314281
13	13.07431403244734	1.9605988124330817e11	8.720084582957656e8	0.07431403244734014
14	13.914755591802127	3.5751347823104315e11	2.100530498133618e10	0.08524440819787316
15	15.075493799699476	8.21627123645597e11	9.0205643837176e10	0.07549379969947623
16	15.946286716607972	1.5514978880494067e12	1.1093925108150577e11	0.05371328339202819
17	17.025427146237412	3.694735918486229e12	5.420826950832621e11	0.025427146237412046
18	17.99092135271648	7.650109016515867e12	2.0817615965429592e12	0.009078647283519814
19	19.00190981829944	1.1435273749721195e13	4.420887520697798e12	0.0019098182994383706
20	19.999809291236637	2.7924106393680727e13	3.2780945625202856e12	0.00019070876336257925

Tabela 4: Wielomian P(x)