**Sprawozdanie z listy 2**

**Laboratorium, Obliczenia naukowe**

**Autor: Piotr Klepczyk**

1. **Zadanie 1**

**1.1. Cel**

Celem zadania jest zaimplementowanie czterech różnych sposobów obliczania sumy przy wyliczaniu iloczynu skalarnego. Metody które należy zaimplementować to sumowanie w przód, czyli w kolejności po współrzędnych, sumowanie w tył, czyli sumowanie w odwrotnej kolejności niż w poprzednim przykładzie. Kolejne dwa sposoby opierają się na posortowaniu wartości po przeprowadzeniu mnożenia. A następnie jaki wpływ na wyniki ma podane w treści zadania niewielkie zmiany wartości.

**1.2. Realizacja**

Wszystkie cztery sposoby obliczania sumy będą wykonywane dal arytmetyk Float32 i Float64. Pierwszym sposobem jest sumowanie w przód, czyli sumowanie współrzędnych po wymnożeniu po indeksach rosnąco. Drugi sposób to sumowanie w tył, czyli sumowanie po indeksach malejąco. Kolejnym sposobem jest sumowanie z wykorzystaniem sum częściowych. Jedna suma częściowa to suma wartości ujemnych liczona w kolejności od największej takiej liczby do najmniejszej, a następnie na sumowaniu sum częściowych. Ostatni sposób polega na tym samym co trzeci tylko sumy częściowe liczne są od najmniejszej do największej.

|  |  |  |
| --- | --- | --- |
| Sposób | Przed zmianą danych | Po zmianie danych |
| Float32 ‘w przód’ | -0.4999443 | -0.4999443 |
| Float32 ‘w tył’ | -0.4543457 | -0.4543457 |
| Float32 ‘malejąco’ | -0.5 | -0.5 |
| Float32 ‘rosnąco’ | -0.5 | -0.5 |
| Float64 ‘w przód’ | 1.0251881368296672e-10 | -0.004296342739891585 |
| Float64 ‘w tył’ | -1.5643308870494366e-10 | -0.004296342998713953 |
| Float64 ‘malejąco’ | 0.0 | -0.004296342842280865 |
| Float64 ‘rosnąco’ | 0.0 | -0.004296342842280865 |

Tab1.0: Tabela wyników algorytmów

Po porównaniu wyników widać że wyniki po zmianie danych dla arytmetyki Float64 są bliższe siebie niezależnie od przyjętej metody liczenia. Natomiast dla arytmetyki Float32 uzyskane wyniki są identyczne jak te uzyskane przed modyfikacją danych. Można z tego wywnioskować że we Float32 mimo zmniejszenia danych to ciągle dochodzi do tych samych zaokrągleń związanych z arytmetyką co wcześniej. Natomiast dla Float64 można zauważyć że nie dochodzi do zaokrągleń związanych z arytmetyką mających aż tak, jak miało to miejsce wcześniej, wpływ na wynik pracy algorytmu.

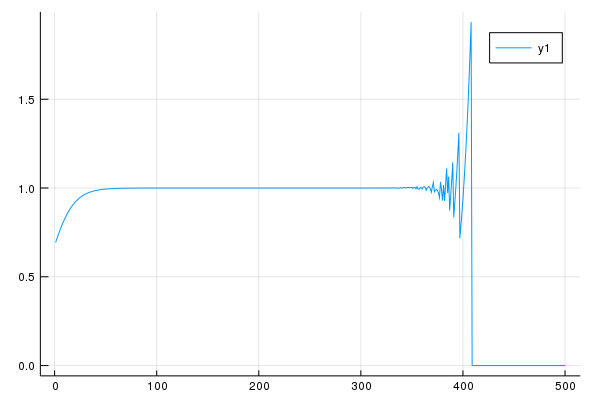
1. **Zadanie 2**

**2.1. Cel**

Celem zadania jest sporządzenie wykresu funkcji *f(x)=exln(1+e-x)* w przynajmniej dwóch programach do wizualizacji, a następnie policzyć granicę funkcji *f(x)* dla *x* dążącego do nieskończoności.

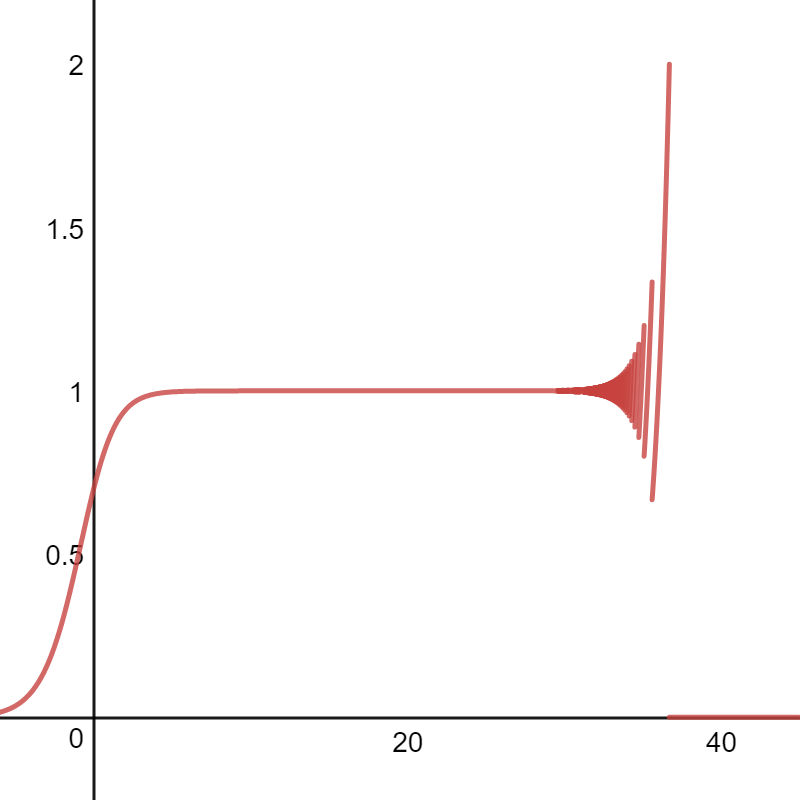
**2.2. Realizacja**

Do realizacji zadania użyto biblioteki Plots dla języka Julia oraz programu Microsoft Excel, a także strony internetowej do generowania wykresów online: www.desmos.com/calculator.



Wyk2.0: Wykres wygenerowany przez Julia Plots

Wyk2.1: Wykres wygenerowany przez program Microsoft Excel



Wyk2.2: Wykres wygenerowany przez serwis www.desmos.com/calculator

Na powyższych wykresach widać różnice przy ich konstrukcji oraz dokładności w zależności od użytego programu.

Obliczenie granicy: *limx->∞ exln(1+e-x)=1*

Widać niezgodność pomiędzy wykresami funkcji a jej granicą. Według granicy funkcja powinna dążyć do 1, lecz na wykresach widać że od wartości około 37 wartość funkcji wynosi 0. Jest to spowodowane arytmetyką, ponieważ podnosimy *e* do dużych potęg i w pewnym momencie wartości znaczące tej potęgi są tracone w skutek zaokrąglenia związanego z rozmiarem arytmetyki.

1. **Zadanie 3**

**3.1. Cel**

Celem zadania jest rozważenie dwóch sposobów rozwiązywania układów równań liniowych dla dwóch rodzajów macierzy, dla macierzy Hilberta oraz dla losowej macierzy. Następnie policzyć błędy względne i porównać uzyskane wartości. Dane należało wygenerować dla macierzy Hilberta dla n>1, gdzie n to wielkość macierzy. A dla macierzy losowej dla dł. 5,10 i 20 ze wskaźnikiem umiarkowania 1, 10, 103, 107, 1012, 1016.

**3.2. Realizacja**

Do generowania macierzy zostały wykorzystane funkcje udostępnione wraz z listą zadań. Zostały napisane dodatkowo funkcje liczące błąd względny, funkcja generująca wyniki dla macierzy Hilberta, najpierw zostały wygenerowane macierz oraz wektor x, o wszystkich wartościach równych 1. Następnie została obliczona wartość b będąca iloczynem macierzy oraz wektora x. Następnie wyliczamy wartość x przy użyciu dwóch metod, pierwsza to metoda Gaussa oraz mnożeniem b przez odwrotność. Wyniki dla macierzy losowej zostały obliczone w sposób analogiczny jak to ma miejsce dla macierzy Hilberta. Poniżej przedstawiono dane uzyskane podczas pracy algorytmu.

|  |  |  |  |
| --- | --- | --- | --- |
| Wielkość macierzy | Wskaźnik uwarunkowania | Błąd względny, metoda Gaussa | Błąd względny, drugiej metody |
| 2 | 19.28147006790397 | 5.661048867003676e-16 | 1.1240151438116956e-15 |
| 3 | 524.0567775860644 | 8.022593772267726e-15 | 9.825526038180824e-15 |
| 4 | 15513.73873892924 | 4.4515459601812086e-13 | 2.950477637286781e-13 |
| 5 | 476607.25024259434 | 1.6828426299227195e-12 | 8.500055777753297e-12 |
| 6 | 1.4951058642254665e7 | 2.618913302311624e-10 | 3.3474135070361745e-10 |
| 7 | 4.75367356583129e8 | 1.2606867224171548e-8 | 5.163959183577243e-9 |
| 8 | 1.5257575538060041e10 | 1.026543065687064e-7 | 2.698715074276819e-7 |
| 9 | 4.931537564468762e11 | 4.83235712050215e-6 | 9.175846868614517e-6 |
| 10 | 1.6024416992541715e13 | 0.0006329153722983848 | 0.00045521422517408853 |
| 11 | 5.222677939280335e14 | 0.011543958596122112 | 0.00804446677343116 |
| 12 | 1.7514731907091464e16 | 0.2975640310734787 | 0.34392937091205217 |
| 13 | 3.344143497338461e18 | 2.375017867706776 | 5.585796893150773 |
| 14 | 6.200786263161444e17 | 5.281004646755168 | 4.800641929017436 |
| 15 | 3.674392953467974e17 | 1.177294734836712 | 4.8273577212576475 |
| 16 | 7.865467778431645e17 | 20.564655823804095 | 31.736467496266126 |
| 17 | 1.263684342666052e18 | 17.742214635179074 | 15.910335962604142 |
| 18 | 2.2446309929189128e18 | 4.2764564411159425 | 6.281223433472033 |
| 19 | 6.471953976541591e18 | 22.119937292648906 | 22.92561401563632 |
| 20 | 1.3553657908688225e18 | 14.930069669294001 | 21.53949860251383 |

Tab3.0: Tablica wyników dla macierzy Hilberta

|  |  |  |  |
| --- | --- | --- | --- |
| Wielkość macierzy | Wskaźnik uwarunkowania | Błąd względny, metoda Gaussa | Błąd względny, drugiej metody |
| 5 | 1.0 | 9.930136612989092e-17 | 1.1102230246251565e-16 |
| 5 | 10.0 | 4.839349969133126e-16 | 4.0943002132167223e-16 |
| 5 | 1000.0 | 4.42616833736712e-14 | 4.211071688205494e-14 |
| 5 | 1.0e7 | 5.498758979832288e-10 | 5.168565618273073e-10 |
| 5 | 1.0e12 | 1.766282632375195e-5 | 2.1200191905874188e-5 |
| 5 | 1.0e16 | 0.040736262996224896 | 0.08809471777956406 |
| 10 | 1.0 | 5.661048867003676e-16 | 3.14018491736755e-16 |
| 10 | 10.0 | 3.1006841635969763e-16 | 2.220446049250313e-16 |
| 10 | 1000.0 | 2.2549938791612368e-14 | 1.530280447843421e-14 |
| 10 | 1.0e7 | 5.893106993272604e-11 | 7.081867086070856e-11 |
| 10 | 1.0e12 | 2.164086343724321e-6 | 5.617067745101069e-6 |
| 10 | 1.0e16 | 0.03373992624965958 | 0.08125335507380423 |
| 20 | 1.0 | 5.478475358467051e-16 | 4.1317602818954126e-16 |
| 20 | 10.0 | 4.4200263976433584e-16 | 3.789423922623494e-16 |
| 20 | 1000.0 | 4.2786099824341434e-15 | 2.6603686650593682e-15 |
| 20 | 1.0e7 | 2.765389386066196e-10 | 2.295849997731973e-10 |
| 20 | 1.0e12 | 3.622987363098445e-5 | 3.6087851099776866e-5 |
| 20 | 1.0e16 | 0.17075812951143313 | 0.16317655136097986 |

Tab3.1: Tabela wyników dla losowej macierzy

Dla macierzy Hilberta wartość błędu względnego rośnie dla na n z przedziału [2,14]. Dla n większych od 14 nie ma już jakiejś zależności pomiędzy kolejnymi wartościami błędu. Warto też zauważyć że metoda Gaussa daje mniejsze błędy względne. W danych dla macierzy losowej, błąd względny wzrasta wraz ze wzrostem rozmiaru macierzy oraz ze wzrostem wskaźnika uwarunkowania macierzy. W tym wypadku wartości uzyskane były dokładniejsze przy wykorzystaniu metody Gaussa.

1. **Zadanie 4**

**4.1. Cel**

Celem zadania było wyznaczenie pierwiastków dla naturalnej postaci wielomianu Wilkinsona, a następnie obliczyć następujące wartości: *|P(zk)|*, *|p(zk)|*, *|zk-k|* gdzie zk to pierwiastki wielomianu dla *k* z przedziału [1,20], a P to postać normalna wielomianu Wilkinsona, a p to postać iloczynowa tego samego wielomianu. Następnie należało powtórzyć eksperyment dla zmodyfikowanych danych: należało zastąpić -210 przez -210-2-23.

**4.2. Realizacja**

W zadaniu wykorzystano bibliotekę Polynomials dla języka Julia oraz dane podane wraz z treścią zadania w postaci tablicy współczynników wielomianu. Do stworzenia wielomianu w postaci normalnej została wykorzystana funkcja Poly(), a do stworzenia postaci iloczynowej wykorzystano funkcję poly().Do liczenia wartości wielomianu w danym punkcie wykorzystano funkcje polyval(), a do liczenia wartości bezwzględnej funkcję abs(). Funkcję roots() wykorzystano do wyznaczenia wszystkich pierwiastków wielomianu. Poniżej przedstawiono dane uzyskane przy realizacji algorytmu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *k* | *zk* | *|P(zk)|* | *|p(zk)|* | *|zk-k|* |
| 1 | 0.9999999999996989 | 36352.0 | 38400.0 | 3.0109248427834245e-13 |
| 2 | 2.0000000000283182 | 181760.0 | 198144.0 | 2.8318236644508943e-11 |
| 3 | 2.9999999995920965 | 209408.0 | 301568.0 | 4.0790348876384996e-10 |
| 4 | 3.9999999837375317 | 3.106816e6 | 2.844672e6 | 1.626246826091915e-8 |
| 5 | 5.000000665769791 | 2.4114688e7 | 2.3346688e7 | 6.657697912970661e-7 |
| 6 | 5.999989245824773 | 1.20152064e8 | 1.1882496e8 | 1.0754175226779239e-5 |
| 7 | 7.000102002793008 | 4.80398336e8 | 4.78290944e8 | 0.00010200279300764947 |
| 8 | 7.999355829607762 | 1.682691072e9 | 1.67849728e9 | 0.0006441703922384079 |
| 9 | 9.002915294362053 | 4.465326592e9 | 4.457859584e9 | 0.002915294362052734 |
| 10 | 9.990413042481725 | 1.2707126784e10 | 1.2696907264e10 | 0.009586957518274986 |
| 11 | 11.025022932909318 | 3.5759895552e10 | 3.5743469056e10 | 0.025022932909317674 |
| 12 | 11.953283253846857 | 7.216771584e10 | 7.2146650624e10 | 0.04671674615314281 |
| 13 | 13.07431403244734 | 2.15723629056e11 | 2.15696330752e11 | 0.07431403244734014 |
| 14 | 13.914755591802127 | 3.65383250944e11 | 3.653447936e11 | 0.08524440819787316 |
| 15 | 15.075493799699476 | 6.13987753472e11 | 6.13938415616e11 | 0.07549379969947623 |
| 16 | 15.946286716607972 | 1.555027751936e12 | 1.554961097216e12 | 0.05371328339202819 |
| 17 | 17.025427146237412 | 3.777623778304e12 | 3.777532946944e12 | 0.025427146237412046 |
| 18 | 17.99092135271648 | 7.199554861056e12 | 7.1994474752e12 | 0.009078647283519814 |
| 19 | 19.00190981829944 | 1.0278376162816e13 | 1.0278235656704e13 | 0.0019098182994383706 |
| 20 | 19.999809291236637 | 2.7462952745472e13 | 2.7462788907008e13 | 0.00019070876336257925 |

Tab4.0: Tabela wyników dla programu dla a)

|  |  |  |  |
| --- | --- | --- | --- |
| *k* | *zk* | *|P(zk)|* | *|p(zk)|* |
| 1 | 0.9999999999998357+0.0im | 20992.0 | 22016.0 |
| 2 | 2.0000000000550373+0.0im | 349184.0 | 365568.0 |
| 3 | 2.99999999660342+0.0im | 2.221568e6 | 2.295296e6 |
| 4 | 4.000000089724362+0.0im | 1.046784e7 | 1.0729984e7 |
| 5 | 4.99999857388791+0.0im | 3.9463936e7 | 4.3303936e7 |
| 6 | 6.000020476673031+0.0im | 1.29148416e8 | 2.06120448e8 |
| 7 | 6.99960207042242+0.0im | 3.88123136e8 | 1.757670912e9 |
| 8 | 8.007772029099446+0.0im | 1.072547328e9 | 1.8525486592e10 |
| 9 | 8.915816367932559+0.0im | 3.065575424e9 | 1.37174317056e11 |
| 10 | 10.095455630535774-0.6449328236240688im | 7.143113638035824e9 | 1.4912633816754019e12 |
| 11 | 10.095455630535774+0.6449328236240688im | 7.143113638035824e9 | 1.4912633816754019e12 |
| 12 | 11.793890586174369-1.6524771364075785im | 3.357756113171857e10 | 3.2960214141301664e13 |
| 13 | 11.793890586174369+1.6524771364075785im | 3.357756113171857e10 | 3.2960214141301664e13 |
| 14 | 13.992406684487216-2.5188244257108443im | 1.0612064533081976e11 | 9.545941595183662e14 |
| 15 | 13.992406684487216+2.5188244257108443im | 1.0612064533081976e11 | 9.545941595183662e14 |
| 16 | 16.73074487979267-2.812624896721978im | 3.315103475981763e11 | 2.7420894016764064e16 |
| 17 | 16.73074487979267+2.812624896721978im | 3.315103475981763e11 | 2.7420894016764064e16 |
| 18 | 19.5024423688181-1.940331978642903im | 9.539424609817828e12 | 4.2525024879934694e17 |
| 19 | 19.5024423688181+1.940331978642903im | 9.539424609817828e12 | 4.2525024879934694e17 |
| 20 | 20.84691021519479+0.0im | 1.114453504512e13 | 1.3743733197249713e18 |

Tab4.1: Tabela wyników dla programu dla b), cz1

|  |  |  |
| --- | --- | --- |
| *k* | *zk* | *|zk-k|* |
| 1 | 0.9999999999998357+0.0im | 1.6431300764452317e-13 |
| 2 | 2.0000000000550373+0.0im | 5.503730804434781e-11 |
| 3 | 2.99999999660342+0.0im | 3.3965799062229962e-9 |
| 4 | 4.000000089724362+0.0im | 8.972436216225788e-8 |
| 5 | 4.99999857388791+0.0im | 1.4261120897529622e-6 |
| 6 | 6.000020476673031+0.0im | 2.0476673030955794e-5 |
| 7 | 6.99960207042242+0.0im | 0.00039792957757978087 |
| 8 | 8.007772029099446+0.0im | 0.007772029099445632 |
| 9 | 8.915816367932559+0.0im | 0.0841836320674414 |
| 10 | 10.095455630535774-0.6449328236240688im | 0.6519586830380406 |
| 11 | 10.095455630535774+0.6449328236240688im | 1.1109180272716561 |
| 12 | 11.793890586174369-1.6524771364075785im | 1.665281290598479 |
| 13 | 11.793890586174369+1.6524771364075785im | 2.045820276678428 |
| 14 | 13.992406684487216-2.5188244257108443im | 2.5188358711909045 |
| 15 | 13.992406684487216+2.5188244257108443im | 2.7128805312847097 |
| 16 | 16.73074487979267-2.812624896721978im | 2.9060018735375106 |
| 17 | 16.73074487979267+2.812624896721978im | 2.825483521349608 |
| 18 | 19.5024423688181-1.940331978642903im | 2.454021446312976 |
| 19 | 19.5024423688181+1.940331978642903im | 2.004329444309949 |
| 20 | 20.84691021519479+0.0im | 0.8469102151947894 |

Tab4.2: Tabela wyników dla programu dla b), cz2

W podpunkcie a, uzyskane pierwiastki wielomianu mają niewielki błąd bezwzględny, jego wartość rośnie wraz z wartością pierwiastka. Następnie widzimy że wartości wielomianów dla uzyskanych pierwiastków są duże, a wiemy że powinny wynosić 0. Dzieje się tak ze względu na ograniczenia arytmetyki dla bardzo dużych liczb, ponieważ dla liczenia pierwiastków wartości wielomianu dochodzi do liczenia dużych potęg i następuje obcięcie wartości przez co nie uzyskujemy wyników dokładnych, tylko taki które mają bardzo duże błędy bezwzględne. Ale arytmetyka ma również wpływ na niedokładne wyznaczenie pierwiastków, gdyż wiemy że dla *zk* pierwiastek powinien być równy *k*. Jednak tak nie jest co widać po analizie wartości błędów bezwzględnych pierwiastków. Natomiast w podpunkcie b można zauważyć że wartości *|P(zk)|* są nieco mniejsze, a co za tym idzie dokładniejsze, mimo iż błąd bezwzględny w wartości pierwiastków urósł, urosły również wartości dla *|p(zk)|.* Może być to spowodowane właśnie błędem bezwzględnym pierwiastków. Gdyż dochodzi do zaokrągleń jak to miało miejsce w przypadku a. Różnica między wartościami pierwiastków wynika również z ograniczeń arytmetyki ponieważ zmodyfikowana dana znajduje się przy x19 więc jest mnożona przez liczby już poddane zaokrągleniu wynikającym z arytmetyki, a po modyfikacji sama ta liczba po modyfikacji ma więcej bitów znaczących w arytmetyce niż przed nią.

1. **Zadanie 5**

**5.1. Cel**

Celem zadania jest implementacja równania rekurencyjnego oraz przeprowadzić eksperymenty dla danych po=0.001, r=3. Należało wykonać 40 iteracji oraz wykonać 10 iteracji, przerwać działanie, obciąć uzyskane dane do trzeciego miejsca po przecinku, a następnie dokończyć do 40 iteracji i porównać uzyskane wyniki. Obliczenia należy wykonać w arytmetyce Float32. W drugiej części należało wywołać program dla danych z pierwszego uruchomienia ale dla arytmetyk Float32, Float64.

**5.2. Realizacja**

Do realizacji użyto programu w języku Julia, gdzie stworzono dwie funkcje rekurencyjne liczące równanie podane w treści zadania. Jedna zapewnia obliczeniom arytmetykę Float32, a druga Float64. Poniżej przedstawiono dane uzyskane przez program.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr iteracji | Bez przerwy Float32 | Z przerwą Float32 + obcięcie | Bez przerwy Float64 |
| 1 | 0.0397 | 0.0397 | 0.0397 |
| 2 | 0.15407173 | 0.15407173 | 0.15407173000000002 |
| 3 | 0.5450726 | 0.5450726 | 0.5450726260444213 |
| 4 | 1.2889781 | 1.2889781 | 1.2889780011888006 |
| 5 | 0.1715188 | 0.1715188 | 0.17151914210917552 |
| 6 | 0.5978191 | 0.5978191 | 0.5978201201070994 |
| 7 | 1.3191134 | 1.3191134 | 1.3191137924137974 |
| 8 | 0.056273222 | 0.056273222 | 0.056271577646256565 |
| 9 | 0.21559286 | 0.21559286 | 0.21558683923263022 |
| 10 | 0.7229306 | 0.7229306 | 0.722914301179573 |
| 11 | 1.3238364 | 1.3241479 | 1.3238419441684408 |
| 12 | 0.037716985 | 0.036488414 | 0.03769529725473175 |
| 13 | 0.14660022 | 0.14195944 | 0.14651838271355924 |
| 14 | 0.521926 | 0.50738037 | 0.521670621435246 |
| 15 | 1.2704837 | 1.2572169 | 1.2702617739350768 |
| 16 | 0.2395482 | 0.28708452 | 0.24035217277824272 |
| 17 | 0.7860428 | 0.9010855 | 0.7881011902353041 |
| 18 | 1.2905813 | 1.1684768 | 1.2890943027903075 |
| 19 | 0.16552472 | 0.577893 | 0.17108484670194324 |
| 20 | 0.5799036 | 1.3096911 | 0.5965293124946907 |
| 21 | 1.3107498 | 0.09289217 | 1.3185755879825978 |
| 22 | 0.088804245 | 0.34568182 | 0.058377608259430724 |
| 23 | 0.3315584 | 1.0242395 | 0.22328659759944824 |
| 24 | 0.9964407 | 0.94975823 | 0.7435756763951792 |
| 25 | 1.0070806 | 1.0929108 | 1.315588346001072 |
| 26 | 0.9856885 | 0.7882812 | 0.07003529560277899 |
| 27 | 1.0280086 | 1.2889631 | 0.26542635452061003 |
| 28 | 0.9416294 | 0.17157483 | 0.8503519690601384 |
| 29 | 1.1065198 | 0.59798557 | 1.2321124623871897 |
| 30 | 0.7529209 | 1.3191822 | 0.37414648963928676 |
| 31 | 1.3110139 | 0.05600393 | 1.0766291714289444 |
| 32 | 0.0877831 | 0.21460639 | 0.8291255674004515 |
| 33 | 0.3280148 | 0.7202578 | 1.2541546500504441 |
| 34 | 0.9892781 | 1.3247173 | 0.29790694147232066 |
| 35 | 1.021099 | 0.034241438 | 0.9253821285571046 |
| 36 | 0.95646656 | 0.13344833 | 1.1325322626697856 |
| 37 | 1.0813814 | 0.48036796 | 0.6822410727153098 |
| 38 | 0.81736827 | 1.2292118 | 1.3326056469620293 |
| 39 | 1.2652004 | 0.3839622 | 0.0029091569028512065 |
| 40 | 0.25860548 | 1.093568 | 0.011611238029748606 |

Tab5.0: Tablica wyników algorytmu

Wyniki z podpunktu pierwszego czyli analiza wpływu zatrzymania rekurencji, przeprowadzenie obcięcia i dalsza praca rekurencji wskazują że obcięcie ma znaczący wpływ na dalsze wyniki, można stwierdzić że różnice się pogłębiają. Co pokrywa się z intuicją gdyż każdy wynik ma wpływ na kolejny. Natomiast analiza danych dla dwóch arytmetyk Float32 i Float64 pokazuje wpływ dokładności arytmetyki na uzyskiwane dane. Dla początkowych iteracji wyniki są bardzo zbliżone, można powiedzieć że są po prostu dokładniejsze w Float64 ale różnice są niewielkie, natomiast później widać różnicę, można powiedzieć że daje o sobie znać dokładność arytmetyki, ponieważ jak dało się zauważyć w pierwszej części zadania zmiana nawet niewielka jednej z danych zmienia późniejsze wyniki i tak jest również w tym wypadku, gdyż różnice wynikają z innych zaokrągleń w arytmetykach.

1. **Zadanie 6**

**6.1. Cel**

Celem zadania było rozważenie równania rekurencyjnego *xn+1=xn2+c*, gdzie c to pewna stała. Należało przeprowadzić siedem eksperymentów w arytmetyce Float64 w języku Julia i zaobserwować zachowanie generowanych ciągów.

**6.2. Realizacja**

W implementacji równaniu rekurencyjnemu odpowiada funkcja rekurencyjna, która wykona zadaną w parametrze liczbę iteracji. Poniżej przedstawiono wyniki dla poszczególnych danych początkowych.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | eksperyment 1  *c=-2, x0=1* | eksperyment 2  *c=-2, x0=2* | eksperyment 3  *c=-2, x0=1.99999999999999* | eksperyment 4  *c=-1, x0=1* |
| 1 | -1.0 | 2.0 | 1.99999999999996 | 0.0 |
| 2 | -1.0 | 2.0 | 1.9999999999998401 | -1.0 |
| 3 | -1.0 | 2.0 | 1.9999999999993605 | 0.0 |
| 4 | -1.0 | 2.0 | 1.999999999997442 | -1.0 |
| 5 | -1.0 | 2.0 | 1.9999999999897682 | 0.0 |
| 6 | -1.0 | 2.0 | 1.9999999999590727 | -1.0 |
| 7 | -1.0 | 2.0 | 1.999999999836291 | 0.0 |
| 8 | -1.0 | 2.0 | 1.9999999993451638 | -1.0 |
| 9 | -1.0 | 2.0 | 1.9999999973806553 | 0.0 |
| 10 | -1.0 | 2.0 | 1.999999989522621 | -1.0 |
| 11 | -1.0 | 2.0 | 1.9999999580904841 | 0.0 |
| 12 | -1.0 | 2.0 | 1.9999998323619383 | -1.0 |
| 13 | -1.0 | 2.0 | 1.9999993294477814 | 0.0 |
| 14 | -1.0 | 2.0 | 1.9999973177915749 | -1.0 |
| 15 | -1.0 | 2.0 | 1.9999892711734937 | 0.0 |
| 16 | -1.0 | 2.0 | 1.9999570848090826 | -1.0 |
| 17 | -1.0 | 2.0 | 1.999828341078044 | 0.0 |
| 18 | -1.0 | 2.0 | 1.9993133937789613 | -1.0 |
| 19 | -1.0 | 2.0 | 1.9972540465439481 | 0.0 |
| 20 | -1.0 | 2.0 | 1.9890237264361752 | -1.0 |
| 21 | -1.0 | 2.0 | 1.9562153843260486 | 0.0 |
| 22 | -1.0 | 2.0 | 1.82677862987391 | -1.0 |
| 23 | -1.0 | 2.0 | 1.3371201625639997 | 0.0 |
| 24 | -1.0 | 2.0 | -0.21210967086482313 | -1.0 |
| 25 | -1.0 | 2.0 | -1.9550094875256163 | 0.0 |
| 26 | -1.0 | 2.0 | 1.822062096315173 | -1.0 |
| 27 | -1.0 | 2.0 | 1.319910282828443 | 0.0 |
| 28 | -1.0 | 2.0 | -0.2578368452837396 | -1.0 |
| 29 | -1.0 | 2.0 | -1.9335201612141288 | 0.0 |
| 30 | -1.0 | 2.0 | 1.7385002138215109 | -1.0 |
| 31 | -1.0 | 2.0 | 1.0223829934574389 | 0.0 |
| 32 | -1.0 | 2.0 | -0.9547330146890065 | -1.0 |
| 33 | -1.0 | 2.0 | -1.0884848706628412 | 0.0 |
| 34 | -1.0 | 2.0 | -0.8152006863380978 | -1.0 |
| 35 | -1.0 | 2.0 | -1.3354478409938944 | 0.0 |
| 36 | -1.0 | 2.0 | -0.21657906398474625 | -1.0 |
| 37 | -1.0 | 2.0 | -1.953093509043491 | 0.0 |
| 38 | -1.0 | 2.0 | 1.8145742550678174 | -1.0 |
| 39 | -1.0 | 2.0 | 1.2926797271549244 | 0.0 |
| 40 | -1.0 | 2.0 | -0.3289791230026702 | -1.0 |

Tab6.0: Tabela wyników cz1

|  |  |  |  |
| --- | --- | --- | --- |
| Nr | eksperyment 5  *c=-1, x0=-1* | eksperyment 6  *c=-1, x0=0.75* | eksperyment 7  *c=-1, x0=0.25* |
| 1 | 0.0 | -0.4375 | -0.9375 |
| 2 | -1.0 | -0.80859375 | -0.12109375 |
| 3 | 0.0 | -0.3461761474609375 | -0.9853363037109375 |
| 4 | -1.0 | -0.8801620749291033 | -0.029112368589267135 |
| 5 | 0.0 | -0.2253147218564956 | -0.9991524699951226 |
| 6 | -1.0 | -0.9492332761147301 | -0.0016943417026455965 |
| 7 | 0.0 | -0.0989561875164966 | -0.9999971292061947 |
| 8 | -1.0 | -0.9902076729521999 | -5.741579369278327e-6 |
| 9 | 0.0 | -0.01948876442658909 | -0.9999999999670343 |
| 10 | -1.0 | -0.999620188061125 | -6.593148249578462e-11 |
| 11 | 0.0 | -0.0007594796206411569 | -1.0 |
| 12 | -1.0 | -0.9999994231907058 | 0.0 |
| 13 | 0.0 | -1.1536182557003727e-6 | -1.0 |
| 14 | -1.0 | -0.9999999999986692 | 0.0 |
| 15 | 0.0 | -2.6616486792363503e-12 | -1.0 |
| 16 | -1.0 | -1.0 | 0.0 |
| 17 | 0.0 | 0.0 | -1.0 |
| 18 | -1.0 | -1.0 | 0.0 |
| 19 | 0.0 | 0.0 | -1.0 |
| 20 | -1.0 | -1.0 | 0.0 |
| 21 | 0.0 | 0.0 | -1.0 |
| 22 | -1.0 | -1.0 | 0.0 |
| 23 | 0.0 | 0.0 | -1.0 |
| 24 | -1.0 | -1.0 | 0.0 |
| 25 | 0.0 | 0.0 | -1.0 |
| 26 | -1.0 | -1.0 | 0.0 |
| 27 | 0.0 | 0.0 | -1.0 |
| 28 | -1.0 | -1.0 | 0.0 |
| 29 | 0.0 | 0.0 | -1.0 |
| 30 | -1.0 | -1.0 | 0.0 |
| 31 | 0.0 | 0.0 | -1.0 |
| 32 | -1.0 | -1.0 | 0.0 |
| 33 | 0.0 | 0.0 | -1.0 |
| 34 | -1.0 | -1.0 | 0.0 |
| 35 | 0.0 | 0.0 | -1.0 |
| 36 | -1.0 | -1.0 | 0.0 |
| 37 | 0.0 | 0.0 | -1.0 |
| 38 | -1.0 | -1.0 | 0.0 |
| 39 | 0.0 | 0.0 | -1.0 |
| 40 | -1.0 | -1.0 | 0.0 |

Tab6.1: Tabela wyników cz2

Jak można zauważyć dla eksperymentów 1,2,4,5 uzyskane dane są dokładne, natomiast w eksperymencie 6 i 7 można zauważyć że arytmetyka okazała się niewystarczająca do obliczenia wartości dokładnych i nastąpiło w obu przypadkach zaokrąglenie pewnej potęgi wyniku poprzedniego do 0. Natomiast w eksperymencie 3 uzyskane wyniki wydają się być prawidłowe, na pewno doszło do zaokrągleń związanych z arytmetyką, co można wywnioskować po dokładności danych początkowych, lecz mimo to nie doszło do oczywistego i od razu zauważalnego zaburzenia wyniku.