**Arytmetyka Komputerowa**

**Krystian Madej, 28.02.2024**

**1. Treść zadania**

Dana jest zależność rekurencyjna **3xk-1 − 10xk + 3xk+1 = 0**. Wartości początkowe **x0 = 1**, **x1 = 1/3**. Wyznaczyć wartości **xk** i **xk+1** dla **k = 45**. Następnie korzystając z wyznaczonych wartości **xk** i **xk+1** obliczyć **x1** i **x0**, wykonując rekurencję w tył. Porównać wyznaczone wartości **x1** i **x0** z dokładnymi wartościami początkowymi 1 i 1/3. Wykonać obliczenia dla różnej precyzji zmiennych (**float**, **double**, **long double**). Skomentować różnice. Co będzie, jeśli wszędzie liczbę 3 zastąpimy przez liczbę 2 lub 20, lub 30?

**2. Środowisko obliczeń**

**­**Obliczenia zostały wykonane przy pomocy języka **C++20** na systemie Ubuntu 22.04.3 (WSL2 na Windows 11), procesor **64-bitowy** Intel Core i5-11400H 2.70GHz, kod kompilowany kompilatorem **g++** (wersja 13.1).

**3. Użyte biblioteki i programy pomocnicze**

W rozwiązaniu użyto następnujących plików nagłówkowych (wszystkie należą do biblioteki standardowej):

* <vector> - tablica dynamiczna
* <format> - łatwe formatowanie
* <cfloat> - predefiniowane stałe opisujące typy zmiennoprzecinkowe
* <fstream> - zapis wyników do plików
* <iostream> - operacje wyjścia

Do rysowania wykresów użyto programu Excel z pakietu MS Office. Do wyznaczania wzorów jawnych niektórych ciągów użyto usługi Wolfram Alpha.

**4.1. Typy zmiennoprzecinkowe w obliczeniach**

Obliczenia były wykonywane na następujących typach zmiennoprzecinkowych:

* float
  + zgodny ze standardem IEEE 754
  + rozmiar w pamięci: 4 bajty
  + Mantysa: 23 bity
  + Wykładnik: 8 bity
  + Znak: 1 bit
* double
  + zgodny ze standardem IEEE 754
  + rozmiar w pamięci: 8 bajtów
  + Mantysa: 52 bity
  + Wykładnik: 11 bity
  + Znak: 1 bit
* long double
  + zgodny ze standardem IEEE 754
  + rozmiar w pamięci: 16 bajtów (10 faktycznych)
  + Mantysa: 63 bity
  + Wykładnik: 15 bity
  + Znak: 1 bit

**4.2. Implementacja obliczeń**

Na początek przekształcono równanie **3xk-1 − 10xk + 3xk+1 = 0 (1)** do postaci **xk+1 = 10/3xk − xk-1** (2). Zależność rekurencyjną w tej postaci przepisano do języka C++. W celu zmniejszenia ilości powtarzanego kodu użyto funkcji i zmiennych szablonowych. Zainicjalizowano vector o rozmiarze k+2 (aby można było się odwołać do elementu o indeksie k+1), a jego dwie pierwsze wartości ustawiono na 1 i 1/3. Następnie obliczano resztę wartości formułą: **(F)10 / a<F> \* get\_x<F>(k - 1) - get\_x<F>(k - 2),** gdzie **F** jest typem szablonowym. W analogiczny sposób obliczano wartości **x0** i **x1**, tym razem z użyciem formuły **xk-1 = 10/3xk − xk+1** (3).  
  
Wyniki obliczeń są zapisywane w plikach xn.txt, x0.txt, x1.txt, xk.txt, xkp1.txt.

**4.3. Wyniki obliczeń**

Obliczenia rekurencyjne **xk** i **xk+1**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| xk | | | | xk+1 | | | |
| **a** | **float** | **double** | **long double** | **a** | **float** | **double** | **long double** |
| **2** | 2.6519E+29 | 2.6519E+29 | 2.6519E+29 | **2** | 1.2706E+30 | 1.2706E+30 | 1.2706E+30 |
| **3** | 1.7086E+13 | 25437.4825 | 3.13546993 | **3** | 5.1257E+13 | 76312.4476 | 9.40640978 |
| **4** | -5.864E+12 | -5.864E+12 | -5.864E+12 | **4** | -1.173E+13 | -1.173E+13 | -1.173E+13 |
| **5** | -35.000046 | -35 | -35 | **5** | -35.800049 | -35.8 | -35.8 |
| **6** | -0.793156 | -0.793154 | -0.793154 | **6** | -1.407807 | -1.4078064 | -1.4078064 |
| **7** | -0.685644 | -0.685643 | -0.685643 | **7** | 0.275806 | 0.27580728 | 0.27580728 |
| **8** | -1.18686 | -1.1868604 | -1.1868604 | **8** | -0.710378 | -0.7103787 | -0.7103787 |
| **9** | 0.875934 | 0.87593343 | 0.87593343 | **9** | -0.112066 | -0.1120672 | -0.1120672 |
| **10** | -1 | -1 | -1 | **10** | -0.1 | -0.1 | -0.1 |
| **11** | 0.983662 | 0.98366264 | 0.98366264 | **11** | 0.844541 | 0.8445403 | 0.8445403 |
| **12** | 0.147449 | 0.14744967 | 0.14744967 | **12** | -0.897486 | -0.8974855 | -0.8974855 |
| **13** | -1.03976 | -1.0397606 | -1.0397606 | **13** | -0.559814 | -0.5598135 | -0.5598135 |
| **14** | -0.435474 | -0.4354738 | -0.4354738 | **14** | 0.732523 | 0.7325236 | 0.7325236 |
| **15** | 0.662266 | 0.66226608 | 0.66226608 | **15** | 0.975829 | 0.97582924 | 0.97582924 |
| **16** | 1.029637 | 1.02963683 | 1.02963683 | **16** | 0.231085 | 0.23108506 | 0.23108506 |
| **17** | 0.603565 | 0.6035647 | 0.6035647 | **17** | -0.620027 | -0.6200277 | -0.6200277 |
| **18** | -0.130099 | -0.1301 | -0.1301 | **18** | -1.014199 | -1.0141993 | -1.0141993 |
| **19** | -0.732802 | -0.7328023 | -0.7328023 | **19** | -0.882233 | -0.8822325 | -0.8822325 |
| **20** | -1.006158 | -1.0061581 | -1.0061581 | **20** | -0.420116 | -0.4201161 | -0.4201161 |
| **21** | -0.949703 | -0.9497028 | -0.9497028 | **21** | 0.132749 | 0.13274885 | 0.13274885 |
| **22** | -0.661609 | -0.661609 | -0.661609 | **22** | 0.602155 | 0.60215563 | 0.60215563 |
| **23** | -0.26137 | -0.2613706 | -0.2613706 | **23** | 0.901252 | 0.90125161 | 0.90125161 |
| **24** | 0.150976 | 0.15097619 | 0.15097619 | **24** | 1.01256 | 1.01256015 | 1.01256015 |
| **25** | 0.508327 | 0.50832709 | 0.50832709 | **25** | 0.960467 | 0.96046657 | 0.96046657 |
| **26** | 0.775093 | 0.77509275 | 0.77509275 | **26** | 0.787911 | 0.78791105 | 0.78791105 |
| **27** | 0.939869 | 0.93986821 | 0.93986821 | **27** | 0.540922 | 0.54092207 | 0.54092207 |
| **28** | 1.007182 | 1.00718186 | 1.00718186 | **28** | 0.260182 | 0.26018157 | 0.26018157 |
| **29** | 0.990517 | 0.99051694 | 0.99051694 | **29** | -0.022457 | -0.0224569 | -0.0224569 |
| **30** | 0.907221 | 0.90722113 | 0.90722113 | **30** | -0.284471 | -0.284471 | -0.284471 |

Tabela 1.  
Wartości xk i xk+1 dla a ∈ [2; 30]

Jak widać, dla małych a żaden z typów zmiennoprzecinkowych nie poradził sobie z dokładnymi obliczeniami. Na przykład można wykazać, że wzór jawny formuły (1) to **3-n**. Wartości **x45**i **x46** obliczone tym wzorem to odpowiednio **3.384882\*10-22** i **1.128294\*10-22**, które znacząco różnią się od wyników w tabeli 1, które w dodatku są znacząco rózne w zależności od typu zmiennoprzecinkowego. Inaczej jest w przypadku  
a=4 i a=5. Wolfram Alpha wyznaczył wzory jawne w takich przypadkach na odpowiednio **(7\*2-n-1-2n-1)/3** i **1 – 4/5 \* n**. Używając takich wzorów jawnych otrzymujemy wartości  
**xk** ≈ **-5.864\*1012** i **-35**, oraz wartości  
**xk+1** ≈ **-1.173\*1013** i **-35.8**. Wartości te są bliskie wynikom z tabeli 1.  
W przypadku gdy a=20 Wolfram Alpha wyznaczył wzór jawny na **1/75 \* 2-2n-1 \* ((75 - 4i√(15))(1 - i√(15))n + (75 + 4i√(15))(1 + i√(15))n)**, a wartości tego wzoru dla n=45 i n=46 są bliskie odpowiednio **-1.00616** i **-0.420116**, które są bliskie wartościom z tabeli 1.  
W przypadku gdy a=30 Wolfram Alpha wyznaczył wzór jawny na **1/350 \* ((175 - 4i√(35))(1/6 \* (1 - i√(35)))n + (175 + 4i√ (35))(1/6 \* (1 + i√(35)))n)**, a wartości tego wzoru dla n=45 i n=46 są bliskie odpowiednio **0.907221** i **-0.284471**, które są bliskie wartościom z tabeli 1.

Obliczenia rekurencyjne **x0** i **x1**.

Tabela 2.  
Wartości x0 i x1 dla a ∈ [2; 30]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x0 | | | | | x1 | | | | |
| **a** | **float** | **double** | **long double** | **faktyczna** | **a** | **float** | **double** | **long double** | **faktyczna** |
| **2** | nan | -7.704E+43 | -1.735E+40 | 1 | **2** | nan | -1.608E+43 | -3.621E+39 | 0.5 |
| **3** | -8.214E+26 | -8.799E+09 | -351.51014 | 1 | **3** | -2.738E+26 | -2.933E+09 | -117.17005 | 0.333333 |
| **4** | 1.2298E+19 | 1.1453E+10 | 11184810.5 | 1 | **4** | 6.1489E+18 | 5726623061 | 5592405 | 0.25 |
| **5** | 1.000092 | 1 | 1 | 1 | **5** | 0.200089 | 0.2 | 0.2 | 0.2 |
| **6** | 1 | 1 | 1 | 1 | **6** | 0.166667 | 0.16666667 | 0.16666667 | 0.166667 |
| **7** | 1 | 1 | 1 | 1 | **7** | 0.142857 | 0.14285714 | 0.14285714 | 0.142857 |
| **8** | 1 | 1 | 1 | 1 | **8** | 0.125 | 0.125 | 0.125 | 0.125 |
| **9** | 1 | 1 | 1 | 1 | **9** | 0.111111 | 0.11111111 | 0.11111111 | 0.111111 |
| **10** | 1 | 1 | 1 | 1 | **10** | 0.1 | 0.1 | 0.1 | 0.1 |
| **11** | 1 | 1 | 1 | 1 | **11** | 0.090909 | 0.09090909 | 0.09090909 | 0.090909 |
| **12** | 1 | 1 | 1 | 1 | **12** | 0.083333 | 0.08333333 | 0.08333333 | 0.083333 |
| **13** | 1 | 1 | 1 | 1 | **13** | 0.076923 | 0.07692308 | 0.07692308 | 0.076923 |
| **14** | 1 | 1 | 1 | 1 | **14** | 0.071429 | 0.07142857 | 0.07142857 | 0.071429 |
| **15** | 1 | 1 | 1 | 1 | **15** | 0.066667 | 0.06666667 | 0.06666667 | 0.066667 |
| **16** | 1 | 1 | 1 | 1 | **16** | 0.0625 | 0.0625 | 0.0625 | 0.0625 |
| **17** | 1 | 1 | 1 | 1 | **17** | 0.058824 | 0.05882353 | 0.05882353 | 0.058824 |
| **18** | 1 | 1 | 1 | 1 | **18** | 0.055556 | 0.05555556 | 0.05555556 | 0.055556 |
| **19** | 1 | 1 | 1 | 1 | **19** | 0.052631 | 0.05263158 | 0.05263158 | 0.052632 |
| **20** | 1 | 1 | 1 | 1 | **20** | 0.05 | 0.05 | 0.05 | 0.05 |
| **21** | 1 | 1 | 1 | 1 | **21** | 0.047619 | 0.04761905 | 0.04761905 | 0.047619 |
| **22** | 1 | 1 | 1 | 1 | **22** | 0.045455 | 0.04545455 | 0.04545455 | 0.045455 |
| **23** | 1 | 1 | 1 | 1 | **23** | 0.043478 | 0.04347826 | 0.04347826 | 0.043478 |
| **24** | 1 | 1 | 1 | 1 | **24** | 0.041667 | 0.04166667 | 0.04166667 | 0.041667 |
| **25** | 1 | 1 | 1 | 1 | **25** | 0.04 | 0.04 | 0.04 | 0.04 |
| **26** | 1 | 1 | 1 | 1 | **26** | 0.038462 | 0.03846154 | 0.03846154 | 0.038462 |
| **27** | 1 | 1 | 1 | 1 | **27** | 0.037037 | 0.03703704 | 0.03703704 | 0.037037 |
| **28** | 1 | 1 | 1 | 1 | **28** | 0.035714 | 0.03571429 | 0.03571429 | 0.035714 |
| **29** | 1 | 1 | 1 | 1 | **29** | 0.034483 | 0.03448276 | 0.03448276 | 0.034483 |
| **30** | 1 | 1 | 1 | 1 | **30** | 0.033333 | 0.03333333 | 0.03333333 | 0.033333 |

W przypadku **x0** i **x1** znane są wartości oczekiwane obliczeń, równe odpowiednio **1** i **1/a**. Podobnie jak poprzednio, żaden typ zmiennoprzecinkowy nie poradził sobie z małymi wartościami **a**, a typ float dla **a=2** zwrócił wartość **NaN**. Dla **a=5** błąd był bardzo mały, a dla kolejnych **a** zanikł. Duży wkład w to miały w miarę dokładne wartości **xk** i **xk+1** obliczone wcześniej.

Wykres 4.  
Wartości x0 i x1 dla a ∈ [5; 30]

Wykres 3.  
Wartości x0 i x1 dla a ∈ [5; 30]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xn, dla a = 3 | | | | |
| **n** | **float** | **double** | **long double** | **faktyczna wartość** |
| **0** | 1 | 1 | 1 | 1 |
| **1** | 0.333333 | 0.33333333 | 0.33333333 | 0.33333333 |
| **2** | 0.111111 | 0.11111111 | 0.11111111 | 0.11111111 |
| **3** | 0.037037 | 0.03703704 | 0.03703704 | 0.03703704 |
| **4** | 0.012346 | 0.01234568 | 0.01234568 | 0.01234568 |
| **5** | 0.004117 | 0.00411523 | 0.00411523 | 0.00411523 |
| **6** | 0.001376 | 0.00137174 | 0.00137174 | 0.00137174 |
| **7** | 0.00047 | 0.00045725 | 0.00045725 | 0.00045725 |
| **8** | 0.00019 | 0.00015242 | 0.00015242 | 0.00015242 |
| **9** | 0.000165 | 5.0805E-05 | 5.0805E-05 | 5.0805E-05 |
| **10** | 0.000358 | 1.6935E-05 | 1.6935E-05 | 1.6935E-05 |
| **11** | 0.00103 | 5.645E-06 | 5.645E-06 | 5.645E-06 |
| **12** | 0.003075 | 1.8817E-06 | 1.8817E-06 | 1.8817E-06 |
| **13** | 0.009221 | 6.2724E-07 | 6.2723E-07 | 6.2723E-07 |
| **14** | 0.027662 | 2.0912E-07 | 2.0908E-07 | 2.0908E-07 |
| **15** | 0.082984 | 6.9815E-08 | 6.9692E-08 | 6.9692E-08 |
| **16** | 0.248952 | 2.3601E-08 | 2.3231E-08 | 2.3231E-08 |
| **17** | 0.746857 | 8.8555E-09 | 7.7437E-09 | 7.7435E-09 |
| **18** | 2.240572 | 5.917E-09 | 2.5816E-09 | 2.5812E-09 |
| **19** | 6.721715 | 1.0868E-08 | 8.6163E-10 | 8.6039E-10 |
| **20** | 20.165144 | 3.0309E-08 | 2.905E-10 | 2.868E-10 |
| **21** | 60.495434 | 9.0162E-08 | 1.067E-10 | 9.5599E-11 |
| **22** | 181.486298 | 2.7023E-07 | 6.5172E-11 | 3.1866E-11 |
| **23** | 544.458862 | 8.1061E-07 | 1.1054E-10 | 1.0622E-11 |
| **24** | 1633.37647 | 2.4318E-06 | 3.0329E-10 | 3.5407E-12 |
| **25** | 4900.12891 | 7.2954E-06 | 9.0042E-10 | 1.1802E-12 |
| **26** | 14700.3867 | 2.1886E-05 | 2.6981E-09 | 3.9341E-13 |
| **27** | 44101.1602 | 6.5659E-05 | 8.0933E-09 | 1.3114E-13 |
| **28** | 132303.469 | 0.00019698 | 2.428E-08 | 4.3712E-14 |
| **29** | 396910.406 | 0.00059093 | 7.2839E-08 | 1.4571E-14 |
| **30** | 1190731.13 | 0.00177278 | 2.1852E-07 | 4.8569E-15 |
| **31** | 3572193.25 | 0.00531835 | 6.5555E-07 | 1.619E-15 |
| **32** | 10716580 | 0.01595504 | 1.9666E-06 | 5.3966E-16 |
| **33** | 32149738 | 0.04786511 | 5.8999E-06 | 1.7989E-16 |
| **34** | 96449216 | 0.14359533 | 1.77E-05 | 5.9962E-17 |
| **35** | 289347648 | 0.430786 | 5.3099E-05 | 1.9987E-17 |
| **36** | 868042944 | 1.292358 | 0.0001593 | 6.6625E-18 |
| **37** | 2604128768 | 3.87707401 | 0.0004779 | 2.2208E-18 |
| **38** | 7812385792 | 11.631222 | 0.00143369 | 7.4027E-19 |
| **39** | 2.3437E+10 | 34.893666 | 0.00430106 | 2.4676E-19 |
| **40** | 7.0311E+10 | 104.680998 | 0.01290317 | 8.2253E-20 |
| **41** | 2.1093E+11 | 314.042994 | 0.03870951 | 2.7418E-20 |
| **42** | 6.328E+11 | 942.128983 | 0.11612852 | 9.1392E-21 |
| **43** | 1.8984E+12 | 2826.38695 | 0.34838555 | 3.0464E-21 |
| **44** | 5.6952E+12 | 8479.16085 | 1.04515664 | 1.0155E-21 |
| **45** | 1.7086E+13 | 25437.4825 | 3.13546993 | 3.3849E-22 |
| **46** | 5.1257E+13 | 76312.4476 | 9.40640978 | 1.1283E-22 |

Obliczenia **xn** dla **a=3**.

Tabela 3.  
Wartości xn dla a = 3

Wykres 5.  
Wartości xn dla a = 3

Na tabeli 3 i wykresie 5 wyraźnie widać, kiedy który typ zmiennoprzecinkowy zaczyna odchodzić od faktycznej wartości. Dla float jest to n=7, dla double jest to n=17, a dla long double jest to n=21.

**5. Wnioski**

Jak pokazały powyższe przykłady, komputery mają ograniczone możliwości prowadzenia operacji zmiennoprzecinkowych. Mały błąd w jednej operacji przenosi się na kolejne działania, które też same z siebie mogą dawać błędne wyniki. W ten sposób błąd rośnie wykładniczo, co widać na tabeli 3 i wykresie 5. W przypadku takim, jak w tym zadaniu, błędy można zniwelować porzucając typy zmiennoprzecinkowe na rzecz specjalnych klas reprezentujących ułamki zwykłe.