Sprawozdanie z zadania P1.5

Sławomir Górawski

12 listopada 2017

1. Cel wykonanych doświadczeń

Liczby niewymierne można przybliżać przy użyciu skończonych ułamków łańcuchowych. Potrzebujemy do tego danych wejściowych, tj. w tym przypadku dwóch wektorów: $a = [a_1, a_2, ..., a_n]$ oraz $b = [b_0, b_1, ..., b_n]$, jak również odpowiedniego algorytmu, który wartość danego ułamka potrafi obliczyć. W zadaniu przedstawione zostały dwie metody: *"schodami w górę"* oraz *"schodami w dół"*. Oto używane w nich wzory, gdzie C_n to wynik działania algorytmu, oraz ich główne cechy:

1. *"Schodami w górę"*

$$
\begin{aligned}
U_n &= b_n, \\
U_k &= \frac{a_k + 1}{U_k + 1} + b_k, \qquad\qquad k = (n-1, n-2, \dots, 0) \\
C_n &= U_0
\end{aligned}
$$

   - Bardziej intuicyjna - tak samo liczyłby człowiek "na papierze"
   - Prostsza w implementacji
   - Zajmująca mniej pamięci - dane wejściowe,jedna zmienna i licznik pętli
   - Złożoność liniowa: stała liczba c_1 operacji poza pętlą, n powtórzeń pętli ze stałą liczbą c_2 operacji, zatem $T(n) = c_1 + nc_2 = O(n)$ (przy założeniu efektywnej, iteracyjnej implementacji)

2. *"Schodami w dół"*

$$
\begin{aligned}
&P_{-1} = 1, \qquad P_0 = b_0, \quad P_k = b_k P_{k-1} + a_k P_{k-2} \quad k = (1, 2, \dots, n); \\
&Q_{-1} = 0, \qquad Q_0 = 1, \quad Q_k = b_k Q_{k-1} + a_k Q_{k-2} \quad k = (1, 2, \dots, n). \\
&C_n = P_n / Q_n
\end{aligned}
$$

   - Możliwa do zdefiniowania funkcja kroku iteracji dla P i Q - wtedy, po zaprogramowaniu odpowiedniego algorytmu, po obliczeniu wartości ułamka łańcuchowego można szybko zwiększyć precyzję dokładając wartości a_{n+1} i b_{n+1} i otrzymując nowy wynik ze złożonością $O(1)$. W przypadku poprzedniej metody konieczne byłoby obliczenie wartości całego nowego ułamka od początku.

- Zajmująca nieco więcej pamięci - sześć zmiennych, tj. p_{k-1}, p_k, p_{k+1} oraz q_{k-1}, q_k, q_{k+1} plus licznik pętli
- Złożoność liniowa - rozumowanie analogiczne jak poprzednio

Z tej wstępnej analizy wynikałoby, że metody są bardzo podobne pod względem kosztów realizacji, z wyłączeniem szczególnego przypadku dynamicznego zwiększania wektórów a i b wspomnianego przy porównaniu. Ponadto oba algorytmy są numerycznie poprawne, co łatwo jest sprawdzić. Warto zatem przetestować w praktyce działanie obu algorytmów na ułamkach, które z założenia przybliżać mają znane wartości.

2. Opis użytych metod

W pliku `program.jl` zaimplementowane zostały oba algorytmy metodami iteracyjnymi gwarantującymi optymalną złożoność obliczeniową, w postaci funkcji dwuargumentowych przyjmujących wektory a i b i zwracających wynik w postaci liczby zmiennoprzecinkowej. Ponadto wydzielona została funkcja pojedynczego kroku iteracji dla metody *"schodami w dół"*, co pozwoliło przetestować w praktyce przewagę tej metody w przypadku dynamicznego wyliczania coraz bardziej dokładnych wartości.

Z kolei w dokumencie `program.ipynb` oba te algorytmy zostały przetestowane dla różnych ułamków łańcuchowych. Przyjęte zostało odpowiednio dokładne przybliżenie ich oczekiwanych wartości. Następnie wyliczane były wartości ułamków łańcuchowych przy użyciu obu metod dla kolejnych wartości k, co z założenia skutkować powinno dokładniejszemu przybliżeniu oczekiwanej wartości z każdą koleją iteracją. Wyniki w odniesieniu do dokładnej wartości przedstawione zostały na wykresach.

Testy dokładności przeprowadzane były na różnych precyzjach arytmetyki dla każdego ułamka. Do demonstracji na wykresach wybrane zostały określone przedziały wartości k, które pozwalają najlepiej zauważyć interesujące wyniki. Brane pod uwagę były wartości kolejnych przybliżeń w odniesieniu do aproksymowanej wartości oraz błędy bezwzględne pojawiające się w kolejnych iteracjach.

Oprócz dokładności mierzony był również czas wyliczania wartości ułamków dla kolejnych k trzema metodami: *"schodami w górę"*, *"schodami w dół"* oraz *"schodami w dół"* przy dynamicznym wyliczaniu kolejnych P i Q bez konieczności odtwarzania od nowa wcześniejszych przybliżeń.

3. Opis wykonanych doświadczeń

3.1 Przykład *b(i)* - przybliżanie wartości $\frac{\pi}{4}$

Wartości kolejnych wyrazów ciągów a i b:

$$a_1 = 1,\ a_2 = 1,\ a_3 = 9,\ a_4 = 25,\ a_5 = 49, \ldots,\ a_k = (2k-3)^2$$

$$b_0 = 0,\ b_1 = 1, \ldots,\ b_k = min(2, k)$$

Wykonane zostały pomiary dokładności dla co piątego k z przedziału $[100, 200]$, dla precyzji 64-, 32- i 16-bitowej. We wszystkich tych przypadkach zaobserwowano oczekiwane wyniki: wartości zbiegające do $\pi/4$ w kolejnych iteracjach oraz malejące błędy bezwgledne. Z kolei przybliżenie oczekiwanej wartości w przypadku tego ułamka nawet po 200 iteracjach jest mało dokładne w porównaniu do następnych przykładów.

W przypadku pomiarów czasu metody *"schodami w górę"* oraz *"schodami w dół"* wykazywały liniowy wzrost potrzebnego czasu wraz z kolejnymi iteracjami, ta druga około dwa razy większy od pierwszej, co nie dziwi, biorąc pod uwagę znacznie większą ilość obliczeń. Z kolei metoda *"schodami w dół"* poddana modyfikacji utrzymywała stałą liczbę potrzebnego czasu w miarę wzrostu wartości k. Potwierdza to przypuszczenie, że w przypadku, gdy chcemy wyliczać kolejne przybliżenia po kolei, jest ona znacznie bardziej efektywna. Innymi słowy, jeśli znamy wartości ułamka dla iteracji $1, 2, \ldots, k-1$, to wyliczenie następnej metodą *"schodami w górę"* będzie miało złożoność czasową $O(k)$, zaś odpowienio zaimplementowaną metodą *"schodami w dół"* - $O(1)$.

3.2 Przykład *b(ii)* - przybliżanie wartości e

Wartości kolejnych wyrazów ciągów a i b:

$$b_0 = 2,\ a_k = b_k = k + 1\ (k \geqslant 1)$$

W tym przypadku pomiary dokładności wykonywane były na precyzjach 128-, 64- i 32-bitowej. Obie metody do pewnego momentu przybliżały wartość e z niewielkimi błędami. W każdej precyzji jednak od pewnego momentu w wynikach otrzymanych metodą *"schodami w dół"* zaczęły pojawiać się błędy - tym wcześniej i tym większe, im mniejsza była precyzja arytmetyki. Nic podobnego nie zaobserwowano w przypadku metody *"schodami w górę"*.

Z pomiarów czasu wysnuto identyczne wnioski, jak w poprzednim przykładzie - liniowy wzrost potrzebnego czasu w metodach *"schodami w górę"* oraz *"schodami w dół"*, stały dla dynamicznej metody iteracyjnej.

3.3 Przykład dodatkowy - przybliżanie wartości $\sqrt{2}$

Wartości kolejnych wyrazów ciągów a i b:

$$b_0 = 1,\ a_k = 1,\ b_k = 2\ (k \geqslant 1)$$

Przetestowano dokładność dla precyzji: 128-, 64- i 32-bitowej. W pierwszym przypadku obie metody zbieżne do $\sqrt{2}$, w drugim i trzecim błędy w wyniku uzyskanym metodą *"schodami w dół"*, podobnie jak w przykładzie 3.2. Metoda *"schodami w górę"* wciąż nie wykazuje żadnych objawów niepoprawnego działania.

Wyniki pomiaru czasu po raz kolejny analogiczne jak w przykładzie 3.1.

3.4 Przykład dodatkowy - przybliżanie warości $\phi = \dfrac{1+\sqrt{5}}{2}$

Wartości kolejnych wyrazów ciągów a i b:

$$b_0 = 1,\ a_k = b_k = 1\ (k \geqslant 1)$$

Dla tego przykładu otrzymano wyniki podobne jak w 3.2 - obie metody zbiegają do ϕ, jednak od pewnego momentu w metodzie *"schodami w dół"* pojawiają się błędy - co ciekawe, zawsze po mniej więcej tej samej iteracji, niezależnie od precyzji (przetestowano na arytmetykach: 256-, 128- i 64-bitowej). Błędy te są radykalnie wręcz duże - rzędu wielkości 10^1, a nawet więcej. Przy spodziewanym przybliżaniu liczby niewymiernej mniejszej od 2 jest to nie do przyjęcia.

Pomiary czasu w tym przypadku dały ciekawe wyniki, inne niż dla dwóch poprzednich przykładów - metoda *"schodami w dół"*, nawet bez stosowania iteracyjnego ulepszenia, okazała się znacznie szybsza od metody *"schodami w górę"*. W obu poprzednich przykładach było odwrotnie. Najwyraźniej postać ułamka, który w tym przypadku jest bardzo prosty i oznacza tylko mnożenia przez 1 podczas wyliczania go *"schodami w dół"*, była w stanie zrobić dużą różnicę w czasie.

4. Wnioski

Zarówno patrząc teoretycznie, jak i analizując wyniki doświadczeń, łatwo jest dojść do wniosku, że metoda *"schodami w górę"* ma wiele zalet - jest prostsza w implementacji, bardziej intuicyjna i nie wykazuje podatności na błędy. Jedyna zaleta metody *"schodami w dół"*, jaką udało mi się znaleźć podczas pracy nad zadniem, to szybkość, w przypadku, gdy potrzebujemy nie tylko przybliżenia dla wybranego k, ale także wszystkich poprzednich.

Ilość zastosowań, w których byłoby to decydującą oklicznością przemawiającą za wyborem określonej metody, biorac pod uwagę moc obliczeniową obecnych komputerów oraz ryzyko pojawienia się błędów po którejś z kolei iteracji, wydaje mi się niewielka i w zdecydowanej większości przypadków rekomendowałbym zastosowanie metody *"schodami w górę"* do przybliżania danych wartości przy użyciu skończonych ułamków łańcuchowych.

Literatura

[1] P. Van der Cruyssen, A continued fraction algorithm, Numerische Mathematik 37 (1981), 149–156