# Pracownia z analizy numerycznej
Sprawozdanie do zadania **P1.15**

Marcin Rogala

Wrocław, 22 listopada 2018

## 1 Wstęp

Zastosowania *funkcji Bessela* obejmują między innymi mechanikę, elektrodynamikę oraz fizykę ciała stałego. Dlatego wyliczanie ich wartości jest ważnym zadaniem. Ciąg *funkcji Bessela* $J_n$ definiujemy wzorem

$$J_n(x) := \frac{1}{\pi} \int_0^\pi \cos(x \sin t - nt)\, \mathrm{d}t \qquad (\mathrm{n} = 0,\ 1,..).$$

Wiemy, że zachodzi własność

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x) \qquad (\mathrm{n} = 1,\ 2,..).$$

Powyższa zależność znaczenie ułatwia wyznaczanie wartości *funkcji Bessela* dla naturalnych parametrów $n$. W poniższym sprawozdaniu przeanalizowane zostaną dwie metody wyliczania wartości *funkcji Bessela*:

- rekurencja do przodu, korzystająca wprost z powyższej zależności
- algorytm J.C.P Millera

Omówione zostaną też wyniki przeprowadzonego doświadczenia przybliżania wartości *funkcji Bessela* przy pomocy obu algorytmów. Obliczenia wykonane zostały z pojedynczą precyzją, a następnie z precyzją 256 bitową.

## 2 Wyliczanie funkcji Bessela - rekurencja w przód

Korzystając z zależności rekurencyjnej

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x) \qquad (\mathrm{n} = 1,\ 2,..).$$

oraz znając

$$J_0(1) \approx 0.7651976866$$
$$J_1(1) \approx 0.4400505857$$

można w łatwy sposób podjąć próbę wyliczenia wartości $J_n(1)$ dla $n > 1$.

## 2.1 Częściowe wyniki obliczeń

Tabela 1: Część wyników obliczonych sposobem rekurencji w przód, typ Float32.

| $n$ | **Wartość przybliżona** | **Wartość obliczona** | **Błąd bezwzględny** |
|---|---|---|---|
| 0 | 0.7651976866 | 0.7651976866 | 0.0 |
| 1 | 0.4400505857 | 0.4400505857 | 0.0 |
| 2 | 0.1149034849319 | 0.11490345001220703 | 3.0390455947643653e-7 |
| 3 | 0.019563353982668405 | 0.019563227891921997 | 6.445252001176373e-6 |
| 4 | 0.002476638964109 | 0.002475917339324951 | 0.00029137262011392817 |
| 5 | 0.00024975773021123 4 | 0.0002441108226776123 | 0.022609540568957767 |
| 15 | 7.186396586807492 8286e-19 | -2.0073776e7 | 2.7933020057438322e25 |
| 16 | 2.11537556805326134e-20 | -6.0149536e8 | 2.843444772095692e28 |
| 17 | 5.880344573595758340e-22 | -1.9227777024e10 | 3.269838490475134e31 |
| 18 | 1.5484784412116534205e-23 | -6.5314291712e11 | 4.217965841416098e34 |
| 19 | 3.87350300852465771 89147e-25 | -2.3493915705344e13 | 6.0652891332831e37 |
| 20 | 9.227621982096670229e-27 | -8.92115643006976e14 | 2.3031236610469696e39 |

Z powyższej tabeli łatwo wywnioskować, że wraz z rosnącym $n$ błąd wyliczonej wartości bardzo szybko rośnie. Pomyłka jest tak znacząca, że już po kilku iteracjach otrzymana wartość nie ma nic wspólnego z oczekiwanym wynikiem.

## 2.2 Źródło błędu obliczeń

**Definicja 1** *Niech $f(x)$ i $g(x)$ będą funkcjami o wartościach rzeczywistych i*

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} < \infty \tag{1}$$

*Mówimy wtedy, że $g$ dominuje $f$ lub $f$ jest dominowana przez $g$.*

**Definicja 2** *Rozwiązanie zależności rekurencyjnej nazywamy minimalnym, gdy jest zdominowane przez dowolne inne rozwiązanie tej samej zależności dla prawie wszystkich argumentów.*

**Definicja 3** *Rozwiązanie zależności rekurencyjnej nazywamy dominującym, jeśli dominuje ono wszystkie inne rozwiązania tej zależności dla prawie wszystkich argumentów.*

Dwoma znanymi rozwiązaniami zależności

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x) \qquad \text{(n = 1, 2,..).}$$

są $J_n(x)$ oraz $Y_n(x)$ będące odpowiednio *funkcjami Bessela* pierwszego i drugiego rodzaju. $J_n(x)$ jest rozwiązaniem *minimalnym* co oznacza, że od pewnego $n$ wartość wyliczonego $J_{n+1}(x)$ jest bardzo blisko $Y_n(x)$, a $J_n(x)$ staje się niezauważalne (zdominowane przez $Y_n(x)$). Wiemy, że

$$\lim_{n\to\infty} J_n(1) = 0 \tag{2}$$

oraz

$$\lim_{n\to\infty} Y_n(1) = -\infty. \tag{3}$$

więc

$$\lim_{n\to\infty} |\frac{\widetilde{J_n(1)} - J_n(1)}{J_n(1)}| = \infty \tag{4}$$

gdzie $\widetilde{J_n(1)}$ to wyliczone przybliżenie $J_n(1)$.

# 3 Algorytm Millera

## 3.1 Opis algorytmu

W rozwiązaniu problemu wyliczania *minimalnego* rozwiązania zależności rekurencyjnej pomaga algorytm zaproponowany przez J.C.P Millera. Algorytm opiera się na obliczaniu kolejnych wartości $J_n(x)$ od końca, a nie jak w poprzednim przypadku od 0. Minimalne rozwiązanie $J_n(x)$ staje się wtedy dominujące. Oczywistym faktem jest to, że nie znamy początkowych wartości $J_n$ aby móc wyliczyć z nich kolejne przybliżenia. Okazuje się jednak, że wystarczy wybrać dowolne wartości a następnie po wykonaniu obliczeń przeskalować wyniki. Więcej o metodzie Millera można przeczytać m.in w [1].

## 3.2 Schemat algorytmu

- Wybrać $N > 20$ i określić pomocnicze wartości

$$c_{n+1}^{(N)} = 0.0$$
$$c_n^{(N)} = 1.0$$
$$c_{k-1}^{(N)} = \frac{2k}{x} c_k^N - c_{k+1}^N \qquad \text{(k = N, N - 1,.., 1).}$$

- Następnie obliczyć stałą

$$\lambda := J_0(x)/c_0^{(N)}$$

oraz przeskalować wyliczone wartości $c_k^{(N)}$.

$$j_k^{(N)} = \lambda c_k^{(N)}$$

- Wtedy $j_k^{(N)} \approx J_k(x)$ dla $(k = 0, 1, ..., N)$.

## 3.3 Częściowe wyniki obliczeń metodą Millera

Tabela 2: Część obliczeń algorytmem Millera dla $N = 25$, typ Float32.

| $n$ | **Wartość przybliżona** | **Wartość obliczona** | **Błąd bezwzględny** |
|---|---|---|---|
| 0 | 0.7651976866 | 0.7651976866 | 0.0 |
| 1 | 0.4400505857 | 0.4400505982355161 | 2.8486534349641307e-8 |
| 2 | 0.1149034849319 | 0.11490349147389585 | 5.693470351832169e-8 |
| 3 | 0.019563353982668405 | 0.01956335616185704 | 1.1139136147253551e-7 |
| 4 | 0.002476638964109 | 0.0024766391732307554 | 8.443772327460019e-8 |
| 5 | 0.000249757730211234 | 0.0002497577450016566 | 5.921907827801868e-8 |
| 20 | 3.87350300852465771891 47e-25 | 3.8735031986789687e-25 | 4.9091045156014705e-8 |
| 21 | 9.227621982096670229e-27 | 9.22762216319746e-27 | 1.9625943690207422e-8 |
| 22 | 2.0982239559437773488e-28 | 2.098223997123327e-28 | 1.962590780471673e-8 |
| 23 | 4.563424055950105648e-30 | 4.56342414517956e-30 | 1.9553180543027622e-8 |
| 24 | 9.5110979327124938e-32 | 9.511096592704376e-32 | 1.4088889914377573e-7 |
| 25 | 1.902951751891382e-33 | 9.1.9022193185408752e-33 | 0.00038489328475031266 |

Tabela 3: Część obliczeń algorytmem Millera dla $N = 30$, typ Float64.

| $n$ | **Wartość przybliżona** | **Wartość obliczona** | **Błąd bezwzględny** |
|---|---|---|---|
| 0 | 0.7651976866 | 0.7651976866 | 0.0 |
| 1 | 0.4400505857 | 0.44005058576910616 | 1.5704141178399132e-10 |
| 2 | 0.1149034849319 | 0.1149034849382123 | 5.493565238980972e-11 |
| 3 | 0.019563353982668405 | 0.019563353983743047 | 5.493147274669376e-11 |
| 28 | 1.211364502417112392e-38 | 1.2113645024547946e-38 | 3.11073406613847e-11 |
| 29 | 2.0891599817181681732 18e-40 | 2.0891598202727125e-40 | 7.727768907050898e-8 |
| 30 | 3.4828697942514829e-42 | 3.4819330337878546e-42 | 0.0002689622406139385 |

Dla $N = 30$ wszystkie otrzymane wyniki wynoszą *NaN*, jeśli zastosujemy pojedynczą precyzje.

Tabela 4: Część obliczeń algorytmem Millera dla $N = 25$, typ Float32, z zastosowanymi innymi wartościami początkowymi

| $n$ | **Wartość przybliżona** | **Wartość obliczona** | **Błąd bezwzględny** |
|---|---|---|---|
| 0 | 0.7651976866 | 0.7651976866 | 0.0 |
| 1 | 0.4400505857 | 0.44005057203004627 | 3.106450523817067e-8 |
| 2 | 0.1149034849319 | 0.1149034848101227 | 1.0598224819047203e-9 |
| 3 | 0.019563353982668405 | 0.019563353535429463 | 2.2861056479513607e-8 |
| 23 | 4.563424055950105648e-30 | 4.563425178332598e-30 | 2.459518289142531e-7 |
| 24 | 9.5110979327124938e-32 | 9.511260039953354e-32 | 1.7044009220316945e-5 |
| 25 | 1.902951751891382e-33 | 1.9795487901013617e-33 | 0.04025169746623808 |

## 3.4 Analiza wyników

Błąd obliczeń przedstawionych w powyższej tabeli oscyluje w okolicach $10^{-7}$. Jest to spowodowane dokładnością zastosowanych przybliżeń wartości funkcji. W rzeczywistości Algorytm Millera może otrzymać wyniki z jeszcze większa dokładnością. Wykonując obliczenia metodą Millera dla $n = 30$ należy zastosować precyzję co najmniej 64 bitową. Jest to spowodowane tym, że nieprzeskalowane wartości wyliczane podczas obliczeń bardzo szybko rosną. Zastosowanie pojedynczej precyzji spowoduje, że wartości zmiennopozycyjne zaczną osiągać nieskończoność dla *Float32*, co spowoduje utracenie wszystkich obliczeń podczas skalowania odpowiedzi. W Tabeli 4 przestawione zostały częściowe wyniki obliczeń algorytmem Millera dla zmienionych wartości początkowych. Otrzymane wyniki mają podobną dokładność do tych w Tabeli 2. Potwierdza to, że wybór wartości początkowych nie ma znaczenia.

Tabela 5: Wybrane wartości $c_k^{(N)}$ wyliczone algorytmem Millera dla $N = 30$

| $k$ | $c_k^{(N)}$ |
|---|---|
| 29 | 60.0 |
| 27 | 194764.0 |
| 6 | 6.013423106980771e36 |
| 4 | Inf |
| 0 | NaN |

# 4 Obliczenia dla podwyższonej precyzji

Wykonując obliczenia metodą rekurencji w przód dla precyzji 256 bitowej łatwo zaobserwować, że wyniki poprawiają się nieznacznie. Tak jak dla pojedynczej precyzji już $J_7(x)$ jest bardzo daleko od oczekiwanej wartości. Błąd rośnie w podobnym tempie. Jest to spowodowane tym, że błąd przybliżenia nie wynika z niedokładności obliczeń zmiennoprzecinkowych. Algorytm wylicza rozwiązanie dominujące zależności rekurencyjnej które jest równe $Y_n(x)$. Zjawisko zostało opisane w 2.2.

# 5 Wnioski

## 5.1 Rekurencja w przód

Algorytm wyliczania przybliżeń *funkcji Bessela* korzystający wprost z zależności rekurencyjnej

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x) \qquad \text{(n = 1, 2,..).}$$

charakteryzuje się bardzo małą skutecznością. Jesteśmy w stanie przy jego użyciu wyliczyć zaledwie kilka pierwszych wartości, niezależnie od użytej precyzji. Zaletą tego sposobu jest jego czytelność oraz bardzo łatwa implementacja.

## 5.2 Algorytm Millera

Największą zaletą algorytmu Millera jest jego duża skuteczność. Korzystając z tej metody jesteśmy w stanie obliczyć z bardzo dużą dokładnością zdecydowanie więcej wyrazów, niż poprzednią metodą. Niestety jesteśmy ograniczeni przez dostępne typy danych, ponieważ wyliczane w algorytmie wartości przed przeskalowaniem osiągają bardzo duże wartości.

# Literatura


[1] WALTER GAUTSCHI: *COMPUTATIONAL ASPECTS OF THREE-TERM RECURRENCE RELATIONS*, Vol. 9, No. 1, January, 1967

[2] J. R. CASH: *Stability Concepts in the Numerical Solution of Difference and Differential Equations*, Computers Math. Applic. Vol. 28, No. 1-3, pp. 45-53, 1994

[3] Jet Wimp: *Computation with Recurrence Relations* , 1984