1. naloga: Airyjevi funkciji

Samo Krejan, 28231092

1 Uvod

Airyjevi funkciji Ai in Bi se v fiziki pojavljata predvsem v optiki in kvantni mehaniki [**?**]. Definirani sta kot neodvisni rešitvi enačbe

$$y''(x) - xy(x) = 0$$

in sta predstavljivi v integralski obliki

$$\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos(t^3/3 + xt)\,\mathrm{d}t\,, \quad \mathrm{Bi}(x) = \frac{1}{\pi}\int_0^\infty \left[\mathrm{e}^{-t^3/3+xt} + \sin(t^3/3+xt)\right]\,\mathrm{d}t\,.$$

Za majhne x lahko funkciji Ai in Bi izrazimo z Maclaurinovima vrstama

$$\mathrm{Ai}(x) = \alpha f(x) - \beta g(x)\,, \qquad \mathrm{Bi}(x) = \sqrt{3}\left[\alpha f(x) + \beta g(x)\right]\,,$$

kjer v $x = 0$ veljata zvezi $\alpha = \mathrm{Ai}(0) = \mathrm{Bi}(0)/\sqrt{3} \approx 0.355028053887817239$ in $\beta = -\mathrm{Ai}'(0) = \mathrm{Bi}'(0)/\sqrt{3} \approx 0.258819403792806798$. Vrsti za f in g sta

$$f(x) = \sum_{k=0}^\infty \left(\frac{1}{3}\right)_k \frac{3^k x^{3k}}{(3k)!}\,, \qquad g(x) = \sum_{k=0}^\infty \left(\frac{2}{3}\right)_k \frac{3^k x^{3k+1}}{(3k+1)!}\,,$$

kjer je

$$(z)_n = \Gamma(z+n)/\Gamma(z)\,, \qquad (z)_0 = 1\,.$$

Za velike vrednosti $|x|$ Airyjevi funkciji aproksimiramo z njunima asimptotskima razvojema. Z novo spremenljivko $\xi = \frac{2}{3}|x|^{3/2}$ in asimptotskimi vrstami

$$L(z) \sim \sum_{s=0}^\infty \frac{u_s}{z^s}\,, \qquad P(z) \sim \sum_{s=0}^\infty (-1)^s \frac{u_{2s}}{z^{2s}}\,, \qquad Q(z) \sim \sum_{s=0}^\infty (-1)^s \frac{u_{2s+1}}{z^{2s+1}}\,,$$

s koeficienti

$$u_s = \frac{\Gamma(3s+\frac{1}{2})}{54^s s!\,\Gamma(s+\frac{1}{2})}$$

za velike pozitivne x izrazimo

$$\mathrm{Ai}(x) \sim \frac{\mathrm{e}^{-\xi}}{2\sqrt{\pi}x^{1/4}}\,L(-\xi)\,, \qquad \mathrm{Bi}(x) \sim \frac{\mathrm{e}^{\xi}}{\sqrt{\pi}x^{1/4}}\,L(\xi)\,,$$

za po absolutni vrednosti velike negativne x pa

$$\mathrm{Ai}(x) \sim \frac{1}{\sqrt{\pi}(-x)^{1/4}}\Big[\ \ \sin(\xi-\pi/4)\,Q(\xi) + \cos(\xi-\pi/4)\,P(\xi)\Big]\,,$$
$$\mathrm{Bi}(x) \sim \frac{1}{\sqrt{\pi}(-x)^{1/4}}\Big[-\sin(\xi-\pi/4)\,P(\xi) + \cos(\xi-\pi/4)\,Q(\xi)\Big]\,.$$

2 Naloga

Z uporabo kombinacije Maclaurinove vrste in asimptotskega razvoja poišči čim učinkovitejši postopek za izračun vrednosti Airyjevih funkcij Ai in Bi na vsej realni osi z **absolutno** napako, manjšo od 10^{-10}. Enako naredi tudi z **relativno** napako in ugotovi, ali je tudi pri le-tej dosegljiva natančnost, manjša od 10^{-10}. Pri oceni napak si pomagaj s programi, ki znajo računati s poljubno natančnostjo, na primer z MATHEMATICO in/ali paketi MPMATH in DECIMAL v programskem jeziku PYTHON.

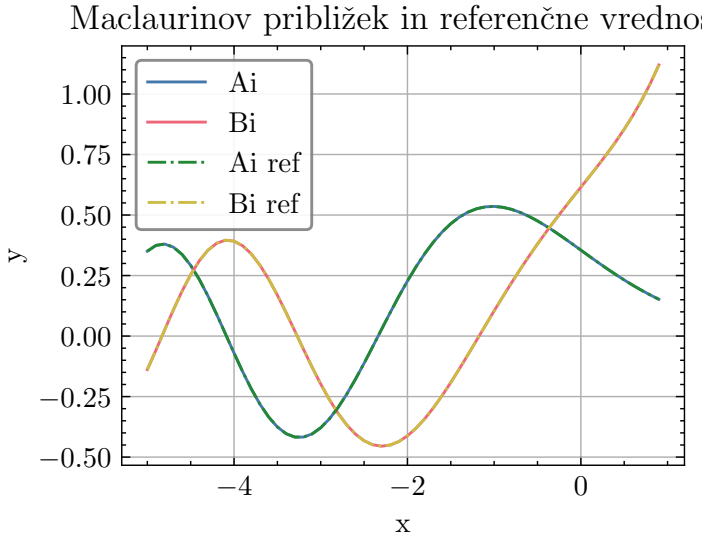
2.1 Maclaurinov približek

Člene vrst f in g zapišemo rekurzivno, in n-ti člen zapišemo kot produkt prejšnjega in količnika:

$$f_n = f_{n-1} \cdot \frac{f_n}{f_{n-1}} = f_{n-1} \cdot \frac{x^3}{3n(3n-1)}\,, \qquad f_0 = 1$$

$$g_n = g_{n-1} \cdot \frac{g_n}{g_{n-1}} = g_{n-1} \cdot \frac{x^3}{3n(3n+1)}\,, \qquad g_0 = x$$

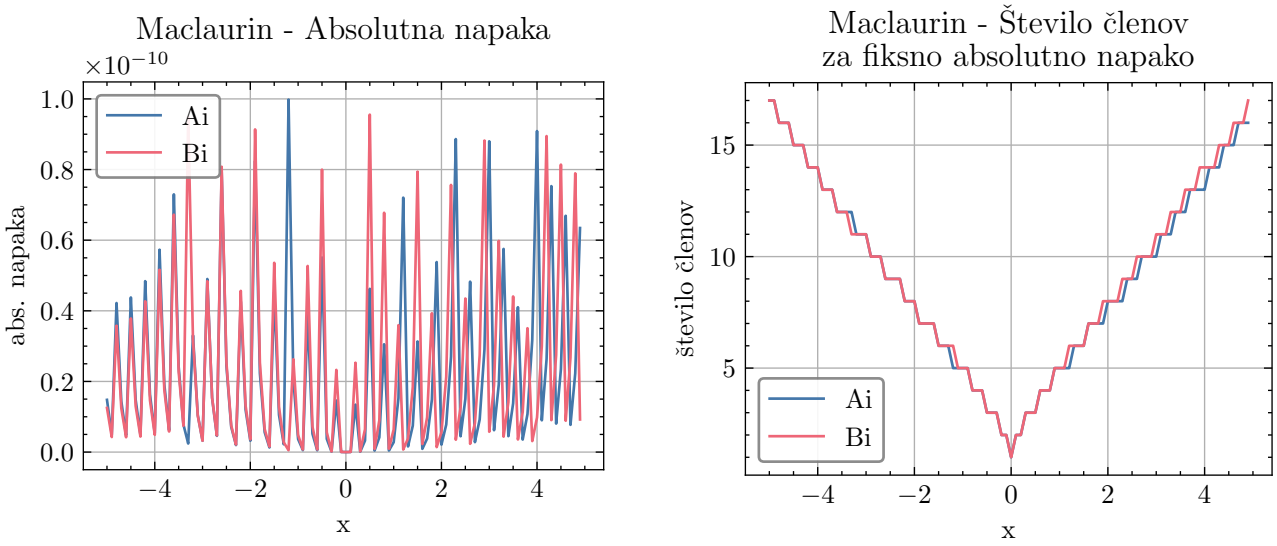
Dobljene rezultate bomo skozi nalogo primerjali z vgrajenima funkcijama *mpmath.airyai*() in *mpmath.airybi*(), ki gotovo od dejanske vrednosti odstopata manj kot $\cdot 10^{-20}$ [**?**].



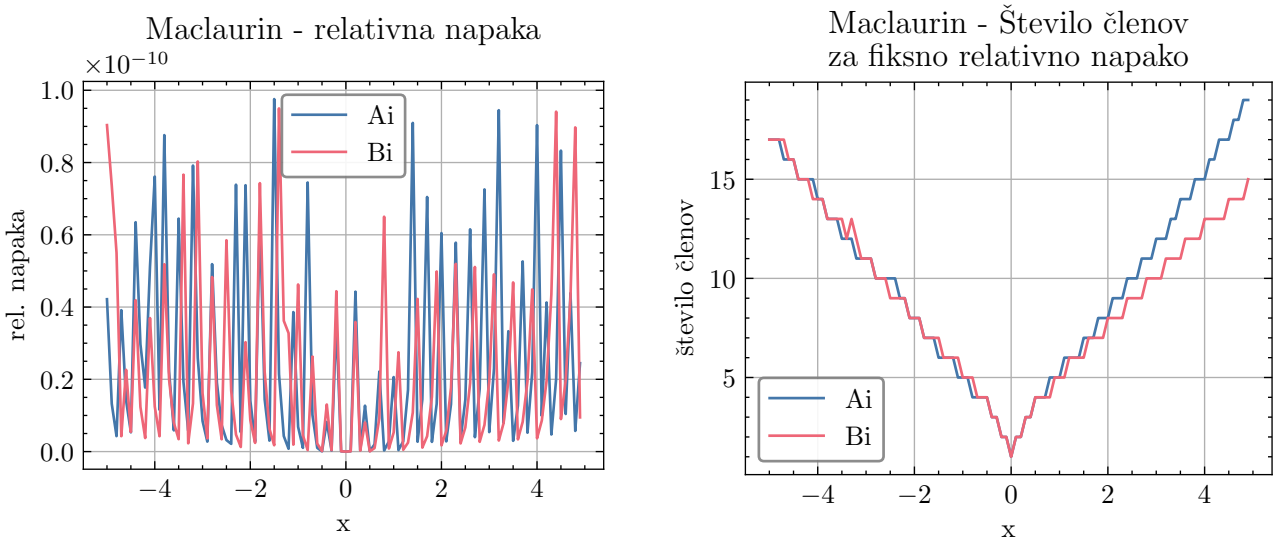
Slika 1: Maclaurinov približek *Ai* in *Bi* primerjan z vgrajenima funkcija

Že na prvi pogled, ko grafiramo dobljene vrednosti za obe funkciji in jih primerjamo z vgrajenima funkcijama vidimo (slika: 1), da je Maclaurinov približek precej uspešen. To še dodatno potrdita sliki 2 in 3, kjer vidimo, da nam je na območju $(-4, 4)$ uspelo *ujeti* tako željeno absolutno kot relativno napako z manj kot dvajsetimi členi.

Zanimivost, ki se mi jo zdi vredno omeniti lahko opazimo na desni sliki 3, kjer vidimo, da se število potrebnih členov funkcije *Ai* hitro veča. To je seveda smiselno, ko pomislimo, da tam *Ai* dosega vrednosti zelo blizu 0 in je tako seveda za enako relativno natančnost, potrebna veliko boljša absolutna natančnost.



Slika 2: Absolutna napaka naše implementacije in število členov potrebnih, da le ta ostane pod 10^{-10}. Iz grafov vidimo, da se napaka sicer z oddaljenostjo od izhodišča strmo veča, a vsakič ko bi le ta zgrešila željeno natančnost se vrsti doda en člen. To pojasni močno zobničasto naravo napake.



Slika 3: Relativna napaka in potrebno število členov vrste za doseg te napake. Karakteristika je ekvivalentna tisti pri absolutni napaki. Tudi potrebno število členov se ne spremeni znatno, le pri večjih pozitivnih vrednostih opazimo, da potrebujemo vedno več členov za funkcijo Ai.

2.2 Asimptotski približek za velike negativne x

Člene vrst razvoja funcij P in Q prav tako, kot pri Maclaurinovem razvoju zapišemo rekurzivno, saj tako močno izboljšamo hitrost algoritma.

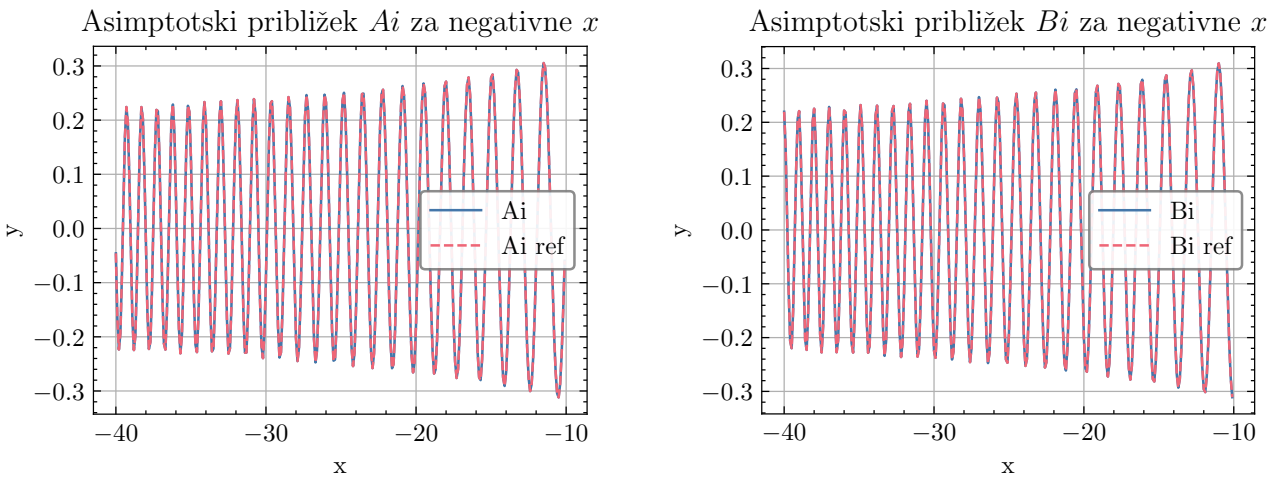
$$p_n = p_{n-1} \cdot \frac{p_n}{p_{n-1}} = -p_{n-1} \cdot \frac{(6n-1/2)(6n-5/2)(6n-7/2)(6n-11/2)}{(2n-1)(2n)18^2x^2}, \qquad p_0 = 1$$

$$q_n = q_{n-1} \cdot \frac{q_n}{q_{n-1}} = -q_{n-1} \cdot \frac{(6n-1/2)(6n-5/2)(6n+1/2)(6n+5/2)}{(2n+1)(2n)18^2x^2}, \qquad q_0 = \frac{5}{72x}$$

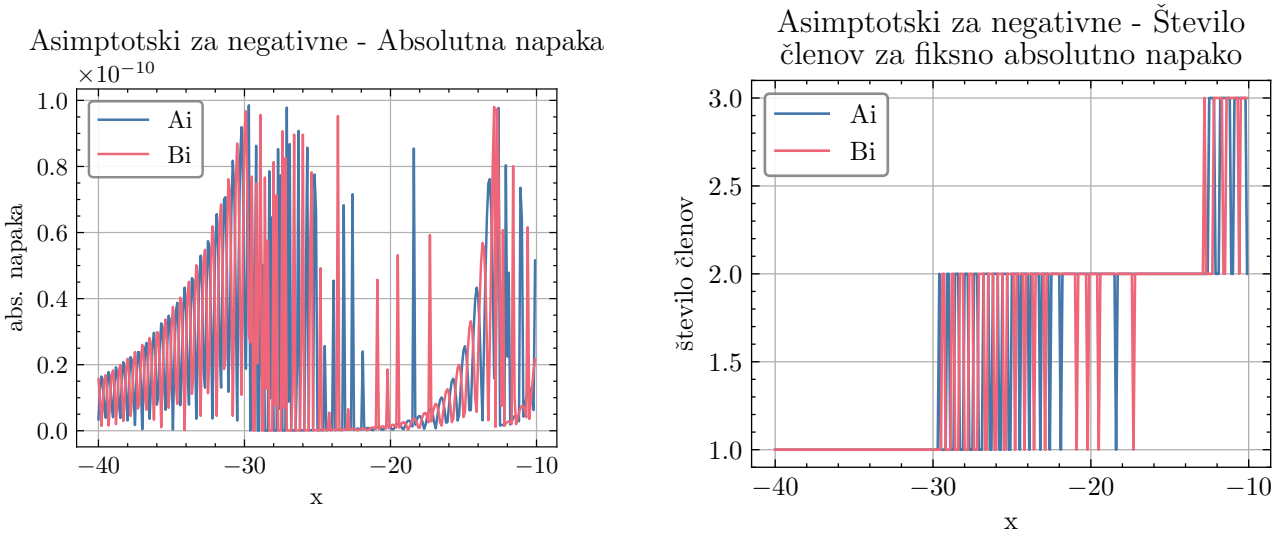
Pri obeh asimptotskih priblizkih moramo sicer paziti, da ne uporabimo preveč členov v vrsti. Izkaže se

namreč, da se začne napaka pri nekem členu nazaj večati. Iz tega razloga vrsto odrežemo, ko naslednji člen postane večji od prejšnjega. Poraja se vprašanje, če je to res najboljša meja za *odsek* vrste:

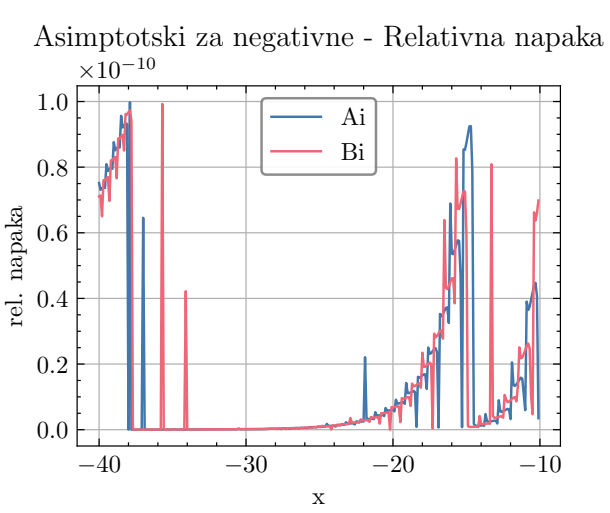
> Odgovor je da je to tok blizu, da je praktično optimalno. (Jakob Kralj, 2025)

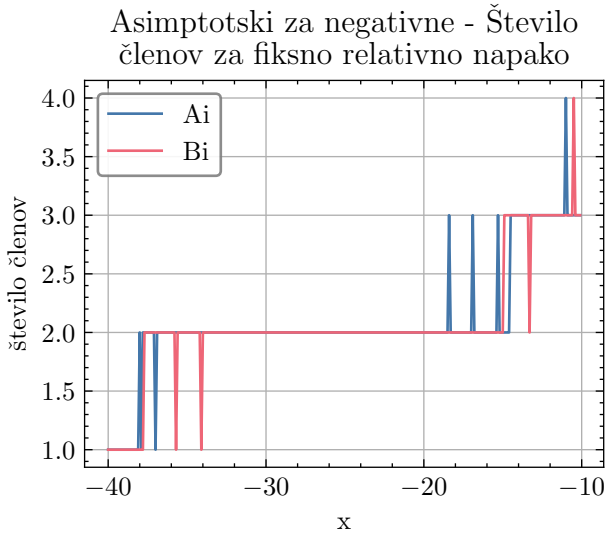


Slika 4: Asimptotski približek za Ai (levo) in Bi (desno) primerjan z vgrajenima funkcijama.



Slika 5: Absolutna napaka (levo) in število členov asimptotske vrste potrebnih za doseg le te. Zanimivo vidimo, da bolj kot je negativen x manj manjša je napaka in je zato posledično potrebnih manj členov.





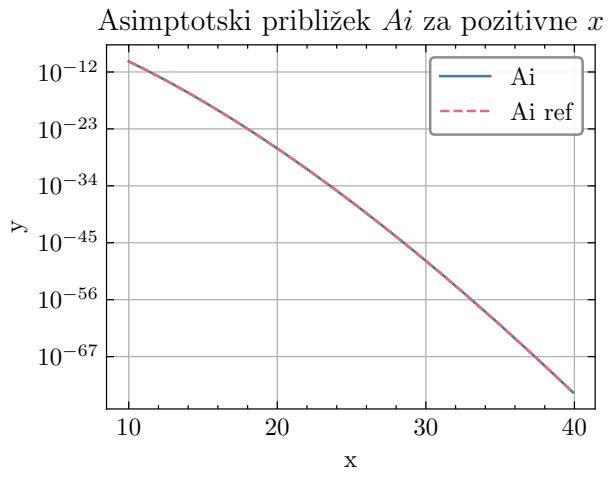
Slika 6: Relativna napaka (levo) in število členov potrebnih za doseg le te (desno). Sama karakteristika je seveda zelo podobna kot pri absolutni napaki (glej sliko 5) a lahko opazimo, da je v splošnem za konstantno relativno napako potrebnih nekoliko več členov

2.3 Asimptotski približek za velike pozitivne x

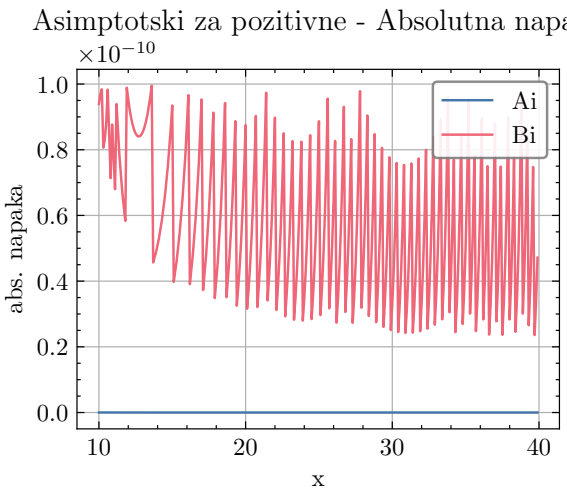
Za velike pozitivne x uporabimo vrsto L opisano v poglavju 1. Katere člene lahko spet zapišemo rekurzivno za potrebe optimizacije:

$$l_n = l_{n-1} \cdot \frac{l_n}{l_{n-1}} = l_{n-1} \cdot \frac{(3n+1/2)(3n+5/2)}{(n+1)18x}, \qquad l_0 = 1$$

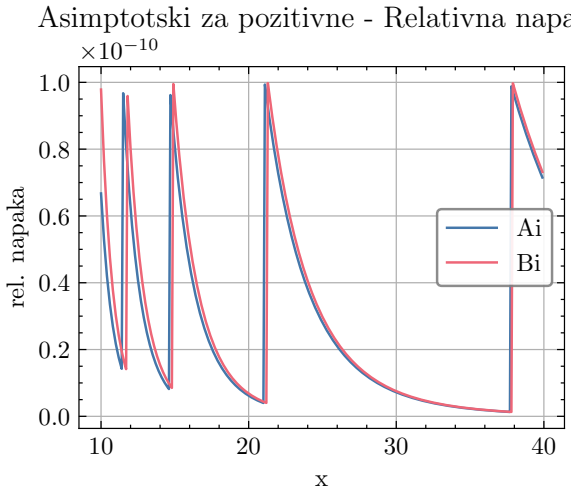
Pristop od tu naprej je seveda ekvivalenten tistemu za velike negativne x.



Slika 7: Asimptotski približek za Ai (levo) in Bi (desno) primerjan z vgrajenima funkcijama v logoritemski skali.

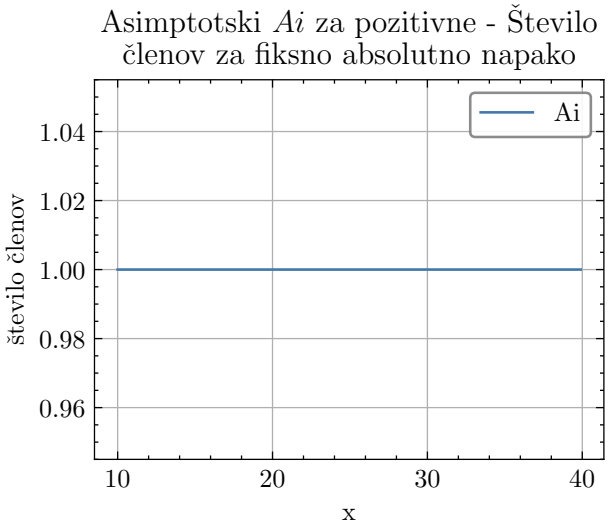


(a) Absolutna napaka asimptotske vrste za pozitivne x, napaka za Ai je praktično 0 saj je tudi sama vrednost funkcije zelo blizu ničle.

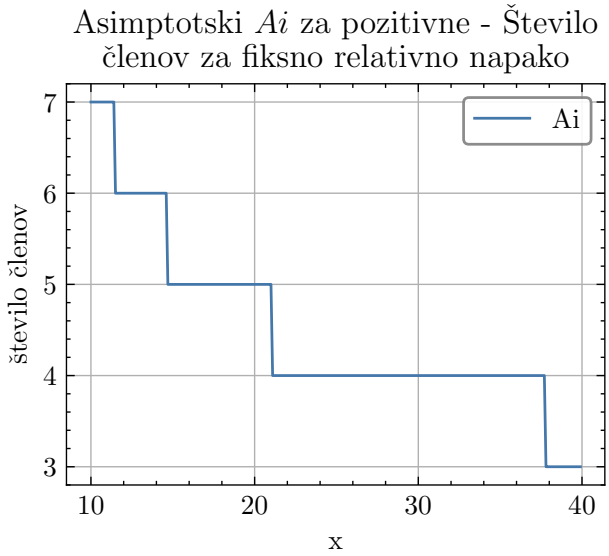


(b) Relativna napaka asimptotske vrste za pozitivne x. Opazimo, da sta napaki nadvse podobni, kar je smiselno glede na naravo razvoja funkcij.

Slika 8: Primerjava absolutne in relativne napake



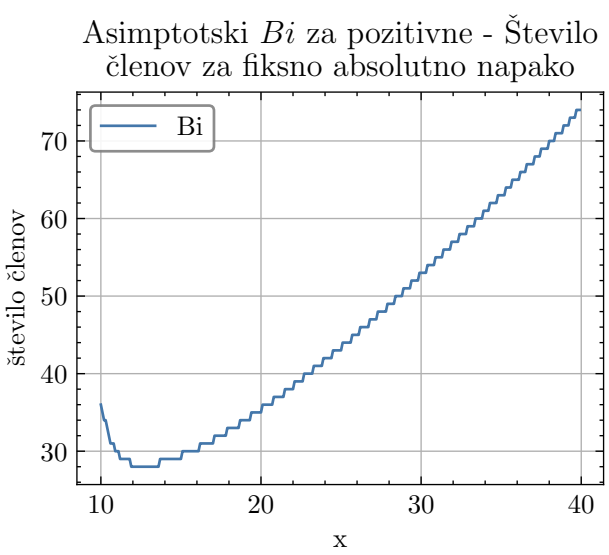
(a) Število členov potrebnih za fiksno absolutno napako. Ta ostane 1 saj se vrednost funkcije praktično ne spreminja, ko pride tako blizu ničle.



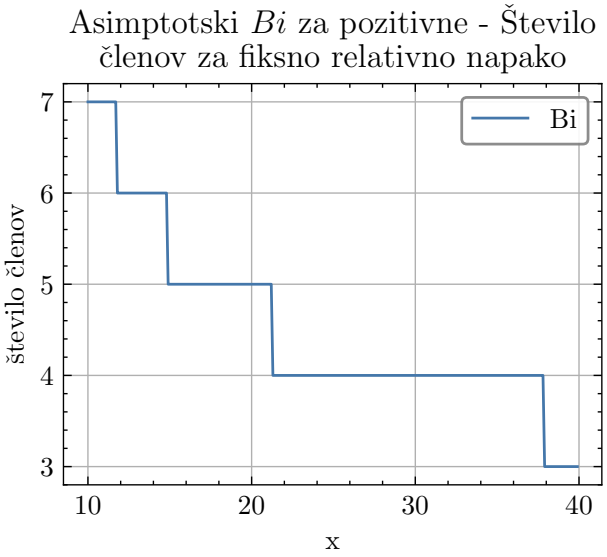
(b) Število členov potrebnih za fiksno relativno napako. Vidimo, da se le ta manjša.

Slika 9: Število členov za fiksno absolutno (levo) in relativno (desno) napako za Ai

Vidimo, da se asimptotski razvoj za pozitivne x obnaša zelo podobno, ko zahtevamo konstantno relativno napako, čeprav se funkciji močno razlikujeta - Ai eksponentno pada, medtem ko Bi eksponentno narašča (vidno na sliki: 7). Razlika med funkcijama se zares izrazi šele ko fiksiramo absolutno napako. Očitno enaka relativna napaka pomeni mnogo večjo absolutno napako pri Bi in skoraj nično napako pri Ai. Posledično za enako absolutno napako seveda potrebujemo mnogo več delcev.



(a) Število členov potrebnih za fiksno absolutno napako. Ta z večanjem x hitro raste.



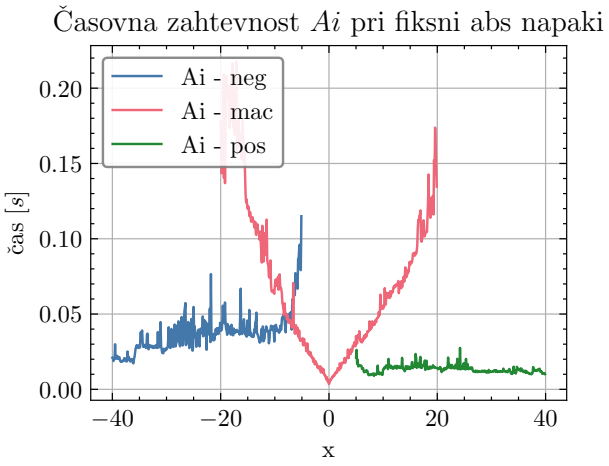
(b) Število členov potrebnih za fiksno relativno napako. Karakteristika je podobna tisti pri 9b

Slika 10: Število členov za fiksno absolutno (levo) in relativno (desno) napako za Bi

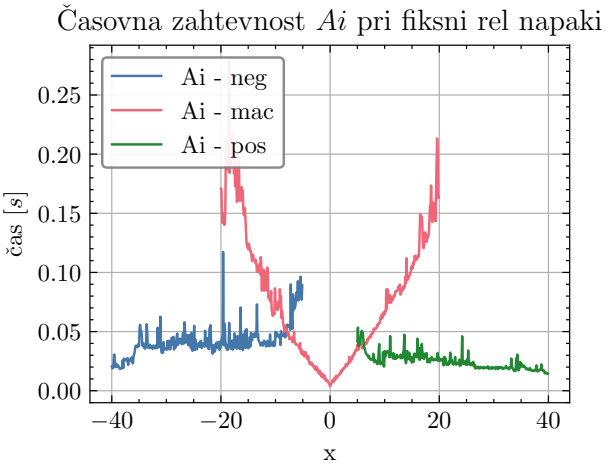
2.4 Zlepek

Hitro postane očitno, da nobena izmed metod ni učinkovita po celotni realni osi. Maclaurinov približek deluje le za relativno majhne x po absolutni vrednosti, saj za večje potrebujemo mnogo členov, kar pa vzame veliko časa. Tu v pomoč prideta asimptotska razvoja posebaj za velike negativne x in velike pozitivne x. Torej kar preostane ugotoviti je na katerih točkah bomo nehali upoštevati Maclaurinov razvoj in začeli uporabljati asimptotski razvoj. Za potrebe iskanja teh točk grafiramo čas porabljen za izračun funkcij z različnimi metodami preko realne osi z različnimi zahtevami za napako (sliki 11 in 12).

Za vsakega izmed štirih *režimov* lahko tako zgrafiramo *zlepek*, kjer so meje med posameznimi uporabljenimi metodami določene s presečišči na grafih porabljenega časa.

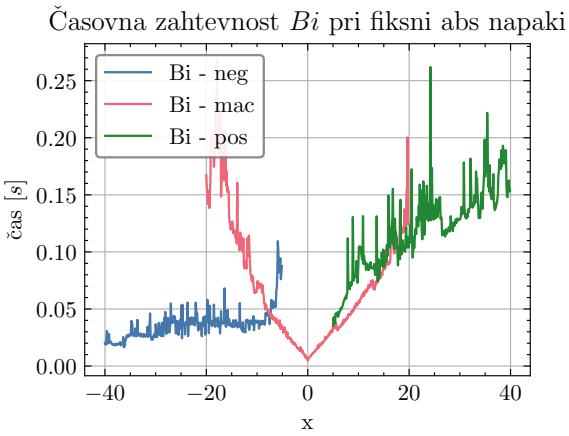


(a) Čas potreben za 100 iteracij Ai pri vsaki vrednosti x za različne metode pri konstantni absolutni napaki.

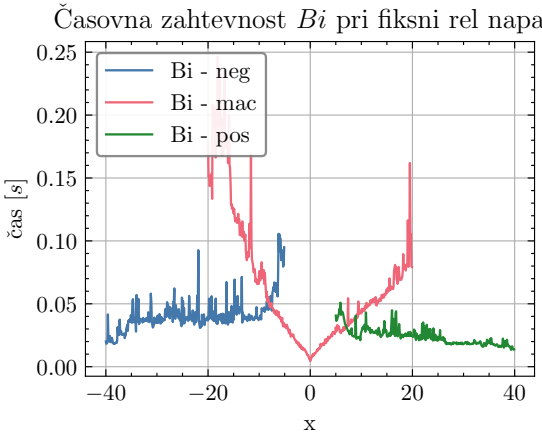


(b) Čas potreben za 100 iteracij Ai pri vsaki vrednosti x za različne metode pri konstantni relativni napaki.

Slika 11: Časovna zahtevnost Ai

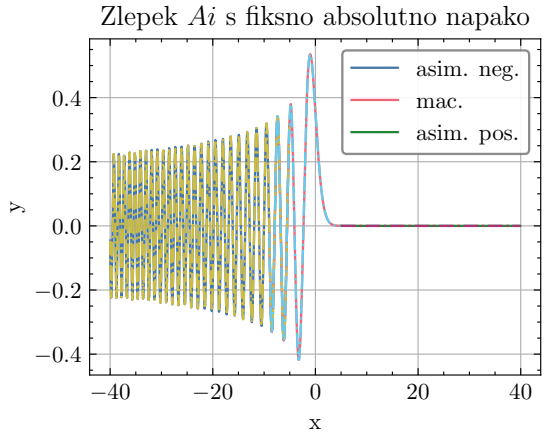


(a) Čas potreben za 100 iteracij Bi pri vsaki vrednosti x za različne metode pri konstantni absolutni napaki.

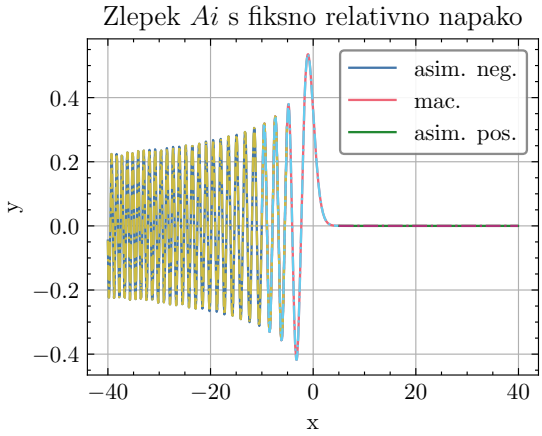


(b) Čas potreben za 100 iteracij Bi pri vsaki vrednosti x za različne metode pri konstantni relativni napaki.

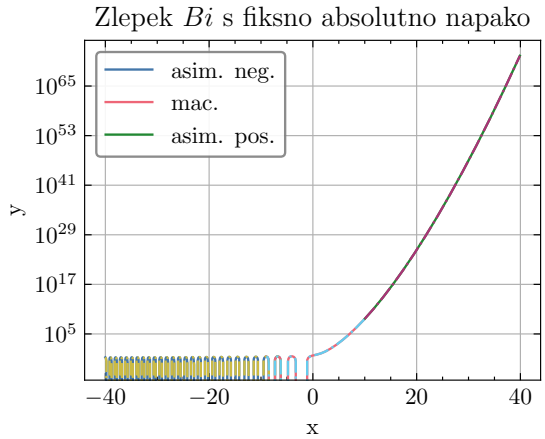
Slika 12: Časovna zahtevnost Bi



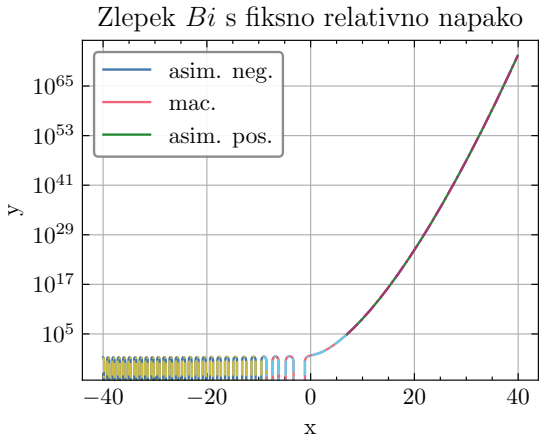
(a) Čas potreben za 100 iteracij Ai pri vsaki vrednosti x za različne metode pri konstantni absolutni napaki.



(b) Čas potreben za 100 iteracij Ai pri vsaki vrednosti x za različne metode pri konstantni relativni napaki.



(a) Čas potreben za 100 iteracij Bi pri vsaki vrednosti x za različne metode pri konstantni absolutni napaki.



(b) Čas potreben za 100 iteracij Bi pri vsaki vrednosti x za različne metode pri konstantni relativni napaki.

Pri naši implementaciji se seveda zanašamo, da vemo koliko členov je potrebnih za željeno numerično natančnost. Mislim, da razen pri asimptotski vrsti pozitivnih x za funkcijo Bi ne bi smelo biti težko postaviti zgornjih mej za število potrebnih členov. Tudi, če meje za število členov ne bi imeli, bi naša funkcija delovala dobro, saj smo v asimptotskem razvoju implementirali pogoj za zaustavitev, ko se napaka začne večati, za Maclaurinovo vrstopa lahko preprosto postavimo mejo za maksimalno ≈ 50 členov, kar nam zagotovi željeno natančnost v vseh režimih. S tem bi sicer zgubili nekaj na časovni zahtevnosti, a bi morda pridobili nekaj na spominu.

3 Dodatna naloga

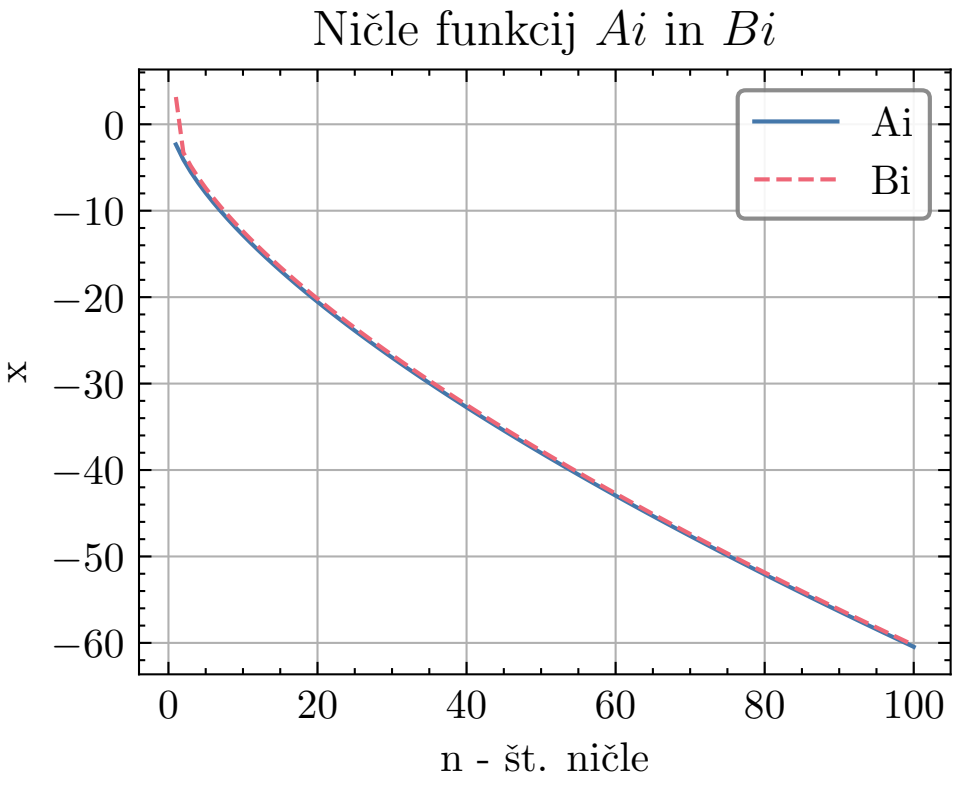
Ničle funkcije Ai pogosto srečamo v matematični analizi pri določitvi intervalov ničel specialnih funkcij in ortogonalnih polinomov [**?**] ter v fiziki pri računu energijskih spektrov kvantnomehanskih sistemov [**?**]. Poišči prvih sto ničel $\{a_s\}_{s=1}^{100}$ Airyjeve funkcije Ai in prvih sto ničel $\{b_s\}_{s=1}^{100}$ funkcije Bi pri $x < 0$ ter dobljene vrednosti primerjaj s formulama

$$a_s = -f\left(\frac{3\pi(4s-1)}{8}\right), \qquad b_s = -f\left(\frac{3\pi(4s-3)}{8}\right), \qquad s = 1, 2, \dots,$$

kjer ima funkcija f asimptotski razvoj [**?**]

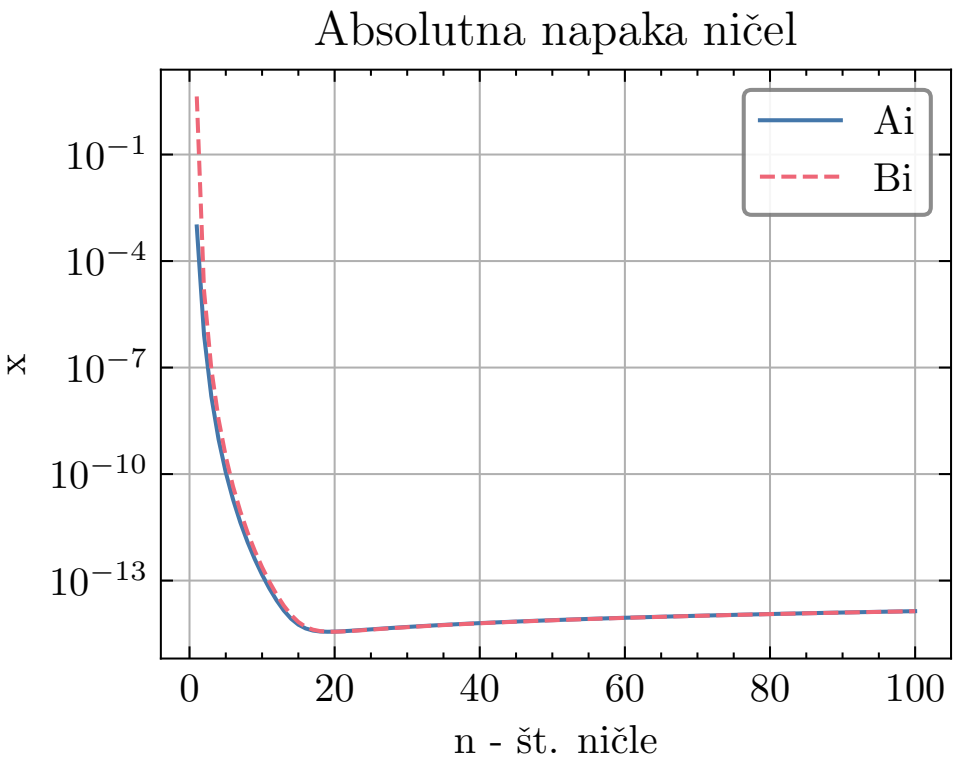
$$f(z) \sim z^{2/3}\left(1 + \frac{5}{48}\,z^{-2} - \frac{5}{36}\,z^{-4} + \frac{77125}{82944}\,z^{-6} - \frac{108056875}{6967296}\,z^{-8} + \dots\right).$$

Vrsto za a_s in b_s smo implementirali s funkcijo f in rezultate primerjali z vgrajenima funkcijama *mpmath.airyaizero*() in *mpmath.airybizero*()



Slika 15: x vrednost n-te ničle

Vidimo sicer, da za nekaj začetnih ničel vrsta ni najbolj natančna, a že po petih členih absolutna napaka pade pod $\cdot 10^{-10}$.



Slika 16: Absolutna napaka n-te ničle

4 Zaključek

Menim, da je bila naloga opravljena uspešno. Na testnem območju nam je uspelo ujeti željene natančnosti v precej dobrem času glede na to da smo uporabili Python.

Tekom dela sem naletel na mnoge probleme, ki so bili zame novi. Od tega, da sem se naučil delati s knjižnico *mpmath*, ki omogoča računanje do poljubne numerične natančnosti, do tega da sem zaradi ekstenzivnega testiranja paraleliziral nekatere procese. Poleg tega, se mi je v kodo prikradlo nekaj hroščev, ki so skupaj vzeli vsaj nekaj ur mojega življenja, saj so bili zelo dobro skriti. Za pomoč pri odkrivanju teh hroščev in razjasnjevanje misterij neznane knjižnice se moram zahvaliti $Chat-GPT$ju, ki je marsikatero nejasnost hitro razrešil.

Kljub temu, da mi je naloga definitivno vzela veliko več časa kot sem pričakoval (krat 2 in naslednja časovna enota), sem v izdelovanju naloge užival, posebaj za to, ker sem se naučil več novega kot sem pričakoval.