1 Ničle Airyjeve funkcije

Avtor: Luka Bajić

1.1 Opis problema

Problem, ki ga rešujemo je sestavljen iz dveh delov: numeričnega reševanja diferencialne enačbe drugega reda in iskanja ničel funkcije z metodami kot so bisekcija in regula falsi. Rezultati morajo biti natančni na deset decimalnih mest.

Želimo poiskati čimveč ničel Airyjeve funkcije, ki je dana z naslednjo diferencialno enačbo:

$$Ai''(x) - xAi(x) = 0$$

pri začetnih pogojih

$$Ai(0) = \frac{1}{3^{\frac{2}{3}}\Gamma(\frac{2}{3})}, Ai'(0) = -\frac{1}{3^{\frac{1}{3}}\Gamma(\frac{1}{3})}$$

Vrednosti funkcije lahko računamo z uporabo Magnusove metode, pri kateri se z izbranim korakom h premikamo od začetne vrednosti v levo po x-osi (ker vemo, da ima funkcija Ai vse ničle negativne). Premik izvajamo z naslednjo formulo:

$$y_{k+1} = exp(\frac{h}{2}(A_1 + A_2) - \frac{\sqrt{3}}{12}h^2[A_1, A_2])$$

pri čemer je $A_{1,2} = A(x_k + (\frac{1}{2} \pm \frac{\sqrt{3}}{6})h)$ in $[A_1, A_2] = A_1A_2 - A_2A_1$. Matriko A pa dobimo tako, da zgornjo diferencialno enačbo drugega reda prevedemo na sistem diferencialnih enačb prvega reda, in sicer:

$$Y'(x) = \begin{bmatrix} Ai'(x) \\ Ai''(x) \end{bmatrix} = \begin{bmatrix} Ai'(x) \\ xAi'(x) \end{bmatrix}$$

kar lahko preoblikujemo v ustrezno obliko za Magnusovo metodo:

$$Y'(x) = \begin{bmatrix} 0 & 1 \\ x & 0 \end{bmatrix} \begin{bmatrix} Ai(x) \\ Ai'(x) \end{bmatrix}$$

kjer je iskana matrika torej $A = \begin{bmatrix} 0 & 1 \\ x & 0 \end{bmatrix}$

1.2 Opis rešitve

Za izračun vrednosti Airyjeve funkcije enostavno sledimo zgoraj navedenim formulam, pri čemer lahko določene vrednosti, npr. vrednosti začetnih pogojev, vnaprej izračunamo z orodjem kot je Wolfram Alpha, ker je njihova vrednost neodvisna od ostalih parametrov. Ostali izračuni se izvajajo v metodi *airy_premik*, ki na podlagi prejšnjih vrednosti y_k, in y'_k

pri x_k, in parametra za korak h, ki ga lahko poljubno določimo, izračuna vrednosti funkcije pri $x_k + h$, torej y_{k+1} in y'_{k+1}.

1.2.1 Iskanje ničel

Implementacija ponuja dve možnosti za iskanje ničel: uporabnik specificira interval $[a, 0]$, na katerem metoda *airy_nicle_na_intervalu* najde vse ničle, ali pa specificira vrednost k, nakar metoda *airy_k_nicel* vrne prvih k ničel od koordinatnega izhodišča proti $-\infty$.

Razlika med metodama je samo v zaustavitvenem pogoju, postopek iskanja posamezne ničle ostaja enak, in sicer: z zgoraj omenjenim postopkom se s korakom h premikamo po funkciji in na vsakem koraku preverjamo ali se predznak vrednosti funkcije razlikuje od predznaka vrednosti funkcije na prejšnjem koraku. Ker vemo, da je funkcija zvezna, razlika med prezdnakoma pomeni, da se na intervalu med x_k in x_{k+1} nahaja ničla. Ker je zahtevana natančnost na deset decimalk (h pa je tipično bistveno večji in posledično ni dovolj natančen), se na tem mestu izvede ena izmed metod regula falsi, bisekcija ali tangentna metoda (izbiro podamo kot argument) za iskanje ničle.

Bisekcija Interval $[a, b]$, pri čemer sta $y(a)$ in $y(b)$ različno predznačena, razpolovimo, izračunamo vrednost funkcije v srednji točki z Magnusovo metodo (torej z metodo *airy_premik*) in nato izberemo podinterval $[a, \frac{b-a}{2}]$ ali $[\frac{b-a}{2}, b]$ pri katerem se predznaka v robnih točkah razlikujeta. Postopek ponavljamo dokler ni razlika med sosednjima vrednostma, tako po x-u kot po y-u manjša od želene tolerance.

Regula falsi Podobno kot pri bisekciji, na vsakem koraku izberemo podinterval kjer sta predznaka v robnih točkah različna, razlika je le, da pri regula falsi ne vzamemo srednje točke, temveč jo izračunamo s sledečo formulo:

$$c = b - \frac{y(b)(b-a)}{y(b) - y(a)}$$

Nato izberemo ustrezen interval izmed $[a, c]$, $[c, b]$. Pogoj za zadovoljeno toleranco ostaja enak.

Tangentna metoda Pri tangentni metodi je pogosto problematično računanje odvoda funkcije, v našem primeru pa nam ga Magnusova metoda računa v vsakem koraku, torej ga lahko direktno uporabimo v sledeči formuli:

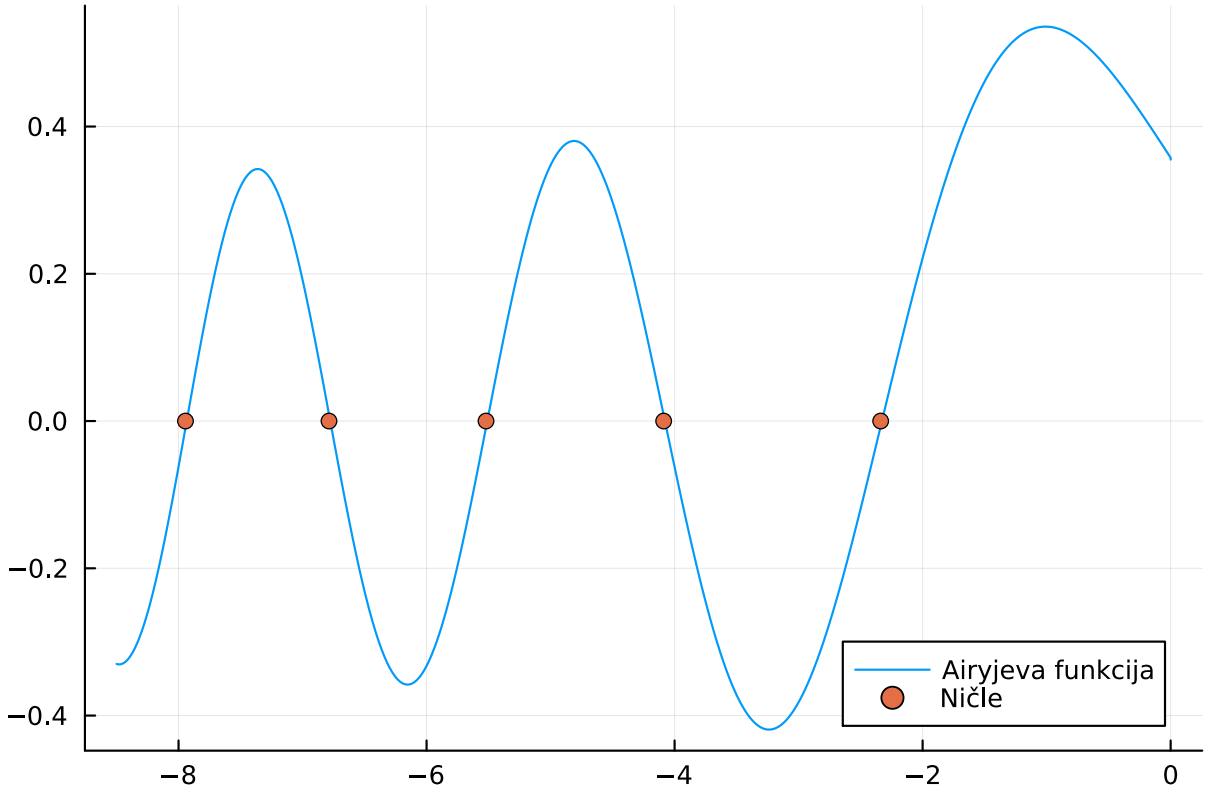
$$x_{k+1} = x_k - \frac{y(x_k)}{y'(x_k)}$$

Pri tej metodi ne gledamo predznakov, temveč samo ponavljamo iteracijo z zgornjo enačbo dokler ni izpolnjen zaustavitveni pogoj.

1.3 Primer uporabe

Spodnji izsek kode prikazuje primer uporabe metode *airy_nicle_na_intervalu*, ki najde vse ničle na specificiranem intervalu. Z zastavico *vrni_vmesne* dobimo vrnjene tudi vse izračunane vmesne vrednosti, ki jih lahko izrišemo s paketom Plots.

```
using Airy, Plots
nicle, xs, ys = airy_nicle_na_intervalu(-8.5, vrni_vmesne=true)
plot(xs[:], ys[:], label="Airyjeva funkcija")
scatter!(nicle[:], zeros(size(nicle)), label="Ničle")
```



Še ena dodatna funkcionalnost, ki jo paket omogoča, je izračun povprečnega števila iteracij, ki ga posamezna metoda potrebuje, da skonvergira k dovolj točni rešitvi. Spodnji izsek kode prikazuje primer uporabe te funkcionalnosti.

```
k = 3000
_, avg1 = airy_k_nicel(k, metoda="bisekcija", vrni_povp_iter=true)
_, avg2 = airy_k_nicel(k, metoda="regula", vrni_povp_iter=true)
_, avg3 = airy_k_nicel(k, metoda="tangentna", vrni_povp_iter=true)

println("Bisekcija potrebuje povprecno ", round(avg1, sigdigits=5)," iteracij za izracun
prvih $k nicel.")
println("Regula falsi potrebuje povprecno ", round(avg2, sigdigits=5)," iteracij za
izracun prvih $k nicel.")
println("Tangentna metoda potrebuje povprecno ", round(avg3, sigdigits=5)," iteracij za
izracun prvih $k nicel.")
```

```
Bisekcija potrebuje povprecno 28.165 iteracij za izracun prvih 3000 nicel.
Regula falsi potrebuje povprecno 27.825 iteracij za izracun prvih 3000 nice
l.
Tangentna metoda potrebuje povprecno 3.1527 iteracij za izracun prvih 3000
nicel.
```

Kot vidimo iz izpisa, je tangentna metoda bistveno hitrejša in ker vemo, da je Airyjeva funkcija vedno strma blizu ničle, ni težav s konvergenco in lahko torej tangentno metodo nastavimo kot privzeto metodo, če uporabnik ne specificira druge.