

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM
1. naloga: Airyjevi funkciji

Žiga Šinigoj, 28191058

Ljubljana, oktober 2021

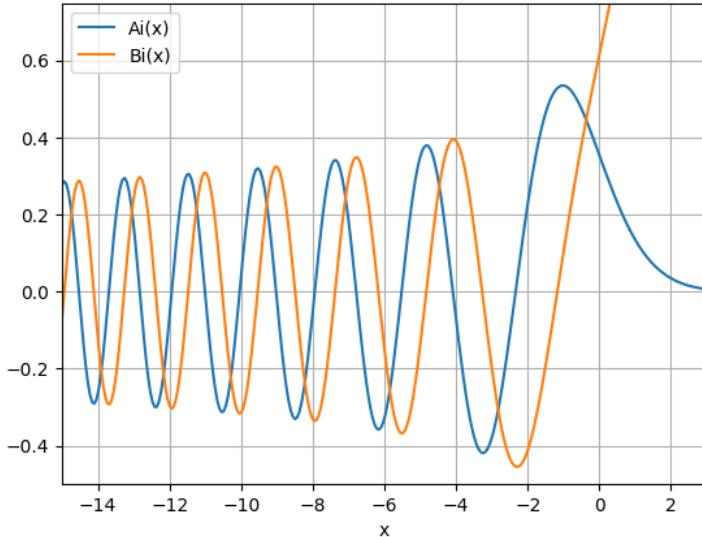
1 Uvod

Airyjevi funkciji A_i in B_i (slika 1) se v fiziki pojavljata predvsem v optiki in kvantni mehaniki. Definirani sta kot neodvisni rešitvi enačbe

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

in sta predstavljeni v integralski obliki

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



Slika 1: Graf Airyevih funkcij A_i in B_i za realne argumente. Za izračun funkcij je bil uporabljen integralski približek s simpsonovo metodo z mejami $t=0$, $t=50$ in korakom $dt = 5 \cdot 10^{-4}$

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

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

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

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

kjer je

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

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

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

s koeficienti

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

za velike pozitivne x izrazimo

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

za po absolutni vrednosti velike negativne x pa

$$\begin{aligned}\text{Ai}(x) &\sim \frac{1}{\sqrt{\pi}(-x)^{1/4}} \left[\sin(\xi - \pi/4) Q(\xi) + \cos(\xi - \pi/4) P(\xi) \right], \\ \text{Bi}(x) &\sim \frac{1}{\sqrt{\pi}(-x)^{1/4}} \left[-\sin(\xi - \pi/4) P(\xi) + \cos(\xi - \pi/4) Q(\xi) \right].\end{aligned}$$

Če želimo vrste natančno izračunati je potrebno zapisati člene vrst rekurzivno:

$$\begin{aligned}f_{n+1}(x) &= \frac{(1/3+n) 3 x^3}{(3n+3)(3n+2)(3n+1)} f_n(x) \\ g_{n+1}(x) &= \frac{(2/3+n) 3 x^3}{(3n+4)(3n+3)(3n+2)} g_n(x) \\ L_{n+1}(x) &= \frac{(5/2+3n)(3/2+3n)(1/2+3n)}{(n+1)(n+1/2) x^{54}} L_n(x) \\ P_{n+1}(x) &= \frac{(-1)(11/2+6n)(7/2+6n)(5/2+6n)(3/2+6n)(1/2+6n)}{(2n+3/2)(2n+1/2)(2n+2)(2n+1) x^2 54^2} P_n(x) \\ Q_{n+1}(x) &= \frac{(-1)(17/2+6n)(15/2+6n)(13/2+6n)(11/2+6n)(9/2+6n)(7/2+6n)}{(2n+3/2)(2n+5/2)(2n+3)(2n+2) x^2 54^2} Q_n(x)\end{aligned}$$

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} .

Dodatna naloga: Ničle funkcije Ai pogosto srečamo v matematični analizi pri določitvi intervalov ničel specjalnih 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

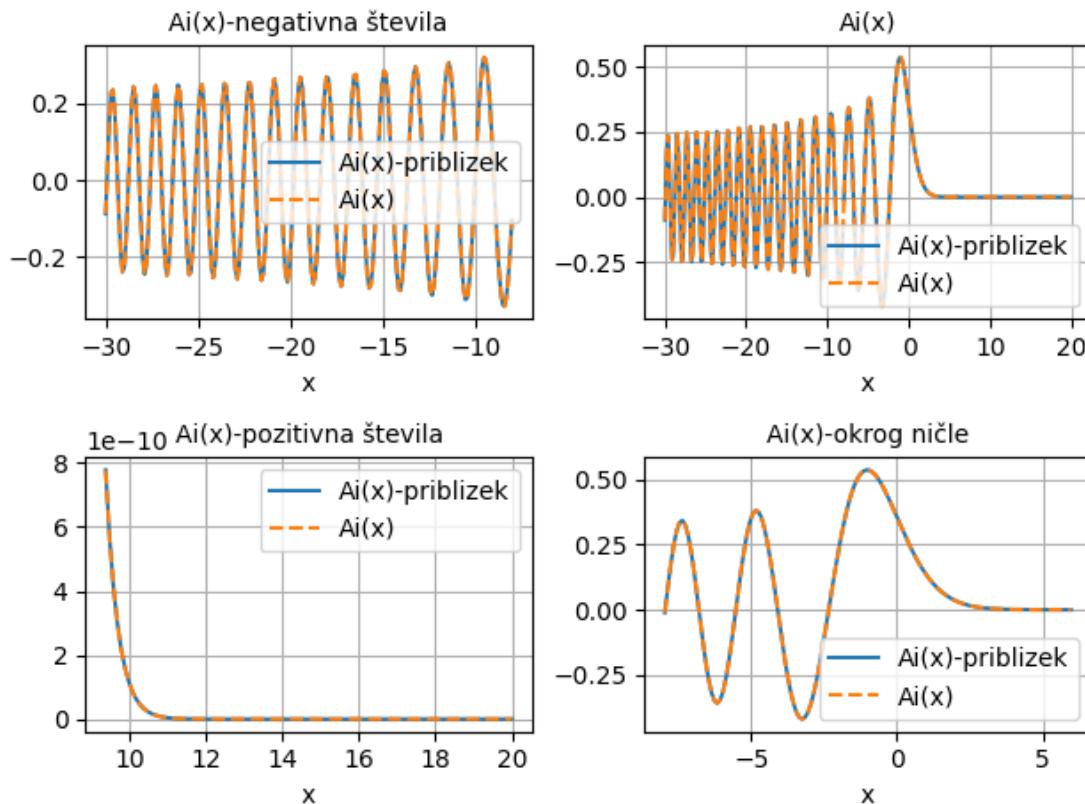
$$a_s = -f\left(\frac{3\pi(4s-1)}{8}\right), \quad b_s = -f\left(\frac{3\pi(4s-3)}{8}\right), \quad s = 1, 2, \dots,$$

kjer ima funkcija f asimptotski razvoj

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

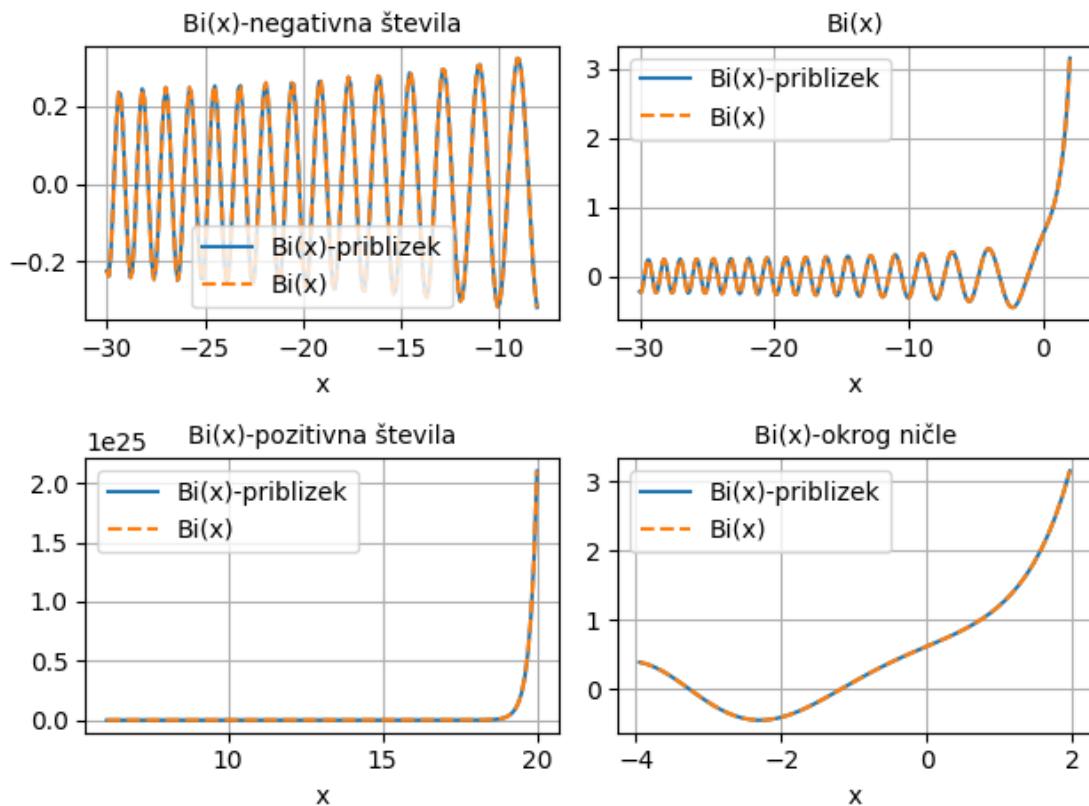
3 Rezultati

Z rekurzivnimi formulami sem izračunal vsote vrst, ki so potrebne za izračun funkcij $Ai(x)$ in $Bi(x)$. Vsoto sem ustavil ko je bil dodani člen k vrsti pod željeno natančnostjo $1e-10$. Če je več vrst tvorilo $Ai(x)$ oz. $Bi(x)$ lahko ta parameter še zmanjšamo, saj lahko obe prineseta napako na deseti decimalki. Če sta obe vrsti naraščajoči kot v primeru razvoja okrog $x=0$, sem vzel za zaustavljalni pogoj kar vsoto obeh dodatkov pod $1e-10$. Izračunane vrste je potrebno pomnožiti s konstantami oz. funkcijami da dobimo Airyjevi funkciji. Te prinesajo nekaj napake, ampak za našo natančnost ne vplivajo na končni rezultat. V Pythonu sem uporabljal knjižnico Numpy in Math in SciPy za izračun vrednosti želenih funkcij. Preizkusil sem tudi knjižnici Decimal in Mpmath ampak nisem opazil razlik v izračunih na 10 decimalki. Razlike se pojavijo nekje na 16 oz. 17 decimalnem mestu.

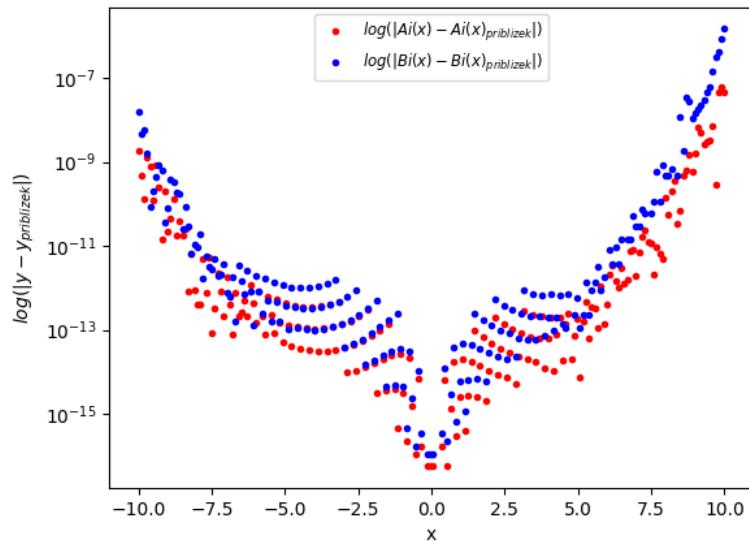


Slika 2: Graf Ai v različnih območjih za realne argumente. Za pravo vrednost $Ai(x)$ sem vzel Airyjevo funkcijo iz paketa SciPy.

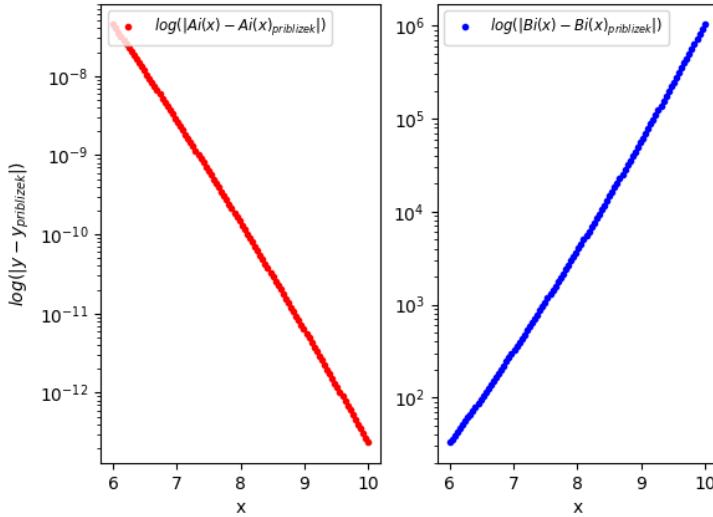
Iz slike 2 lahko vidimo, da se približek okrog $x = 0$ ujema samo za majhne vrednosti x-a. Približek za funkcijo $Bi(x)$ (slika 3) je prav tako okrog $x=0$ dober samo za majhne vrednosti argumenta. Težave nastanejo tudi pri asimptotskem približku za pozitivne vrednosti, saj začne $Bi(x)$ hitro naraščati. Da preverimo dejansko ujemanje je potrebno pogledati razlike med približkom in pravo vrednostjo po absolutni vrednosti (slike 4, 5, 6). Relativno natančnost približka prikazujejo slike 7, 8, 9.



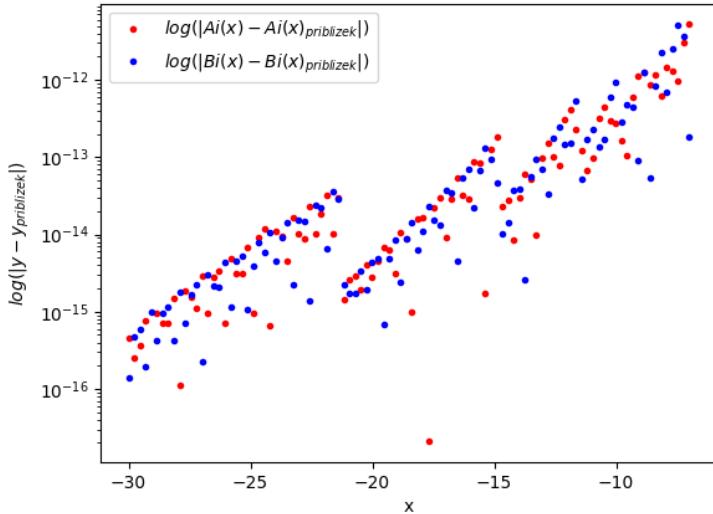
Slika 3: Graf Bi v različnih območjih za realne argumente. Za pravo vrednost $\text{Bi}(x)$ sem vzel Airyjevo funkcijo iz paketa SciPy.



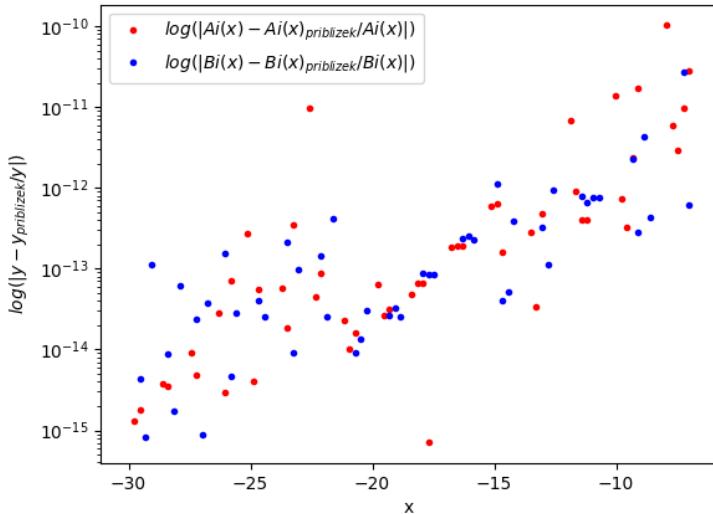
Slika 4: Razlika med pravo vrednostjo in približkom po absolutni vrednosti za obe Airyjevi funkciji. Približek se ujema za $|x| = 8$.



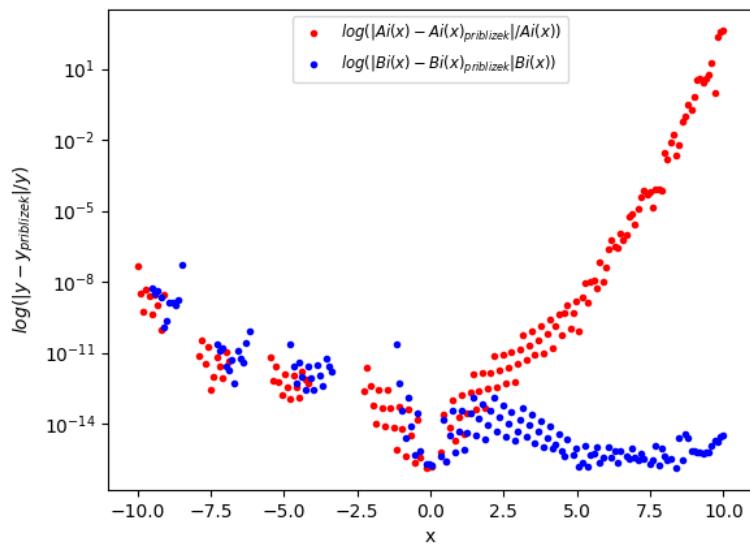
Slika 5: Graf razlike približka in prave vrednosti po absolutni vrednosti za $Ai(x)$ in $Bi(x)$. $Ai(x)$ začne padati proti ničli in se njena napaka manjša. $Bi(x)$ pa začne divergirati in približek postane slab. Sklepam da potem postanejo velike razlike zaradi natančnosti zapisa velikih števil.



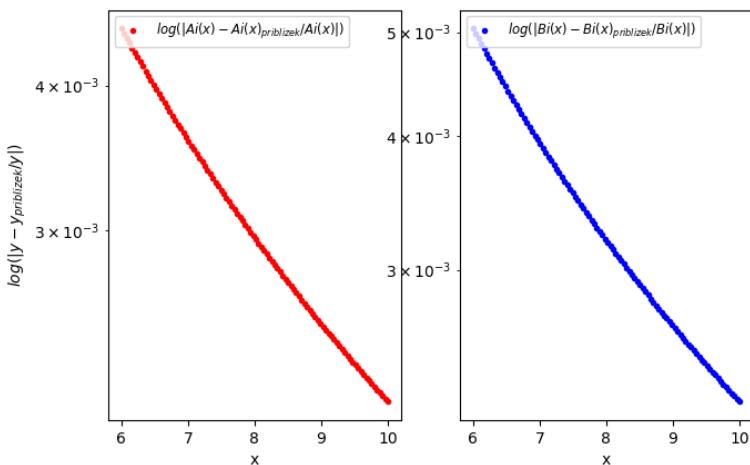
Slika 6: Graf razlike približka in prave vrednosti po absolutni vrednosti za $Ai(x)$ in $Bi(x)$. Vidimo da napaka pada ko se x manjša.



Slika 7: Graf relativne razlike približka in prave vrednosti $Ai(x)$ in $Bi(x)$ za negativne vrednosti argumenta.



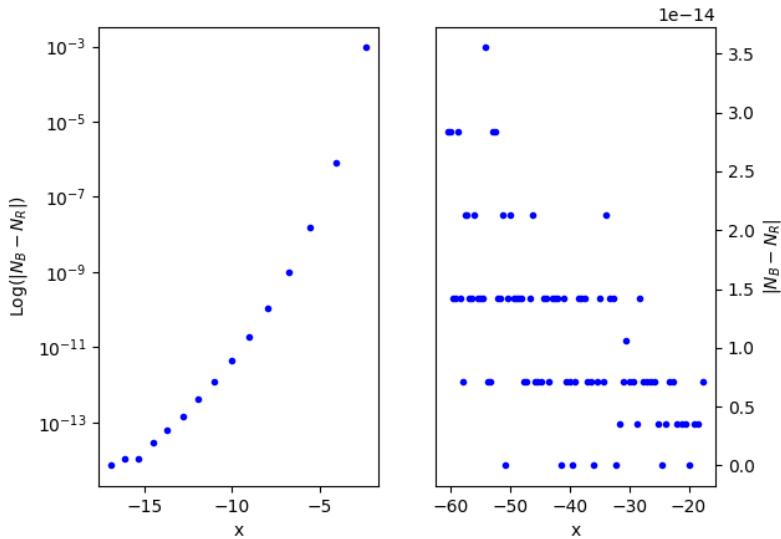
Slika 8: Graf relativne razlike približka in prave vrednosti $Ai(x)$ in $Bi(x)$ za vrednosti argumenta okrog $x=0$.



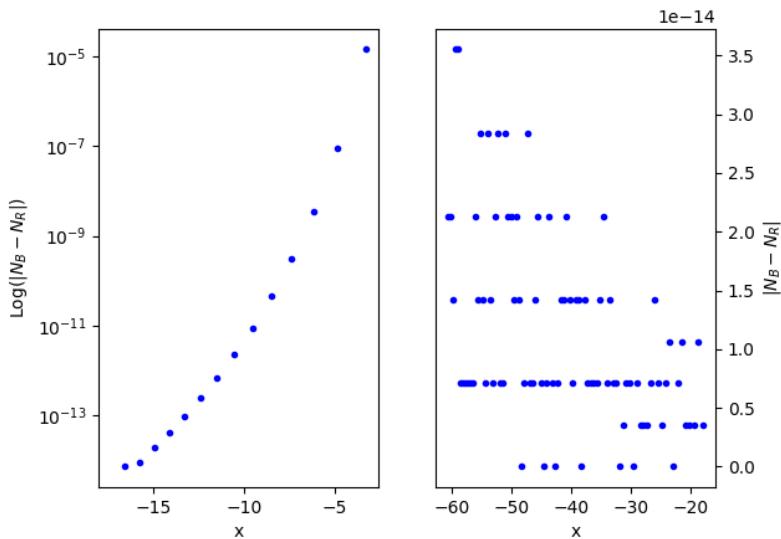
Slika 9: Graf relativne razlike približka in prave vrednosti $Ai(x)$ in $Bi(x)$ za pozitivne vrednosti argumenta. Maksimalna natančnost je nekaj manj kot 10^{-3} .

3.1 Dodatna naloga

Nicelje funkcije sem poiskal z bisekcijo. Območje iskanja sem razdelil na majhne intervale. Najprej je funkcija preverila če je na danem intervalu ničla, če je ni bilo je nadaljevala z drugim intervalom. Če se je na intervalu nahajala ničla je funkcija uporabila bisekcijsko metodo. Smiselno se mi je zdelo prikazati razliko med pravo ničlo funkcije in ničlo, ki jo je našla bisekcijska metoda (slika 10 in 11). Natančnost bisekcije se manjša ko se bliža izhodišču.



Slika 10: Graf absolutne razlike med pravo ničlo (N_R) in ničlo najdeno z bisekcijsko metodo (N_B) za funkcijo $Ai(x)$.



Slika 11: Graf absolutne razlike med pravo ničlo (N_R) in ničlo najdeno z bisekcijsko metodo (N_B) za funkcijo $Bi(x)$.

4 Zaključek

Naloga je zahtevala kar nekaj dela. Za pridobiti natančne rezultate je potrebno veliko truda in pri nalogi vidimo, da nimamo rezultata pri željeni natančnosti na celotni realni osi, ampak samo na določenih intervalih. Pri natančnih izračunih je dobro vedeti kako si računalnik zapisuje števila in računa z njimi, saj lahko drugače pride do napačnih rezultatov in posledično napačnih interpretacij.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM
2. naloga: Naključni sprehodi

Žiga Šinigoj, 28191058

Ljubljana, oktober 2021

1 Uvod

Naključni sprehodi so vrsta gibanja, pri katerem v velikem številu korakov napredujemo iz izhodišča v neko končno lego, tako da se parametri vsakega naslednjega koraka sproti naključno določajo. Običajni zgled je Brownovo gibanje (difuzija) drobnih delcev barvila po mirujoči homogeni tekočini, kjer je spočetka barvilo zbrano v izhodišču. Zanimive je opazovati naključne sprehode, pri katerih dovolimo nadpovprečno dolge korake. Verjetnostno gostoto porazdelitve po dolžinah posameznih korakov parametrizirajmo v potenčni obliki

$$p(l) \propto l^{-\mu}, \quad (1)$$

kjer naj bo $1 < \mu < 3$. Tedaj postane drugi moment porazdelitve

$$\langle l^2 \rangle = \int l^2 p(l) dl$$

neskončen. Govorimo o anomalni difuziji, prisotni pri celi družini kinematičnih distribucij dolžin poti z "debelimi repi".

Ustrezno sliko naključnega gibanja, povezanega s temi dolgimi koraki, lahko interpretiramo na dva načina:

- Lévyjev pobeg oz. polet (*flight*), implicira, da vsak korak iz porazdelitve (1) traja enako dolgo, medtem ko se hitrost gibanja med koraki (*divje*) spreminja.
- Lévyjev sprehod (*walk*), ki interpretira korak iz porazdelitve (1) kot gibanje s konstantno hitrostjo in tako koraki trajajo različno dolgo časa (dolžina koraka je sorazmerna s časom).

V prvem primeru (pobeg, flight) je pretečeni čas direktno sorazmeren s številom korakov, v drugem primeru (sprehod, walk) pa je pretečeni čas sorazmeren z vsoto dolžine korakov.

Pri anomalni difuziji razmazanost (varianca) velike množice končnih leg naključnih Lévyjevih **sprehodov (walks)** narašča z drugačno potenco časa. Velja $\sigma^2(t) \sim t^\gamma$, kjer je

$$\begin{aligned} 1 < \mu < 2, \quad \gamma &= 2 && \text{(balistični režim)}, \\ 2 < \mu < 3, \quad \gamma &= 4 - \mu && \text{(super-difuzivni režim)}, \\ \mu > 3, \quad \gamma &= 1 && \text{(normalna difuzija)}. \end{aligned}$$

Za $\mu = 2$ pričakujemo $\sigma^2(t) \sim t^2 / \ln t$, za $\mu = 3$ pa $\sigma^2(t) \sim t \ln t$. Slika je nekoliko drugačna pri opazovanju naključnih Lévyjevih **poletov (flights)**. Spet vzamemo zvezo $\sigma^2(t) \sim t^\gamma$ in dobimo odvisnosti

$$\begin{aligned} 1 < \mu < 3, \quad \gamma &= \frac{2}{\mu - 1} && \text{(super-difuzivni režim)}, \\ \mu > 3, \quad \gamma &= 1 && \text{(normalna difuzija)}. \end{aligned}$$

Pri $\mu = 2$ očitno pričakujemo $\sigma^2(t) \sim t^2$, torej balistični režim.

Statistični komentar: v primerih, ko je drugi moment porazdelitve neskončen, bo tudi račun razmazanosti končnih leg x_n v obliki

$$\sigma^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \langle x \rangle)^2 \quad (2)$$

divergiral oziroma bo imel ob ponovnih zagonih naključnega sprehoda močno raztresene vrednosti. Ena izmed možnosti je na primer MAD, "median absolute deviation"

$$\text{MAD} \equiv \text{median}_i (|X_i - \text{median}_j X_j|).$$

Z njo merimo povprečje absolutne vrednosti deviacije na način, ki je zelo malo občutljiv na oddaljene vrednosti v repih porazdelitve, saj te vrednosti na račun mediane bistveno manj vplivajo kot na račun običajne povprečne vrednosti.

2 Naloga

Napravi računalniško simulacijo dvorazsežne naključne hoje za **polete in sprehode**. Začni vedno v izhodišču ($x = y = 0$), nato pa določi naslednjo lego tako, da naključno izbereš smer koraka in statistično neodvisno od te izbire še njegovo dolžino, torej

$$\begin{aligned} x &\leftarrow x + l \cos \varphi, \\ y &\leftarrow y + l \sin \varphi, \end{aligned}$$

kjer je φ enakomerno naključno porazdeljen po intervalu $[0, 2\pi]$, dolžina koraka l pa naj bo porazdeljena v skladu s potenčno obliko (Enačba 1). Dolžine l_i je v tem primeru potrebno generirati po verjetnostni porazdelitvi $w(l) \sim p(l)$ (Enačba 1). Za izračun algoritma je osnova naslednja formula:

$$\int_a^l w(t) dt = \rho \cdot \int_a^b w(t) dt, \quad (3)$$

ki jo je potrebno rešiti in iz nje izraziti spremenljivko l . Tu je ρ (psevdo-)naključno število na intervalu $[0, 1]$ ter je $[a, b]$ relevantni interval vzorčenja. Za nekatere porazdelitve je izračun preprost, npr $w(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}}$ nam da kar:

$$l = -\tau \ln(1 - \rho). \quad (4)$$

Dodatna naloga: Naključno spreminjaš še čas, ko delec pred naslednjim korakom miruje (s tako dodatno prostostno stopnjo poskušamo modelirati tako imenovani “sticking time” ali “trapping time” pri anomalni difuziji elektronov v amorfnih snoveh). Ustrezna verjetnostna gostota naj ima potenčno odvisnost

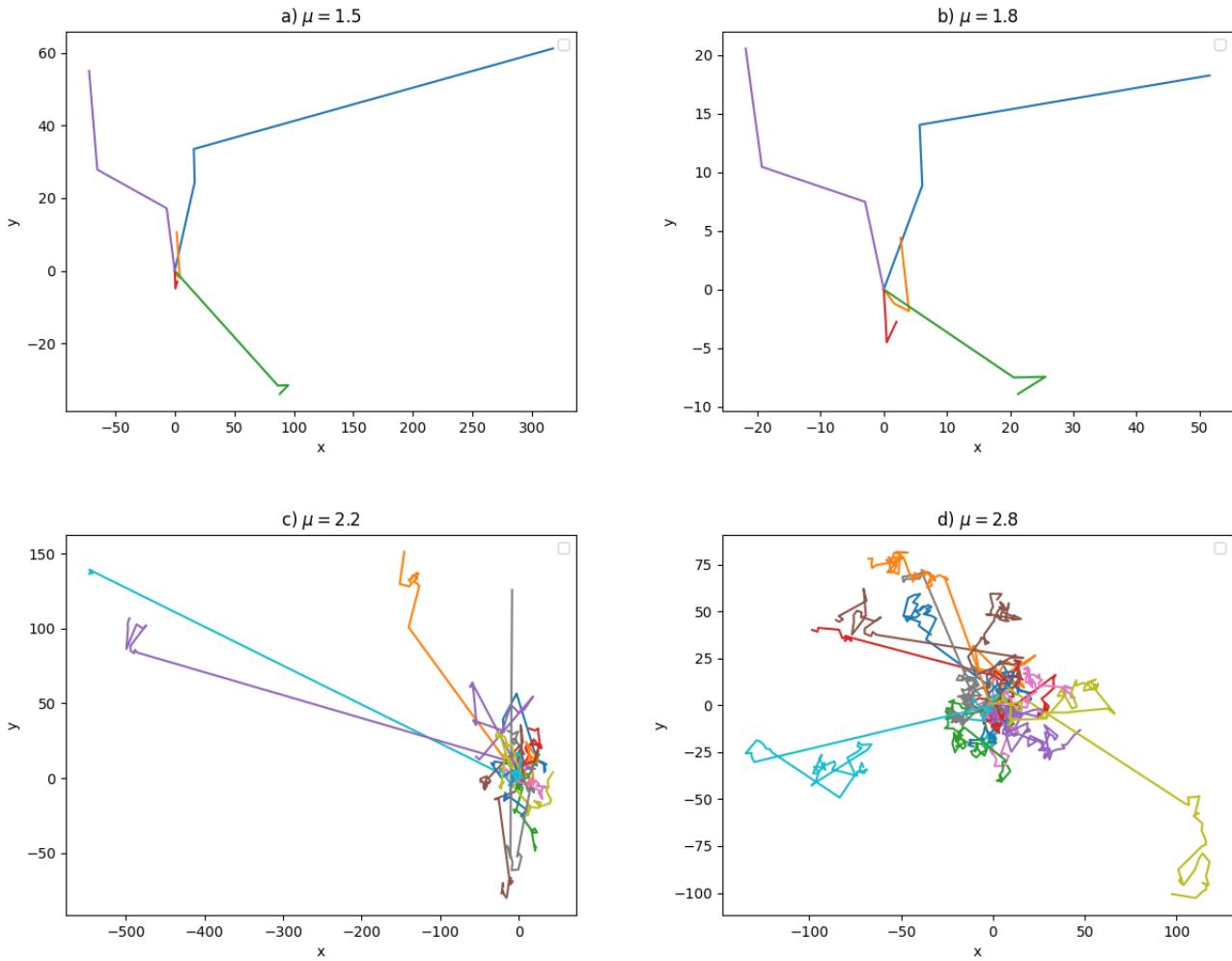
$$p(t) \propto t^{-\nu},$$

kjer $1 < \nu < 2$. Je ta odvisnost razklopljena od porazdelitve osnovnega naključnega sprehoda po dolžinah (ozioroma časih) posameznih korakov?

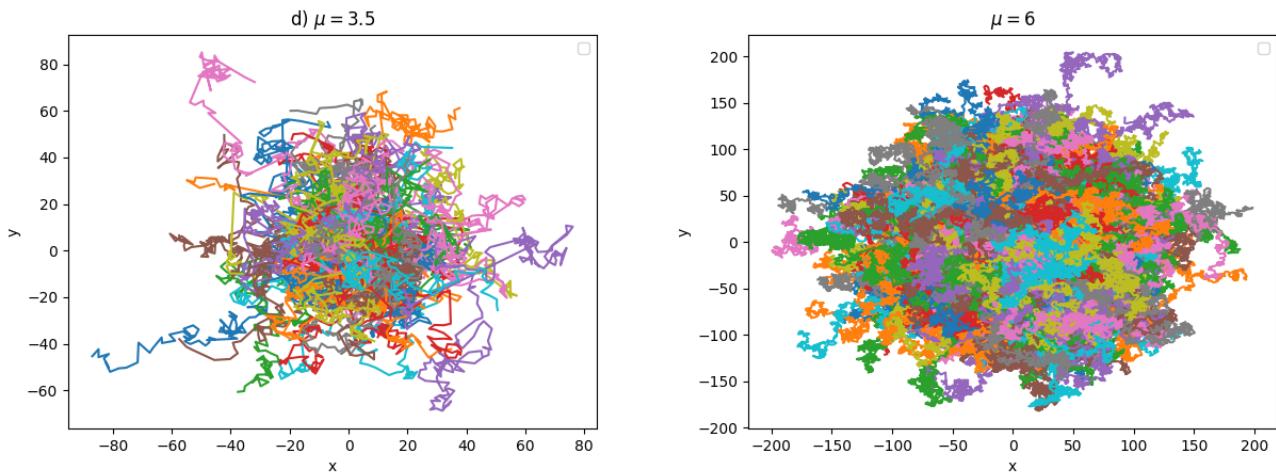
3 Rezultati

3.1 Sprehodi

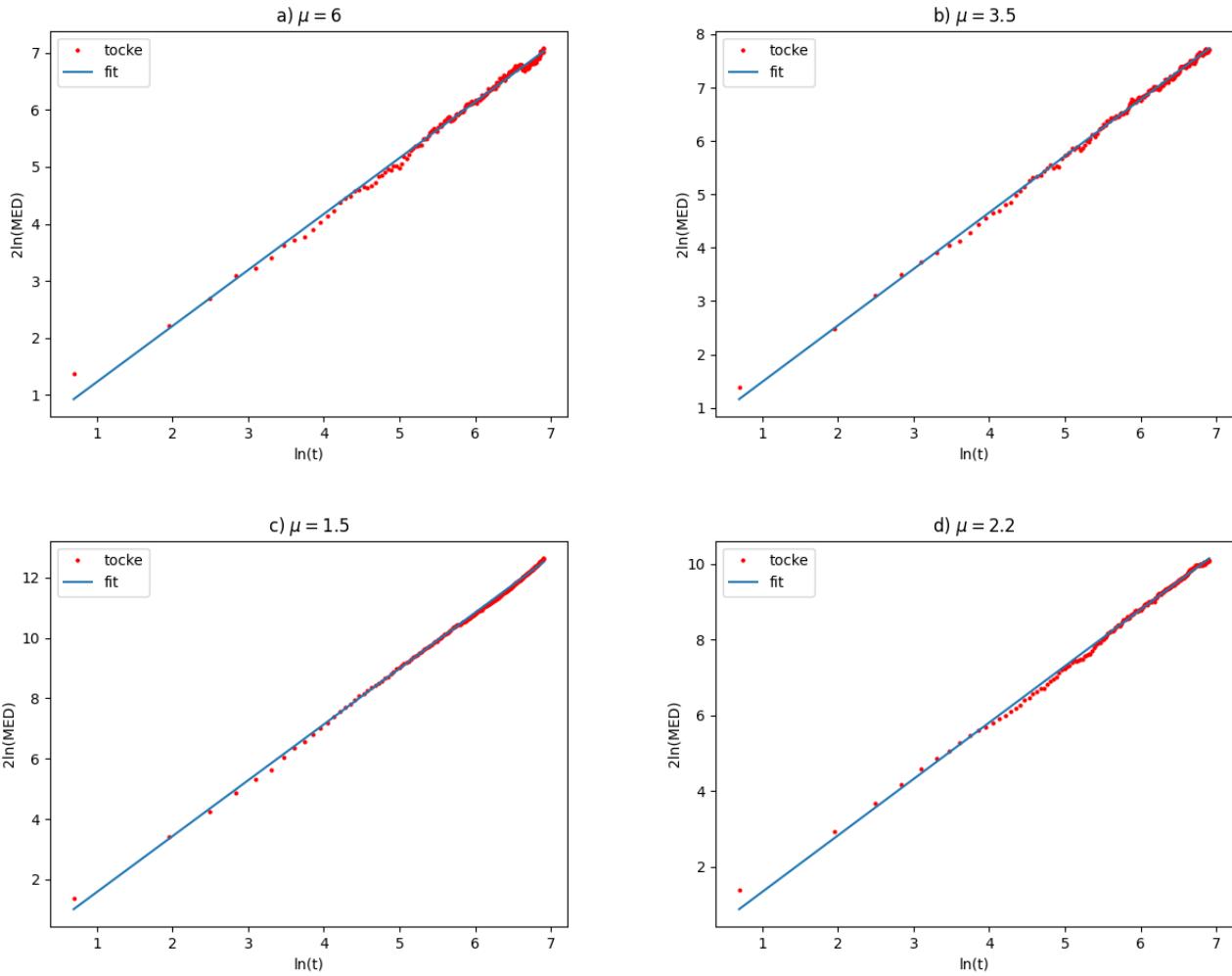
Pri sprehodih sem štel čas kar kot $t_{korak} = l_{korak}$ oz. $t_{cel} = l_{cel}$, kjer je l_{cel} celotna pot delca in t_{cel} celotni čas, ki ga potrebuje za dano pot. Vrednosti kota φ sem generiral iz enakomerne porazdelitve, dolžino koraka l pa s Paretovo porazdelitvijo *pareto.random* s parametrom $m = 2$ iz knjižnice NumPy. Nekaj primerov sprehodov prikazuje slika 1 in 2. Potrebno je določiti tudi γ faktor, ki pove kako se gibljejo delci. Zanima nas naklon premice v logaritemski skali. Točke ob izbranih časih, ki jih mogoče nisem imel, sem interpoliral z linearno interpolacijo. Za izračun MED sem uporabil funkcijo *stats.median_abs_deviation* iz paketa SciPy. Za izračun parametra γ sem uporabil 500 delcev s 500 koraki. Napako parametra γ sem ocenil iz diagonalnih elementov kovariančne matrike, ki jo da metoda *optimize.curve_fit* iz paketa SciPy. Vrednosti parametra γ določene iz naklonov premic se precej dobro ujemajo z napovedanimi vrednostimi in rezimi iz teorije.



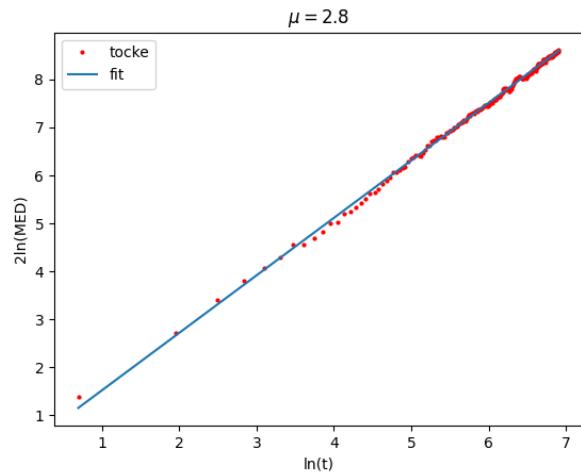
Slika 1: a) 5 delcev, 4 koraki, pri vrednosti $\mu = 1.5$ smo v balističnem režimu in delci hitro podivjajo. b) 5 delcev, 4 koraki, delci ne tako hitro pobegnejo kot pri $\mu = 1.5$, ampak smo še vedno v balističnem režimu. c) 20 delcev, 20 korakov v super-difuzivnem režimu, delci difundirajo počasneje kot v balističnem režimu in hitreje kot pri normalni difuziji. d) 20 delcev, 50 korakov, delci so bližje režimu normalne difuzije.



Slika 2: d) 100 delcev, 100 korakov, pri vrednosti $\mu = 3.5$ se nahajamo v režimu normalne difuzije in delci se počasi premikajo radialno navzven. Desna slika: 1000 delcev, 1000 korakov, pri vrednosti $\mu = 6$, povprečna dolžina koraka je najmanjša in povprečna lega delca se premika še počasneje kot na levi slikah.



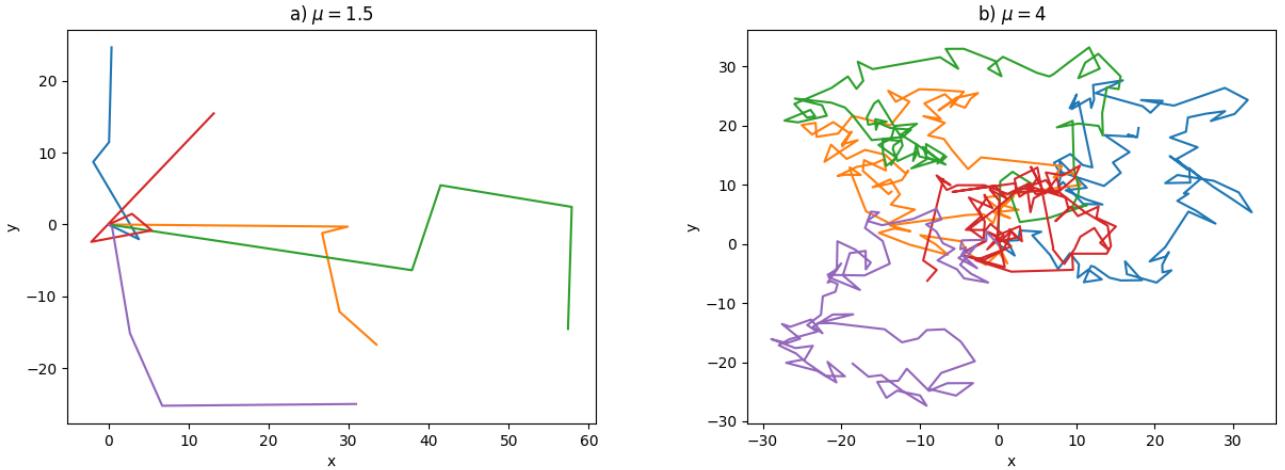
Slika 3: Logaritem cenilke Median Absolute Deviation v odvisnosti od logaritma časa. Iz naklona premice dobim vrednosti iskanega parametra γ . a) $\gamma = 0.982 \pm 0.005$, b) $\gamma = 1.0578 \pm 0.003$, c) $\gamma = 1.851 \pm 0.004$, d) $\gamma = 1.492 \pm 0.005$.



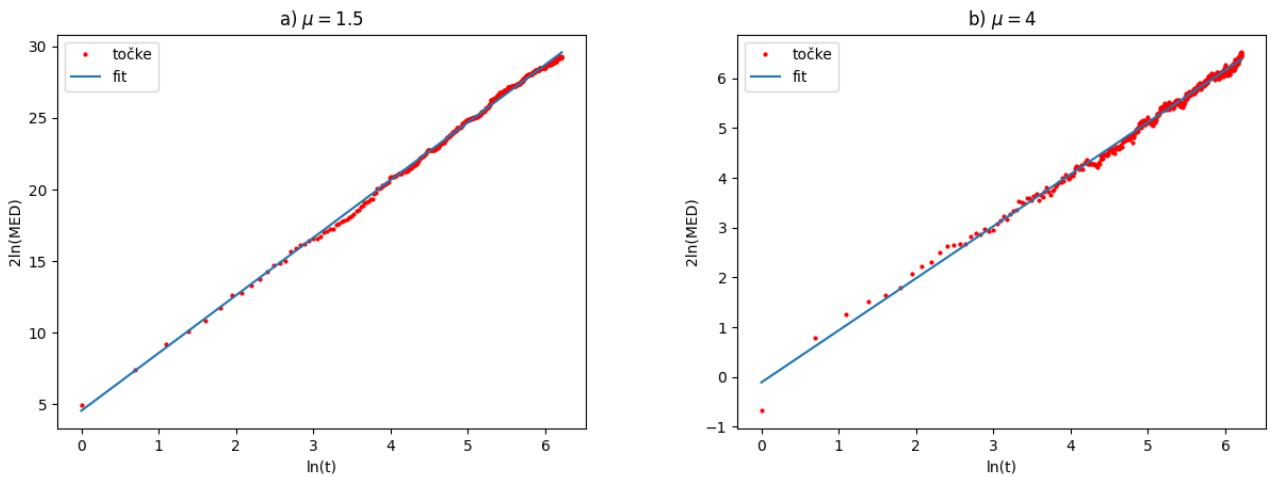
Slika 4: Logaritem cenilke Median Absolute Deviation v odvisnosti od logaritma časa. $\gamma = 1.198 \pm 0.003$.

3.2 Poleti

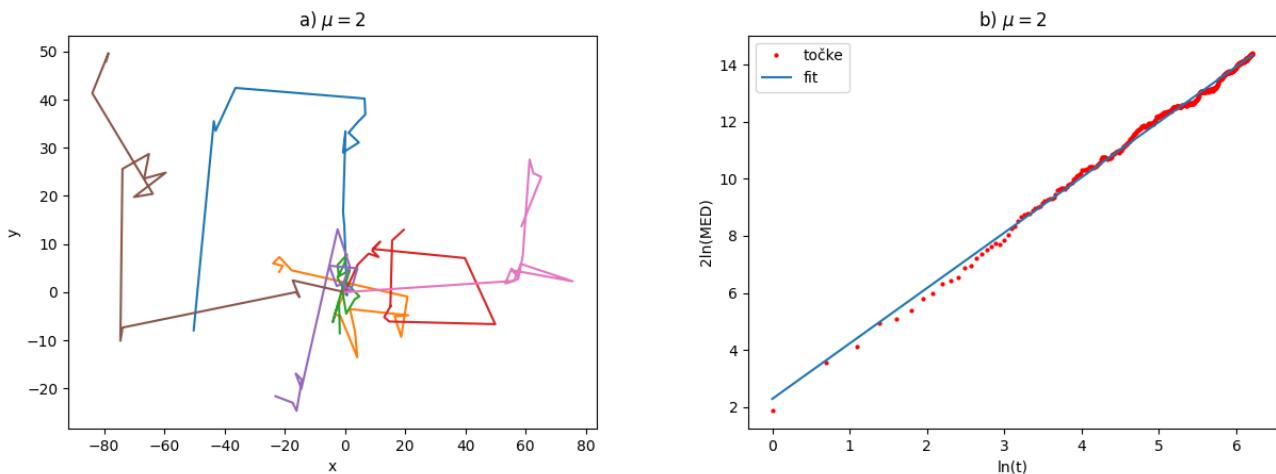
Pri poletih sem štel čas kot dolžino korakov, zaradi tega ni bilo potrebno interpolirati točk, saj sem imel koordinate vseh točk ob enakih časih. Dinamika premikanja je enaka kot pri sprehodih, le štetje časa je drugačno. Za izračun parametra γ sem prav tako uporabil 500 delcev s 500 koraki. Ocenjene napake in fiti so določeni po enakem postopku kot pri sprehodi. Vrednosti parametra γ določene iz naklonov premic se precej dobro ujemajo z napovedanimi vrednostimi in režimi iz teorije.



Slika 5: a) 5 delcev, 5 korakov v super-difuzivnem režimu. Povprečna lega delcev se hitro spreminja. b) 5 delcev, 100 korakov v režimu normalne difuzije. Povprečna lega delcev se počasneje spreminja kot v super-difuzivnem režimu.



Slika 6: Logaritem cenzilke Median Absolute Deviation v odvisnosti od logaritma časa. Iz naklona premice dobim vrednosti iskanega parametra γ . a) $\gamma = 4.03 \pm 0.01$, b) $\gamma = 1.046 \pm 0.004$.



Slika 7: a) 7 delcev, 15 sprehodov v balističnem režimu. b) Logaritem cenilke Median Absolute Deviation v odvisnosti od logaritma časa. $\gamma = 1.942 \pm 0.006$. Vrednost fita t^γ se ujema z napovedjo t^2 .

4 Zaključek

Z naključnimi sprehodi lahko napovemo gibanje delcev v raznih snoveh in s tem lahko boljše razumemo fizikalne pojave. Vrednosti iskanega eksponenta se ujemajo z teorijo precej natančno. Če bi vzel več delcev pri fitanju premice, bi lahko dobil še natančnejšo vrednost γ . Za simulacijo delcev je potrebno veliko operacij in če bi hotel simulirati pravo število delcev v neki snovi bi potreboval boljši računalnik in optimizirano kodo.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

3. naloga: Anharmonski oscilator

Žiga Šinigoj, 28191058

Ljubljana, oktober 2021

1 Uvod

Enodimensionalni linearни harmonski oscilator (delec mase m s kinetično energijo $T(p) = p^2/2m$ v kvadratičnem potencialu $V(q) = m\omega^2q^2/2$) opišemo z brezdimenzijsko Hamiltonovo funkcijo

$$H_0 = \frac{1}{2} (p^2 + q^2) ,$$

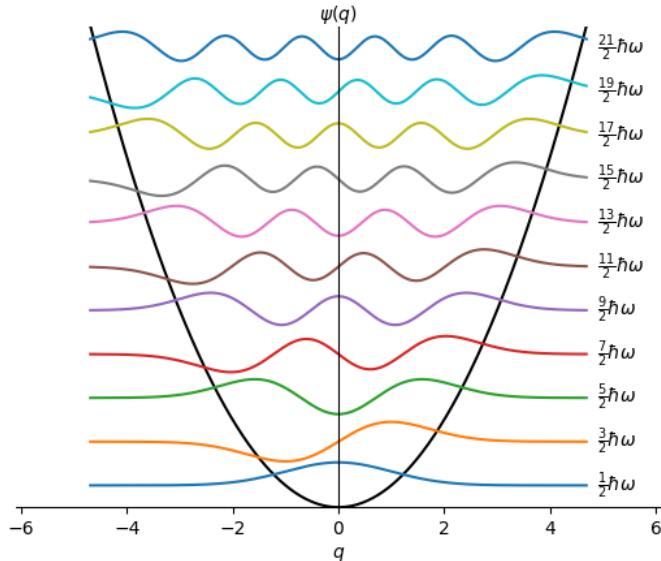
tako da energijo merimo v enotah $\hbar\omega$, gibalne količine v enotah $(\hbar m\omega)^{1/2}$ in dolžine v enotah $(\hbar/m\omega)^{1/2}$. Lastna stanja $|n\rangle$ nemotenega Hamiltonovega operatorja H_0 poznamo iz osnovnega tečaja kvantne mehanike [Strnad III]: v koordinatni reprezentaciji so lastne valovne funkcije

$$|n\rangle = (2^n n! \sqrt{\pi})^{-1/2} e^{-q^2/2} \mathcal{H}_n(q) ,$$

kjer so \mathcal{H}_n Hermitovi polinomi. Lastne funkcije zadoščajo stacionarni Schrödingerjevi enačbi

$$H_0 |n^0\rangle = E_n^0 |n^0\rangle$$

z nedegeneriranimi lastnimi energijami $E_n^0 = n + 1/2$ za $n = 0, 1, 2, \dots$. Matrika $\langle i | H_0 | j \rangle$ z $i, j = 0, 1, 2, \dots, N - 1$ je očitno diagonalna, z vrednostmi $\delta_{ij}(i + 1/2)$ po diagonali. Lastne funkcije so tako Nemoteni Hamiltonki dodamo anharmonski člen



Slika 1: Lastne funkcije harmonskega oscilatorja pri lastnih vrednostih energije.

$$H = H_0 + \lambda q^4 .$$

Kako se zaradi te motnje spremenijo lastne energije? Iščemo torej matrične elemente $\langle i | H | j \rangle$ motenega Hamiltonovega operatorja v bazi nemotenih valovnih funkcij $|n^0\rangle$, kar vemo iz perturbacijske teorije v najnižjem redu. Pri računu si pomagamo s pričakovano vrednostjo prehodnega matričnega elementa za posplošeno koordinato

$$q_{ij} = \langle i | q | j \rangle = \frac{1}{2} \sqrt{i + j + 1} \delta_{|i-j|,1} ,$$

ki, mimogrede, uteleša izbirno pravilo za električni dipolni prehod med nivoji harmonskega oscilatorja. V praktičnem računu moramo seveda matriki q_{ij} in $\langle i | H | j \rangle$ omejiti na neko končno razsežnost N . Namesto da računamo matrične elemente $q_{ij} = \langle i | q | j \rangle$ in perturbacijsko matriko razumemo kot $[q_{ij}]^4$, bi lahko računali tudi matrične elemente kvadrata koordinate

$$q_{ij}^{(2)} = \langle i | q^2 | j \rangle$$

in motnjo razumeli kot kvadrat ustreznih matrik,

$$\lambda q^4 \rightarrow \lambda \left[q_{ij}^{(2)} \right]^2 ,$$

ali pa bi računali matrične elemente četrte potence koordinate

$$q_{ij}^{(4)} = \langle i | q^4 | j \rangle$$

in kar to matriko razumeli kot motnjo,

$$\lambda q^4 \rightarrow \lambda \left[q_{ij}^{(4)} \right] .$$

Potrebujemo še rekurzivni zvezi:

$$\langle i | q^2 | j \rangle = \frac{1}{2} \left[\sqrt{j(j-1)} \delta_{i,j-2} + (2j+1) \delta_{i,j} + \sqrt{(j+1)(j+2)} \delta_{i,j+2} \right]$$

ter

$$\begin{aligned} \langle i | q^4 | j \rangle = \frac{1}{2^4} \sqrt{\frac{2^i i!}{2^j j!}} & \left[\delta_{i,j+4} + 4(2j+3) \delta_{i,j+2} + 12(2j^2 + 2j + 1) \delta_{i,j} \right. \\ & \left. + 16j(2j^2 - 3j + 1) \delta_{i,j-2} + 16j(j^3 - 6j^2 + 11j - 6) \delta_{i,j-4} \right], \end{aligned}$$

2 Naloga

Z diagonalizacijo poišči nekaj najnižjih lastnih vrednosti in lastnih valovnih funkcij za moteno Hamiltonko $H = H_0 + \lambda q^4$ ob vrednostih parametra $0 \leq \lambda \leq 1$. Rešujemo torej matrični problem lastnih vrednosti

$$H |n\rangle = E_n |n\rangle .$$

Nove (popravljene) valovne funkcije $|n\rangle$ so seveda linearna kombinacija starih (nemotenih) valovnih funkcij $|n^0\rangle$. Matrike velikosti do $N = 3$ ali $N = 4$ lahko za silo diagonaliziramo paš; za diagonalizacijo pri večjih N uporabi enega ali več numeričnih postopkov, na primer rutine `tred2` in `tqli` iz zbirke Numerical Recipes ali iz kakega drugega vira (npr Python). Vsaj enega izmed postopkov izvedi 'ročno' (sprogramiraj, uporabi izvorno kodo). Preveri, da v limiti $\lambda \rightarrow 0$ velja $E_n \rightarrow E_n^0$ (če ne velja, je verjetno nekaj narobe s programom). Raziski, kako so rezultati odvisni od razsežnosti N matrik H_0 oziroma q^4 . Kakšna je konvergenca lastnih vrednosti pri velikih N ? Kakšne so razlike med naštetimi tremi načini izračuna lastnih vrednosti in funkcij?

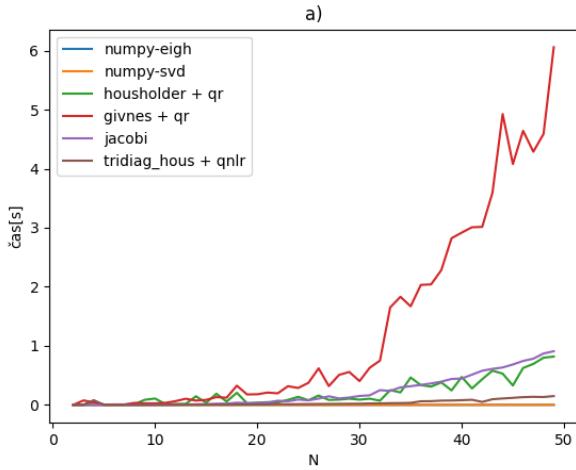
Dodatna naloga: Poišči še nekaj najnižjih lastnih energij in lastnih funkcij za problem v potencialu z dvema minimumoma

$$H = \frac{p^2}{2} - 2q^2 + \frac{q^4}{10} .$$

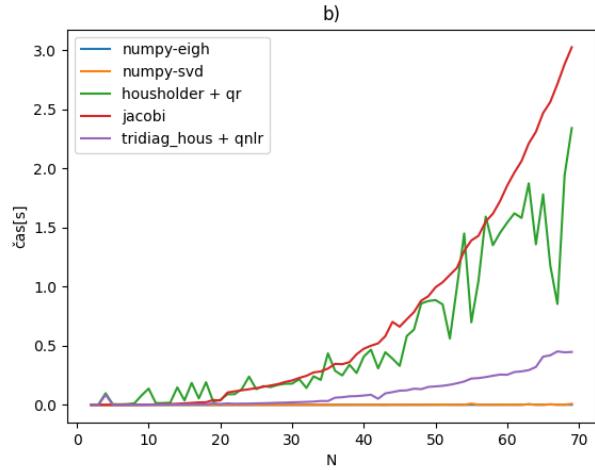
3 Rezultati

3.1 Primerjava metod za diagonalizacijo

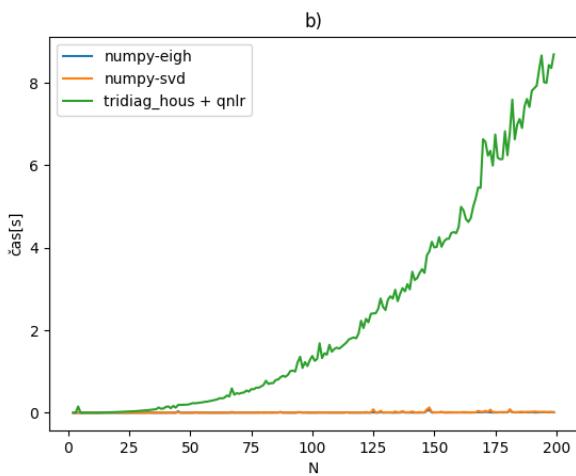
Primerjal sem vgrajeni metodi `svd` in `eigh` iz knjižnice `linalg` programskega paketa NumPy in pa metode iz predavanj. Sam sem dodal še Jacobijev metodo. Givensove rotacije in Householderjeva zrcaljenja sem uporabil za QR razcep. Z QR iteracijo pa potem do lastnih vrednosti. Ker je časovna zahtevnost QR razcepa reda n^3 , je bolje če matriko spravimo v tridiagonalno in potem nadaljujemo z QR iteracijo. Tukaj sem uporabil funkciji iz predavanj. Meril sem časovno zahtevnost in natančnost, kjer sem vzel lastne vrednosti metode `eigh` za prave. Za parameter λ sem vzel vrednost $\lambda = 0.2$.



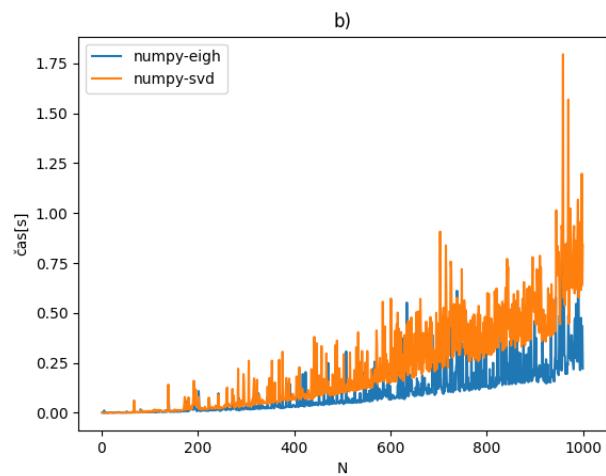
Slika 2: Časovna zahtevnost algoritmov v odvisnosti velikosti kvadratne matrike N.



Slika 3: Časovna zahtevnost algoritmov v odvisnosti velikosti kvadratne matrike N.

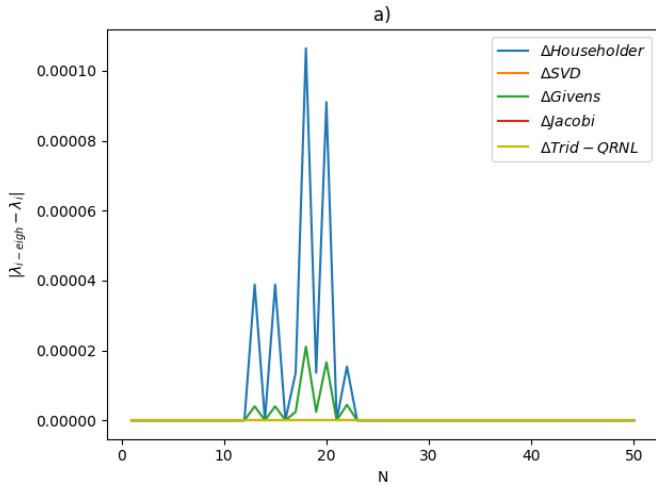


Slika 4: Časovna zahtevnost algoritmov v odvisnosti velikosti kvadratne matrike N.

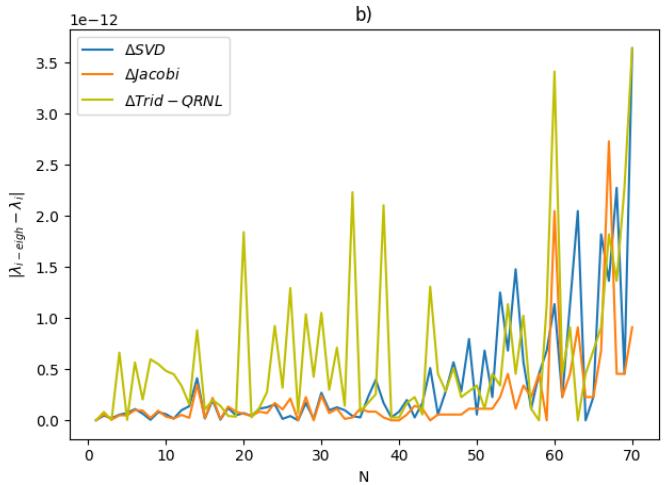


Slika 5: Časovna zahtevnost algoritmov v odvisnosti velikosti kvadratne matrike N.

Metode, ki so implementirane brez optimizacij (moja in iz predavanj) so veliko počasnejše in se ne morejo primerjati z vgrajenimi. Vidimo lahko da je najpočasnejša Givensova, nato Householder in Jacobijeva metoda. Tridiagonalna metoda z QR razcepom je boljša, mogoče bi bilo dobro če bi dodal še vgrajeno metodo za diagonalizacijo tridiagonalnih matrik, ki bi v teoriji morala biti najhitrejša. Za zaustavljalni pogoj pri QR iteraciji sem vzel takrat, ko se lastne vrednosti na diagonali približajo lastnim vrednostim funkcije poljubno natančno ($1e-5$) oz. ko dosežejo maksimalno število iteracij (200). Hitrost je seveda odvisna tudi od teh dveh parametrov. Za merjenje izvajanja funkcije sem uporabljal *default_timer* iz knjižnice *timeit*. Natančnost lastnih vrednosti prikazujejo slike 6 in 7. Pri primerjavah nisem upošteval konvergencije zaradi hilbertovega prostora in neskončnih lastnih funkcij, zato lastne vrednosti po metodi *eigh* lahko odstopajo od pravih lastnih vrednosti. Gre samo za primerjavo natančnosti metod. Slika 7 je pričakovana, odstopanja, ko se bližamo večjim lastnim vrednostim. Na sliki 6 pa nisem ravno prepričan zakaj so vrhi samo pri določenih vrednostih. Kvečjemu bi taka odstopanja pričakoval pri lastnih vrednostih blizu $N = 50$.



Slika 6: Odsopanje lastne vrednosti od *prave* pri N-ti lastni vrednosti.

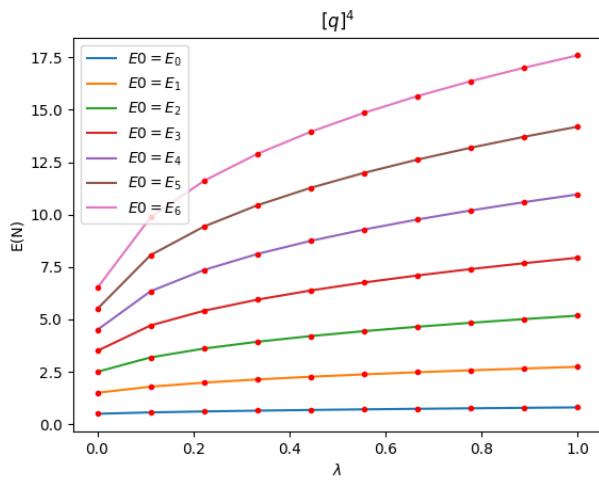


Slika 7: Odsopanje lastne vrednosti od *prave* pri N-ti lastni vrednosti.

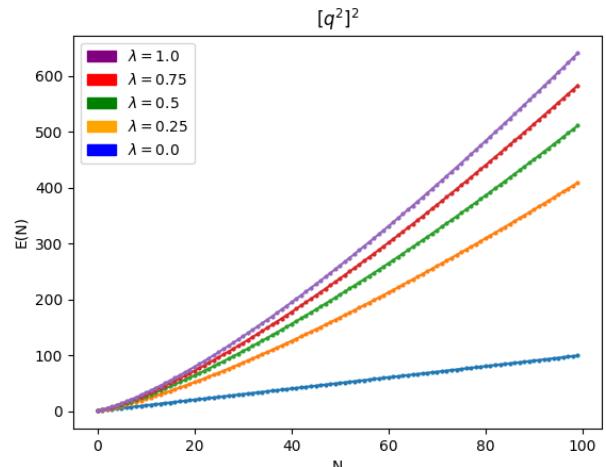
3.2 Anharmonski oscilator

Za nadaljno analizo in primerjavo sem uporabljal vgrajeno metodo `eigh`. Najprej lahko pogledamo energije v odvisnosti od parametra λ (slika 16, 17).

Lastne vrednosti so odvisne od velikosti matrike. Ker smo v Hilbertovem prostoru bi v splošnem morala

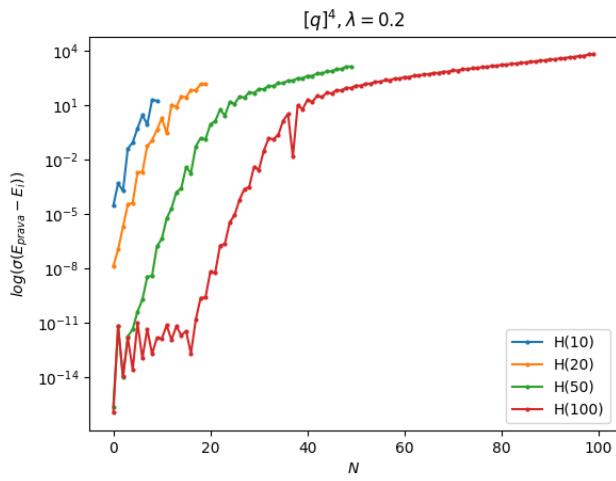


Slika 8: Prvih 7 lastnih energij v odvisnosti od para- metra λ , izračunane na način četrte potence koordinate.

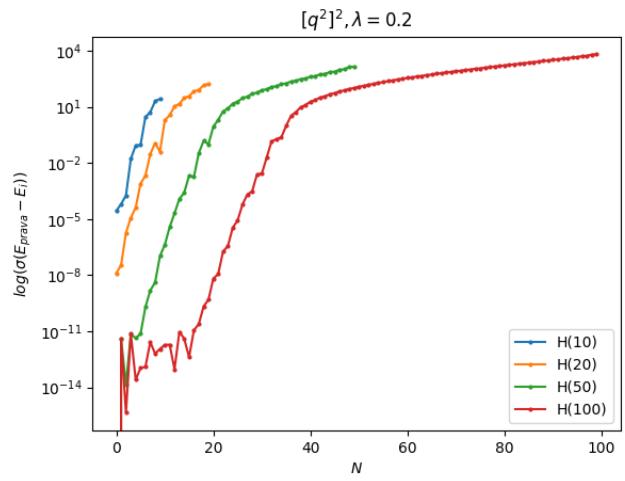


Slika 9: Prvih 100 lastnih energij sistema pri različnih vrednostih parametra λ .

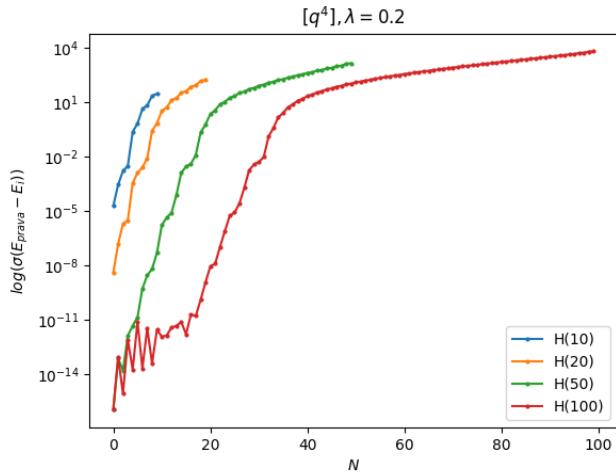
biti matrika neskončna, da bi dobili natančne lastne vrednosti. Ko naredimo matriko končno pa niso nujno več vse lastne vrednosti natančne. Natančnost v odvisnosti od velikosti matrike sem preveril tako, da sem za referenčno matriko oz. lastne vrednosti vzel matriko 1000×1000 in primerjal z lastnimi vrednostimi manjših matrik pri različnih načinih konstrukcije matrike.



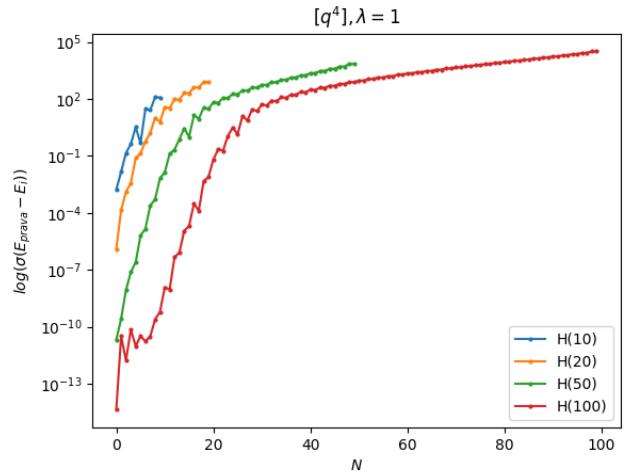
Slika 10: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$



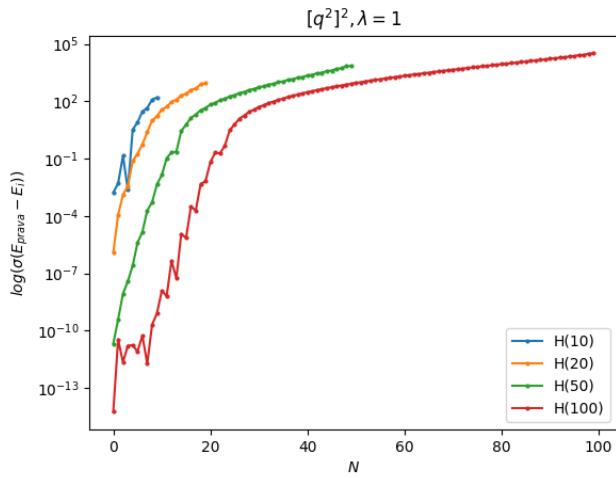
Slika 11: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$



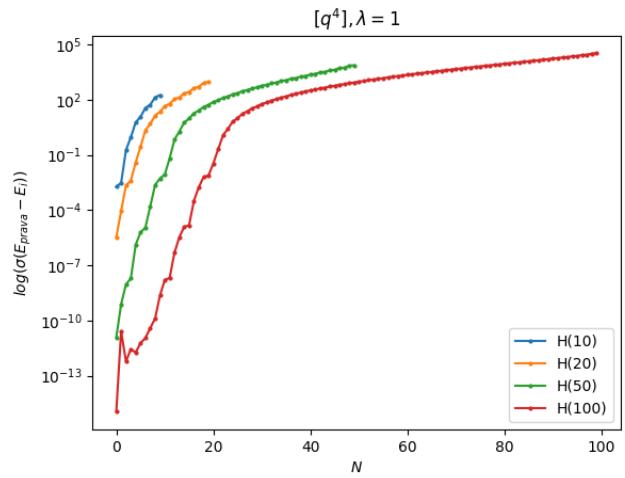
Slika 12: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$



Slika 13: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$



Slika 14: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$

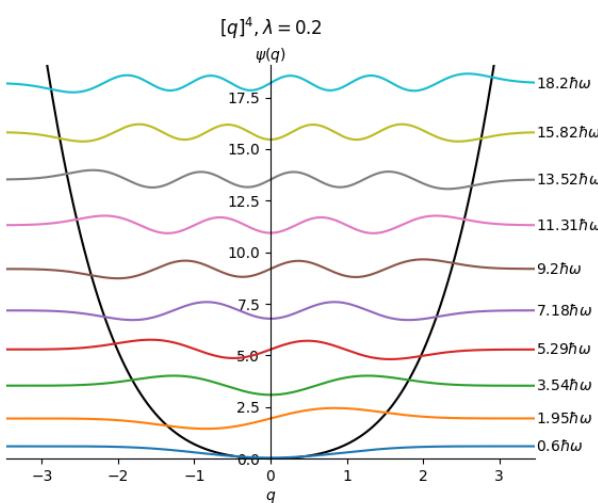


Slika 15: Razlika med pravo in lastno vrednostjo izračunano z $H(i)$

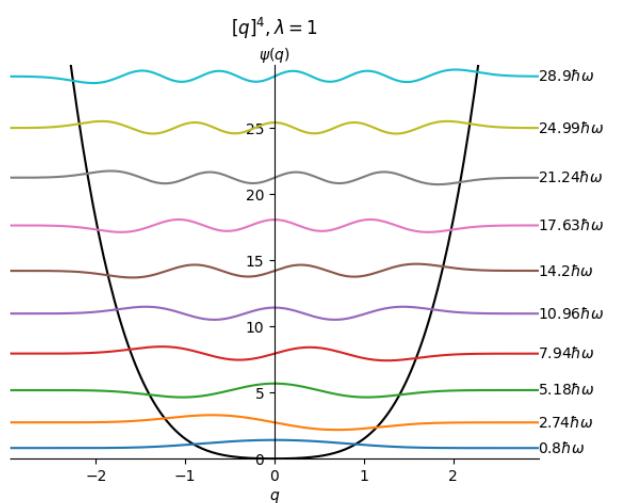
Iz danih slik (10-14) ni opaziti prevelike razlike v natančnosti pri različnem načinu konstrukcije hamiltonke. Da bi pogledal če kakšna metoda odstopa sem naredil simulacijo, ki za lastno vrednost vzame povprečje izračunanih lastnih vrednosti pri različnih konstrukcijah hamiltonke. Animacija prikazuje razliko med dano in povprečno lastno vrednostjo v odvisnosti od lambda, kjer se večajo začetne energije od $N = 1 \dots 30$. Iz animacije je tudi razvidno kako z večanjem lastnih energij začnejo le te divergirati, saj je matrika končna.

Hamiltonovo matriko velikosti 1000×1000 lahko konstruiramo na 3 omenjene načine. Najhitreje se konstruira matrika $[q^4]$, $t = 2.4s$, nato $[q^2]^2$, $t = 24.8$ in kot zadnja $[q]^4$, $t = 71.5$.

Lastne funkcije anharmonskega oscilatorja dobimo kot linearno kombinacijo lastnih funkcij harmonskega oscilatorja



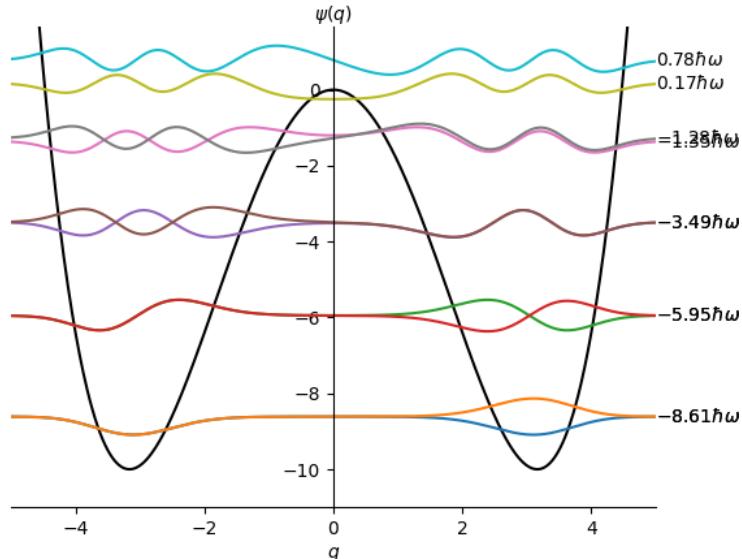
Slika 16: Prvih 10 lastnih funkcij pri $\lambda = 0.2$



Slika 17: Prvih 10 lastnih funkcij pri $\lambda = 1$

3.3 Dodatna naloga

Za izračun lastnih energij in funkcij sem izraz prepisal v obliko $H_0 - 5/2 q^2 + q^4/10$ in z eigh izračunal lastne vrednosti in lastne vektorje.



Slika 18: Lastne funkcije anharmonskega oscilatorja pri danem potencialu.

3.4 Zaključek

Za izračun perturbacijske matrike je najboljši način $[q^4]$, saj je najhitrejši. Pri natančnosti pa bi mislil, da potenciranje matrike povzroči numerične napake, ampak večjih odstopanj nisem opazil. Ker smo v Hilbertovem prostoru je cilj vzeti čim večjo matriko, da dobimo največ natančnih lastnih vrednosti in vektorjev. Ko se lastne vrednosti večajo, začnejo v neki točki divergirati.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM
4. naloga: Fourierova analiza

Žiga Šinigoj, 28191058

Ljubljana, november 2021

1 Uvod

Pri numeričnem izračunavanju Fourierove transformacije

$$H(f) = \int_{-\infty}^{\infty} h(t) \exp(2\pi i f t) dt \quad (1)$$

$$h(t) = \int_{-\infty}^{\infty} H(f) \exp(-2\pi i f t) df \quad (2)$$

je funkcija $h(t)$ običajno predstavljena s tablico diskretnih vrednosti

$$h_k = h(t_k), \quad t_k = k\Delta, \quad k = 0, 1, 2, \dots, N - 1. \quad (3)$$

Pravimo, da smo funkcijo vzorčili z vzorčno gostoto (frekvenco) $f = 1/\Delta$. Za tako definiran vzorec obstaja naravna meja frekvenčnega spektra, ki se imenuje *Nyquistova frekvenca*, $f_c = 1/(2\Delta)$: harmonični val s to frekvenco ima v vzorčni gostoti ravno dva vzorca v periodi. če ima funkcija $h(t)$ frekvenčni spekter omejen na interval $[-f_c, f_c]$, potem ji z vzorčenjem nismo odvzeli nič informacije, kadar pa se spekter razteza izven intervala, pride do *potujitve (aliasing)*, ko se zunanji del spektra preslikava v interval.

Frekvenčni spekter vzorčene funkcije (3) računamo samo v N točkah, če hočemo, da se ohrani količina informacije. Vpeljemo vsoto

$$H_n = \sum_{k=0}^{N-1} h_k \exp(2\pi i kn/N), \quad n = -\frac{N}{2}, \dots, \frac{N}{2}, \quad (4)$$

ki jo imenujemo diskretna Fourierova transformacija in je povezana s funkcijo v (1) takole:

$$H\left(\frac{n}{N\Delta}\right) \approx \Delta \cdot H_n.$$

Zaradi potujitve, po kateri je $H_{-n} = H_{N-n}$, lahko pustimo indeks n v enačbi (4) teči tudi od 0 do N . Spodnja polovica tako definiranega spektra ($1 \leq n \leq \frac{N}{2} - 1$) ustrezata pozitivnim frekvencam $0 < f < f_c$, gornja polovica ($\frac{N}{2} + 1 \leq n \leq N - 1$) pa negativnim, $-f_c < f < 0$. Posebna vrednost pri $n = 0$ ustrezata frekvenci nič ("istosmerna komponenta"), vrednost pri $n = N/2$ pa ustrezata tako f_c kot $-f_c$.

Količine h in H so v splošnem kompleksne, simetrija v enih povzroči tudi simetrijo v drugih. Posebej zanimivi so trije primeri:

če je	h_k realna	tedaj je	$H_{N-n} = H_n^*$
	h_k realna in soda		H_n realna in soda
	h_k realna in liha		H_n imaginarna in liha

(ostalih ni težko izpeljati). V tesni zvezi s frekvenčnim spektrom je tudi moč. *Celotna moč* nekega signala je neodvisna od reprezentacije, Parsevalova enačba pove

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2$$

(lahko preveriš). Pogosto pa nas bolj zanima, koliko moči je vsebovane v frekvenčni komponenti med f in $f + df$, zato definiramo enostransko spektralno gostoto moči (*one-sided power spectral density*, PSD)

$$P_n = |H_n|^2 + |H_{N-n}|^2.$$

Pozor: s takšno definicijo v isti koš mečemo negativne in pozitivne frekvence, vendar sta pri realnih signalih h_k prispevka enaka, tako da je $P_n = 2|H_n|^2$.

Z obratno transformacijo lahko tudi rekonstruiramo h_k iz H_n

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n \exp(-2\pi i kn/N) \quad (5)$$

(razlika glede na enačbo (4) je le predznak v argumentu eksponenta in utež $1/N$).

2 Naloga

- Izračunaj Fourierov obrat Gaussove porazdelitve in nekaj enostavnih vzorcev, npr. mešanic izbranih frekvenc. Za slednje primerjaj rezultate, ko je vzorec v intervalu periodičen (izbrane frekvence so mnogokratniki osnovne frekvence), z rezultati, ko vzorec ni periodičen (kako naredimo Gaussovo porazdelitev 'periodično' za FT?). Opazuj pojav potujitve na vzorcu, ki vsebuje frekvence nad Nyquistovo frekvenco. Napravi še obratno transformacijo (5) in preveri natančnost metode. Poglej, kaj se dogaja z časom računanja - kako je odvisen od števila vzorčenj?
- Po Fourieru analiziraj 2.3s dolge zapise začetka Bachove partite za violino solo, ki jih najdeš na spletni strani Matematičnofizikalnega praktikuma. Signal iz začetnih taktov partite je bil vzorčen pri 44100Hz, 11025Hz, 5512Hz, 2756Hz, 1378Hz in 882Hz. S poslušanjem zapisov v formatu .mp3 ugotovi, kaj se dogaja, ko se znižuje frekvenca vzorčenja, nato pa s Fourierovo analizo zapisov v formatu .txt to tudi prikaži.
- Dodatno:** Napravi Fourierovo analizo signalov, ki jih dobiš pri vaji *Akustični resonator* pri Fizikalnem praktikumu II. Posnetke treh različnih signalov prav tako najdeš na spletni strani.

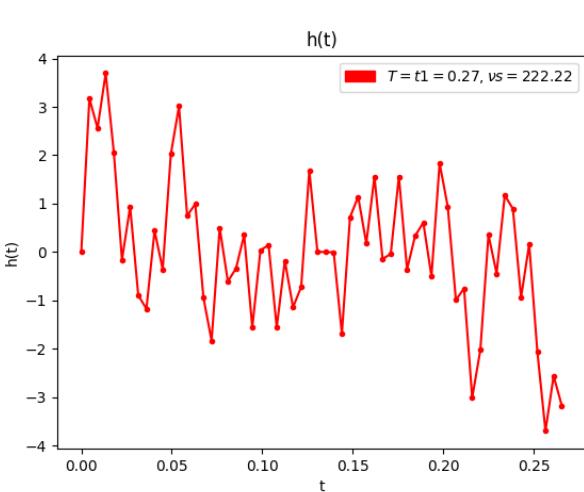
3 Rezultati

3.1 Fourierve transformacije in lastnosti

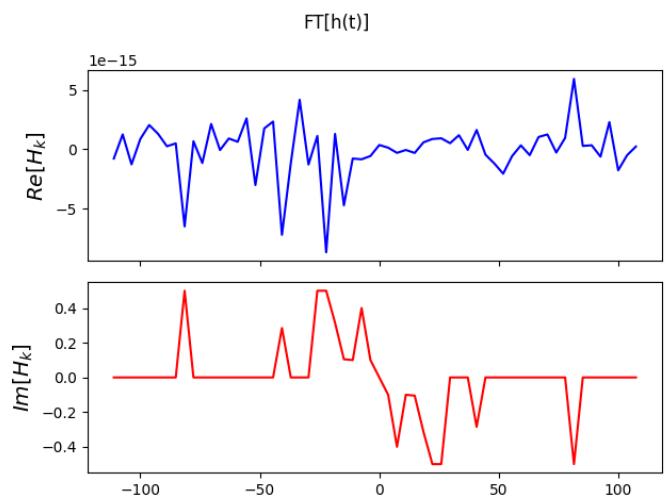
Najprej sem pogledal različne tipe funkcij (periodične, neperiodične, ...) in njihove transformacije, lastnosti in pomankljivosti. Najprej bo v podpoglavlju napisana funkcija in nato njena obdelava in analiza. Na grafih predstavlja T periodo vzorčenja in νs frekvenco vzorčenja.

3.1.1 $h(t) = 0.2\sin(2\pi t/t_01)+0.8\sin(2\pi t/(2t_01)+0.2\sin(2\pi t/(3t_01))+0.21\sin(2\pi t/(4t_01))+0.63\sin(2\pi t/(5t_01))-\\ \sin(2\pi t/(6t_01)) + \sin(2\pi t/(7t_01)) + 0.57\sin(2\pi t/(11t_01)) + \sin(2\pi t/(22t_01))$

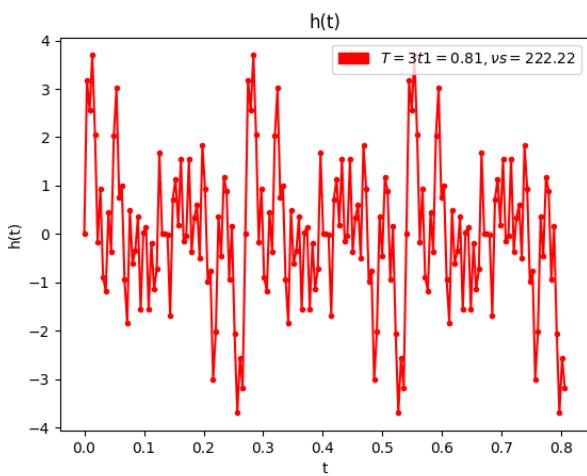
Funkcija $h(t)$ je periodična s periodo $t_{01} = 0.27$ in njenimi večkratniki (slika 1). Perioda vzorčenja je 1 periodo funkcije $h(t)$, ki ima periodo t_{01} . Fourierova transformacija funkcije ima precej slabo ločljivost, to lahko popravimo tako, da vzorčimo več časa (slika 3, 4). Tako dobimo večjo ločljivost vzorca. Pri periodičnih funkcijah se lahko izognemo puščanju tako, da naš vzorec pokrije periodo oz. večkratnik periode. Veljati mora, da je naslednja točka, ki bi jo vzorčili enaka kot naša prva vzorčena točka. Primer neperiodičnega vzorčenja in posledično puščanja prikazujeta sliki 5 in 6. Z večanjem točk vzorčenja večamo dosegljiv frekvenčni interval v frekvenčnem spektru. Spodnjo mejo frekvence vzorčenja predstavlja dvakratnik Nyquistove frekvence. To je najnižja frekvenca vzorčenja pri kateri še ohranimo polno informacijo signala, če se spustimo pod to frekvenco se frekvence, ki so večje od Nyquistove, preslikajo v interval do Nyquistove frekvence (animacija).



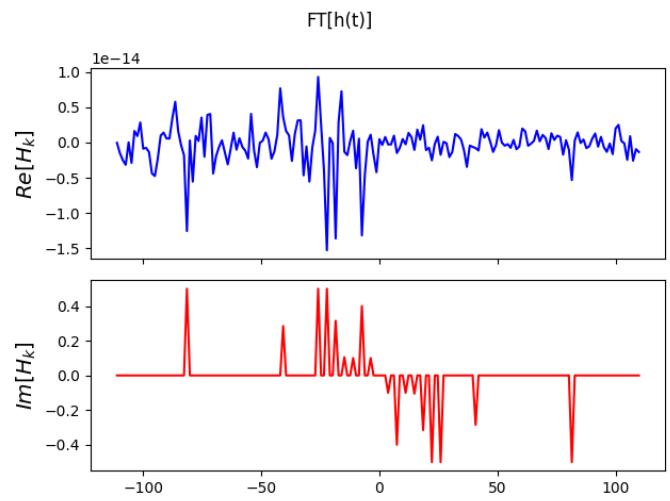
Slika 1



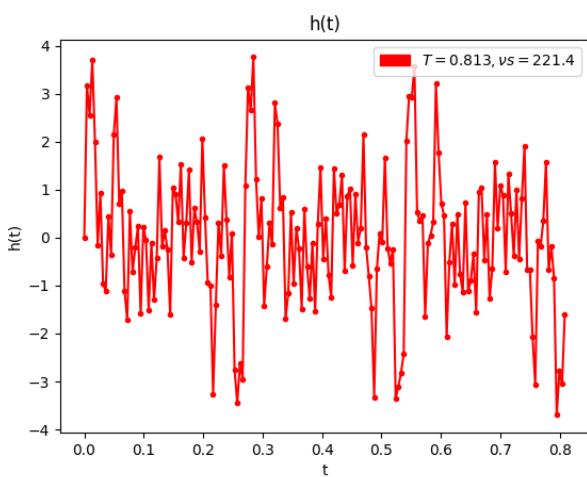
Slika 2



Slika 3

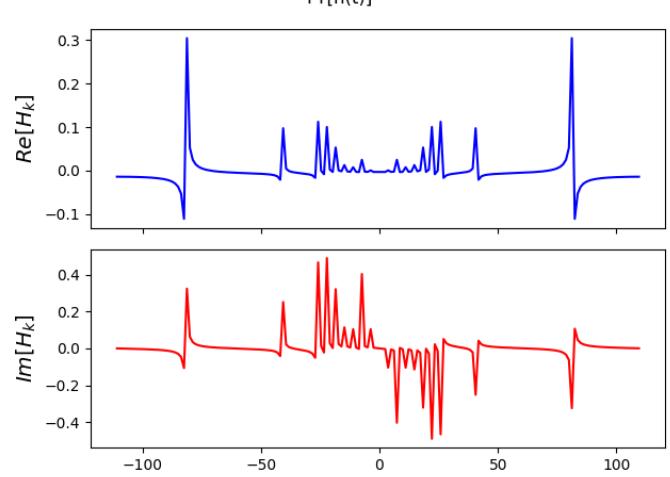


Slika 4

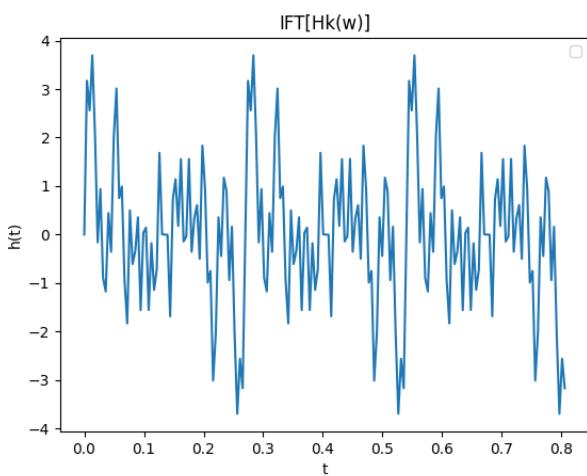


Slika 5

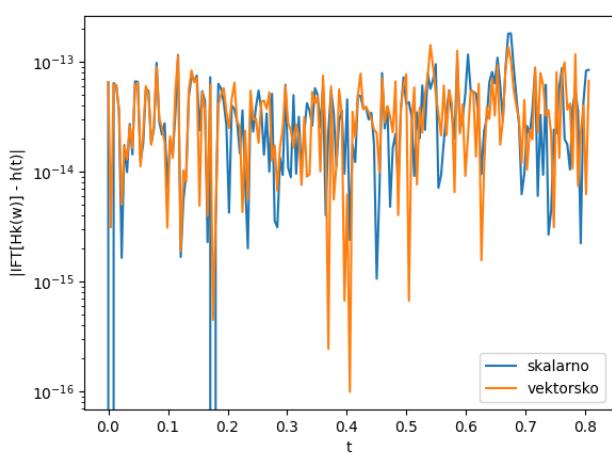
Seveda lahko z majhno spremembo pogledamo tudi inverzno transformacijo in njeno ujemanje z začetno funkcijo (slika 7 in 8)



Slika 6



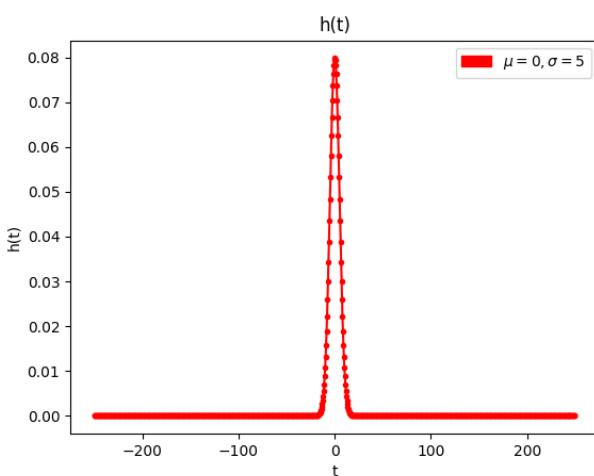
Slika 7



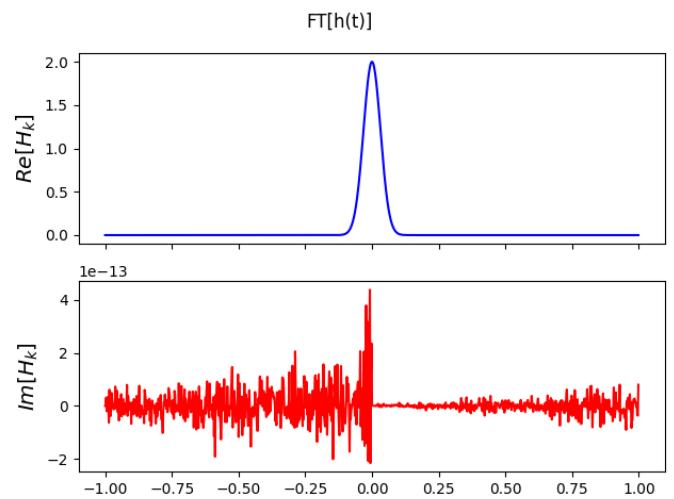
Slika 8

3.1.2 $h(t)$ -Gaussova funkcija

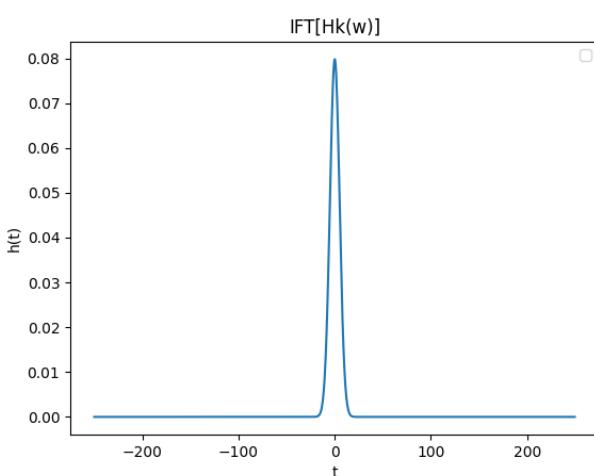
Pri neperiodičnih funkcijah kot je Gaussova funkcija je potrebno biti previden. DFT predpostavlja, da je začetna točka v izhodišču. Če imamo Gaussovo funkcijo centrirano na sredini, potem lahko naredimo dvoje. Funkcijo lahko zamaknemo in potem fourierovo transformacijo pomnožimo s faktorjem premika. Druga možnost, ki sem jo uporabil, je ta da damo vzorčni interval na polovico in ju zamenjamo (to sm naredil z `np.roll()` iz paketa NumPy). Tako dobimo 'periodično' funkcijo, ki jo transformiramo. Po transformaciji spet zamenjamo polovici frekvenčnega spektra, da dobimo Gaussovo funkcijo (slika 9 in 10). Iz frekvenčnega spektra lahko naredimo inverzno transformacijo z enakim postopkom in pogledamo če se vrednosti ujemajo pri obeh metodah transformacije (slika 11 in 12).



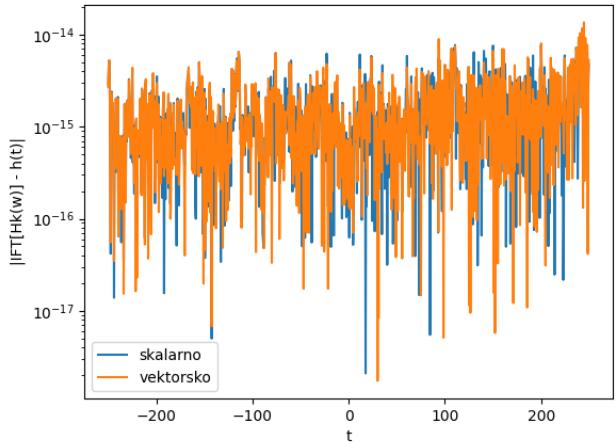
Slika 9



Slika 10



Slika 11

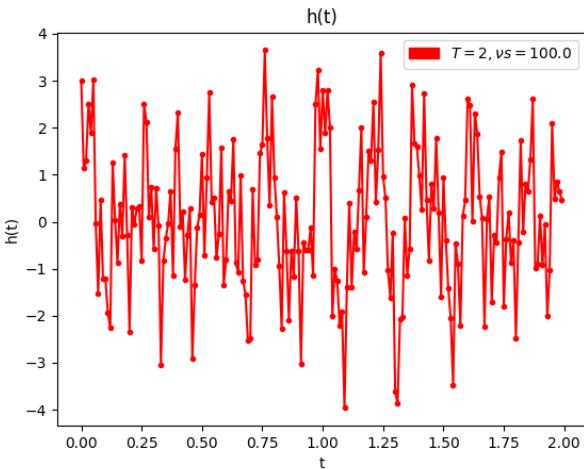


Slika 12: Razlika originalne Gaussove funkcije in Gaussove funkcije po transformacijah pri čemer sta DFT in IDFT izračunana na dva načina - vektorsko in skalarno.

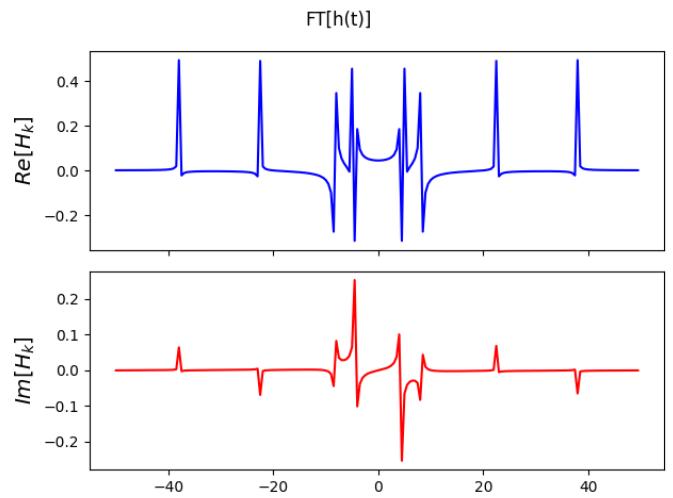
3.1.3 $h(t) = \sin(2\pi t/0.23) + \cos(2\pi t/0.2) + \cos(2\pi t/0.0444) + \cos(2\pi t/0.02633) + \sin(2\pi t/0.121679)$

Funkcija $h(t)$ je sestavljena iz lihih in sodih, zato bomo frekvence dobili v imaginarnem in realnem delu. Ker funkcija ni periodična bomo v frekvenčnem spektru dobili nekaj puščanja (slika 13, 14). Pri tem mislim to, da katerokoli točko vzamemo za konec vzorčenja, bodo pri funkcijah z različnimi periodami nekatere, ki ne končajo svoje periode v naši zadnji točki. Če želimo poudariti frekvence signala lahko uporabimo zero padding metodo, kjer dodamo ničle na konec intervala in tako navidezno izboljšamo meritev (slika ??). To pa ne vedno pomeni, da smo stvar izboljšali, na sliki dobimo dobro frekvenčno sliko sodih funkcij. V frekvenčnem spektru lihih funkcij pa smo ojačali tudi frekvence, ki niso vir našega signala, ampak po mojem

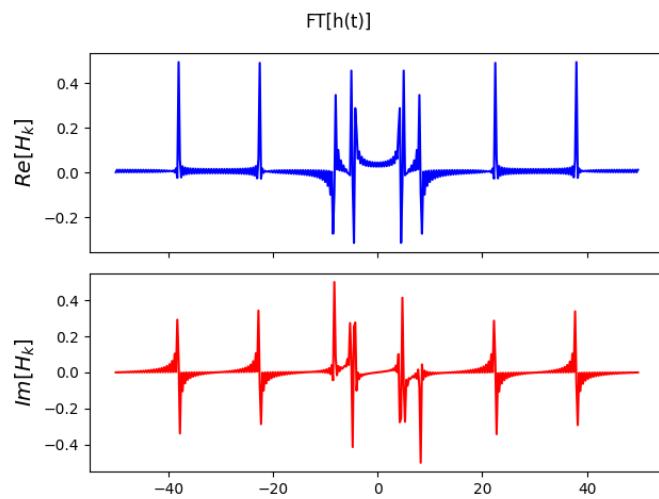
posledica puščanja.



Slika 13



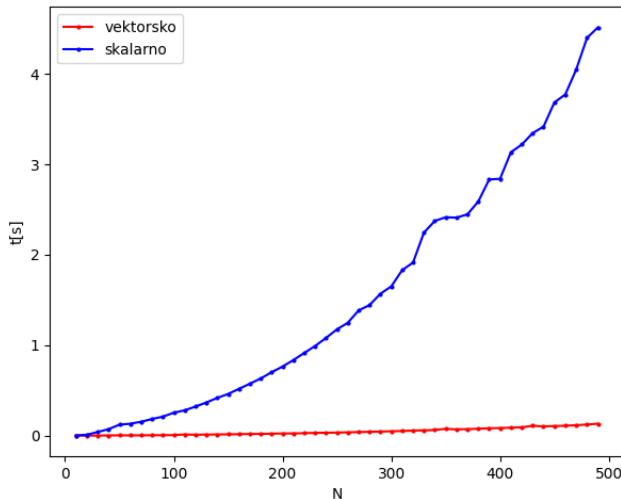
Slika 14



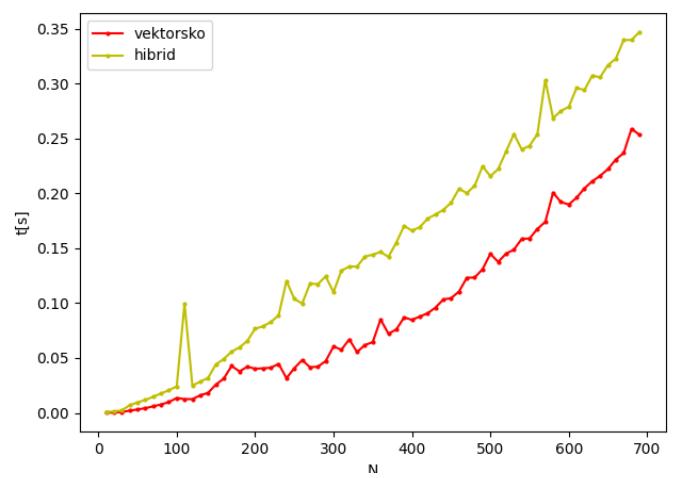
Slika 15

3.1.4 Hitrost in poraba pomnilnika

Primerjal sem hitrosti obeh metod DFT, kjer ena izračuna transformacijo tako, da naredi matriko in matrično množi, druga pa računa elemente preko dveh zank. Druga metoda ima dve zanki kar pomeni, da je $t \propto n^2$. Časovno odvisnost podaja slika 16. Težava vektorske metode je ta, da izračuna celotno matriko. Velike matrike porabijo veliko RAM-a. To se predvsem vidi v naslednjih nalogih, ko imamo veliko vzorcev. Zato sem naredil še 'hibrid', torej mešanico med prvo in drugo metodo. Za izračun ene frekvence potrebujemo eno vrstico matrike in vektor iz časovnega prostora. Metoda posledično uporablja samo eno zanko (slika 17).



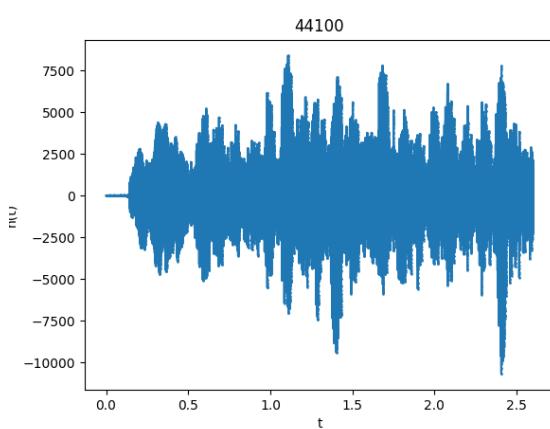
Slika 16



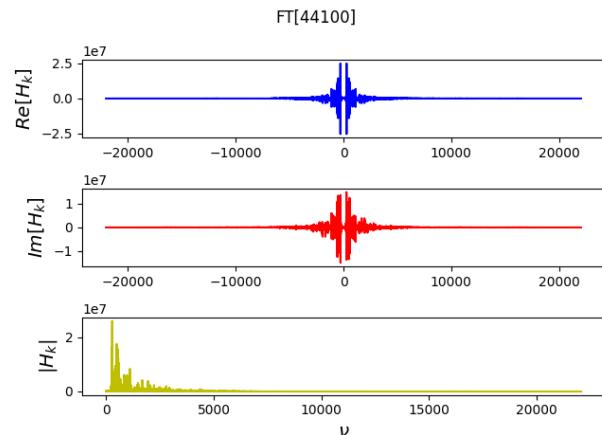
Slika 17

3.2 Bachova partita

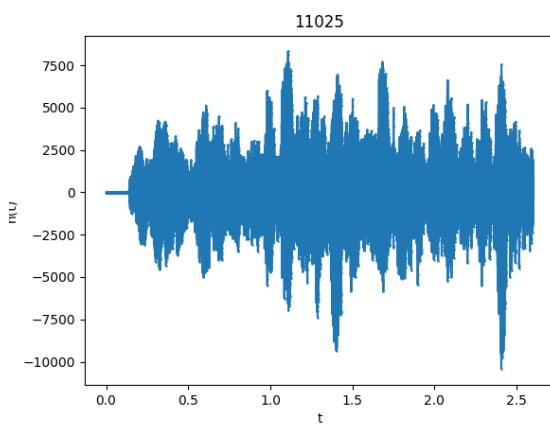
Pri različnih vrekvencah vzorčenja Bachove partite opazim, da se z zmanjševanjem frekvence vzorčenja kvari melodija. Najverjetneje prihaja z zmanjševanjem frekvence do potujevanja signala in se višje frekvence preslikujejo v frekvence do Nyquistove in posledično vsebuje melodija vedno več nižjih frekvenc. To lahko opazimo tudi z opazovanjem spektra. Pri zmanjševanju frekvence vzorčenja lahko tudi opazimo, da se veča reža med negativnimi in pozitivnimi frekvencami. Razloga za to nevem, mogoče je zaradi preslikave višjih frekvenc z majhno amplitudo v začetek spektra, ampak se frekvence ob preslikavi seštevajo, zato nisem ravno prepričan.



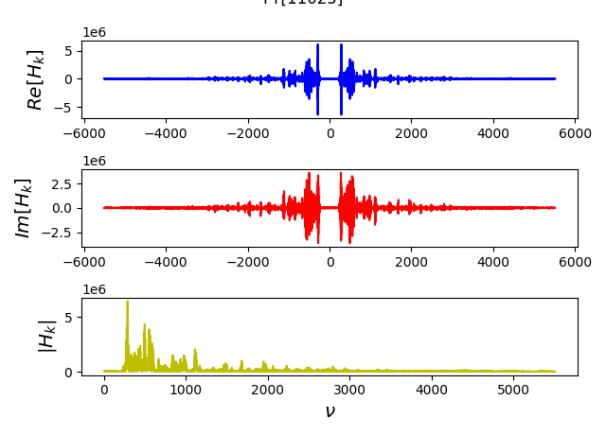
Slika 18



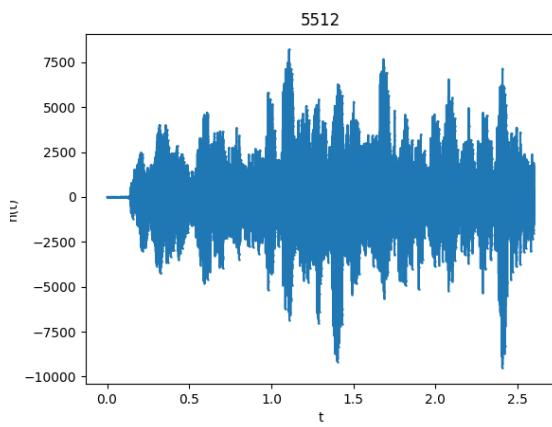
Slika 19



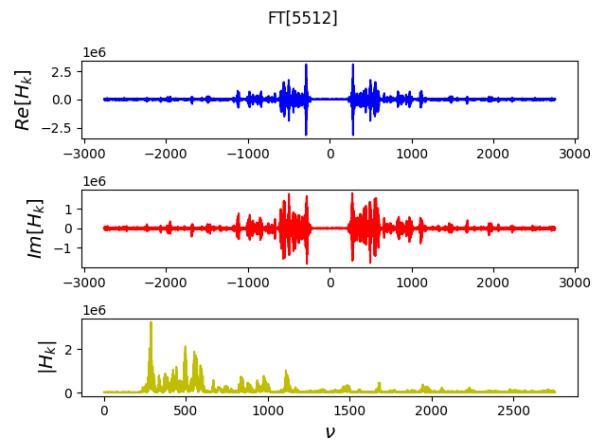
Slika 20



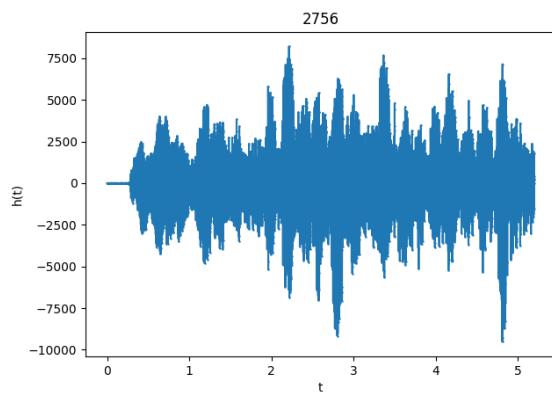
Slika 21



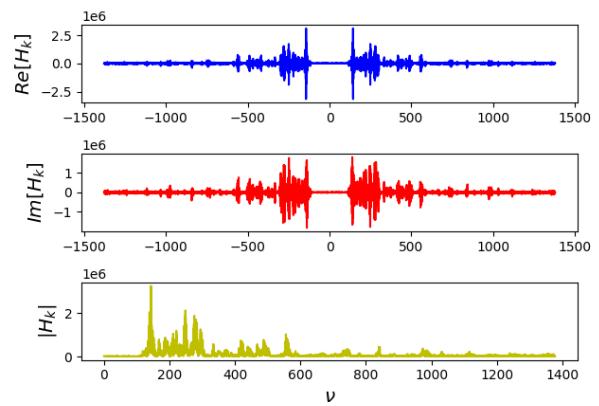
Slika 22



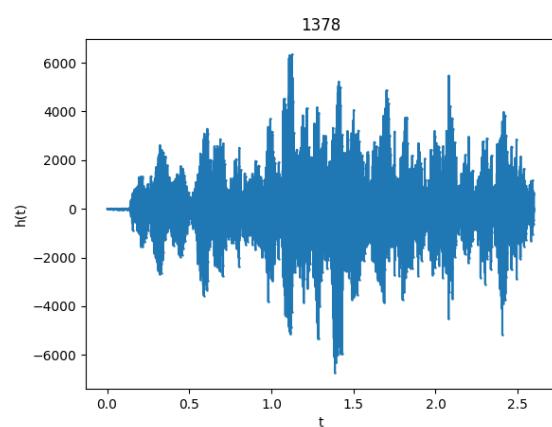
Slika 23



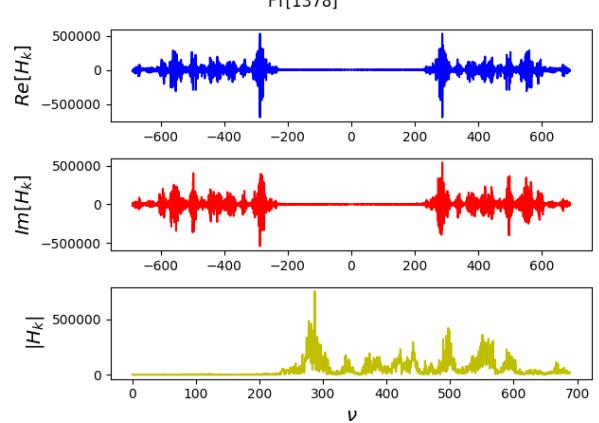
Slika 24



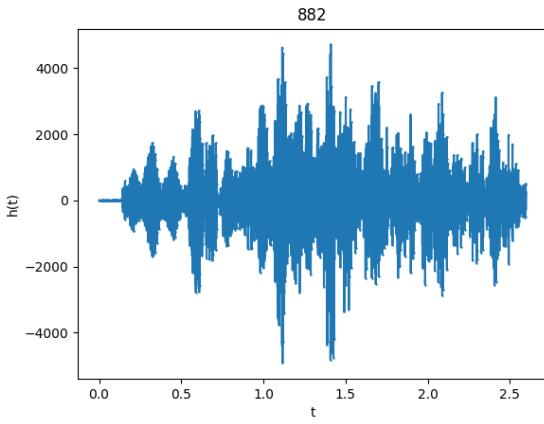
Slika 25



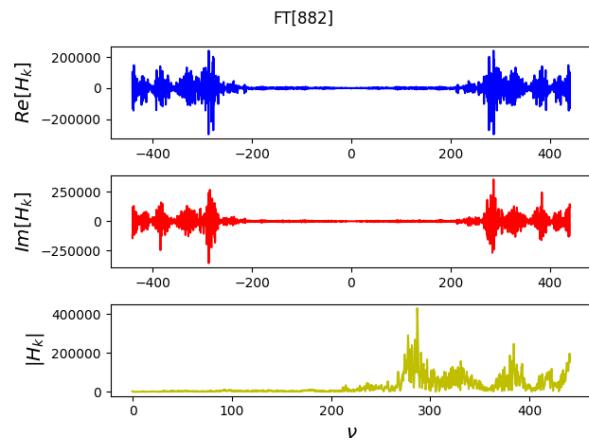
Slika 26



Slika 27



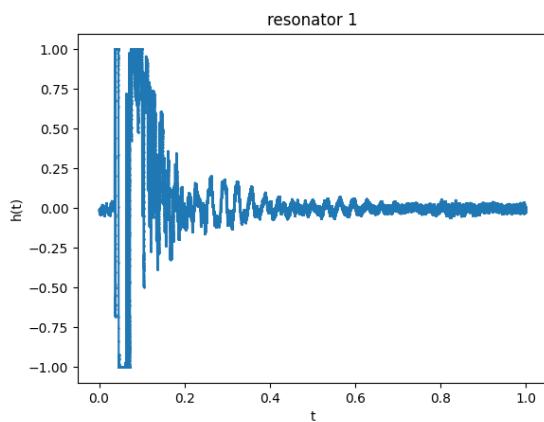
Slika 28



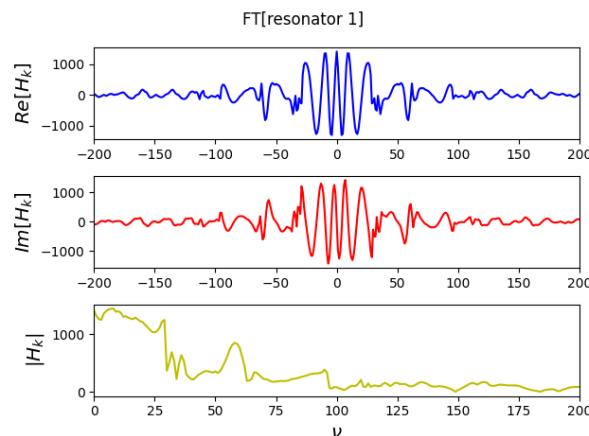
Slika 29

3.3 Akustični rezonator

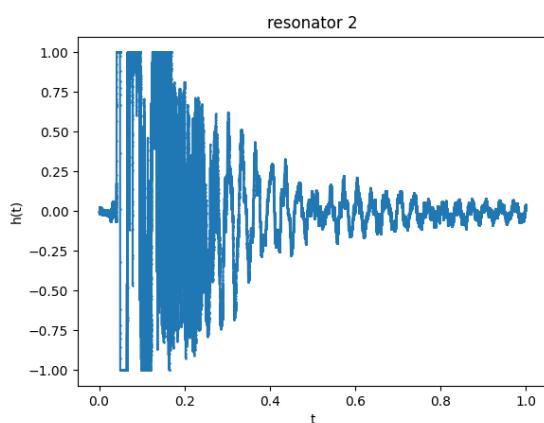
Pri transformaciji vzorcev akustičnega rezonatorja sem vzel samo prve 4 vzorce iz spletnne učilnice, saj nisem uspel odpreti ostalih. Števila na grafih od 1 do 3 pomenijo moč udarca, kjer je 1 blag udarec. Pri blažjih udarcih so zastopane nižje frekvence in je spekter moči manjši po velikosti. Z večanjem udarca se večajo frekvence in njihova gostota ter moč. V rezonatorju bi načeloma morali videti lastne frekvence in njihove večkratnike, torej vrhove v frekvenčnem spektru, kar pa je precej težko videti.



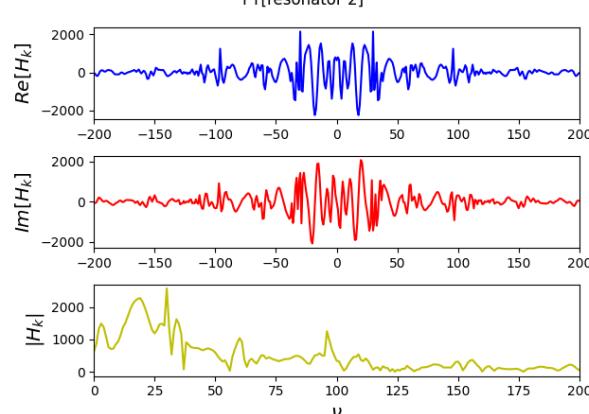
Slika 30



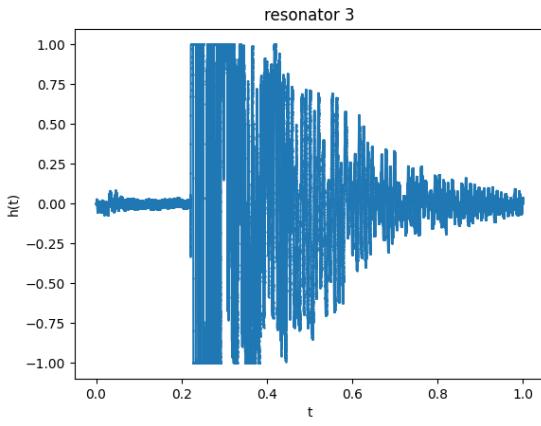
Slika 31



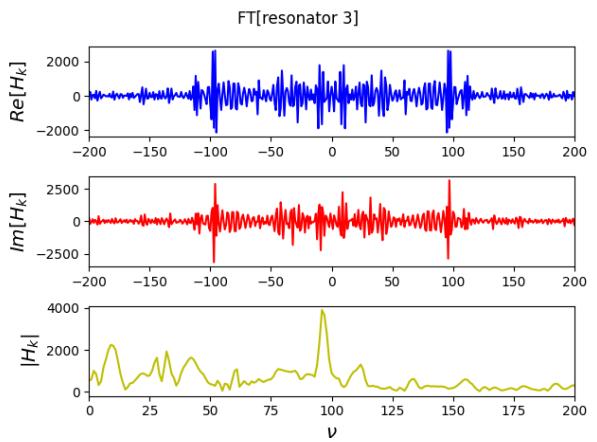
Slika 32



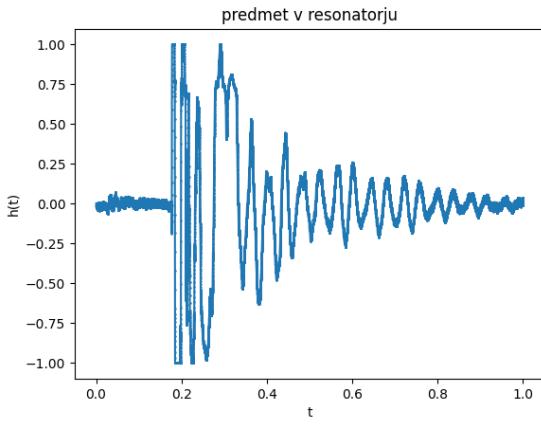
Slika 33



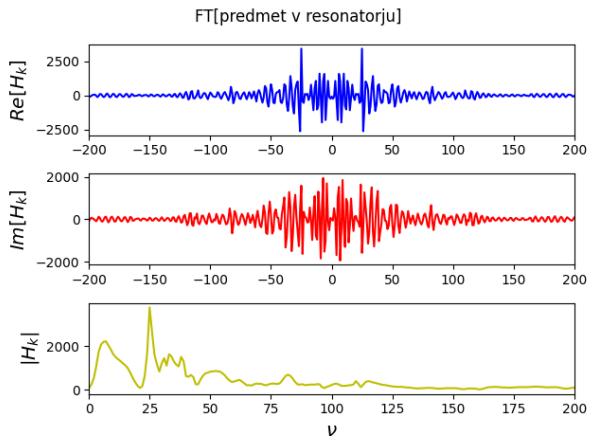
Slika 34



Slika 35



Slika 36



Slika 37

3.4 Zaključek

Ob reševanju naloge sem se spoznal s težavami, na katere lahko naletim pri diskretni Fourierovi transformaciji in kakšne so rešitve oz. načini da rezultate izboljšam. Pri zajemanju podatkov je pomembna frekvenca vzorčenja, pri transformacijah pa razne simetrije in zamikanje podatkov in računska oz. prostorska zahtevnost transformacije.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

5. naloga: Hitra Fourierova transformacija (FFT)

Žiga Šinigoj, 28191058

Ljubljana, november 2021

1 Uvod

Diskretno Fourierovo transformacijo smo definirali kot

$$H_k = \sum_{n=0}^{N-1} h_n \exp(2\pi i kn/N), \quad k = -\frac{N}{2}, \dots, \frac{N}{2},$$

oziroma

$$H_k = \sum_{n=0}^{N-1} W_N^{nk} h_n, \quad W_N = \exp(2\pi i/N).$$

Ta postopek ima očitno časovno zahtevnost N^2 . Račun pa je mogoče izvesti tudi z bistveno manj operacijami. Osnovni premislek je razcep

$$H_k = H_k^{\text{sod}} + W_N^k H_k^{\text{lih}},$$

kjer smo transformiranko H izrazili s transformacijama njenih sodih in lihih členov, pri čemer je vsota vsake od transformacij zdaj dolžine $N/2$. Gornjo relacijo lahko uporabljamo rekurzivno: če je N enak potenci števila 2, lahko rekurzijo razdrobimo do nizov, ki imajo samo še en člen. Zanj je transformacija identiteta. Za obrat pri eni vrednosti frekvence (pri danem m) je potrebno na vsakem koraku rekurzije le eno množenje s potenco W , korakov pa je $\log_2 N$. Skupna časovna zahtevnost je torej le še $N \log_2 N$.

Da ne iščemo pripadnikov niza po vsej tabeli, si podatke preuredimo. Lahko je pokazati, da je v prvotni tabeli treba med seboj zamenjati podatke, katerih vrstna števila v binarnem zapisu so obrnjena: v novem redu jemljemo člene kar po vrsti. Tudi potenc W ne izražamo vedno znova s sinusom in kosinusom, pač pa jih računamo z rekurzijo. Tak ali podoben postopek je osnova vseh algoritmov hitre Fourierove transformacije (FFT).

Z neko transformacijo iz družine FFT bomo izračunali korelacijsko funkcijo dveh signalov. Korelacija periodičnih funkcij $g(t)$ in $h(t)$ s periodo T je definirana kot:

$$\varphi_{gh}(\tau) = \frac{1}{T} \int_0^T g(t + \tau) h(t) dt,$$

oziroma diskretno

$$\varphi_{gh}(n) = \frac{1}{N} \sum_{k=0}^{N-1} g_{k+n} h_k.$$

Računamo torej skalarni produkt funkcij, ki sta časovno premaknjeni za τ oziroma n . Če je za določeno vrednost premika ta funkcija višja kot v okolini, potem to pomeni, da sta si funkciji podobni, le da ju je treba premakniti, da se to vidi.

V primeru, da sta funkciji (signal) ki ju primerjamo, enaki, računamo njuno avtokorelacijsko funkcijo: ta je mera za to, ali signal ostaja s pretekanjem časa sam sebi podoben. Če je signal slabo koreliran (sam s sabo), korelacija $\varphi_{hh}(n)$ relaksira h kvadratu povprečnega signala $\langle h \rangle^2$, kjer je

$$\langle h \rangle = \frac{1}{N} \sum_{k=0}^{N-1} h_k.$$

Iz lokalnih maksimov v avtokorelacijski funkciji sklepamo na periodičnosti, bodisi popolne ali približne. Pri periodičnih signalih je tudi avtokorelacijska funkcija striktno periodična, za stohastične procese pa je značilna eksponentna avtokorelacijska funkcija. še bolj nas zanima, kako hitro se korelacija izgublja: računamo rajši reskalirano obliko avtokorelacije

$$\tilde{\varphi}_{hh}(n) = \frac{\varphi_{hh}(n) - \langle h \rangle^2}{\varphi_{hh}(0) - \langle h \rangle^2},$$

kjer je imenovalec nekakšno merilo za varianco signala,

$$\sigma^2 = \varphi_{hh}(0) - \langle h \rangle^2 = \frac{1}{N} \sum_{k=0}^{N-1} (h_k - \langle h \rangle)^2.$$

Pri zgornjih enačbah moramo še “peš” poskrbeti za periodično zaključenost signala pri $n = N$, torej da je perioda enaka velikosti vzorca. Če tega ne moremo narediti, je bolj pravilna definicija avtokorelacije

$$\varphi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{N-n-1} h_{k+n} h_k.$$

Praktičen račun po zgornji formuli lahko postane za velike vzorce prezamuden. Avtokorelacijo rajši računamo s FFT (DFT) \mathcal{F} , saj je korelacija obratna Fourierova transformacija \mathcal{F}^{-1} produkta Fourierovih transformacij \mathcal{F} , torej z $G = \mathcal{F}g$ in $H = \mathcal{F}h$ dobimo

$$\varphi_{gh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} [G \cdot (H)^*]$$

oziroma

$$\varphi_{hh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} [|H|^2].$$

Za račun s FTT signale dolžine N najprej prepišemo v dvakrat daljše, periodično zaključene podatkovne nize, $\tilde{h}_n = h_n$, $\tilde{h}_{n+N} = 0$ za $n = 0, \dots, N-1$ in $\tilde{h}_{n+2N} = \tilde{h}_n$. Tedaj se avtokorelacija zapiše v obliki

$$\varphi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{2N-1} \tilde{h}_{k+n} \tilde{h}_k,$$

kar lahko izračunamo s FFT.

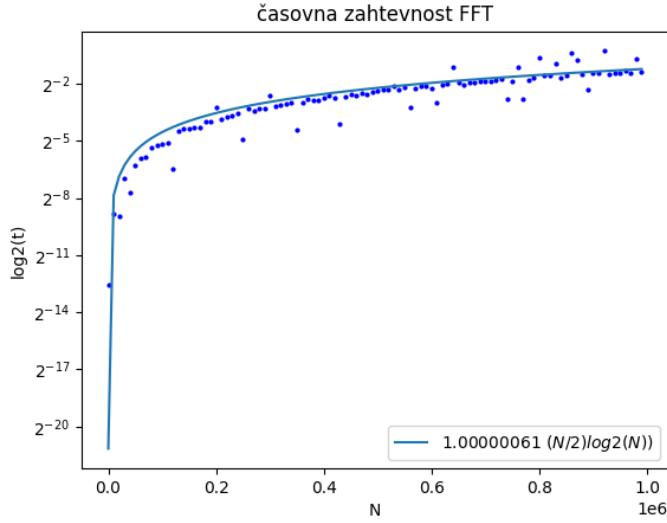
2 Naloga

Na spletni strani MF praktikuma najdeš posnetke oglašanja velike uharice, naše največje sove. Posneti sta dve sovi z minimalnim ozadjem (`bubomon` in `bubo2mon`) in nekaj mešanih signalov, ki zakrivajo njuno oglašanje (`mix`, `mix1`, `mix2` in `mix22`). V signalih `mix2` in `mix22` je oglašanje sove komaj še zaznavno. Izračunaj avtokorelacijsko funkcijo vseh signalov in poskusi ugotoviti, za katero sovo gre pri teh najbolj zašumljenih signalih!

3 Rezultati

3.0.1 Časovna zahtevnost FFT

Najprej sem želet pogledati, če se teorija ujema z prakso. Po teoriji je časovna zahtevnost FFT $N/2 \log_2 N$. Pri računanu FFT sem uporabljal `fft` metodo iz programskega paketa `SciPy` v Pythonu (slika 1). Iz danega grafa vidim, da to drži, odstopanja na decimalnih mestih bi bila manjša, če bi vzel večjo gostoto in število členov.

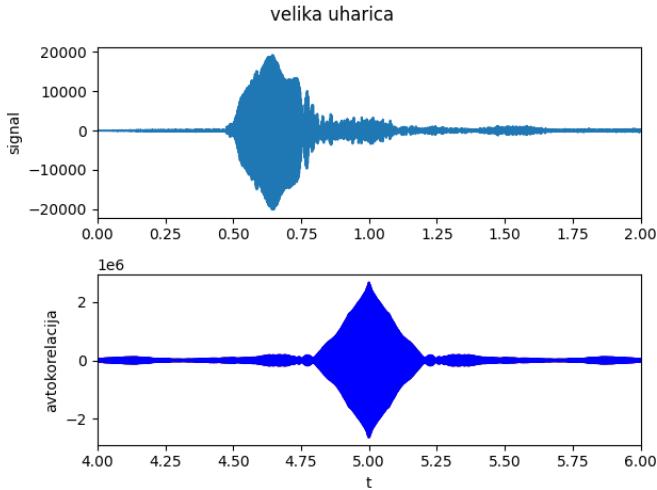


Slika 1

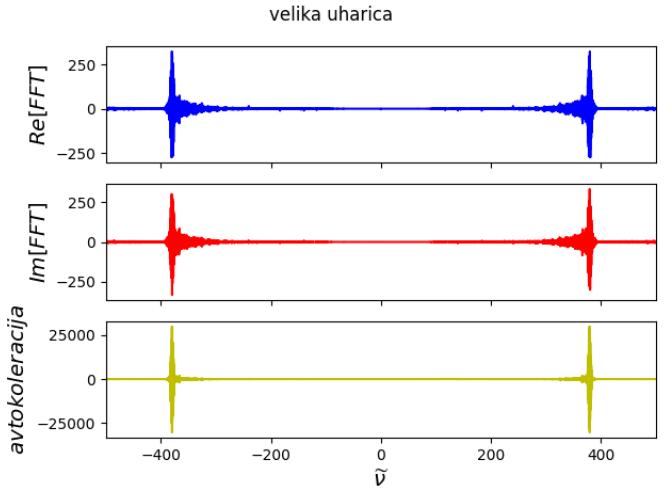
3.1 Velika uharica

Za izračun Fourierove transformacije sem uporabil *fft* funkcijo iz *SciPy*, za izračun koleracije sem uporabil svojo implementirano funkcijo in *signal.correlate* prav tako iz knjižnice *SciPy*. Pri moji implementaciji sem dodal še zero-padding, da se frekvenčni spektri ne prekrivajo.

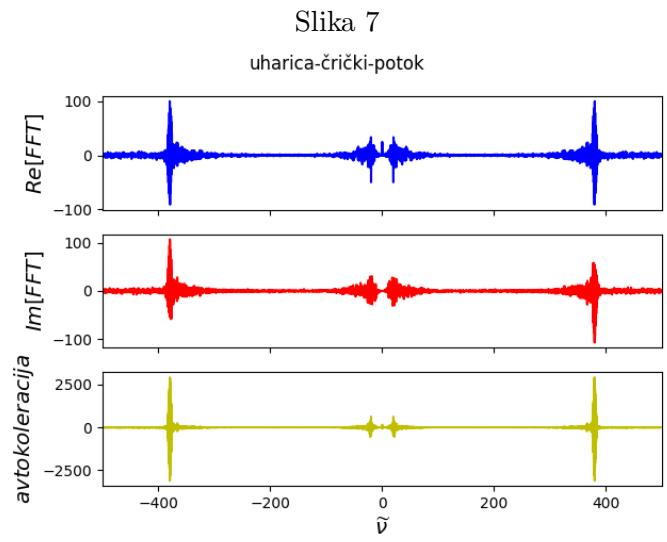
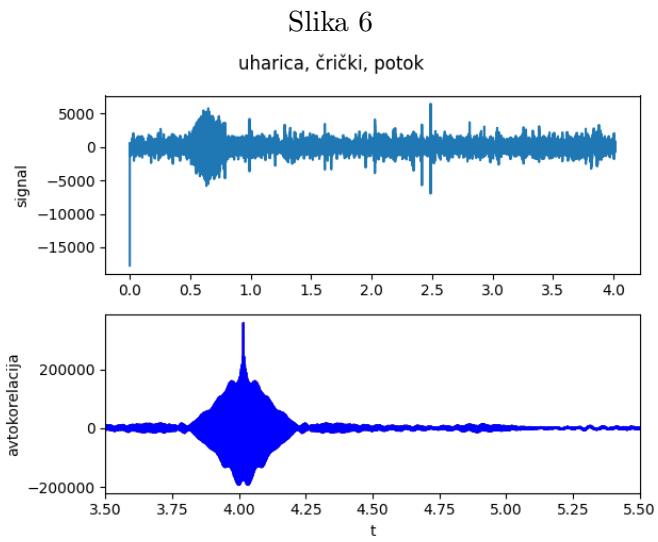
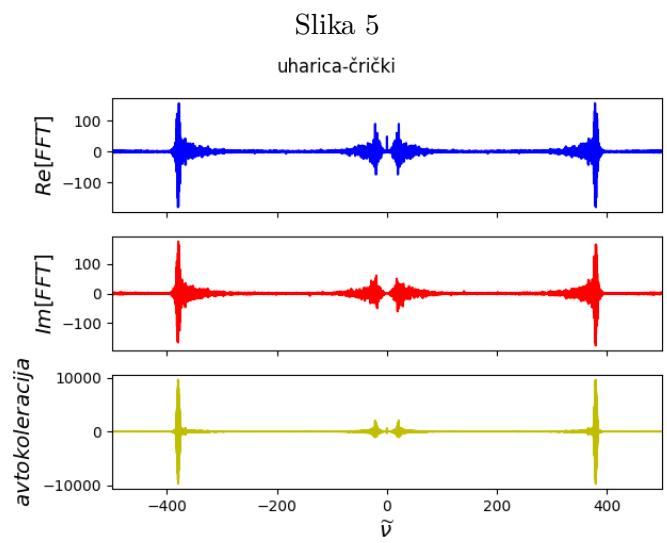
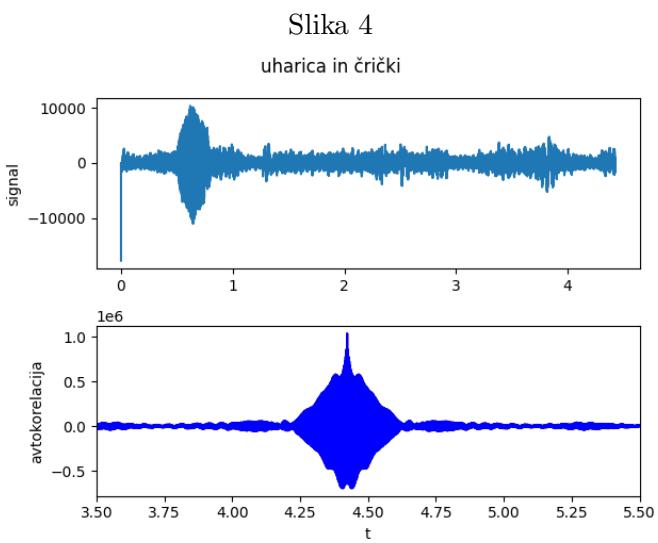
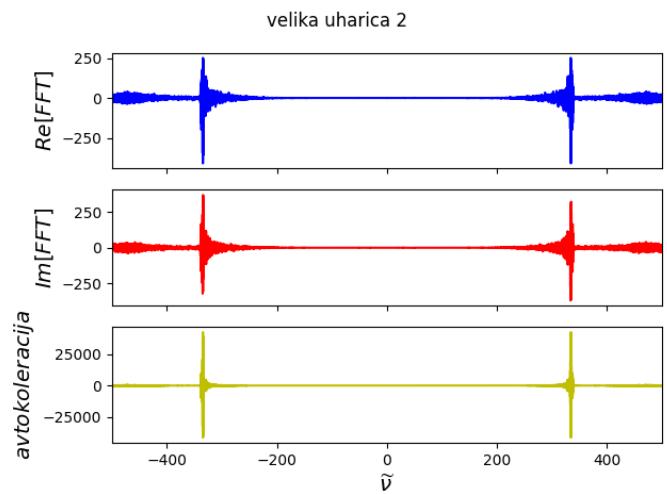
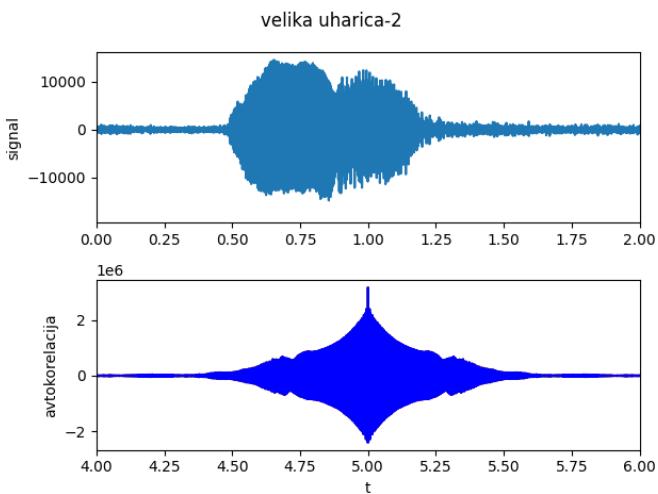
Spodnji grafi prikazujejo na levi strani časovno sliko signala in avtokoleracijo, na desni pa signal in avtokoleracijo v frekvenčnem spektru. Z uporabo avtokoleracije se znebimo šuma v signalu. Ker množimo dve enaki premaknjeni funkciji in integriramo, izločimo vse naključne signale. Veliko lažje je če to naredimo preko Fourierove transformacije, saj je v splošnem potrebno samo množiti funkcijo v frekvenčnem spektru z konjugirano vrednostjo in jo transformirati nazaj. V frekvenčnem spektru dobimo tako samo realni del signala. Čeprav je veliko lažje ugotoviti za katero sovo gre v frekvenčnem spektru mislim, da to nekako pokaže tudi avtokoleracija v času. Pri uharici z višjim tonom je očiščen signal bolj stisnjen kar pomeni, da je 'povprečna' valovna dolžina signala manjša in s tem večja frekvence. Pri veliki uharici 2, pa je signal bolj razširjen in velja ravno obratno. Seveda pa se frekvenca glasu sove bolje vidi v očiščenem frekvenčnem spektru.



Slika 2

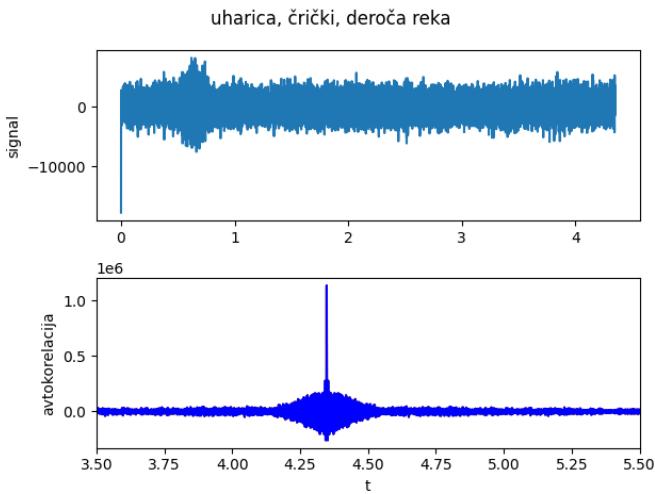


Slika 3

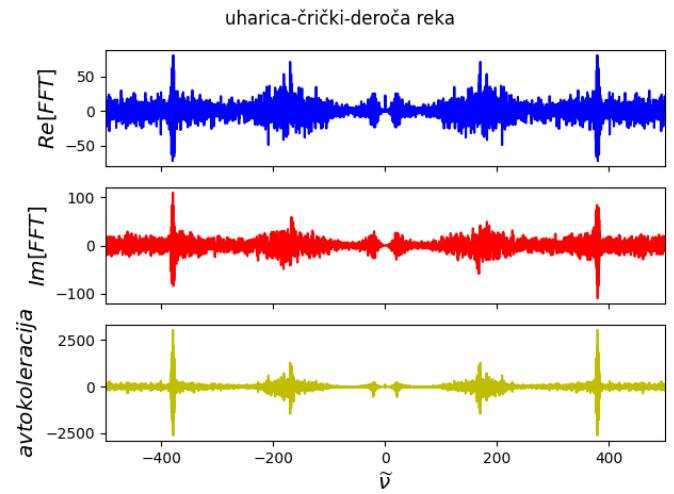


Slika 8

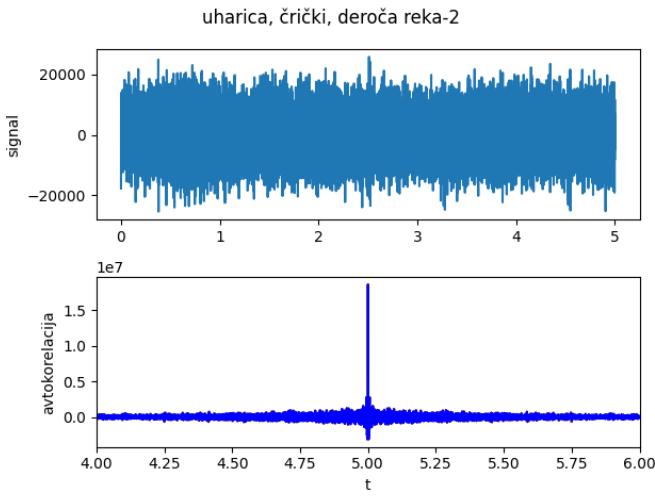
Slika 9



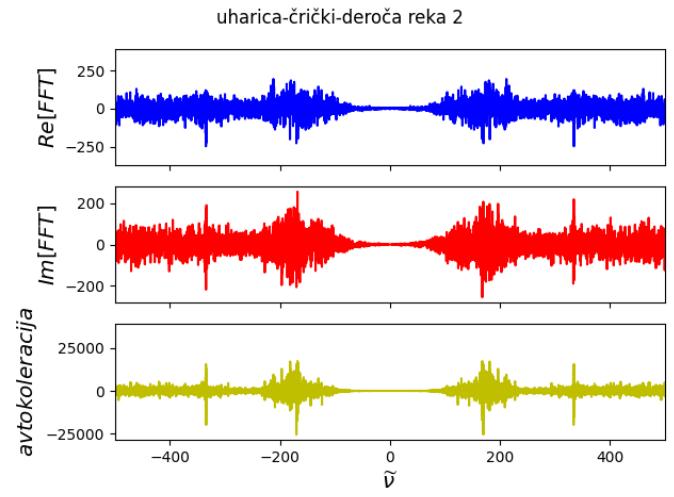
Slika 10



Slika 11

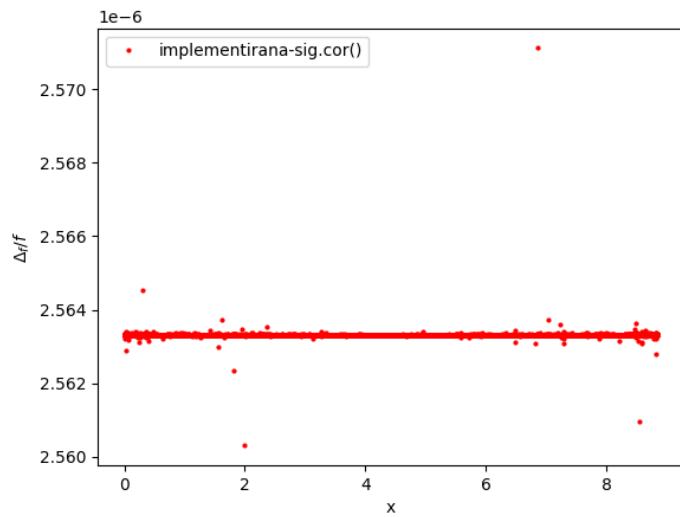


Slika 12



Slika 13

Oglejmo si še relativno napako implementirane funkcije za avtokoleracijo (slika 14). Za pravo vrednost sem vzel vrednosti, ki jih da `signal.correlate()`.



Slika 14

3.2 Zaključek

Hitra fourierova transformacija je zaradi časovne zahtevnosti zelo uporabna. Z njo lahko računamo transformacije pri velikih frekvencah vzorčenja realnih signalov, kar pa ne nujno velja za ostale metode. Z avtokoleracijo lahko izluščimo željen spekter signala in se znebimo odvečnega šuma. Pri transformacijah je potrebno paziti na zasuk podatkov, da dobimo prave slike v časovnem in frekvenčnem spektru.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

6. naloga: Enačba hoda

Žiga Šinigoj, 28191058

Ljubljana, november 2021

1 Uvod

Za opis najpreprostejših fizikalnih procesov uporabljamо navadne diferencialne enačbe, ki povezujejo vrednosti spremenljivk sistema z njihovimi časovnimi spremembami. Tak primer je na primer enačba za časovno odvisnost temperature v stanovanju, ki je obdano s stenami z neko toplotno prevodnostjo in določeno zunanjо temperaturo. V najpreprostejšem primeru ima enačba obliko

$$\frac{dT}{dt} = -k(T - T_{\text{zun}}) \quad (1)$$

z analitično rešitvijo

$$T(t) = T_{\text{zun}} + e^{-kt} (T(0) - T_{\text{zun}}) .$$

Enačbam, ki opisujejo razvoj spremenljivk sistema y po času ali drugi neodvisni spremenljivki x , pravimo *enačbe hoda*. Pri tej nalogi bomo proučili uporabnost različnih numeričnih metod za reševanje enačbe hoda oblike $dy/dx = f(x, y)$, kot na primer (??). Najbolj groba prva inačica, tako imenovana osnovna Eulerjeva metoda, je le prepisana aproksimacija za prvi odvod $y' \approx (y(x+h) - y(x))/h$, torej

$$y(x+h) = y(x) + h \left. \frac{dy}{dx} \right|_x . \quad (2)$$

Diferencialno enačbo smo prepisali v diferenčno: sistem spremljamo v ekvidistantnih korakih dolžine h . Metoda je večinoma stabilna, le groba: za večjo natančnost moramo ustreznno zmanjšati korak. Za red boljša ($\mathcal{O}(h^3)$, t.j. lokalna natančnost drugega reda) je simetrizirana Eulerjeva (ali sredinska) formula, ki sledi iz simetriziranega približka za prvi odvod, $y' \approx (y(x+h) - y(x-h))/2h$. Računamo po shemi

$$y(x+h) = y(x-h) + 2h \left. \frac{dy}{dx} \right|_x , \quad (3)$$

ki pa je praviloma nestabilna. Želeli bi si pravzaprav nekaj takega

$$y(x+h) = y(x) + \frac{h}{2} \left[\left. \frac{dy}{dx} \right|_x + \left. \frac{dy}{dx} \right|_{x+h} \right] , \quad (4)$$

le da to pot ne poznamo odvoda v končni točki intervala (shema je implicitna). Pomagamo si lahko z iteracijo. Zapišimo odvod kot:

$$\left. \frac{dy}{dx} \right|_x = f(x, y)$$

ter

$$x_{n+1} = x_n + h, \quad y_n = y(x_n)$$

Heunova metoda ($\mathcal{O}(h^3)$ lokalno) je približek idealne formule z:

$$\hat{y}_{n+1} = y_n + h \cdot f(x_n, y_n) \quad (5)$$

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, \hat{y}_{n+1})] \quad (6)$$

Izvedenka tega je nato Midpoint metoda (tudi $\mathcal{O}(h^3)$ lokalno):

$$k_1 = f(x_n, y_n) \quad (7)$$

$$k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}h k_1) \quad (8)$$

$$y_{n+1} = y_n + h k_2 \quad (9)$$

Le-to lahko potem izboljšamo kot modificirano Midpoint metodo itd...

V praksi zahtevamo natančnost in numerično učinkovitost, ki sta neprimerno boljši kot pri opisanih preprostih metodah. Uporabimo metode, zasnovane na algoritmih prediktor-korektor, metode višjih redov iz družine Runge-Kutta (z adaptivnimi koraki), ali ekstrapolacijske metode. Brez dvoma ena najbolj priljubljenih je metoda RK4,

$$\begin{aligned} k_1 &= f(x, y(x)) , \\ k_2 &= f\left(x + \frac{1}{2}h, y(x) + \frac{h}{2}k_1\right) , \\ k_3 &= f\left(x + \frac{1}{2}h, y(x) + \frac{h}{2}k_2\right) , \\ k_4 &= f(x + h, y(x) + hk_3) , \\ y(x + h) &= y(x) + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5) . \end{aligned} \quad (10)$$

2 Naloga

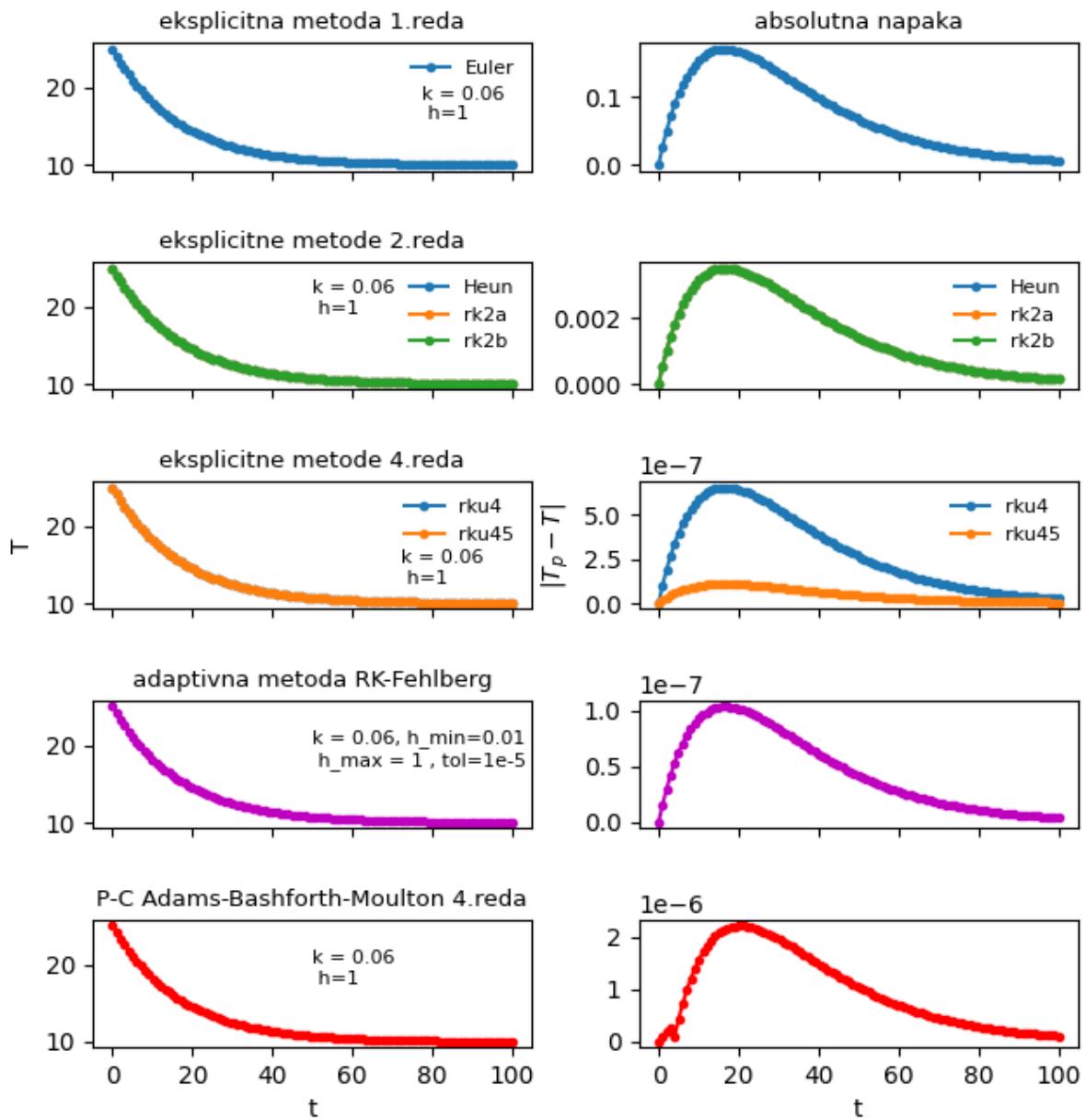
Preizkusite preprosto Eulerjevo metodo ter nato še čim več naprednejših metod(Midpoint, Runge-Kutta 4. reda, Adams-Bashfort-Moultonov prediktor-korektor ...) na primeru z začetnima temperaturama $y(0) = 21$ ali $y(0) = -15$, zunanjega temperaturo $y_{zun} = -5$ in parametrom $k = 0.1$. Kako velik (ali majhen) korak h je potreben? Izberi metodo (in korak) za izračun družine rešitev pri različnih vrednostih parametra k .

Dodatna naloga: temperatura prostora se lahko še dodatno spreminja zaradi denimo sončevega segrevanja skozi okna, s 24-urno periodo in nekim faznim zamikom δ , kar opišemo z dva- ali triparametrično enačbo

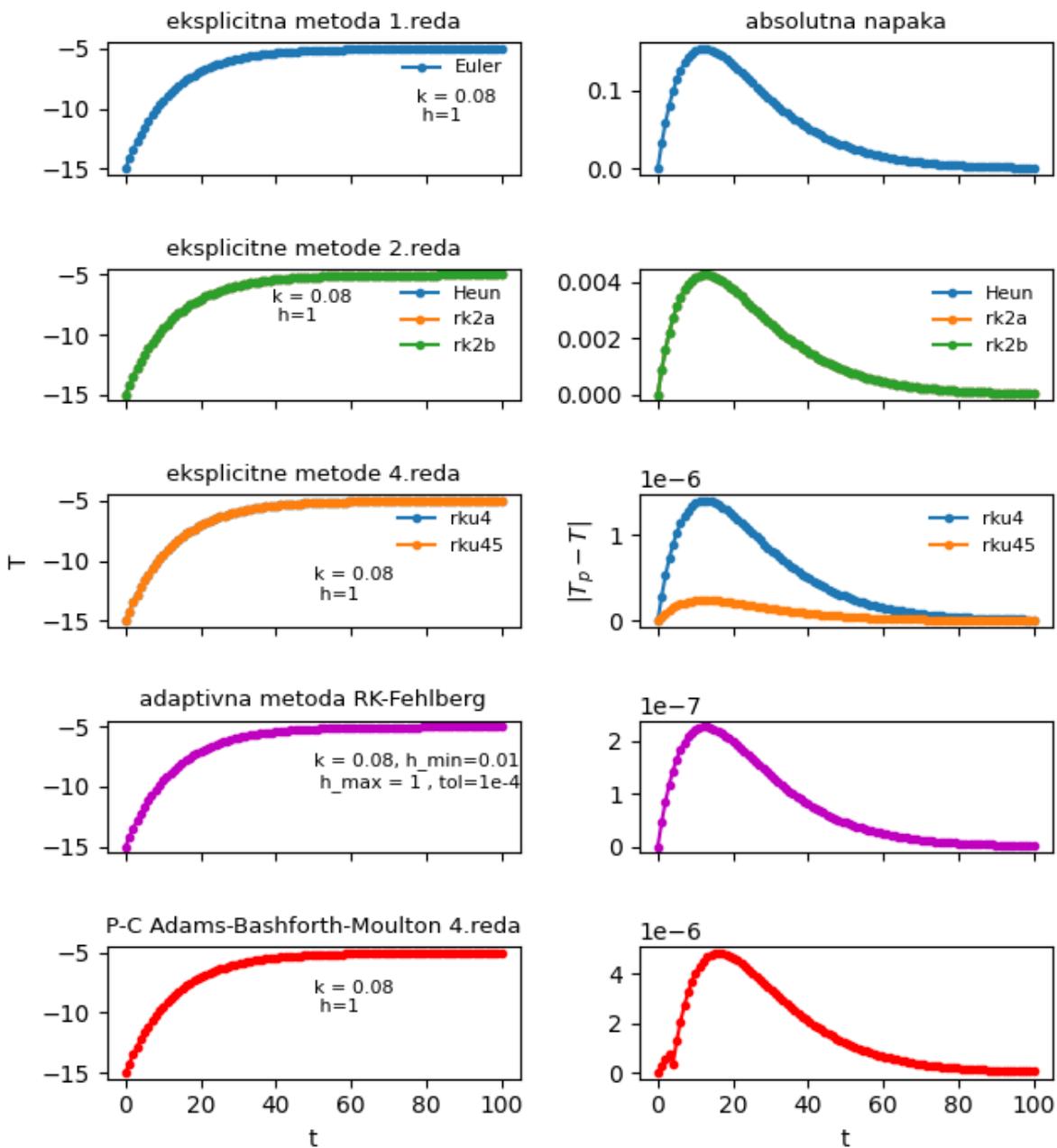
$$\frac{dT}{dt} = -k(T - T_{zun}) + A \sin\left(\frac{2\pi}{24}(t - \delta)\right) . \quad (11)$$

3 Rezultati

Rešitev enačbe 1 lahko numerično dobimo z različno natančnimi in hitrimi metodami. Poznamo eksplisitne metode več redov, kjer se število redov šteje po številu klicov funkcije za izračun naslednjega koraka, implicitne metode, kjer običajno z polinomom predvidimo naslednjo točko in potem z korekcijo izboljšamo rezultat. Posebnost je adaptivna eksplisitna metoda, kjer se velikost koraka prilagaja, da dosežemo željeno natančnost. Rešitev enačbe 1 z različnimi metodami predstavlja sliki 1 in 2. Prikazana so tudi absolutna odstopanja od prave vrednosti. Natančnost metode je odvisna od dolžine koraka h . Metode višjih redov dosežejo večjo natančnost pri enaki dolžini koraka kot metode njžjih redov. Parameter k v enačbi nam določa hitrost približevanja k asimptotki temperaturi. Zunanja temperatura nam določa samo premik v y smeri. Če je zunanjega temperatura nižja od notranje, potem imamo rešitev kot na sliki 1, če pa velja obratno potem toplota teče iz zunanjosti proti notranjosti in velja rešitev na sliki 2. Spreminjanje širine koraka in koeficiente 'dušenja' prikazujeta animaciji. Pri spremenjanju dolžine koraka opazim, da se napake z večanjem koraka večajo. Napake ohranjajo obliko do neke dolžine koraka, nato pa spremenijo obliko in metoda postane nestabilna. To prikazuje tudi primer oz. slika 3. Animacija, ki prikazuje spremenjanje koeficiente k kaže, da z večanjem vrednosti, kar bi fizikalno pomenilo, da dodajamo izolator, se zmanjševanje temperature upočasnjuje. Če je k majhen je to ekvivalentno materialu, ki dobro prevaja toploto.

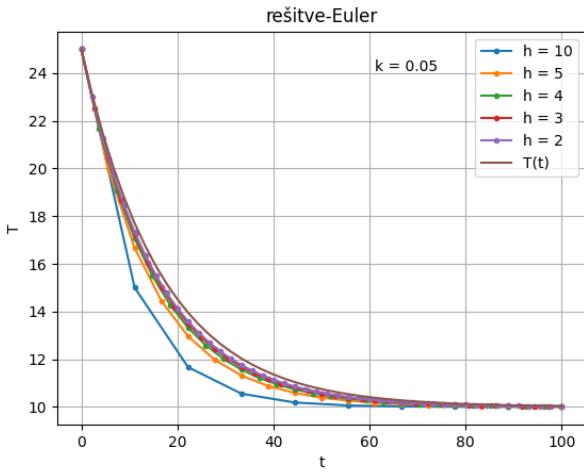


Slika 1: Numerična rešitev enačbe. $T(0) = 25$, $T_{zun} = 10$. Na grafih je parameter h velikost koraka, k koeficient prevajanja, h_{min} , h_{max} in tol pa parametri adaptivne metode, ki določajo minimalni in maksimalni korak ter maksimalno lokalno napako.

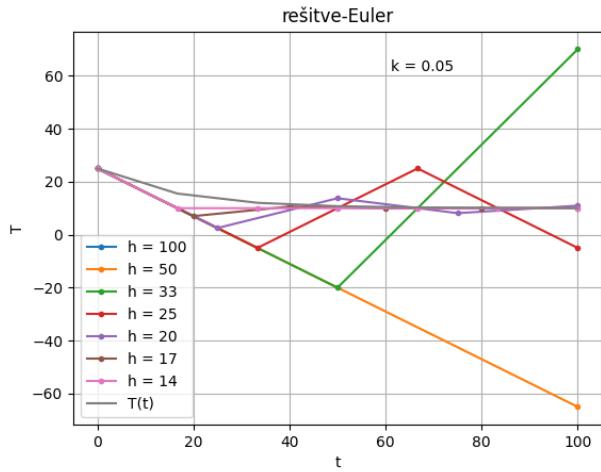


Slika 2: Numerična rešitev enačbe. $T(0) = -15$, $T_{zun} = -5$. Na grafih je parameter h velikost koraka, k koeficient prevajanja, h_{min} , h_{max} in tol pa parametri adaptivne metode, ki določajo minimalni in maksimalni korak ter maksimalno lokalno napako.

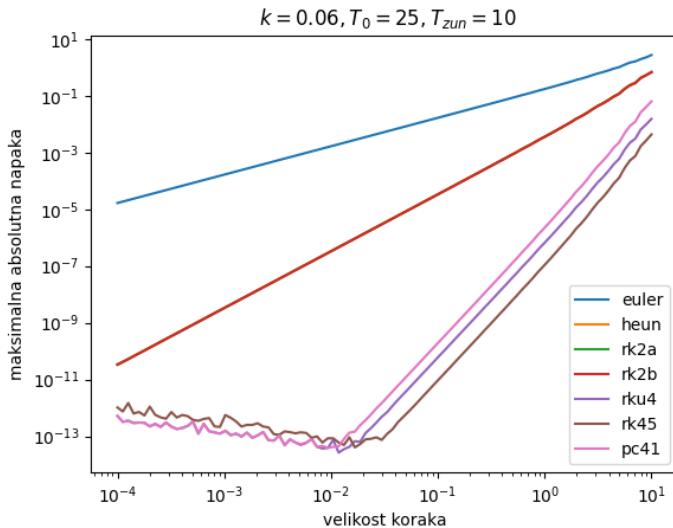
Natančnost in stabilnost metod je odvisna od velikosti koraka h . Za zgled sem vzel kar Eulerjevo metodo (slika 3 in 4) za katero velja $-2 < hk < 0$. V mojem primeru je $k = 0.05$, kar pomeni, da mora biti velikost koraka $h < 40$. Iz grafov opazim, da so nekatere rešitve pri koraku nižjem od 40 precej neuporabne. Metode pač pri določeni velikosti koraka ponorijo in postanejo neuporabne. Zanimivo se mi je zdelo tudi pogledati odstopanja metod od prave vrednosti v odvisnosti od velikosti koraka za različne metode. To lahko naredimo samo če poznamo analitično rešitev (5). Pričakoval bi, da se z zmanjševanjem koraka vedno bolj približujemo analitični vrednosti dokler ni korak okrog reda 10^{-16} in se zaradi numeričnih napak natančnost zmanjša. Iz grafa vidim, da metode višjih redov konvergirajo hitreje in dosežejo maksimalno natančnost pri relativno veliki velikosti koraka, nato pa se natančnost z zmanjševanjem koraka manjša. Za metode prvega in drugega reda pa se zdi, kot da bodo dosegle natančnost pri predvidenih vrednostih velikosti koraka.



Slika 3: Rešitev enačbe pri različnih dolžinah koraka h .



Slika 4: Rešitev enačbe pri različnih dolžinah koraka h .



Slika 5: Maksimalna absolutna napaka pri dani velikosti koraka.

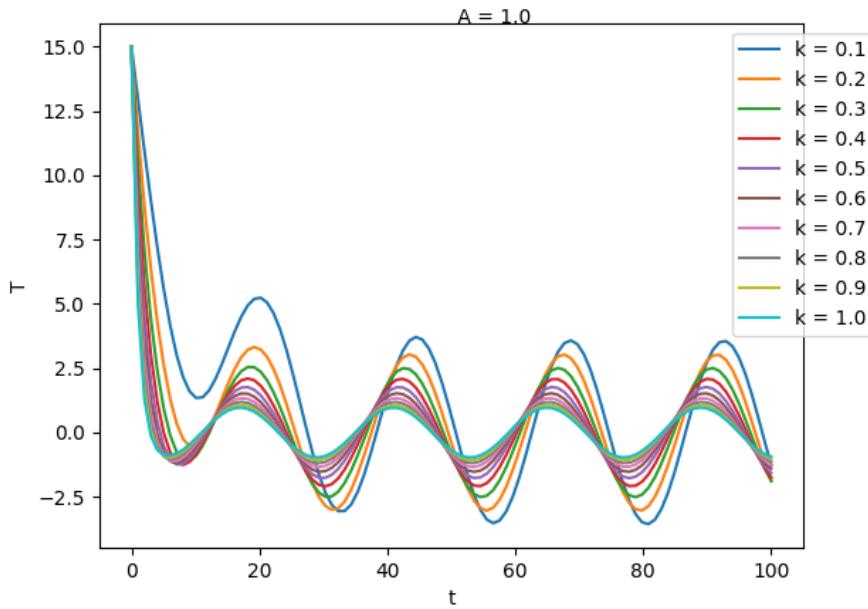
Preveril sem še časovno zahtevnost. Spreminjal sem število točk v intervalu in meril čas, ki ga porabi metoda, da pride do rešitve. Izkaže se, da ima premica zelo majhno strmino za prvih 10000 točk, zato sem vzel kar povprečje vseh časov pri danih korakih. Metode višjega reda so malce počasnejše, ampak še vedno enakega reda. Posebnost je adaptivna metoda RKF, kjer se korak prilagaja, dokler rešitev ne doeže zahtevane natančnosti, zato je tudi počasnejša.

metoda	t[s]
Euler	1.34e-04
Heun	2.23e-04
RK2a	2.33e-04
RK2b	2.23e-04
RK4	4.20e-04
RK45	8.57e-04
RKF	1.39e-03
PC4	3.37e-04

Tabela 1: Povprečni čas, ki ga potrebuje metoda za izračun rešitve.

3.1 Dodatna naloga

V diferencialni enačbi imamo še dodaten člen, ki predstavlja periodično gretje. Rešitev sem dobil z metodo RK45 (slika 6). Spreminjanje amplitudo ob različnih vrednostih parametra k sem predstavil v animaciji, kot tudi spremjanje amplitudo in faze ob danih vrednostih k .



Slika 6: Rešitev enačbe 11. $T(0) = 15$, $T_{zun} = 0$, $\delta = 0.1$.

4 Zaključek

Namen naloge je bil, da se boljše spoznam z metodami reševanja enačb hoda. Različne metode imajo svoje prednosti in slabosti, odvisno od tega kaj potrebujemo. Za natančnost so boljše metode višjega reda, za hitrost pa preprostejše in manj natančne metode. Pri iskanju rešitve je treba biti pozoren na to, da se nahajamo v območju kjer je metoda stabilna, sploh če ne poznamo analitične rešitve.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

7. naloga: Newtonov zakon

Žiga Šinigoj, 28191058

Ljubljana, december 2021

1 Uvod

Gibanje masne točke v polju sil v eni dimenziji opišemo z diferencialno enačbo drugega reda, z Newtonovim zakonom

$$m \frac{d^2x}{dt^2} = F.$$

Enačba je seveda enakovredna sistemu enačb prvega reda

$$m \frac{dx}{dt} = p, \quad \frac{dp}{dt} = F$$

in tako jo tudi rešujemo: kot sistem dveh enačb prvega reda.

Seveda morajo biti na voljo tudi ustrezeni začetni pogoji, tipično $x(t=0) = x_0$ in $dx/dt = v(t=0) = v_0$. Splošnejše gre tu za sistem diferencialnih enačb drugega reda:

$$\frac{d^n y}{dx^n} = f(x, y, y', y'', \dots),$$

ki ga lahko prevedemo na sistem enačb prvega reda z uvedbo novih spremenljivk v slogu gibalne količine pri Newtonovi enačbi ($y' = v, y'' = z, \dots$).

Z nekaj truda se da eksplicitno dokazati, mi pa lahko privzamemo, da so metode za reševanje enačb hoda (Runge-Kutta 4. reda, prediktor-korektor...) neposredno uporabne za reševanje takšnih sistemov enačb in torej aplikabilne v poljubno dimenzijah, kar naj bi v principu zadovoljilo večino naših zahtev.

Obstaja še posebna kategorija tako imenovanih *simplektičnih* metod, za enačbe, kjer je f le funkcija koordinat, $f(y)$, ki (približno) ohranjajo tudi Hamiltonian, torej energijo sistema. Najbolj znana metoda je Verlet/Störmer/Encke metoda, ki je globalno natančna do drugega reda in ki točno ohranja tudi vrtilno količino sistema (če je ta v danem problemu smiselna). Rešujemo torej za vsak diskretni korak n velikosti h , $x_n = x_0 + n \cdot h$:

$$\frac{d^2y}{dx^2} = f(y)$$

in pri diskretizaciji dobimo recept za korak y_n in $v_n = y'_n$:

$$\begin{aligned} y_{n+1} &= y_n + h \cdot v_n + \frac{h^2}{2} \cdot f(y_n) \\ v_{n+1} &= v_n + \frac{h}{2} \cdot [f(y_n) + f(y_{n+1})]. \end{aligned}$$

Alternativno lahko to shemo zapišemo tudi s pomočjo dodatnih vmesnih točk in preskakujemo med lego in hitrostjo z zamikom $h/2$ (od tod angleško ime 'leapfrog' za ta zapis):

$$\begin{aligned} y_{n+1} &= y_n + h \cdot v_{n+1/2} \\ v_{n+3/2} &= v_{n+1/2} + h \cdot f(y_{n+1}). \end{aligned}$$

V še enem drugačnem zapisu je metoda poznana tudi kot metoda "Središčne razlike" (Central Difference Method, CDM), če nas hitrost ne zanima:

$$y_{n+1} - 2y_n + y_{n-1} = h^2 \cdot f(y_n),$$

kjer prvo točko y_1 izračunamo po originalni shemi. Metodo CDM lahko uporabljam tudi za primere, ko je f tudi funkcija 'časa' x , $f(x,y)$, le da tu simplektičnost ni zagotovljena (in tudi verjetno ne relevantna). Za simplektične metode višjih redov je na voljo na primer Forest-Ruth metoda ali Position Extended Forest-Ruth Like (PEFRL) metoda, ki sta obe globalno četrtega reda in enostavni za implementacijo.

2 Naloga

Čim več metod uporabi za izračun nihanja matematičnega nihala z začetnim pogojem $\vartheta(0) = \vartheta_0 = 1, \dot{\vartheta}(0) = 0$. Poisci korak, ki zadošča za natančnost na 3 mesta. Primerjaj tudi periodično stabilnost shem: pusti, naj teče račun čez 10 ali 20 nihajev in poglej, kako se amplituda nihajev sistematično kvari. Pomagaš si lahko tudi tako, da občasno izračunaš energijo $E \propto 1 - \cos \vartheta + \frac{\dot{\vartheta}^2}{2\omega_0^2}$. Nariši tudi ustrezne fazne portrete!. Z analitično rešitvijo dobimo za nihajni čas $\frac{4}{\omega_0} K \left(\sin^2 \frac{\vartheta_0}{2} \right)$, kjer je $K(m)$ popolni eliptični integral prve vrste, ki je v SciPy knjižnici in v članku na spletni učilnici podan z:

$$K(m) = \int_0^1 \frac{dz}{\sqrt{(1-z^2)(1-mz^2)}} = \int_0^{\frac{\pi}{2}} \frac{du}{\sqrt{(1-m \sin^2 u)}}$$

Previdno, obstaja tudi definicija z m^2 v integralu - potem je prav $K \left(\sin \frac{\vartheta_0}{2} \right)$, brez kvadrata (npr že v Wikipediji)!

(Dodatno lahko tudi sprogramirate eliptični integral, ki je analitična rešitev dane enačbe ali pa ga vzamete iz ustreznih programskeh knjižnjic).

Dodatna naloga: Razišči še resonančno krivuljo vzbujenega dušenega matematičnega nihala

$$\frac{d^2x}{dt^2} + \beta \frac{dx}{dt} + \sin x = v \cos \omega_0 t,$$

kjer je β koeficient dušenja, v in ω_0 pa amplituda in frekvenca vzbujanja. Opazuj obnašanje odklonov in hitrosti nihala pri dušenju $\beta = 0.5$, vzbujevalni frekvenci $\omega_0 = 2/3$ in amplitudo vzbujanja na območju $0.5 < v < 1.5$. Poskusi opaziti histerezno obnašanje resonančne krivulje pri velikih amplitudah vzbujanja (Landau, Lifšic, CTP, Vol. 1, Mechanics).

Dodatna dodatna naloga: če ti gre delo dobro od rok, si oglej še odmike in hitrosti (fazne portrete) van der Polovega oscilatorja

$$\frac{d^2x}{dt^2} - \lambda \frac{dx}{dt} (1-x^2) + x = v \cos \omega_0 t$$

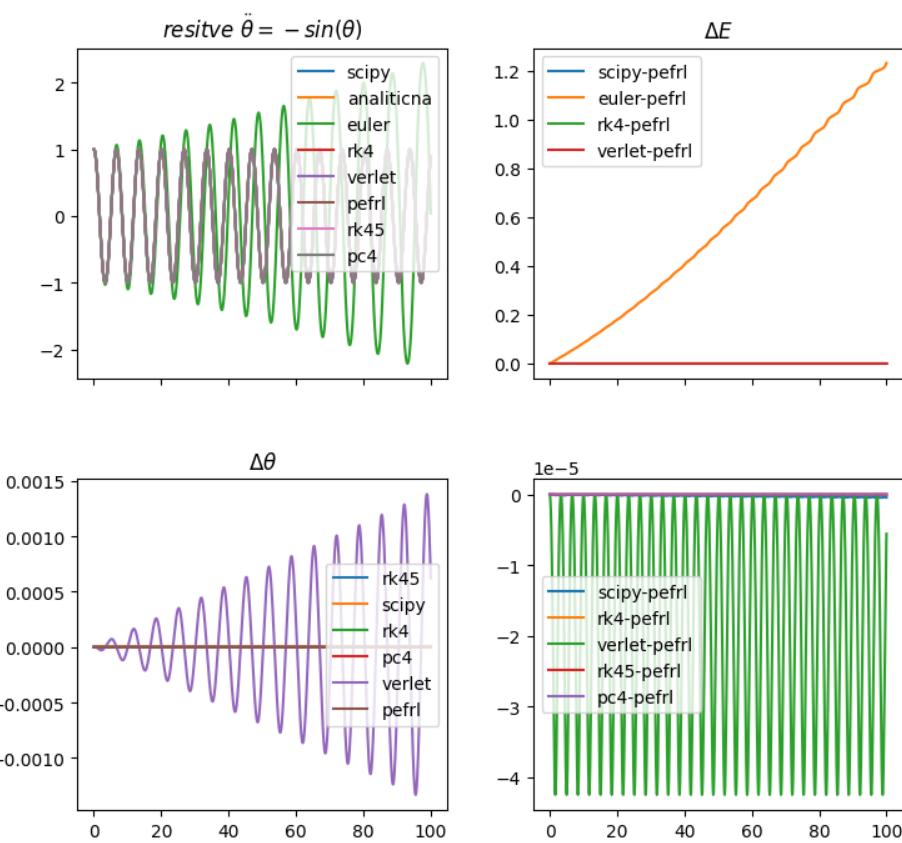
s parametri $\omega_0 = 1$, $v = 10$ ter $\lambda = 1$ ali 100. Tu se ne trudi s preprostimi diferenčnimi shemami: problem je nelinearen in tog, zato uporabi neko preverjeno metodo (na primer iz družine Runge-Kutta ali ekstrapolacijsko metodo) s prilagajanjem velikosti koraka.

3 Rezultati

3.1 Matematično nihalo

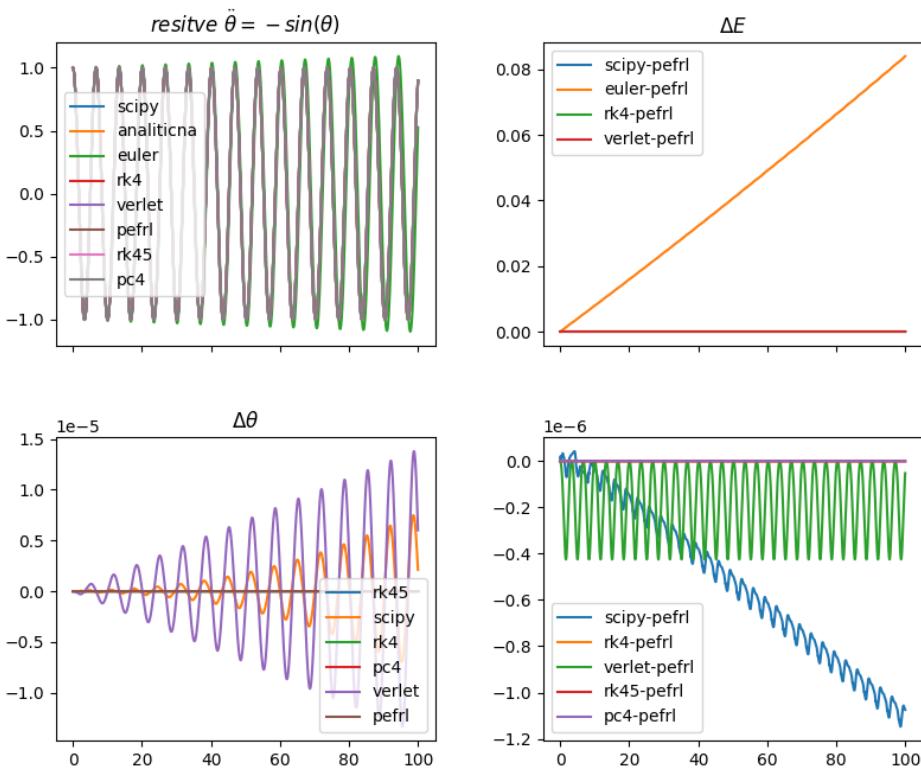
Za izračun gibanja nihala sem uporabil več metod. Začetne pogoje označuje vektor $\vec{\vartheta}$. V splošnem je natančnost odvisna od velikosti koraka in dolžine intervala na katerem iščemo rešitev. Na sliki 1 in 2, so prikazane rešitve, ki jih podajo različne metode in energije sistema, pri dveh različnih korakih. Pri energijah sem vzel za pravo energijo, energijo simplektične metode PEFRL, ki jo ohranja. To ni bilo nujno potrebno, saj lahko gledam energijo, ki jo poda dana metoda in morebitno spreminjanje s časom. To sem kasneje tudi naredil. Gre pa za eno in isto stvar, zanima nas če metoda ohranja energijo in razlika ne spremeni funkcionalnosti v času. Z zmanjševanjem koraka se natančnost metod povečuje in energija sistema se počasneje spreminja. Zanimivo je, da se pri koraku 0.002 veča napaka SciPy metode *odeint*, zato sem hotel pogledati natančnost rešitev metod v odvisnosti od velikosti koraka (slika 13). Za oceno napake na intervalu sem vzel 2. normo vektorja odstopanj leg nihala. Težava, na katero sem naletel je, da če vzamem vedno enak interval in drobim korak, ima vektor razlik vedno večjo dolžino in se napaka metode z manjšim korakom povečuje. Moral sem vzeti vedno enako število točk, oz. konstantno velikost vektorja. To povzroči, da sta velikost in dolžina koraka povezani. Z velikostjo koraka se povečuje interval in posledično tudi napaka, ki se z dolžino intervala veča. To pomeni, da se lahko krivulje na sliki 13 premikajo v vertikalni smeri, odvisno od izbrane dolžine intervala. Vseeno je videti, da se na neki točki natančnost Verlet in Odeint metode obrne.

$dt=0.02$

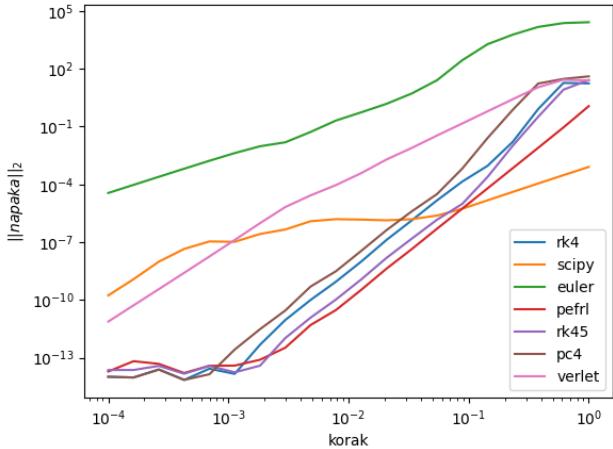


Slika 1: Rešitev nihala, energija in odstopanja. $\vec{\vartheta}(0) = [1, 0]$

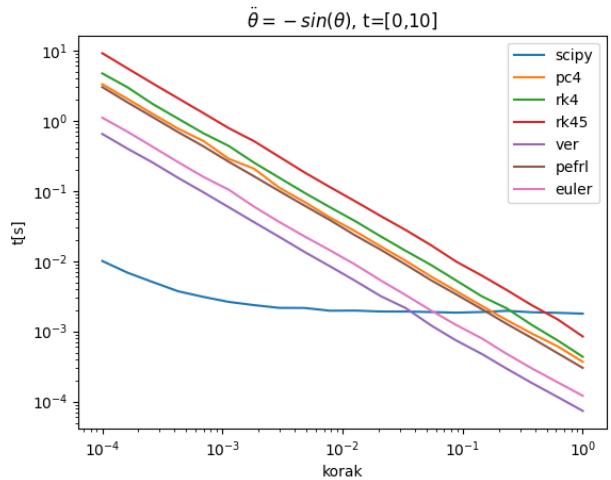
$dt=0.002$



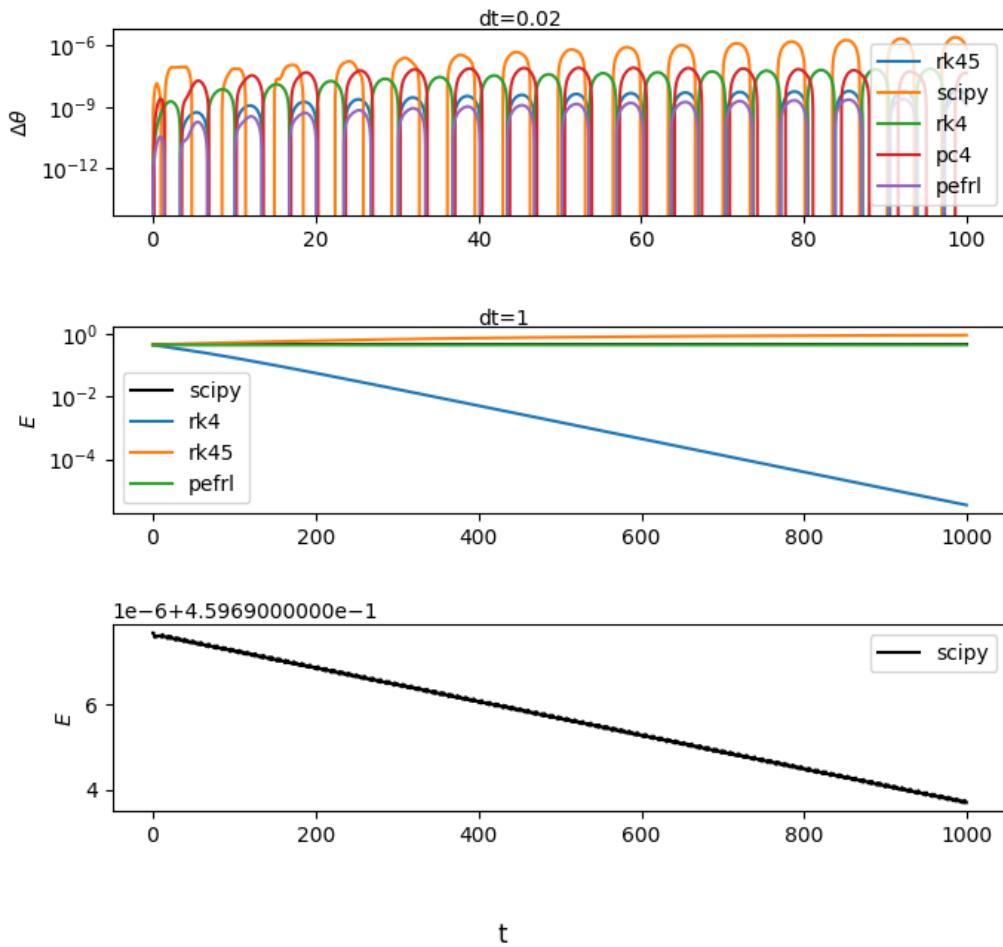
Slika 2: Rešitev nihala, energija in odstopanja. $\vec{\vartheta}(0) = [1, 0]$



Slika 3: Odstopanje metode v odvisnosti od velikosti koraka.

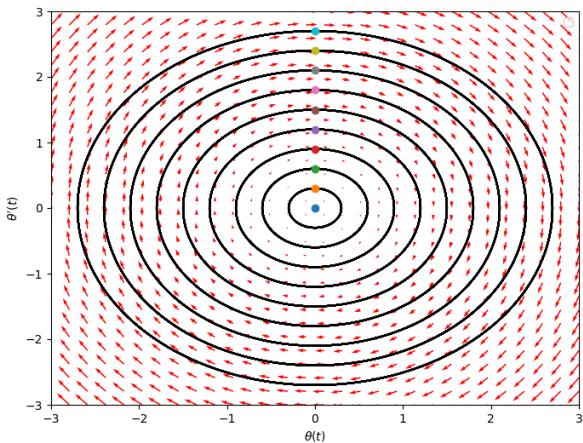


Slika 4: Časovna zahtevnost

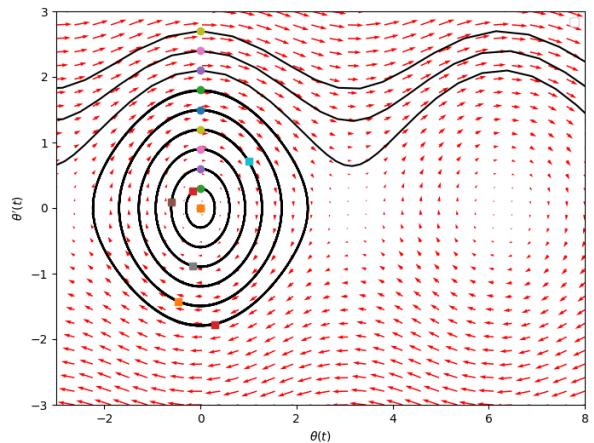


Slika 5: Maksimalna absolutna napaka pri dani velikosti koraka.

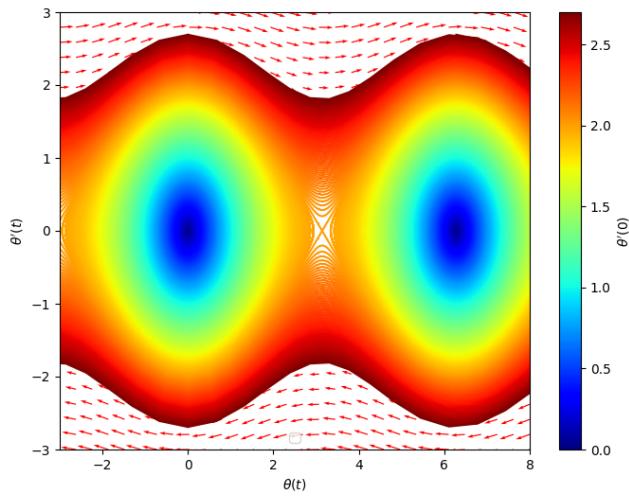
Časovna zahtevnost (slika 4) je eksponentna razen pri SciPy metodi, ki je najhitrejša. Natančnost metod še v logaritemski skali prikazuje slika 5. Energijo izgubljajo Runge-Kutta metode in Scipyjeva metoda, čeprav jo slednja zelo počasi. Simplektične metode (PEFRL, Verlet) so edine, ki ohranajo energijo, ponavadi nihajo okrog oz. ob energiji sistema. Na faznih diagramih (slika 14, 8) predstavljajo okople točke začetne vrednosti, kvadratne pa končne.



Slika 6: Fazni diagram linearnega nihala.



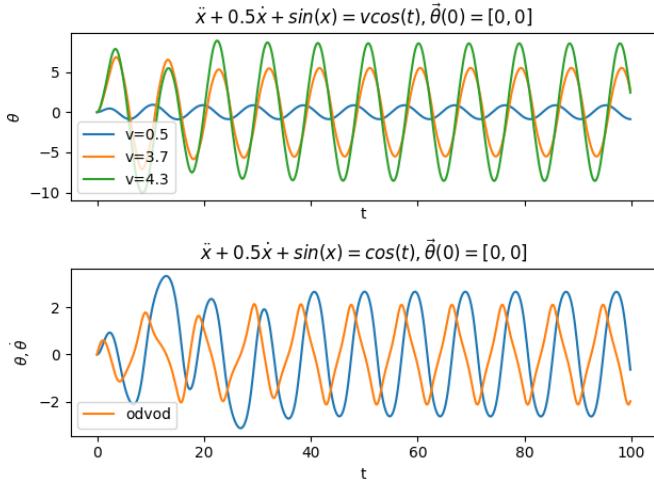
Slika 7: Fazni diagram matematičnega nihala.



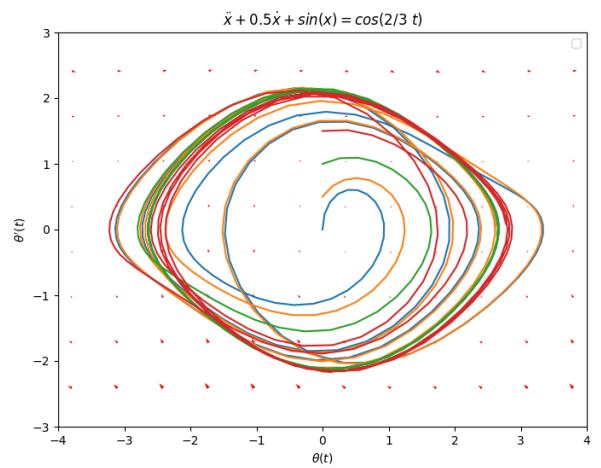
Slika 8: Fazni diagram matematičnega nihala z umetniškim navdihom.

3.2 Dušeno matematično nihalo z vzbujanjem

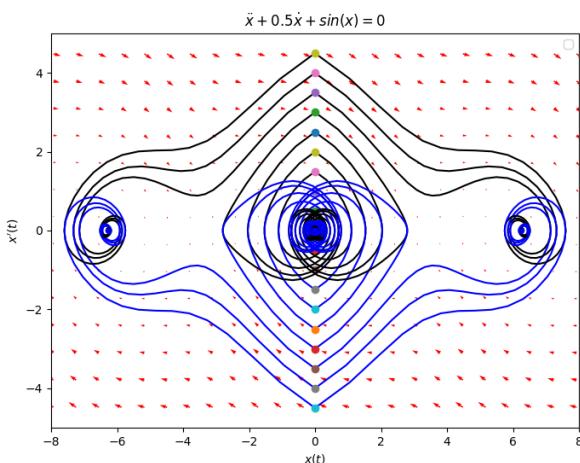
Za izračun sistema sem uporabil SciPyjevo funkcijo `integrate.odeint()`. Ker metoda ne ohranja energije sem moral pri faznih diagramih paziti, da nisem vzel veliko period nihanja. Če bi vzel veliko period bi dobil fazni diagram, ki bi izgubljal energijo in krivulje bi se spremenjale. To lahko privede do napačne interpretacije rezultatov. Fazni diagrami so periodični s periodo 2π .



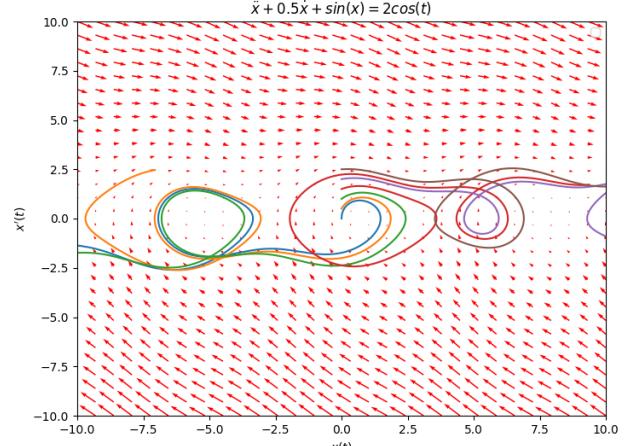
Slika 9: Rešitev matematičnega nihala z vzbujanjem in nekaj resonančnih krivulj.



Slika 10: Fazni diagram dušenega matematičnega nihala z vzbujanjem, pri različnih začetnih pogojih.



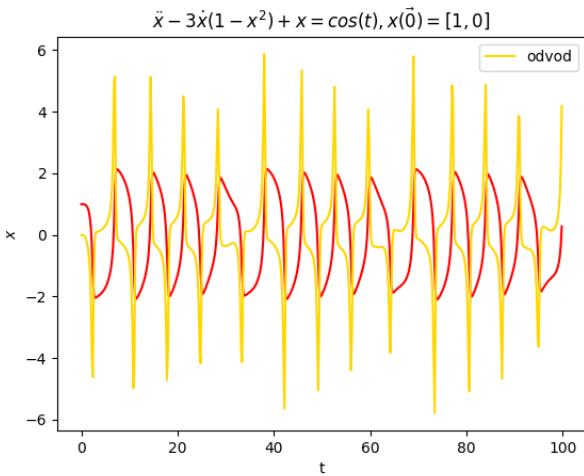
Slika 11: Fazni diagram dušenega matematičnega nihala, pri različnih začetnih pogojih.



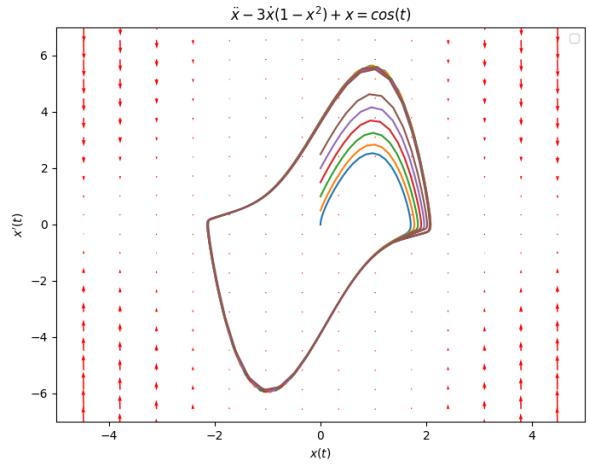
Slika 12: Fazni diagram dušenega matematičnega nihala s prevelikim vzbujanjem, pri različnih začetnih pogojih.

3.3 Van der Polov oscilator

Za izračun sistema sem uporabil SciPyjevo funkijo `integrate.odeint()`.



Slika 13: Rešitev Van der Polovega nihala.



Slika 14: Fazni diagram, pri različnih začetnih vrednostih.

4 Zaključek

Pri reševanju diferencialnih enačb z začetnimi pogoji so obravnavane metode, razen Eulerjeve, precej natančne. Vedno pa je treba preverjati ali so rezultati smiselnii za dani fizikalni problem. Pri novih fizikalnih problemih, kjer rezultata ne poznamo, je dobro uporabiti tudi simplektične metode, ki ohranjajo energijo sistema. S tem se izognemo morebitni napačni interpretaciji rešitve.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

8. naloga: Robni problem lastnih vrednosti

Žiga Šinigoj, 28191058

Ljubljana, december 2021

1 Uvod

Pri robnem problemu lastnih vrednosti poznamo diferencialno enačbo in nekaj robnih pogojev (običajno vsaj toliko, kolikor je red enačbe) Za rešitev problema moramo v splošnem v enem zamahu določiti tako (lastne) funkcije, ki ustrezajo danim robnim pogojem, kot (lastne) vrednosti, ki skupaj zadoščajo diferencialni enačbi. Reševanje robnih problemov je zato lahko bistveno bolj zapleteno kot integracija začetnih problemov.

Numerično bomo reševali stacionarno Schrödingerjevo enačbo

$$-\frac{\hbar^2}{2m} \frac{d^2\psi}{dx^2} + V(x)\psi = E\psi$$

za neskončno potencialno jamo ($V(-a/2 < x < a/2) = 0$ in $V(|x| \geq a/2) \rightarrow \infty$) ter za končno potencialno jamo ($V(|x| \geq a/2) = V_0$), za kateri poznamo analitične rešitve; glej Strnad, *Fizika II*. Dva značilna pristopa, diferenčna metoda in strelska metoda, nas bosta pripravila na resnejše probleme, za katere analitičnih rešitev ne poznamo.

Pri *diferenčni metodi* razdelimo interval $[-a/2, a/2]$ na N točk ($x_i = -a/2 + ia/N$) in prepišemo drugi krajevni odvod v drugo differenco, tako da ima brezdimenzijska enačba obliko

$$\frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{h^2} + E\psi_i = 0$$

oziroma

$$\psi_{i-1} - (2 - \lambda)\psi_i + \psi_{i+1} = 0,$$

kjer je $\lambda = Eh^2 = k^2 h^2$. Diskretizirati je treba tudi robna pogoja pri $x = -a/2$ in $x = a/2$, ki sta v splošnem (in tudi pri končni jami) mešanega tipa,

$$\begin{aligned} c_1\psi_0 + c_2 \frac{\psi_1 - \psi_{-1}}{2h} &= 0, \\ d_1\psi_N + d_2 \frac{\psi_{N+1} - \psi_{N-1}}{2h} &= 0, \end{aligned}$$

medtem ko sta pri neskončni jami preprostejša, $\psi_0 = \psi_N = 0$. V primerih potencialnih jam tako dobimo tridiagonalni sistem N oziroma $N - 1$ linearnih enačb

$$A\underline{\psi} = \lambda\underline{\psi}$$

za lastne vektorje $\underline{\psi}$ in lastne vrednosti λ , ki ga rešujemo z diagonalizacijo.

Pri *strelskej metodi* začnemo s "kosinusnim" začetnim pogojem v izhodišču $\psi(0) = 1$, $\psi'(0) = 0$ ali "sinusnim" pogojem $\psi(0) = 0$, $\psi'(0) = 1$, nato pa z nekim izbranim E diferencialno enačbo s poljubno integracijsko shemo (npr. RK4) integriramo do roba $x = a/2$ in tam preverimo, ali je izpolnjen drugi robeni pogoj, $\psi(a/2) = 0$. Vrednost E spremojamo tako dolgo, dokler robeni pogoj ni izpolnjen do zahtevane natančnosti, in tako dobimo sode in lihe rešitve enačbe skupaj z ustreznimi lastnimi vrednostmi energije.

2 Naloga

Določi nekaj najnižjih lastnih funkcij in lastnih vrednosti za neskončno potencialno jamo z diferenčno metodo in metodo streljanja, lahko pa poskusиш še iterativno in s kakšno drugo metodo. Problem končne jame je s strelske metodo le trivialna posplošitev problema neskončne jame: spremeni se le robeni pogoj pri $x = a/2$, ki ima zaradi zahteve po zveznosti in zvezni odvedljivosti valovne funkcije zdaj obliko $c_1\psi(a/2) + c_2\psi'(a/2) = 0$. Kaj ima pri diferenčni metodi večjo vlogo pri napaki: končna natančnost difference, s katero aproksimiramo drugi odvod, ali zrnatost intervala (končna razsežnost matrike, ki jo diagonaliziramo)?

3 Rezultati

3.1 Neskončna potencialna jama

Ker je problem simetrijski sem reševal samo za pozitivno polravnino in potem prezrcalil sodo ali liho rešitev na negativno polravnino. Za končno jamo so rešitve sinusi oz. kosinusi. Za energijo sem vzel kar $E_n = n^2$.

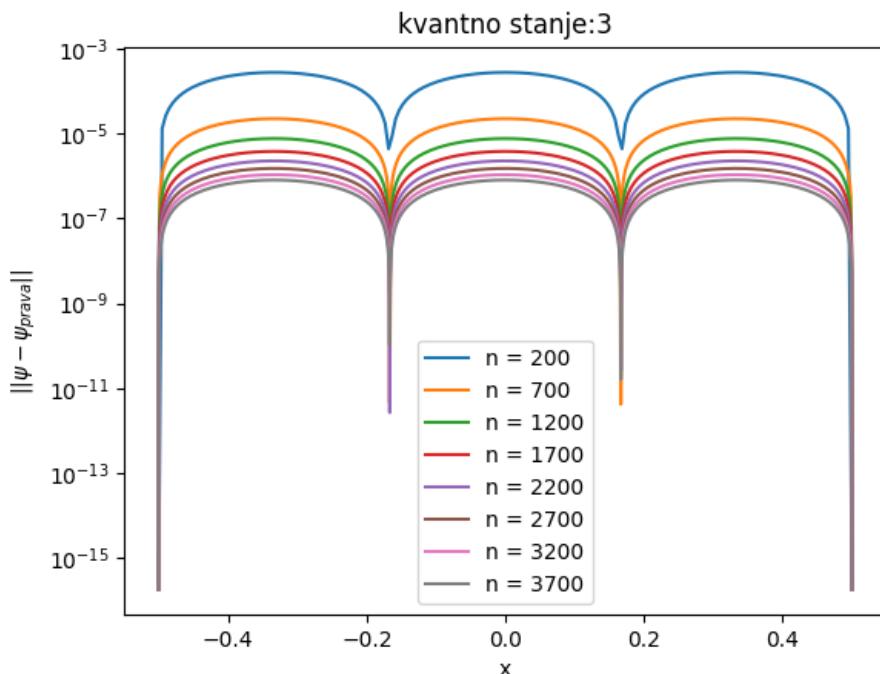
3.1.1 Implementacija strelske metode

Kot integrator sem uporabil Runge-Kutta 4 metodo z začetnimi pogoji $[1,0]$ in $[0, 1]$ za sode in lihe rešitve potencialne jame. Za dano energijo sem izračunal rešitev do roba jame, nato sem pogledal ali zadošča robnemu pogoju $\psi(a/2) = 0$. Uporabil sem bisekcijo, ki je za dani interval energij pregledala zadnje točke rešitev in vrnila tisto vrednost energije, za katero je zadnja točka poljubno blizu ničle-v mojem primeru $\epsilon = 10^{-7}$. Če bil interval med energijama manjši od natančnosti pa ni nujno, da je imela zadnja točka rešitve takšno natančnost. Ko sem dobil energije sem z RK4 poiskal rešitve oz. lastne funkcije jame (slika 2.)

3.1.2 Implementacija diferenčne metode

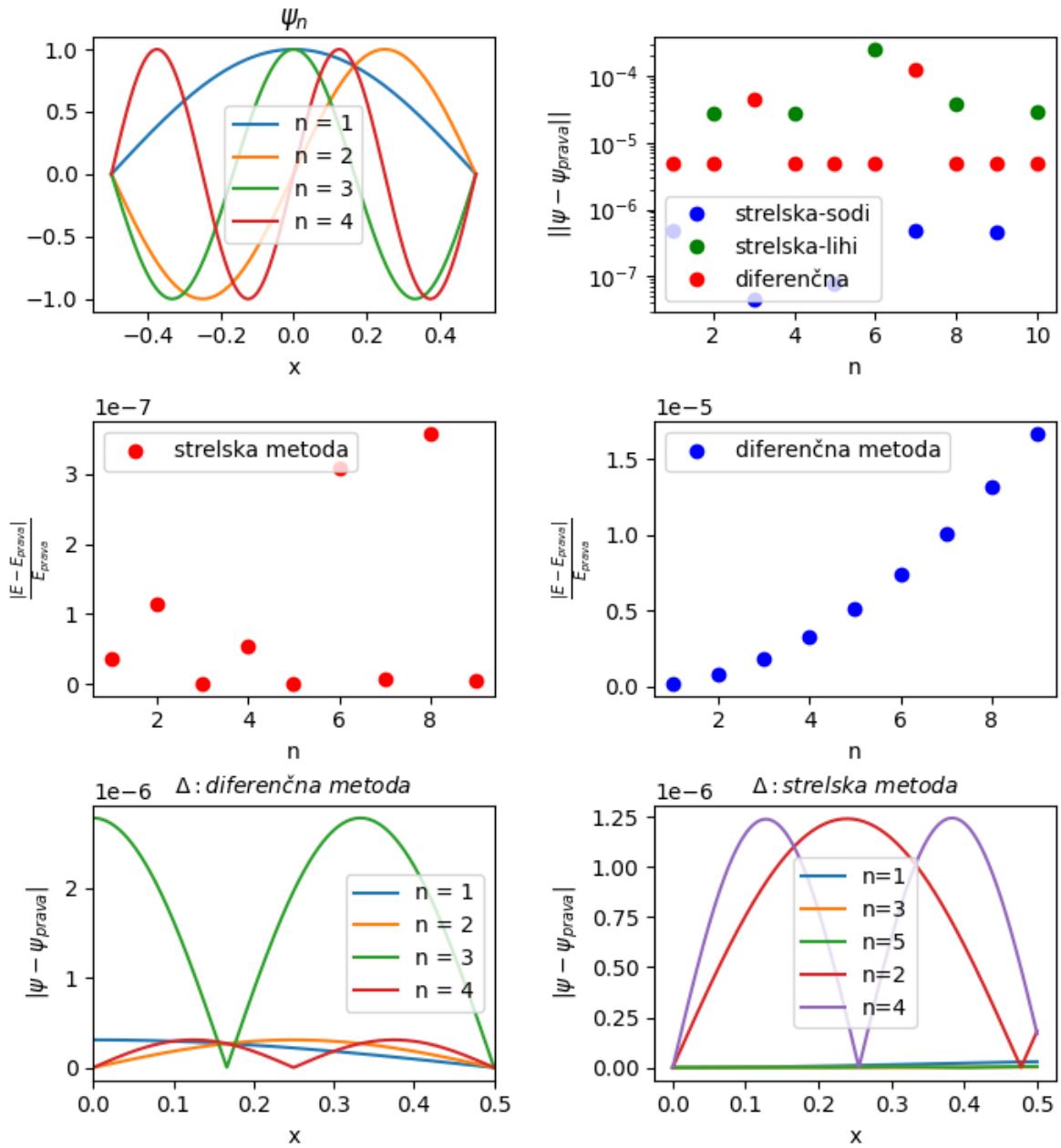
Pri metodi končnih differenc sem naredil kvadratno matriko velikosti, ki je enaka številu točk na mreži intervala. Za mrežo intervala sem vzel 2000 točk na intervalu $x \in [-0.5, 0.5]$. Pri diferenčni metodi dobimo naenkrat vse lastne funkcije in lastne vrednosti. Matrika je v našem primeru tridiagonalna z -2 na diagonali in 1 na obeh obdiagonalah. Vektorjem, sem dodal ničlo na koncu in začetku, tako sem upošteval robne pogoje. Lastne vrednosti in funkcije sem dobil z metodo *linalg.eigh()* iz paketa NumPy (slika 2).

Pogledal sem tudi odstopanje diferenčne metode v odvisnosti od velikosti matrike oz. številu točk na mreži pri določenem kvantnem stanju (slika 1). Natančnost se z drobljenjem mreže pričakovanovno izboljšuje. Opazim lahko, da na natančnost vpliva tudi približek drugega odvoda, ki ga uporabimo za rešitev sistema. Iz grafa je razvidno, da se napaka povečuje, ko se oddaljujemo od ničel lastne funkcije in narašča hitreje, ko se spreminja odvod funkcije. Če povečamo število točk na intervalu za 16 krat ($n : 200 \rightarrow 3200$), se maksimalna napaka zmanjša za nekaj manj kot tisoč krat. Sprememba napake med maksimalno in napako večine točk je nekje do 10-kratnika, pri $n = 3200$, razen za točke blizu ničle. Zrnatost intervala več prispeva k napaki, kljub temu pa je prispevek zapisa drugega odvoda nezanemarljiv.



Slika 1: Odstopanja diferenčne metode od števila točk na intervalu.

∞ potencialna jama



Slika 2: Od leve proti desni: prve 4 lastne funkcije, odstopanje rešitev po drugi normi, odstopanje energij pri strelske in diferenčni metodi, odstopanje funkcij od lastnih za diferenčno in strelsko metodo.

Iz slike 2 je razvidno, da je strelska metoda precej natančnejša pri določitvi lastnih energij sistema. Pri diferenčni metodi se relativna napaka povečuje z večanjem kvantnega števila oz. stanja. Zanimivo je tudi videti, da imajo antisimetrična stanja nekoliko večjo napako, vsaj pri strelski metodi, tako po normi kot po absolutni vrednosti razlike. Mogoče je to posledica začetnega pogoja.

3.2 Končna potencialna jama

3.2.1 Rešitev transcendentne enačbe

Za 'analitično' rešitev sem vzel energije, ki mi jih data transcendentni enačbi. Enačbi sem reševal z metodo `optimize.brentq` iz knjižnice SciPy, ki uporablja sekantno metodo in bisekcijo za iskanje ničel. Izkazalo se je, da lastne energije, ki mi jih da ta način niso prav zelo natančne. Na sliki 4 se vidi, da se rešitev za osnovno stanje v logaritemski skali približa ničli, a hitro ponori.

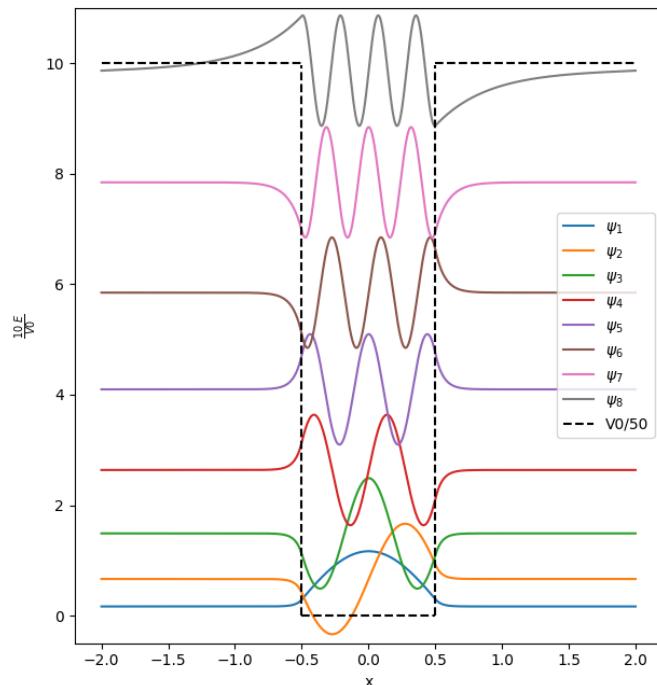
3.2.2 Implementacija strelske metode

Pri implementaciji sem najprej poskušal dobiti rešitev za območje z potencialno jamo in potem na robu podal vrednosti rešitve in odvoda za RK4, ki je potem reševal drugo diferencialno enačbo z bisekcijo vrednosti zadnje točke, ki mora biti poljubno blizu ničli ($\epsilon = 1e^{-8}$). Ta način ni bil najboljši, saj se pri tem načinu spreminja samo druga rešitev in ta ni neodvisna od prve rešitve.

V delujoči implementaciji sem v integrator RK4 vnesel pogoj, da integrator rešuje eno diferencialno enačbo za del v jami in drugo diferencialno enačbi za območje izven jame. Rešitev sem iskal samo na desni polravnini zaradi simetrije problema. Zadnjo točko rešitve sem z bisekcijo, kjer enako kot v pri neskončni jami, spreminal interval energije in poiskal tako, da je zanja točka rešitve poljubno blizu ničli.

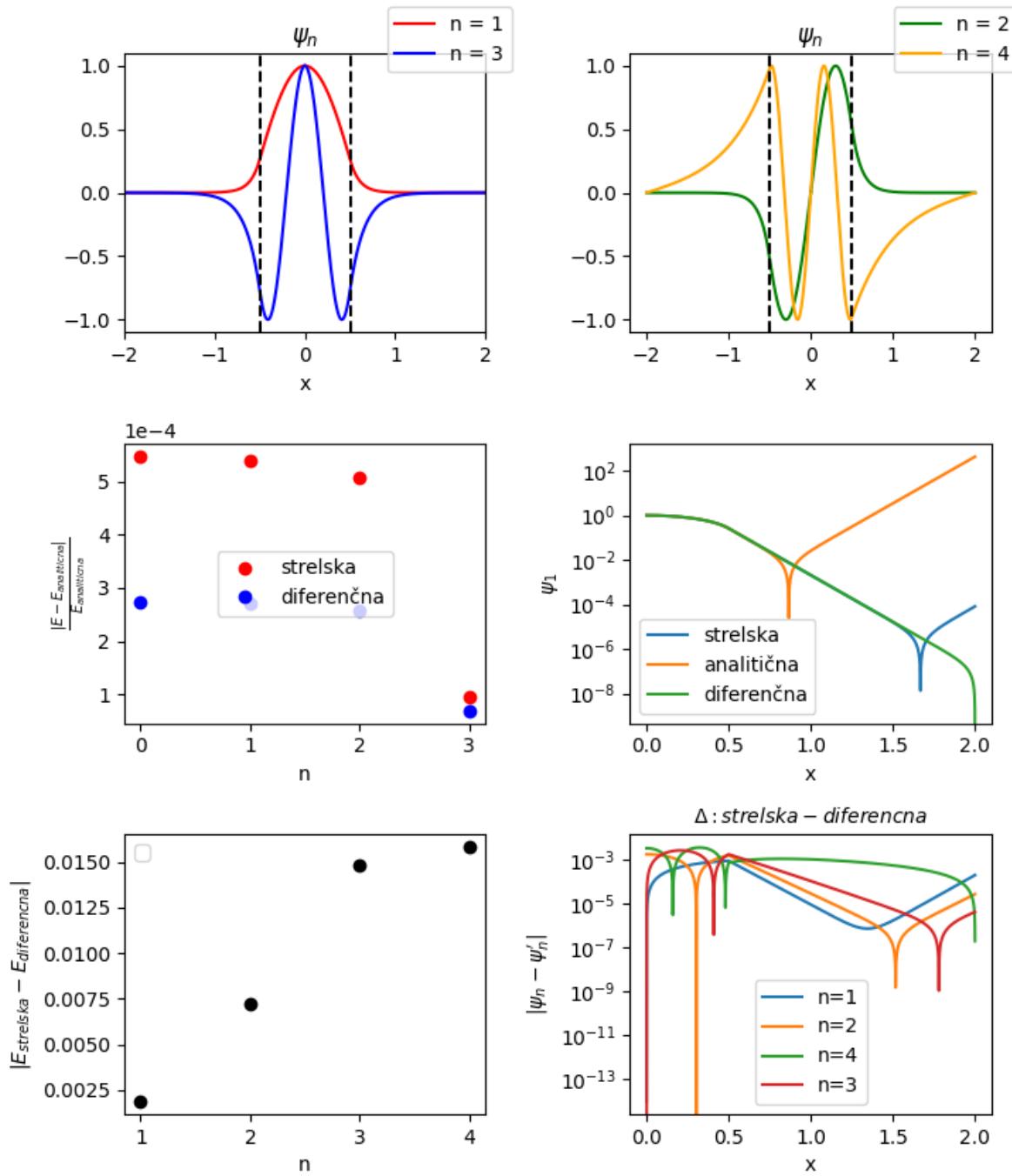
3.2.3 Implementacija diferenčne metode

Pri diferenčni metodi sem skonstruiral kinetični del matrike, ki je za 2 dimenziji manjša od intervala. Ta je enaka kot pri neskončni potencialni jami. Potencialna matrika, enake dimenzijs, vsebuje na diagonali vrednosti potenciala na mestih, kjer so koordinate izven jame, ničle pa na mestih, ki so v jami. Matriki sem nato seštel in z enako metodo poiskal lastne vrednosti in vektorje sistema. Na koncu sem dodal vektorjem ničlo na koncu in začetku, da sem upošteval robne pogoje. Diferenčna metoda nam da vse lastne vrednosti in vektorje naenkrat.



Slika 3: Normalizirane rešitve končne potencialne jame pri danih energijah v potencialu $V_0 = 500$ in št. točk na mreži $N = 6000$ z diferenčno metodo.

končna potencialna jama



Slika 4: Od leve proti desni: Rešitev s strelskej metódy za sode in lihe lastne funkcie, relativná razlika energij, rešitev prvega stanja v log. skali, razlika lastnih energij med strelskej in diferenčnej metódou, razlika po absolutnej vrednosti med rešitvijo strelskej in diferenčnej metód. $V_0 = 100$.

Na sliki 4 ima liha rešitev $n = 4$ velike repe, zato ker je lastna energija blizu potencialni energiji in stanje ni močno vezano. Kot sem omenil, so 'analiticne' rešitve manj natančne, kar je razvidno pri osnovni rešitvi. Diferenčna metoda se približuje ničli do konca danega intervala. Strelska metoda doseže maksimalno natančnost pred koncem intervala in potem začne počasi divergirati. Kljub temu, da ni bil izpolnjen pogoj o predpisani natančnosti funkcije okrog ničle, je bil očitno interval med energijama tako majhen, da je to prva lastna energija. V neskončni potencialni jami je videti, da lastne energije pri diferenčni metodi odstopajo z večanjem kvantnega števila. Kot kaže, velja enako za končno potencialno jamo.

4 Zaključek

Pri reševanju robnih problemov je potrebno paziti na pravilno implemetacijo, saj lahko manjše napake privedejo do rezultatov, ki ne nujno izgledajo tako napačni. Prednost strelske metode je v boljši določitvi lastne energije in posledično lastne funkcije. Prednost diferenčne metode je v tem, da z rešitvijo sistema naenkrat dobimo vse lastne vrednosti in vektorje, vendar na njeno natančnost vpliva približek končnih differenc.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

9. naloga: Spektralne metode za začetne probleme PDE

Žiga Šinigoj, 28191058

Ljubljana, december 2021

1 Uvod

Za reševanje začetnih problemov s parcialnimi diferencialnimi enačbami (PDE) imamo na voljo dva obsežna razreda metod. Pri *diferenčnih metodah* na neki način aproksimiramo časovne in krajevne parcialne odvode s končnimi diferencami. Reševanje PDE nato prevedemo na reševanje algebrajskih enačb ali sistemov enačb za približne vrednosti funkcij, ki v teh diferencah nastopajo. Diferenčne metode spoznamo pri naslednji vaji. Pri tej vaji obravnavamno *spektralne metode*: pri njih iskano rešitev formalno izrazimo z nekim naborom funkcij, nato pa v času spremljamo koeficiente v takem razvoju. Kako se selimo med krajevno in časovno sliko problema, je odvisno od posamezne metode. Osnovne prijeme spoznamo ob Fourierovi metodi in metodi končnih elementov s kubičnimi *B*-zlepki (*B*-splines).

Fizikalno ozadje naj bo enorazsežna difuzijska enačba, ki opisuje na primer temperaturno polje $T(x, t)$ v homogeni neskončni plasti s končno debelino a brez izvorov topote:

$$\frac{\partial T}{\partial t} = D \frac{\partial^2 T}{\partial x^2}, \quad 0 \leq x \leq a, \quad D = \frac{\lambda}{\rho c}.$$

Temperaturo v poljubni točki x ob času t izrazimo s Fourierovo vrsto

$$T(x, t) \simeq \sum_{k=0}^{N-1} \tilde{T}_k(t) e^{-2\pi i f_k x},$$

kjer je $f_k = k/a$, torej

$$\sum_k \frac{\partial \tilde{T}_k(t)}{\partial t} e^{-2\pi i f_k x} = D \sum_k (-4\pi^2 f_k^2) \tilde{T}_k(t) e^{-2\pi i f_k x}.$$

Od tod sledi evolucijska enačba za Fourierove koeficiente

$$\frac{\partial \tilde{T}_k(t)}{\partial t} = D (-4\pi^2 f_k^2) \tilde{T}_k(t). \quad (1)$$

Pogosto uporabimo spektralno reprezentacijo za krajevni odvod, medtem ko časovni korak naredimo z neko eksplicitno integracijsko shemo, na primer kar po Eulerju

$$\tilde{T}_k(t+h) = \tilde{T}_k(t) + hD(-4\pi^2 f_k^2) \tilde{T}_k(t). \quad (2)$$

Reprezentacijo $T(x, t)$ v običajnem prostoru nato dobimo z obratno Fourierovo transformacijo.

Tu lahko v splošnem časovni korak izvedeš po Eulerju, v tem konkretnem primeru pa obstaja tudi enostavna analitična rešitev enačbe 1, ki jo lahko uporabiš za primerjavo. V numerični metodi tako najprej izračunaj Fourierovo reprezentacijo $\tilde{T}_k(0)$ začetnega pogoja, nato pa jo po Eulerju evolviraj v času. Pri tem moraš paziti na stabilnost Eulerjeve diferenčne sheme: pri katerem koli koraku mora veljati

$$\left| \frac{\tilde{T}_k(t+h)}{\tilde{T}_k(t)} \right| = |1 + hD(-4\pi^2 f_k^2)| < 1.$$

Nekaj pozornosti zahteva tudi diskretizacija: za vsak k seveda velja $-f_{Nyquist} < f_k < f_{Nyquist}$ in s tem povezan možen pojav *aliasinga* (Kaj je že to?). Ta pojav lahko študiraš, če se spomniš, da obstaja analitična rešitev FT Gaussove funkcije (je spet Gaussova funkcija) - kaj se z le-to dogaja znotraj dovoljenega frekvenčnega intervala? Če izbereš veliko število točk, je seveda smiselnouporabiti kar algoritmom FFT. Temperaturni profil $T_j(t) \equiv T(x, t)$ ob poljubnem času nato dobiš z inverzno FFT.

Pri razvoju $T(x, t)$ nismo omejeni na trigonometrične funkcije. Rešitev PDE na $0 \leq x \leq a$ lahko aproksimamo tudi z drugačno vrsto funkcij, na primer kubičnimi *B*-zlepki,

$$T(x, t) = \sum_{k=-1}^{N+1} c_k(t) B_k(x), \quad (3)$$

kjer je $B_k(x)$ kubični zlepek s središčem okrog $x = x_k$. Aastnosti B -zlepkov so navedene v dodatku. Tako zasnujemo *metodo končnih elementov*, s *kolokacijskim pogojem*, da naj se zlepek ujema z rešitvijo v določenih izbranih točkah. Podobno kot pri Fourierovi metodi tudi pri tej metodi zahtevamo, da razvoj (3) zadošča osnovni PDE in robnim pogoju. Razvoj (3) vstavimo v PDE in izvrednotimo rezultat pri $x = x_j$. (Interval $[0, a]$ diskretiziramo na N podintervalov širine Δx s točkami $x_j = j\Delta x$, kjer je $j = 0, 1, \dots, N$. Za kolokacijo je smiselno izbrati enake točke kot za diskretno mrežo.) Tako dobimo

$$\sum_{k=-1}^{N+1} \dot{c}_k(t) B_k(x_j) = D \sum_{k=-1}^{N+1} c_k(t) B''_k(x_j), \quad j = 0, 1, \dots, N.$$

Upoštevamo lastnosti B -zlepkov in dobimo sistem diferencialnih enačb za koeficiente $c_j(t)$:

$$\dot{c}_{j-1}(t) + 4\dot{c}_j(t) + \dot{c}_{j+1}(t) = \frac{6D}{\Delta x^2} (c_{j-1}(t) - 2c_j(t) + c_{j+1}(t)),$$

kjer je $j = 0, 1, \dots, N$. Iz robnega pogoja pri $x = 0$ ugotovimo $c_{-1} = -4c_0 - c_1$. Če dodamo še zahtevo za 'naravnii' kubični zlepek, da je na robu $\sum_{k=-1}^{N+1} c_k(t) B''_k(x = (0, a)) = 0$, sledi $c_0 = c_N = 0$ in $c_{-1} = -c_1$ ter $c_{N-1} = -c_{N+1}$. Reševanje enačbe (3) smo torej prevedli na reševanje matričnega sistema

$$\mathbf{A} \frac{d\vec{c}}{dt} = \mathbf{B} \vec{c},$$

kjer je

$$\mathbf{A} = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \vdots & & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{pmatrix}, \quad \mathbf{B} = \frac{6D}{\Delta x^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \vdots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}$$

in $\vec{c} = (c_1(t), c_2(t), \dots, c_{N-1}(t))^T$. Začetni pogoj za PDE je $T(x_j, 0) = f(x_j)$, torej je začetni približek za kolokacijsko aproksimacijo

$$\mathbf{A} \vec{c}^0 = \vec{f},$$

kjer je $\vec{f} = (f(x_1), f(x_2), \dots, f(x_{N-1}))^T$. To zdaj rešujemo s kako metodo, ki jo poznamo iz prejšnjih nalog, recimo z eksplicitno Eulerjevo metodo: ob zaporednih časih $n\Delta t$ dobimo

$$\vec{c}^{n+1} = \vec{c}^n + \Delta t \mathbf{A}^{-1} \mathbf{B} \vec{c}^n = (1 + \Delta t \mathbf{A}^{-1} \mathbf{B}) \vec{c}^n.$$

Ob poljubnem času nato dobimo temperaturni profil tako, da znova izračunamo vsoto (3). Ker nam je že znano, da je Eulerjeva ob predolgih časovnih korakih lahko nestabilna, lahko uporabimo stabilno implicitno metodo, kjer v vsakem časovnem koraku rešujemo

$$\left(A - \frac{\Delta t}{2} B \right) \vec{c}^{n+1} = \left(A + \frac{\Delta t}{2} B \right) \vec{c}^n.$$

2 Naloga

Naloga:

- Reši difuzijsko enačbo v eni razsežnosti $x \in [0, a]$ z začetnim pogojem po plasti gaussovsko porazdeljene temperature

$$T(x, 0) \propto e^{-(x-a/2)^2/\sigma^2}$$

(izberi razumne vrednosti za D , a in σ) in

1. periodičnim robnim pogojem $T(0, t) = T(a, t)$.
2. homogenim Dirichletovim robnim pogojem $T(0, t) = T(a, t) = 0$.

po Fourierovi metodi.

- Kolokacijsko metodo uporabi ob Gaussovem začetnem pogoju in homogenih Dirichletovih robnih pogojih $T(0, t) = T(a, t) = 0$ ter primerjaj obe metodi.

Dodatna naloga: Izberi si še kakšen primer začetnih pogojev, recimo:

$$T(x, 0) = f(x) = \sin(\pi x/a)$$

in preizkusi obe metodi.

Dodatek: kubični B -zlepki

Kubični B -zlepki imajo obliko

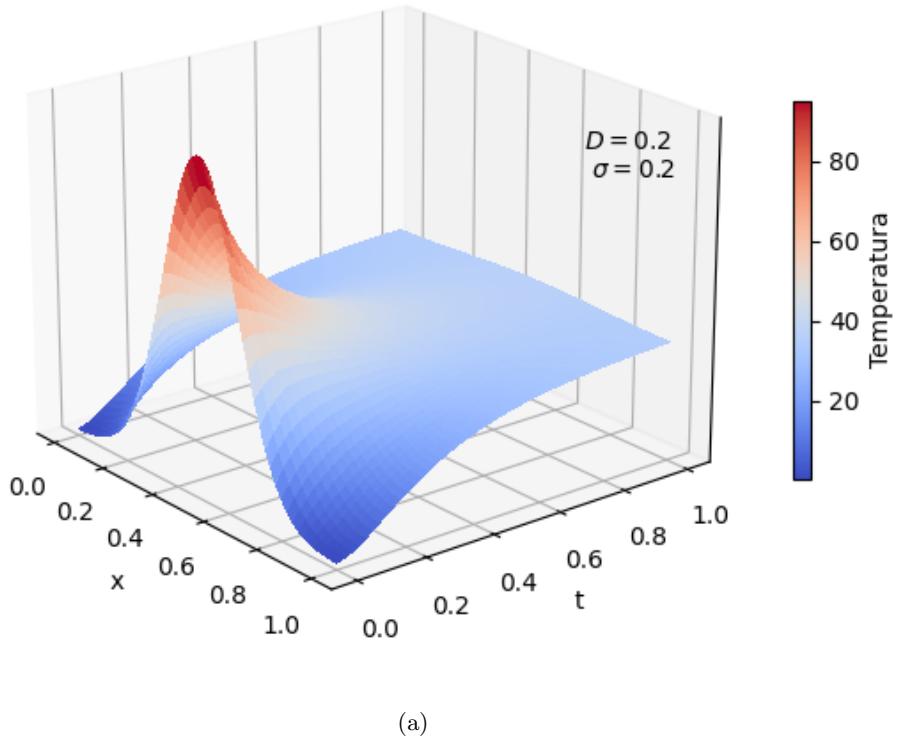
$$B_k(x) = \begin{cases} 0 & \text{če } x \leq x_{k-2} \\ \frac{1}{\Delta x^3}(x - x_{k-2})^3 & \text{če } x_{k-2} \leq x \leq x_{k-1} \\ +\frac{1}{\Delta x^3}(x - x_{k-2})^3 - \frac{4}{\Delta x^3}(x - x_{k-1})^3 & \text{če } x_{k-1} \leq x \leq x_k \\ +\frac{1}{\Delta x^3}(x_{k+2} - x)^3 - \frac{4}{\Delta x^3}(x_{k+1} - x)^3 & \text{če } x_k \leq x \leq x_{k+1} \\ \frac{1}{\Delta x^3}(x_{k+2} - x)^3 & \text{če } x_{k+1} \leq x \leq x_{k+2} \\ 0 & \text{če } x_{k+2} \leq x \end{cases}$$

3 Rezultati

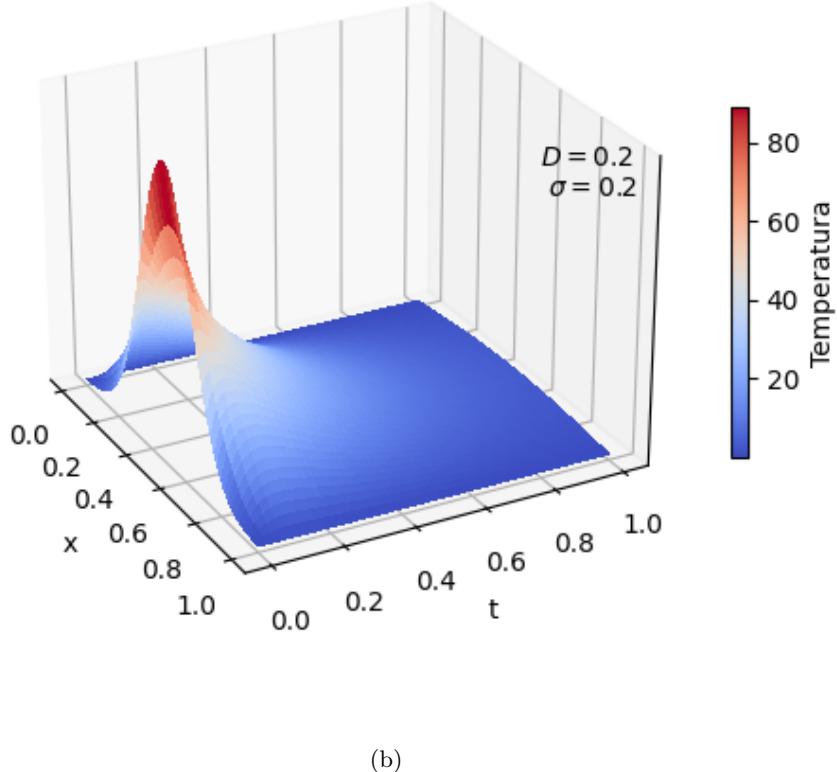
3.1 Fourierova metoda

Metodo sem implementiral z uporabo FFT. Začetni pogoj sem najprej transformiral v frekvenčni spekter, tako sem dobil koeficiente T_k . Časovni razvoj koeficientov lahko dobimo analitično ali z npr. Eulerjevo metodo. V našem primeru so T_k krajevne točke v frekvenčnem spektru, katere razvijemo v času. Dano matriko nato po krajevnih vektorjih trasnformiramo nazaj z obratno FFT. Rešitvi difuzijske enačbe pri danem robnem pogoju prikazujeta sliki 1a in 1b. Za analitično rešitev sem izbral FFT rešitev z analitično rešeno enačbo 2. Za velikost mreže pri analitični rešitvi sem vzel $N_x \times N_t = 5000 \times 5000$ točk. Razlike v rešitvah pri periodičnih pogojih prikazuje slika 2a. Pri periodičnem pogoju sta robni vrednosti temeptrure enaki, kar pomeni, da temperatura ne uhaja izven našega območja, zato se bo Gaussovsko porazdeljen začetni profil razlezil do neke končne neničelne vrednosti (slika 1a). Pri Dirichletovih pogojih velja, da je temperatura na obeh robovih enaka $0^\circ C$, kar pomeni, da toplota odteka iz našega območja, da se robna pogoja ohranjata. V tem primeru bo končna temperatura palice enaka 0° .

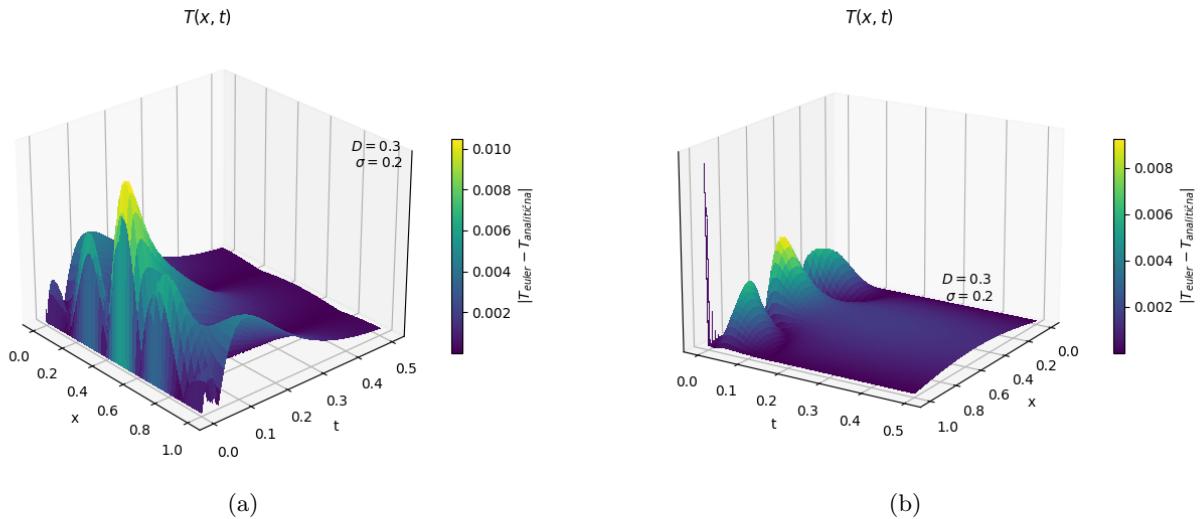
$$T(x, t)$$



$$T(x, t)$$

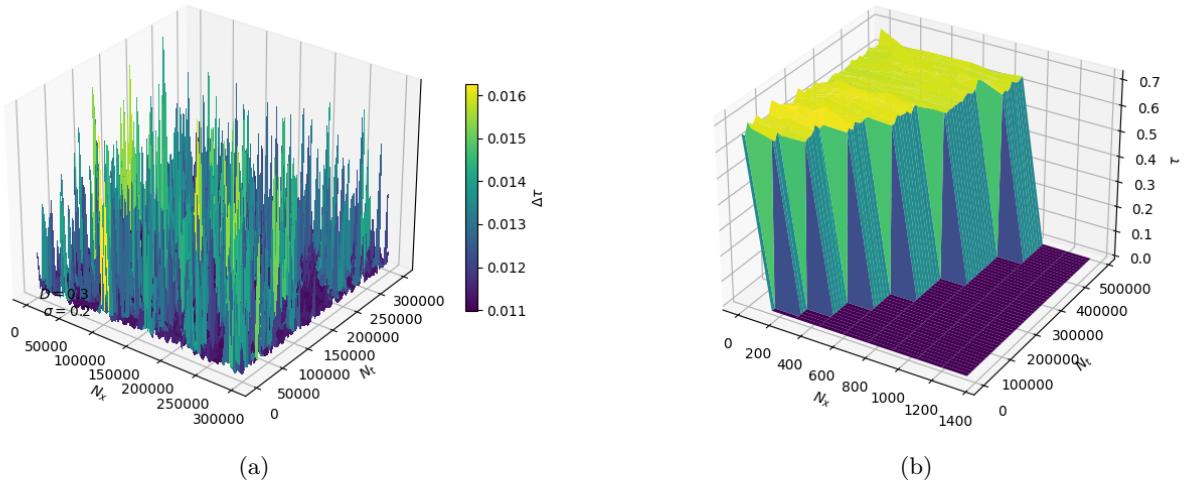


Slika 1: (a) Analitična rešitev difuzijske enačbe na mreži $N_x \times N_t = 1000 \times 1000$, pri periodičnih pogojih.(b) Analitična rešitev difuzijske enačbe na mreži $N_x \times N_t = 1000 \times 1000$, pri Dirichletovih pogojih.



Slika 2: (a) Razlika med analitično rešitvijo in rešitvijo, kjer so koeficienti izračunani z Eulerjevo metodo na mreži $N_x \times N_t = 100 \times 10000$, pri periodičnih pogojih.(b) Razlika med analitično rešitvijo in rešitvijo, kjer so koeficienti izračunani z Eulerjevo metodo na mreži $N_x \times N_t = 100 \times 15000$, pri Dirichletovih pogojih. Mreža ni kvadratna, saj sem moral izpolniti pogoj o stabilnosti Eulerjeve metode.

Če želimo izračunati koeficiente z Eulerjevo metodo je potrebno paziti na območje, kjer je metoda nestabilna. Iz enačbe za stabilnost sledi, da mora biti časovni interval veliko manjši kot dana frekvenca f_k , ki raste kvadratično. Časovno odvisnost analitične rešitve pri periodičnih pogojih, rešitve z Eulerjevo metodo in območje stabilnosti prikazuje slika 3. Iz slike lahko vidim, da je pričakovano z Eulerjevo metodo iskanje rešitve počasnejše. Odvisnosti od števila korakov v času in kraju pa izgleda precej konstantna v obeh primerih, kar je malce presenetljivo, zato sem pogledal vajo Enačba hoda, kjer sem izračunal časovno odvisnost. Tudi tam sem dobil približno konstantno vrednost časovne odvisnosti Eulerjeve metode.

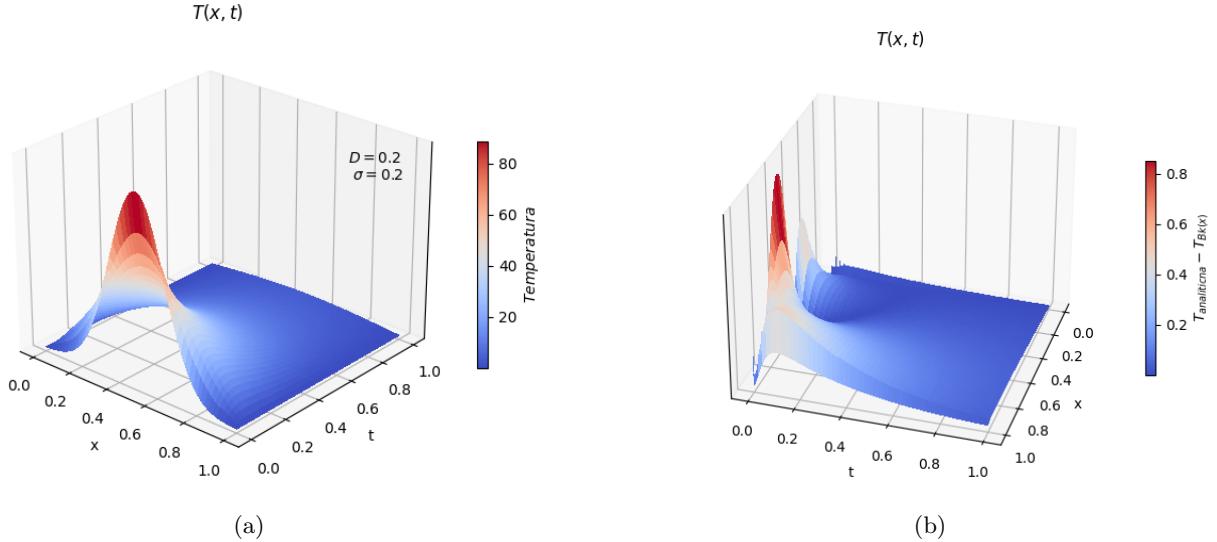


Slika 3: (a) Časovna odvisnost Fourierove metode z analitičnimi koeficienti v odvisnosti od števila korakov v času in kraju.(b) Območje stabilnosti in časovna odvisnost Fourierove metode s koeficienti izračunanimi z Eulerjevo metodo v odvisnosti od števila korakov v času in kraju.

3.2 Kolokacijska metoda

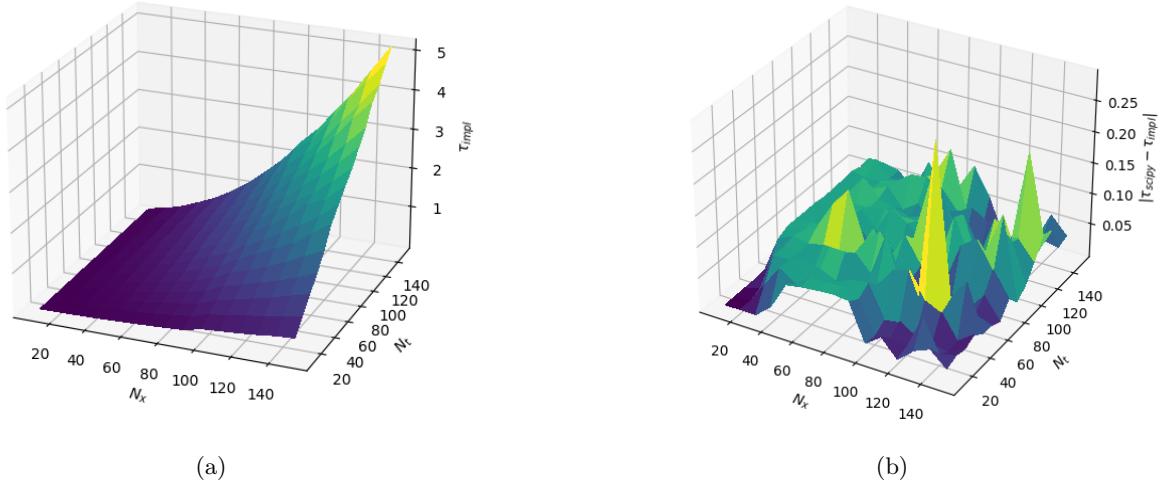
Pri metodi skonstruiral A in B matriko in iz reševanja začetnega trikotnrega sistema dobil začetni vektor. Vsak naslednji vektor (naslednja rešitev ob času $t + dt$) sem dobil tako, da sem vsakič reševal tridiagonalen sistem, ki nam da pomožno rešitev ob naslednjem času. Za reševanje sistema sem uporabil LU razcep in metodo *solve*, ki sem jo dobil v eni od pomožnih zgledov iz prejšnjih nalog. Za primerjavo sem uporabil še

funkciji iz paketa SciPy in sicer funkcijo za inverz `linalg.inv()` in funkcijo za reševanje trikotnega sistema `solve.triangular()`. Da dobimo rešitev $T(x,t)$ je potrebno izračunati vsoto 3, s katero sem imel nekaj težav. Na koncu sem vzel večjo matriko ničel in v njej izračunal, z reševanjem sistema, vektorje in potem izračunal vsoto, kjer pomnožimo element matrike z zlepkom in seštevamo po krajevni koordinati. Metodo sem uporabil za Dirichletova robna pogoja, saj se v tem primeru precej poenostavi. Rešitev difuzijske enačbe in razlika z analitično rešitvijo Fourierove metode prikazuje slika 4a in b.



Slika 4: (a) Rešitev difuzijske enačbe na mreži $N_x \times N_t = 500 \times 500$, pri Dirichletovih pogojih.(b) Razlika med analitično rešitvijo Fourierove metode in Kolokacijske metode

Pogledal sem še časovne zahtevnosti implementirane metode in metode z uporabo SciPy funkcij (slika 5). Metode so precej počasne za veliko število točk in pričakovano narašča čas z dimenzijo. Tudi z uporabo vgrajenih funkcij ne pridobimo veliko na času.



Slika 5: (a) Časovna zahtevnost implementirane metode v odvisnosti od velikosti mreže.(b) Razlika med časovno zahtevnostjo implementirane metode in metode, kjer so uporabljene vgrajene funkcije iz paketa SciPy.

3.3 Dodatna naloga

Rešeni difuzijski enačbi pri Dirichletovih robnih pogojih in periodičnih robnih pogojih z Fourierovo metodo na mreži 5000×5000 . (slikivpriponki)

4 Zaključek

Pri reševanju PDE se lahko poslužimo obeh metod. Fourierova metoda je po mojem mnenju natančnejša, saj lahko vzamemo veliko točk in ni tako časovno zahtevna. Če ne poznamo analitične rešitve pri koeficientih je potrebno reševati diferencialno enačbo kar povzroči dodatno časovno zakasnitev. Fourierova metoda je nekoliko hitrejša od Kolokacijske, sploh če v slednji vzamemo veliko matriko. Kolokacijska metoda nam da rešitev že ob sorazmerno majhni matriki, medtem ko za Fourierovo potrebujemo večjo matriko če želimo uporabiti FFT. Če uporabimo DFT se metoda upočasni.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

10. naloga: Diferenčne metode za parcialne diferencialne enačbe

Žiga Šinigoj, 28191058

Ljubljana, januar 2022

1 Uvod

Enorazsežna nestacionarna Schödingerjeva enačba

$$\left(i\hbar \frac{\partial}{\partial t} - H \right) \psi(x, t) = 0$$

je osnovno orodje za nerelativistični opis časovnega razvoja kvantnih stanj v različnih potencialih. Tu obravnavamo samo od časa neodvisne hamiltonske operatorje

$$H = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x).$$

Z menjavo spremenljivk $H/\hbar \mapsto H$, $x\sqrt{m/\hbar} \mapsto x$ in $V(x\sqrt{m/\hbar})/\hbar \mapsto V(x)$, efektivno postavimo $\hbar = m = 1$,

$$H = -\frac{1}{2} \frac{\partial^2}{\partial x^2} + V(x). \quad (1)$$

Razvoj stanja $\psi(x, t)$ v stanje $\psi(x, t + \Delta t)$ opišemo s približkom

$$\psi(x, t + \Delta t) = e^{-iH\Delta t} \psi(x, t) \approx \frac{1 - \frac{1}{2}iH\Delta t}{1 + \frac{1}{2}iH\Delta t} \psi(x, t), \quad (2)$$

ki je unitaren in je reda $\mathcal{O}(\Delta t^3)$. Območje $a \leq x \leq b$ diskretiziramo na krajevno mrežo $x_j = a + j\Delta x$ pri $0 \leq j < N$, $\Delta x = (b - a)/(N - 1)$, časovni razvoj pa spremljamo ob časih $t_n = n\Delta t$. Vrednosti valovne funkcije in potenciala v mrežnih točkah ob času t_n označimo $\psi(x_j, t_n) = \psi_j^n$ oziroma $V(x_j) = V_j$. Krajevni odvod izrazimo z diferenco

$$\Psi''(x) \approx \frac{\psi(x + \Delta x, t) - 2\psi(x, t) + \psi(x - \Delta x, t)}{\Delta x^2} = \frac{\psi_{j+1}^n - 2\psi_j^n + \psi_{j-1}^n}{\Delta x^2}.$$

Ko te približke vstavimo v enačbo (2) in razpišemo Hamiltonov operator po enačbi (1), dobimo sistem enačb

$$\psi_j^{n+1} - i\frac{\Delta t}{4\Delta x^2} [\psi_{j+1}^{n+1} - 2\psi_j^{n+1} + \psi_{j-1}^{n+1}] + i\frac{\Delta t}{2} V_j \psi_j^{n+1} = \psi_j^n + i\frac{\Delta t}{4\Delta x^2} [\psi_{j+1}^n - 2\psi_j^n + \psi_{j-1}^n] - i\frac{\Delta t}{2} V_j \psi_j^n,$$

v notranjih točkah mreže, medtem ko na robu ($j \leq 0$ in $j \geq N$) postavimo $\psi_j^n = 0$. Vrednosti valovne funkcije v točkah x_j uredimo v vektor

$$\Psi^n = (\psi_1^n, \dots, \psi_{N-1}^n)^T$$

in sistem prepišemo v matrično obliko

$$\mathbf{A}\Psi^{n+1} = \mathbf{A}^*\Psi^n, \quad \mathbf{A} = \begin{pmatrix} d_1 & a & & & \\ a & d_2 & a & & \\ & a & d_3 & a & \\ & & \ddots & \ddots & \ddots \\ & & & a & d_{N-2} & a \\ & & & & a & d_{N-1} \end{pmatrix},$$

kjer je

$$b = i\frac{\Delta t}{2\Delta x^2}, \quad a = -\frac{b}{2}, \quad d_j = 1 + b + i\frac{\Delta t}{2} V_j.$$

Dobili smo torej matrični sistem, ki ga moramo rešiti v vsakem časovnem koraku, da iz stanja Ψ^n dobimo stanje Ψ^{n+1} . Matrika \mathbf{A} in vektor Ψ imata kompleksne elemente, zato račun najlažje opraviš v kompleksni aritmetiki¹.

¹#include <complex.h> v c, #include <complex> v c++, from cmath import * za kompleksne funkcije v Pythonu (sama kompleksna aritmetika pa je vgrajena).

2 Naloga

Spremljaj časovni razvoj začetnega stanja

$$\Psi(x, 0) = \sqrt{\frac{\alpha}{\sqrt{\pi}}} e^{-\alpha^2(x-\lambda)^2/2}$$

v harmonskem potencialu $V(x) = \frac{1}{2}kx^2$, kjer je v naravnih enotah $\alpha = k^{1/4}$, $\omega = \sqrt{k}$. Analitična rešitev je koherentno stanje

$$\psi(x, t) = \sqrt{\frac{\alpha}{\sqrt{\pi}}} \exp \left[-\frac{1}{2} (\xi - \xi_\lambda \cos \omega t)^2 - i \left(\frac{\omega t}{2} + \xi \xi_\lambda \sin \omega t - \frac{1}{4} \xi_\lambda^2 \sin 2\omega t \right) \right],$$

kjer je $\xi = \alpha x$, $\xi_\lambda = \alpha \lambda$. Postavi parametre na $\omega = 0.2$, $\lambda = 10$. Krajevno mrežo vpni v interval $[a, b] = [-40, 40]$ z $N = 300$ aktivnimi točkami. Nihajni čas je $T = 2\pi/\omega$ – primerno prilagodi časovni korak Δt in stanje opazuj deset period.

Opazuj še razvoj gaussovskega valovnega paketa

$$\psi(x, 0) = (2\pi\sigma_0^2)^{-1/4} e^{ik_0(x-\lambda)} e^{-(x-\lambda)^2/(2\sigma_0)^2}$$

v prostoru brez potenciala. Postavi $\sigma_0 = 1/20$, $k_0 = 50\pi$, $\lambda = 0.25$ in območje $[a, b] = [-0.5, 1.5]$ ter $\Delta t = 2\Delta x^2$. Časovni razvoj spremljaj, dokler težišče paketa ne pride do $x \approx 0.75$. Analitična rešitev je

$$\psi(x, t) = \frac{(2\pi\sigma_0^2)^{-1/4}}{\sqrt{1+it/(2\sigma_0^2)}} \exp \left[\frac{-(x-\lambda)^2/(2\sigma_0)^2 + ik_0(x-\lambda) - ik_0^2 t/2}{1+it/(2\sigma_0^2)} \right]$$

Dodatna naloga: Z uporabljenim približkom za drugi odvod reda $\mathcal{O}(\Delta x^2)$ dobimo tridiagonalno matriko. Z diferencami višjih redov dobimo večdiagonalno (pasovno) matriko, a dosežemo tudi večjo krajevno natančnost. Diference višjih redov lahko hitro izračunaš na primer v Mathematici s funkcijo

```
FD[m,n,s]:=CoefficientList[Normal[Series[x^s Log[x]^m,{x,1,n}]/h^m],x];
```

kjer je m red diference (odvoda), n število intervalov širine $h = \Delta x$, ki jih diferenca upošteva, in s število intervalov med točko, kjer diferenco računamo, in skrajno levo točko diferenčne sheme. Zgornjo tritočkovno sheme za drugo diferenco dobimo kot $FD[2, 2, 1]$, saj se razpenja čez $n=2$ intervala, sredinska točka pa je v točki z indeksom $s=1$.

Tudi korakanje v času je mogoče izboljšati z uporabo Padéjeve aproksimacije za eksponentno funkcijo.

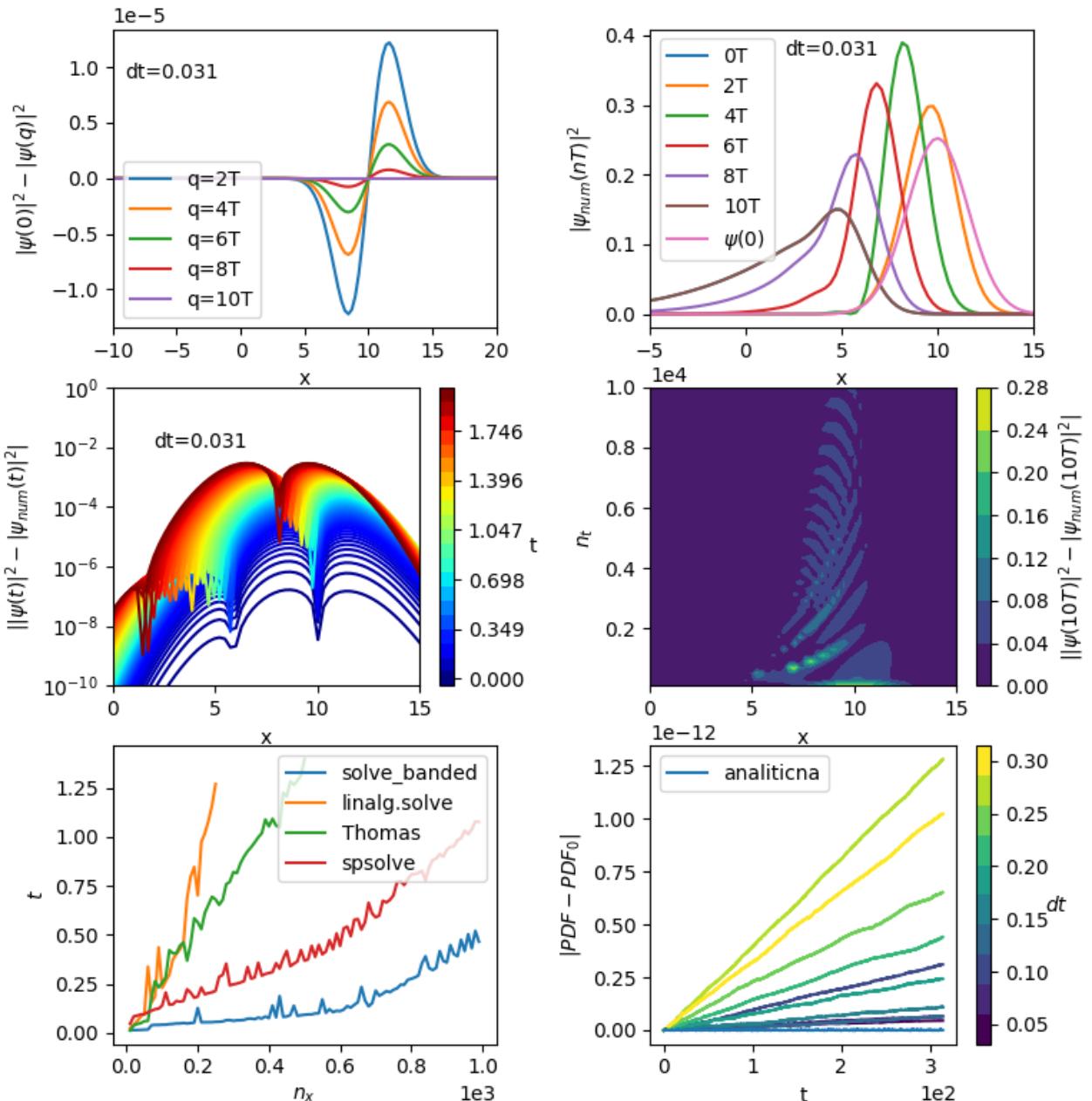
3 Rezultati

Če Schrödingerjevo enačbo zapišemo s končnimi diferencami po Crank-Nicolsonovi metodi se problem prevede na reševanje matrična sistema. Reševanje se razlikuje od reševanja difuzijske enačbe po tem, da imamo kompleksne spremenljivke. Matrika A je v najnižjem redu tridiagonalna. Za reševanje sistema sem uporabil metodo *solvebanded* iz paketa SciPy. Pri časovni zahtevnosti sem primerjal še naslednje metode: Thomasov algoritem, *spsolve* iz paketa SciPy in *linalg.solve* iz paketa NumPy. Slike bom označeval kot matrične elemente $(1,0), \dots$

3.1 Valovni paket v harmonskem potencialu

Z danim začetnim pogojem lahko dobim časovni razvoj tako, da vsakič rešim matrični sistem in tako dobim rešitev ob naslednjem času. Verjetnostna gostota je merljiva količina, zato sem jo risal raje kot valovne funkcije. V harmonskem potencialu gostota ohranja obliko (slika 1). Na sliki izgleda kot da se zelo počasi spreminja, a mislim da je to zaradi tega ker nisem nujno zadel periode, saj je časovni interval diskreten. Če bi vzel več točk, bi se napaka zmanjšala. Numerična rešitev je precej nestabilna, že približno po 1/200 periode se razlikuje od analitične rešitve že na 2 decimalki. Prr numerični metodi je sedaj pomembno število

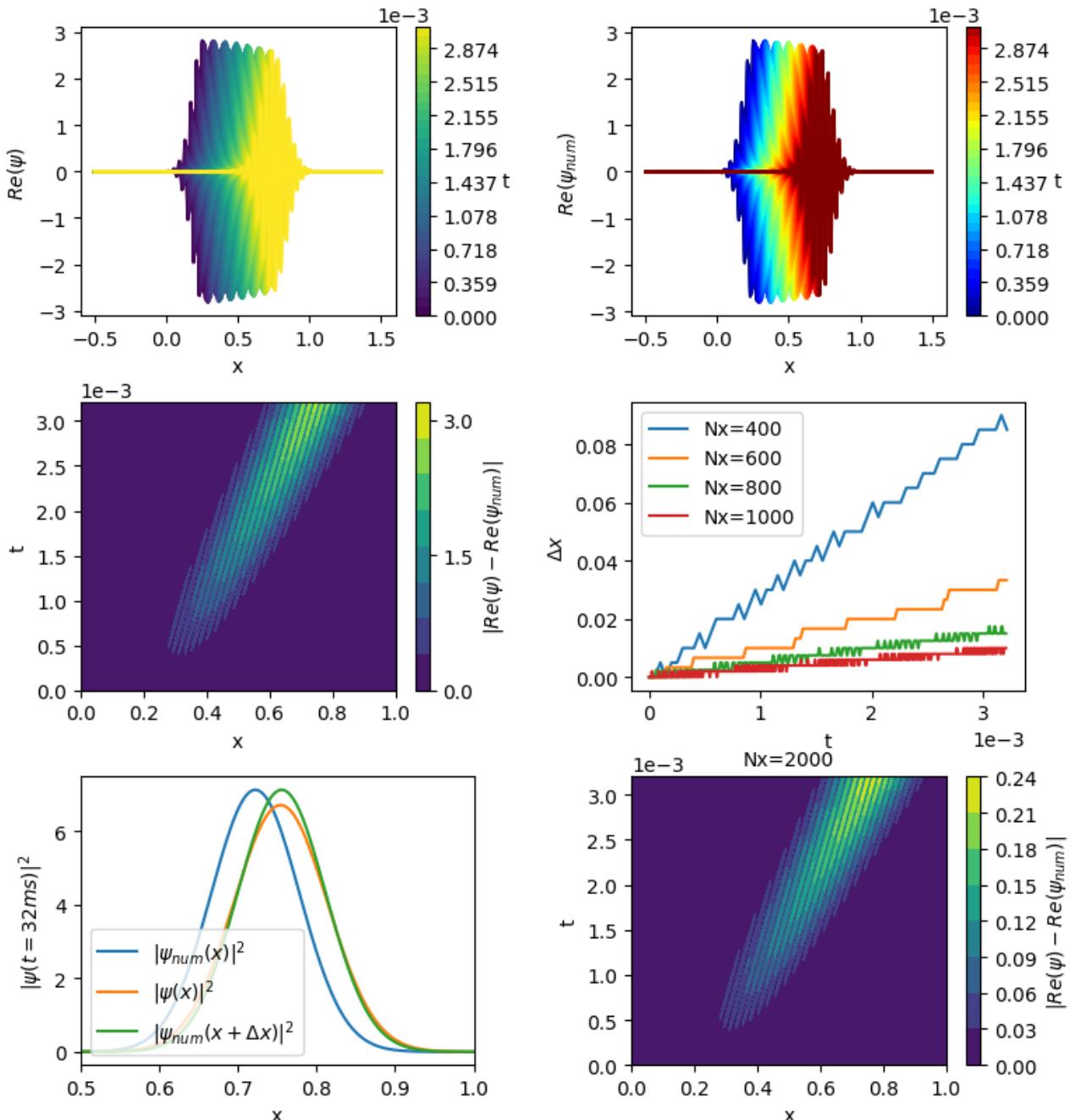
korakov v času in kraju. Večje število točk v obeh dimenzijah pomeni boljšo aproksimacijo (slika 1:(1,1)). Da so rezultati lepsi, sem opazoval funkcije ob periodah, saj bi takrat teoretično morale biti enake kot začetni pogoj. Ob drugih časih funkcije oscilirajo, kar se pokaže pri računanju ohranitve ploščine (PDF), saj ima funkcija z manjšim časovnim korakom manjše odstopanje od prave ploščine (zgornji 2 krivulji), kar mislim da je posledica velikega osciliranja. Efektivno pa izgleda, da večji korak povzroči dissipacijo verjetnostne gostote. Pri časovni zahtevnosti je pričakovano najpočasnejša metoda linalg.solve, saj je narejena za polno matriko A. Precej dobra je tudi metoda spsolve za razpršene matrike in je presenetljivo hitrejša od Thomasovega algoritma, ki je narejen za trikotne sisteme.



Slika 1: ψ brez indeksa predstavlja analitično rešitev, $n_t = 10000$ (če ni drugače označeno na grafu). Od leve proti desni: Razlike med analitično rešitvijo ob različnih časih, numerična verjetnostna amplituda ob periodah, absolutna razlika med numerično in analitično vrednostjo ob danih časih, razlika med analitično in numerično funkcijo ob 10 periodih v odvisnosti od velikosti koraka v času, časovna zahtevnost algoritmov, ploščina v odvisnosti časa pri različnih velikostih koraka.

3.2 Gaussov valovni paket

Po teroiji se Gaussov valovni paket v prostoru brez potenciala "razleze", amplituda se mu s časom zmanjšuje, širina pa povečuje, da se verjetnostna gostota ohranja (slika 2:(1,0)). Numerična rešitev izgleda na prvi pogled precej podobna, a je napaka (slika 2:(1,0)) zelo velika. Pogledal sem posamezne funkcije, zdi se, da se s časom premaknejo v kraju oz. se zamikajo od prave vrednosti.

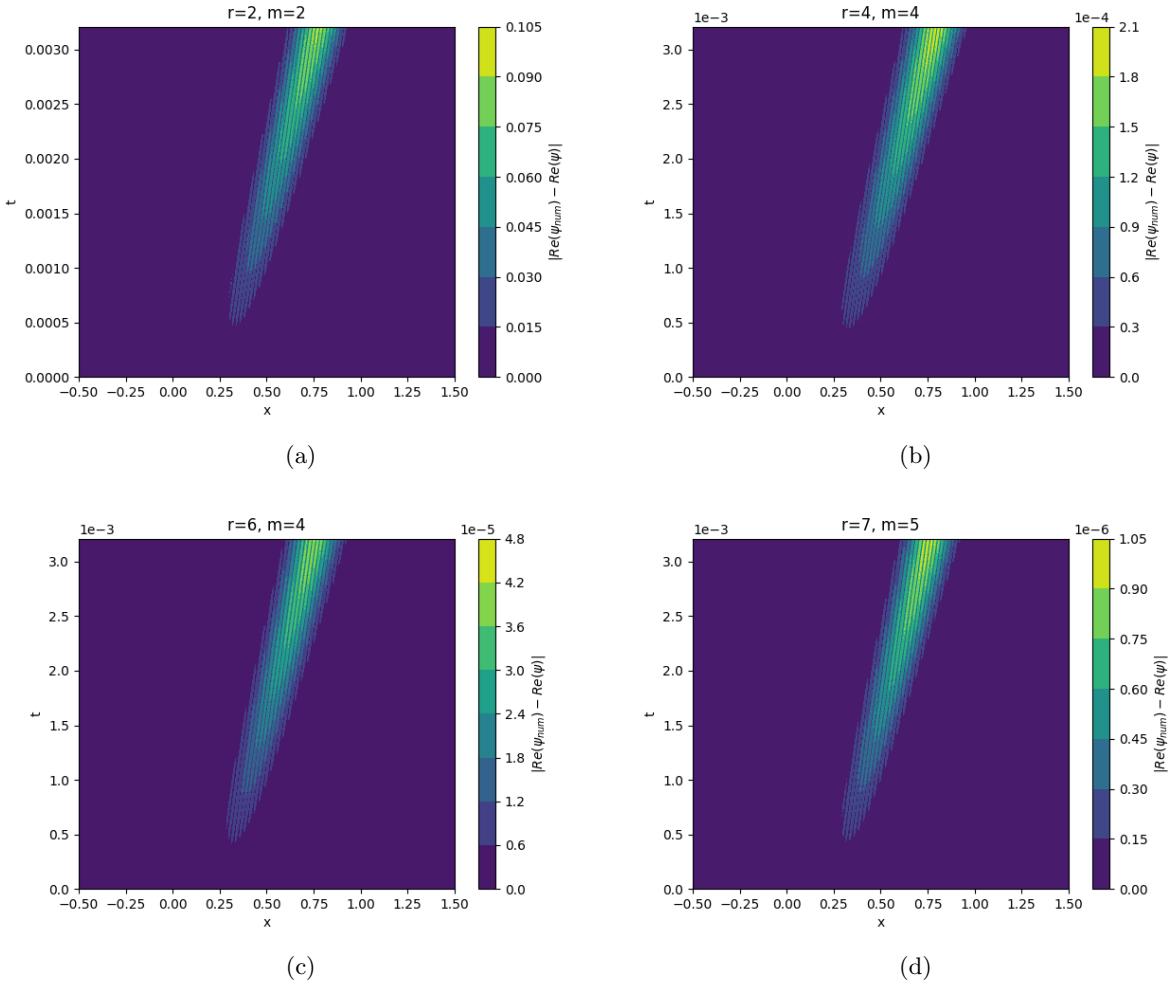


Slika 2: ψ brez indeksa predstavlja analitično rešitev, $n_x = 600$ (če ni drugače označeno na grafu). Od leve proti desni: analitična rešitev ob različnih časih, numerična rešitev ob različnih časih, absolutna razlika med numerično in analitično vrednostjo ob danih časih, razlika v poločaju med analitičnim in numeričnim vrhom v odvisnosti od časa pri različno gosti mreži, primerjava analitične in numerično premaknjene verjetnostne gostote pri danem času, absolutna razlika med numerično in analitično vrednostjo ob danih časih.

Če funkcije poravnam (izničim napako v časovnem korakanju), se precej dobro prilegajo (ostane samo napaka zaradi diference v kraju) (slika 2:(2,0)). Krajevne zamike sem dobil tako, da sem pogledal krajevni interval med vrhom analitične in numerične funkcije. Krajevni zamik, ki je posledica diferenčne sheme v časovnem intervalu prikazuje slka 2:(1,1). Za število točk na mreži sem vzel 600 v krajevni smeri. Število točk v časovni mreži je povezano s številom v krajevni smeri. Časovni interval sem omejil na vrednost, ko pride vrh Gaussove krivulje na $x \approx 0.75$. 600 točk v krajevni smeri ustrezata 124 točkam v časovni. Z zvečanjem točk v krajevni in posledično v časovni smeri (slika 2:(2,1)) lahko pridem do bolj natančnih izračunov.

3.3 Dodatna naloga: Metode višjih redov

Metode višjih redov sem uporabil na Gaussovem paket, saj je malce lažje zapisati matriko. Za število točk sem vzel $Nx = 600$, sistem sem reševal z metodo *solve banded* saj je najhitrejša. Pri metodah višjih redov je opaziti občutno razliko v natančnosti, kar tudi prikazujejo spodnje slike.



Slika 3: Razlika med analitično in numerično valovno funkcijo po absolutni vrednosti v odvisnosti od časa, r predstavlja red odvoda v kraju, m pa red Padejevih polinomov.

4 Zaključek

Z diferenčnimi metodami lahko relativno enostavno pridemo do rešitve problema. Morebitne težave povzročajo nižji redi aproksimacij odvodov, ki so numerično nestabilni. Če uporabimo višje rede za računanje, je metoda zelo dobra. Z uporabo višjih redov se povečuje tudi časovna zahtevnost.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

11. naloga: Reševanje PDE z metodo Galerkina

Žiga Šinigoj, 28191058

Ljubljana, januar 2022

1 Uvod

Pri opisu enakomernega laminarnega toka viskozne in nestisljive tekočine po dolgi ravni cevi pod vplivom stalnega tlačnega gradienta p' se Navier-Stokesova enačba poenostavi v Poissonovo enačbo

$$\nabla^2 v = \Delta v = -\frac{p'}{\eta},$$

kjer je v vzdolžna komponenta hitrosti, odvisna samo od koordinat preseka cevi, η pa je viskoznost tekočine. Enačbo rešujemo v notranjosti preseka cevi, medtem ko je ob stenah hitrost tekočina enaka nič. Za pretok velja Poiseuillov zakon

$$\Phi = \int_S v \, dS = C \frac{p' S^2}{8\pi\eta},$$

kjer je koeficient C odvisen samo od oblike preseka cevi ($C = 1$ za okroglo cev). Določili bomo koeficient za polkrožno cev z radijem R . V novih spremenljivkah $\xi = r/R$ in $u = v\eta/(p'R^2)$ se problem glasi

$$\Delta u(\xi, \varphi) = -1, \quad u(\xi = 1, \varphi) = u(\xi, 0) = u(\xi, \varphi = \pi) = 0,$$

$$C = 8\pi \iint \frac{u(\xi, \varphi) \xi \, d\xi \, d\varphi}{(\pi/2)^2}.$$

Če poznamo lastne funkcije diferencialnega operatorja za določeno geometrijo¹ se reševanje parcialnih diferencialnih enačb včasih lahko prevede na razvoj po lastnih funkcijah. Da bi se izognili računanju lastnih (za primer Besselovih) funkcij in njihovih ničel, ki jih potrebujemo v razvoju, lahko zapišemo aproksimativno rešitev kot linearно kombinacijo nekih poskusnih (*trial*) funkcij

$$\tilde{u}(\xi, \varphi) = \sum_{i=1}^N a_i \Psi_i(\xi, \varphi), \tag{1}$$

za katere ni nujno, da so ortogonalne, pač pa naj zadoščajo robnim pogojem, tako da jim bo avtomatično zadoščala tudi vsota (1). Ta pristop nam pride prav v kompleksnejših geometrijah, ko je uporabnost lastnih funkcij izključena in potrebujemo robustnejši pristop. Približna funkcija \tilde{u} seveda ne zadosti Poissonovi enačbi: preostane majhna napaka ε

$$\Delta \tilde{u}(\xi, \varphi) + 1 = \varepsilon(\xi, \varphi).$$

Pri metodi Galerkina zahtevamo, da je napaka ortogonalna na vse poskusne funkcije Ψ_i ,

$$(\varepsilon, \Psi_i) = 0, \quad i = 1, 2, \dots, N.$$

V splošnem bi lahko zahtevali tudi ortogonalnost ε na nek drug sistem utežnih (*weight*) oziroma testnih (*test*) funkcij Ψ_i . Metoda Galerkina je poseben primer takih metod (*Methods of Weighted Residuals*) z izbiro $\Psi_i = \Psi_i$. Omenjena izbira vodi do sistema enačb za koeficiente a_i

$$\sum_{j=1}^N A_{ij} a_j = b_i, \quad i = 1, 2, \dots, N, \tag{2}$$

$$A_{ij} = (\Delta \Psi_j, \Psi_i), \quad b_i = (-1, \Psi_i),$$

tako da je koeficient za pretok enak

$$C = -\frac{32}{\pi} \sum_{ij} b_i A_{ij}^{-1} b_j.$$

Za kotni del poskusne funkcije obdržimo eksaktne funkcije $\sin((2m+1)\varphi)$, Besselove funkcije za radialni del pa nadomestimo s preprostejšimi funkcijami $\xi^{2m+1}(1-\xi)^n$. Pozor: indeks i pomeni seveda dvojni indeks (šteje obenem m in n)². Zaradi ortogonalnosti po m razpade matrika A v bloke, obrneš pa jo lahko s kako pripravljeno rutino, npr. s spodnjim in zgornjim trikotnim razcepom `ludcmp` in `lubksb` iz NRC.

¹Spomni se na primer na vodikov atom v sferični geometriji, kjer smo imeli $\widehat{L}^2 Y_{lm}(\vartheta, \varphi) = \hbar^2 l(l+1) Y_{lm}(\vartheta, \varphi)$ in $\widehat{L}_z Y_{lm}(\vartheta, \varphi) = m\hbar Y_{lm}(\vartheta, \varphi)$.

²Glej tudi prilogo na spletni učilnici.

2 Naloga

Izračunaj koeficient C . V ta namen moraš dobiti matriko A in vektor b ; preuči, kako je natančnost rezultata (vsote za koeficient C) odvisna od števila členov v indeksih m in n . Zaradi ortogonalnosti po m lahko oba učinka preučuješ neodvisno.

3 Rezultati

3.1 Analitična rešitev

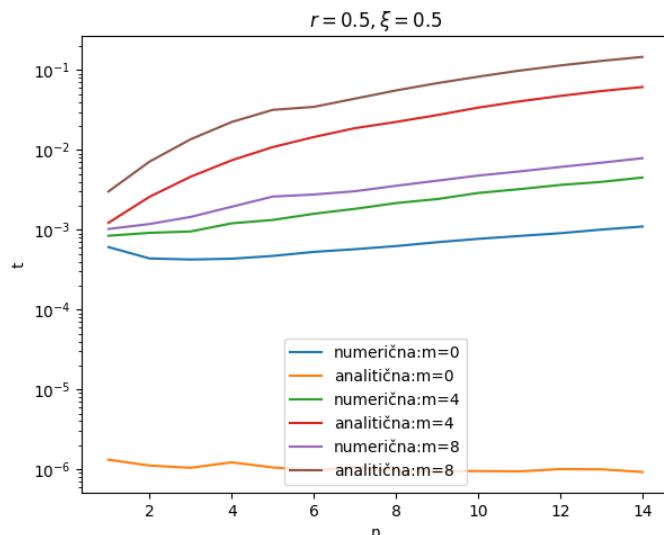
Analitična rešitev amplitudne enačbe je podana kot

$$u(\xi, \varphi) = \sum_{m=0}^{\infty} \sum_{s=1}^{\infty} = C_{ms} J_{2m+1}(y(2m+1)s\xi) \sin((2m+1)\varphi)$$

Če naredim skalarni produkt z analitično rešitvijo, kar je ekvivalentno temu, da homogenost razvijem po baznih funkcijah, lahko dobim iskane koeficiente kot

$$C_{ms} = \frac{4}{(1+2m)(y(2m+1)s)^2} \frac{\int_0^1 J_{2m+1}(y(2m+1)s\xi) \xi \, d\xi}{\int_0^1 J_{2m+1}(y(2m+1)s\xi)^2 \xi \, d\xi} \quad (3)$$

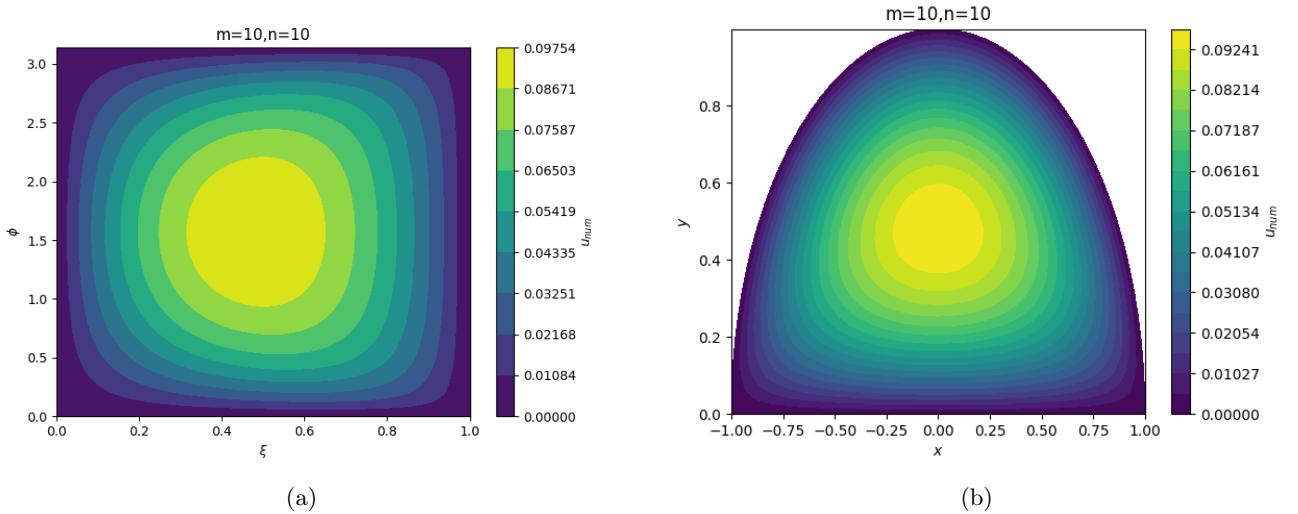
Integralne Besslovi funkcij sem dobil z metodo *quad* iz paketa Scipy. Najprej sem za računanje vsot uporabil for zanke, kar je precej zamudno. Koeficiente sem zato raje zapisal v matriki in potem množil in seštel z baznimi funkcijami. Prednost metode Galerkina je v tem primeru ta, da lahko izračunamo veliko večjo vsoto in se bolj približamo pravi rešitvi kot pa z analitično rešitvijo, kjer moram prav tako seštetи neskončno vsoto. Časovno zahtevnost izračuna brezdimenzijske hitrosti v dani točki prikazuje slika 1. Računanje koeficientov pri analitični vrednosti se izkaže za zelo zamudno ko dodajamo člene. Časovna razlika je okrog enega reda velikosti.



Slika 1: Časovna zahtevnost izračuna histrosti v dani točki pri analitični in 'numerični' rešitvi v odvisnosti od števila členov v vsoti (n in m).

3.2 Numerična rešitev

Da dobim rešitev z metodo Galerkina sem rešil matrični sistem, kjer je matrika bločno diagonalna. Sistem sem reševal z metodo *spsolve* iz paketa SciPy, metoda je namenjena reševanju razpršenih matrik in je po hitrosti takoj za metodo *solvebanded*, ki je namenjena reševanju trikotnega sistema (10. naloga). Iz rešitve dobim koeficiente v vsoti, ki so pomnoženi z baznimi funkcijami. Rešitev sem narisal na mreži 50x50 točk. Velikost mreže ni bistvena v tem primeru, saj je natančnost v dani točki odvisna od števila členov v vsoti. Oznaka npr. $m=10$ pomeni, da ima vsota po m dejansko 11 členov, saj se indeks začne z 0. Pri velikosti posameznega bloka v matriki (n), pa se indeks začne z 1 in npr. $n=10$ ustrezava vsoti desetim členom po n . Numerično rešitev hitrostnega profila prikazuje slika 2

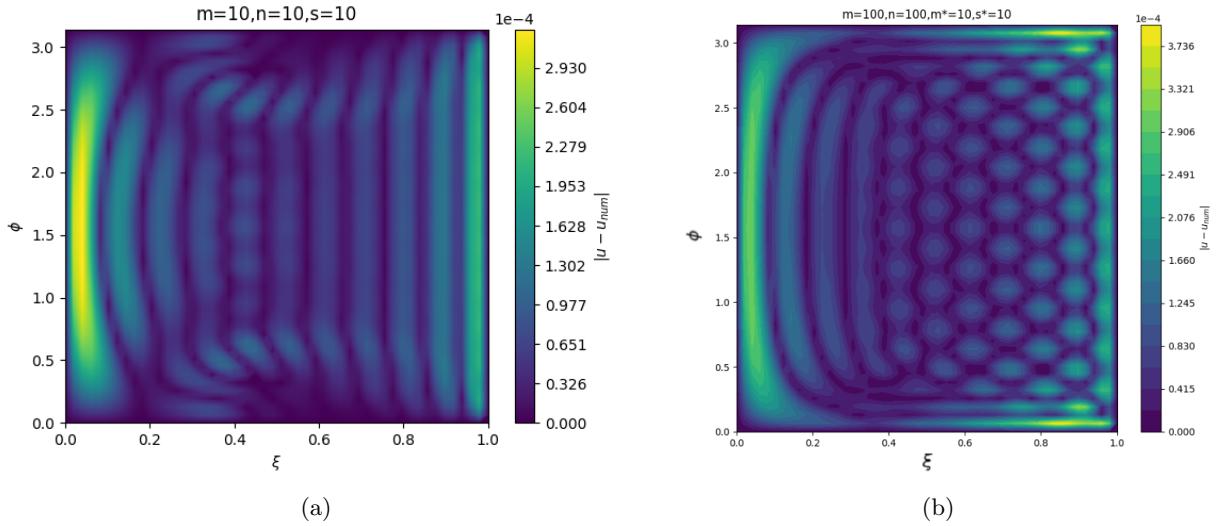


Slika 2: a)Numerična rešitev hitrostnega profila v polarnih koordinatah. b)Numerična rešitev hitrostnega profila v kartezičnih koordinatah. Števili m in n označujeta število členov v posamezni vsoti.

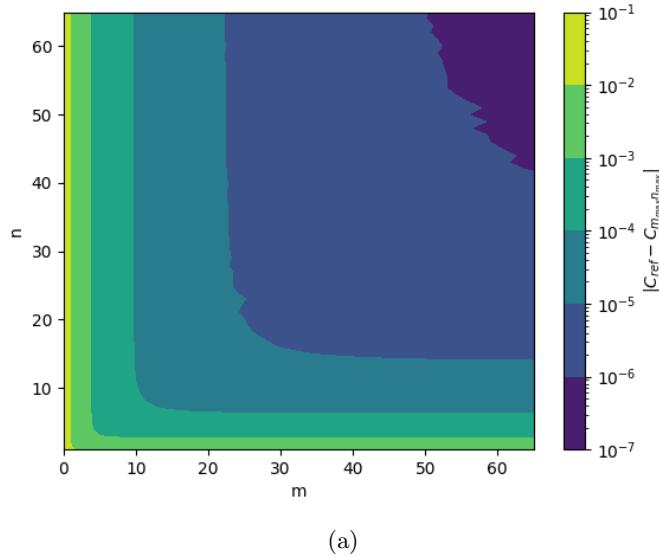
Smiselno se je vprašati kako natančna je numerična rešitev. Odstopanje od analitične rešitve pri enakem številu členov v vsoti prikazuje slika 3a. Odstopanje od analitične rešitve pri stokrat večjem številu členov v numerični vsoti pa prikazuje slika 3b. Pri enakem številu členov dobimo precej natančno numerično rešitev, ki odstopa na četrti decimalki. Pri večjem številu členov v numerični rešitvi se napaka malce poveča, kar pa je malce nenavadno. Če bi bila naša analitična rešitev popolnoma pravilna, bi se z večjim številom členov v numeričnem približku napaka morala zmanjšati. To najverjetneje pomeni, da je analitična rešitev samo približek in z večjim številom numeričnih členov mogoče dosežemo celo bolj natančno rešitev kot pa analitični približek s stokrat manj členi. Pri analitični rešitvi smo omejeni na časovno zahtevnost izračuna koeficientov.

Za določitev pretoka skozi cev je potrebno vedeti faktor pred fizikalnimi količinami. Konstanta je odvisna od oblike preseka cevi. Za izračun konstante lahko rešim sistem z bločno diagonalno matriko in vektorjem \mathbf{b} ter rezultat pomnožim z enakim vektorjem \mathbf{b} ter konstanto. Za rešitev sem vzel vrednosti konstante C , ki jo dobim z reševanjem sistema velikosti $m=100$, $n=100$, kar pomeni 101 bločnih matrik velikosti 100x100 na diagonali v eni matriki. To je tudi referenčna vrednost konstante na sliki 4, ki prikazuje natančnost izračuna konstante v odvisnosti od velikosti in števila podmatrik. Vrednost iskane konstante znaša

$$C_{ref} = 0.757721876571637$$



Slika 3: a) Absolutna razlika med analitično in numerično rešitvijo pri enakem številu členov v vsoti. b) Absolutna razlika med analitično in numerično rešitvijo s stokrat več členi v vsoti, m in n označujeta število elementov v vsoti pri numerični rešitvi, m^* in s^* pa število členov v analitični rešitvi.



Slika 4: a) Absolutno odstopanje vrednosti koeficijenta od referenčne vrednosti ($m=100$, $n=100$) v odvisnosti od števil m in n .

4 Zaključek

Metoda Galerkina je zelo uporabna. Do precej natančne rešitve problema lahko pridem tudi brez poznavanja lastnih funkcij sistema. Tudi če poznam analitično rešitev sistema, ta ni vedno enostavno izračunljiva in je lahko omejena z numeričnimi omejitvami (v tem primeru s časovno zahtevnostjo izračuna) se izkaže, da je lahko metoda zelo dober približek pridobljen v krajšem času.

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA FIZIKO

MATEMATIČNO-FIZIKALNI PRAKTIKUM

12. naloga: Strojno učenje

Žiga Šinigoj, 28191058

Ljubljana, marec 2022

1 Uvod

Dandanes je uporaba različnih algoritmov strojnega učenja (Machine Learning, ML) v znanosti že rutinsko opravilo. Poznamo tri osnovne vrste stojnega učenja:

- Nadzorovano učenje (Supervised learning):
 - Klasifikacija (Classification): sortiranje v različne kategorije.
 - Regresija (Regression): modeliranje oz. ‘fitanje’ napovedi.
- Nenadzorovano učenje (npr. sam najdi kategorije).
- Stimulirano učenje (Artificial Intelligence v ožjem pomenu besede).

V fiziki (in tej nalogi), se tipično ukvarjamo s prvo kategorijo, bodisi za identifikacijo novih pojavov/delcev/... ali pa za ekstrakcijo napovedi (netrivialnih funkcijskih odvisnosti etc).

ML algoritmi imajo prednost pred klasičnim pristopom, da lahko učinkovito razdrobijo kompleksen problem na enostavne elemente in ga ustrezno opišejo:

- pomisli na primer, kako bi bilo težko kar predpostaviti/uganiti pravo analitično funkcijo v več dimenzijah (in je npr. uporaba zlepkov (spline interpolacija) mnogo lažja in boljša).
- Pri izbiri/filtriranju velike količine podatkov z mnogo lastnostmi (npr dogodki pri trkih na LHC) je zelo težko najti količine, ki optimalno ločijo signal od ozadnja, upoštevati vse korelacije in najti optimalno kombinacijo le-teh...

Če dodamo malce matematičnega formalizma strojnega učenja: Predpostavi, da imamo na voljo nabor primerov $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1..N}$, kjer je $\mathbf{x}_k = (x_k^1, \dots, x_k^M)$ naključno izbrani vektor M lastnosti (karakteristik) in je $\mathbf{y}_k = (y_k^1, \dots, y_k^Q)$ vektor Q ciljnih vrednosti, ki so lahko bodisi binarne ali pa realna števila¹. Vrednosti $(\mathbf{x}_k, \mathbf{y}_k)$ so neodvisne in porazdeljene po neki neznani porazdelitvi $P(\cdot, \cdot)$. Cilj ML metode je določiti (priučiti) funkcijo $h : \mathbb{R}^Q \rightarrow \mathbb{R}$, ki minimizira pričakovano vrednost *funkcije izgube* (*expected loss*)

$$\mathcal{L}(h) = \mathbb{E} L(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \frac{1}{N} \sum_{k=1}^N L(\mathbf{y}_k, \mathbf{h}(\mathbf{x}_k)).$$

Tu je $L(\cdot, \cdot)$ gladka funkcija, ki opisuje oceno za kvaliteto napovedi, pri čemer so vrednosti (\mathbf{x}, \mathbf{y}) neodvisno vzorčene iz nabora \mathcal{D} po porazdelitvi P . Po koncu učenja imamo torej na voljo funkcijo $\mathbf{h}(\mathbf{x})$, ki nam za nek vhodni nabor vrednosti $\hat{\mathbf{x}}$ poda napoved $\hat{\mathbf{y}} = \mathbf{h}(\hat{\mathbf{x}})$, ki ustrezno kategorizira ta nabor vrednosti.

Funkcije \mathbf{h} so v praksi sestavljene iz (množice) preprostih funkcij z (nekaj) prostimi parametri, kar na koncu seveda pomeni velik skupni nabor neznanih parametrov in zahteven postopek minimizacije funkcije izgube. Osnovni gradnik odločitvenih dreves je tako kar stopničasta funkcija $H(x_i - t_i) = 0, 1$, ki je enaka ena za $x_i > t_i$ in nič drugače in kjer je x_i ena izmed karakteristik in t_i neznani parameter. Iz skupine takšnih funkcij, ki predstavljajo binarne odločitve lahko skonstruiramo končno uteženo funkcijo

$$\mathbf{h}(\mathbf{x}) = \sum_{i=1}^J \mathbf{a}_i H(x_i - t_i),$$

kjer so \mathbf{a}_i vektorji neznanih uteži. Tako t_i kot \mathbf{b}_i , lahko določimo v procesu učenja. Nadgradnjo predstavljajo nato *pospešena* odločitvena drevesa (BDT), kjer nadomestimo napoved enega drevesa z uteženo množico le-teh, tipično dobljeno v ustreznih iterativnih postopkih (npr. AdaBoost, Gradient Boost ipd.).

Pri nevronskih mrežah je osnovni gradnik t.i. *perceptron*, ki ga opisuje preprosta funkcija

$$h_{w,b}(\mathbf{X}) = \vartheta(\mathbf{w}^T \cdot \mathbf{X} + b),$$

¹...ali pa še kaj, prevedljive na te možnosti, npr barve...

kjer je \mathbf{X} nabor vhodnih vrednosti, \mathbf{w} vektor vrednosti uteži, s katerimi tvorimo uteženo vsoto ter b dodatni konstatni premik (bias). Funkcija ϑ je preprosta gladka funkcija (npr. arctan), ki lahko vpelje nelinearnost v odzivu perceptronja. Nevronska mreža je nato sestavljena iz (poljubne) topologije takšnih perceptronov, ki na začetku sprejme karakteristiko dogodka \mathbf{x} v končni fazi rezultirajo v napovedi $\hat{\mathbf{y}}$, ki mora seveda biti čim bližje ciljni vrednosti \mathbf{y} . Z uporabo ustrezone funkcije izgube (npr MSE: $\mathcal{L}(h) = \mathbb{E} \|\mathbf{y} - \hat{\mathbf{y}}\|^2$), se problem znova prevede na minimizacijo, kjer iščemo optimalne vrednosti (velikega) nabora uteži \mathbf{w}_i ter b_i za vse perceptrone v mreži. Globoke nevronske mreže (DNN) niso nič drugega, kot velike nevronske mreže ali skupine le-teh.

Že namizni računalniki so dovolj močni za osnovne računske naloge, obstajajo pa tudi že zelo uporabniku prijazni vmesniki v jeziku Python, na primer:

- Scikit-Learn (scikit-learn.org): odprtakodni paket za strojno učenje,
- TensorFlow (tensorflow.org): odprtakodni Google-ov sistem za ML, s poudarkom na globokih nevronskeh mrežah (Deep Neural Networks, DNN) z uporabo vmesnika Keras. Prilagojen za delo na GPU in TPU.
- Catboost: (Catboost.ai) : odprtakodna knjižnica za uporabo pospešenih odločitvenih dreves (Boosted Decision Trees, BDT). Prilagojena za delo na GPU.

Za potrebe naloge lahko uporabimo tudi spletni vmesnik Google Collab (colab.research.google.com), ki dopušča omejen dostop do večjih računskih zmogljivosti.

2 Naloga

Na spletni učilnici je na voljo material (koda, vzorci) za ločevanje dogodkov Higgsovega bozona od ostalih procesov ozadja. V naboru simuliranih dogodkov je 18 karakteristik (zveznih kinematičnih lastnosti), katerih vsaka posamezno zelo slabo loči 'signal' od ozadja, z uporabo BDT ali (D)NN, pa lahko tu dosežemo zelo dober uspeh. Na predavanjih smo si ogledali glavne aspekte pomembne pri implementaciji ML, kot so uporaba ustreznih spremenljivk (GIGO), učenje in prekomerno učenje (training/overtraining), vrednotenje uspeha metode kot razmerje med učinkovitostjo (efficiency) in čistostjo (precision) vzorca (Receiver Operating Characteristic, ROC). Določi uspešnost obeh metod (in nariši ROC) za nekaj tipičih konfiguracij BDT in DNN, pri čemer:

- Študiraj vpliv uporabljenih vhodnih spremenljivk - kaj, če vzamemo le nekatere?
- Študiraj BDT in NN in vrednoti uspešnost različnih nastavitev, če spreminjaš nekaj konfiguracijskih parametrov (npr. število perceptronov in plasti nevronskeh mrež pri DNN in število dreves pri BDT).

Dodatna naloga: Implementiraj distribucije iz 'playground' zgleda v BDT (lahko tudi RandomForests) in DNN, te distribucije so na voljo v vseh popularnih ML paketih (npr. Scikit...).

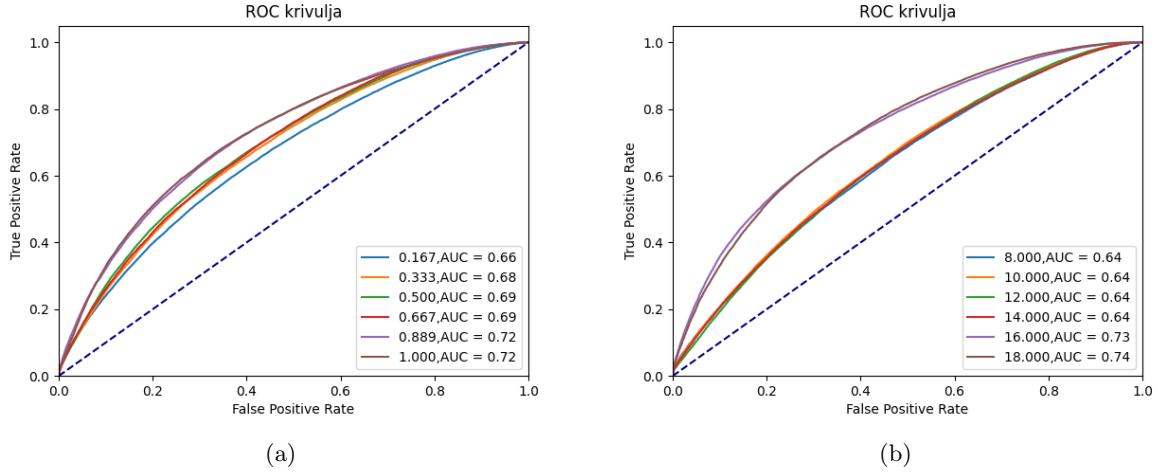
3 Rezultati

3.1 DNN

Za konstrukcijo nevronske mreže sem uporabljal TensorFlow. Globoke nevronske mreže sestavimo tako, da nevronski mreži dodajamo 'skrite' plasti med vhodno in izhodno plastjo. Če na je na sliki označeno število nevronskeh plasti npr. 15, je mišljeno 15 plasti + izhodna plast. Prva plasti ima vhodno obliko prilagojeno številu vhodnih lastnosti oz. spremenljivk. V našem primeru je to 18. Število perceptronov v vseh plasteh razen v zadnji je enako. Vse plasti imajo aktivacijsko funkcijo 'ReLU' razen izhodne plasti, ki je sestavljena iz enega perceptronja z aktivacijsko funkcijo 'sigmoid'. Za iskanje minimalne vrednosti 'loss' funkcije je porabljen algoritem 'ADAM'. Vhodni podatki so normirani na interval [0,1]. Naša nevronska mreža je binarni klasifikator, saj loči podatke na signal ali ozadje. Za določanje učinkovitosti nevronske mreže se uporablja ROC krivulja (in tudi njena ploščina), ki je merilo za uspešnost delovanja nevronske

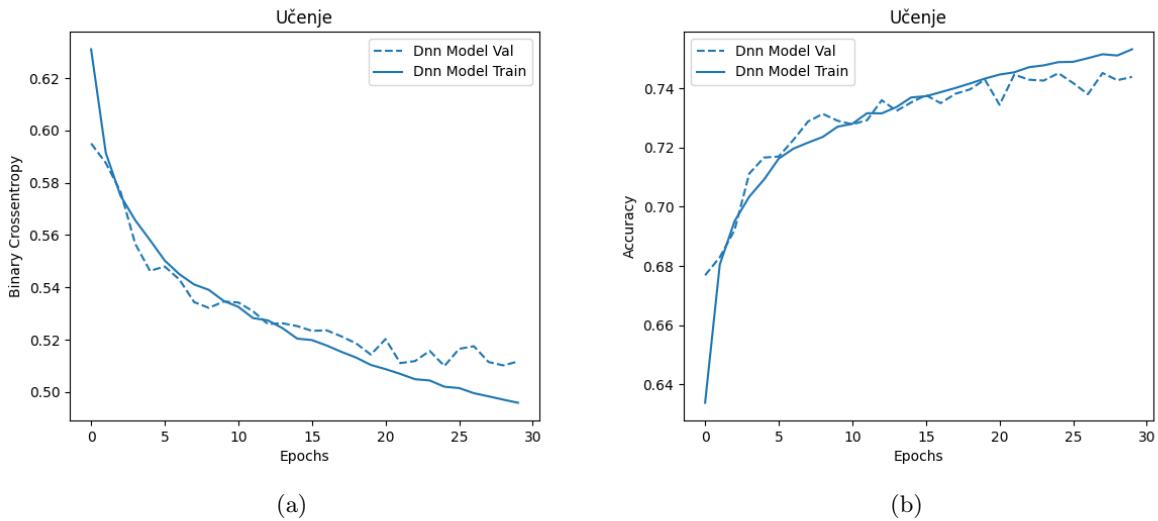
mreže pri ločevanju podatkov v 2 skupini. Perfektni klasifikator bi imel ploščino 1, naključni klasifikator pa 0.5. Za risanje ROC krivulje sem uporabil neodvisne podatke iz mape 'valid'.

Najprej lahko preverimo kako velikost vhodnih podatkov za treniranje mreže in število podanih razpadov oz. vhodnih lastnosti (slika 1) vpliva na ROC krivuljo in njeno ploščino (AUC). Nevronska mreža je sestavljena iz 10 plasti po 32 nevronov + izhodna plast. V tem primeru je omogočeno zgodnje ustavljanje algoritma, da ne pride do 'over-fitovanja'. Velikost podatkov vpliva na učenje nevronske mreže in več podatkov ponavadi pomeni boljšo klasifikacijo. Pri omejitvi števila vhodnih lastnosti (slika 1b) je opaziti precejšnjo razliko, če uporabimo manj kot 16 parametrov od 18. Verjetno je tudi pomembno katere paramtere spustimo, ampak se nisem preveč poglabljal v to.



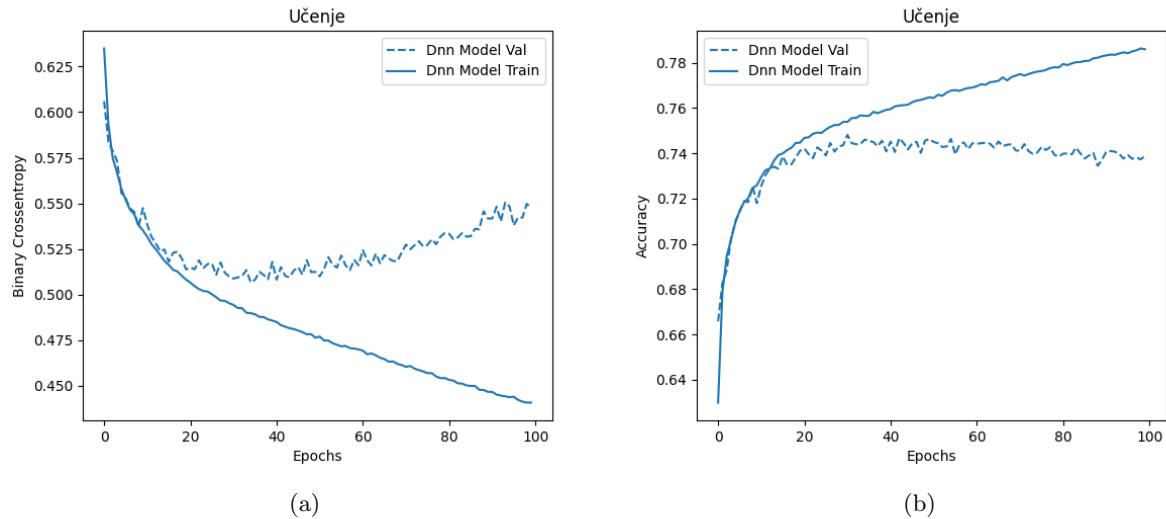
Slika 1: a)ROC krivulja pri različnem deležu vhodnih podatkov za treniranje mreže. b)ROC krivulja pri različnem številu vhodnih parameterov (lastnosti).

Pri učenju je potrebno paziti na število paramterov, ki jih mora nevronska mreža optimizirati, kar je povezano z velikostjo nevronske mreže. Če je število parametrov preveliko in je večje od števila podatkov oz. v enakem redu kot število podatkov, je nevarno, da bo prišlo do 'over-fitovanja' in bo sicer nevronska mreža doseгла odlične rezultate na učnem vzorcu, ampak zato toliko slabše na testnem vzorcu oz. na klasifikaciji drugih vzorcev. Če imamo premalo parametrov nevronska mreža doseže slabše rezultate v klasifikaciji kot bi jih lahko. Učinkovitost je odvisna tudi od števila prehodov (epoch) celotnega nabora podatkov za treniranje skozi mrežo (slika 5a). Za primer over-fitovanja sem vzel nevronske mreže z 10 plasti po 100 perceptronov, kar je okvirno 100000 parametrov, če so vsi povezani z vsemi. Z uporabo funkcije 'earlystopping' lahko



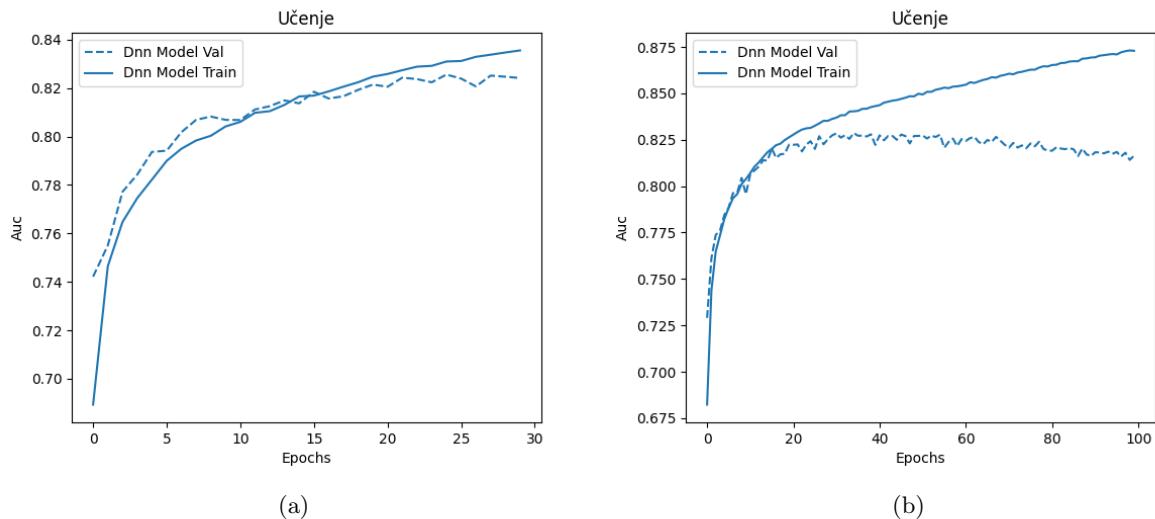
Slika 2: Val-označuje testne podatke, Train-označuje podatke na katerih se mreža uči. a)Ena izmed 'cost' funkcij v odvisnosti od ciklov treniranja (Epochs). b)Natančnost v odvisnosti od ciklov treniranja (Epochs)

ustavimo algoritem ko doseže 'loss' funkcija minimum, tudi če ne opravi vseh ciklov treniranja (slika 2). Na sliki lahko vidimo potek učenja. 'Binary Crossentropy' (slika 2a) je ena od 'cost' funkciji, ki jo je potrebno minimizirati, da dosežemo najboljše delovanje nevronske mreže, natančnost (slika 2b) pa raste s cikli treniranja na podatkih. Če ne ustavim treniranja ob doseženem minimumu dobimo boljše rezultate za trenirano množico, vendar nevronska mreža deluje slabše za množice, ki niso enake kot trenirane (slika 3), kar pa ni cilj nevronske mreže.

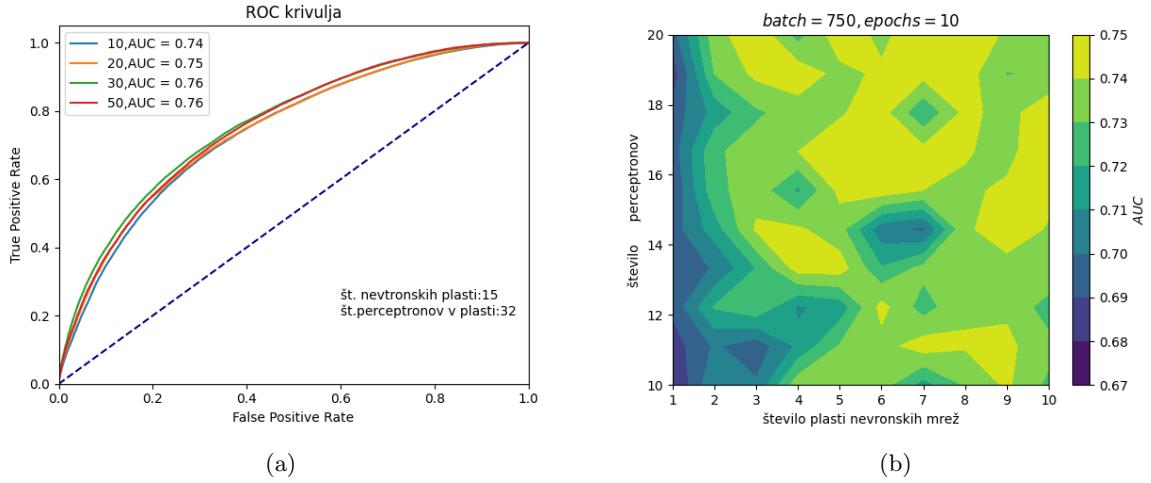


Slika 3: Val-označuje testne podatke, Train-označuje podatke na katerih se mreža uči. a)Ena izmed 'cost' funkcij v odvisnosti od ciklov treniranja (Epochs). b)Natančnost v odvisnosti od ciklov treniranja (Epochs)

Pogledamo lahko še odvisnost ploščine (AUC) od treniranja in pretiranega treniranja nevronske mreže (slika 4 a in b). Če s treniranjem ne dosežemo minimuma zaradi premajhnega števila iteracij se, kot pričakovano, nevronska mreža izboljšuje (slika 5a). Zanimivo je tudi pogledati ploščino pod ROC krivuljo v odvisnosti od števila perceptronov in nevronskih plasti (slika 5b). Izgleda da je v splošnem dobro vzeti več plasti, ampak ni pa mogoče razbrati neke odvisnosti, ki bi povedala kakšno je optimalno število perceptronov in plasti.



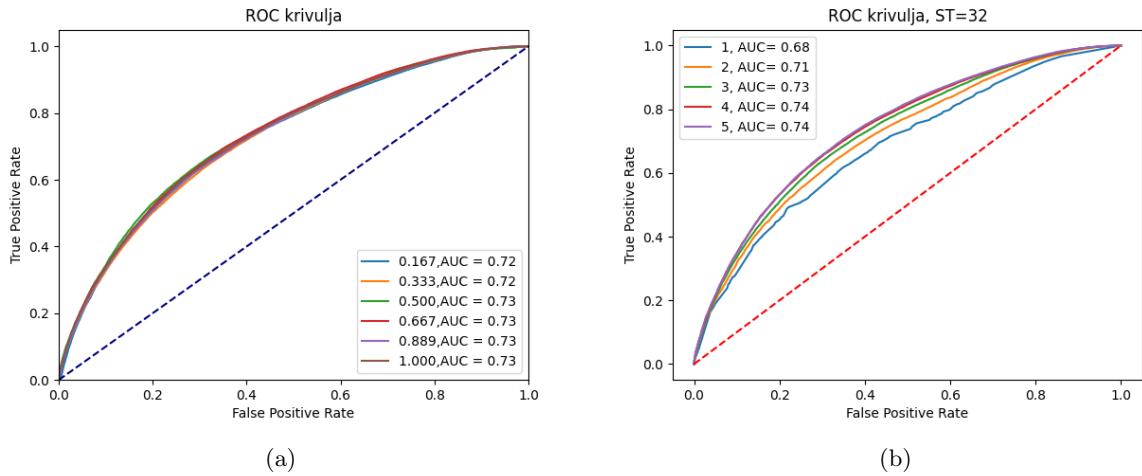
Slika 4: Val-označuje testne podatke, Train-označuje podatke na katerih se mreža uči. a)AUC v odvisnosti od ciklov treniranja (Epochs). b)AUC v odvisnosti od ciklov treniranja (Epochs), kjer je onemogočena funkcija 'earlystopping'.



Slika 5: a)ROC v odvisnosti od števila Epoch. b)AUC v odvisnosti od števila perceptronov in števila plasti.

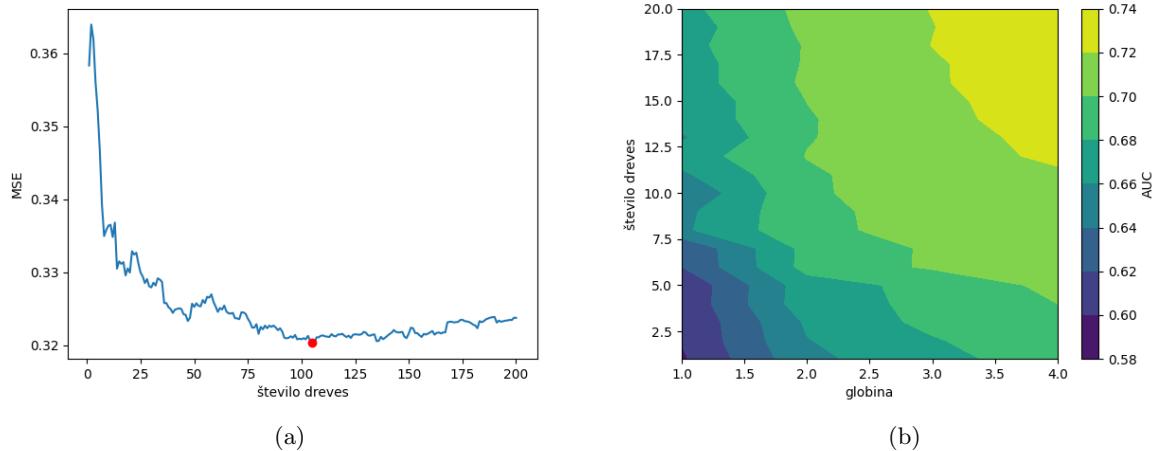
3.2 Boosted Decision Trees in Gradient Boosting

Za implementacijo pospešenih določitvenih dreves sem uporabil Scikit-Learn-ov klasifikator AdaBoost. Algoritem je zasnovan tako, da najprej določi uteži na podatkih za treniranje, nato pa se osredotoči na več podmnožic podatkov, ki jih je prejšnja iteracija premalo utežila in tako iterativno popravlja vrednosti uteži s treniranjem na določenih podmnožicah. Tako kot pri DNN lahko najprej pogledamo, kako se odziva algoritom na manjše število podatkov (slika 6a). Na sliki je videti, da so BDT precej manj občutljiva na količino podatkov kot DNN, in že samo 16 % delež vseh podatkov nam da skoraj enake rezultate kot če uporabimo celotno količino podatkov za treniranje algoritma. Slika 6b prikazuje ROC krivuljo BDT v odvisnosti od globine. Parameter globine pomeni, skozi koliko pogojev se klasificirajo podatki v enem drevesu. Če je globina 1, pomeni da imamo samo en pogoj in 2 možni podmnožici, v kateri se razvrstijo podatki.



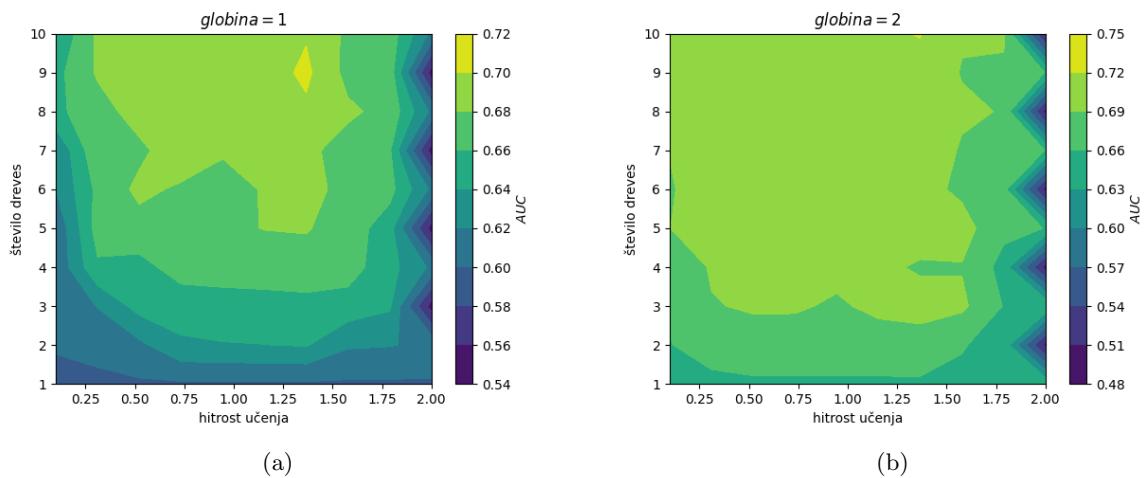
Slika 6: a)ROC krivulja v odvisnosti od deleža uporabljenih podatkov za treniranje, število dreves: 32, globina: 3. b)ROC krivulja v odvisnosti od globine, ST označuje število dreves.

Tudi pri DBT lahko pretirano natreniramo algoritmom. Če določimo preveliko število dreves in s tem preveč razdrobimo podmnožice bo prišlo do 'over-fitovanja' kar prikazuje slika 7. Minimum 'loss' funkcije, v tem primeru MSE označuje rdeča pika. Težavo se da odpraviti na več načinov. Eden izmed njih je ta, da DBT določimo veliko število dreves in potem poiščemo pri kateri vrednosti doseže 'loss' funkcija minimum. Tako uporabimo optimalno število dreves za dani problem. Odvisnost AUC od globine in števila dreves prikazuje slika 7b. Pričakovano se vidi, da se z večanjem globine in števila dreves algoritmom izboljšuje.



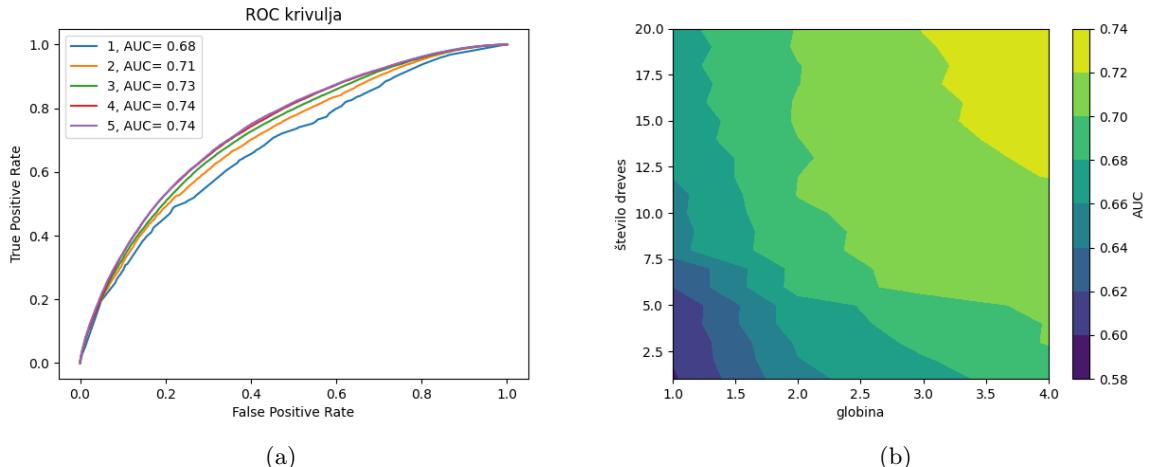
Slika 7: a) MSE v odvisnosti od števila dreves. Rdeča pika označuje minimum, globina: 3 b) AUC v odvisnosti od globine in števila dreves.

Zanimivo je pogledati tudi odvisnost AUC od hitrosti učenja in števila dreves, pri dani globini (slika 8). Privzeta hitrost učenja algoritma AdaBoost je 0.1. Pri globini 1 očitno privzeta hitrost ni optimalna. Paziti je potrebno tudi na zgornjo mejo hitrosti, saj lahko pri prevelikih korakih preskočimo minimum 'loss' funkcije.

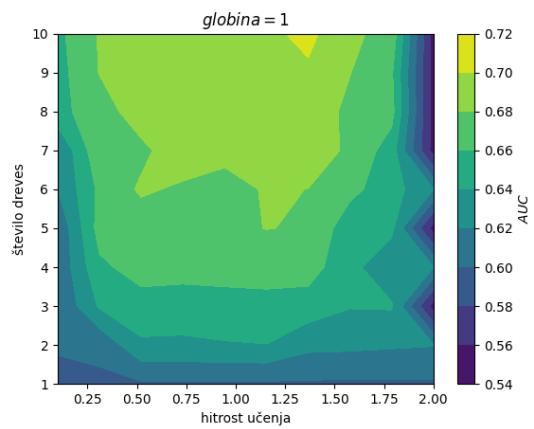


Slika 8: a)AUC v odvisnosti od števila dreves in hitrosti učenja pri globini 1.b) AUC v odvisnosti od števila dreves in hitrosti učenja pri globini 2

Algoritem Gradient Boosting je podoben kot algoritem AdaBoost. Za razliko od AdaBoost, ki po vsaki iteraciji popravi uteži, Gradient Boosting algoritem poskuša 'fitati' nove uteži k ostankom napak, ki jih podajo prejšnje uteži. Pri implementaciji sem uporabil Scikit-Learn-ov algoritem 'GradientBoostingClassifier'. Rezultati so zelo podobni kot pri AdaBoost algoritmu (slika 9, 10).



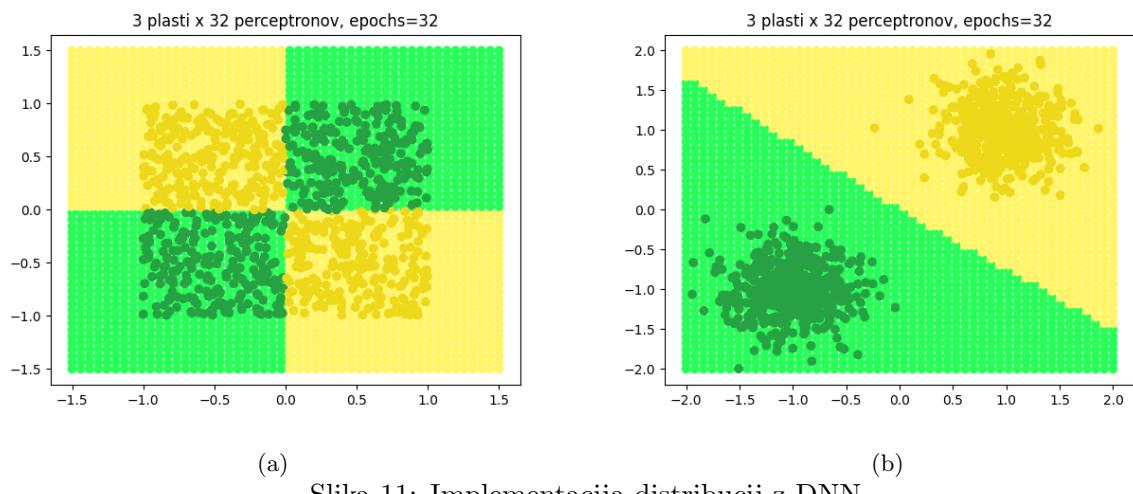
Slika 9: a)ROC krivulja pri različnih globinah, število dreves: 32 b)AUC v odvisnosti od števila dreves in globine.



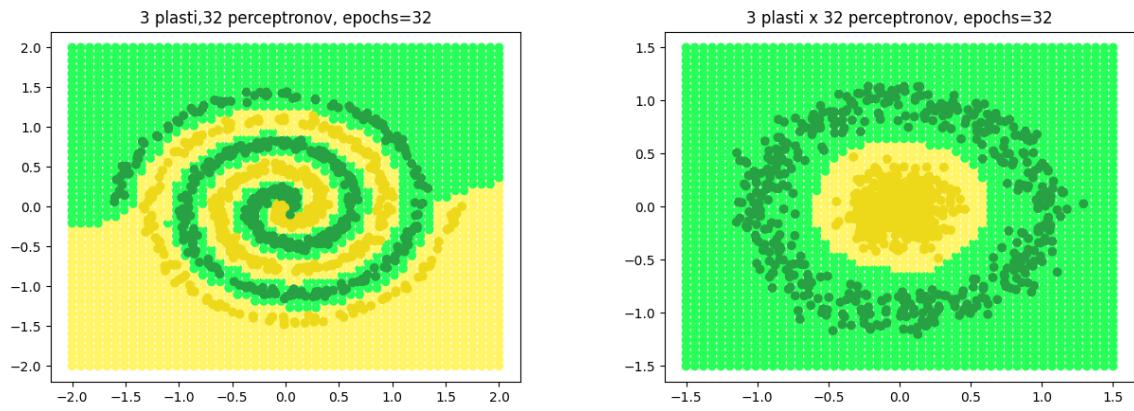
Slika 10: AUC v odvisnosti od hitrosti učenja in števila dreves.

3.3 Dodatna naloga

Na slikah so narisani testni podatki in predvideno ozadje.



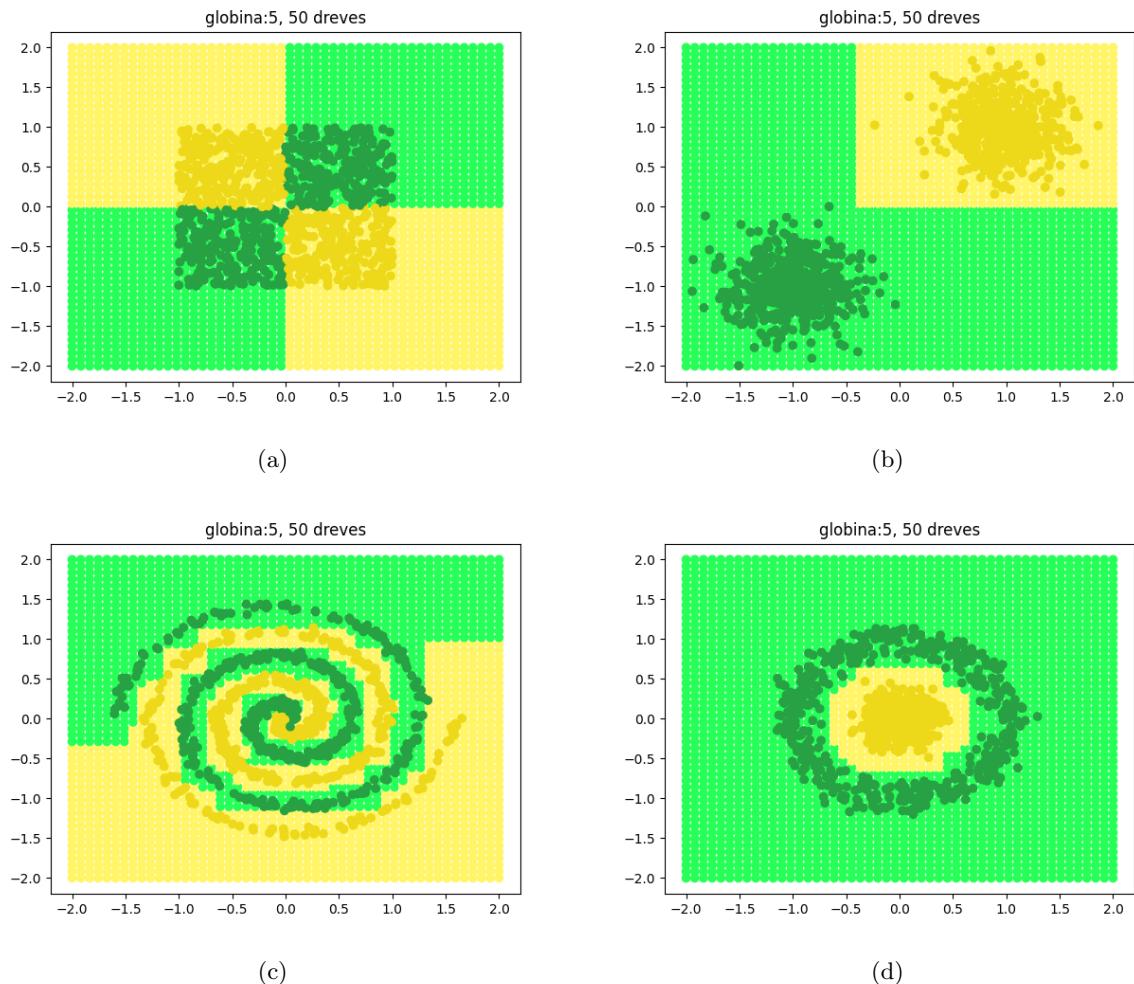
Slika 11: Implementacija distribucij z DNN.



(a)

(b)

Slika 12: Implementacija distribucij z DNN.



(a)

(b)

(c)

(d)

Slika 13: Implementacija distribucij z AdaBoost.

4 Zaključek

Strojno učenje je lahko uporabno.