Matematično-fizikalni praktikum 2025/26

3. naloga: Lastne vrednosti in lastni vektorji

Samo Krejan, 28231092

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

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

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

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

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 $n = 0, 1, 2, \ldots$. Matrika $\langle i|H_0|j\rangle$ z $i, j = 0, 1, 2, \ldots, N-1$ je očitno diagonalna, z vrednostmi $\delta_{ij}(i + 1/2)$ po diagonali. Nemoteni Hamiltonki dodamo anharmonski člen

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

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.

1 Naloga

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

1.1 Sestavljanje Hamiltonke

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 ustrezne matrike,

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

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,

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

Pri implementaciji slednjih načinov si lahko pomagamo z naslednjima formulama za povprečno vrednost matričnega elementa

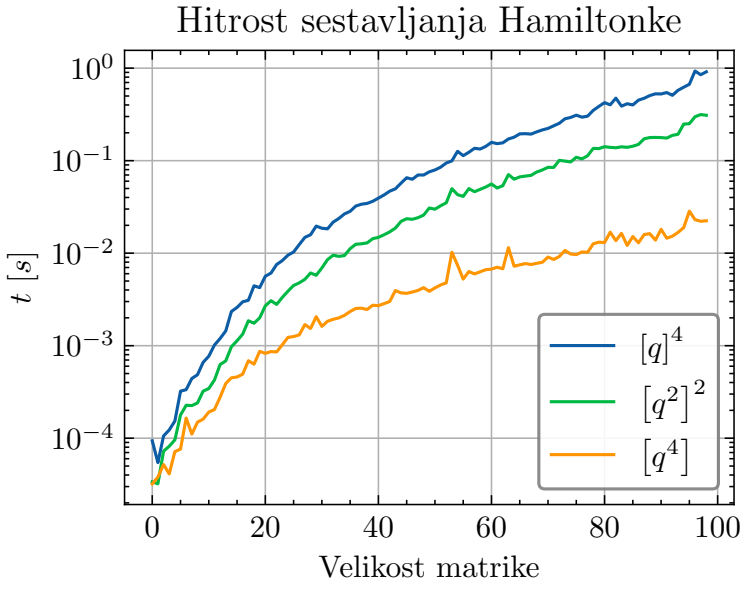
$$\langle i|q^2|j\rangle = \frac{1}{2}\left[\sqrt{j(j-1)}\,\delta_{i,j-2} + (2j+1)\,\delta_{i,j} + \sqrt{(j+1)(j+2)}\,\delta_{i,j+2}\right]$$

ter

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

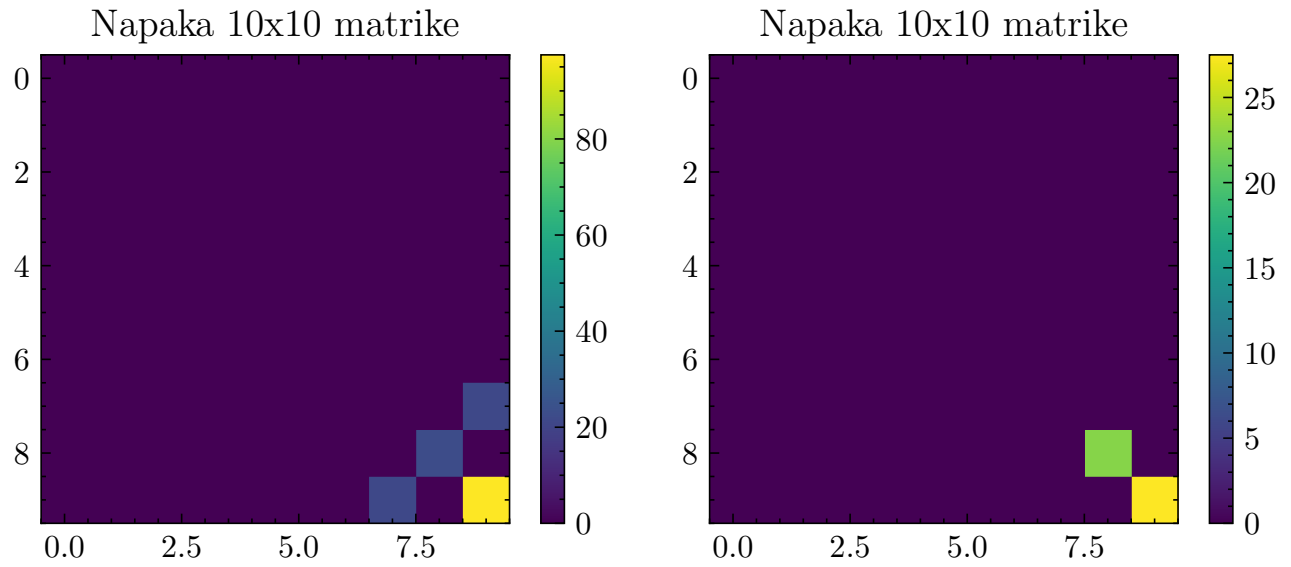
ki ju ni težko izpeljati iz rekurzijskih zvez za Hermitove polinome.

Torej imamo tri pametne načine kako dobiti elemente Hamiltonke. Ti se seveda razlikujejo tako v natančnosti, kot pa tudi v času potrebnem za sestavo. Poglejmo si najprej kako je s časom, odvisnost potrebnega časa od velikosti matrike je za vse tri načine prikazana na grafu 1



Slika 1: Odvisnost potrebnega časa za sestavljanje Hamiltonke v odvisnosti od njene velikosti

Vidimo da je daleč najhitrejša zadnja metoda. Seveda, vemo le da to velja le za našo implementacijo, ki verjetno ni optimalna. Namreč, matrike s katerimi imamo opravka imajo večino členov enakih 0, kar bi lahko upoštevali tako, da bi definirali redko matriko, a ker tega ne storimo je kvadriranje matrike časovno zelo zahtevna operacija. Kljub temu ostanemo pri uporabi načina, ki se ne zanaša na potenciranje matrike saj se s tem izognemo tudi robnim defektom, ki jih srečamo, ko potenciramo matriko, ki bi morala biti neskončna. Napako ki jo na ta način pridelamo z metodama $[q]^4$ in $\left[q^2\right]^2$ predstavimo na grafu 2 kjer dobimo zelo dober občutek za dogajanje na robu matrike:



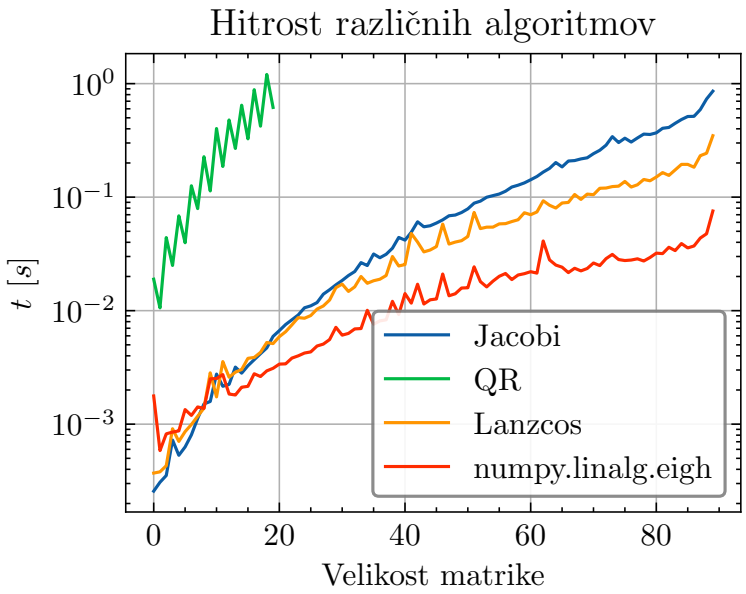
Slika 2: Na levi je prikazana napaka matrike sestavljene kot $[q]^4$ na desni pa kot $\left[q^2\right]^2$. Obe sta prikazani v obliki t.i. *heatmapa*, kjer barva predstavlja vrednost v vsaki celici. Vidimo, da do napak pride res zgolj na robovim matrike.

Vidimo, da je res precej lažje le ne skrbeti in uporabiti $\left[q^4\right]$ način. To nato le pomnožimo z λ in prištejemo H_0 in dobimo našo Hamiltonko.

1.2 Računanje lastnih vrednosti

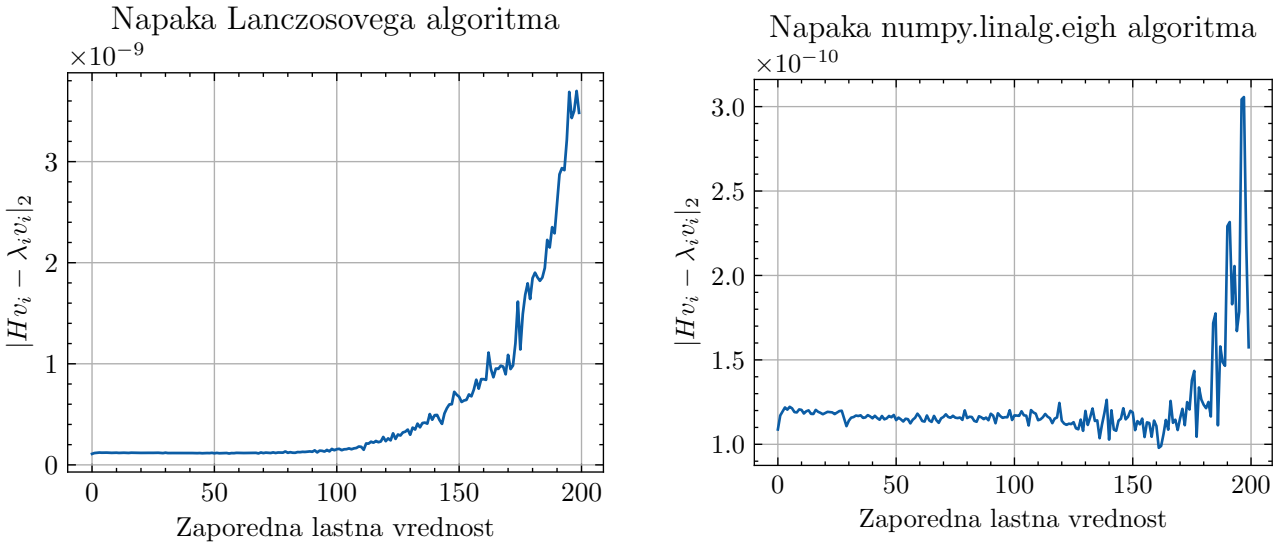
Če želimo rešiti stacionarno Schrödingerjevo enačbo moramo poiskati lastne vrednosti in lastne vektorje Hamiltonke, sepravi nekakšne matrike. Za ta namen obstaja več različnih algoritmov, med drugimi tudi: QR iteracija, Jacobijev algoritem in Lanzcosov algoritem, ki sem jih implementiral na roke v programskem jeziku *Rust*, ki je mnogo hitrejši od *Python* programskega jezika. Na tej točki moram priznati, da je veliko dela zame naredil ChatGPT. Sam sem se namreč šele učil Rust jezika, tako da sem ChatGPT prosil če mi napiše tri omenjene algoritme, kar pa mu je uspelo še preveč dobro. Večino mojega dela je bilo tako le preverjati, da je vse implementirano pravilno, kjer sem moral narediti le manjše popravke. Opazil sem tudi, da bi lahko spet pri vseh algoritmih upošteval da gre za *redke* matrike in tako verjetno mnogo izboljšal učinkovitost kode. Izkazalo se je, da to ni zares potrebno, saj je Rust dovolj hiter, da tudi matrike velikosti 2000 x 2000 ne vzamejo preveč časa.

Ostane nam torej primerjava predvsem med hitrostmi različnih algoritmov in pa v *numpy* vgrajeno knjižnico za diagonalizacijo hermitskih matrik. Primerjava je vidna na sliki 3.



Slika 3: Čas potreben za diagonalizacijo Hamiltonke v odvisnosti od velikosti le te.

Vidimo, da razen za nekaj malih vrednosti Numpy krepko premaga kateregakoli izmed mojih algoritmov. Še najbolje se odreže Lanzcosov algoritem, tako da sem se zanj odločil preveriti tudi natančnost, tako da sem za vsak par lastne vrednosti in pripadajočega lastnega vektorja (λ_i, v_i), matrike H; izračunal: $|Hv_i - \lambda_i v_i|_2$, ki nekako ustreza absolutni napaki naše diagonalizacije. Na ta način je nastal graf 4

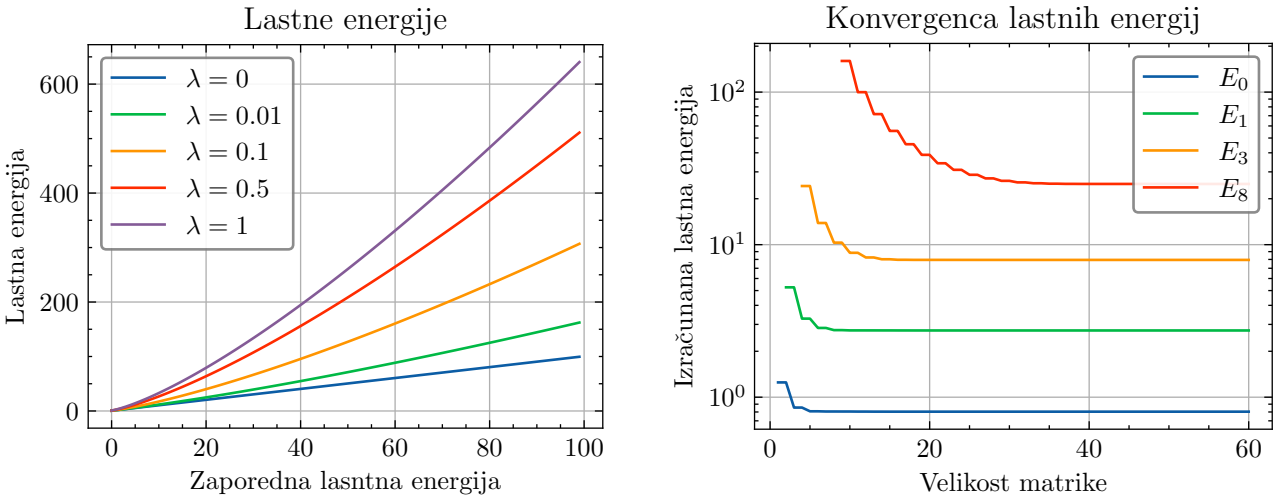


Slika 4: Na levi je prikazana napaka prvih dvesto lastnih vrednosti in vektorjev izračunanih kot $|Hv_i - \lambda_i v_i|_2$ za mojo implementacijo Lanczosovega algoritma in na desni enako za numpy.linalg.eigh algoritem.

Vidimo, da smo lahko precej zadovoljni z natančnostjo našega najuspešnejšega algoritma, čeprav je numpyjeva implementacija desetkrat natančnejša in hitrejša. Kljub dejstvu, da mi je lastna implementacija všeč, sem od te točke dalje uporabljal zgolj še numpy.linalg.eigh.

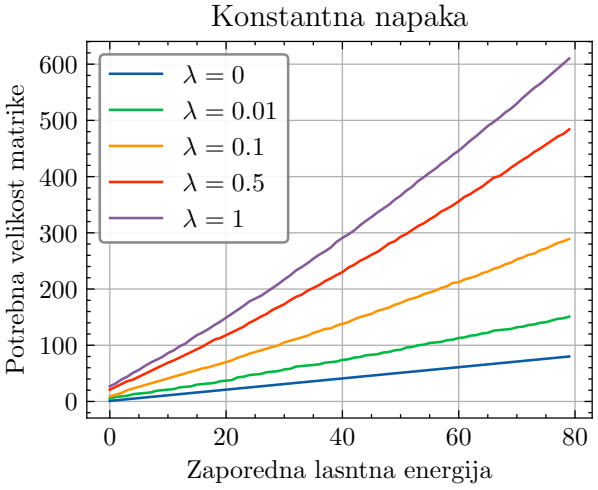
1.3 Lastne energije

Poglaviten del naloge je iskanje lastnih energij motene Hamiltonke. Zanima nas kako so le te odvisne od parametra λ. A predno samo slepo računamo lastne vrednosti Hamiltonke in jih enačimo z lastnimi energijami se moramo vprašati, kako veliko matriko potrebujemo, da izračunamo določeno lastno energijo. Seveda lahko samo preprosto pretiravamo in vzamemo veliko preveliko matriko, recimo 2000 x 2000 in z njo poiščemo prvih 100 lastnih vrednosti, lahko pa analiziramo, ali lastne vrednosti Hamiltonke konvergirajo ko le to večamo. Prav te dve stvari lahko opazujemo na grafu 5.



Slika 5: Na levi strani vidimo, kako se lastne energije večajo z redom vzbuditve, na desni pa vidimo, kako vrednost lastnih energij konvergira z večanjem velikosti Hamiltonke (narisano za primer $\lambda = 0.5$)

Iz grafa konvergence vidimo, da je potrebna vedno večja Hamiltonka, preden se lastna vrednost ustali. Smiselno se je vprašati, kako velika matrika je dejansko potrebna za določeno lastno vrednost, če želimo konstantno absolutno napako manjšo od recimo 10^{-6}. Za oceno napake seveda potrebujemo 'pravo' vrednost posamezne lastne energije, za katero sem vzel preprosto lastne vrednosti izračunane iz matrike z velikostjo 3000. Potrebno velikost Hamiltonke za določene vrednosti parametra λ smo nato poiskali z bisekcijo, in rezultate prikazali na grafu 6



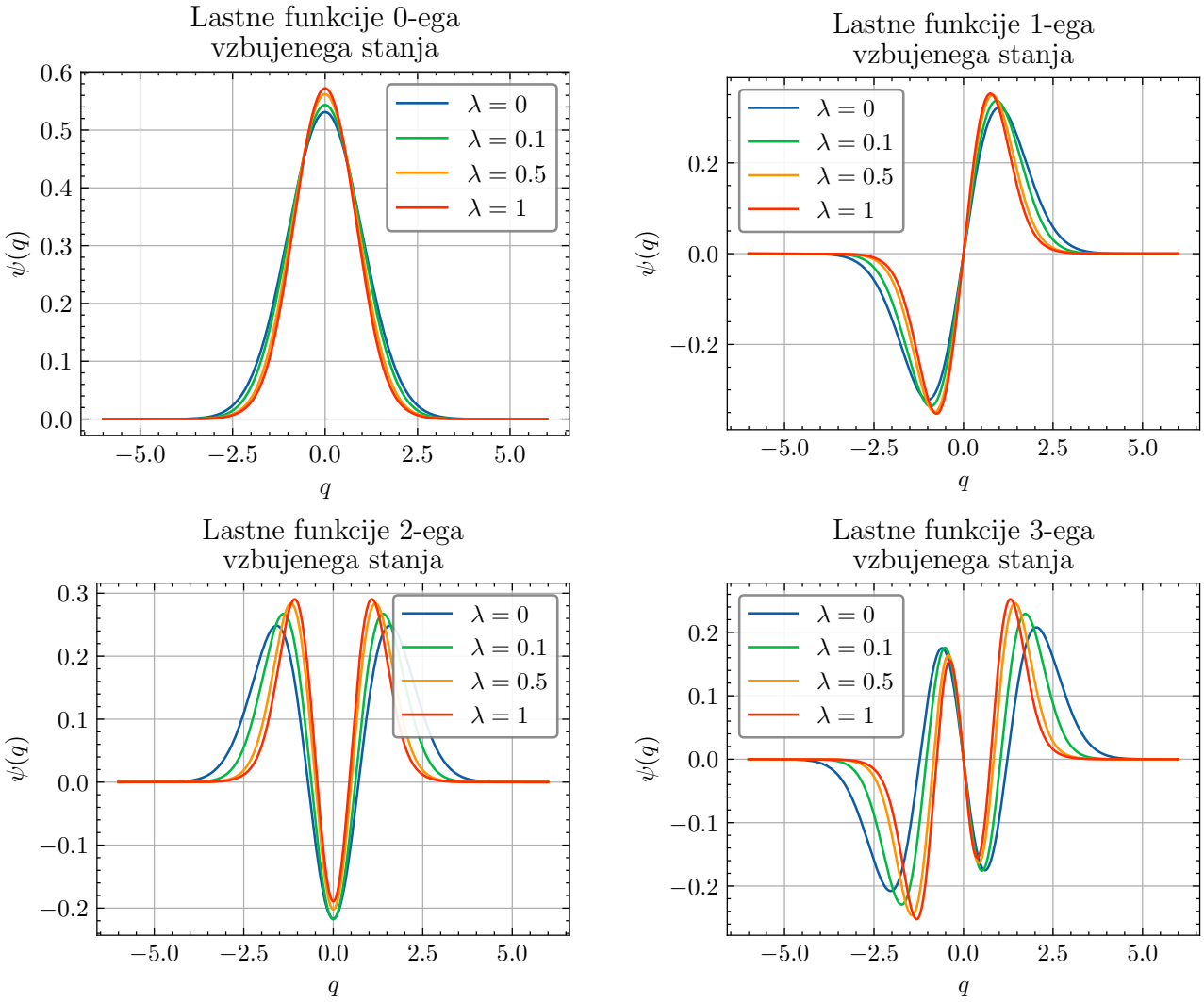
Slika 6: Potrebna velikost Hamiltonke za izračun lastne energije posameznega vzbujenega stanja

Iz dobljenih grafov bi lahko ekstraporilali kako veliko matriko potrebujemo za natančen izračun tudi

mnogo višjih vzbujenih stanj, a ko se približamo potrebni velikosti 3000, naša metoda seveda postane neuporabna.

1.4 Lastne funkcije

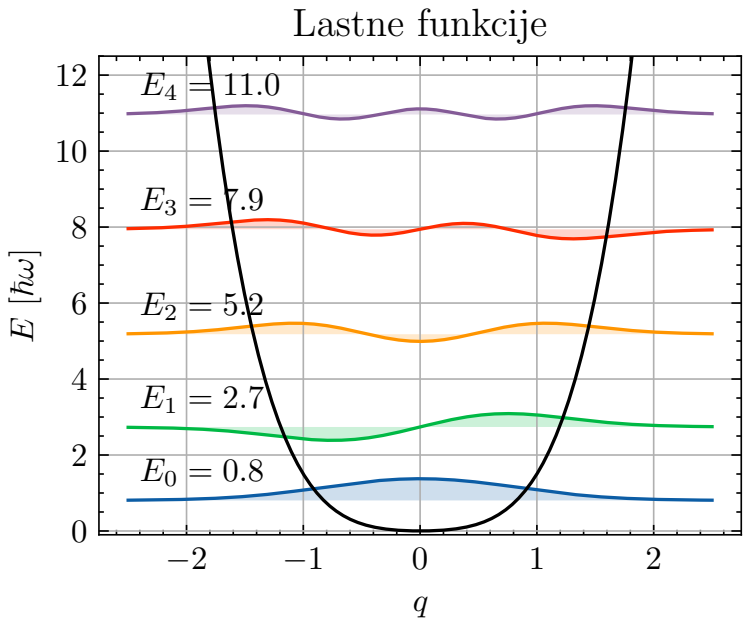
Ko smo računali lastne energije smo pravzaprav vedno zraven računali tudi lastne funkcije Hamiltonke. Pravzaprav smo računali lastne vektorje matrike Hamiltonke, kar pa je pravzaprav zgolj razvoj lastne funkcije po lastnih funkcijah harmonskega oscilatorja, ki je opisan v uvodu. Torej lastne funkcije izračunamo kot linearno kombinacijo lastnih funkcij harmonskega oscilatorja. Prve štiri lastne funkcije za različne λ lahko prikažemo kar na istih oseh, da vidimo kako se funkcije spreminjajo z λ. To prikažemo na grafu 7.



Slika 7: Prve štiri lastne funkcije za različne vrednosti parametra λ

Takoj opazimo nekaj lastnosti. Prva je ta, da so lastne funkcije ali sode ali lihe, kar je, kot smo se pri kvantni mehaniki naučili, posledica simetričnega potenciala. Za tem opazimo tudi, da ima večanje λ nekakšen učinek *stiskanja* lasnte funkcije, samo obliko pa le ta ohranja precej konstantno.

Lastne funkcije lahko narišemo še bolj intuitivno tako da jih narišemo na skupen graf, skupaj s skico potenciala, kjer vsako lastno funkcijo dvignemo za njeno lastno energijo. Točno to smo storili na grafu 8 za parameter $\lambda = 1$



Slika 8: Lastne funkcije anharmonskega oscilatorja za $\lambda = 1$

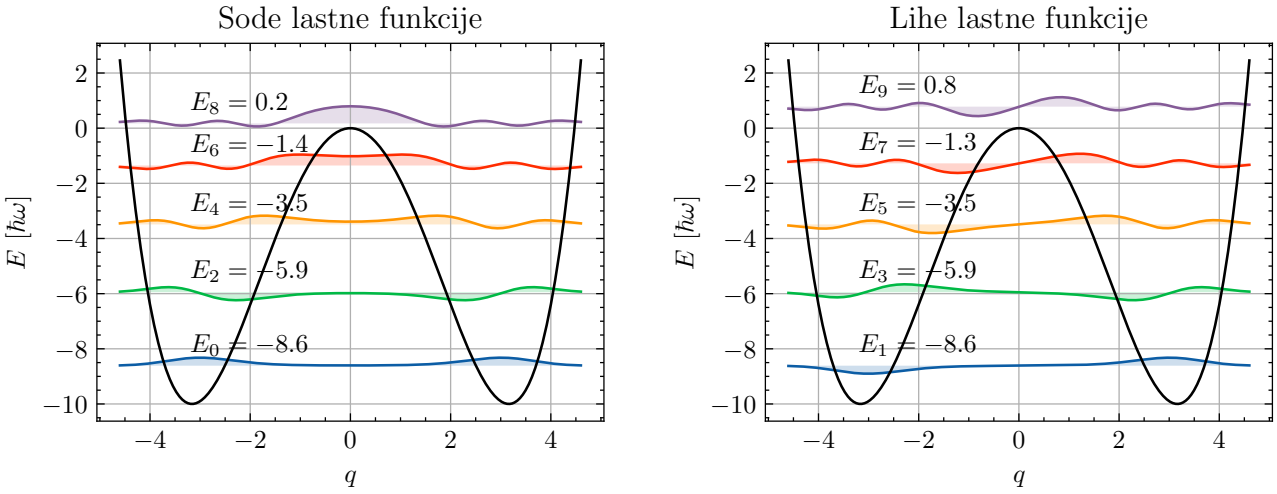
Slednja reprezentacija se mi zdi zelo elegantna, saj jo je veliko lažje interpretirati kot pa vse različne črte na grafu 7

2 Dodatna naloga

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

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

2.1 Reševanje



Slika 9: Sode in lihe lastne funkcije prikazane na ločenih grafih

Samo reševanje problea se praktično nič ne razlikuje od tega, ki smo ga uporabljali do sedaj, tako da

se lahko pravzaprav takoj osredotočimo na rezultat ki je prikazan na sliki 9. Prikaz lastnih funkcij smo razdelili na dva ločena grafa saj so zaporedne sode in lihe lastne energije skoraj enake. Pravzaprav bi skoraj lahko govorili o degeneriranih stanjih, a se izkaže da je tudi med osnovnim in prvim vzbujenim stanjem nekaj malega ($10^{-7}\hbar\omega$) razlike. To da so stanja, še posebaj tista blizu osnovnemu skoraj degenerirana je tudi intuitivno smiselno, saj se pri nizkih energijah delec očitno najverjetneje nahaja v enem izmed minimumov. Ker sta ta dva minimuma relativno daleč narazen je za samo energijo skoraj vseeno ali sta stanja v vsakem minimumu poravnana (soda funkcija) ali ravno obratna (liha funkcija).

3 Zaključek

Po končani nalogi se počutim, kot da bi lahko rešil Schrödingerjevo enačbo za skoraj vsak potencial. To mi da seveda misliti, da je bila naloga uspešno rešena.

Zelo sem zadovoljen, da sem se tekom reševanja odločil pomočiti prste v Rust programski jezik, v katerem sem iskreno precej užival. Poleg tega, sem se naučil marsikaj novega tudi glede algoritmov za diagonalizacijo matrik, ter o matričnem reševanju Schrödingerjevih enačb.

Tudi tokrat mi je uspelo pošteno namučiti računalnik, sploh takrat, ko sem ponesreči dodal eno ničlo ko sem definiral velikost Hamiltonke. Seveda to ni bil edini problem na katerega sem naletel. Ko sem se učil uvažati Rust funkcije v Python sem ponesreči skoraj uničil celotno Python okolje v katerem sem reševal nalogo. Ta, sicer mala, napaka me je stala nekaj dodatnih ur, ki bi jih veliko raje preživel ležeče v postelji.