

## 10. naloga: Diferenčne metode za parcialne diferencialne enačbe

Samo Krejan, 28231092

norazsežna nestacionarna Schrödingerjeva 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  $\Psi^n$  dobimo stanje  $\Psi^{n+1}$ . Matrika  $A$  in vektor  $\Psi$  imata kompleksne elemente, zato račun najlažje opraviš v kompleksni aritmetiki<sup>1</sup>. Izkaže se, da so za zadovoljivo natančnost višji redi nujni!

Z uporabljenim približkom za drugi odvod reda  $\mathcal{O}(\Delta x^2)$  dobimo tridiagonalno matriko. Z diferencami višjih redov dobimio večdiagonalno (pasovno) matriko, a dosežemo tudi večjo krajevno natančnost. Diference višjih redov so bile predstavljene na predavanjih, lahko pa jih 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 differenco računamo, in skrajno levo točko differenčne sheme. Zgornjo tritočkovno sheme za drugo differenco dobimo kot  $FD[2, 2, 1]$ , saj se razpenja čez  $n=2$  intervala, sredinska točka pa je v točki z indeksom  $s=1$ .

## 1 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  $\Delta t$  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

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

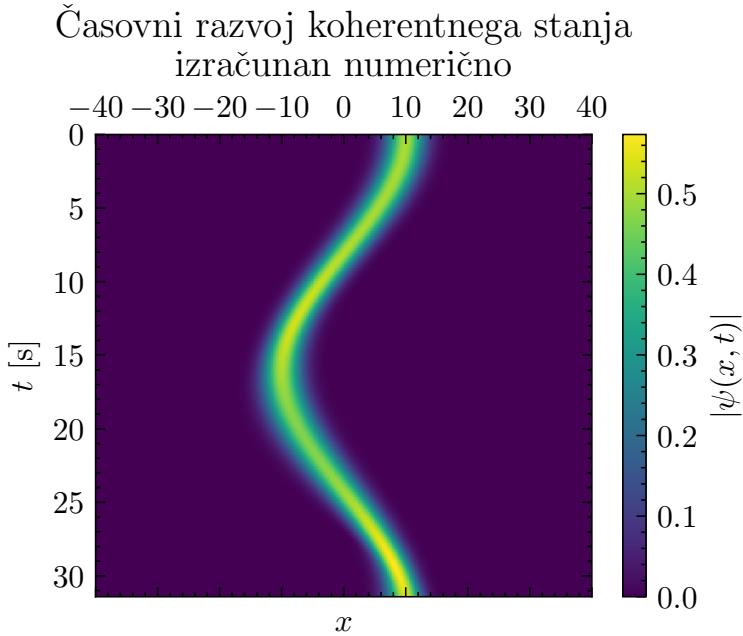
V vseh obravnavanih primerih ugotovi in uporabi dovolj natančno metodo višjega reda.

### 1.1 Harmonski potencial

Najprej sem si ogledal numerično izračunani razvoj skozi eno periodo (300 časovnih točk). Kot pričakovano sem dobil sliko nihanja valovnega paketa, ki kvalitativno deluje popolnoma pravilno:

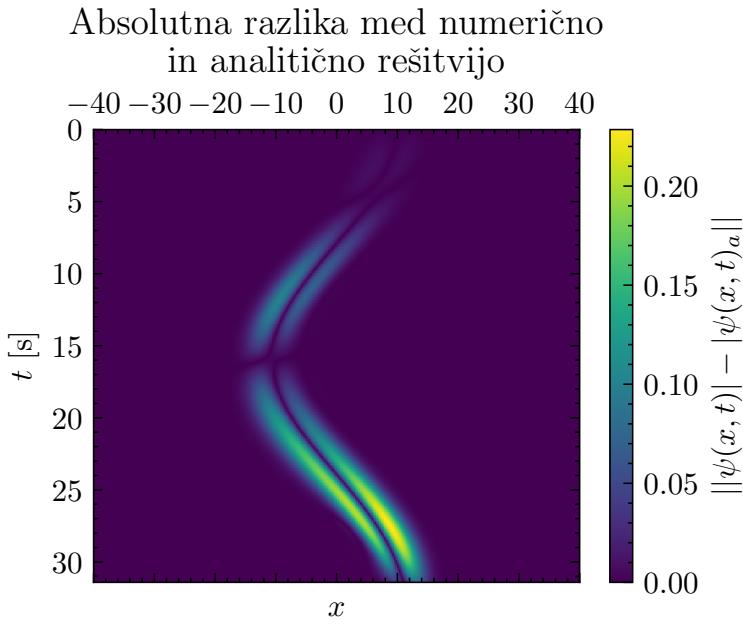
---

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



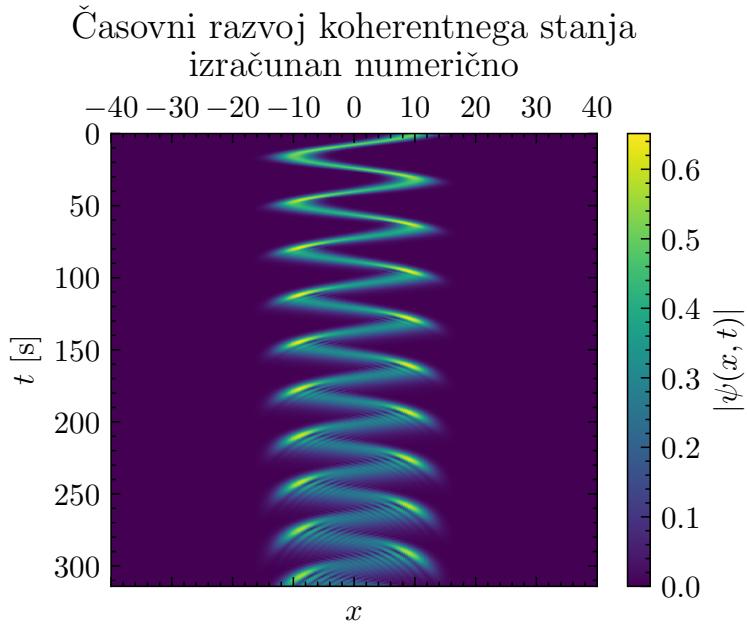
Slika 1: Časovni razvoj valovnega paketa izračunan numerično. Izgleda kot bi pričakovali.

A zadeva postane mnogo manj optimistična, ko primerjam dobljeni rezultat z analitičnim. Čeprav kvalitativno izgledata identično se čez čas očitno nakopiči kar nekaj numeričnih napak, ki povzročijo vedno slabše rezultate. Grafigranje razlike med numerično in analitično rešitvijo proizvede graf 2

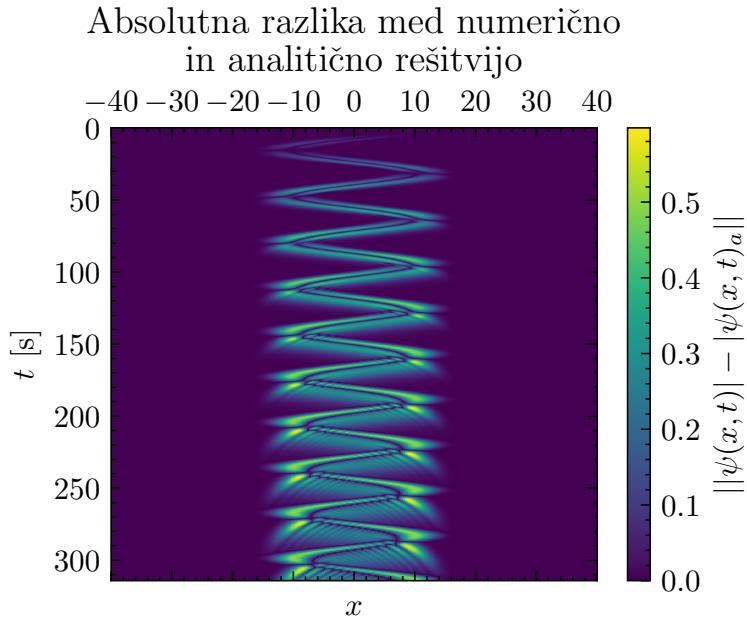


Slika 2: Napaka numerične metode skozi čas. Opazimo lahko, da dlje kot opazujemo zadevo, slabše postaja.

Že po enem samem nihaju je napaka ponekod veika skoraj polovico amplitудe valovne funkcije. S časom postaja zgodba vedno slabša, če si pogledamo rezultate za deset nihajev (3000 časovnih točk) (sliki 3, 4) vidimo, da proti koncu tudi že kvalitativno postane jasno, da se numerična napaka akumulira. Napaka je na tej točki že skoraj tako velika kot sama amplituda valovne funkcije.



Slika 3: Časovni razvoj valovnega paketa izračunan numerično. Proti koncu zaradi numeričnih napak izgubi pričakovano obliko.

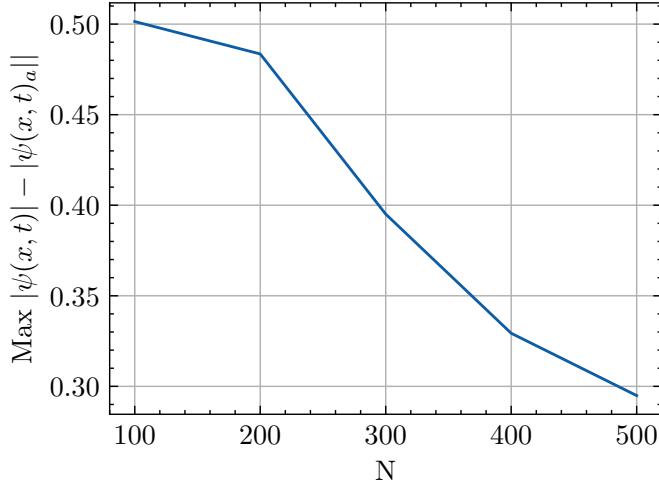


Slika 4: Napaka numerične metode skozi čas. Po desetih nihajih je že zelo slabo.

## 1.2 Analiza metode

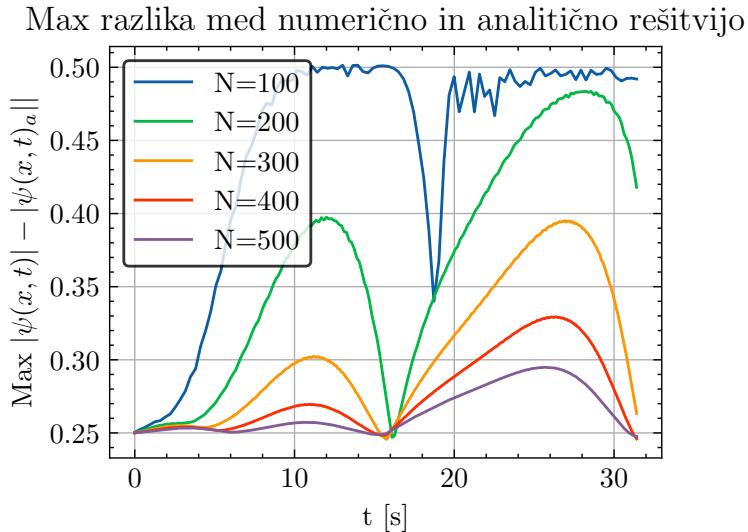
Skozi en nihajni čas s tristo časovnimi točkami sem za različna števina aktivnih točk  $N$  analiziral natančnost in "performance" metode. Najprej sem si ogledal kakšno maksimalno napako metoda pridela v enem nihaju v odvisnosti od  $N$ . Rezultat je prikazan na grafu 5

Max razlika med numerično in analitično rešitvijo



Slika 5: Maksimalno odstopanje numerične metode od analitično izračunane valovne funkcije skozi en nihaj v odvisnosti od  $N$ .

Raziskoval sem tudi, na katerih točkah metoda doseže te napake, zato sem grafigral tudi maksimalno odstopanje numerične metode skozi čas. Tako sem pridelal graf 6

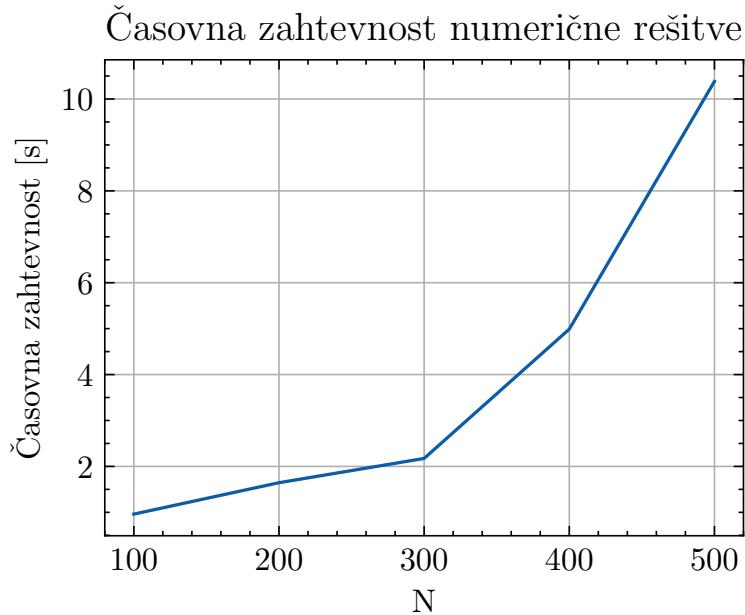


Slika 6: Maksimalno odstopanje numerične metode skozi čas za različne  $N$ .

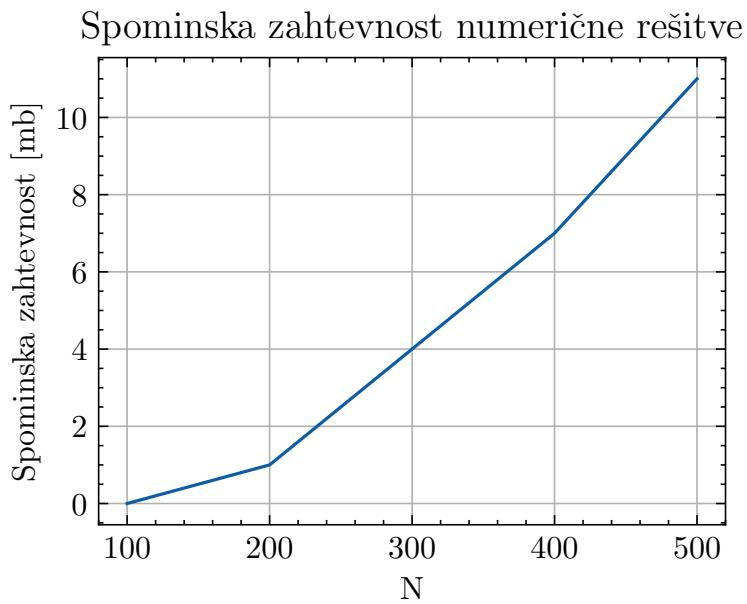
Na grafu je opaziti nekakšno nihanje, kar me je vzpodudilo k nekaj dodatne raziskave. Našel sem poročilo Simona Bukovška<sup>2</sup> - enega izmed predhodnikov, ki je opazil, da pravzaprav niha amplituda valovnega paketa, ko izračunamo valovno funkcijo numerično. Izkaže se, da se tega efekta ne da znebiti niti s 3000 aktivnimi točkami, kar je več kot bi mojemu računalniku uspelo prebaviti v zglednem času, zato višjih vrednosti nisem preverjal.

Očitno sicer vidimo, da se natančnost izboljuje z večanjem  $N$ , a se s tem seveda veča potreben čas in spomin za izvedbo. Za odločitev kako velik  $N$  se splača uporabiti, se mi je zdelo smiseln raziskati časovno in spominsko zahtevnost metode. Rezultate sem prikazal na grafih 7 in 8

<sup>2</sup>Glej [https://sim0nbu.github.io/Simonovi-fizikalni-zapiski/Files/Mafijski\\_20praktikum/simon'bukovsek'10.pdf](https://sim0nbu.github.io/Simonovi-fizikalni-zapiski/Files/Mafijski_20praktikum/simon'bukovsek'10.pdf) pg. 7



Slika 7: Časovna zahtevnost metode v odvisnosti od N

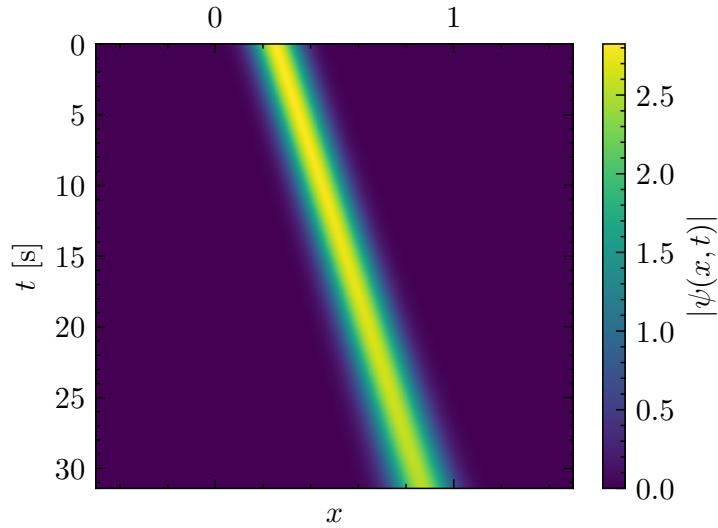


Slika 8: Spominska zahtevnost metode.

### 1.3 Prost valovni paket

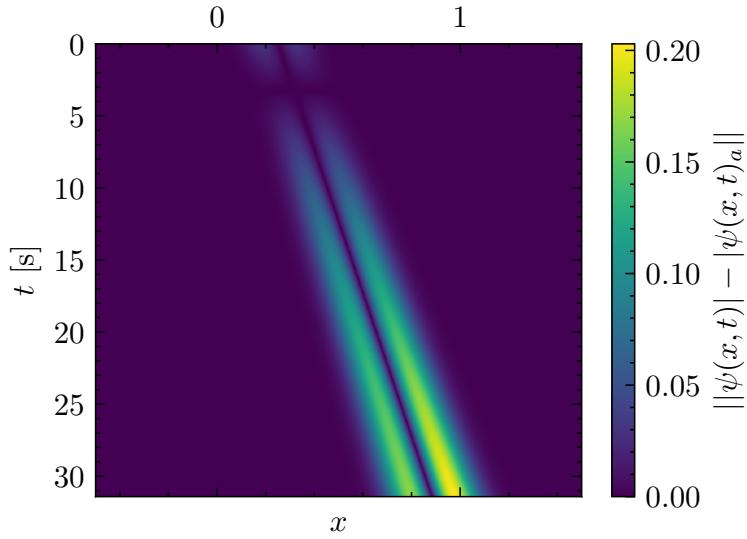
V drugem delu sem si pogledal še kako se širi valovni paket izven potenciala. Delo je bilo razen spremembe začetnega stanja in implementacije potenciala, identično kot prej, zato bom na tej točki le predstavil rezultat metode ter odstopanje le te od analitične rešitve (slike 9 in 10)

Časovni razvoj Gaussovega valovnega paketa  
izračunan numerično



Slika 9: Časovni razvoj valovnega paketa izračunan numerično. Rezultat je kvalitativno pravilen.

Absolutna razlika med numerično  
in analitično rešitvijo



Slika 10: Napaka numerične metode skozi čas. Proti koncu opazimo, da je napaka že res precej velika.

Napake so bile v obeh delih naloge zares precej velike. Smiselno bi bilo uporabiti metode višjega reda  
...

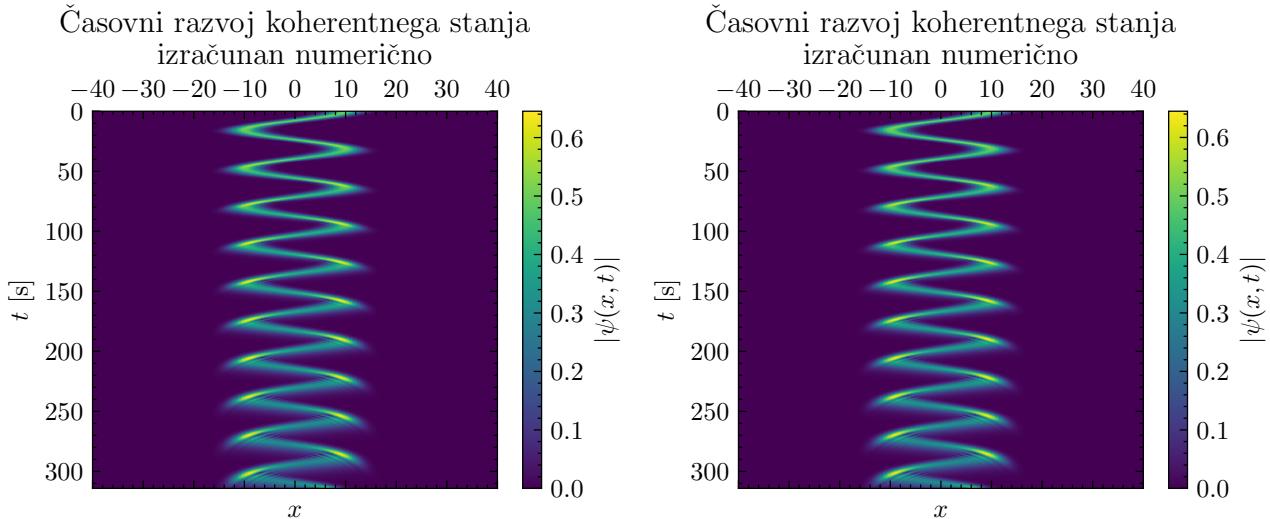
#### 1.4 Metode višjega reda

Če sem čisto iskren, sem skoraj pozabil vključiti metode višjega reda, zato sem jih nekoliko nerodno umestil sem, a se mi zdi, da je razvrstitev še vseeno smiselna. Edina potrebna sprememba v implementaciji je bila v tem kako zgenerirati matriko  $A$ , kjer pa je bilo preprosto potrebno slediti receptu opisanemu na predavanjih. Tako sem implementiral še metodo drugega in tretjega reda, s katerima

sem pravzaprav ponovil delo in se posledično tudi ne bom ukvarjal preveč z besedami, pač pa bom le predstavil rezultate.

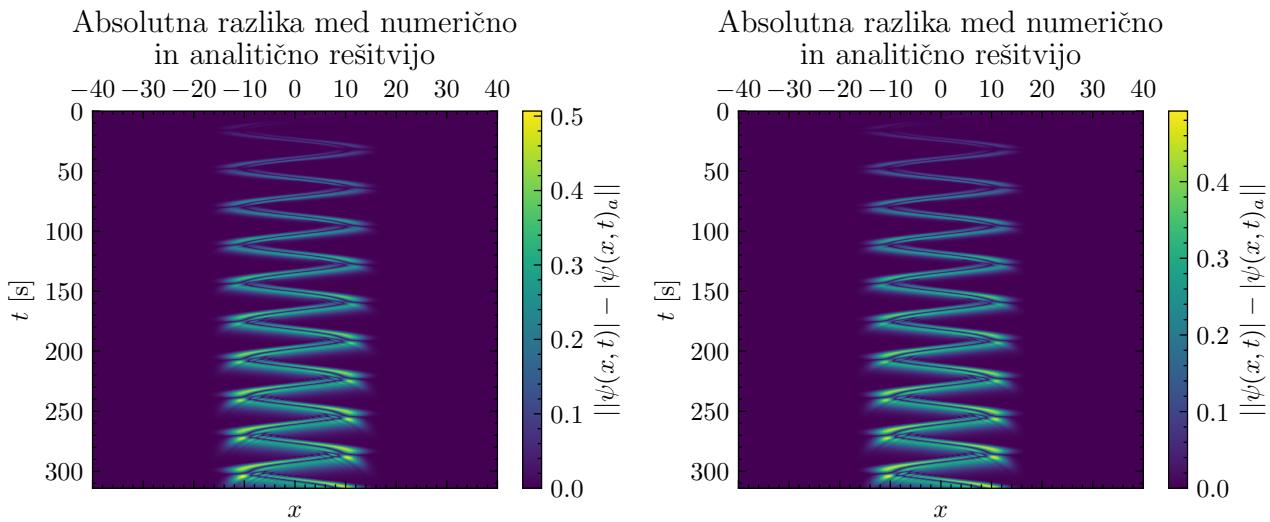
#### 1.4.1 Harmonski potencial

Najprej sem z višjim redom narisal razvoj v harmonskem otencialu 1.4.1



Slika 11: Časovni razvoj valovnega paketa izračunan numerično, za drugi red (levo) in tretji red (desno)

Ter nato še napako posamezne metode 1.4.1

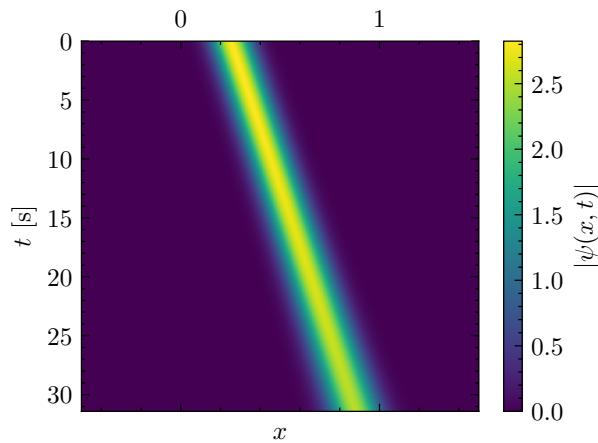


Slika 12: Napaka metode drugega reda (levo) in tretjega reda (desno) skozi čas

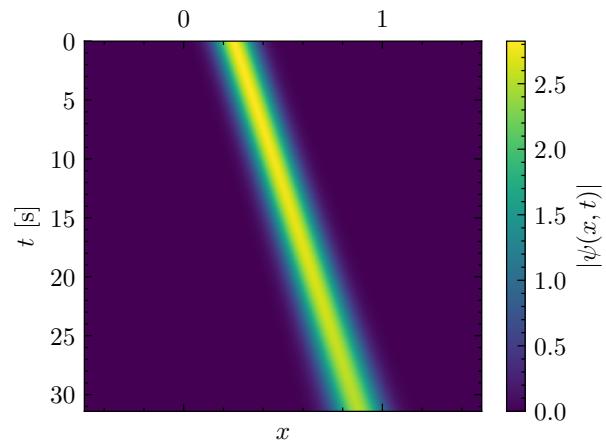
#### 1.4.2 Prost valovni paket

Najprej sem z višjim redom narisal razvoj prostega valovnega paketa skozi čas 1.4.2

Časovni razvoj Gaussovega valovnega paketa izračunan numerično



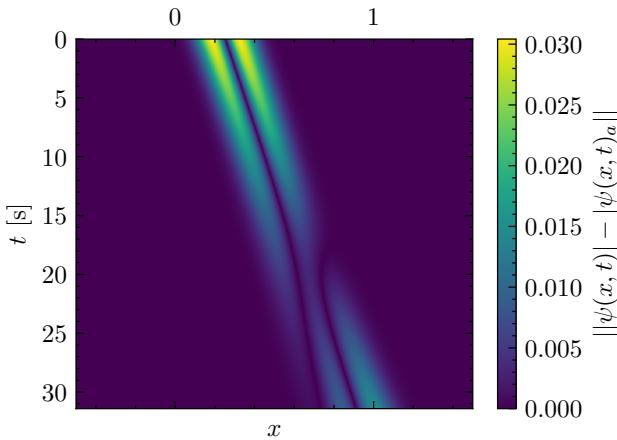
Časovni razvoj Gaussovega valovnega paketa izračunan numerično



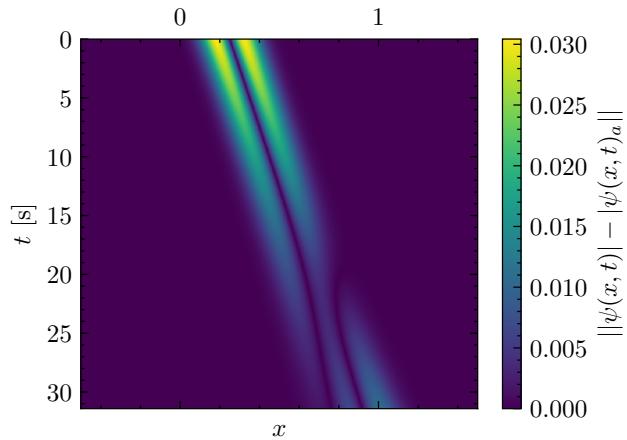
Slika 13: Časovni razvoj valovnega paketa izračunan numerično, za drugi red (levo) in tretji red (desno)

Ter nato še napako posamezne metode 1.4.2

Absolutna razlika med numerično in analitično rešitvijo



Absolutna razlika med numerično in analitično rešitvijo

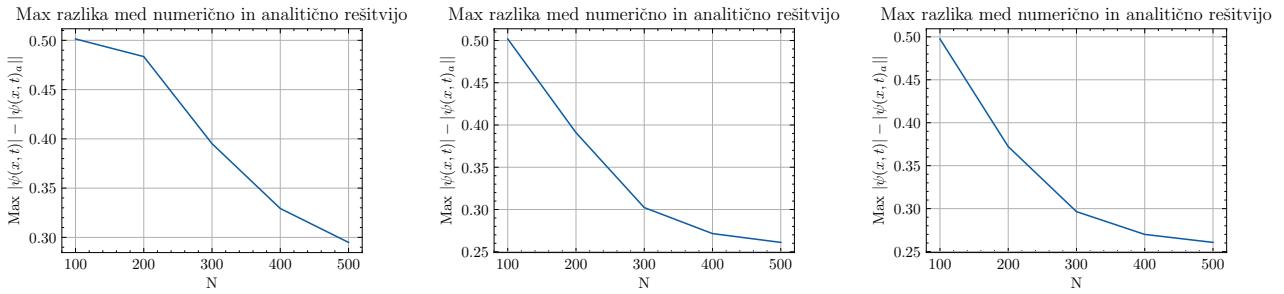


Slika 14: Napaka metode drugega reda (levo) in tretjega reda (desno) skozi čas

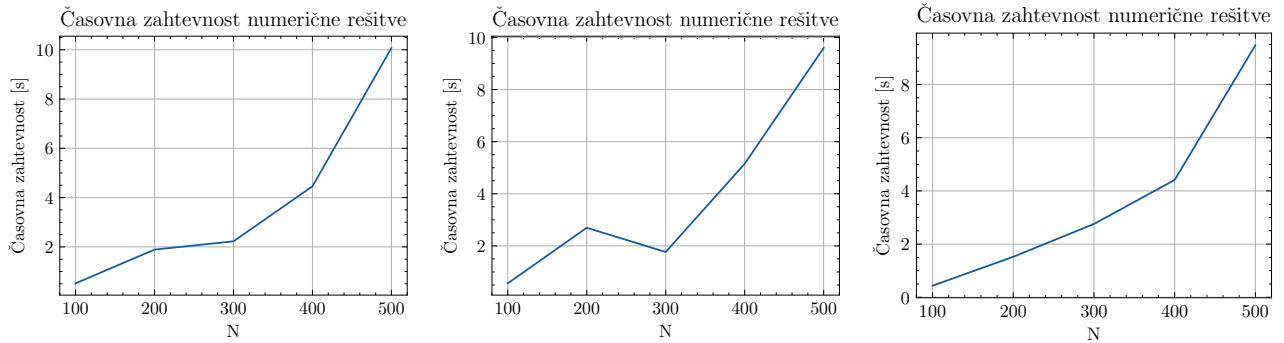
Dejstva, da se napaka s časom manjša nisem znal pojasniti.

#### 1.4.3 Primerjava različnih redov metod

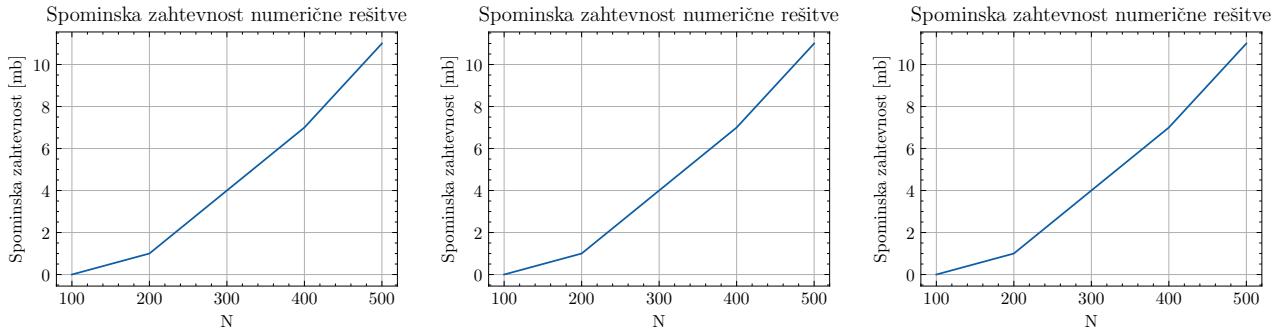
Primerjal sem natančnost, hitrost in spominsko zahtevnost pri prvem, drugem in tretjem redu metode, za deset nihajev in na ta način dobil slednje rezultate 1.4.3, 1.4.3 in 1.4.3



Slika 15: Od leve proti desni maksimalna razlika med numerično in analitično rešitvijo za prvi, drugi in tretji red.



Slika 16: Od leve proti desni potreben čas za izračun za prvi, drugi in tretji red.



Slika 17: Od leve proti desni potreben spomin za izračun za prvi, drugi in tretji red.

Rezultati kažejo na majhne razlike med metodami - razlike ki sploh niso statistično pomembne. Tukaj je sicer treba priznati, da nisem nobene matrike optimiziral, tako da sem vedno shranil kar celo matriko  $A$  in s tem računsko naredil vse tri metode enako zahtevne. Navkljub rezultatom primerjave pa se lahko samo s kvalitativnim ogledom rezultatov, strinjam, da sta metodi višjega reda bolj uspešni - sploh pri prostem valovnem paketu.

## 2 Zaključek

Naloga se je začela zelo obetavno. Užival sem v implementaciji metod in v lastni zavestni odločitvi, da se omejim na uporabo Pythona s prijaznim Numpy paketom, saj si ne predstavljam slednjega implementirati v nižje-nivojnem jeziku. Vseeno pa mi je tukaj uspelo priti do težav, ker nisem dovolj posplošil funkcij in parametrov v kodi in sem imel zato, ker sem delal v Jupyterovem zvezku, kar nekaj težav s tem, kdaj ima katera spremenljivka katero vrednost. Pomagalo bi seveda tudi boljše

poimenovanje spremenljivk in manj nepremišljenega „copy-pastanja“. A ko sem že videl ciljno ravnino, se mi ni zdelo vredno spremnijati kode, čeprav bi mi to morda celo olajšalo delo. No in seveda se opravičujem tudi vam, če se boste lotili brati kodo za mano - ne predstavljam si, da zna biti prijetna izkušnja.