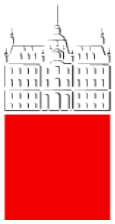
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Diferenčne metode za parcialne diferencialne enačbe

10. naloga pri Matematično-fizikalnem praktikumu

Avtor: Marko Urbanč (28191096)
Predavatelj: prof. dr. Borut Paul Kerševan

4.9.2023

Kazalo

1 Uvod

V prejšnji nalogi smo rekli, da obstajata v glavnem dva velika razreda za reševanje parcialnih diferencialnih enačb (PDE). To sta spektralne metode, ki smo jih raziskali v prejšnji nalogi in pa diferencialne metode, ki jih spoznamo tu. Diferencialne metode so v glavnem metode, ki rešujejo PDE tako, da jih diskretizirajo in jih pretvorijo v sistem linearnih enačb. Te metode so v glavnem zelo podobne kot metode za reševanje sistemov navadnih diferencialnih enačb (ODE). V glavnem se razlikujejo v tem, da so PDE lahko tudi nelinearne in moramo posledično uporabiti iterativne metode za reševanje sistemov linearnih enačb. Tu bomo spoznali metodo končnih diferenc (FDM). Ta temleji na Taylorjevem razvoju s katerim lahko aproksimiramo odvod funkcije. To aproksimacijo nato vstavimo v PDE in dobimo sistem linearnih enačb. Ta sistem nato rešimo iterativno in dobimo končno rešitev.

Fizikalni kontekst za to nalogo bo reševanje enodimenzionalne nestacionarne Schrödingerjeve enačbe, ki se glasi

$$\left(i\hbar\frac{\partial}{\partial t} - H\right)\psi(x,\, t) = 0\,. \tag{1}$$

Predstavlja osnovno orodje za nerelativistični opis kvantnih sistemov. V enačbi 1 je H Hamiltonian sistema, ki je v splošnem odvisen od časa. V našem primeru bomo obravnavali časovno neodvisen Hamiltonian

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

Z menjavo spremenljivk $H/\hbar \to H$, $x\sqrt{m/\hbar} \to x$ efektivno postavimo $\hbar = m = 1$. V tem primeru je Hamiltonian enak

$$H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x)\,. \tag{3}$$

Razvoj stanja $\psi(x,\, t)$ v času $\psi(x,\, t + \Delta t)$ je opisan z približkom

$$\psi(x,\, t + \Delta t) = e^{-iH\Delta t}\psi(x,\, t) \approx \frac{1 - \frac{1}{2}iH\Delta t}{1 + \frac{1}{2}iH\Delta t}\psi(x,\, t)\,. \tag{4}$$

Območje $x \in [a,\, b]$ diskretiziramo na krajevno mrežo z N točkami $x_j = a + j\Delta x$, kjer je $\Delta x = (b - a)/(N - 1)$. Časovni razvoj spremljamo ob časovni mreži z M točkami $t_m = m\Delta t$, kjer je Δt časovni korak. Vrednosti valovne funkcije in potenciala v mrežnih točkah ob času t_m označimo z ψ_j^m in V_j. Krajevni odvod izrazimo z diferenco

$$\Psi''(x) \approx \frac{\psi(x + \Delta x,\, t) - 2\psi(x,\, t) + \psi(x - \Delta x,\, t)}{\Delta x^2} = \frac{\psi_{j+1}^m - 2\psi_j^m + \psi_{j-1}^m}{\Delta x^2}\,. \tag{5}$$

Te približke vstavimo v razvoj stanja 4 in razpišemo Hamiltonov operator, da dobimo sistem enačb

$$\psi_j^{m+1} - i\frac{\Delta t}{4\Delta x^2}\left[\psi_{j+1}^{m+1} - 2\psi_j^{m+1} + \psi_{j-1}^{m+1}\right] + i\frac{\Delta t}{2}V_j\psi_j^{m+1} = \psi_j^m + i\frac{\Delta t}{4\Delta x^2}\left[\psi_{j+1}^m - 2\psi_j^m + \psi_{j-1}^m\right] - i\frac{\Delta t}{2}V_j\psi_j^m \,. \quad (6)$$

v notranjih točkah mreže, medtem ko na robu (pri $j \leq 0$ in $j \geq N$) postavimo $\psi_j^m = 0$. Vrednosti valovne funkcije uredimo v vektor Ψ^m in sistem 6 prepišemo v matrično obliko

$$\mathbf{A}\Psi^{m+1} = \mathbf{A}^*\Psi^m \,, \quad (7)$$

kjer je $\mathbf{A}$ tridiaonalna matrika z elementi

$$b = 1 + i\frac{\Delta t}{2\Delta x^2} \,, \quad a = -\frac{b}{2} \,, \quad d_j = 1 + b + i\frac{\Delta t}{2}V_j \,. \quad (8)$$

Torej je $\mathbf{A}$ oblike

$$\mathbf{A} = \begin{bmatrix} d_1 & a & 0 & \cdots & 0 \\ a & d_2 & a & \cdots & 0 \\ 0 & a & d_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_{N-1} \end{bmatrix} . \quad (9)$$

Kot smo napovedali, je to torej matrični sistem, ki ga moramo rešiti v vsakem časovnem koraku iterativno.

2 Naloga

Naloga je sestavljena iz dveh delov. V prvem delu naloga od nas zahteva, da spremljamo časovni razvoj začetnega stanja

$$\Psi(x,\, 0) = \sqrt{\frac{\alpha}{\sqrt{\pi}}}e^{-\alpha^2(x-\lambda)^2/2} \,, \quad (10)$$

v harmonskem porencialu $V(x) = 1/2 \cdot kx^2$, kjer je $\alpha = k^{1/4}$ in $\omega = \sqrt{k}$. Analitična rešitev za to stanje je

$$\psi(x,\, t) = \sqrt{\frac{\alpha}{\sqrt{\pi}}}\exp\left[-\frac{1}{2}(\xi - \xi_\lambda\cos\omega t)^2 - i\left(\frac{\omega t}{2} + \xi\xi_\lambda\sin\omega t - \frac{1}{4}\xi_\lambda^2\sin 2\omega t\right)\right] , \quad (11)$$

kjer je $\xi = \alpha x$ in $\xi_\lambda = \alpha\lambda$. Postavimo $\omega = 0.2$ in $\lambda = 10$. Za krajevno mrežo vzamemo razpon $x \in [-40, 40]$ in $N = 300$ točk. Nihajni čas je $T = 2\pi/\omega$, zato primerno prilagodimo časovni korak Δt in stanje opazujemo deset period. V drugem delu naloge pa moramo spremljati razvoj gaussovskega valovnega paketa

$$\Psi(x,\, 0) = (2\pi\sigma_0^2)^{-1/4}e^{ik_0(x-\lambda)}e^{-(x-\lambda)^2/(2\sigma_0)^2} \,. \quad (12)$$

v praznem prostoru $V(x) = 0$. Analitična rešitev za to stanje je

$$\psi(x,\, t) = \frac{(2\pi\sigma_0^2)^{-1/4}}{\sqrt{1 + it/(2\sigma_0^2)}}\exp\left[\frac{-(x-\lambda)^2/(2\sigma_0^2) + ik_0(x-\lambda) - ik_0^2 t/2}{\sqrt{1 + it/(2\sigma_0^2)}}\right] \quad (13)$$

3 Opis reševanja

Zdaj praktično delam "faks zaključekšpeedrun, torej ta naloga žal ni tako podrobno rešena, kot bi mogoče želel. Uporabil sem standardni ansambel programskih paketov, ki sem jih uporabljal tudi pri prejšnji nalogi. Torej `numpy`, `scipy` in `matplotlib`. Pri tej nalogi žal nisem imel možnosti (zaradi omejenega časa), da bi delal kakšne divje eksperimente, torej je to praktično to kar se tiče programske opreme.

3.1 FDMSolver

Programersko sem se naloge lotil identično kot pri prejšnji nalogi. Torej sem napisal razred `FDMSolver`, ki vsebuje vse potrebne metode za reševanje te naloge. V konstruktorju razreda `FDMSolver` se inicializira vse potrebno, od tam naprej pa lahko uporabnik kliče razne funkcije. Večina jih je pravzaprav za risanje, saj je razred priročen storage za vse potrebne podatke in posledično risarski funkciji ni potrebno podati ogromno argumentov. V glavnem je razred `FDMSolver` zelo podoben kot `SpectralSolver` iz prejšnje naloge. Glavna razlika in cel point nalog pa je njegova metoda `solve()`. Tu se incializirajo še potrebne matrike in se začne iterativno reševanje sistema. Pravzaprav tu tokrat nisem naredil res čisto nič posebnega, samo običajno matrično množenje `numpy ndarray`-ov. Se mi zdi škoda, ker bi lahko preverjal svojo implementacijo proti kaki, ki jo ima recimo `scipy` ali pa `numpy`. Recimo `scipy.sparse.linalg.bicgstab()` se mi. Ali pa v našem primeru, ker imamo tridiagonalno matriko bi moral dobro delati `scipy.linalg.solve_banded()`. Anyways žal tega nisem imel časa narediti.

4 Rezultati

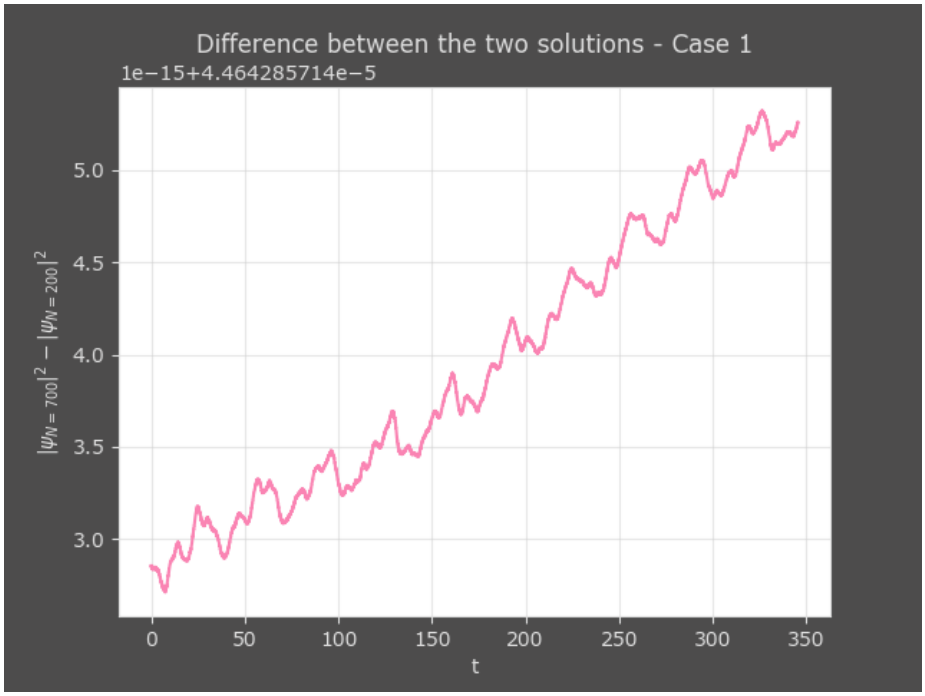
4.1 Prvi primer - harmonski oscilator

Za prvi primer sem uporabil začetno stanje 10. Začetno stanje sem diskretiziral na mrežo z $N = 300$ točkami, vsaj prvotno. Izkazalo se je, da pri nastavitvah, ki so predlagane v nalogi, rešitev čez nekaj nihajev razpade. Nedopustno. Čas sem spremljal originalno na mreži z $M = 1100$ točkami torej 100 na periodo, kjer je bilo period potem 11. To se mi je zdelo sila bedno. Plačal sem za cel procesor, uporabljal bom cel procesor. Tako sem časovno mrežo razširil na $M = 11000$ točk. Matrično množenje v `numpy` je že po defaultu paralelizirano, torej very nice.

Zdelo se mi je zanimivo, da bi poiskal okvirno koliko točk potrebjemo za smiselno rešitev (torej, takšno, ki ne razpade po 10 nihajih). Ugotovil sem, da je spodnja meja okoli $N = 200$. Zdaj, če bi imel čas bi šel seveda te stvari vse dejansko izvest, ampak zanimivo bi bilo, kolikšen je dejansko performance gain oz. krajši čas računanja. Ker meni se je zdelo $N = 200$, $M = 11000$ opazno hitrejše kot $N = 300$, $M = 1100$. Recimo če zdaj gledamo large scale in si predstavljamo, da rešujemo res nek zakompliciran velik model, na porazdeljenem sistemu, se mi zdi zelo koristna metrika, za koliko več elektrike (oz. cloud hosting resources) bo potrebno plačati za tistih več točk in kolikšna bo izboljšava

v dobljenem rezultatu.

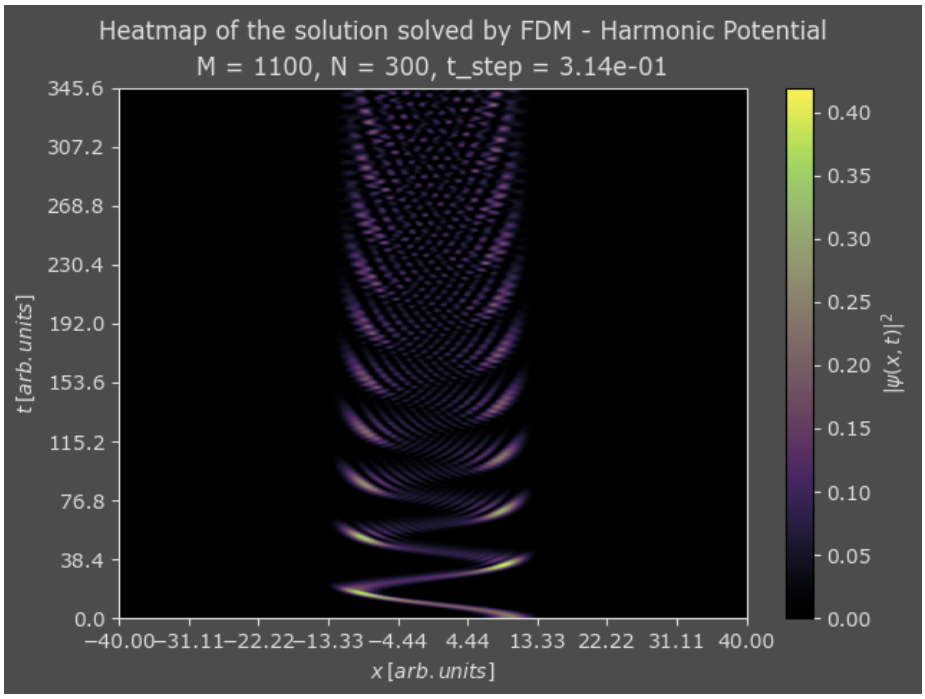
Predstavim recimo lahko tale graf, ki se mi je zdel razmeroma zanimiv (pa zato, ker drugih, niti nimam). Predstavlja razliko med rešitvijo z zelo gosto mrežo $N = 700$ in z našo domnevno spodnjo mejo $N = 200$. Časovni grid je bil v obeh primerih $M = 11000$. Vidimo, da je razlika v rešitvi zelo zelo majhna, do te mere, da me skrbi, da sem kje kaj zamočil in tega nisem pravočasno opazil. Ampak vizualno, ko človek gleda dobljene animacije zgledata res identično.



Razlika med rešitvama z $N = 700$ in $N = 200$

Pa ja, razlika med rešitvama definitivno narašča z časom, kot je pričakovano, saj se rešitve še vedno kvarijo in sklepam, da se tista pri $N = 200$ kvari hitreje, ampak bralec, poglej ta red velikosti razlike.

Kakorkoli zdaj je dovolj o tem, da sem se igral z različnimi parametri. Glavni obrok je serviran! Na sliki **??** je prikazana rešitev pri originalnih parametrih, torej $N = 300$, $M = 1100$.



Rešitev pri originalnih parametrih.

5 Komentarji in izboljšave

Literatura