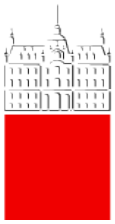
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Spektralne metode za začetne probleme PDE

9. naloga pri Matematično-fizikalnem praktikumu

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

1.9.2023

Kazalo

1 Uvod

Parcialne diferencialne enačbe (PDE) lahko rešujemo na več različnih načinov. Glavna razlika je v tem, kako diskretiziramo prostor in čas. V tej nalogi bomo reševali PDE z spektralnimi metodami.(Drugo možnost; diferenčne metode, bomo obravnavali v naslednji nalogi.) Pri spektralnih metodah diskretiziramo prostor s tem, začetni pogoj izrazimo z nekim naborom baznih funkcij in nato iščemo rešitev tako, da računamo, kako se koeficienti teh baznih funkcij spreminjajo s časom. V tej nalogi bomo preizkusili reševanje preko Fourierove metode in metode končnih elementov s kubičnimi B-zlepki.

1.1 Fourierova metoda

Fizikalno gledano rešujemo enodimenzionalno toplotno enačbo, torej difuzijsko enačbo, v homogeni neskončni plasti, s končno debelino a, brez izvirov in ponorov toplote.

$$D\frac{\partial^2 T}{\partial x^2} = \frac{\partial T}{\partial t} \quad 0 \leq x \leq a, \quad D = \frac{\lambda}{\rho c}\,. \tag{1}$$

Če Temperaturo $T(x, y)$ izrazimo kot Fourierovo vrsto dobimo

$$T(x,t) = \sum_{k=0}^{N-1} \hat{T}_k(t) \exp\left(\frac{-2\pi i k x}{a}\right). \tag{2}$$

Torej se PDE (1) zdaj zapiše kot

$$\sum_{k=0}^{N-1}\left(-\frac{4\pi^2 k^2 D}{a^2}\right)\hat{T}_k(t)\exp\left(-\frac{2\pi i k x}{a}\right) = \sum_{k=0}^{N-1}\left(\frac{\mathrm{d}\hat{T}_k(t)}{\mathrm{d}t}\right)\exp\left(-\frac{2\pi i k x}{a}\right). \tag{3}$$

Torej se naša naloga prevede na iskanje koeficientov $\hat{T}_k(t)$, ki jih dobimo preko **evolucijske enačbe**

$$\frac{\mathrm{d}\hat{T}_k(t)}{\mathrm{d}t} = -\frac{4\pi^2 k^2 D}{a^2}\hat{T}_k(t)\,. \tag{4}$$

Pogosto se uporabi spektralno reprezentacijo za krajevni odvod, časovni korak pa naredimo z neko ekspliticitno metodo. V našem primeru bomo uporabili **Eulerjevo metodo**.

$$\hat{T}_k(t+k) = \hat{T}_k(t) + \frac{4\pi^2 k^2 D}{a^2}\hat{T}_k(t)k\,. \tag{5}$$

Reprezentacijo $T(x, y)$ v običajnem prostoru dobimo z obratno Fourierovo transformacijo. Enačba (4) ima analitično rešitev

$$\hat{T}_k(t) = \hat{T}_k(0)\exp\left(-\frac{4\pi^2 k^2 D}{a^2}t\right). \tag{6}$$

To bo koristno za preverjanje numeričnih metod.

1.2 Metoda končnih elementov

Pri razvoju $T(x, y)$ nismo omejeni samo na trigonometrične funkcije. Lahko uporabimo tudi poljubne druge funkcije. V našem primeru bomo uporabili kubične B-zlepke. To so funkcije oblike

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k} - x_i} B_{i,k-1}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1,k-1}(x) \,. \tag{7}$$

Začetni pogoj bomo izrazili kot linearno kombinacijo teh funkcij in nato iščemo rešitev v obliki

$$T(x,t) = \sum_{i=-1}^{N+1} c_k(t) B_{i,3}(x) \,. \tag{8}$$

Tako zasnujemo metodo končnih elementov, s kolokacijskim pogojem. To pomeni, da se začetni pogoj mora ujemati z rešitvijo na nekem končnem številu točk. Podobno kot pri Fourierovi metodi vstavimo razvoj v PDE in dobimo sistem enačb za koeficiente $\hat{T}_i(t)$

$$\sum_{i=-1}^{N+1} D c_k(t) B_{i,3}(x) = \sum_{i=-1}^{N+1} \left(\frac{\partial c_k(t)}{\partial t} \right) B_{i,3}(x) \,. \tag{9}$$

Uporabimo lastnosti B-zlepkov in dobimo sistem diferencialnih enačb za koeficiente $c_k(t)$

$$\dot{c}_{j-1}(t) + 4\dot{c}_j(t) + \dot{c}_{j+1}(t) = \frac{6D}{h^2}(c_{j-1}(t) - 2c_j(t) + c_{j+1}(t)) \,, \tag{10}$$

kjer je h razdalja med točkami, ki smo jih izbrali za kolokacijski pogoj. Iz robnega pogoja pri $x = 0$ ugotovimo, da je $c_{-1} = -4c_0 - c_1$. Podobno iz robnega pogoja pri $x = a$ sledi $c_0 = C_N = 0$ in $c_{-1} = -c_1$ ter $c_{N-1} = -c_{N+1}$. Reševanje smo torej prevedli na reševanje matričnega sistema

$$\mathbf{A} \frac{\mathrm{d}\vec{c}}{\mathrm{d}t} = \mathbf{B}\vec{c} \,, \tag{11}$$

kjer je $\mathbf{A}$ tridiagonalna matrika z 4 na diagonali in 1 na pod in nad diagonalo

$$\mathbf{A} = \begin{bmatrix} 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \,, \tag{12}$$

$\mathbf{B}$ tridiagonalna matrika z -2 na diagonali in 1 na pod in nad diagonalo pomnožena z $6D/h^2$

$$\mathbf{B} = \frac{6D}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -2 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}, \tag{13}$$

in $\vec{c}$ vektor koeficientov $c_k(t)$. Začetni pogoj za PDE je $T(x,0) = f(x)$, torej je začetni približek za kolokacijsko aproksimacijo

$$\mathbf{A}\vec{c} = \vec{f}\,, \tag{14}$$

kjer je $\vec{f}$ vektor $f(x_i)$. To zdaj rešujemo z neko eksplitično metodo. V našem primeru bomo uporabili **Implicitno Eulerjevo metodo** zaradi njene stabilnosti. To pomeni, da za časovni korak k velja

$$\mathbf{A}\frac{\vec{c}_{k+1} - \vec{c}_k}{\Delta t} = \mathbf{B}\vec{c}_{k+1}\,. \tag{15}$$

2 Naloga

Naloga od nas zahteva da v eni razsežnosti rešimo PDE (1) z začetnim pogojem po plasti gaussovsko porazdeljene temperature

$$T(x,0) \propto \exp\left(-\frac{(x-a)^2}{2\sigma^2}\right). \tag{16}$$

Rešiti moramo za periodični robni pogoj $T(0,t) = T(a,t)$ in homogeni Dirichletov robni pogoj $T(0,t) = T(a,t) = 0$ po Fourierevi metodi. Kolokacijsko metodo nato uporabi za reševanje PDE z nehomogenim Dirichletovim robnim pogojem $T(0,t) = T(a,t) = 0$ in istim začetnim pogojem. in primerjamo obe metodi.

3 Opis reševanja

Po (pre)dolgem premoru sem se lotil reševanja naloge, spet na tradicionalen način, torej z uporabo Pythona in knjižnic `numpy` in `scipy`. Za risanje grafov sem uporabil knjižnico `matplotlib`. Zdaj imam tudi nekaj trikov v rokavu, ki sem jih uspel na hitro preizkusiti.

Po tem ko sem se lotil reševanja naloge sem ugotovil, da je bil moj prvoten pristop (opisan v datoteki `src.py`) popolnoma nepraktičen in napačen. Lotil sem se reševanja na novo in sicer z bolj sistematičnim pristopom. Za obe metodi sem napisal razred, ki vsebuje vse potrebne funkcije za reševanje. Poglejmo si to podrobneje.

3.1 SpectralSolver

Razred `SpectralSolver` je namenjen reševanju PDE po Fourierovi metodi. V konstruktorju razreda se inicializirajo vsi potrebni parametri, ki jih potrebujemo za reševanje. To so difuzijska konstanta D, debelina plasti a, število točk v prostoru N, začetni pogoj $f(x)$ in niz časov za katere računamo rešitev. Od tu naprej uporabnik pokliče eno od metod za reševanje PDE. Torej ali `solve_Analytically()` ali `solve_Numerically()`. Prva metoda uporablja analitično rešitev, ki je podana z enačbo (6). Druga metoda pa uporablja `scipy.integrate.odeint()` za reševanje sistema enačb (4).

Po opravljenem reševanju vsebuje razred tudi vse metode za izris raznih grafov. Zna risati te zvezne line plot-e, ki jih uporabljam za risanje temperaturnega profila v odvisnosti od časa. Zna risati "Heatmap"grafe in tudi animacije, ki jih žal ni smiselno dati v statično poročilo. Obstajajo tudi eksperimentalne metode za neke hexbin grafe in 3D grafe, ki jih na koncu nisem uporabil, ampak obstajajo pa, če bi jih kdo rad uporabil.

3.2 ColocationSolver

Podobno, obstaja razred `ColocationSolver`, ki je namenjen reševanju PDE po metodi končnih elementov. Tudi ta razred vsebuje vse potrebne parametre za reševanje, ki se incializira v konstruktorju. Parametri so enaki kot pri prejšnjem razredu. Razred ima tudi dve metode za reševanje PDE. Prva je `solve_Properly()`, ki uporablja `scipy.linalg.solve_banded()`, da reši sistem enačb. Druga metoda je `solve_Manually()`, ki reši sistem enačb z Thomasovim algoritmom. Ta metoda je načeloma precej hitrejša. Razred vsebuje tudi metode za risanje grafov, ki so enake kot pri prejšnjem razredu.

3.3 MPI_Node

Oče mi je dal nasvet, da naj poskusim pri zaključku 1. stopnje faksa biti pragmatičen in se ne ukvarjati z nekimi nepotrebnimi stvarmi. No jaz sem pač nekoliko glup in imam zdaj precej časovno stisko, kako naj bi uspel vse končati... ampak imam pa paralelizirane metode preko MPI-ja heh. (Pri tržni raziskavi te naloge se je izkazalo, da so tudi vsi ostali istega mnenja kot moj oče.)

Razred `MPI_Node` je namenjen paralelizaciji reševanja PDE. V konstruktorju se inicializira MPI komunikator, ki ga uporabljamo za komunikacijo med procesi. V osnovi je to samo ovoj (angl. wrapper) okoli prejšnjih dveh razredov. Vsebuje metode za delitev intervalov podatkov in za pošiljanje podatkov med procesi. Vsebuje tudi metode za risanje grafov, ki so enake kot pri prejšnjih dveh razredih, le da jih opravlja lahko samo "root"proces.

3.4 GatherStatistics

Zato da res pripeljem do konca idejo, da sem nepraktičen, sem napisal še en razred, pravzaprav še en ovoj okoli `MPI_Node`, ki je namenjen temu da zbira

statistiko o času izvajanja. Želel sem dobiti značilne premice v log-log skali, ki kažejo na uspešno paralelizacijo, ampak sem to le deloma uspel.

4 Rezultati

5 Komentarji in izboljšave

Literatura