Matematično-fizikalni Praktikum

Reševanje PDE z metodo Galerkina

Tilen Šket, 28221057

Navodila:

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.

23. januar 2025

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

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

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

$$\Delta u(\xi,\phi) = -1\,, \qquad u(\xi=1,\phi) = u(\xi,0) = u(\xi,\phi=\pi) = 0\,,$$

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

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

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

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

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

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

$$(\varepsilon,\Psi_i) = 0\,, \qquad i = 1,2,\ldots,N\,.$$

V splošnem bi lahko zahtevali tudi ortogonalnost ε na nek drug sistem utežnih oziroma testnih 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

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

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

tako da je koeficient za pretok enak

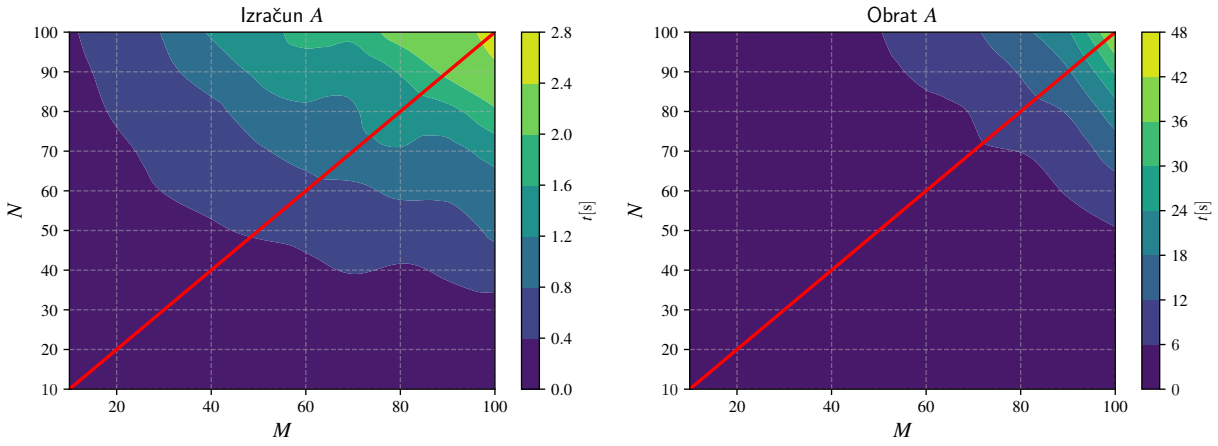
$$C = -\frac{32}{\pi}\sum_{ij} b_i A^{-1}_{ij} b_j\,.$$

Za kotni del poskusne funkcije obdržimo eksaktne funkcije $\sin((2m+1)\phi)$, Besselove funkcije za radialni del pa nadomestimo s preprostejšimi funkcijami $\xi^{2m+1}(1-\xi)^n$, kjer m teče od 0 do M in n od 1 do N.

2 Časovne zahtevnosti priprave

Za izračun C, bomo vrednosti morali vrednosti zapakirati v matrike, kar pri velikih parametrih M in N postane drago. Zato se lahko najprej lotimo pogledati, kakšna je časovna odvisnost različnih rutin priprave matrik. Pogledali si bomo izračun same matrike in obrat matrike ter izračun matrike uporabne za `SciPy` `solve_banded` rutino, ki se nekoliko razlikuje od navadnega.

Ker imamo dva parametra M in N, ki ju lahko spreminjamo, moramo naše podatke prikazati v višji dimenziji. To lahko storimo tako, da jih prikažemo kot polje, kar je narejeno na sliki 1. Iz slike opazimo, da je obrat



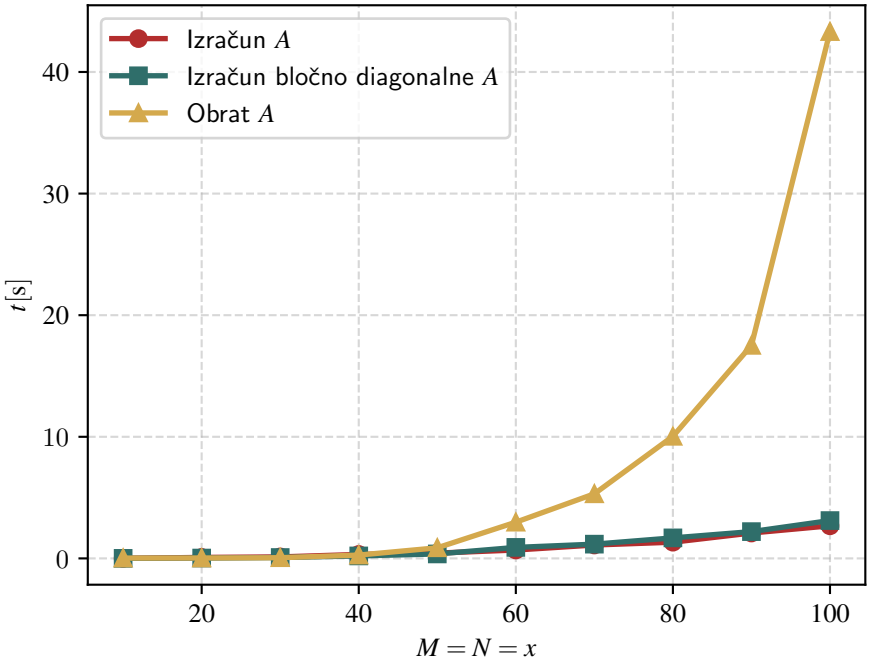
Slika 1: Časovna zahtevnost priprave oziroma obrata matrike za različne vrednosti parametrov M in N. Pazi, skali na slikah nista enaki.

časovno veliko bolj potrošen, še toliko bolj pa če upoštevamo dejstvo, da moramo matriko A tako ali tako prej izračunati, da lahko izračunamo njen obrat. Uporabljena rutina je `NumPy`-eva `linalg.inv`. Izračun pa je narejen s pomočjo zank in analitičnega izračuna vrednosti matrik na posameznem mestu, ki izvira iz navodil naloge. Sami podatki v obliki polja morda izgledajo lepo, vendar nam ne dajo pravega občutka za primerjavo med časovno zahtevnostjo teh dveh procesov, zato se lahko lotimo postopka redukcije dimenzije, da vse skupaj prikažemo na tradicionalnem grafu. To lahko storimo po že označeni rdeči črti, ki poteka po diagonali na sliki 1, kjer velja $N = M$. Tako dobimo graf na sliki 2. Na tej sliki, je poleg izračuna A in obrata A izrisana še zahtevnost izračuna bločno diagonalne A, ki se od navadne nekoliko razlikuje, vendar iz stališča trenutne obravnave ne. Iz teh dveh grafov zaključimo, da je direkten izračun matrike A časovno najbolj uporaben.

3 Časovne zahtevnosti metod

Čeprav je v osnovi naloga enaka pri vseh metoda, se lahko reševanje le te lotimo na veliko različnih načinov. Pri reševanju lahko uporabimo več rutin iz različnih knjižnic ali pa tudi le te nekoliko obrnemo na glavo, na primer s tem da matriko obrnemo vnaprej.

Izberimo si recimo 5 metod, in sicer od tega 4, ki neposredno rešijo sistem $A\vec{a} = \vec{b}$ in kot zadnjo obrnemo



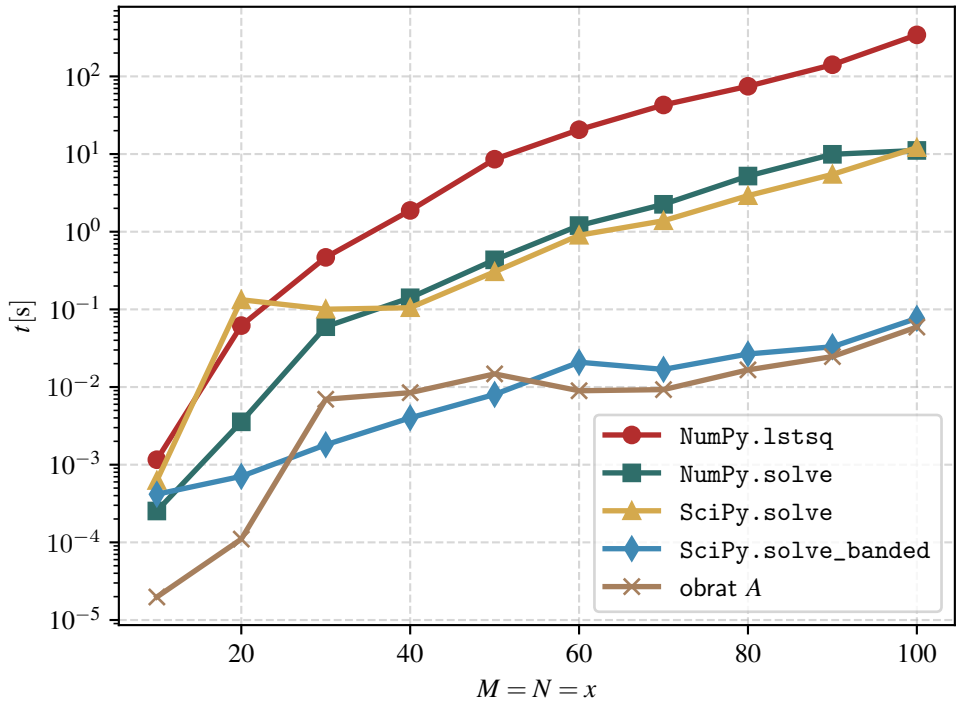
Slika 2: Dimenzijsko reducirana predstava časovnih zahtevnosti, kjer se bolje vidi primerjalna razlika časovnih zahtevnosti metod.

A in izračunamo $\vec{a} = A^{-1}\vec{b}$. Predstave rezultatov se lahko ponovno lotimo s poljem vrednosti parametrov M in N. Ali pa se iz prejšnjega poglavja izučimo, da je bolje uporabiti redukcijo dimenzije, saj tako lažje neposredno primerjamo časovne zahtevnosti uporabljenih metod. Uporabimo enak pogoj kot prej, in sicer $M = N = x$, in s tem dobimo graf na sliki 3. Tokrat je bila tudi smiselna uporaba logaritemske skale, saj imamo veliko krivulj, ki se razprostirajo čez precej velik razpon velikostnih redov. Kot najhitrejša metoda pa se izkaže za tisto, kjer opravimo obrat in nato le skalarno množimo vektor z matriko. Ta rezultat ni presenetljiv, saj ne upošteva dodatnega časa priprave matrike. Z upoštevanjem tega opisana metoda postane druga najpočasnejša, in sicer le hitrejša od `NumPy.lstsq`. Tudi to ni presenetljivo, saj deluje le na principu optimizacije in torej ne uporablja vseh dodatnih lastnosti danega problema, ki bi jo pohitrile. Kar pa je ravno obratno od efektivno najhitrejše metode, ki je `SciPy.solve_banded`. Pri le tej je priprav matrike podobno časovno zahtevna kot navadna matrika A, sama metoda pa se kosa tudi z direktnim skalarnim množenjem matrik. To lahko doseže, ker uporablja dejstvo, da je A bločno diagonalna.

Pri vsej tej obravnavi časovnih zahtevnosti pa ne smemo pozabiti na najpomembnejši del, in sicer natančnost rezultata.

4 Primerjava rezultatov

Poglejmo si vrednosti, ki jih dobimo za C z uporabo različnih metod. Najprej se lahko vprašamo, s čim bi rezultate primerjali, da lahko določimo njihovo natančnost in točnost. Ker analitičnega izračuna v tem



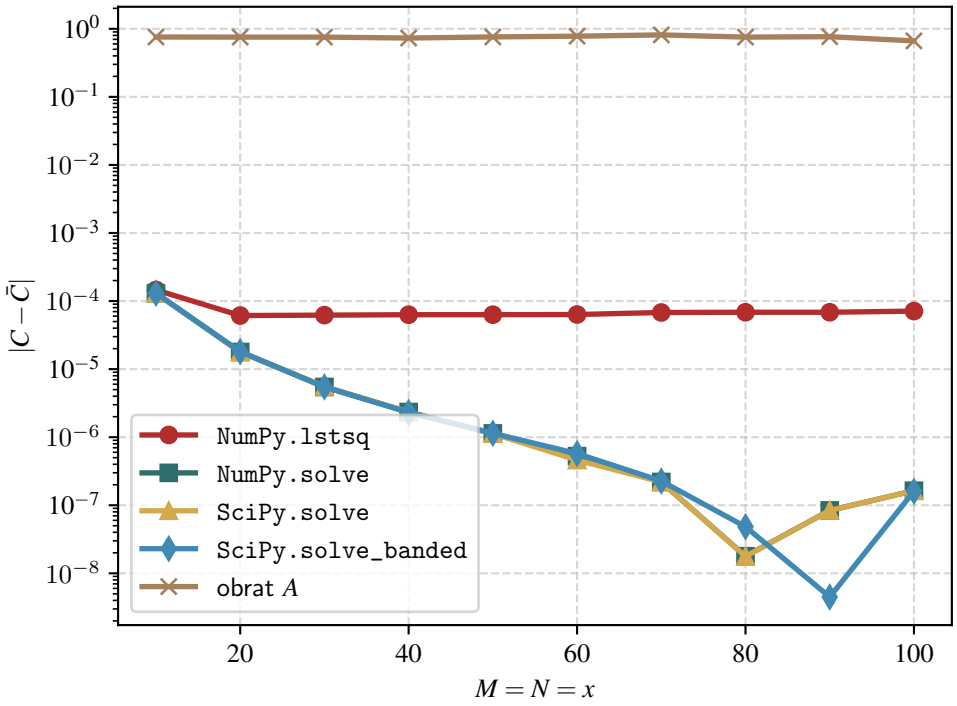
Slika 3: Graf časovnih zahtevnosti različnih metod uporabljenih za reševanje sistema.

poročilu ni, se osredotočimo le na natančnost, torej koliko so rezultati posameznih metod med seboj oddaljeni. Za referenčno točko $\bar{C}$ torej vzemimo kar mediano nekaj izračunov, katerih parametra M in N sta med večjimi, saj slutimo, da bodo takrat tudi rezultati najboljši. Razlog za izbor mediane je, da morebitne močno odstopajoče vrednosti nimajo prevelikega pomena. Ponovno uporabimo prikaz s pogojem $M = N = x$ in si oglejmo absolutno oddaljenost od referenčne točke $|C - \bar{C}|$ posamezne metode pri različnih vrednostih x. To je prikazano na sliki 4. Na grafu najbolj izstopa velika napaka metode skalarnega množenja z obratom matrike A. Iz tega rezultata rezultata časovne zahtevnosti, kjer metoda prav tako ni blestela, lahko zaključimo, da za dani problem ni uporabna. Potem imamo `NumPy.lstsq`, katere odstopanje ni preveliko, vendar zaradi zelo slabe časovne zahtevnosti lahko sklenemo, da tudi ta metoda ni uporabna. Preostale so nam tri metode, ki pa se skoraj popolnoma skladajo pri izračunanem rezultatu, kar je pri `NumPy` in analognih `SciPy` rutinah pogosto res. Odločitev za najbolj uporabno metodo izmed teh treh, določi časovna zahtevnost, ki daje prednost `SciPy.solve_banded`, kar je pričakovano, saj predpostavi kar največ informacij o matriki in je s tem tudi najbolj optimizirana za dani problem.

5 Izračunana vrednost

Ostalo je le še vprašanje, kolikšna je vrednost C. Le to po podani analizi izračunamo s pomočjo rutine `SciPy.solve_banded` pri parametrih $M = N = 181$ in dobimo

$$C = 0{,}757\,722\,068\,9(2)\,.$$



Slika 4: Prikaz oddaljenosti posameznih metod od referenčne točke, ki je določena z mediano nekaj vrednosti z velikima parametroma M in N.

Napako ocenimo tako, da pogledamo velikost naslednjega popravka. Sam končen izračun matrike in C, pa je trajal 29,3 s.

6 Zaključek

V tem poročilu lahko bralec opazi, da je kar nekaj sprememb pri samem izgledu pisave. Izbral novo pisavo, ki jo bom verjetno nekaj časa uporabljal za namene pisanja seminarskih nalog in poročil. Idejo za menjavo pisave sem dobil iz navodil za LaTeX datoteke naložene na stran *arXiv*. Prav tako sem končno poskrbel, da se velikosti pisav v poročilu in na grafih skladajo. Sem pa v tem poročilu tudi poskrbel za to, da skozi celotnega teče rdeča nit, kar menim, da je bil en večjih problemov mojih nalog do sedaj.

Ta naloga mi je v celoti vzela 9 h, od katerih sem jih veliko porabil za izboljšavo izgleda samega dokumenta. Sem pa tudi za dobro uro poleg tega poganjal vse možne kombinacije reševanja, iz katerih sem dobil časovne zahtevnosti. To poročilo na žalost ponovno delam sredi noči na četrtek, vendar tokrat z manj slabe vesti, saj jutri ni predavanj, saj smo že močno zajadrali v izpitno obdobje.