

11. naloga: Reševanje PDE z metodo Galerkina

Samo Krejan, 28231092

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

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

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

$$\begin{aligned} \Delta u(\xi, \varphi) &= -1, & u(\xi = 1, \varphi) &= u(\xi, 0) = u(\xi, \varphi = \pi) = 0, \\ C &= 8\pi \iint \frac{u(\xi, \varphi) \xi \, d\xi \, d\varphi}{(\pi/2)^2}. \end{aligned}$$

Če poznamo lastne funkcije diferencialnega operatorja za določeno geometrijo¹ 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 nekaj poskusnih (*trial*) funkcij

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

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

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

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

$$(\varepsilon, \Psi_i) = 0, \quad i = 1, 2, \dots, N.$$

V splošnem bi lahko zahtevali tudi ortogonalnost ε na nek drug sistem utežnih (*weight*) oziroma testnih (*test*) 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

$$\sum_{j=1}^N A_{ij} a_j = b_i, \quad i = 1, 2, \dots, N, \quad (2)$$

¹Spomni se na primer na vodikov atom v sferični geometriji, kjer smo imeli $\hat{L}^2 Y_{lm}(\vartheta, \varphi) = \hbar^2 l(l+1) Y_{lm}(\vartheta, \varphi)$ in $\hat{L}_z Y_{lm}(\vartheta, \varphi) = m\hbar Y_{lm}(\vartheta, \varphi)$.

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

tako da je koeficient za pretok enak

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

Za kotni del poskusne funkcije obdržimo eksaktne funkcije $\sin((2m+1)\varphi)$, Besselove funkcije za radialni del pa nadomestimo s preprostejšimi funkcijami $\xi^{2m+1}(1-\xi)^n$. Pozor: indeks i pomeni seveda dvojni indeks (šteje obenem m in n)². Zaradi ortogonalnosti po m razpade matrika A v bloke, obrneš pa jo lahko s kako pripravljeno rutino, npr. s spodnjim in zgornjim trikotnim razcepom `ludcmp` in `lubksb` iz NRC.

1 Naloga

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.

1.1 Računanje C

Ravno pred delom naloge, sem si uredil popolnoma nov sistem, tako da sem se odločil, da je dana naloga odlična za preverjanje komputacijskih limit. Kot bo bralec kmalu videl, je bila naloga iz tega vidika res idealna.

1.1.1 Osnovna metoda

Začel sem z najosnovnejšo implementacijo, ki jo je bilo tudi najlažje implementirati. Glaven korak metode ustvari matriko A po receptu, ki je bil izpeljan na predavnjih. Nato z namenom, da bi se znebili potrebe po inverzih matrike, rešimo sistem enačb:

$$A\vec{a} = \vec{b}$$

in nazadnje le še izračunamo C kot:

$$C = -\frac{32}{\pi} \vec{b} \cdot \vec{a}$$

Dano metodo sem optimistično otvoril z $N = 100$ in $M = 100$ in presenečen ugotovil, da je "šlo skozi" (v pičlih desetih sekundah) in mi dalo rezultat:

$$C = 0.7577218680$$

Ves navdušen sem se nato lotil iskanja mej metode, a sem hitro ostal razočaran, saj sem res hitro zadel zid zaradi spominske potratnosti metode. Ker metoda res ustvari celotno matriko A , le ta porabi precej prostora, kar sem najprej spoznal na lep način, ko me je Python še opozoril, da ne mora dodeliti objektu 2TiB prostora. Kasneje pa, ko sem se približal pametnim mejam so se opozorila umaknila, problemi pa so ostali in tako mi je računalnik trikrat popolnoma zmrznil in postal popolnoma neodziven. Takrat sem vedel, da če želim z delom nadaljevati, moram metodo najprej optimizirati.

²Glej tudi prilogo na spletni učilnici.

1.1.2 Bločna metoda

Prva optimizacija je sledila opazki, da je matrika A res tako zelo bločna. To seveda pomeni, da lahko tudi enačbo:

$$A\vec{a} = \vec{b}$$

rešujemo bločno in posledično po delih. Ta metoda se je popolnoma znebila problemov z dodeljavanjem spomina, saj je bilo potrebno konstruirati le vsak posamezen blok posebaj. Tako je računalnik ostal uporaben tudi med računanjem.

Je pa prišel do izraza drugačen problem - ko večamo M , postajajo posamezni bloki matrike A vedno bolj singularni. Paket Scipy je hitro začel s pošiljanjem opozoril in ko sem ga prignal še nekoliko dlje, je metoda popolnoma odpovedala.

ChatGPT mi je na tej točki svetoval naj rešim problem s *Tikhonovo regulacijo*³. Z le to se rešimo singularnosti tako, da našim blokom matrike A dodamo identitetno matriko pomnoženo s čim manjšim faktorjem, ki še deluje. Čeprav je metoda tako postala stabilna in predvsem *vedno delujoča*, je sproducirala rezultat z relativno veliko napako (0.1% pri $N = M = 500$) - izkaže se, da namerno "kvariti" matriko A ni najboljši način.

1.1.3 Spektralna dekonstrukcija

Ideja te podmetode sloni na dejstvu, da ko rešujemo:

$$A\vec{a} = \vec{b}$$

nas rešitev za a pravzaprav ne zanima in izkaže se, da lahko njen izračun preskočimo. Opremo se namreč na dejstvo, da so bloki matrike A simetrični, ter imajo zato unitarno dekompozicijo:

$$A^{(m)} = U\Lambda U^T$$

,
kjer je U sestavljena iz lastnik vektorjev u_k . Iz tega sledi:

$$b^T A^{(m)} b = b^T U \Lambda^{-1} U^T b = \sum_k \frac{(u_k^T b)^2}{\lambda_k}$$

Tega sedaj ni težko implementirati, edina težava, ki se pojavi je možnost $\lambda_k = 0$. V teoriji nas to ne skrbi, saj je takrat limita $\frac{(u_k^T b)^2}{\lambda_k} = 0$, v praksi, pa le ignoriramo res majhne λ_k , saj končni vsoti prinesejo zelo malo.

Z dano metodo je izračun za $M = N = 100$ vzel zgolj 0.2 sekunde in nam dal praktično identičen rezultat kot osnovna metoda, tako da sem precej prepričan v veljavnost metode.

Po uspehu z implementacijo sem sicer postal zelo ambiciozen in se lotil računanja primera z $M = N = 3000$, ki je vzel dobre tri ure in podal vrednost:

$$C = 0.7577221531,$$

ki jo od te točke naprej tretiram kot "eksaktna vrednost".

1.2 Uspešnost metode

³https://en.wikipedia.org/wiki/Ridge_regression