Matematično-fizikalni seminar
Priročnik za samostojno učenje

Teodor Jovanovski

12. junij 2025

Kazalo

1 Natančnost števil in Gaussov integral

Okvir 1

Pri mnogih numeričnih algoritmih je ključno razlikovanje med **točnostjo** (accuracy) in **natančnostjo** (precision). Točnost se nanaša na to, kako blizu je meritev ali izračun pravi vrednosti. Natančnost pa se nanaša na to, kako blizu so si ponovljene meritve ali izračuni med seboj, ne glede na njihovo točnost.

Predstavljajte si, da streljate v tarčo. Kateri od naslednjih opisov najbolje opiše rezultat, ki je **natančen, a ne točen**?

[a] Vsi streli so zbrani tesno skupaj v sredini tarče. [b] Vsi streli so naključno razpršeni po vsej tarči. [c] Vsi streli so zbrani tesno skupaj, vendar v zgornjem levem kotu, daleč od sredine tarče.

Izberite odgovor in pojdite na Okvir 2.

Okvir 2

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [c]. Tesno zbrani streli kažejo na visoko natančnost, a ker je skupina strelov daleč od sredine tarče (prave vrednosti), rezultat ni točen.

a Visoka točnost in visoka natančnost.

b Nizka točnost in nizka natančnost.

Razumevanje te razlike je ključno pri programiranju, saj lahko način, kako računalniki shranjujejo števila, vpelje napake, ki vplivajo na točnost naših rezultatov, četudi se izračuni zdijo natančni.

Pojdite na Okvir 3.

Okvir 3

Začnimo s tem, kako računalniki shranjujejo preprosta števila. Predstavitev naravnih števil (pozitivnih celih števil) v dvojiškem sistemu vam je verjetno znana. Vsaka pozicija predstavlja potenco števila 2.

Na primer, dvojiško število 11001101_2 se pretvori v desetiško število takole:

$$(1 \cdot 2^7) + (1 \cdot 2^6) + (0 \cdot 2^5) + (0 \cdot 2^4) + (1 \cdot 2^3) + (1 \cdot 2^2) + (0 \cdot 2^1) + (1 \cdot 2^0)$$
$$= 128 + 64 + 0 + 0 + 8 + 4 + 0 + 1 = 205_{10}$$

To je preprosto. Shranjevanje celih števil je prav tako razmeroma enostavno, običajno z uporabo najpomembnejšega bita (most significant bit) kot predznaka (0 za pozitivno, 1 za negativno).

Kaj pa števila z decimalno vejico, kot je 0,1? Tu se začne pravi izziv.

Pojdite na Okvir 4.

Okvir 4

Osnovni problem je, da je dvojiški sistem osnovan na bazi 2. Popolnoma lahko predstavi vsak ulomek, katerega imenovalec je potenca števila 2. Na primer:

- $0,5 = 1/2 = 1 \cdot 2^{-1} = 0,1_2$
- $0,25 = 1/4 = 1 \cdot 2^{-2} = 0,01_2$

- $0,75 = 3/4 = 1/2 + 1/4 = 0,11_2$

Vendar pa mnogi pogosti desetiški ulomki, kot je $0,1 = 1/10$, nimajo imenovalca, ki bi bil potenca števila 2. To pomeni, da jih ni mogoče predstaviti s končnim številom dvojiških mest. V dvojiškem zapisu postanejo periodični ulomki, tako kot je 1/3 periodičen ulomek (0,333...) v desetiškem sistemu.

Za obravnavo tega in za predstavitev ogromnega obsega števil (od zelo majhnih do zelo velikih) računalniki uporabljajo standardiziran sistem za števila s "plavajočo vejico"(floating-point). Najpogostejši standard je **IEEE 754**.

Pojdite na Okvir 5.

Okvir 5

Poglejmo si standard IEEE 754 za 32-bitno število s plavajočo vejico enojne natančnosti, pogosto imenovano 'float'. 32 bitov je razdeljenih na tri dele:

- **Predznak (S):** 1 bit. 0 za pozitivno, 1 za negativno.
- **Eksponent (E):** 8 bitov. Hrani potenco števila 2.
- **Ulomek (F) ali Mantisa:** 23 bitov. Hrani pomembne števke števila.

```
 S | EEEEEEEE | FFFFFFFFFFFFFFFFFFFFFFF
[1]|[  8 bitov ]|[        23 bitov       ]
```

Kako so ti trije deli združeni, da predstavljajo število?

Pojdite na Okvir 6, da vidite formulo.

Okvir 6

Za normalna števila je vrednost V podana z formulo:

$$V = (-1)^S \times 2^{(E-127)} \times (1.F)_2$$

Poglejmo si to podrobneje:

- $(-1)^S$: To samo določi predznak.
- $2^{(E-127)}$: Eksponent je shranjen s **pomerkom** (bias) 127. Da dobimo pravi eksponent, moramo od vrednosti, shranjene v bitih E, odšteti 127. To omogoča, da eksponent predstavlja tako pozitivne kot negativne potence.
- $(1.F)_2$: To je mantisa. '1.' je *implicitni vodilni bit*. Ni shranjen, vendar se predpostavlja, da obstaja za vsa normalna števila, kar nam učinkovito daje 24 bitov natančnosti za ceno 23. F predstavlja ulomljeni del v dvojiškem zapisu.

Če v prosojnicah vidite formulo '(1 + B)', je 'B' vrednost mantise, interpretirana kot dvojiški ulomek. Na primer, če so biti mantise 'F' enaki '1010...', je celoten izraz $(1,101)_2 = 1 + 1/2 + 0/4 + 1/8$.

Pojdite na Okvir 7, da vidite, kaj se zgodi v posebnih primerih.

Okvir 7

Standard IEEE 754 rezervira določene vrednosti eksponenta za posebna števila.

- Če je Eksponent = 255 (vse enice) in Ulomek = 0, število predstavlja ±**Neskončnost**.
- Če je Eksponent = 255 (vse enice) in Ulomek ≠ 0, je vrednost **NaN** (Not a Number - Ni število), ki se uporablja za rezultate neveljavnih operacij, kot je 0/0.
- Če je Eksponent = 0 in Ulomek = 0, je število **0**.
- Če je Eksponent = 0 in Ulomek ≠ 0, so števila **sub-normalna** (ali denormalizirana). To so posebna števila, ki nimajo implicitnega vodilnega 1, kar omogoča predstavitev vrednosti, ki so bližje ničli, kot bi to dopuščala normalna števila.

Ta skrbna zasnova omogoča robusten sistem za obravnavo mnogih računskih situacij.

Pojdite na Okvir 8.

Okvir 8

Zaradi te končne predstavitve obstajajo "vrzeli"med predstavljivimi števili. Velikost te vrzeli se spreminja z velikostjo števila.

Funkcije, kot sta 'std::nextafter' v C++ ali 'numpy.nextafter' v Pythonu, omogočajo iskanje naslednjega predstavljivega števila s plavajočo vejico za danim številom.

Razmislite o številih 1,0 in 1000,0. Ali je vrzel (razlika) med 1,0 in naslednjim predstavljivim številom 'float' *manjša, večja ali enaka* kot vrzel za 1000,0?

[a] Manjša kot [b] Večja kot [c] Enaka kot

Izberite odgovor in pojdite na Okvir 9.

Okvir 9

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [a] Manjša kot. Vrzel med predstavljivimi števili raste z velikostjo števil. To je zato, ker eksponentni del števila 'float' učinkovito skalira velikost koraka, ki jo določa zadnji bit mantise. Za 1,0 je eksponent majhen, zato so koraki majhni. Za 1000,0 je eksponent večji, zato so koraki večji.

To vodi do nekaj pomembnih lekcij za programiranje.

Pojdite na Okvir 10.

Okvir 10

Tukaj so ključni nauki za vsakega programerja, še posebej v fiziki in tehniki:

1. **Natančnost je končna:** 'float' (enojna natančnost) ima približno 7 decimalnih mest natančnosti. 'double' (64-bitna) ima približno 16. Izberite tisto, ki je primerna za vaš problem.
2. **Nikoli ne preverjajte enakosti števil s plavajočo vejico.** Zaradi napak pri predstavitvi (kot pri 0,1) lahko izračun, za katerega pričakujete, da bo 'a == b', ne uspe.

Namesto 'if (a == b)', kateri je veliko varnejši način za primerjavo dveh števil s plavajočo vejico 'a' in 'b'? _____

Pojdite na Okvir 11 za odgovor.

Okvir 11

Varnejši način za primerjavo števil 'a' in 'b' je preverjanje, ali je njuna razlika manjša od neke majhne tolerance, imenovane epsilon (ϵ).

```
if (abs(a - b) < epsilon) {
    // Obravnavaj ju kot enaka
}
```

kjer je ‘epsilon‘ lahko majhno število, kot je ‘1e-7‘.

Zdaj, ko se zavedamo teh težav z natančnostjo, se lotimo glavne naloge iz predstavitve: programiranja Gaussovega integrala.

Pojdite na Okvir 12.

Okvir 12

Naloga: Gaussov integral (funkcija napake)

Cilj je natančno sprogramirati funkcijo napake, ‘erf(z)‘, ki je definirana kot:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

Ta funkcija je monotono naraščajoča, je 0 pri $z = 0$ in se približuje 1, ko gre $z \to \infty$.

Za ta integral ne obstaja preprosta, zaprta rešitev. Moramo ga aproksimirati. Raziskali bomo dve pogosti metodi.

Pojdite na Okvir 13.

Okvir 13

Metoda 1: Razvoj v Taylorjevo vrsto

Dobro znan pristop je razvoj eksponentnega člena, e^{-t^2}, v Taylorjevo vrsto in nato integracija po členih. To da:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n z^{2n+1}}{n!(2n+1)}$$

To je potenčna vrsta v z. Glede na delovanje potenčnih vrst, za kateri obseg vrednosti z pričakujete, da bo ta aproksimacija najbolj uporabna in učinkovita?

[a] Zelo velike vrednosti z. [b] Vrednosti z blizu 0 (majhne vrednosti). [c] Deluje enako dobro za vse vrednosti z.

Izberite odgovor in pojdite na Okvir 14.

Okvir 14

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b]. Taylorjeve vrste so razvoji okoli točke (v tem primeru $z = 0$) in so najnatančnejše blizu te točke. Za majhne z se členi zelo hitro manjšajo in potrebujete le nekaj členov za dobro aproksimacijo. Za velike z bi potrebovali ogromno število členov, kar je zelo neučinkovito.

Torej, kako obravnavamo velike z?

Pojdite na Okvir 15.

Okvir 15

Metoda 2: Asimptotična vrsta

Za velike vrednosti z je veliko boljša drugačna vrsta, imenovana asimptotična vrsta. Pogosto jo uporabljamo za komplementarno funkcijo napake, $\mathrm{erfc}(z) = 1 - \mathrm{erf}(z)$. S postopkom ponavljajoče se integracije po delih (per partes) lahko pridemo do naslednje aproksimacije:

$$\mathrm{erfc}(z) \approx \frac{e^{-z^2}}{z\sqrt{\pi}} \left[1 - \frac{1}{2z^2} + \frac{1 \cdot 3}{4z^4} - \frac{1 \cdot 3 \cdot 5}{8z^6} + \dots\right]$$

Asimptotične vrste imajo nenavadno lastnost: ne konvergirajo. Po določeni točki začnejo členi spet naraščati. Trik je v tem, da seštevamo člene samo do najmanjšega člena za dani z. Večji kot je z, več členov lahko uporabimo, preden začnejo naraščati.

Ta metoda je odlična aproksimacija samo za **velike vrednosti z**.

Pojdite na Okvir 16.

Okvir 16

Metoda 3: Racionalne aproksimacije (Padé)

Imate eno metodo za majhne z (Taylorjeva) in drugo za velike z (asimptotična). Kako lahko ustvarite eno samo funkcijo, ki dobro deluje za vse vrednosti? En odgovor je, da jih šestavite": uporabite Taylorjevo vrsto, če je z majhen, in asimptotično vrsto, če je z velik.

Naprednejša tehnika je uporaba **racionalne aproksimacije**, kot je Padéjev aproksimant. Ta uporablja razmerje dveh polinomov za aproksimacijo funkcije. Predstavitev podaja znano racionalno aproksimacijo:

$$\mathrm{erf}(z) = 1 - (a_1 t + a_2 t^2 + \dots + a_5 t^5) e^{-z^2} + \epsilon(z) \quad \text{kjer je} \quad t = \frac{1}{1 + pz}$$

Koeficienti $p, a_1, \dots, a_5$ so skrbno izbrani konstanti, da se minimizira napaka $\epsilon(z)$ v širokem območju.

Pojdite na Okvir 17.

Okvir 17

Zaključek in vaša naloga

Cilj prvotne seminarske naloge je bil, da te ideje uporabite v praksi. Izziv je:

1. Implementirati aproksimacijo za ‘erf(z)‘ z uporabo Taylorjeve vrste.
2. Implementirati aproksimacijo za ‘erfc(z)‘ z uporabo asimptotične vrste.
3. Primerjati rezultate z zanesljivo, vgrajeno knjižnično funkcijo za ‘erf(z)‘ (kot je ‘math.erf‘ v Pythonu).
4. Narisati absolutno napako, $\log|f_{\text{približek}} - f_{\text{prava}}|$, da vidite, kje je katera metoda najnatančnejša.
5. Razmisliti, kako bi lahko združili te metode ali racionalno aproksimacijo, da bi ustvarili eno samo, robustno funkcijo, ki je natančna za vse vrednosti z.

Ta praktična vaja poudarja resnične kompromise med različnimi numeričnimi metodami in temeljne omejitve, ki jih nalaga aritmetika s plavajočo vejico.

Konec poglavja.

2 Iskanje ničel in preprosta numerična integracija v 1D

Okvir 1

Ko izvajamo numerične naloge, kot je iskanje ničel funkcij (kjer je funkcija enaka nič) ali njihova integracija, je ključno razumeti **velikostni red** problema.

Z drugimi besedami, vprašati se moramo: za naš specifičen problem, kaj je "majhno"in kaj je "veliko"?

- V astronomskem problemu je časovni interval 1 dan lahko infinitezimalno majhen.
- V fiziki osnovnih delcev je 1 pikosekunda (10^{-12} s) lahko relativno dolg čas.

Uporabna tehnika je, da enačbe zapišemo v **brezdimenzijski obliki**, na primer z definiranjem nove spremenljivke $z = t/\tau$, kjer je τ značilna časovna skala sistema. V takem sistemu je vrednost $z = 1$ naravna, objektivna skala.

V tem poglavju si bomo ogledali nekatere najpreprostejše algoritme, da bi razumeli njihovo osnovno logiko. Mnoge profesionalne programske knjižnice vsebujejo veliko boljše algoritme, vendar so pogosto preproste metode dovolj dobre.

Pojdite na Okvir 2.

Iskanje ničel funkcij v 1D

Okvir 2

Naš cilj je najti vse možne rešitve (ničle) enačbe $f(x) = 0$.

Začnimo s trdo resnico: ne obstaja univerzalen numerični algoritem, ki bi našel *vse* ničle poljubne funkcije. Vendar pa obstajajo odlični algoritmi, ki lahko najdejo ničlo, *če že vemo, da obstaja znotraj določenega intervala* $[a, b]$.

Kateri pogoj mora veljati za funkcijo $f(x)$ na intervalu $[a, b]$, da zagotovimo, da se vsaj ena ničla nahaja znotraj njega? (Namig: pomislite na predznak funkcije na krajiščih.) _____

Pojdite na Okvir 3 za odgovor.

Okvir 3

Da zagotovimo vsaj eno ničlo na intervalu $[a, b]$, mora imeti funkcija na krajiščih nasprotna predznaka. To pomeni:

$$f(a) \cdot f(b) < 0$$

Če je $f(a)$ pozitivna in $f(b)$ negativna (ali obratno) in predpostavimo, da je funkcija zvezna, mora nekje med a in b prečkati os x.

Torej je prvi korak pri iskanju ničel pogosto iskanje takega intervala. Najbolj robusten način za to je preiskovanje območja zanimanja z mrežo točk, $\{x_i\}$, ločenih z majhnim korakom h, kjer je $x_{i+1} = x_i + h$. Nato iščemo poljuben podinterval $[x_i, x_{i+1}]$, kjer se predznak funkcije spremeni, tj. kjer je $f(x_i) \cdot f(x_{i+1}) < 0$.

Ko najdemo tak interval, kako se "približamo"ničli?

Pojdite na Okvir 4.

Okvir 4

Metoda bisekcije

Najpreprostejša in najbolj robustna metoda je **metoda bisekcije**. Deluje z večkratnim deljenjem intervala na polovico.

1. Izberite začetni interval $[a_1, b_1]$, kjer je $f(a_1) \cdot f(b_1) < 0$.

2. Izračunajte središče intervala: $x = (a_1 + b_1)/2$.

3. Preverite predznak funkcije na sredini.

Če je $f(a_1) \cdot f(x) < 0$, katera točka postane novo krajišče za naslednji, manjši interval? [a] $a_2 = a_1$ [b] $b_2 = b_1$ [c] $b_2 = x$ [d] $a_2 = x$

Izberite odgovor in pojdite na Okvir 5.

Okvir 5

Vaš odgovor je bil [a — b — c — d].

Pravilen odgovor je [c]. Če je $f(a_1) \cdot f(x) < 0$, to pomeni, da ničla leži v prvi polovici intervala, $[a_1, x]$. Zato nastavimo naš nov interval na $[a_2, b_2] = [a_1, x]$.

Obratno, če je $f(a_1) \cdot f(x) > 0$, to pomeni, da je ničla v drugi polovici, in naš nov interval postane $[a_2, b_2] = [x, b_1]$.

Te korake (2 in 3) ponavljamo, dokler velikost našega intervala, $|b_n - a_n|$, ni manjša od želene natančnosti, ϵ. Ker se velikost intervala pri vsakem koraku prepolovi, ta metoda zagotovo konvergira, čeprav je lahko počasna.

Pojdite na Okvir 6.

Okvir 6

Metoda regula falsi (metoda napačne predpostavke)

Metoda bisekcije je robustna, vendar ne uporablja nobenih informacij o vrednostih funkcije, le njihove predznake. Metoda **regula falsi** poskuša biti pametnejša.

Osnovna ideja je, da potegnemo premico (sekanto) med točkama $(a_1, f(a_1))$ in $(b_1, f(b_1))$. Naša naslednja ocena za ničlo, x, je točka, kjer ta premica seka os x. Formula za to presečišče je:

$$x = \frac{f(b_1)a_1 - f(a_1)b_1}{f(b_1) - f(a_1)}$$

Po najdbi x je pravilo za posodobitev enako kot pri bisekciji: preverimo predznake, da določimo nov, manjši interval.

Potencialna težava te metode je, da če je funkcija zelo ukrivljena, lahko eno od krajišč ostane žataknjeno" več iteracij, kar upočasni konvergenco.

Pojdite na Okvir 7.

Okvir 7

Sekantna metoda

Sekantna metoda je zelo podobna metodi regula falsi, vendar s ključno razliko. Prav tako uporablja premico med dvema točkama za iskanje naslednje ocene, vendar ne zahteva, da ti dve točki objemata ničlo (tj. ne potrebujemo, da je $f(x_n) \cdot f(x_{n-1}) < 0$).

1. Začnemo z dvema začetnima ocenama, x_0 in x_1.

2. Uporabimo enako formulo linearne interpolacije kot pri metodi regula falsi, da najdemo naslednjo točko, x_2:
$$x_{n+1} = \frac{f(x_n)x_{n-1} - f(x_{n-1})x_n}{f(x_n) - f(x_{n-1})}$$

3. Za naslednjo iteracijo zavržemo najstarejšo točko in uporabimo x_1 in x_2 za iskanje x_3, in tako naprej.

Ta metoda je pogosto hitrejša od metode regula falsi, a ker ne ohranja ničle znotraj intervala, ni zagotovljeno, da bo konvergirala. Če so začetne ocene slabe ali če je funkcija neprimerna, lahko metoda ne uspe.

Pojdite na Okvir 8.

Okvir 8

Newton-Raphsonova metoda

Kaj pa, če poleg vrednosti funkcije $f(x)$ poznamo tudi njen odvod, $f'(x)$? **Newton-Raphsonova** (ali Newtonova) metoda uporablja to dodatno informacijo za zelo hitro konvergenco.

Namesto da bi risali sekanto med dvema točkama, rišemo **tangento** na krivuljo pri naši trenutni oceni, x_n. Naslednja ocena, x_{n+1}, je točka, kjer ta tangenta seka os x.

Glede na to, da je enačba tangente v točki $(x_n, f(x_n))$ enaka $t(x) = f(x_n) + f'(x_n)(x - x_n)$, kakšna je formula za x_{n+1}? (Nastavite $t(x_{n+1}) = 0$ in rešite za x_{n+1}). _____

Pojdite na Okvir 9 za odgovor.

Okvir 9

Z nastavitvijo enačbe tangente na nič dobimo:

$$0 = f(x_n) + f'(x_n)(x_{n+1} - x_n)$$

Z reševanjem za x_{n+1} dobimo iteracijsko formulo Newton-Raphsonove metode:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ta metoda je lahko izjemno hitra (pogosto konvergira kvadratično). Vendar pa, tako kot sekantna metoda, lahko ne uspe konvergirati, če je začetna ocena slaba ali če je odvod blizu ničle v bližini ničle. Mnogi robustni, profesionalni algoritmi (kot je 'scipy.optimize.brentq') združujejo varnost metode bisekcije s hitrostjo teh hitrejših metod.

Pojdite na Okvir 10.

Preprosta numerična integracija v 1D

Okvir 10

Formalno lahko določeni integral funkcije v 1D, $\int_a^b f(x)dx$, interpretiramo kot ploščino pod krivuljo $y = f(x)$ na intervalu $[a, b]$. Metode za numerično računanje tega integrala se pogosto imenujejo **kvadraturna pravila**.

Osnovna ideja je, da interval $[a, b]$ razdelimo na N ozkih trakov enake širine $h = (b - a)/N$. Nato aproksimiramo ploščino vsakega traku in jih seštejemo.

Najpreprostejša metoda je **pravilo središčne točke** (midpoint rule). Kako s tem pravilom aproksimiramo ploščino traku od x_n do x_{n+1}?

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx \ ?$$

Pojdite na Okvir 11 za odgovor.

Okvir 11

Pravilo središčne točke aproksimira ploščino traku s ploščino pravokotnika, katerega višina je vrednost funkcije na sredini intervala:

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx f\left(\frac{x_n + x_{n+1}}{2}\right) \cdot h$$

Seštevanje teh pravokotnikov je enakovredno izračunu ploščine histograma funkcije.

Nekoliko boljša aproksimacija je **trapezno pravilo**. Namesto pravokotnika aproksimiramo ploščino vsakega traku s trapezom, ki ga tvorimo s povezavo točk $(x_n, f(x_n))$ in $(x_{n+1}, f(x_{n+1}))$ z ravno črto.

Kakšna je formula za ploščino enega takega trapeza? _____

Pojdite na Okvir 12 za odgovor.

Okvir 12

Ploščina trapeza je povprečje dolžin vzporednih stranic, pomnoženo s širino. Za naš trak je to:

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx \frac{f(x_n) + f(x_{n+1})}{2} \cdot h$$

Ko seštejemo te trapeze po celotnem intervalu $[a, b]$, dobimo sestavljeno trapezno pravilo:

$$\int_a^b f(x)dx \approx h\left[\frac{f_0}{2} + f_1 + f_2 + \cdots + f_{N-1} + \frac{f_N}{2}\right]$$

kjer je $f_n = f(x_n)$. To pravilo je točno za linearne funkcije. Napaka na enem intervalu je reda $O(h^3)$, medtem ko je skupna (globalna) napaka reda $O(h^2)$.

Pojdite na Okvir 13.

Okvir 13

Formule višjega reda

Veliko boljše rezultate z enakim številom vrednotenj funkcije lahko dosežemo z uporabo interpolacije višjega reda.

Simpsonovo pravilo prilega parabolo (kvadratni polinom) skozi tri sosednje točke. To zahteva vrednotenje integrala čez "dvojni trak"širine $2h$. Formula za en tak segment je:

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3}[f_0 + 4f_1 + f_2]$$

Po srečnem naključju je ta formula točna ne le za kvadratne polinome, ampak tudi za kubične polinome. To jo naredi zelo natančno. Napaka celotnega integrala se skalira kot $O(h^4)$.

Obstajajo tudi pravila višjega reda, kot je Bodejevo pravilo (ki uporablja 5 točk), ki še izboljšajo natančnost. Splošni razred teh metod se imenuje Newton-Cotesove formule. Napredne metode, kot je Gaussova-Jacobijeva kvadratura, uporabljajo neenakomerno razporejene točke za doseganje optimalne natančnosti za dano število vrednotenj funkcije.

S tem zaključujemo naš hiter pregled iskanja ničel in integracije. Naloga je zdaj, da te metode uporabite za reševanje praktičnih problemov, kot so tisti, predlagani na prosojnicah (iskanje ničel polinomov ali integriranje funkcij, kot je x^3e^{-x}).

Konec poglavja.

3 Monte Carlo integracija

Okvir 1

To poglavje predstavlja **Monte Carlo (MC) metode**. Glavni cilj je naučiti se izvajati integracijo in simulirati procese z uporabo naključnosti.

Osnovna ideja teh metod je starejša od sodobnih računalnikov. Prvotno je bil "računalnik"oseba, ki je pogosto delala v veliki sobi z drugimi in izvajala izračune ročno. Elektronski "digitalni računalnik"je kasneje prevzel te naloge, z zgodnjimi aplikacijami na ključnih področjih, kot je dešifriranje med drugo svetovno vojno v Bletchley Parku.

Temelj vseh Monte Carlo metod je zmožnost generiranja naključnih števil.

Pojdite na Okvir 2.

Naključna števila

Okvir 2

Prva lekcija računalništva je ključna: **prava naključna števila v programiranju ne obstajajo**.

Računalniki so deterministične naprave. Ne morejo ustvariti prave naključnosti. Namesto tega uporabljamo algoritme, ki generirajo zaporedja števil, ki se *zdijo* naključna. Ta se imenujejo **psevdo-naključna števila**.

Katere lastnosti imajo ti algoritmi za generiranje psevdo-naključnih števil (PRNG)?

- Generirajo števila, ki so običajno enakomerno porazdeljena na intervalu $[0, 1]$. To pomeni, da ima vsako število med 0 in 1 enako verjetnost, da bo generirano.
- So popolnoma **deterministični**.
- So inherentno **periodični**.

Pojdite na Okvir 3.

Okvir 3

Ker so PRNG-ji deterministični algoritmi, če jih zaženete z enako začetno vrednostjo, boste dobili popolnoma enako zaporedje števil. Ta začetna vrednost se imenuje **naključno seme** (random seed).

Zakaj je uporaba določenega semena uporabna v znanstvenem računalništvu? [a] Zagotavlja, da so rezultati resnično naključni. [b] Pospeši izvajanje programa. [c] Omogoča ponovljivost rezultatov in lažje odpravljanje napak.

Izberite odgovor in pojdite na Okvir 4.

Okvir 4

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [c]. Uporaba semena za generator naključnih števil omogoča komurkoli, da ponovi vašo natančno simulacijo ali izračun, kar je bistveno za preverjanje znanstvenih rezultatov. Prav tako izjemno pomaga pri odpravljanju napak, saj je "naključno"obnašanje predvidljivo.

Dober PRNG bi moral imeti dve ključni lastnosti:

- Biti mora **hiter**.
- Imeti mora zelo **dolgo periodo**, preden se zaporedje števil ponovi.

Eden najbolj znanih in široko uporabljanih PRNG-jev je **Mersenne Twister**, ki ima ogromno periodo $2^{19937} - 1$.

Pojdite na Okvir 5.

Okvir 5

Čeprav števila iz PRNG niso resnično naključna, so zasnovana tako, da so **statistično neodvisna**. To je ključna lastnost, saj nam omogoča, da generirana števila obravnavamo kot izide naključnega procesa in zanje uporabljamo orodja statistike.

Obstaja še ena kategorija števil, imenovana **kvazi-naključna števila**. Za razliko od psevdo-naključnih števil, ki poskušajo posnemati naključnost, so kvazi-naključna zaporedja zasnovana tako, da čim bolj enakomerno pokrijejo prostor. Namerno *niso* neodvisna.

Katera vrsta števil je bistvena za simulacije, ki temeljijo na statistični analizi? [a] Psevdo-naključna [b] Kvazi-naključna

Pojdite na Okvir 6 za odgovor.

Okvir 6

Pravilen odgovor je [a] Psevdo-naključna. Ker so statistično neodvisna, pravilno modelirajo naključne procese. Kvazi-naključna števila so uporabna na drugih področjih, kot je računalniška grafika (CGI) za generiranje enakomerno razporejenih vzorcev, ne pa za statistično simulacijo.

Zdaj, ko razumemo orodje (psevdo-naključna števila), ga uporabimo za integracijo.

Pojdite na Okvir 7.

Monte Carlo integracija

Okvir 7

Najlažji način za razumevanje Monte Carlo integracije je z izračunom ploščine lika. To se pogosto imenuje metoda žadeni ali zgreši"(hit-or-miss).

Izračunajmo ploščino enotskega kroga (polmer $r = 1$). To lahko storimo tako:

1. Krog včrtamo v preprost lik, katerega ploščino poznamo, na primer kvadrat. Uporabimo kvadrat, ki se razteza od -1 do 1 v smereh x in y. Ploščina tega kvadrata je $S_0 = (2r)^2 = 4$.
2. Generiramo veliko število, N, naključnih točk (x, y), ki so enakomerno porazdeljene znotraj tega kvadrata.
3. preštejemo, koliko točk pade znotraj kroga (N_{in}) v primerjavi s tistimi zunaj (N_{out}).

Kakšna je verjetnost, P_{in}, da bo naključno vržena točka padla znotraj kroga? (Namig: To je razmerje ploščin).

Pojdite na Okvir 8 za odgovor.

Okvir 8

Verjetnost je razmerje med ploščino kroga ($S_k = \pi r^2 = \pi$) in ploščino kvadrata ($S_0 = 4$):

$$P_{in} = \frac{S_k}{S_0} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

To verjetnost lahko ocenimo iz naše simulacije z deležem točk, ki so padle znotraj:

$$\hat{P}_{in} = \frac{N_{in}}{N_{in} + N_{out}} = \frac{N_{in}}{N_{\text{skupaj}}}$$

Z enačenjem teoretične verjetnosti z našo eksperimentalno oceno lahko izrazimo neznano ploščino, $S_k = S_0 \cdot \hat{P}_{in}$. To lahko celo uporabimo za oceno $\pi \approx 4 \cdot \hat{P}_{in}$.

Pojdite na Okvir 9.

Okvir 9

Napaka te Monte Carlo ocene se zmanjšuje s številom poskusov, N. Napaka je statistične (binomnske) narave in se skalira kot $1/\sqrt{N}$. Ta počasna konvergenca je "boleče dejstvo"MC metod, vendar je velika prednost njena preprostost in splošnost.

Metodo žadeni ali zgreši"lahko uporabimo za lik *katerekoli* zahtevnosti, dokler lahko definiramo mejo (test za znotraj/zunaj) in ga včrtamo v preprost volumen.

Ta ideja se lahko enostavno razširi na višje dimenzije za izračun volumnov. Kateri dve stvari bi potrebovali za izračun volumna kompleksnega 3D objekta z uporabo te metode? 1. ____2. ____

Pojdite na Okvir 10 za odgovor.

Okvir 10

Za izračun 3D volumna z Monte Carlo metodo žadeni ali zgreši"potrebujete: 1. Preprost, včrtan volumen, katerega prostornino poznate (npr. kocka). 2. Kriterij za preverjanje, ali je naključna točka (x, y, z) znotraj ali zunaj objekta.

Pojdite na Okvir 11.

Okvir 11

Metoda povprečne vrednosti za 1D integrale

Sedaj uporabimo to idejo za 1D integral, $\int_a^b f(x)dx$. Namesto metode žadeni ali zgreši"lahko uporabimo koncept **povprečne vrednosti** funkcije. Integral lahko izrazimo kot povprečno vrednost $f(x)$ na intervalu, pomnoženo z dolžino intervala.

$$\int_a^b f(x)dx = \langle f(x) \rangle \cdot (b - a)$$

Povprečno vrednost, $\langle f(x) \rangle$, lahko ocenimo z vzorčenjem funkcije v N naključnih točkah, x_i, ki so enakomerno izbrane iz $[a, b]$, in izračunom njihovega povprečja:

$$\langle f(x) \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

Zato je naša Monte Carlo ocena integrala:

$$\int_a^b f(x)dx \approx \frac{b - a}{N} \sum_{i=1}^{N} f(x_i)$$

Napaka te metode se prav tako skalira kot $1/\sqrt{N}$. Ta "metoda povprečne vrednosti"je na splošno učinkovitejša od metode žadeni ali zgreši".

Pojdite na Okvir 12.

Okvir 12

Prednostno vzorčenje (Importance Sampling)

Metoda povprečne vrednosti vzorči točke enakomerno. Kaj pa, če ima naša funkcija $f(x)$ velik vrh v ozkem območju? Enakomerno vzorčenje lahko ta vrh zgreši, kar vodi do slabe ocene.

Našo oceno lahko izboljšamo z vzorčenjem več točk tam, kjer je funkcija velika. To je ideja **prednostnega vzorčenja**.

Integral prepišemo z uvedbo verjetnostne porazdelitvene funkcije (PDF), $w(x)$, iz katere bomo vzorčili:

$$\int_a^b f(x)dx = \int_a^b \frac{f(x)}{w(x)} w(x)dx$$

To izgleda bolj zapleteno, a je močno. Integral je sedaj pričakovana vrednost funkcije $g(x) = f(x)/w(x)$ za naključne spremenljivke x_i, vzorčene iz porazdelitve $w(x)$.

$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i) = \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{w(x_i)}$$

Katera bi bila *idealna* izbira za porazdelitev vzorčenja $w(x)$, da bi minimizirali napako (varianco) ocene?

Pojdite na Okvir 13 za odgovor.

Okvir 13

Idealna izbira za $w(x)$ bi bila porazdelitev, ki je oblikovana natanko tako kot funkcija, ki jo poskušamo integrirati, vendar normalizirana. To pomeni, $w(x) = f(x)/\int f(x)dx$.

Če bi to lahko storili, bi bila funkcija $g(x) = f(x)/w(x)$ konstanta! Varianca konstante je nič, in naša ocena integrala bi bila popolna že z enim samim vzorcem.

Seveda, če bi poznali $\int f(x)dx$ za ustvarjanje popolne $w(x)$, nam integracije sploh ne bi bilo treba izvajati! Praktična ideja je, da izberemo preprosto PDF $w(x)$, iz katere *znamo* vzorčiti in ki *približno* posnema obliko $f(x)$. To osredotoči vzorčne točke v pomembna območja in dramatično zmanjša napako za enako število vzorcev N.

Pojdite na Okvir 14.

Monte Carlo simulacija

Okvir 14

Poleg integracije se MC metode uporabljajo za simulacijo dogodkov, ki so porazdeljeni po določeni verjetnostni porazdelitvi, $w(x)$. Na primer, simulacija energije delcev pri radioaktivnem razpadu.

Najpreprostejša metoda je spet **zadeni ali zgreši (ali metoda zavračanja)**.

1. Poiščite maksimalno vrednost, h, porazdelitve $w(x)$ na domeni $[a, b]$.
2. Generirajte enakomerno naključno x-koordinato, x_i, v $[a, b]$.
3. Generirajte enakomerno naključno y-koordinato, $\rho \cdot h$, v $[0, h]$.
4. Če je točka $(x_i, \rho \cdot h)$ *pod* krivuljo $w(x)$, šprejmemo"x_i kot naš dogodek. Če je nad, jo žavrnemo"in poskusimo znova.

To je preprosto, a je lahko potratno, če ima funkcija ostre vrhove.

Pojdite na Okvir 15.

Okvir 15

Vzorčenje z inverzno transformacijo

Močnejša, unitarna metoda (ki da veljaven dogodek za vsako naključno število) je **vzorčenje z inverzno transformacijo**. Ta metoda temelji na kumulativni porazdelitveni funkciji (CDF).

CDF, $F(x)$, je integral PDF: $F(x) = \int_a^x w(z)dz$. CDF poteka od 0 do neke maksimalne vrednosti (1, če je $w(x)$ normalizirana).

Metoda deluje tako:

1. Generiramo enakomerno naključno število, ρ, med 0 in 1.
2. To enačimo z normalizirano CDF.
3. Rešimo za x z inverzijo funkcije: $x = F^{-1}(\rho)$.

Ta metoda je izjemno učinkovita, če je inverzna CDF, F^{-1}, znana analitično. Na primer, za generiranje dogodkov iz eksponentne porazdelitve $w(t) = \frac{1}{\tau}e^{-t/\tau}$, je ustrezna naključna spremenljivka $t = -\tau \ln(1 - \rho)$.

S tem zaključujemo naš hiter pregled Monte Carlo metod.

Konec poglavja.

4 Iskanje funkcijskih ekstremov in prilagajanje funkcij meritvam

Okvir 1

Danes se bomo naučili postopkov za iskanje **ekstremov** (maksimumov ali minimumov) funkcij v eni dimenziji. Kot pri vseh numeričnih metodah se kompleksnost teh postopkov povečuje z dimenzionalnostjo prostora, v katerem je funkcija definirana. Zaradi tega so najbolj robustne in zanesljive metode razvite za 1D.

Izkazalo se je tudi, da je postopek **prilagajanja funkcije podatkom** (znan tudi kot modeliranje ali "fitanje") neposredno povezan z iskanjem ekstremov. Tudi o tej povezavi se bomo naučili.

Pojdite na Okvir 2.

Iskanje ekstremov v 1D

Okvir 2

Iskanje ekstremov funkcije $f(x)$ v eni dimenziji lahko razdelimo na dva podproblema, odvisno od tega, kaj vemo o odvodu funkcije, $f'(x)$.

Primer 1: Odvod $f'(x)$ je znan. Če imamo analitični izraz za odvod, se iskanje ekstrema prevede na problem, ki smo ga že rešili. Kateri problem je to? (Namig: Kaj velja za odvod v maksimumu ali minimumu?) _____

Pojdite na Okvir 3 za odgovor.

Okvir 3

Če je odvod $f'(x)$ znan, se iskanje ekstrema prevede na **iskanje ničel** odvoda, tj. reševanje enačbe $f'(x) = 0$. To že znamo rešiti z metodami, kot sta bisekcija ali Newton-Raphson.

Primer 2: Odvod $f'(x)$ ni znan. Obstaja veliko razlogov, zakaj morda ne poznamo odvoda. Na primer, $f(x)$ je lahko sama rezultat zapletenega numeričnega postopka (kot je integracija), ali pa je odvod preprosto pretežko ali računsko predrago izračunati.

V naših vajah se bomo osredotočili na ta drugi, zahtevnejši primer.

Pojdite na Okvir 4.

Okvir 4

Preden začnemo, se moramo zavedati ključnega dejstva: **ne obstaja splošna metoda, ki bi samodejno našla globalni ekstrem** (absolutno najnižjo ali najvišjo točko) poljubne funkcije. Vsaka iterativna metoda se lahko žatakne"v *lokalnem* ekstremu.

Zato iskanje globalnega ekstrema pogosto zahteva dodatno preiskavo, na primer poskušanje z različnimi začetnimi točkami ali primerjavo več lokalnih ekstremov.

Poleg tega so skoraj vsi standardni numerični postopki zasnovani za iskanje **minimuma** funkcije. Kako lahko uporabimo algoritem za minimizacijo za iskanje *maksimuma* funkcije $f(x)$? ____

Pojdite na Okvir 5 za odgovor.

Okvir 5

Da bi našli maksimum funkcije $f(x)$, lahko preprosto poiščemo minimum funkcije $-f(x)$. Vrednost x, ki minimizira $-f(x)$, je ista vrednost x, ki maksimizira $f(x)$.

Še zadnja opomba: vedno preverite vrednosti funkcije na robovih intervala, ki vas zanima! Globalni ekstrem je lahko na krajišču, ne tam, kjer je odvod enak nič.

Pojdite na Okvir 6.

Okvir 6

Da bi numerično identificirali obstoj minimuma na določenem območju, potrebujemo vsaj tri točke, recimo A, B in C, z znanimi funkcijskimi vrednostmi $f(A)$, $f(B)$ in $f(C)$.

Katere pogoje morajo te tri točke izpolnjevati, da "objamejo"minimum? (Namig: Pomislite na vrstni red točk in njihovih ustreznih funkcijskih vrednosti.)

Pojdite na Okvir 7 za odgovor.

Okvir 7

Da bi objeli minimum, potrebujemo trojico točk (A, B, C), tako da:

1. Točke so urejene: $A < B < C$.
2. Srednja točka je najnižja: $f(B) < f(A)$ in $f(B) < f(C)$.

Če so ti pogoji izpolnjeni, vemo, da mora vsaj en lokalni minimum obstajati znotraj intervala $[A, C]$. Točka B služi kot naša prva aproksimacija minimuma.

Naš cilj je sedaj izboljšati to začetno oceno in zožiti interval $[A, C]$, dokler njegova širina ni manjša od neke želene natančnosti, ϵ.

Pojdite na Okvir 8.

Okvir 8

Metoda zlatega reza

Imamo začetni interval $[A, C]$, ki objema minimum, z notranjo točko B. Izbrati moramo novo točko, D, da bi zožili interval. Kratek razmislek pokaže, da bi morali novo točko postaviti v *večjega* od obeh podintervalov, $[A, B]$ ali $[B, C]$.

Recimo, da postavimo novo točko D v interval $[B, C]$. Zdaj imamo štiri točke A, B, D, C. Izračunamo $f(D)$.

- Če je $f(D) < f(B)$, naša nova trojica, ki objema minimum, postane (B, D, C).
- Če je $f(D) > f(B)$, naša nova trojica postane (A, B, D).

V obeh primerih imamo nov, manjši interval, ki objema minimum. Ta postopek lahko ponavljamo, dokler interval ni dovolj majhen.

Toda kam natančno naj postavimo točko D, da bomo najbolj učinkoviti?

Pojdite na Okvir 9.

Okvir 9

Najbolj eleganten in učinkovit način za izbiro nove točke uporablja **zlati rez**, φ. Zlati rez ima edinstveno lastnost samopodobnosti, ki poenostavi iskanje.

Zlati rez φ je definiran z razmerjem $\frac{a+b}{a} = \frac{a}{b} = \varphi$. To vodi do kvadratne enačbe $\varphi^2 - \varphi - 1 = 0$.

Kakšna je vrednost φ?

$$\varphi = \frac{1 \pm \sqrt{1 - 4(1)(-1)}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

Ker gre za razmerje dolžin, vzamemo pozitivno korenino. Torej, $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.61803...$

Pojdite na Okvir 10.

Okvir 10

Moč metode zlatega reza izhaja iz postavitve notranjih točk na določenem deležu širine intervala. Naj bo začetni interval $[A, C]$ s širino $h = C - A$. Izberemo dve notranji točki, B in D, tako da delita interval po zlatem rezu.

$$B = A + (1 - 1/\varphi)h = A + h/\varphi^2$$

$$D = A + (h/\varphi)$$

Opazite, da je $1/\varphi = \varphi - 1 \approx 0,618$ in $1/\varphi^2 = (\varphi - 1)^2 \approx 0,382$.

Čarovnija je v tem: po enem koraku, ko imamo novo, manjšo trojico, ki objema minimum (npr. (A, B, D)), je *stara* notranja točka (B) sedaj popolnoma postavljena, da postane ena od *novih* notranjih točk za naslednjo iteracijo. Na vsakem koraku moramo izračunati samo eno novo točko.

Pri vsakem koraku se širina intervala zmanjša za faktor $1/\varphi$.

Pojdite na Okvir 11.

Okvir 11

Parabolična interpolacija

Alternativna metoda, ki je lahko veliko hitrejša, je **parabolična interpolacija**. Ideja je, da prilegamo parabolo skozi naše tri točke $(A, f(A))$, $(B, f(B))$ in $(C, f(C))$, ki objemajo minimum. Ker ima parabola edinstven minimum, lahko analitično izračunamo lokacijo tega minimuma in jo uporabimo kot našo naslednjo oceno, D.

Ta metoda je zelo hitra, če je funkcija dobro aproksimirana s parabolo blizu svojega minimuma. Kaj je potencialna slabost te metode? (Namig: Pomislite na funkcije, ki niso "parabolične"oblike.)

Pojdite na Okvir 12 za odgovor.

Okvir 12

Glavna slabost čiste parabolične interpolacije je, da je lahko nezanesljiva. Če funkcija ni primerno obnašajoča (npr. zelo položna, ali so točke kolinearne), je lahko izračun minimuma nestabilen ali pa celo pošlje naslednjo oceno izven intervala, ki objema minimum.

Robusten algoritem mora preveriti, ali je nova točka smiselna, in se v nasprotnem primeru zateči k varnejši metodi.

Pojdite na Okvir 13.

Okvir 13

Brentova metoda

Brentova metoda je hibridni algoritem, ki združuje najboljše iz obeh svetov. Poskuša uporabiti hitro parabolično interpolacijo, kadar koli je to mogoče. Vendar pa pri vsakem koraku preveri, ali je bil narejen korak produktiven.

Če je parabolični korak premajhen ali ne izboljša situacije, algoritem samodejno preklopi na počasnejšo, a zagotovljeno konvergentno metodo zlatega reza za enega ali več korakov.

Ta kombinacija hitrosti in robustnosti naredi Brentovo metodo standardno izbiro za 1D minimizacijo v mnogih znanstvenih knjižnicah (npr. ‘scipy.optimize.brent‘).

Pojdite na Okvir 14.

Prilagajanje funkcij podatkom

Okvir 14

Ena najpogostejših aplikacij minimizacije je prilagajanje modelne funkcije nizu izmerjenih podatkov. Recimo, da imamo niz podatkovnih točk (x_i, y_i) z negotovostmi σ_i in modelno funkcijo $y = f(x, \vec{\alpha})$, ki je odvisna od niza neznanih parametrov $\vec{\alpha}$.

Naš cilj je najti vrednosti parametrov $\vec{\alpha}$, ki zagotovijo, da se funkcija "najbolje prilega"podatkom. Potrebujemo način, kako opredeliti, kaj pomeni "najbolje". Najpogostejša metoda je **prilagajanje po metodi najmanjših kvadratov**, ki uporablja statistiko hi-kvadrat (χ^2) kot cenilko.

Kaj mislite, da naredimo z vrednostjo χ^2, da najdemo parametre najboljšega prileganja?

$$\chi^2(\vec{\alpha}) = \sum_i \frac{(y_i - f(x_i, \vec{\alpha}))^2}{\sigma_i^2}$$

[a] Jo maksimiziramo [b] Jo minimiziramo [c] Iščemo, kje je enaka nič

Pojdite na Okvir 15.

Okvir 15

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b] Jo minimiziramo. Vrednost χ^2 predstavlja vsoto kvadratov razlik med podatki in modelom, uteženih z negotovostjo. Manjša vrednost χ^2 pomeni boljše ujemanje. Zato je iskanje parametrov najboljšega prileganja $\vec{\alpha}$ problem minimizacije.

Če je naša modelna funkcija premica, $f(x, \alpha_1, \alpha_2) = \alpha_1 x + \alpha_2$, se ta postopek imenuje **linearna regresija**. Vendar pa načelo velja za katero koli modelno funkcijo, linearno ali ne. Za iskanje minimuma funkcije χ^2 in s tem parametrov najboljšega prileganja lahko uporabimo večdimenzionalne različice algoritmov, kot je Brentova metoda (npr. metoda Downhill Simplex). Profesionalne knjižnice, kot je 'scipy.optimize.curve_fit', ponujajo močna orodja za to nalogo.

S tem zaključujemo naš pregled minimizacije funkcij in prilagajanja.

Konec poglavja.

5 Diagonalizacija matrik, lastne vrednosti in vektorji

Okvir 1

To poglavje obravnava zelo pogost problem v matematiki in fiziki: iskanje lastnih vrednosti in lastnih vektorjev matrike. Za dano kvadratno matriko $\mathbf{A}$ iščemo posebne vektorje $\mathbf{x}$ in skalarje λ, ki zadoščajo enačbi lastnih vrednosti:

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x}$$

Tu je λ **lastna vrednost** in $\mathbf{x}$ je ustrezen **lastni vektor**. To pomeni, da ko matrika $\mathbf{A}$ deluje na svoj lastni vektor $\mathbf{x}$, je rezultat preprosto isti vektor, pomnožen z lastno vrednostjo λ.

Osredotočili se bomo na iskanje lastnih vrednosti IN njihovih ustreznih lastnih vektorjev.

Pojdite na Okvir 2.

Okvir 2

Enačbo lastnih vrednosti lahko prepišemo kot $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$, kjer je $\mathbf{I}$ identitetna matrika. Da bi ta enačba imela netrivialno rešitev (tj. $\mathbf{x} \neq 0$), mora biti matrika $(\mathbf{A} - \lambda\mathbf{I})$ singularna. Kaj to pomeni za njeno determinanto?

[a] $\det(\mathbf{A} - \lambda\mathbf{I}) > 0$ [b] $\det(\mathbf{A} - \lambda\mathbf{I}) < 0$ [c] $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$

Izberite odgovor in pojdite na Okvir 3.

Okvir 3

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [c]. Netrivialna rešitev obstaja le, če je $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. Ta enačba se imenuje **karakteristični polinom**. Iskanje njegovih korenov nam da lastne vrednosti λ. Čeprav ta analitični pristop deluje za majhne matrike, postane nepraktičen za dimenzije, večje od približno 4x4. Potrebujemo numerične metode.

Osredotočili se bomo na **simetrične matrike** ($\mathbf{A} = \mathbf{A}^T$). Simetrične matrike imajo dve zelo lepi lastnosti:

- Njihove lastne vrednosti so vedno **realne**.
- Diagonalizirajo se lahko z **ortogonalno matriko** $\mathbf{Z}$ (kjer je $\mathbf{Z}^{-1} = \mathbf{Z}^T$).

Stolpci te matrike $\mathbf{Z}$ so lastni vektorji matrike $\mathbf{A}$.

Pojdite na Okvir 4.

Iterativne metode

Okvir 4

Potenčna iterativna metoda

Najpreprostejši iterativni pristop za iskanje največje lastne vrednosti je **potenčna metoda**.

1. Začnemo z naključnim začetnim vektorjem, $\mathbf{x}^{(0)}$.
2. Normaliziramo ta vektor.

3. Ponavljajoče uporabljamo matriko $\mathbf{A}$ na vektorju v iterativnem postopku:

$$\mathbf{y}^{(i+1)} = \mathbf{A}\mathbf{x}^{(i)} \qquad \text{in} \qquad \mathbf{x}^{(i+1)} = \frac{\mathbf{y}^{(i+1)}}{|\mathbf{y}^{(i+1)}|}$$

Drugi korak je ponovna normalizacija, da preprečimo nenadzorovano rast velikosti vektorja.

H kateri vrednosti konvergira vektor $\mathbf{x}^{(i)}$, ko število iteracij i narašča? _____

Pojdite na Okvir 5 za odgovor.

Okvir 5

Vektor $\mathbf{x}^{(i)}$ konvergira k **lastnemu vektorju, ki ustreza največji lastni vrednosti**.

Največjo lastno vrednost samo, $\lambda_{\max}$, lahko nato najdemo z Rayleighovim kvocientom:

$$\lambda_k = \frac{(\mathbf{x}^{(k)})^T(\mathbf{A}\mathbf{x}^{(k)})}{(\mathbf{x}^{(k)})^T\mathbf{x}^{(k)}} = \frac{\mathbf{x}^{(k)} \cdot (\mathbf{A}\mathbf{x}^{(k)})}{|\mathbf{x}^{(k)}|^2}$$

Ta metoda je lahko počasna in lahko ne uspe, če je začetna ocena $\mathbf{x}^{(0)}$ nesrečna (npr. ortogonalna na želeni lastni vektor). Prilagodimo jo lahko tudi za iskanje najmanjše lastne vrednosti z uporabo potenčne iteracije na inverzni matriki, $\mathbf{A}^{-1}$.

Kako lahko najdemo *druge* lastne vrednosti in lastne vektorje?

Pojdite na Okvir 6.

Okvir 6

Ko najdemo največjo lastno vrednost λ_1 in njen ustrezen lastni vektor $\mathbf{x}_1$, lahko najdemo naslednjo tako, da prvo odstranimo iz matrike. Ta postopek se imenuje **deflacija**.

Wielandtova deflacija konstruira novo matriko $\mathbf{A}'$, ki ima enake lastne vrednosti kot $\mathbf{A}$, le da je λ_1 zamenjana z 0:

$$\mathbf{A}' = \mathbf{A} - \lambda_1\mathbf{x}_1\mathbf{x}_1^T$$

(Tu je $\mathbf{x}_1\mathbf{x}_1^T$ zunanji produkt). Nato lahko na $\mathbf{A}'$ uporabimo potenčno metodo, da najdemo naslednjo največjo lastno vrednost. Ta postopek lahko ponavljamo, vendar se napake kopičijo.

Pojdite na Okvir 7.

Transformacijske metode

Okvir 7

Bolj robusten in napreden pristop je iskanje transformacijske matrike $\mathbf{Z}$, ki diagonalizira $\mathbf{A}$ naenkrat. Cilj je najti ortogonalno matriko $\mathbf{Z}$, tako da:

$$\mathbf{Z}^{-1}\mathbf{A}\mathbf{Z} = \mathbf{Z}^T\mathbf{A}\mathbf{Z} = \mathbf{D}$$

kjer je $\mathbf{D}$ diagonalna matrika, katere elementi so lastne vrednosti matrike $\mathbf{A}$. Stolpci matrike $\mathbf{Z}$ bodo ustrezni normalizirani lastni vektorji.

Strategija je, da $\mathbf{Z}$ konstruiramo kot zaporedje preprostejših ortogonalnih transformacij:

$$\mathbf{Z} = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \cdots$$

Vsaka transformacija $\mathbf{P}_k$ je izbrana tako, da naredi matriko "bolj diagonalno", dokler ne konvergira.

Pojdite na Okvir 8.

Okvir 8

Jacobijeva metoda

Jacobijeva metoda je eleganten pristop, ki uporablja zaporedje **ravninskih rotacij** za izničevanje izvendiagonalnih elementov matrike.

Poljubna rotacija v N-dimenzijah, ki vpliva le na p-to in q-to koordinato, je predstavljena z matriko $\mathbf{P}_{pq}$, ki je podobna identitetni matriki, razen štirih elementov:

$$(\mathbf{P}_{pq})_{pp} = c, \quad (\mathbf{P}_{pq})_{qq} = c, \quad (\mathbf{P}_{pq})_{pq} = s, \quad (\mathbf{P}_{pq})_{qp} = -s$$

kjer sta $c = \cos(\theta)$ in $s = \sin(\theta)$ za nek kot rotacije θ.

Transformacija $\mathbf{A}' = \mathbf{P}_{pq}^T \mathbf{A} \mathbf{P}_{pq}$ spremeni le vrstice in stolpce p in q matrike $\mathbf{A}$. Kako izberemo kot θ?

Pojdite na Okvir 9 za odgovor.

Okvir 9

Kot θ izberemo posebej tako, da je izvendiagonalni element a'_{pq} v novi matriki enak nič. To vodi do pogoja:

$$\cot(2\theta) = \frac{a_{qq} - a_{pp}}{2a_{pq}}$$

Rešitev te enačbe za $\cos(\theta)$ in $\sin(\theta)$ nam da potrebno rotacijsko matriko.

Problem je v tem, da bo naslednja rotacija (za izničevanje drugega elementa) na splošno "pokvarila"ničlo, ki smo jo pravkar ustvarili. Vendar pa Jacobijeva metoda zagotovo konvergira, ker se vsota kvadratov izvendiagonalnih elementov z vsakim korakom zmanjšuje.

Postopek je, da večkrat "pometamo"čez vse izvendiagonalne elemente in jih izničujemo, dokler matrika ni diagonalna do želene tolerance.

Pojdite na Okvir 10.

Okvir 10

Metoda QR in tridiagonalizacija

Sodobnejši in na splošno hitrejši pristop za iskanje vseh lastnih vrednosti je **metoda QR**. To je iterativna metoda, ki temelji na **razcepu QR**, kjer lahko vsako matriko $\mathbf{A}$ razcepimo na $\mathbf{A} = \mathbf{QR}$, pri čemer je $\mathbf{Q}$ ortogonalna matrika in $\mathbf{R}$ zgornje-trikotna matrika.

Iterativni postopek je:

1. Začnemo z $\mathbf{A}_0 = \mathbf{A}$.
2. V koraku k razcepimo $\mathbf{A}_k = \mathbf{Q}_k \mathbf{R}_k$.
3. Konstruiramo naslednjo matriko kot $\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k$.

Kakšno je razmerje med $\mathbf{A}_{k+1}$ in $\mathbf{A}_k$? (Namig: Vstavite $\mathbf{R}_k = \mathbf{Q}_k^T \mathbf{A}_k$ v tretji korak). _____

Pojdite na Okvir 11.

Okvir 11

Razmerje je **podobnostna transformacija**:

$$\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k = (\mathbf{Q}_k^T \mathbf{A}_k)\mathbf{Q}_k = \mathbf{Q}_k^T \mathbf{A}_k \mathbf{Q}_k$$

Ker gre za podobnostno transformacijo, ima $\mathbf{A}_{k+1}$ enake lastne vrednosti kot $\mathbf{A}_k$. Ta iterativni postopek konvergira k zgornje-trikotni matriki (ali diagonalni matriki, če je $\mathbf{A}$ simetrična). Lastne vrednosti se pojavijo na diagonali.

Za splošne matrike je ta postopek računsko drag. Ključno spoznanje je, da je metoda *veliko hitrejša* za matrike, ki so že skoraj diagonalne. Standardni postopek je zato dvostopenjski:

1. Najprej uporabimo končno število transformacij (kot so Householderjeve refleksije), da simetrično matriko reduciramo na **tridiagonalno obliko**.

2. Nato uporabimo algoritem QR na veliko preprostejši tridiagonalni matriki, da najdemo lastne vrednosti.

Ta dvostopenjski postopek je v središču večine sodobnih, robustnih reševalcev lastnih vrednosti, kot so tisti v NumPy in SciPy.

Pojdite na Okvir 12.

Splošna diagonalizacija: SVD

Okvir 12

Kaj pa, če matrika ni kvadratna ali simetrična? Koncepta lastnih vrednosti in lastnih vektorjev ne veljata na enak način. Posplošitev diagonalizacije za *poljubno* matriko $m \times n$ je **Singular Value Decomposition (SVD) - razcep singularnih vrednosti**.

SVD pravi, da lahko vsako matriko $\mathbf{A}$ razcepimo kot:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

- $\mathbf{U}$ je ortogonalna matrika $m \times m$.
- $\mathbf{V}$ je ortogonalna matrika $n \times n$.
- $\mathbf{\Sigma}$ je diagonalna matrika $m \times n$, ki na diagonali vsebuje nenegativne **singularne vrednosti**.

SVD je izjemno močno in stabilno numerično orodje. Razkriva temeljno delovanje matrike kot kombinacijo rotacije ($\mathbf{V}^T$), skaliranja ($\mathbf{\Sigma}$) in še ene rotacije ($\mathbf{U}$). Uporablja se povsod, od podatkovne znanosti in strojnega učenja do stiskanja slik.

S tem zaključujemo naš hiter pregled diagonalizacije matrik.

Konec poglavja.

6 Enačbe hoda

Okvir 1

Naslov tega poglavja, "Enačbe hoda,"je le drugo ime za zelo pogost in pomemben tip problema: **navadne diferencialne enačbe (NDE) prvega reda** z danim začetnim pogojem. To je znano kot **problem začetnih vrednosti (IVP)**.

Splošna oblika IVP je:

$$y'(x) = f(x, y), \quad \text{z znano začetno točko} \quad y(x_0) = y_0$$

Podan imamo odvod funkcije, $y'(x)$, ki je lahko odvisen tako od položaja x kot od vrednosti funkcije y. Podana imamo tudi eno točko (x_0, y_0), skozi katero mora potekati rešitev. Naš cilj je najti funkcijo $y(x)$ za druge vrednosti x.

Metode, ki jih obravnavamo za ta skalarni (1D) problem, je mogoče razširiti na sisteme enačb prvega reda, kar nam omogoča reševanje problemov z večdimenzionalnimi vektorji.

Pojdite na Okvir 2.

Okvir 2

Čeprav so analitične rešitve včasih možne, naletimo na težave, takoj ko je desna stran, $f(x, y)$, odvisna od neznane funkcije y na kakršen koli netrivialen način.

Numerično pa lahko te probleme rešimo z različnimi aproksimacijskimi metodami. Osnovna ideja vseh teh metod je, da začnemo v točki (x_0, y_0) in delamo majhne korake za oceno vrednosti y v naslednjih točkah.

Najpreprostejša od vseh metod je **Eulerjeva metoda**. Poglejmo, kako deluje.

Pojdite na Okvir 3.

Okvir 3

Eulerjeva metoda

Za rešitev IVP najprej postavimo enakomerno mrežo točk, ki se začne pri x_0, z velikostjo koraka h:

$$x_i = x_0 + i \cdot h, \quad \text{za } i = 0, 1, \ldots, N$$

Našo numerično aproksimacijo prave rešitve $y(x_i)$ bomo označili z Y_i. Že vemo, da je $Y_0 = y_0$. Kako najdemo Y_1?

Eulerjeva metoda uporablja dve ključni aproksimaciji:

1. Na desni strani enačbe $y' = f(x, y)$ vrednost funkcije f izračunamo v naši trenutno znani točki: $f(x_i, Y_i)$.

2. Na levi strani odvod $y'(x)$ aproksimiramo s preprosto **končno diferenco**:

$$y'(x_i) \approx \frac{y(x_{i+1}) - y(x_i)}{h} \approx \frac{Y_{i+1} - Y_i}{h}$$

Z enačenjem teh dveh aproksimacij, kakšna je posledična formula za iskanje naslednje točke, Y_{i+1}?

Pojdite na Okvir 4 za odgovor.

Okvir 4

Z enačenjem obeh strani dobimo:

$$\frac{Y_{i+1} - Y_i}{h} = f(x_i, Y_i)$$

Z reševanjem za Y_{i+1} dobimo iteracijsko formulo **Eulerjeve metode**:

$$Y_{i+1} = Y_i + h \cdot f(x_i, Y_i), \quad \text{z} \quad Y_0 = y_0$$

Ta metoda je zelo preprosta: da bi našli naslednjo točko, vzamemo trenutno točko in prištejemo velikost koraka h, pomnoženo z naklonom, izračunanim v trenutni točki. To je enakovredno sledenju tangenti za celoten korak.

Ta metoda je točna za linearne funkcije (premice).

Pojdite na Okvir 5.

Okvir 5

Napaka Eulerjeve metode

Če je metoda točna za premico, ne pa za parabolo, lahko z uporabo Taylorjevega razvoja ugotovimo, da je napaka, ki jo naredi v enem koraku (**lokalna napaka**), reda h^2. To zapišemo kot $O(h^2)$. Manjša kot je velikost koraka h, veliko manjša je napaka na korak.

Vendar pa moramo za pot od a do b narediti $N = (b - a)/h$ korakov. Napake iz vsakega koraka se seštevajo. Skupna **globalna napaka** je vsota teh lokalnih napak. Pri Eulerjevi metodi je globalna napaka za en red slabša od lokalne napake.

Globalna napaka $\approx N \times$ (Lokalna napaka) $\propto \frac{1}{h} \times h^2 = O(h)$.

To pomeni, da če velikost koraka prepolovite, prepolovite tudi skupno napako. To ni zelo učinkovito.

Ali je manjši korak vedno boljši? Katera praktična omejitev nam preprečuje uporabo neskončno majhnih korakov? _____

Pojdite na Okvir 6 za odgovor.

Okvir 6

Manjši korak ni vedno boljši zaradi **končne natančnosti računalnika**. Kot kaže graf iz prosojnic, je skupna napaka pri numerični metodi vsota dveh stvari:

- **Napaka diskretizacije:** Napaka, ki izhaja iz aproksimacije metode (npr. aproksimacija krivulje s premicami). Ta se zmanjšuje, ko se h manjša.
- **Napaka zaokroževanja:** Napaka, ki izhaja iz nezmožnosti računalnika, da bi shranil števila z neskončno natančnostjo. Ta napaka se kopiči z vsakim izračunom in postane *slabša*, ko naredite več korakov (tj. ko se h manjša).

Obstaja **optimalna velikost koraka,** h, ki uravnoteži ta dva konkurenčna vira napak. Zmanjšanje velikosti koraka pod ta optimum bo dejansko naredilo končni rezultat manj točen.

Pojdite na Okvir 7.

Okvir 7

Heunova metoda (Modificirana Eulerjeva)

Eulerjevo metodo lahko izboljšamo tako, da dobimo boljšo oceno naklona za korak. **Heunova metoda**, imenovana tudi metoda prediktor-korektor, to stori v dveh fazah:

1. **Prediktor:** Najprej naredimo začasen Eulerjev korak, da napovemo vrednost v naslednji točki: $\tilde{Y}_{i+1} = Y_i + h \cdot f(x_i, Y_i)$.

2. **Korektor:** Nato izračunamo naklon v tej napovedani točki, $f(x_{i+1}, \tilde{Y}_{i+1})$. Končni korak naredimo z uporabo *povprečja* naklona na začetku in napovedanega naklona na koncu:

$$Y_{i+1} = Y_i + \frac{h}{2}\left[f(x_i, Y_i) + f(x_{i+1}, \tilde{Y}_{i+1})\right]$$

Ta metoda je točna za parabole in ima veliko boljšo globalno napako $O(h^2)$. Kompromis je, da zahteva dve vrednotenji funkcije na korak.

Pojdite na Okvir 8.

Okvir 8

Metode Runge-Kutta

Idejo uporabe vmesnih korakov za boljšo oceno naklona je mogoče posplošiti. To vodi do družine **metod Runge-Kutta (RK)**.

Metoda središčne točke je še ena metoda drugega reda (globalna napaka $O(h^2)$), ki uporablja dve vrednotenji na korak. Naredi polovični korak z uporabo Eulerjevega naklona, izračuna naklon na tej središčni točki in nato uporabi ta središčni naklon za celoten korak iz prvotne točke.

Najbolj razširjen od vseh reševalcev NDE je **klasična metoda Runge-Kutta 4. reda (RK4)**. Uporablja uteženo povprečje štirih izračunov naklona na korak, da doseže zelo natančno globalno napako $O(h^4)$. To pomeni, da prepolovitev velikosti koraka zmanjša napako za faktor 16! Njene formule so zapletene, vendar je rezultat zelo močan in splošno uporaben algoritem.

Pojdite na Okvir 9.

Okvir 9

Metode s prilagodljivim korakom

Do sedaj smo predpostavljali fiksno velikost koraka h. To ni vedno optimalno. Če se rešitvena krivulja spreminja počasi, lahko naredimo velike korake, če pa se spreminja hitro, moramo narediti majhne korake za ohranjanje natančnosti.

Prilagodljive metode samodejno prilagajajo velikost koraka. Pogosta tehnika, uporabljena v metodi **Runge-Kutta-Fehlberg (RKF45)**, je, da se naslednja točka izračuna z dvema različnima metodama hkrati (npr. z metodo 4. in 5. reda).

Razlika med obema rezultatoma, $|Y_{i+1}^{(5)} - Y_{i+1}^{(4)}|$, daje oceno lokalne napake. Kako lahko to oceno napake uporabimo za nadzor naslednjega koraka? _____

Pojdite na Okvir 10 za odgovor.

Okvir 10

Ocena napake se uporablja za nadzor velikosti koraka na naslednji način:

- Če je ocenjena napaka večja od želene tolerance, se trenutni korak zavrne, velikost koraka h se zmanjša, in korak se ponovno izračuna.

- Če je ocenjena napaka veliko manjša od tolerance, se korak sprejme, in velikost koraka h se za naslednji korak poveča, da se izboljša učinkovitost.

To omogoča algoritmu, da samodejno dela majhne korake le takrat, ko je to potrebno, kar ga naredi tako natančnega kot učinkovitega.

Pojdite na Okvir 11.

Okvir 11

Stabilnost metode

Pri reševanju NDE moramo upoštevati tudi **stabilnost** metode. Pri določenih enačbah, še posebej tistih, ki opisujejo procese razpadanja (kot je $y' = -ky$, kjer je $k > 0$), lahko numerična metoda "eksplodira"in da divje nihajoče ali neskončne rezultate, če je velikost koraka h prevelika.

Za Eulerjevo metodo, uporabljeno na $y' = -ky$, je pogoj stabilnosti $|1 - kh| < 1$, kar zahteva, da je velikost koraka $h < 2/k$. Če je h izbran večji od tega, bo numerična rešitev nestabilna in bo divergirala od prave rešitve, čeprav je prava rešitev preprost razpad proti ničli.

Različne metode imajo različna območja stabilnosti, kar je ključen dejavnik pri izbiri pravega algoritma za problem.

S tem zaključujemo naš pregled metod za reševanje problemov začetnih vrednosti.

Konec poglavja.

7 Newtonov zakon

Okvir 1

Danes bomo naše znanje o reševanju diferencialnih enačb razširili s prvega reda na **diferencialne enačbe drugega reda**.

Drugi Newtonov zakon gibanja je odličen fizikalni primer NDE drugega reda, zaradi česar je odličen primer za izgradnjo intuicije. Ta razširitev nam bo omogočila reševanje diferencialnih enačb poljubnega reda in v poljubnih dimenzijah.

Pojdite na Okvir 2.

Okvir 2

V svoji najsplošnejši vektorski obliki je drugi Newtonov zakon zapisan kot:

$$\vec{F} = m\frac{d^2\vec{r}}{dt^2}$$

kjer je $\vec{F}$ vektor sile, m je masa in $\vec{r}$ je vektor položaja.

To je NDE drugega reda, ker vključuje drugi odvod položaja. Da bi našli specifično trajektorijo, potrebujemo več kot le enačbo. Kaj še je potrebno za enolično določitev rešitve? (Namig: Pomislite, kaj morate vedeti, da bi napovedali pot vržene žoge). _____

Pojdite na Okvir 3 za odgovor.

Okvir 3

Za enolično rešitev potrebujemo ustrezne **začetne pogoje**. Za NDE drugega reda moramo poznati začetno stanje tako funkcije kot njenega prvega odvoda. Za Newtonov zakon to pomeni, da potrebujemo začetni položaj in začetno hitrost:

$$\vec{r}(t=0) = \vec{r}_0 \quad \text{in} \quad \dot{\vec{r}}(t=0) = \vec{v}_0$$

V treh prostorskih dimenzijah, $\vec{r} = (x, y, z)$, to pomeni, da imamo sistem treh NDE drugega reda in potrebujemo šest začetnih pogojev (začetni položaj in hitrost za vsako dimenzijo).

Sila $\vec{F}$ je lahko funkcija časa t, položaja $\vec{r}$ in hitrosti $\dot{\vec{r}}$. To naredi splošni problem zelo zapleten.

Pojdite na Okvir 4.

Okvir 4

Ključni korak za numerično reševanje NDE višjega reda je njena pretvorba v sistem **NDE prvega reda**.

Kako lahko to storimo za Newtonov zakon? Vpeljemo vmesno spremenljivko. Najbolj fizikalna izbira je **hitrost** $\vec{v}$ ali **gibalna količina** $\vec{p}$. Izberimo gibalno količino, $\vec{p} = m\dot{\vec{r}}$.

Z uporabo te definicije, kako lahko prepišemo drugi Newtonov zakon, $\ddot{\vec{r}} = \vec{F}/m$, kot sistem dveh *diferencialnih enačb prvega reda* za spremenljivki $\vec{r}$ in $\vec{p}$? 1. $\dot{\vec{r}} =?$ 2. $\dot{\vec{p}} =?$

Pojdite na Okvir 5 za odgovor.

Okvir 5

Z uvedbo gibalne količine pretvorimo eno enačbo drugega reda v sistem dveh sklopljenih enačb prvega reda:

$$\dot{\vec{r}}(t) = \vec{p}(t)/m$$
$$\dot{\vec{p}}(t) = \vec{F}(t, \vec{r}, \vec{p})$$

Prva enačba je preprosto definicija gibalne količine. Druga enačba izhaja iz dejstva, da je $\dot{\vec{p}} = m\ddot{\vec{r}} = \vec{F}$.

Tudi začetni pogoji se pretvorijo: $\vec{r}(t=0) = \vec{r}_0$ in $\vec{p}(t=0) = \vec{p}_0 = m\vec{v}_0$.

Zdaj imamo sistem šestih sklopljenih NDE prvega reda (tri za komponente $\vec{r}$ in tri za komponente $\vec{p}$) s šestimi začetnimi pogoji.

Pojdite na Okvir 6.

Okvir 6

To lahko naredimo še bolj prepoznavno, če naše spremenljivke združimo v en šestdimenzionalni vektor stanja, $\vec{y}$:

$$\vec{y} = (\vec{r}, \vec{p}) = (x, y, z, p_x, p_y, p_z)$$

Naš sistem dveh vektorskih enačb prvega reda lahko sedaj zapišemo kot en sam IVP prvega reda za vektor stanja $\vec{y}$:

$$\dot{\vec{y}}(t) = \vec{f}(t, \vec{y}), \quad \text{z} \quad \vec{y}(t=0) = \vec{y}_0$$

To izgleda popolnoma enako kot problem začetnih vrednosti iz prejšnjega poglavja! Edina razlika je, da so naše spremenljivke sedaj vektorji namesto skalarjev.

To pomeni, da lahko za reševanje tega problema neposredno uporabimo vse metode, ki smo se jih že naučili (Euler, Midpoint, Runge-Kutta, itd.). Na primer, metoda središčne točke v N-dimenzijah je:

$$\vec{Y}_{i+1/2} = \vec{Y}_i + \frac{h}{2}\vec{f}(x_i, \vec{Y}_i)$$

$$\vec{Y}_{i+1} = \vec{Y}_i + h \cdot \vec{f}(x_i + h/2, \vec{Y}_{i+1/2})$$

Edina razlika od 1D primera je pravilna uporaba vektorske aritmetike. Mnoge programske knjižnice, kot je NumPy v Pythonu, to opravijo samodejno.

Pojdite na Okvir 7.

Simplektične metode

Okvir 7

V fiziki se pogosto srečujemo s sistemi, ki imajo **ohranitvene količine**, kot so energija, gibalna količina ali vrtilna količina.

Nobena od standardnih numeričnih metod, ki smo jih obravnavali (Euler, RK4, itd.), ni bila posebej zasnovana za upoštevanje teh ohranitvenih zakonov. Po mnogih korakih bodo njihove nakopičene napake običajno povzročile, da bo izračunana energija sistema odstopala, četudi je velikost koraka majhna.

Kako imenujemo metode, ki so posebej zasnovane za ohranjanje energije sistema? _____

Pojdite na Okvir 8 za odgovor.

Okvir 8

Metode, ki so zasnovane za ohranjanje energije (natančneje, za ohranjanje volumna faznega prostora v Hamiltonovih sistemih), se imenujejo **simplektični integratorji**.

Zgodba je matematično bolj zapletena, a za naše namene je to dobra delovna interpretacija. Čeprav so izpeljave pogosto prikazane v 1D zaradi jasnosti, jih je enostavno razširiti na N-dimenzij.

Pojdite na Okvir 9.

Okvir 9

Ena najpreprostejših in najučinkovitejših simplektičnih metod je **Verletova (ali Störmer-Verletova) metoda**. Znana je tudi kot **metoda "žabjih skokov"(leapfrog)**.

Njena globalna napaka je le $O(h^2)$, vendar zahteva le *eno* vrednotenje sile na korak, zaradi česar je zelo učinkovita. Ključno je, da čeprav ne ohranja energije popolnoma, napaka energije niha okoli konstantne vrednosti, namesto da bi sčasoma odstopala. To jo naredi odlično za dolgoročne simulacije orbitalne mehanike.

Ime "žabji skoki"izvira iz načina, kako posodablja položaj in hitrost. V eni od pogostih formulacij posodablja hitrost na polovičnih korakih in položaj na celih korakih, pri čemer se količini medsebojno "preskakujeta".

Pojdite na Okvir 10.

Okvir 10

Metodo "žabjih skokovža enačbo drugega reda $y'' = f(y)$ lahko izpeljemo na naslednji način. Najprej vpeljemo hitrost $v = y'$ kot vmesno spremenljivko.

$$y' = v, \quad v' = f(y)$$

Korak posodobitve je razdeljen na dve polovici:

1. Najprej posodobimo hitrost do polovičnega koraka:

$$v_{n+1/2} = v_n + \frac{h}{2} f(y_n)$$

2. Nato uporabimo to hitrost na polovičnem koraku, da posodobimo položaj za celoten korak:

$$y_{n+1} = y_n + h \cdot v_{n+1/2}$$

3. Na koncu uporabimo nov položaj, da posodobimo hitrost za drugo polovico koraka:

$$v_{n+1} = v_{n+1/2} + \frac{h}{2} f(y_{n+1})$$

Opazite, da lahko vrednotenje sile $f(y_n)$ iz prvega dela ponovno uporabimo.

Pojdite na Okvir 11.

Okvir 11

Z odpravo vmesne hitrosti v lahko zapišemo Verletovo metodo v njeni najpogostejši obliki, **metodi centralne diference**.

Izhajajoč iz pravil za posodabljanje in z nekaj algebre pridemo do izjemno preproste formule, ki neposredno povezuje položaje v treh zaporednih časovnih korakih:

$$y_{n+1} - 2y_n + y_{n-1} = h^2 f(y_n)$$

To lahko izpeljemo z aproksimacijo drugega odvoda y'' s formulo za centralno diferenco. Ta metoda je simplektična in je v središču mnogih simulacij molekularne dinamike. Manjša neprijetnost je, da ni šamozagonska"; da bi našli y_1, morate poznati tako y_0 kot posebno vrednost za y_{-1} (ali pa y_1 dobite s Taylorjevim korakom).

S tem zaključujemo naš pregled metod za reševanje Newtonovih zakonov.

Konec poglavja.

8 Robni problem lastnih vrednosti

Okvir 1

V zadnjem poglavju smo videli, kako reševati diferencialne enačbe drugega reda, ko so podani *začetni pogoji* — to pomeni, da sta vrednost funkcije in njenega odvoda znani v isti začetni točki. To se imenuje problem začetnih vrednosti (IVP).

Danes bomo to razširili na drugačen razred problemov, kjer so pogoji podani v različnih točkah, običajno na mejah intervala. To se imenuje **robni problem (BVP)**.

Konceptualno ima to poglavje dva dela:

- Učenje reševanja splošnih BVP za diferencialne enačbe.
- Uporaba tega znanja na posebnem tipu BVP: iskanje **lastnih funkcij** in **lastnih vrednosti** diferencialnega operatorja.

Pojdite na Okvir 2.

Okvir 2

Danes rešujemo probleme oblike:

$$y''(x) = f(x, y, y')$$

kjer iščemo funkcijo $y(x)$ na intervalu $[a, b]$. Namesto začetnih pogojev imamo podane robne pogoje. Pogosti tipi vključujejo:

- **Dirichletovi pogoji:** Vrednost funkcije je podana na obeh koncih.

$$y(a) = \alpha, \quad y(b) = \beta$$

- **von Neumannovi pogoji:** Odvod funkcije je podan na obeh koncih.

$$y'(a) = \alpha, \quad y'(b) = \beta$$

- **Mešani pogoji:** Kombinacija funkcije in njenega odvoda je podana na koncih.

Kako lahko rešimo problem, kjer nimamo vseh potrebnih informacij v eni sami začetni točki, da bi začeli z našo postopno integracijo?

Pojdite na Okvir 3.

Strelska metoda

Okvir 3

Močno in intuitivno orodje za reševanje BVP je **strelska metoda**. Osnovna ideja je, da BVP pretvorimo v IVP, ki ga že znamo rešiti. To storimo tako, da "uganemo"manjkajoče začetne pogoje.

Oglejmo si linearni BVP z Dirichletovimi pogoji:

$$y'' + P(x)y' + Q(x)y = R(x), \quad \text{z} \quad y(a) = \alpha, y(b) = \beta$$

Imamo začetni položaj $y(a) = \alpha$, ne poznamo pa začetnega naklona, $y'(a)$.

Kakšna je strategija strelske metode? [a] Uganemo končno vrednost $y(b)$. [b] Uganemo manjkajoči začetni naklon $y'(a)$. [c] Uganemo celotno rešitev $y(x)$.

Izberite odgovor in pojdite na Okvir 4.

Okvir 4

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b]. Uganemo manjkajoči začetni naklon, recimo mu ϑ. Zdaj imamo popoln niz začetnih pogojev: $y(a) = \alpha$ in $y'(a) = \vartheta$. To je IVP!

Ta IVP lahko zdaj rešimo s katero koli standardno metodo (kot je Runge-Kutta) in integriramo od $x = a$ do $x = b$. Na koncu preverimo dobljeno vrednost, $y(b)$. Skoraj zagotovo ne bo enaka zahtevani robni vrednosti β, ker je bila naša začetna ocena za naklon, ϑ, napačna.

Problem se je sedaj preoblikoval. Najti moramo pravilen začetni štrelski kot"ϑ, ki bo našo rešitev pripeljal do ciljne vrednosti β pri $x = b$.

Pojdite na Okvir 5.

Okvir 5

Za **linearno** NDE smo lahko zelo pametni. Zaradi načela superpozicije lahko splošno rešitev $Y(x)$ dobimo s kombinacijo rešitev dveh preprostejših IVP:

1. Partikularno rešitev, $u(x)$, ki zadošča nehomogeni enačbi s pravilnim začetnim položajem, a ničelnim naklonom:

$$u'' + Pu' + Qu = R, \quad \text{z} \quad u(a) = \alpha, u'(a) = 0$$

2. Homogeno rešitev, $v(x)$, ki zadošča homogeni enačbi z ničelnim začetnim položajem, a enotskim naklonom:

$$v'' + Pv' + Qv = 0, \quad \text{z} \quad v(a) = 0, v'(a) = 1$$

Splošna rešitev je nato linearna kombinacija: $Y(x) = u(x)+\vartheta v(x)$. Ta samodejno zadošča $Y(a) = \alpha$ in $Y'(a) = \vartheta$.

Sedaj lahko rešimo za pravilni ϑ_0 tako, da prisilimo to rešitev, da ustreza končnemu robnemu pogoju, $Y(b) = \beta$. Kakšen je posledični izraz za ϑ_0? _____

Pojdite na Okvir 6 za odgovor.

Okvir 6

Z nastavitvijo $Y(b) = \beta$ dobimo:

$$u(b) + \vartheta_0 v(b) = \beta$$

Rešitev za pravilen začetni naklon da:

$$\vartheta_0 = \frac{\beta - u(b)}{v(b)}$$

Torej, za linearni BVP lahko najdemo natančno rešitev z le dvema integracijama (eno za $u(x)$ in eno za $v(x)$) in brez iteracije.

Kaj pa, če je NDE **nelinearna**? Na primer: $y'' = f(x, y, y')$. Ali lahko še vedno uporabimo ta trik? [Da — Ne]

Pojdite na Okvir 7.

Okvir 7

Odgovor je [Ne]. Pri nelinearnih NDE načelo superpozicije ne velja. Rešitve ne moremo konstruirati iz linearne kombinacije preprostejših rešitev.

V nelinearnem primeru smo spet pri problemu iskanja ničel. Definiramo "funkcijo odstopanja" $F(\vartheta)$, ki meri, za koliko naša rešitev, integrirana z začetnim naklonom ϑ, zgreši cilj pri $x = b$:

$$F(\vartheta) = y(b)|_\vartheta - \beta$$

Želimo najti vrednost ϑ, za katero je $F(\vartheta) = 0$. To je problem iskanja ničel! Uporabimo lahko katero koli od naših znanih metod, kot sta bisekcija ali sekantna metoda, da iterativno prilagajamo našo oceno za ϑ, dokler ne zadenemo cilja.

Pojdite na Okvir 8.

Okvir 8

Problem lastnih vrednosti

Poseben, a zelo pomemben tip BVP je problem lastnih vrednosti za diferencialno enačbo. Tu enačba vsebuje prost parameter, λ, in iščemo specifične vrednosti λ (lastne vrednosti), za katere obstaja netrivialna rešitev pri homogenih robnih pogojih (npr. $y(a) = y(b) = 0$). Klasičen primer je časovno neodvisna Schrödingerjeva enačba.

$$y''(x) = f(x, y, y', \lambda), \quad \text{z} \quad y(a) = 0, y(b) = 0$$

Kako lahko to rešimo s strelsko metodo? (Namig: Kaj ugibamo in kaj poskušamo zadeti?) _____

Pojdite na Okvir 9 za odgovor.

Okvir 9

To lahko rešimo s strelsko metodo, vendar so vloge sedaj drugačne.

- Poskušamo najti posebno vrednost λ. Torej je λ tisto, kar "ugibamo".
- Začetni pogoji so $y(a) = 0$ in, ker je celotna skala lastne funkcije poljubna, lahko izberemo fiksen začetni naklon, na primer $y'(a) = 1$.
- Nato integriramo NDE z našo oceno za λ do točke $x = b$.
- Čilj", ki ga poskušamo zadeti, je drugi robni pogoj: $y(b) = 0$.

Spet je to problem iskanja ničel. Iščemo ničle λ funkcije $F(\lambda) = y(b)|_\lambda = 0$. Na splošno bo obstajalo več rešitev, ki ustrezajo različnim lastnim vrednostim $\lambda_1, \lambda_2, \ldots$ sistema.

Pojdite na Okvir 10.

Metoda končnih diferenc

Okvir 10

Popolnoma drugačen pristop k reševanju BVP je **metoda končnih diferenc**. Namesto da bi problem pretvorili v IVP, diskretiziramo celoten interval $[a, b]$ na mreži točk Y_j. Bistveni korak je, da vse odvode v NDE nadomestimo z njihovimi aproksimacijami s končnimi diferencami.

Katera je standardna aproksimacija s centralno diferenco za drugi odvod?

$$y_j'' \approx ?$$

Pojdite na Okvir 11 za odgovor.

Okvir 11

Aproksimacija s centralno diferenco za drugi odvod je:

$$y_j'' \approx \frac{Y_{j+1} - 2Y_j + Y_{j-1}}{h^2}$$

Z zamenjavo odvodov v naši prvotni NDE, $y'' = f(x, y, y')$, s temi algebrskimi aproksimacijami, pretvorimo diferencialno enačbo v velik sistem sklopljenih algebrskih enačb za neznane vrednosti Y_j v vsaki točki mreže.

Za linearno NDE to privede do velikega sistema linearnih enačb, ki ga lahko zapišemo v matrični obliki:

$$\mathbf{A} \cdot \vec{Y} = \vec{R}$$

kjer je $\vec{Y}$ vektor neznanih vrednosti funkcije. Matrika $\mathbf{A}$ je običajno **tridiagonalna**, kar omogoča zelo hitro in učinkovito reševanje s specializiranimi linearno-algebrskimi rutinami (kot je Thomasov algoritem).

Pojdite na Okvir 12.

Okvir 12

Kako metoda končnih diferenc obravnava problem lastnih vrednosti, kot je Sturm-Liouvillova enačba?

$$y'' = -Q(x)y - \lambda V(x)y, \quad \text{z} \quad y(a) = y(b) = 0$$

Ko diskretiziramo to enačbo, se pretvori v matrično enačbo oblike:

$$\mathbf{Z} \cdot \vec{Y} = \lambda h^2 \mathbf{V} \cdot \vec{Y}$$

kjer sta $\mathbf{Z}$ in $\mathbf{V}$ tridiagonalni matriki. To je **posplošen matrični problem lastnih vrednosti**.

Problem iskanja lastnih vrednosti *diferencialnega operatorja* smo pretvorili v problem iskanja lastnih vrednosti *matrike*. Sedaj lahko uporabimo močne matrične metode iz prejšnjega poglavja (kot je QR iteracija) za iskanje lastnih vrednosti λ in lastnih vektorjev $\vec{Y}$ (ki so diskretizirane lastne funkcije).

S tem zaključujemo naš pregled metod za reševanje robnih problemov.

Konec poglavja.

9 Problemi začetnih vrednosti za parcialne diferencialne enačbe

Okvir 1

Danes bomo naše znanje z reševanja navadnih diferencialnih enačb (NDE) razširili na reševanje **parcialnih diferencialnih enačb (PDE)**. PDE so diferencialne enačbe, ki vključujejo odvode po več kot eni neodvisni spremenljivki.

Osredotočili se bomo na dva temeljna primera iz fizike:

- Difuzijska enačba
- Valovna enačba

V bistvu bomo reciklirali pristope, ki smo se jih že naučili za NDE. Ključna tema danes bo **stabilnost** naših numeričnih metod, na katero močno vpliva naša izbira velikosti korakov tako v prostoru kot v času.

Pojdite na Okvir 2.

Difuzijska enačba

Okvir 2

Naš prvi primer je **difuzijska enačba**, ki opisuje procese, kot je prevajanje toplote. Želimo najti temperaturno polje, $T(x,t)$, v enodimenzionalni plasti debeline a. Enačba, ki to opisuje, je:

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} + \frac{q}{\rho c}$$

kjer je D difuzijski koeficient in $q(x,t)$ člen, ki predstavlja toplotni vir.

Za rešitev te enačbe potrebujemo ne le enačbo samo, ampak tudi ustrezne začetne in robne pogoje. Kateri trije pogoji so potrebni za določitev enolične rešitve za $T(x,t)$ na domeni $x \in [0,a]$ in $t \geq 0$? 1. _____ 2. _____ 3. _____

Pojdite na Okvir 3 za odgovor.

Okvir 3

Za določitev problema potrebujemo: 1. **Začetni pogoj**: Temperaturni profil pri $t = 0$.

$$T(x, t = 0) = f(x)$$

2. **Robni pogoj** pri $x = 0$:

$$T(x = 0, t) = g_0(t)$$

3. **Robni pogoj** pri $x = a$:

$$T(x = a, t) = g_1(t)$$

Difuzijska enačba ima lastnost, imenovano **princip maksimuma**: pri čisti difuziji (brez virov) se morajo maksimalne in minimalne vrednosti rešitve $T(x,t)$ pojaviti bodisi ob začetnem času ($t = 0$) bodisi na prostorskih robovih ($x = 0$ ali $x = a$). To je fizikalno intuitivno: najbolj vroča/hladna točka v ohlajajočem/segrevajočem se telesu se ne more spontano pojaviti na sredi.

Pojdite na Okvir 4.

Okvir 4

Diskretizacija

Prvi korak pri numeričnem reševanju PDE je **diskretizacija** problemske domene. Ustvarimo mrežo tako v prostoru kot v času.

- Prostorsko domeno $[0, a]$ razdelimo na M korakov velikosti $h = a/M$. Točke mreže so $x_m = m \cdot h$.
- Delamo časovne korake velikosti κ. Točke mreže so $t_n = n \cdot \kappa$.

Naša zvezna funkcija $u(x, t)$ je sedaj predstavljena z njenimi vrednostmi na tej mreži: $u_m^n = u(x_m, t_n)$.

Naslednji korak je zamenjava parcialnih odvodov z aproksimacijami s končnimi diferencami. Kakšna je preprosta, prvega reda aproksimacija za časovni odvod $\partial u/\partial t$? (Namig: Pomislite na najpreprostejši možen način za oceno hitrosti spremembe.)

Pojdite na Okvir 5 za odgovor.

Okvir 5

Najpreprostejša aproksimacija za časovni odvod je **diferenca naprej**:

$$\frac{\partial u}{\partial t} \approx \frac{u_m^{n+1} - u_m^n}{\kappa}$$

Ta ima lokalno napako $O(\kappa)$.

Za prostorski odvod, $\partial^2 u/\partial x^2$, bi morali uporabiti metodo, ki je simetrična in ima boljše lastnosti stabilnosti. Standardna izbira je **aproksimacija s centralno diferenco**, ki smo jo že videli:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2}$$

Ta ima lokalno napako $O(h^2)$.

Pojdite na Okvir 6.

Okvir 6

Z vstavljanjem teh končnih diferenc v našo prvotno PDE, $\partial_t u = D\partial_x^2 u + Q$, pretvorimo PDE v algebrsko **diferenčno enačbo**.

$$\frac{u_m^{n+1} - u_m^n}{\kappa} = D\frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} + Q$$

To je **eksplicitna metoda**, znana kot **FTCS** (Forward Time, Centered Space). Neznano prihodnjo vrednost u_m^{n+1} lahko izrazimo neposredno z znanimi sedanjimi vrednostmi:

$$u_m^{n+1} = u_m^n + r\left(u_{m+1}^n - 2u_m^n + u_{m-1}^n\right) + \kappa Q, \quad \text{kjer je } r = \frac{D\kappa}{h^2}$$

Ta metoda je zelo enostavna za programiranje. Vendar ima veliko slabost. Katera je to? [a] Je računsko zelo draga. [b] Je le pogojno stabilna. [c] Deluje le za linearne enačbe.

Pojdite na Okvir 7.

Okvir 7

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b]. Metoda FTCS je stabilna le, če so velikosti korakov skrbno izbrane. von Neumannova analiza stabilnosti pokaže, da metoda postane nestabilna in "eksplodira", če je parameter r prevelik. Pogoj stabilnosti je:

$$r = \frac{D\kappa}{h^2} \leq \frac{1}{2}$$

To je zelo omejujoč pogoj. Pomeni, da če želite podvojiti prostorsko ločljivost (prepoloviti h), morate časovni korak κ zmanjšati za faktor štiri.

Pojdite na Okvir 8.

Okvir 8

Crank-Nicolsonova metoda

Veliko boljša metoda je **Crank-Nicolsonova metoda**. Dosega boljšo stabilnost, ker je **implicitna metoda**. Prostorski odvod ne aproksimira v trenutnem časovnem koraku n, ampak kot *povprečje* med trenutnim korakom n in prihodnjim korakom $n+1$:

$$\frac{u_m^{n+1} - u_m^n}{\kappa} = \frac{D}{2}\left[\left(\frac{\delta^2 u}{\delta x^2}\right)^{n+1} + \left(\frac{\delta^2 u}{\delta x^2}\right)^{n}\right] + Q$$

(kjer je $\frac{\delta^2 u}{\delta x^2}$ operator centralne diference).

Kaj je glavna slabost takšne implicitne metode? (Namig: Poglejte člene. Ali lahko neposredno izrazite u_m^{n+1}?)

Pojdite na Okvir 9 za odgovor.

Okvir 9

Slabost implicitne metode je, da se neznane prihodnje vrednosti (u^{n+1}) pojavijo na obeh straneh enačbe. Ne moremo jih rešiti eno za drugo.

Namesto tega diferenčna enačba postane **sistem linearnih enačb** za vse neznane točke u_m^{n+1} hkrati. To lahko zapišemo v matrični obliki:

$$\left(\mathbf{I} - \frac{r}{2}\mathbf{A}\right)\vec{u}^{\,n+1} = \left(\mathbf{I} + \frac{r}{2}\mathbf{A}\right)\vec{u}^{\,n} + \vec{b}$$

kjer je $\mathbf{A}$ tridiagonalna matrika, ki predstavlja operator prostorskega odvoda.

Velika prednost Crank-Nicolsonove metode je, da je **brezpogojno stabilna** za katero koli izbiro r. Ima tudi boljšo natančnost reda $O(\kappa^2 + h^2)$.

Pojdite na Okvir 10.

Valovna enačba

Okvir 10

Naš drugi primer je **valovna enačba**, ki opisuje pojave, kot je nihanje strune. Iščemo odmik strune, $u(x,t)$. Enačba je:

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$

To je PDE drugega reda v času. Za rešitev potrebujemo dva začetna pogoja (npr. začetni položaj in začetno hitrost strune) in dva robna pogoja (npr. struna je vpeta na obeh koncih).

To enačbo lahko diskretiziramo na enak način kot difuzijsko enačbo, z uporabo centralnih diferenc tako za časovni kot za prostorski odvod.

$$\frac{u_m^{n+1} - 2u_m^n + u_m^{n-1}}{\kappa^2} = c^2 \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2}$$

To nam da eksplicitno shemo za iskanje prihodnjega odmika u_m^{n+1} na podlagi odklona v prejšnjih dveh časovnih korakih.

Pojdite na Okvir 11.

Okvir 11

Ta eksplicitna metoda končnih diferenc za valovno enačbo je prav tako le pogojno stabilna. Pogoj stabilnosti je znan kot **Courant-Friedrichs-Lewy (CFL) pogoj**:

$$\frac{c\kappa}{h} \leq 1$$

Kakšna je fizikalna interpretacija tega pogoja? (Namig: c je hitrost valovanja, κ je časovni korak in h je prostorski korak).

Pojdite na Okvir 12 za odgovor.

Okvir 12

Courantov pogoj pomeni, da mora numerična domena odvisnosti vsebovati fizikalno domeno odvisnosti. Povedano preprosteje, v enem časovnem koraku κ val ne more potovati dlje kot en prostorski mrežni korak h. Informacija v simulaciji ne more potovati hitreje od fizikalne hitrosti valovanja.

Izkazalo se je, da je ta preprosta metoda končnih diferenc za valovno enačbo tudi simplektična metoda. Zato, ko je stabilna, deluje bolje, kot bi pričakovali, in dobro ohranja energijo v dolgoročnih simulacijah.

S tem zaključujemo naš hiter pregled metod za reševanje parcialnih diferencialnih enačb.

Konec poglavja.

10 Fourierova analiza

Okvir 1

Danes se bomo lotili numeričnega pristopa k **Fourierovim transformacijam (FT)**. Fourierova transformacija je eno najmočnejših orodij v znanosti in tehniki. Uporablja se za spektralno analizo vseh vrst, od svetlobe oddaljenih zvezd do trendov na borzi. Je tudi ključno orodje za reševanje določenih vrst diferencialnih enačb.

Osnovna ideja Fourierove analize je razgradnja funkcije (ali signala) na frekvence, ki jo sestavljajo.

Pojdite na Okvir 2.

Okvir 2

Par Fourierove transformacije

Funkcijo časa, $h(t)$, lahko transformiramo v funkcijo frekvence, $H(f)$. Ta par je definiran s Fourierovo transformacijo in njeno inverzno transformacijo. Uporabili bomo naslednjo konvencijo:

$$\text{FT naprej:} \quad H(f) = \int_{-\infty}^{\infty} h(t) e^{-2\pi i f t} dt$$

$$\text{Inverzna FT:} \quad h(t) = \int_{-\infty}^{\infty} H(f) e^{+2\pi i f t} df$$

Tu je f frekvenca. Upoštevajte, da fiziki pogosto raje delajo s krožno frekvenco $\omega = 2\pi f$. Obstajajo različne konvencije za postavitev faktorja 2π, zato vedno preverite, katera se uporablja.

Kakšna funkcija je $H(f)$ na splošno, tudi če je $h(t)$ povsem realna funkcija? [a] Realna in soda [b] Čisto imaginarna [c] Kompleksna

Pojdite na Okvir 3.

Okvir 3

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [c] Kompleksna. Člen $e^{-2\pi i f t} = \cos(2\pi f t) - i \sin(2\pi f t)$ je kompleksen, zato bo imela dobljena transformacija $H(f)$ na splošno tako realni kot imaginarni del.

Vendar pa ima transformacija $H(f)$ realne funkcije $h(t)$ posebno simetrijo:

$$H(-f) = H^*(f)$$

kjer je H^* kompleksna konjugacija. To pomeni, da je realni del $H(f)$ soda funkcija, imaginarni del pa liha funkcija. Ta lastnost pomeni, da so vse informacije vsebovane v pozitivnih frekvencah; del z negativnimi frekvencami je odvečen.

Druga ključna lastnost je **Parsevalov teorem**, ki povezuje skupno energijo ali moč v časovni domeni z energijo v frekvenčni domeni:

$$\int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(f)|^2 df$$

Pojdite na Okvir 4.

Diskretna Fourierova transformacija (DFT)

Okvir 4

Za numerični izračun Fourierove transformacije moramo delati z diskretnim naborom vzorcev. To vodi do **diskretne Fourierove transformacije (DFT)**.

Začnemo s funkcijo $h(t)$, ki jo vzorčimo v N točkah v skupnem časovnem intervalu T. Časovni korak je $\Delta = T/N$, in naši časi vzorčenja so $t_n = n\Delta$ za $n = 0, 1, \ldots, N-1$. Frekvenca vzorčenja je $\nu_s = 1/\Delta$.

Zvezni integral za FT je nadomeščen s končno vsoto. Kaj je temeljna predpostavka, ki jo implicitno naredimo o naši funkciji $h(t)$, ko to storimo? (Namig: kaj končni vzorec pomeni o obnašanju signala izven okna $[0, T]$?) _____

Pojdite na Okvir 5 za odgovor.

Okvir 5

Z zamenjavo neskončnega integrala s končno vsoto na intervalu $[0, T]$ implicitno predpostavljamo, da je naša funkcija $h(t)$ **periodična** s periodo T. To pomeni, $h(t + T) = h(t)$.

Diskretne frekvence so nato definirane kot večkratniki osnovne frekvence $1/T$:

$$f_k = \frac{k}{T} = \frac{k}{N\Delta}, \quad \text{za } k = 0, 1, \ldots, N-1$$

DFT in njen inverz sta nato definirana kot:

$$\text{DFT:} \quad H_k = \sum_{n=0}^{N-1} h_n e^{-i2\pi kn/N}$$

$$\text{Inverzna DFT:} \quad h_n = \frac{1}{N} \sum_{k=0}^{N-1} H_k e^{+i2\pi kn/N}$$

kjer je $h_n = h(t_n)$ in $H_k \approx H(f_k) \cdot T$. (Opazite normalizacijski faktor $1/N$ v inverzni transformaciji).

Pojdite na Okvir 6.

Okvir 6

Nyquistova frekvenca

Naš diskretni frekvenčni indeks k teče od 0 do $N - 1$. Vendar pa se te frekvence zaradi periodičnosti kompleksnega eksponenta "ovijejo". Frekvenca, ki ustreza indeksu $k = N/2$, je posebna. Imenuje se **Nyquistova frekvenca**, ν_c:

$$\nu_c = \frac{N/2}{T} = \frac{N/2}{N\Delta} = \frac{1}{2\Delta} = \frac{\nu_s}{2}$$

Nyquistova frekvenca je natanko polovica frekvence vzorčenja.

Kaj o tej frekvenci pravi znameniti **Nyquist-Shannonov izrek o vzorčenju**? _____

Pojdite na Okvir 7 za odgovor.

Okvir 7

Izrek o vzorčenju pravi, da če zvezni signal $h(t)$ ne vsebuje frekvenc, višjih od ν_c, ga je mogoče *popolnoma rekonstruirati* iz njegovih diskretnih vzorcev, vzetih s frekvenco $\nu_s = 2\nu_c$.

Z drugimi besedami, če vzorčimo dovolj hitro (vsaj dvakrat hitreje od najvišje frekvence v našem signalu), ne izgubimo nobenih informacij.

Kaj pa se zgodi, če naš signal *vsebuje* frekvence, višje od ν_c? [a] Te frekvence se preprosto izgubijo. [b] Te frekvence se "prepognejo"ali "aliasirajo"v nižji frekvenčni razpon. [c] Transformacija da napako.

Pojdite na Okvir 8.

Okvir 8

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b]. Če signal vsebuje frekvence nad Nyquistovo frekvenco, se te frekvence ne izgubijo, ampak se napačno predstavijo kot nižje frekvence. Ta pojav se imenuje **aliasing**.

Na primer, visokofrekvenčni signal se lahko preslika v nizko frekvenco in se pojavi kot vzorec "utripanja"ali popolnoma drugačen ton. To je razlog za čudne vzorce, ki jih včasih vidite na videoposnetkih z naperami koles ali lopaticami helikopterja.

Da bi preprečili aliasing, moramo bodisi vzorčiti z veliko višjo frekvenco ali pa uporabiti **nizkopasovni filter** na analognem signalu, da odstranimo visoke frekvence, *preden* vzorčimo.

Pojdite na Okvir 9.

Okvir 9

Puščanje (Leakage)

Še en pomemben numerični artefakt izhaja iz naše predpostavke o periodičnosti. Recimo, da je naš pravi signal čisti sinusni val, vendar interval vzorčenja T ne vsebuje natančnega celega števila njegovih ciklov.

Ko DFT predpostavi, da je signal periodičen s periodo T, vidi ostro nezveznost na meji, kjer se konec signala "ovije"in poveže z začetkom.

Kakšen učinek ima ostra nezveznost v časovni domeni na frekvenčni spekter? (Namig: kakšne frekvence so potrebne za predstavitev ostrih značilnosti?) _____

Pojdite na Okvir 10 za odgovor.

Okvir 10

Ostre značilnosti, kot so nezveznosti, zahtevajo zelo širok razpon visokih frekvenc za svojo predstavitev.

Zato moč našega enofrekvenčnega sinusnega vala "uide"v številne druge frekvenčne razrede v DFT. Namesto enega samega ostrega vrha dobimo razširjen vrh z znatnimi stranskimi režnji. To se imenuje **spektralno puščanje**.

Da bi zmanjšali puščanje, bi morali izbrati okno vzorčenja, ki se ujema s periodičnostjo signala, če je to mogoče, ali pa uporabiti "okenske funkcije", ki gladko zmanjšajo signal proti ničli na mejah.

Pojdite na Okvir 11.

Okvir 11

Hitra Fourierova transformacija (FFT)

Neposreden izračun DFT z uporabo vsote zahteva približno N^2 kompleksnih množenj. Za velik N je to računsko zelo drago.

Leta 1965 sta Cooley in Tukey ponovno odkrila zelo učinkovit algoritem za izračun DFT, danes znan kot **hitra Fourierova transformacija (FFT)**. FFT je pameten rekurzivni algoritem, ki razgradi transformacijo velikosti N na manjše transformacije.

Kakšna je računska zahtevnost algoritma FFT? [a] $O(N^2)$ [b] $O(N \log N)$ [c] $O(N)$

Pojdite na Okvir 12.

Okvir 12

Vaš odgovor je bil [a — b — c].

Pravilen odgovor je [b]. Algoritem FFT zmanjša število potrebnih operacij z $O(N^2)$ na $O(N \log N)$. To je ogromno izboljšanje. Za $N = 1.000.000$ je razlika med trilijonom (10^{12}) operacij in približno 20 milijoni ($2 \cdot 10^7$) operacij. FFT je tisto, kar je omogočilo praktično digitalno obdelavo signalov.

FFT rutine so standard v skoraj vseh programskih jezikih in knjižnicah (npr. 'numpy.fft' v Pythonu). Bistveno je uporabljati pomožne funkcije (kot je 'fftshift'), da razporedimo frekvenčne komponente v pravilen, fizikalno intuitiven vrstni red (od $-\nu_c$ do $+\nu_c$).

S tem zaključujemo naš pregled Fourierove analize.

Konec poglavja.