Numerične metode

izročki predavanj

Aljaž Zalar

Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Verzija 09.12.2022

Literatura

Osnovna vira:

- Bojan Orel, Osnove numerične matematike, Založba FE in FRI.
- Bor Plestenjak: Razširjen uvod v numerične metode, DMFA založništvo.

Tuji viri:

- K. Atkinson, W. Han: Elementary Numerical Analysis, 3rd edition, John Wiley & Sons, Inc., New Jersey, 2003.
- R.L. Burden, J.D. Faires, A.M. Burden: Numerical Analysis, 10th edition, Cengage Learning, Boston, 2016.
- G.H. Golub, C.F. Van Loan: Matrix Computations, 3rd edition, Johns Hopkins Univ. Press, Baltimore, 1996.
- D.R. Kincaid, E.W. Cheney: Numerical Analysis, Mathematics of Scientific Computing, 3rd edition, Brooks/Cole, Pacific Grove, 2002.
- L.N. Trefethen, D. Bau: Numerical Linear Algebra, SIAM, Philadelphia, 1997.

Obveznosti

Potek predmeta:

- Predavanja: 3 ure na teden.
- Vaje: 2 uri na teden.

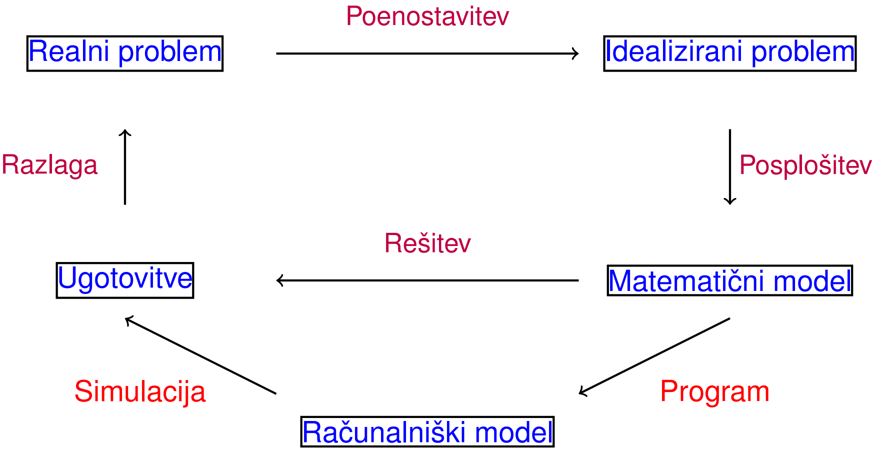
Ocena:

- 3 domače naloge.
- Pisni izpit.
- Ustni izpit.

Programska oprema:

- *Matlab*: Licenca dostopna za študente UL.
- *Octave*: Prosto dostopna alternativa Matlaba.

Vloga numerične matematike



Numerična matematika ima ključno vlogo pri pretvorbi matematičnega modela v računalniškega, reševanju tega modela in razlagi rešitev s stališča napak.

Vsebina predmeta

1. Računanje in vloga napak pri numerični matematiki
2. Reševanje sistemov linearnih enačb
   - Gausova eliminacija in LU razcep - cena in problemi
   - Pivotiranje
   - Iterativne metode - Jacobijeva in Gauss-Seidlova iteracija
3. Reševanje (sistemov) nelinearnih enačb in optimizacija
   - Tangentna oz. Newtonova metoda
   - Metoda fiksne točke
   - Newtonova optimizacijska metoda
4. Aproksimacija in interpolacija
   - Lagrangeov in Newtonov interpolacijski polinom
   - Aproksimacija po metodi najmanjših kvadratov
   - QR razcep za predoločene sisteme

5. Numerično odvajanje in integriranje
    - Trapezna metoda
    - Simpsonova metoda
    - Rombergova metoda
6. Numerično reševanje diferencialnih enačb
    - Eulerjeva metoda
    - Runge-Kutta metode

Prvo poglavje:

Uvod v numerično računanje

- Numerično računanje
- Predstavljiva števila
- Zaokrožitvene napake
- Katastrofalno seštevanje/odštevanje
- Primeri (ne)stabilnega računanja

Numerično in simbolno računanje

Numerično računanje:

- Takoj v formulo vstavljamo števila
- Pridemo do numeričnega rezultata - numerične rešitve

Simbolno računanje:

- simboli predstavljajo števila
- izraz preoblikujemo s simbolnim račuanjem do novega simbolnega izraza - analitična rešitev

Primer

- *Numerično:*

$$\frac{(17.36)^2 - 1}{17.36 + 1} = 16.36; \quad 0.25,\ 0.33333\ldots(?),\ 3.14159\ldots(?)$$

- *Simbolno:*

$$\frac{x^2 - 1}{x + 1} = x - 1; \quad \frac{1}{4},\ \frac{1}{3},\ \pi,\ \tan 83$$

Numerično in simbolno računanje

Primer

```
>> x=rand;(x^2-1)/(x+1)-(x-1)

ans=1.387778780781446e-17
```

Analitično bi rezultat moral biti 0, vendar zaradi numeričnih napak dobimo majhno napako.

Kaj zanima numerično matematiko?

Metoda. . . matematična konstrukcija, s katero rešujemo problem

Algoritem. . . koraki metode

Implementacija. . . zapis algoritma v izbranem jeziku

Kaj pomeni 'biti numerično dober'?

majhna sprememba podatkov $\Rightarrow$ majhna napaka rezultata

Tipična vprašanja numerične matematike:

- Ali je problem občutljiv?
- Ali je metoda 'dobra'?
- Ali je algoritem robusten - deluje na širokem spektru problemov?
- Ali je implementacija hitra - časovna in prostorska zahtevnost?

Občutljivih problemov NM ne more rešiti

Problem je občutljiv, če se ob majhni spremembi začetnih podatkov točen rezultat zelo spremeni.

Občutljivost je odvisna le od narave problema in ne od izbrane numerične metode.

Primer (presečišča premic)

Sistem in njegova perturbacija

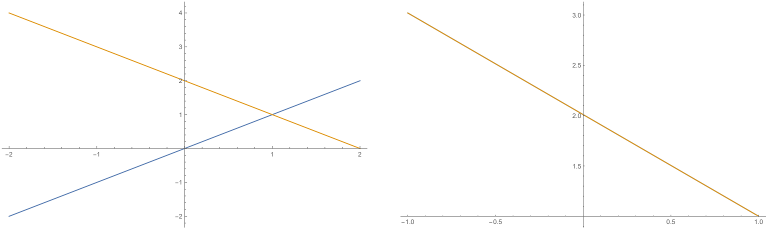
$$
\begin{aligned}
x + y &= 2 \quad \rightarrow \quad x + y = 1.9999 \\
x - y &= 0 \quad \rightarrow \quad x - y = 0.0002
\end{aligned}
$$

ima rešitvi $x = y = 1$ *oz.* $x = 1.00005$ *in* $y = 0.99985$. *Problem je neobčutljiv, saj je šlo za spremembo za isti velikostni razred.*

Sistem in njegova perturbacija

$$\begin{aligned} x + 0.99y &= 1.99 \quad \rightarrow \quad & x + 0.99y &= \color{red}{1.9899} \\ 0.99x + 0.98y &= 1.97 \quad \rightarrow \quad & 0.99x + 0.98y &= \color{red}{1.9701} \end{aligned}$$

ima rešitvi $x = y = 1$ oz. $x = 2.97$ in $y = -0.99$. Problem je občutljiv, saj je majhna sprememba začetnih podatkov povzročila veliko spremembo rezultata.



Na čem temeljijo numerične metode?

- Matrike nadomestimo z enostavnejšimi (upoštevamo samo diagonalni ali zgornjetrikotni del).
- Nelinearne probleme nadomestimo z linearnimi (linearna aproksimacija v točki).
- Neskončne procese nadomestimo s končnimi (uporabimo Taylorjev polinom) .
- Neskončno razsežne prostore nadomestimo s končno razsežnimi (funkcije nadomestimo s polinomi).
- Diferencialne enačbe nadomestimo z algebraičnimi (znebimo se vseh parcialnih odvodov iz enačb).

Zakaj sploh potrebujemo numerično matematiko?

Znanost, ki temelji na matematičnih izračunih, je neposredno odvisna od NM.

Nekatere katastrofe so se zgodile zaradi slabega numeričnega računanja (`http://www-users.math.umn.edu/~arnold//disasters/`):

- *Nesreča Misije Patriot, Zalivska vojna 1991, Savdska Arabija*, 28 žrtev: slaba analiza zaokrožitvenih napak.

  Čas zadetka iraške rakete, usmerjene na Savdsko Arabijo, je bil računan na vsako desetino sekunde v 24-bitnem sistemu. Ker velja

  $$\frac{1}{10} = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + \underbrace{+2^{-24} + 2^{-25} + 2^{-28} + \ldots,}_{\text{zanemarimo}}$$

  je vsako desetinko sekunde napaka $9.5 \cdot 10^{-8}$ s. Po 100 urah računanja je bila napaka $9.5 \cdot 10^{-8}\,\mathrm{s} \cdot 100 \cdot 60 \cdot 60 \cdot 10 = 0.34$ s. Ker je hitrost rakete $1.676\,\mathrm{m/s}$, je bila pozicija rakete za več kot 500 m napačno predvidena in je ta ušla radarjem.

- *Eksplozija rakete Ariana 5, Francoska Gvajana, 1996*: posledica prekoračitve obsega števil.
  https://www.youtube.com/watch?v=PK_yguLapgA
  https://www.youtube.com/watch?v=W3YJeoYgozw

  Ob prenovi rakete so ‘pozabili’ nadgraditi uporabljen številski sistem, ki je horizontalno hitrost meril v 16-bitnem sistemu (1 bit porabimo za predznak). Največja hitrost v tem sistemu je

  $$2^0 + 2^1 + \ldots + 2^{13} + 2^{14} = \frac{2^{15} - 1}{2 - 1} = 32767.$$

  Ker je prenovljena raketa po 37 sekundah presegla to hitrost, je prišlo do zaustavitve motorjev...

- *Potop naftne ploščadi Sleipner A, Stavanger, Norveška, 1991*, miljarda dolarjev škode: nenatančna obdelava obremenitev pri reševanju PDE-jev.
  https://www.youtube.com/watch?v=eGdiPs4THW8

Ponovitev predstavljivih števil

Števila shranjujemo v obliki

$$\boxed{x = \pm 0.d_1 d_2 d_3 \ldots d_m \times \beta^e,}$$

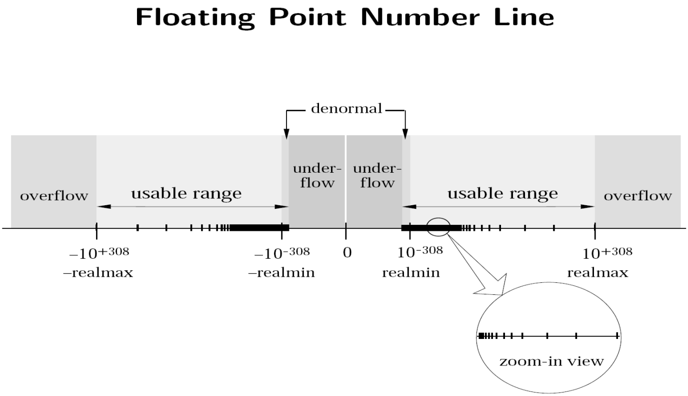
kjer je

- β naravno število (v računalništvu $\beta = 2$),
- $d_1 d_2 d_3 \ldots d_m$ mantisa, e eksponent.

Primer (baza 10)

- *1000.12345 zapišemo kot* $+(0.100012345)_{10} \times 10^4$*.*
- *0.000812345 zapišemo kot* $+(0.812345)_{10} \times 10^{-3}$*.*

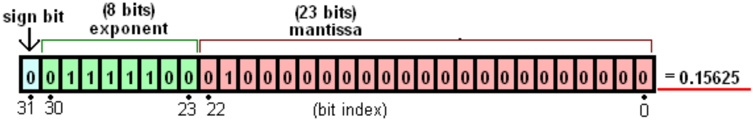
Prekoračitev in podkoračitev

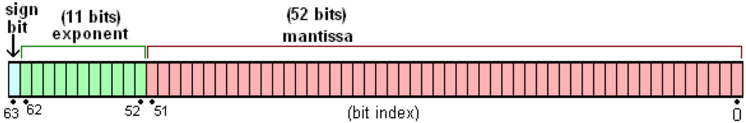


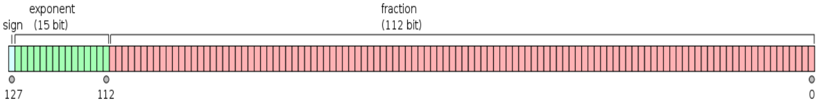
- izračuni preblizu 0 lahko povzročijo podkoračitev
- preveliki izračuni lahko povzročijo prekoračitev
- prekoračitev je v splošnem hujši problem

Različne natančnosti

- *IEEE Enojna natančnost*: števila so predstavljena z 32 biti.
- *IEEE Dvojna natančnost*: števila so predstavljena z 64 biti.
- *Multiprecision Computing Toolbox for MATLAB*: Omogoča računanje v višjih natančnostih. Dostopno na naslovu `https://www.advanpix.com/`







Kaj so zaokrožitvene napake?

- Večine realnih števil ne moremo predstaviti v strojni aritmetiki $\Rightarrow$ zaokrožujemo in delamo zaokrožitvene napake.
- IEEE standard... zaokroži x do najbližjega predstavljivega števila $\mathrm{fl}(x)$. Naj bosta

$$x_- \leqslant x \leqslant x_+$$

najbližji predstavljivi števili števila x. Potem je

$$\boxed{\mathrm{fl}(x) = \begin{cases} x_-, & \text{če je } x \text{ bližje } x_-, \\ x_+, & \text{če je } x \text{ bližje } x_+. \end{cases}}$$

- Kako velika je napaka? Recimo, da je x bližje x_-:

$$\begin{aligned} x &= (0.1b_2b_3\dots b_mb_{m+1})_2 \times 2^e, \\ x_- &= (0.1b_2b_3\dots b_m)_2 \times 2^e, \\ x_+ &= \left((0.1b_2b_3\dots b_m)_2 + 2^{-m}\right) \times 2^e, \end{aligned}$$

$\mathrm{fl}(x) = x(1+\delta)$, $|\delta| < 2^{-m}$

Absolutna napaka:

$$x - x_- \leqslant \frac{x_+ - x_-}{2} = 2^{e-m-1}.$$

Relativna napaka:

$$\frac{x - x_-}{x} \leqslant \frac{2^{e-m-1}}{1/2 \times 2^e} \leqslant \underbrace{2^{-m}}_{u} \ldots$$ osnovna zaokrožitvena napaka

Torej je

$$x_- = x_- - x + x \geqslant -ux + x = x(1-u).$$

Podobno

$$x_+ \leqslant x(1+u).$$

Sledi

$$\boxed{\mathrm{fl}(x) = x(1+\delta)}, \quad \text{kjer je } |\delta| < u.$$

Kako računamo s predstavljivimi števili?

Za predstavljivi števili x, y in katerokoli od osnovnih operacij $\odot \in \{+, -, \cdot, :\}$ število $x \odot y$ ni nujno predstavljivo. Po zgornjem pa velja

$$\boxed{\mathrm{fl}(x \odot y) = (x \odot y)(1 + \delta)}, \quad \text{kjer je } |\delta| \leqslant u.$$

Seštevanje numerično ni asociativna operacija, tj.

$$\boxed{(a + b) + c \neq a + (b + c)}:$$

Primer

```
>> a=rand;b=rand;c=rand;((a+b)+c)-(a+(b+c))

 ans=-2.220446049250313e-16
```

Seštevamo od manjših k večjim številom

$$\begin{aligned}
(a+b)+c &= fl(fl(a+b)+c) = fl((a+b)(1+\delta_1)+c) \\
&= [(a+b)(1+\delta_1)+c]\,(1+\delta_2) \\
&= [(a+b+c)+(a+b)\delta_1)]\,(1+\delta_2) \\
&= (a+b+c)\left[1+\frac{a+b}{a+b+c}\delta_1(1+\delta_2)+\delta_2\right]
\end{aligned}$$

Podobno

$$a+(b+c) = (a+b+c)\left[1+\frac{b+c}{a+b+c}\delta_3(1+\delta_4)+\delta_4\right].$$

Če pozabimo na člena $\delta_1\delta_2$ in $\delta_3\delta_4$ (Zakaj to lahko naredimo?), dobimo

$$\begin{aligned}
(a+b)+c &= (a+b+c)(1+\epsilon_3) \quad \text{kjer je} \quad \epsilon_3 \approx \frac{a+b}{a+b+c}\delta_1+\delta_2, \\
a+(b+c) &= (a+b+c)(1+\epsilon_4) \quad \text{kjer je} \quad \epsilon_4 \approx \frac{b+c}{a+b+c}\delta_3+\delta_4.
\end{aligned}$$

Sklep: Ko seštevamo števila, je za čim manjšo napako najbolje začeti z najmanjšim in prištevati večje.

Napake pri numeričnem računanju

- Neodstranljiva napaka D_n ... nenatančni začetni podatki.
- Napaka metode D_m ... npr. neskončni proces aproksimiramo s končnim.
- Zaokrožitvena napaka D_z ... računanje s približki in zaokroževanje.

Celotna napaka D je

$$\boxed{D = D_n + D_m + D_z}.$$

Stabilnost meri kakovost metode

Stabilnost metode preverimo z analizo zaokrožitvenih napak.

Vrste napak (x naj bo točna vrednost, $\bar{x}$ pa približek zanjo):

- Prva delitev:
  - Absolutna napaka: $\boxed{\bar{x} - x}$.
  - Relativna napaka: $\boxed{\dfrac{\bar{x} - x}{x}}$.
- Druga delitev:
  - Direktna napaka: Numerična napaka rezultata.
  - Obratna napaka: Koliko je potrebno spremeniti začetne podatke, da dobimo izračunan rezultat.

Velja

$$\boxed{|\text{direktna napaka}| \approx \text{občutljivost} \times |\text{obratna napaka}|}.$$

Izračunana vrednost je blizu pravi, če rešujemo neobčutljiv problem z obratno stabilno metode.

Odštevanje in seštevanje sta lahko ‘katastrofalni’

odštevanje dveh približno enakih števil

seštevanje dveh približno nasprotnih števil

$$a = x.xxxx\,xxxx\,xxx1\,\overbrace{ssss\ldots}^{\text{izguba}}$$

$$b = x.xxxx\,xxxx\,xxx0\,\overbrace{tttt\ldots}^{\text{izguba}}$$

Potem

$$\begin{array}{lll} & \overbrace{x.xxx\,xxxx\,xxx1}^{\text{končna natančnost}} & \\ - & x.xxx\,xxxx\,xxx0 & \\ \hline = & 0.000\,0000\,0001 & ????\,???? \\ = & 1.\ \underbrace{????\,????}_{\text{izguba natančnosti}} \cdot \beta^{-m} & \end{array}$$

S ponavljanjem se napake seštevajo.

Primer katastrofalnega odštevanja

Iščemo rešitve kvadratne enačbe

$$x^2 + 2ax + b = 0, \quad \text{kjer je } a > 0 \text{ in } a^2 > b.$$

Rešitev z manjšo absolutno vrednostjo je

$$x_2 = \frac{-2a + \sqrt{4a^2 - 4b}}{2} = -a + \sqrt{a^2 - b}.$$

1 $k_1 := a^2$
2 $k_2 := k_1 - b$
3 $k_3 := \sqrt{k_2}$
4 $k_4 := -a + k_3$

Če je a^2 veliko večji od b, potem ima lahko korak 4 veliko napako. Možna rešitev:

$$x_2 = (-a + \sqrt{a^2 - b}) \cdot \frac{a + \sqrt{a^2 - b}}{a + \sqrt{a^2 - b}} = \frac{-b}{a + \sqrt{a^2 - b}}.$$

$$k_1 := a^2$$
$$k_2 := k_1 - b$$
$$k_3 := \sqrt{k_2}$$
$$k_4 := a + k_3$$
$$k_5 := \frac{-b}{k_4}$$

```
>> a = 10000; b=-1;
>> x = -a+sqrt(a^2 - b)
x = 5.000000055588316e-05

>> x^2 + 2 * a * x +b
ans = 1.361766321927860e-08

>> x = -b/(a+sqrt(a^2-b))
x = 4.999999987500000e-05

>> x^2 + 2 * a * x +b
ans = -9.011402890989895e-17
```

Koda primera: klik

Računanje s stabilnejšo obliko

- Izračun vrednosti funkcije

$$f(x) = x(\sqrt{x+1} - \sqrt{x})$$

ni stabilen za velike x, ker je $\sqrt{x+1} \approx \sqrt{x}$. Tej težavi se lahko izognemo:

$$f(x) = f(x) \cdot \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} = \frac{x}{\sqrt{x+1} + \sqrt{x}}.$$

Koda primera: klik

- Vrsto

$$\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \ldots + \frac{1}{n(n+1)},$$

ki se sešteje v $\frac{n}{n+1}$ (dokaz: indukcija), je bolje numerično računati vzvratno kot

$$\frac{1}{n \cdot (n+1)} + \frac{1}{(n-1) \cdot n} + \ldots + \frac{1}{1 \cdot 2}.$$

Koda primera: klik

- Vrednost integrala $I_n = \int_0^1 x^n e^{-x} \,\mathrm{d}x$ se lahko rekurzivno (integracija per partes) izračuna kot

$$I_n = -\frac{1}{e} + nI_{n-1}, \qquad I_0 = 1 - \frac{1}{e}.$$

Če iz formule izrazimo I_{n-1}, dobimo

$$I_{n-1} = \frac{1}{n} I_n + \frac{1}{ne}.$$

Izkaže se, da je druga formula boljša, pri čemer za začetni približek I_N (pri velikem N) lahko vzamemo karkoli. Zakaj?
Koda primera: klik

Seštevanje in odštevanje v splošnem nista relativno direktno stabilni operaciji

$x, y \in \mathbb{R}$. Računamo približek $\overline{p}$ za $p = x + y$.

$$\begin{aligned}
\overline{p} &= \mathrm{fl}(\mathrm{fl}(x) + \mathrm{fl}(y)) = \mathrm{fl}(x(1+\delta_1) + y(1+\delta_2)) \\
&= (x(1+\delta_1) + y(1+\delta_2))\,(1+\delta_3) \\
&= x(1+\delta_1)(1+\delta_3) + y(1+\delta_2)(1+\delta_3) \\
&= x + y + x(\delta_1 + \delta_3 + \delta_1\delta_3) + y(\delta_2 + \delta_3 + \delta_2\delta_3)
\end{aligned}$$

kjer je $|\delta_i| \leqslant u$, $i = 1, 2, 3$. Relativna napaka je

$$\boxed{\frac{|\overline{p} - p|}{|p|} \leqslant \frac{|x(\delta_1 + \delta_3 + \delta_1\delta_3) + y(\delta_2 + \delta_3 + \delta_2\delta_3)|}{|x + y|}}.$$

Torej:

Če je $x + y$ blizu 0, potem je $\dfrac{|\overline{p} - p|}{|p|}$ veliko.

Množenje (in deljenje) je relativno direktno stabilna operacija

$x, y \in \mathbb{R}$. Računamo približek $\overline{p}$ za $p = x \cdot y$.

$$\begin{aligned}
\overline{p} &= \mathrm{fl}(\mathrm{fl}(x) \cdot \mathrm{fl}(y)) = \mathrm{fl}(x(1+\delta_1) \cdot y(1+\delta_2)) \\
&= x(1+\delta_1) \cdot y(1+\delta_2)(1+\delta_3) \\
&= xy(1+\delta_1+\delta_2+\delta_3+\text{produkti več } \delta),
\end{aligned}$$

kjer je $|\delta_i| \leqslant u$, $i = 1, 2, 3$. Relativna napaka je

$$\boxed{\frac{|\overline{p}-p|}{|p|} \leqslant \frac{|xy||\delta_1+\delta_2+\delta_3+\mathcal{O}(u^2)|}{|xy|} = |\delta_1+\delta_2+\delta_3+\mathcal{O}(u^2)|}.$$

Torej:

Relativna napaka $\dfrac{|\overline{p}-p|}{|p|}$ ni odvisna od velikosti produkta xy.

Večina numeričnih metod ni relativno direktno stabilnih

Vse numerične metode, kjer sta vključeni

$$\text{operaciji } + / -$$

in kot rezultat lahko dobimo npr. vrednost 0 ali nekje po poti kot vmesno vrednost skoraj singularno matriko, niso relativno direktno stabilne, tj. v rezultatu je lahko veliko relativna napaka.

Zato moramo vedno premisliti:

1. V katerih primerih so zgodi velika napaka?
2. Kako nestabilne primere preoblikovati v stabilne?

Primeri takih operacij:

- Računanje vrednosti polinoma.
- Računanje skalarnega produkta.
- Reševanje linearnega sistema.
- $\vdots$

Drugo poglavje:

Linearni sistemi

$$Ax = b$$

- Direktne metode za reševanje
  - *LU* razcep
  - Pivotna rast $\rho(A)$
- Iterativne metode za reševanje
  - Jacobi, Gauss-Seidel, SOR, SSOR, konjugirani gradienti

Direktne metode

$$Ax = b$$

- Gaussova-eliminacija
- *LU* razcep
- Pivotiranje
- Pivotna rast

Reševanje kvadratnih linearnih sistemov

Linearni sistem n enačb z n neznankami $x_1, \ldots, x_n$ je oblike

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ \vdots \quad &\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n, \end{aligned}$$

kjer so a_{ij}, b_j realna števila.

V matrični obliki ga zapišemo kot

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \ldots & \ldots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b}.$$

Geometrijski pomen sistema $Ax = b$

Naj bodo $a_{(1)}, a_{(2)}, \ldots, a_{(n)}$ stolpci matrike A, tj.,

$$a_{(i)} := \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{ni} \end{bmatrix} \in \mathbb{R}^n$$

Linearna kombinacija vektorjev $a_{(1)}, a_{(2)}, \ldots, a_{(n)}$ je vsak vektor oblike

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{bmatrix}, \tag{1}$$

kjer so $x_i \in \mathbb{R}$ realna števila.

Zanima nas, ali obstaja linearna kombinacija (1), ki je enaka vektorju b.

Sistem $Ax = b$ z vidika numerične matematike

- Kako drago je reševanje sistema $Ax = b$?

  cena=število osnovnih računskih operacij $(+, -, \cdot, :)$.

- Kateri problemi in napake se pojavijo med reševanjem $Ax = b$?

  Ali obstajajo slabe matrike? Kako take matrike identificirati?

- Za katere matrike se da enostavno in poceni rešiti tak sistem?

Ponovitev Gaussove eliminacije (GE)

Cilj je pretvoriti sistem v zgornjetrikotnega, nato pa ga rešiti z obratno substitucijo.

Primer

Rešujemo $Ax = b$, kjer sta

$$A = \begin{bmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} -1 \\ -7 \\ -6 \end{bmatrix}.$$

Tvorimo razširjen sistem

$$\widetilde{A} = [\ A \mid b\ ] = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{array}\right]$$

Prištejemo 2-kratnik prve vrstice drugi in 1-kratnik prve vrstice tretji.

$$\widetilde{A}_{(1)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{array}\right]$$

Primer

Odštejemo 1-kratnik druge vrstice od tretje

$$\widetilde{A}_{(2)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{array}\right]$$

Rešimo z obratno substitucijo

$$\begin{aligned} x_3 &= \frac{2}{-2} = -1, \\ x_2 &= \frac{1}{-2}\left(-9 - 5x_3\right) = 2, \\ x_1 &= \frac{1}{-3}\left(-1 - 2x_2 + x_3\right) = 2. \end{aligned}$$

V nadaljevanju bomo:

1. Prešteli število potrebnih računskih operacij za Gaussovo eliminacijo (GE).
2. GE bomo zapisali s pomočjo matričnih množenj.
3. Ukvarjali se bomo s stabilnostjo GE.

Algoritem GE in cena GE

```
-n × n matrika A = [a_ij]_ij in n × 1 vektor b = [b_i]_i
-preoblikujemo [A|b] v zgornjetrikotno z GE

for k = 1...n-1
   for i = k+1...n
      xmult = a_ik / a_kk
      a_ik = 0
      for j = k+1...n
         a_ij = a_ij - (xmult) a_kj
      end
      b_i = b_i - (xmult) b_k
   end
end
```

Izrek

Število računskih operacij $(+, -, \cdot, :)$ za prevedbo matrike A in razširjene matrike $[A|b]$ v zgornjetrikotno obliko je

$$\frac{2}{3}n^3 + \mathcal{O}(n^2).$$

Obratna substitucija in število operacij

```
-zgornjetrikotna $n \times n$ matrika $U = [u_{ij}]_{i,j}$, vektor
    $c = [c_i]_i$
-resimo sistem $Ux = c$

$x_n = c_n / u_{nn}$
for $i = n-1 \ldots 1$
  $s = c_i$
  for $j = i+1 \ldots n$
    $s = s - u_{ij} x_j$
  end
  $x_i = s / u_{ii}$
end
```

Izrek

Število računskih operacij $(+, -, \cdot, :)$ za rešitev sistem $Ux = c$ je

$$n^2.$$

Motivacija za zapis GE v matrični obliki

Videli smo, da je cena pretvorba matrike A oz. sistema $[A|b]$ v zgornjetrikotno obliko bistveno dražja kot pa obratna substitucija.

Če bomo v nekem postoku reševali sisteme $Ax = b$ pri fiksni matriki A, vektor b pa se bo spreminjal, bi bilo iz računskega vidika bistveno učinkoviteje preoblikovanje matrike A v zgornjetrikotno obliko narediti samo enkrat.

Ključno v tem procesu je ugotoviti, kako moramo preblikovati vektor b, ne da bi delali GE na razširjenem sistemu.

LU razcep matrike A

```
-Vhod: $A = [a_{ij}]_{i,j}$ $n \times n$ matrika.
-Izhod: Spodnja trikotna matrika $L$ in zgornja
    trikotna matrika $U$, da je $A = LU$
-$\ell_{ik}$ v spodnjem algoritmu so elementi pod
    diagonalo v $L$, na diagonali so same 1
-preostali elementi $a_{ij}$ v zgornjem trikotniku so
    elementi matrike $U$

for $k = 1, \ldots, n-1$
  for $i = k+1, \ldots, n$
    $\ell_{ik} = a_{ik}/a_{kk}$
    for $j = k+1, \ldots, n$
      $a_{ij} = a_{ij} - \ell_{ik} a_{kj}$
    end
  end
end
```

Izrek

Število računskih operacij $(+, -, \cdot, :)$ za izračun LU razcepa matrike A je $\frac{2}{3}n^3 + \mathcal{O}(n^2)$.

Prema substitucija in število operacij

```
-Vhod: spodnja trikotna n × n matrika L = [ℓ_ij]_{i,j} in
    vektor b = [b_i]_i
-Izhod: resitev y sistema Ly = b

y_1 = b_1/ℓ_11
for i = 2...n
  s = b_i
  for j = 1...i-1
    s = s - ℓ_ij y_j
  end
  y_i = s/ℓ_ii
end
```

Izrek

Število računskih operacij $(+, -, \cdot, :)$ za rešitev sistem $Ly = b$ je

$$n^2.$$

Reševanje sistema $Ax = b$ prek LU razcepa:

1. Izračunamo $A = LU$. Cena: $\frac{2}{3}n^3 + \mathcal{O}(n^2)$.
2. Rešimo $Ly = b$ s premo subsitucijo, tj. od y_1 proti y_n. Cena: $n^2 - n$.
3. Rešimo $Ux = y$ z obratno subsitucijo, t. od x_n proti x_1. Cena: n^2.

Cena preme substitucije je za n operacij manjša kot cena obratne substitucije, saj imamo ne diagonali L same enice in prihranimo v vsaki spremenljivki eno deljenje.

Reševanje sistema $Ax = b$ prek LU razcepa

Primer

$$A = \begin{pmatrix} 2 & 1 & 3 & -4 \\ -4 & -1 & -4 & 7 \\ 2 & 3 & 5 & -3 \\ -2 & -2 & -7 & 9 \end{pmatrix}, \quad b = \begin{pmatrix} 8 \\ -14 \\ 7 \\ -16 \end{pmatrix}.$$

1. $L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ -1 & -1 & 1 & 1 \end{pmatrix}, U = \begin{pmatrix} 2 & 1 & 3 & -4 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & -2 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$
2. *Rešimo $Ly = b$ in dobimo $y = \begin{pmatrix} 8 & 2 & -5 & -1 \end{pmatrix}^T$.*
3. *Rešimo $Ux = y$ in dobimo $x = \begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix}^T$.*

LU razcep brez pivotiranja: koda
Prema substitucija: koda
Obratna substitucija: koda

Primer: koda

Obstoj LU razcepa matrike

V nadaljevanju se bomo ukvarjali z obstojem in stabilnostjo *LU* razcepa.

Problematična sta npr. matriki

$$A = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad B = \begin{bmatrix} 10^{-17} & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

saj je 10^{-17} pod strojnim ϵ. Da pa se natančno povedati, kdaj *LU* razcep obstaja.

Podmatriki matrike $A \in \mathbb{R}^{n \times n}$, zožene na prvih k vrstic in stolpcev, pravimo k–ta glavna vodilna podmatrika.

Izrek (Obstoj LU razcepa)

Za $n \times n$ matriko A sta naslednji trditvi ekvivalentni:

1. *LU razcep matrike A obstaja in je enoličen.*
2. *k-ta glavna vodilna podmatrika matrike A je obrnljiva za vsak $k = 1, \dots, n$.*

LU razcep z delnim pivotiranjem

Pri delnem pivotiranju pred eliminacijo v j-tem stolpcu primerjamo elemente

$$a_{jj}, a_{j+1,j}, \ldots, a_{nj},$$

nato pa zamenjamo j-to vrstico s tisto, ki vsebuje element z največjo absolutno vrednostjo.

Menjava j-te in k-te vrstice pa je množenje z leve s permutacijsko matriko P_{jk}, ki se od identitete razlikuje le v j-ti in k-ti vrstici, ki sta zamenjani:

$$P_{jk} = I_n - E_{jj} - E_{kk} + E_{jk} + E_{kj}.$$

Tu so E_{ij} standardne koordinatne matrike (1 v i-ti vrstici in j-tem stolpcu in 0 drugje).

LU razcep z delnim pivotiranjem - algoritem

```
-Vhod: A = [a_ij]_{i,j}  n × n matrika
-Izhod: permutacijska matrika P, spodnja in
    zgornja trikotna matrika L in U, da je
    PA = LU

P in L identicni n × n matriki
for k = 1,...,n-1
  poisci q-to in k-to vrstico, ki zadosca
      |a_qk| = max_{k ≤ p ≤ n} |a_pk|
  q-to in k-to vrstico v matrikah A, P in strogem
      spodnjem trikotniku L
  for i = k+1,...,n
    ℓ_ik = a_ik / a_kk
    for j = k+1,...,n
      a_ij = a_ij - ℓ_ik a_kj
      end
  end
 end
```

LU razcep z delnim pivotiranjem

Izrek (O računski zahtevnosti LU razcep z delnim pivotiranjem)

Število računskih operacij $(+, -, \cdot, :)$ *za izračun LU razcepa z delnim pivotiranjem je* $\frac{2}{3}n^3 + \mathcal{O}(n^2)$.

Dodatno delo pri LU razcepu z delnim pivotiranjem je $\mathcal{O}(n^2)$ primerjanj in menjav.

Reševanje $Ax = b$ prek LU razcepa z delnim pivotiranjem:

1. Izračunamo $PA = LU$. Cena: $\frac{2}{3}n^3 + \mathcal{O}(n^2)$.
2. Rešimo $Ly = Pb$ s premo subsitucijo. Cena: $n^2 - n$.
3. Rešimo $Ux = y$ z obratno subsitucijo. Cena: n^2.

Izrek (Obstoj LU razcepa z delnim pivotiranjem)

Za $n \times n$ *matriko* A *sta naslednji trditvi ekvivalentni:*

1. *LU razcep matrike* A *z delnim pivotiranjem obstaja.*
2. *Matrika* A *je obrnljiva.*

$Ax = b$ prek LU razcepa z delnim pivotiranjem

Primer.

$$A = \begin{pmatrix} 2 & 1 & 3 & -4 \\ -4 & -1 & -4 & 7 \\ 2 & 3 & 5 & -3 \\ -2 & -2 & -7 & 9 \end{pmatrix}, \quad b = \begin{pmatrix} 8 \\ -14 \\ 7 \\ -16 \end{pmatrix}.$$

1. $L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & -\frac{3}{5} & 1 & 0 \\ -\frac{1}{2} & \frac{1}{5} & -\frac{1}{8} & 1 \end{pmatrix}, U = \begin{pmatrix} -4 & -1 & -4 & 7 \\ 0 & \frac{5}{2} & 3 & \frac{1}{2} \\ 0 & 0 & -\frac{16}{5} & \frac{58}{10} \\ 0 & 0 & 0 & \frac{1}{8} \end{pmatrix},$

   $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$
2. Rešimo $Ly = Pb$ in dobimo $y = \begin{pmatrix} -14 & 0 & -9 & -\frac{1}{8} \end{pmatrix}^T$.
3. Rešimo $Ux = y$ in dobimo $x = \begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix}^T$.

LU razcep z delnim pivotiranjem: koda

Primer: koda

LU s kompletnim pivotiranjem

Pri kompletnem pivotiranju pred eliminacijo v j-tem stolpcu poiščemo element z največjo absolutno vrednostjo v podmatriki $A(j:n, j:n)$ in nato izvedemo ustrezni menjavi vrstic in stolpcev.

Dodatno delo pri LU razcepu s kompletnim pivotiranjem je $\mathcal{O}(n^3)$ primerjanj in menjav. Torej je skupna cena precej dražja od LU razcepa z delnim pivotiranjem. Ker bomo videli, da je LU razcep z delnim pivotiranjem statistično numerično stabilen, se v praksi kompletno pivotiranje redko uporablja.

Stabilnost LU razcepa matrike A

Sistem $Ax = b$ smo rešili prek LU razcepa in dobili približek $\widehat{x}$. Računali smo v treh korakih:

1. *Izračun LU razcepa:* $A + E = \widehat{L}\widehat{U}$.
2. *Prema substitucija:* $\widehat{L}\widehat{y} = b$.
3. *Obratna substitucija:* $\widehat{U}\widehat{x} = \widehat{y}$.

Izkaže se, da je (teoretično) nestabilen samo prvi korak.

Spomnimo se, da z u označujemo osnovno zaokrožitveno napako 2^{-m} kjer je m dolžina mantise. Z $|A| = [|a_{ij}|]_{i,j}$ označimo matriko absolutnih vrednosti vhodov matrike $A = [a_{ij}]_{i,j}$

Izrek (Ocena absolutne napake pri izračunu LU razcepa)

Naj bo $A \in \mathbb{R}^{n\times n}$ obrnljiva matrika, pri kateri se izvede LU razcep brez pivotiranja. Za izračunani matriki $\widehat{L}$, $\widehat{U}$ velja $A = \widehat{L}\widehat{U} + E$, kjer je

$$|E| \leqslant 3(n-1)u\left(|A| + |\widehat{L}||\widehat{U}|\right) + \mathcal{O}(u^2).$$

Stabilnost LU razcepa matrike A

Označimo z $\|X\|_\infty$ največjo vsoto absolutnih vrednosti neke vrstice matrike X.

Izrek (Ocena relativne napake pri izračunu LU razcepa)
Pri LU razcepu z delnim pivotiranjem velja ocena relativne napake:

$$\frac{\|E\|_\infty}{\|A\|_\infty} \leqslant 3(n-1)u + 3(n-1)nu \cdot \frac{\|\widehat{U}\|_\infty}{\|A\|_\infty} + \mathcal{O}(u^2).$$

Pivotna rast

Pivotna rast matrike A je definirana kot

$$\rho(A) := \frac{\max_{i,j} |\widehat{u}_{i,j}|}{\max_{i,j} |a_{i,j}|}.$$

Velja

$$\|\widehat{U}\|_\infty \leqslant n\rho(A)\|A\|_\infty.$$

Trditev

Pri delnem pivotiranju je pivotna rast omejena z 2^{n-1}.

Dokaz. Velja namreč $|\ell_{ij}| \leqslant 1$, a_{ij} pa na vsakem od največ $n-1$ korakov izračunamo kot

$$a_{ij} = a_{ij} - \ell_{ik} a_{kj}.$$

Torej se absolutna vrednost največjega elementa v matriki kvečjemu podvoji.

Pivotna rast pri delnem pivotiranju

Žal pa za vsak n obstajajo matrike s pivotno rastjo 2^{n-1}, tako da LU razcep z delnim pivotiranjem teoretično ni stabilen.

Primer

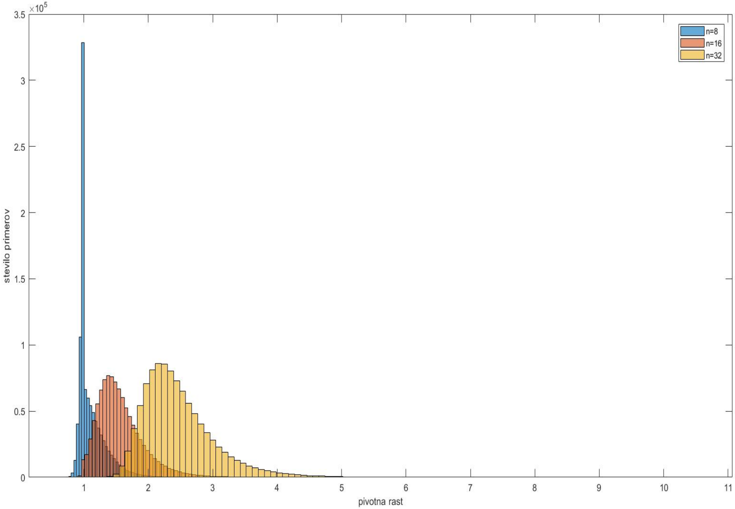
Matrika

$$A_n = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & 1 & \ddots & \vdots & 1 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ -1 & \cdots & \cdots & -1 & 1 \end{pmatrix}$$

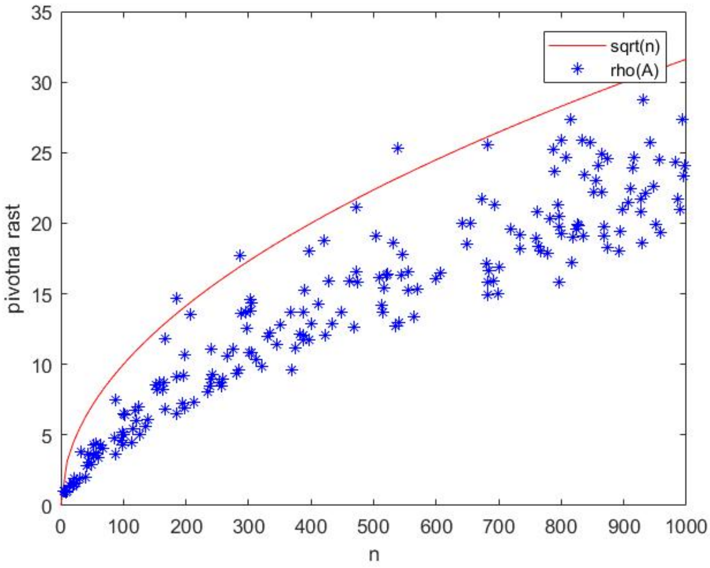
ima pivotno rast 2^{n-1}.

Statistično pa velja, da je pričakovana vrednost pivotne rasti $\mathcal{O}(n^{2/3})$, tako da LU razcep z delnim pivotiranjem v praksi je obratno stabilen.

Verjetnostne porazdelitve slučajne spremenljivke ρ, generirane z milijon naključnimi matrikami velikosti $n \times n$ (tj. vsak vhod naključen element iz enakomerne zvezne porazdelitve na intervalu $[0, 1]$):



Pivotna rast 200 naključnih matrik velikosti $n \times n$ (tj. vsak vhod naključen element iz enakomerne zvezne porazdelitve na intervalu $[0, 1]$):



Iterativne metode

$$Ax = b$$

- Jacobijeva iteracija
- Gauss-Seidlova iteracija
- SOR iteracija

- Veliko bolj specialnih metod (ki jih ne bomo obravnavali): Pospešitev Čebiševa, SSOR, Metode podprostorov Krilova, Metoda konjugiranih gradientov, Hitra Fourierova transformacija, Ciklična redukcija, Večmrežna metoda.

Iterativne metode za reševanje $Ax = b$

Doslej smo iskali točno rešitev x^* sistema

$$Ax = b. \tag{2}$$

Odslej nas bodo zanimali samo približki $\hat{x}$ točnih rešitev x^*.

Naprej si bomo izbrali $\epsilon > 0$ in iskali $\hat{x}$, ki zadošča pogoju

$$\|\hat{x} - x^*\| \leqslant \epsilon.$$

Prednosti iterativnih metod pred direktnimi:

- Če je matrika A velika in ima veliko ničel, je bolje uporabiti iterativne metode.
- Ko je rezultat znotraj vnaprej predpisane natančnosti, lahko končamo računanje. Pri direktnih metodah tega vpliva nimamo.

Recimo, da ugibamo, kaj bi lahko bila prava rešitev sistema (2)

$$x^{(0)} \approx x$$

Kako izboljšati $x^{(0)}$?

Idealno bi prišteli pravi razliko:

$$x^{(1)} = x^{(0)} + (x^* - x^{(0)}),$$

kar lahko drugače zapišemo kot

$$\begin{aligned} x^{(1)} &= x^{(0)} + (x^* - x^{(0)}) \\ &= x^{(0)} + (A^{-1}b - x^{(0)}) \\ &= x^{(0)} + A^{-1}\underbrace{(b - Ax^{(0)})}_{r^{(0)}}. \end{aligned}$$

Toda ta metoda ni smiselna, saj bi morali izračunati A^{-1}.

Kaj pa, če bi znali aproksimariti A^{-1}?

Recimo, da je približek

$$Q^{-1} \approx A^{-1}$$

poceni za izračunati. Potem izračunamo

$$x^{(1)} = x^{(0)} + Q^{-1} r^{(0)}.$$

Nadaljujemo z $k = 2, 3, \dots$

$$x^{(k)} = x^{(k-1)} + Q^{-1} \underbrace{(b - Ax^{(k-1)})}_{r^{(k-1)}}. \qquad (3)$$

Algoritem iterativnih metod

```
A je dana n × n matrika, ki jo aproksimiramo z matriko Q, in n × 1 vektor b.
Izberi zacetni priblizek x = x^(0), toleranco dovoljene relativne napake tol in maksimalno stevilo k_max korakov iteracije.

x^(nov) = ∞
for k = 1 to k_max
    r = b − Ax
    if ||x^(nov) − x|| / ||x|| ≤ tol, stop
    else
        x = x^(nov)
        x^(nov) = x + Q^(−1) r
end
x = x^(nov)
```

Jacobijeva iteracija

Aproksimiramo $A = [a_{ij}]_{i,j}$ z diagonalno matriko

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}.$$

Če pišemo

$$A = S + D + Z, \tag{4}$$

kjer je S strogo spodnjetrikotna matrika in Z strogo zgornjetrikotna matrika, potem (3) postane

$$\begin{aligned} x^{(k)} &= x^{(k-1)} + D^{-1}(b - Sx^{(k-1)} - Dx^{(k-1)} - Zx^{(k-1)}) \\ &= x^{(k-1)} + D^{-1}b - D^{-1}Sx^{(k-1)} - x^{(k-1)} - D^{-1}Zx^{(k-1)} \\ &= D^{-1}(b - Sx^{(k-1)} - Zx^{(k-1)}). \end{aligned} \tag{5}$$

Pišimo

$$x^{(j)} = \begin{pmatrix} x_1^{(j)} & x_2^{(j)} & \cdots & x_n^{(j)} \end{pmatrix}^T$$

Po komponentah (5) pomeni

$$\boxed{x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k-1)}\right)}. \tag{6}$$

Torej vsak preskok (iz $k-1$ na k) potrebuje $O(n)$ operacij za vsak element novega vektorja.

Računska zahtevnost: Če je v vsaki vrstici vsi razen največ m koeficientov a_{ij} neničelnih, potem za vsak korak iteracije potrebujemo $O(mn)$ operacij.

Koda algorima: klik Primer 1: klik Primer 2: klik

Gauss-Seidlova iteracija

Naj bo $A = [a_{ij}]_{i,j} = S + D + Z$ kot v (4). A aproksimiramo s spodnjetrikotno matriko

$$S + D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & \cdots & a_{n,n-1} & a_{nn} \end{pmatrix}.$$

Potem (3) postane

$$\begin{aligned} x^{(k)} &= x^{(k-1)} + (D+S)^{-1}(b - (S+D)x^{(k-1)} - Zx^{(k-1)}) \\ &= x^{(k-1)} + (D+S)^{-1}b - x^{(k-1)} - (D+S)^{-1}Zx^{(k-1)} \\ &= (D+S)^{-1}(b - Zx^{(k-1)}), \end{aligned} \tag{7}$$

Pomnožimo (7) z leve z $D + S$ in dobimo

$$(D+S)x^{(k)} = b - Zx^{(k-1)}. \tag{8}$$

Odštejemo $Sx^{(k)}$ od obeh strani (8) in dobimo

$$Dx^{(k)} = b - Zx^{(k-1)} - Sx^{(k)}$$

oz.

$$x^{(k)} = D^{-1}(b - Zx^{(k-1)} - Sx^{(k)}). \tag{9}$$

Po komponentah (9) pomeni

$$\boxed{x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j<i}^{n} a_{ij}x_j^{(k)} - \sum_{j=1, j>i}^{n} a_{ij}x_j^{(k-1)}\right)}. \tag{10}$$

Računska zahtevnost: Če je v vsaki vrstici vsi razen največ m koeficientov a_{ij} neničelnih, potem za vsak korak iteracije potrebujemo $O(mn)$ operacij.

Koda algorima: klik   Primer 1: klik   Primer 2: klik

Razlika v primerjavi z Jacobijevo metodo je ta, da so popravki shranjeni v obstoječem vektorju in ne potrebujemo še enega dodatnega vektorja. Tako pridobimo precej prihranka v spominu.

SOR iteracija

Ideja ekstrapolirana Gauss-Seidlove iteracija ali SOR(w), kjer je $w \in \mathbb{R}$ relaksacijski parameter, je pospešiti GS–iteracijo tako, da nov približek računamo kot uteženo povprečje

$$x_i^{(k)} = (1-w)x_i^{(k-1)} + wx_i^{(k)},$$

pri čemer $x_i^{(k)}$ na desni strani enačbe izračunamo iz predpisa za GS–iteracijo:

$$x_i^{(k)} = (1-w)x_i^{(k-1)} + \frac{w}{a_{ii}}\Big(b_i - \sum_{j<i}^{n} a_{ij}x_j^{(k)} - \sum_{j>i} a_{ij}x_j^{(k-1)}\Big).$$

Koda algorima: klik   Primer 1: klik   Primer 2: klik

Konvergenčni kriteriji

Matrika je krepko vrstično diagonalno dominantna (kVDD), če za vsak i velja

$$|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}|.$$

Diagonalno dominantne matrike velikokrat nastopajo pri reševanju parcialnih diferencialnih enačb z metodo diskretizacije.

Nekaj konvergenčnih rezultatov:

1. Jacobijeva in GS iteracija za kVDD matrike vedno konvergirata, ne glede na izbiro začetnega približka $x^{(0)}$. GS je hitrejša.
2. V primeru enakosti v kVDD pogojih za konvergenco Jacobija in GS potrebujemo dodatno lastnost matrike A - nerazcepnost.
3. Potreben pogoj za konvergenco $SOR(w)$ je $0 < w < 2$. Za pozitivno definitno matriko A je pogoj tudi zadosten.
4. Primerjavo hitrosti konvergence metod lahko naredimo za matrike A, katerih graf sosednosti je dvodelen. Hkrati lahko za take s formulo določimo optimalen w v $SOR(w)$.

Reševanje nelinearnih enačb in optimizacija

$$* \; f(x) = 0$$

$$* \; f_i(x_1, x_2, \ldots, x_n) = 0, \quad {\scriptstyle i=1,\ldots,n}$$

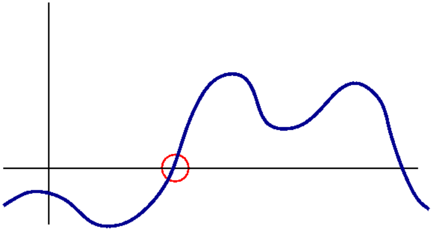
$$* \; \min\{f(x) : x \in K \subseteq \mathbb{R}^n\}$$

- Ena enačba v eni spremenljivki: Bisekcija, tangentna metoda, sekantna metoda, regula falsi, navadna iteracija
- Sistem n enačb v n spremenljivkah: Newtonova metoda, Broydenova metoda
- Optimizacija: Gradientni spust

Motivacija

Problem: Naj bo dana funkcija $f(x)$. Poišči x, ki zadošča

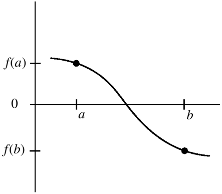
$$f(x) = 0.$$

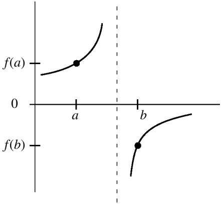


- Nelinearni sistemi niso tako enostavno rešljivi kot sistemi linearnih enačb.
- Ničel polinoma stopnje 5 ne moremo zapisati analitično.
- Kako reševati take probleme? Z iterativnim postopkom, pri čemer se rešitvam čim bolj približamo.

Osnovna strategija reševanja

1. Skiciraj funkcijo.
   - Postavimo začetno domnevo, kaj je lahko ničla.
   - Ničla x gotovo obstaja na intervalu $[a, b]$, če imata $f(a)$ in $f(b)$ različna predznaka in je funkcija f zvezna na $[a, b]$.
   - Toda: Sprememba predznaka funkcije ne pomeni vedno, da je na tem intervalu ničle, kajti lahko imamo na intervalu singularnost:





2. Začnemo z začetno domnevo in uporabimo nek iteracijski algoritem.

Konvergenčni kriteriji za x

Zaustavitveni kriterij je odvisen od narave problema, ki ga rešujemo:

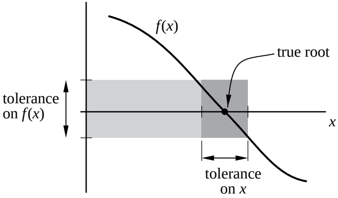
- Lahko nas zanima, kdaj velja

$$|x_k - x_{k-1}| < \text{toleranca}.$$

- Lahko pa nas zanima, kdaj velja

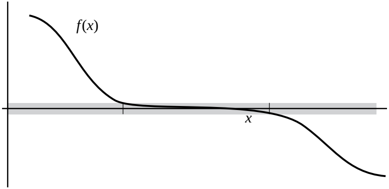
$$|f(x_k)| < \text{toleranca}.$$

- Še najbolje pa je zahtevati izpolnjenost obeh pogojev hkrati.

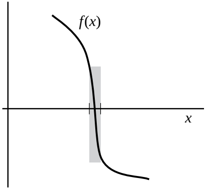


Primerjava obeh konvergenčnih kriterijev

Če je $f'(x)$ majhen v okolici ničle, je lažje zadostiti toleranci na funkcijsko vrednost.



Če je $f'(x)$ velik v bližini ničle, je možno zadostiti toleranci na dolžino intervala, četudi je $|f(x)|$ še vedno velik.



Povezava med obema kriterijama

Vprašanje: Kako sta kriterija na x in $f(x)$ povezana med sabo?

Ko x_a in x_b konvergirata proti x^*, gre razmerje

$$\frac{f(x_b) - f(x_a)}{x_b - x_a} \quad \text{proti} \quad f'(x^*)$$

Zato lahko pričakujemo, da velja

$$|f(x_b) - f(x_a)| \approx |f'(x^*)||x_b - x_a|,$$

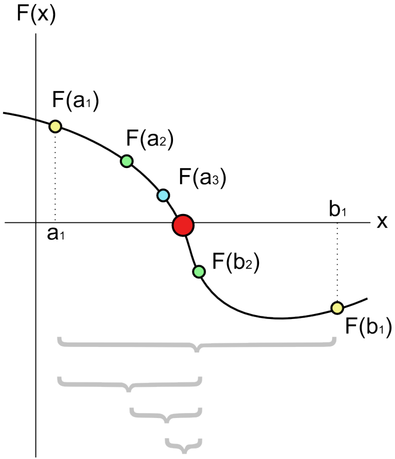
ko x_a in x_b konvergirata proti x^*.

Zaključek: $|f'(x^*)|$ določa povezavo med kriterijema.

Bisekcija

Razpolovišče začetnega intervala $[a, b]$ je točka

$$x_m = \frac{1}{2}(a + b).$$



Postopek:

1. Poišči razpolovišče.
2. Izmed dveh možnih intervalov izberi tistega, kjer ima funkcija različno predznačeni krajišči.
3. Nadaljujemo s prvim korakom.
4. Ustavimo se, ko je interval krajši od naprej predpisane tolerance.

Algoritem za bisekcijo

```
zacetni podatki:  f, a, b, tol
for k = 1, 2, ...
   x_m = a + (b - a)/2
   if sign(f(x_m)) = sign(f(a))
      a = x_m
   else
      b = x_m
   end
   if |b-a|<tol, stop
end
```

Algoritem: klik   Primer 1: klik   Primer 2: klik

Hitrost konvergence in računska zahtevnost

Naj bo δ_n velikost intervala po n-tem koraku bisekcije. Potem velja

$$\delta_0 = b - a, \quad \delta_1 = \frac{1}{2}\delta_0, \quad \delta_2 = \frac{1}{2}\delta_1 = \frac{1}{4}\delta_0, \quad \ldots, \quad \delta_n = \left(\frac{1}{2}\right)^n \delta_0$$

$$\Longrightarrow \qquad \frac{\delta_n}{\delta_0} = \left(\frac{1}{2}\right)^n = 2^{-n} \qquad \text{ali} \qquad n = \log_2\left(\frac{\delta_n}{\delta_0}\right)$$

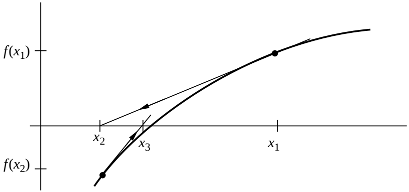
n	$\frac{\delta_n}{\delta_0}$	število izračunov funkcijskih vrednosti
5	3.1×10^{-2}	7
10	9.8×10^{-4}	12
20	9.5×10^{-7}	22
30	9.3×10^{-10}	32
40	9.1×10^{-13}	42
50	8.9×10^{-16}	52

Tangentna metoda

Iteracija:

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}. \tag{11}$$

Izpeljava:



Pri trenutnem približku x_k uporabimo funkcijsko vrednost $f(x_k)$ in odvod $f'(x_k)$, da izračunamo naslednji približek. Enačba tangente na krivuljo v točki $(x_k, f(x_k))$ je

$$y = f(x_k) + (x - x_k)\, f'(x_k).$$

Ker je cilj najti x, tako da je $f(x) = 0$, dobimo

$$0 = f(x_k) + (x_{k+1} - x_k)\, f'(x_k)$$

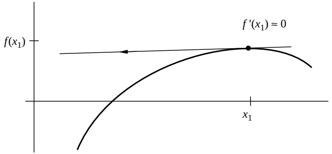
in izrazimo x_{k+1}.

```
zacetni podatki:  funkcija f, priblizek x_1
for k = 2,3,...
  x_k = x_{k-1} - f(x_{k-1})/f'(x_{k-1})
  if x_k znotraj tolerance, stop
end
```

Algoritem: klik   Primer 1: klik   Primer 2: klik

Lastnosti tangentne metode:

- Konvergira precej hitreje kot bisekcija - red konvergence je vsaj 2, tj. na vsakem koraku se število točnih decimalk podvoji.
- Zahteva analitično formulo za $f'(x)$ - če tega ne poznamo, lahko uporabimo sekantno metodo (sledi).
- Ni nujno, da konvergira, saj približki pobegnejo:

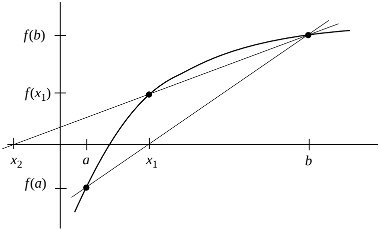


Sekantna metoda

Iteracija:

$$\boxed{x_{k+1} = x_k - f(x_k)\left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right]} \qquad (12)$$

Izpeljava:



S pomočjo dveh zaporednih približkov x_{k-1} in x_k, za nov približek vzamemo x-koordinato presečišča sekatne skozi točki $(x_k, f(x_k))$ in $(x_{k+1}, f(x_{k+1}))$ z abscisno osjo.

Naj bosta dana

$$x_k = \text{trenutni približek za ničlo}, \quad x_{k-1} = \text{prejšnji približek za ničlo.}$$

Aproksimiramo prvi odvod z naklonom sekante skozi točki $(x_k, f(x_k))$ in $(x_{k+1}, f(x_{k+1}))$:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

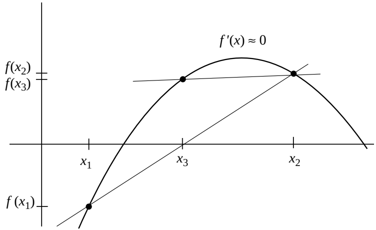
Vstavimo to aproksimacijo v (11) in dobimo (12).

```
zacetni podatki:  f, x_1, x_2
for k = 2,3 ...
  x_{k+1} = x_k - f(x_k)(x_k - x_{k-1})/(f(x_k) - f(x_{k-1}))
  if je izpolnjen tolerancni pogoj, stop
end
```

Algoritem: klik   Primer 1: klik   Primer 2: klik

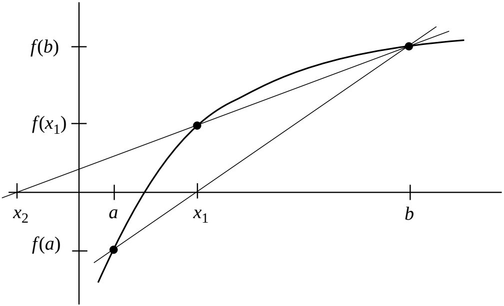
Lastnosti sekantne metode:

- Konvergenca je podobna tisti pri tangentni metodi. Red je ≈ 1.62, tj. na vsakem koraku se število točnih decimalk pomnoži z 1.62.
- Ne potrebujemo odvoda $f'(x)$.
- Naslednji približek ne ostane nujno znotraj začetnega intervala:



Vidimo, da bo nov približek x_{k+1}, daleč vstran od prejšnjega, če bo $f(x_k) \approx f(x_{k-1})$.

Metoda regula falsi



Metoda regula falsi je hibrid bisekcije in sekantne metode:

- Na vsakem koraku namreč izračunamo s pomočjo dveh zaporednih približkov a in b nov približek kot x-koordinato presečišča sekatne skozi točki $(a, f(a))$ in $(b, f(b))$ z abscisno osjo.
- Za nova približka a, b vzamemo interval, kjer je funkcija različno predznačena.

```
zacetni podatki:  a, b
for k = 2, 3 ...
  c = b - f(b)(b - a)/(f(b) - f(a))
  if f(a)f(c) < 0
      b = c
  else
      a = c
  if je izpolnjen tolerancni pogoj, stop
end
```

Algoritem: klik   Primer 1: klik   Primer 2: klik

Lastnosti metode regula falsi:

- Konvergenca je počasnejša kot pri sekantni.
- Naslednji približek vedno ostane znotraj začetnega intervala.

Metode fiksne točke

Metodo fiksne točke dobimo tako, da enačbo

$$f(x) = 0$$

preoblikujemo v ekvivalentno enačbo

$$g(x) = x.$$

Točki x pravimo negibna točka funkcije g.

Algoritem:

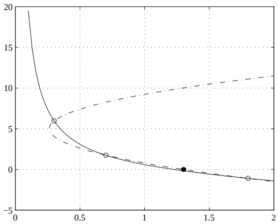
1. Izberi začetni približek x_0.
2. Ponavljaj iteracijo $x_{k+1} = g(x_k)$, dokler tolerančni kriterij ni izpolnjen.

Algoritem: klik   Primer 1: klik   Primer 2: klik
Primer 3: klik   Primer 4: klik

fzero funkcija

`fzero` je hibridna metoda v Matlabu, ki vključuje bisekcijo, sekantno metodo in obratno kvadratno interpolacijo.

Pri obratni kvadratni interpolaciji se išče presečišče parabole skozi tri točke $(x_0, f(x_0))$, $(x_1, f(x_1))$, $(x_2, f(x_2))$, z x-osjo.




```
r = fzero('fun',x0)
```

`fzero` izbere za naslednji približek

1. Rezultat obratne kvadratne interpolacije, če je le-ta znotraj začetnega intervala.
2. Rezultat sekantne metode, če prvi korak ni izpolnjen.
3. Rezultat bisekcije, če tudi drugi korak ni izpolnjen.

Sistemi nelinearnih enačb

Rešujemo sistem nelinearnih enačb:

$$
\begin{aligned}
f_1(x_1,\ldots,x_n) &= 0,\\
f_2(x_1,\ldots,x_n) &= 0,\\
&\vdots\\
f_n(x_1,\ldots,x_n) &= 0.
\end{aligned}
$$

Če definiramo

$$\underline{f} := (f_1,\ldots,f_n) : \mathbb{R}^n \to \mathbb{R}^n,$$

potem lahko sistem na kratko zapišemo kot

$$\underline{f}(\underline{x}) = 0.$$

Newtonova metoda: posplošitev tangentne metode,
Jacobijeva iteracija: posplošitev metode fiksne točke.

Newtonova iteracija

Pri Newtonovi iteraciji tvorimo zaporedje približkov

$$\boxed{\underline{x}^{(r+1)} = \underline{x}^{(r)} - J_{\underline{f}}(\underline{x}^{(r)})^{-1}\underline{f}(\underline{x}^{(r)})},$$

kjer je $J_{\underline{f}}(\underline{x}^{(r)})$ matrika prvih odvodov preslikave $\underline{f}$, ki ji pravimo Jacobijeva matrika:

$$J_{\underline{f}}(\underline{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}(\underline{x}).$$

V praksi pa ne računamo inverza $J_{\underline{f}}(\underline{x}^{(r)})^{-1}$, ampak namesto tega rešimo sistem

$$J_{\underline{f}}(\underline{x}^{(r)})\Delta\underline{x}^{(r)} = -\underline{f}(\underline{x}^{(r)}),$$
$$\underline{x}^{(r+1)} = \underline{x}^{(r)} + \Delta\underline{x}^{(r)}.$$

Algoritem: klik   Primer: klik

Jacobijeva iteracija

1. Sistem $\underline{f}(\underline{x}) = 0$ preoblikujemo v ekvivalentno obliko

$$\underline{g}(\underline{x}) = \underline{x},$$

   kjer je $\underline{g} : \mathbb{R}^n \to \mathbb{R}^n$.

2. Izberemo začetni približek

$$\underline{x}^{(0)} \in \mathbb{R}^n.$$

3. Računamo zaporedje približkov

$$\underline{x}^{(r+1)} = \underline{g}(\underline{x}^{(r)}).$$

Algoritem: klik   Primer: klik

Variacijske metode

Za predstavljene metode moramo imeti dober začetni približek, kajti v nasprotnem nimamo zagotovljene konvergence. Tega lahko dobimo z uporabo variacijskih metod, tj. metod za iskanje lokalnih minimumov. Povezavo med iskanjem ničel in iskanjem lokalnih ekstremov podaja naslednja trditev.

Trditev (Pretvorba iskanja ničel funkcije na iskanje globalnih minimumov)
Ničle funkcije $f(\underline{x})$ so globalni minimumi funkcije

$$g : \mathbb{R}^n \to \mathbb{R}, \qquad g(\underline{x}) = \|f(\underline{x})\|^2 = \sum_{i=1}^{n} \left(f_i(\underline{x})\right)^2.$$

Vprašanje: Kako iščemo lokalne ekstreme neke dvakrat zvezno odvedljive funkcije $g : \mathbb{R}^n \to \mathbb{R}$?

Iskanje lokalnih ekstremov g

- Minimum funkcije lahko iščemo iterativno tako, tekoči približek $\underline{x}^{(r)}$ popravimo v neki smeri v_r:

$$\underline{x}^{(r+1)} = \underline{x}^{(r)} + \lambda_r v_r, \tag{13}$$

  kjer je λ_r neko realno število. Veljalo bo:

$$g(\underline{x}^{(r+1)}) < g(\underline{x}^{(r)}).$$

Imamo več možnosti za izbiro smeri v_r v (13):

- Splošna metoda spusta: Izberemo katero koli smer, ki ni pravokotna na $\nabla g(\underline{x})$.
- Metoda najhitrejšega spusta: Za smer izberemo $v_r = -\nabla g(\underline{x})$.
- Metoda koordinatnega spusta: Za smeri zaporedoma izbiramo koordinatne smeri $e_1, e_2, \ldots, e_n$.

Po izbiri smeri moramo najti še λ_r v (13): Definiramo

$$q(\lambda) = g(\underline{x}^{(r)} + \lambda v_r).$$

Uporabimo eno od naslednjih metod:

- Metodo največjega spusta: Rešimo enačbo $q'(\lambda) = 0$ z eno od metod za reševanje neenačb v eni spremenljivki.
- Metoda tangentnega spusta: Poiščemo presečišče tangente na $y = q(\lambda)$ v točki $\lambda = 0$ z osjo x.
- Metoda paraboličnega spusta: S tangento določimo α, nato pa čez točke $(0, q(0))$, $(\alpha/2, q(\alpha/2))$, $(\alpha, q(\alpha))$ potegnemo parabolo in za λ izberemo njen minimum.

Algoritem: klik Primer 1: klik Primer 2: klik

Uporaba metod za iskanje ničel pri iskanju ekstremov

Trditev (Pretvorba iskanja lokalnih ekstremov ba iskanje ničel sistema)

Lokalni ekstremi funkcije $g(\underline{x})$ so rešitve sistema

$$\nabla g(x) = \begin{bmatrix} \frac{\partial g(x)}{\partial x_1} & \cdots & \frac{\partial g(x)}{\partial x_n} \end{bmatrix} = 0.$$

Vrsta ekstrema. O vrsti in obstoju ekstrema v stacionarni točki odloča Hessejeva matrika

$$H_g(\underline{x}) = \begin{bmatrix} \frac{\partial^2 g(\underline{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 g(\underline{x})}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 g(\underline{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 g(\underline{x})}{\partial x_n^2} \end{bmatrix}.$$

Če ima $H_g(\underline{x})$ same pozitivne lastne vrednosti (oz. negativne lastne vrednosti), je v $\underline{x}$ lokalni minimum (oz. lokalni maksimum).

Algoritem in primeri: klik

Polinomska interpolacija in aproksimacija

$*$ Poišči polinom p, da je
$p(x_i) = y_i, \ i = 0, 1, \ldots, n$.

$*$ Poišči polinom p stopnje k, da je
$\sum_{i=0}^{n} \|f(x_i) - p(x_i)\|^2$ minimalno.

- Interpolacija v standardni bazi
- Interpolacija v Lagrangeovi bazi
- Interpolacija v Newtonovi bazi
- Polinomska aproksimacija

Uvod v interpolacijo in aproksimacijo

Cilj: Aproksimirati želimo funkcijo $f(x)$ z lažjo funkcijo $g(x)$.

Tipi aproksimativnih funkcij: Polinomi, odsekoma polinomske funkcije, racionalne funkcije, trigonometrične funkcije, eksponentna funkcija, itd.

Vprašanje: Kako aproksimirati $f(x)$ z $g(x)$? V kakšnem smislu je aproksimacija dobra? Imamo več kriterijev:

1. Interpolacija: $g(x)$ mora imeti iste vrednost kot $f(x)$ na dani množici točk.
2. Metoda najmanjših kvadratov: $g(x)$ se mora čim bolj prilegati $f(x)$ v smislu 2-norme, tj.

$$\int_a^b |f(t) - g(t)|^2 \, dt \quad \text{mora biti čim manjše.}$$

3. Aproksimacija Čebiševa: $g(x)$ se mora čim bolj prilegati $f(x)$ v smislu supremum norme, tj.

$$\text{minimizirati želimo} \quad \max_{t \in [a,b]} |f(t) - g(t)|.$$

Interpolacijski polinom v standardni bazi

Dani so naslednji podatki:

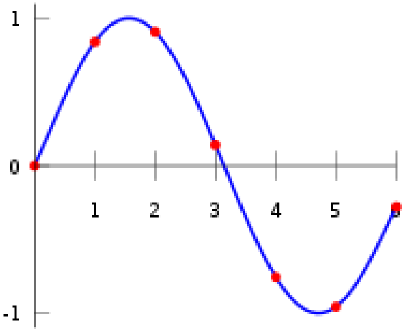
$$n+1 \quad \text{točk} \quad x_0, \ldots, x_n \quad \text{in vrednosti} \quad y_0, \ldots, y_n.$$

Iščemo polinom

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

stopnje n, ki zadošča

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad \ldots, \quad p(x_n) = y_n. \tag{14}$$



Dobimo sistem

$$
\begin{aligned}
a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0, \\
a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1, \\
&\vdots \\
a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n.
\end{aligned} \tag{15}
$$

Polinomu $p(x)$ pravimo interpolacijski polinom.

V matrični obliki lahko sistem (15) zapišemo kot

$$Ax = b,$$

kjer je

$$
A = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}, \quad x = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \quad b = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.
$$

Matriki A pravimo Vandermondova matria na točkah $x_0, \ldots, x_n$, velja pa

$$\boxed{\det(A) = \prod_{0 \leqslant j < i \leqslant n} (x_i - x_j)}.$$

Posledica (O obstoju in enoličnosti interpolacijskega polinoma)

- *Če so točke x_i, $i = 0, \ldots, n$, paroma različne, ima sistem enolično rešitev.*
- *Polinom stopnje največ n skozi $n + 1$ točk je en sam.*

Vprašanje:

1. Kako računsko zahtevno je reševanje sistema (15)?
2. Ali je sistem (15) numerično občutljiv?

Odgovor:

1. Računanje interpolacijskega polinoma s pomočjo Vandermondove matrike ni poceni ($\frac{2}{3}n^3 + \mathcal{O}(n^2)$ operacij).
2. Sistem je lahko že pri majhnem številu točk (npr. 10) zelo občutljiv za numerične napake.

Interpolacijski polinom: Lagrangeova in Newtonova baza

Namesto uporabe standardne baze

$$1, x, x^2, \dots, x^n$$

je bolje uporabiti eno od naslednjih baz:

- Lagrangeova baza:
$\frac{(x-x_1)\cdots(x-x_n)}{(x_0-x_1)\cdots(x_0-x_n)}, \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)}, \dots, \frac{(x-x_0)\cdots(x-x_{n-1})}{(x_n-x_0)\cdots(x_n-x_{n-1})}.$

- Newtonova baza:
$1, x - x_0, (x - x_0)(x - x_1), \dots, (x - x_0)\cdots(x - x_{n-1}).$

Obe zgornji bazi sta stabilni, Newtonova pa je cenejša za računanje v primeru dodajanja novih interpolacijskih točk.

Interpolacijski polinom v Lagrangeovi bazi

Primer

Poišči polinom najnižje stopnje, ki interpolira naslednji točki:

x	1.4	1.25
y	3.7	3.9

Dobimo

$$p_1(x) = \left(\frac{x-1.25}{1.4-1.25}\right)3.7 + \left(\frac{x-1.4}{1.25-1.4}\right)3.9 = 3.7 - \frac{4}{3}(x-1.4)$$

Kaj smo naredili? Zapisali smo $p(x)$ v obliki

$$p(x) = \underbrace{\left(\frac{x-x_1}{x_0-x_1}\right)}_{\substack{\ell_0(x),\\ \ell_0(x_0)=1,\ell_0(x_1)=0}} y_0 + \underbrace{\left(\frac{x-x_0}{x_1-x_0}\right)}_{\substack{\ell_1(x),\\ \ell_1(x_0)=0,\ell_1(x_1)=1}} y_1$$

Danih imamo $n+1$ točk

$$(x_0, y_0),\ (x_1, y_1), \ldots, (x_n, y_n).$$

Cilj je najti Lagrangeove bazne polinome stopnje največ n , ki zadoščajo

$$\ell_i(x_j) = \begin{cases} 0, & j \neq i, \\ 1, & j = i. \end{cases}$$

Torej je

$$\ell_i(x) = \underbrace{C_i}_{\text{konstanta}} \cdot \prod_{j \neq i} (x - x_j), \quad i = 0, \ldots, n.$$

i–ti Lagrangeov bazni polinom je

$$\ell_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \ldots, n.$$

Interpolacijski polinom v Lagrangeovi obliki je

$$p(x) = \sum_{i=0}^{n} \ell_i(x) y_i$$

Primer

Poišči enačbo parabole v Lagrangeovi obliki, ki gre skozi točke

$$(1,6),\ (-1,0),\ (2,12).$$

$$\begin{aligned}
\ell_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} &= \frac{(x+1)(x-2)}{(2)(-1)} \\
\ell_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} &= \frac{(x-1)(x-2)}{(-2)(-3)} \\
\ell_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} &= \frac{(x-1)(x+1)}{(1)(3)}
\end{aligned}$$

$$\begin{aligned}
p_2(x) &= y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) \\
&= -3(x+1)(x-2) + 0(x-1)(x-2) + 4(x-1)(x+1).
\end{aligned}$$

Interpolacijski polinom v Newtonovi bazi

Newtonov interpolacijski polinom na točkah $x_0, x_1, x_2, \ldots, x_n$ je oblike

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots \\ + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

Newtonovi bazni polinomi so

$$1, x - x_0, (x - x_0)(x - x_1), \ldots, \prod_{i=0}^{n-1}(x - x_i).$$

Newtonova baza proti Lagrangeovi bazi:

Prednost Newtonove baze pred Lagrangeovo je v tem, da se z dodajanjem novih točk $x_{n+1}, \ldots, x_{n+m}$ vsi že izračunani koeficienti $c_0, \ldots, c_n$ ne spremenijo.

V primeru zlepkov, ko imamo v naprej določen n, so Lagrangeovi polinomi primernejši, saj imamo koeficiente že dane.

Interpolirajmo podatke (x_0, y_0), (x_1, y_1), (x_2, y_2) v Newtonovi obliki.

Poiskati moramo koeficiente c_0, c_1 in c_2 v polinomu

$$p_2(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1).$$

Iz n podatkov dobimo sistem n linearnih enačb v neznanih koeficientih:

$$\begin{aligned}
x_0 :&\quad y_0 = c_0 + 0 + 0\\
x_1 :&\quad y_1 = c_0 + c_1(x_1 - x_0) + 0\\
x_2 :&\quad y_2 = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1)
\end{aligned}$$

Ali v matrični obliki:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Ker je matrika spodnje trikotna, potrebujemo samo $\mathcal{O}(n^2)$ operacij:

$$
\begin{aligned}
c_0 &= y_0 = f(x_0), \\
c_1 &= \frac{y_1 - c_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, \\
c_2 &= \frac{y_2 - c_0 - (x_2 - x_0)c_1}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{f(x_2) - f(x_0) - (x_2 - x_0)\frac{f(x_1)-f(x_0)}{x_1-x_0}}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0}.
\end{aligned}
$$

Deljena diferenca $f[x_0, \ldots, x_k]$

Iz zgornjega primera opazimo naslednji vzorec. Pojavljajo se izrazi oblike:

$$\frac{f(x_j) - f(x_i)}{x_j - x_i}. \tag{16}$$

Če izraz (16) označimo z oglatimi oklepaji kot $f[x_i, x_j]$, potem bi na na našem primeru dobili:

$$c_0 = f(x_0), \quad c_1 = f[x_0, x_1], \quad c_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

To se da posplošiti do rekurzivnega računanja polinomov v Newtonovi obliki.

Deljena diferenca $f[x_0, \ldots, x_k]$ je vodilni koeficient (pri x^k) interpolacijskega polinoma stopnje največ k, ki se z f ujema v točkah $x_0, \ldots, x_k$.

Izrek (O koeficientih Newtonovega interpolacijskega polinoma)

1. *Koeficienti Newtononovega interpolacijskega polinom p_n stopnje največ n, ki se z f ujema v točkah $x_0, \ldots, x_n$, so enaki*

$$c_i = f[x_0, x_1, \ldots, x_i], \quad i = 0, \ldots, n.$$

2. *Deljene diference povezuje formula*

$$\boxed{f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}}.$$

Če dopuščamo, da se točke x_i v $f[x_0, \ldots, x_k]$ ponavljamo, potem želimo, da se interpolacijski polinom ujema s funkcijo še v odvodu.

Definiramo

$$f[x_0, \ldots, x_k] = \begin{cases} \frac{f^{(k)}(x_0)}{k!}, & x_0 = x_1 = \ldots = x_k, \\ \frac{f[x_1,\ldots,x_k] - f[x_0,\ldots,x_{k-1}]}{x_k - x_0}, & \text{sicer.} \end{cases}$$

Deljene diference pa lahko bolj učinkovito računamo s pomočjo tabel:

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

Primer. Konstruirajmo deljene diference za podatke $(1,3)$, $(\frac{3}{2},\frac{13}{4})$, $(0,3)$, $(2,\frac{5}{3})$.

Iz tabele deljenih diferenc preberimo interpolacijski polinom.

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Interpolacijski polinom je tako

$$p_2(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)\Big(x-\frac{3}{2}\Big) - 2(x-1)\Big(x-\frac{3}{2}\Big)x.$$

Če uporabimo spodnjo stranico trikotnika, pa dobimo $p_2(x)$ izražen v drugi Newtonovi bazi:

$$p_2(x) = \frac{5}{3} - \frac{2}{3}(x-2) - \frac{5}{3}(x-2)x - 2(x-2)x\Big(x-\frac{3}{2}\Big).$$

Višanje stopnje aproksimacije

$\cdots$ ne izboljša vedno aproksimacije funkcije s polinomom.

Znan je Rungejev primer, ko funkcijo

$$f(x) = \frac{1}{1+x^2}$$

interpoliramo na intervalu $[-5, 5]$ z ekvidistantnimi točkami, tj.

$$x_0 = -5, x_1 = -5 + 10 \cdot \frac{1}{n}, \ldots, x_{n-1} = -5 + 10 \cdot \frac{n-1}{n}, x_n = 5.$$

Pričakujemo, da se bo interpolacijski polinom vse bolj prilegal naši funkciji. Izkaže pa se, da temu ni tako. Če interpoliramo v točkah Čebiševa

$$x_i = 5 \cos\left(\frac{\pi}{2(i-1)(n+1)}\right), \quad i = 0, \ldots, n$$

pa z višanjem stopnje res dobimo boljše prileganje.

Primer 1: klik Primer 2: klik Primer 3: klik

Napaka polinomske interpolacije

Ponavadi nas zanima razlika med vrednostjo funkcije f in vrednostjo interpolacijskega polinoma p_n v neki točki t:

$$e_n(t) = p_n(t) - f(t).$$

Naj bo q_{n+1} interpolacijski polinom funkcije f skozi točke $x_0, \ldots, x_n$ in t:

$$q_{n+1}(x) = p_n(x) + f[x_0, x_1, \ldots, x_n, t] \cdot \overbrace{\prod_{i=0}^{n}(x - x_i)}^{\omega(x)}.$$

Iz enakosti $f(t) = q_{n+1}(t)$ sledi

$$e_n(t) = p_n(t) - f(t) = -f[x_0, x_1, \ldots, x_n, t]\omega(t).$$

Za oceno napako moramo oceniti še vrednost $f[x_0, x_1, \ldots, x_n, t]$.

Izrek

$$f[x_0, x_1, \ldots, x_n, t] = \frac{f^{(n+1)}(\xi)}{(n+1)!} \quad \textit{za nek } \xi \in [a, b].$$

Izrek (Napaka polinomske interpolacije)

Naj bo f vsaj $(n+1)$-krat zvezno odvedljiva na intervalu $[a,b]$ in naj bo p_n interpolacijski polinom stopnje največ n skozi točke x_i, $i=0,\ldots,n$, ki vse ležijo na intervalu $[a,b]$. Potem je za vsak $x \in [a,b]$

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)\cdots(x-x_n),$$

kjer ξ leži na intervalu $[a,b]$.

Če znamo odvod $f^{(n+1)}$ na intervalu, ki nas zanima, omejiti, lahko dobimo uporabno oceno.

Aproksimacija po metodi najmanjših kvadratov

Za funkcijo, podano v n točkah

$$(x_0, y_0), \ldots, (x_n, y_n),$$

iščemo polinom p_k stopnje $k \leqslant n$, za katerega ima izraz

$$E_{LSQ} = \sqrt{\sum_{i=0}^{n} (p_k(x_i) - y_i)^2}$$

najmanjšo vrednost. Če zapišemo na dolgo:

$$E_{LSQ} = \sqrt{\sum_{i=0}^{n} (\underbrace{a_0 + a_1 x_i + \ldots + a_k x_i^k}_{p_k(x_i)} - y_i)^2}.$$

Torej iščemo ekstrem funkcije več spremenljivk. Iz analize vemo, da je potreben pogoj za ekstrem

$$\frac{\partial E_{LSQ}}{\partial a_0} = \frac{\partial E_{LSQ}}{\partial a_1} = \ldots = \frac{\partial E_{LSQ}}{\partial a_k} = 0.$$

Naj bo

$$s_1 = x_0 + \ldots + x_n, \quad s_2 = x_0^2 + \ldots + x_n^2, \quad \ldots, \quad s_{2k} = x_0^{2k} + \ldots + x_n^{2k}.$$

Dobimo normalni sistem:

$$\begin{bmatrix} n & s_1 & s_2 & \ldots & s_k \\ s_1 & s_2 & s_3 & \ldots & s_{k+1} \\ s_2 & s_3 & s_4 & \ldots & s_{k+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s_k & s_{k+1} & s_{k+2} & \ldots & s_{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^n y_i \\ \sum_{i=0}^n y_i x_i \\ \sum_{i=0}^n y_i x_i^2 \\ \vdots \\ \sum_{i=0}^n y_i x_i^k \end{bmatrix},$$

ki pa je pri velikem številu točk lahko numerično slabo pogojen.

Predoločeni sistemi

Problem polinomske aproksimacije je poseben primer predoločenega sistema:

Za matriko $A \in \mathbb{R}^{n \times m}$, $n \geqslant m$, in vektor $b \in \mathbb{R}^n$, iščemo vektor $x \in \mathbb{R}^m$, ki zadošča:

$$Ax = b. \tag{17}$$

Kadar je $n > m$, točna rešitev sistema (17) najverjetneje ne obstaja. Zato nas navadno zanima rešitev po metodi najmanjših kvadratov:

$$\text{Poišči } x \in \mathbb{R}^m, \text{ ki minimizira } \|Ax - b\|_2.$$

Trditev

Naj bo $\operatorname{rank}(A) = m$. *Rešitev sistema* (17) *po metodi najmanjših kvadratov je* $x \in \mathbb{R}^m$, *ki reši t.i. normalni sistem:*

$$\underbrace{A^T A}_{m \times m} x = \underbrace{A^T b}_{m \times 1}. \tag{18}$$

QR razcep

Težava, ki se pojavi pri reševanju normalnega sistema (18), je numerična nestabilnost. Problematične so matrike A, pri kateri stolpci niso dovolj linearno neodvisni.

Rešitev tega problema je uporaba t.i. QR razcepa matrike A.

QR razcep matrike $A \in \mathbb{R}^{n \times m}$ sta ortogonalna matrika $Q \in \mathbb{R}^{n \times m}$ ($Q^T Q = I_m$) in zgornjetrikotna matrika $R \in \mathbb{R}^{m \times m}$, ki zadoščata

$$A = QR. \tag{19}$$

Iz (19) sledi, da se sliki matrike A in Q ujemata. Pogoj ortogonalnosti matrike Q pa pomeni, da so njeni stolpci normirani (tj. dolžine 1) in paroma pravokotni.

Če označimo z $a_1, \ldots, a_m$ in $q_1, \ldots, q_m$ stolpce matrik A in Q ter $R = [r_{i,j}]_{i,j}$, potem veljajo zveze:

$$
\begin{aligned}
q_1 &= \frac{1}{r_{11}} a_1, \\
q_2 &= \frac{1}{r_{22}} (a_2 - r_{12} q_1), \\
q_3 &= \frac{1}{r_{33}} (a_3 - r_{13} q_1 - r_{23} q_2), \\
&\vdots \\
q_m &= \frac{1}{r_{mm}} (a_m - r_{1m} q_1 - \ldots - r_{m-1,m} q_{m-1}).
\end{aligned} \tag{20}
$$

Iz (20) lahko izpeljemo enega od načinov za izračun QR razcepa, tj. z uporabo Gram-Schmidtove ortogonalizacije (GSO):

```
A = [a_1, ..., a_m] je n × m matrika s stolpci a_1, ..., a_m

r_{1,1} = ||a_1||_2
q_1 = (1/r_{1,1}) a_1
for i = 2, ..., m
    q_i = a_i
    for j = 1, ..., i-1
      r_{j,i} = q_j^T a_i
      q_i = q_i - r_{j,i} q_j
    end
    r_{i,i} = ||q_i||_2
    q_i = (1/r_{i,i}) q_i
end
```

Izkaže se, da je računska zahtevnost GSO $\frac{4}{3}n^3 + \mathcal{O}(n^2)$, kar je dvakrat dražje od računanja LU razcepa z delnim pivotiranjem in da je GSO stabilna operacija.

Uporaba QR razcepa za reševanje sistema $Ax = b$

Trditev

Naj bo $A \in \mathbb{R}^{n\times m}$ in $\operatorname{rank}(A) = m$. Rešitev sistema $Ax = b$ po metodi najmanjših kvadratov je enaka rešitvi zgornjetrikotnega sistema

$$Rx = Q^T b, \tag{21}$$

kjer je $A = QR$ za zgornjetrikotno matriko R in ortogonalno matriko Q.

Dokaz. Vemo, da je rešitev sistema (17) po metodi najmanjših kvadratov, kar rešitev normalnega sistema (18). Velja:

$$A^T Ax = (QR)^T (QR)x = (R^T Q^T)(QR)x = R^T (Q^T Q)Rx = R^T Rx,$$
$$A^T b = (QR)^T b = R^T Q^T b.$$

Sistem (18) je tako ekvivalenten sistemu $R^T Rx = R^T Q^T b$. Ker je po predpostavki $\operatorname{rank}(A) = m$, je tudi $\operatorname{rank}(R) = \operatorname{rank}(R^T) = m$. Zato je $R^T \in \mathbb{R}^{m\times m}$ obrnljiva in z množenjem zadnje enačbe z leve z $(R^T)^{-1}$, dobimo (21).

Opomba

- *V Matlabu QR razcep matrike izračunamo z ukazom* $[Q, R] = qr(A)$.
- *V Matlabu sistem* $Ax = b$ *rešimo po metodi najmanjših kvadratov z ukazom* $A \backslash b$. *V ozadju operatorja* $\backslash$ *je uporaba QR razcepa.*

Linearna regresija

Iščemo premico, ki se najbolje prilega podatkom $(x_1, y_1), \ldots, (x_n, y_m)$ po metodi najmanjših kvadratov. Premica je oblike $y = a + bx$. Torej sta spremenljivki a in b. Sistem lahko zapišemo v obliki

$$\underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\vec{b}}. \tag{22}$$

Po zgoraj napisanem je rešitev (22) enaka

$$\vec{x} = (A^T A)^{-1} A^T \vec{b} = \begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}.$$

Numerična integracija

Oceni $\int_a^b f(x)dx$.

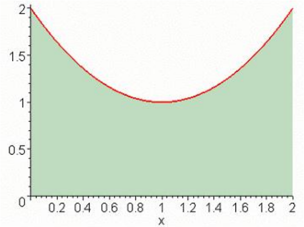
- Newton–Cotesova (NC) pravila: trapezno, Simpsonovo pravilo
- Izbira koraka v NC pravilih
- Adaptivna NC pravila
- Gaussove kvadraturne formule
- Integracija v več spremenljivkah

Numerična integracija

Naš cilj je izračunati določen integral

$$\int_a^b f(x)\, dx = F(b) - F(a)$$

funkcije $f(x)$. Tu je F nedoločen integral funkcije f.

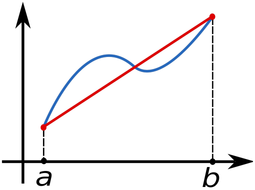


Če ne znamo izračunati nedoločenega integrala F, smo v težavah. Npr. za $f(x) = e^{-x^2}$, $g(x) = \frac{\sin x}{x}$, $h(x) = x \tan x$.

Prav tako ne moremo točno izračunati vrednosti integrala, če imamo funkcijo podano samo na neki množici točk.

Osnovno trapezno pravilo in napaka E

Integral $\int_a^{a+h} f(x)\,dx$ tako, da f aproksimiramo z linearno funkcijo in izračunamo ploščino pod linearno funkcijo oz. trapezom.



$$\boxed{p(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)}$$

Velja

$$\int_a^b f(x)\,dx \approx \int_a^b p(x)\,dx = f(a)(b-a) + \frac{f(b) - f(a)}{b - a}\frac{(b-a)^2}{2}$$
$$= \boxed{\frac{(b-a)}{2}\left(f(a) + f(b)\right)}.$$

Pri tem je napaka naslednja:

$$\begin{aligned} E &= \int_a^b (f(x) - p(x))\,dx = \int_a^b f[a,b,x](x-b)(x-a)\,dx \\ &= f[a,b,\xi] \cdot \int_a^{a+h} (x-b)(x-a)\,dx \\ &= \frac{f''(\eta)}{2}\left(-\frac{1}{6}(b-a)^3\right) \\ &= \boxed{-\frac{(b-a)^3 f''(\eta)}{12}}, \end{aligned}$$

kjer sta $\xi, \eta \in [a,b]$, tretja enakost sledi po izreku o povprečni vrednosti, četrta pa po izreku o $f[a,b,\xi]$.

Sestavljeno trapezno pravilo

Če interval ni zelo kratek, potem očitna naivna linearna transformacija običajno ne da dobrega približka integrala.

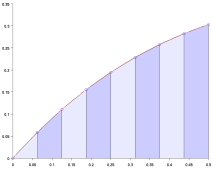
Če interval $[a, b]$ razdelimo z ekvidistantnimi točkami $x_0, x_1, \ldots, x_n$, tj.

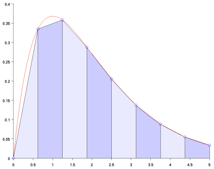
$$h := h_i = x_{i+1} - x_i$$

je konstanta in na vsakem intervalu uporabimo osnovno trapezno pravilo, dobimo:

$$\int_a^b f(x)\,dx \approx \frac{h}{2} \sum_{i=0}^{n-1} f(x_i) + f(x_{i+1})$$

$$= \boxed{\frac{h}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right)}$$





Napaka E_i na intervalu $[x_i, x_{i+1}]$ je enaka

$$E_i = -\frac{h^3 \cdot f''(\eta_i)}{12} \qquad \text{za nek } \eta_i \in [x_i, x_{i+1}].$$

Torej je skupna napaka

$$\begin{aligned} E &= \sum_{i=0}^{n-1} E_i = \sum_{i=0}^{n-1} -\frac{h^3 \cdot f''(\eta_i)}{12} = -n \cdot \frac{h^3 \cdot f''(\eta)}{12} \\ &= \boxed{-\frac{(b-a)h^2 \cdot f''(\eta)}{12}}, \end{aligned}$$

kjer je $\eta \in [a, b]$ in smo v tretji enakosti uporabili izrek o srednji vrednosti.

Algoritem: klik

Primer - $\int_0^1 e^{-x^2}\,dx$

Koliko točk uporabiti, da bo sestavljeno trapezno pravilno natančno z napako omejeno z 10^{-6}?

Želimo

$$\left|\frac{(b-a)h^2 f''(\eta)}{12}\right| \leqslant 10^{-6}$$

Kako velik je drugi odvod $f''(x)$?

$$f'(x) = -2xe^{-x^2}, \quad f''(x) = -2e^{-x^2} + 4x^2e^{-x^2}.$$

Ker je

$$f'''(x) = 12xe^{-x^2} - 8x^3e^{-x^2} = 4x(3-2x^2)e^{-x^2}$$

pozitiven na $[0,1]$, je f'' monotono naraščajoč na $[0,1]$ in zato zavzame maksimum v krajišču: $f''(0) = 2$. Potem lahko omejimo

$$\frac{(b-a)2h^2}{12} \leqslant 10^{-6} \quad \Rightarrow \quad h^2 \leqslant 6{\cdot}10^{-6} \quad \Rightarrow \quad \underbrace{\sqrt{(1/6)}10^3}_{\approx 410} \leqslant n.$$

Trapezno pravilo s kontrolo koraka

Motivacija. Če uporabimo sestavljeno trapezno pravilo, moramo:

- Vnaprej določiti velikost h.
- Če želimo oceniti napako, moramo znati oceniti $f''(\eta)$ na intervalu $[a, b]$.

Obe težavi želimo rešiti, tj. radi bi, da funkcija samo zmanjšuje h, v kolikor napaka ni dovolj manjka. V ta namen moramo znati to napako oceniti. Pridemo do trapeznega pravila s kontrolo koraka.

Naj bo $I = \int_a^b f(x)dx$ in $T(h)$ ocena za I z uporabo sestavljenega trapeznega pravila z velikostjo intervala h.

Spomnimo se, da pri sestavljenem trapeznem pravilu $T(h)$ za napako $E(h)$ velja:

$$E(h) := T(h) - I = \frac{b-a}{12} f''(\xi_h)h^2, \quad \text{kjer je } \xi_h \in (a, b).$$

Želimo se izogniti dejstvu, da moramo poznati f''. Zapišimo napako še v primeru razpolovljenega koraka, tj. $\frac{h}{2}$:

$$E(h/2) := T(h/2) - I = \frac{b-a}{12} f''(\xi_{h/2}) \frac{h^2}{4}, \quad \text{kjer je } \xi_{h/2} \in (a, b).$$

Predpostavimo, da je $\frac{b-a}{12} f''(\xi_h)$ približno enako C za vsak h.

Dobimo:

$$I = T(h) - Ch^2 = T(h/2) - C\frac{h^2}{4}.$$

Sledi:

$$T(h) - T(h/2) = \frac{3}{4}Ch^2 + \mathcal{O}(h^4) \qquad \text{oz.} \qquad Ch^2 \approx \frac{4}{3}(T(h) - T(h/2)).$$

Tako sta

$$\frac{4}{3}(T(h) - T(h/2)), \quad \frac{1}{3}(T(h) - T(h/2))$$

približka za napaki $E(h)$ in $E(h/2)$. Velja

$$T(h/2) = \underbrace{\frac{T(h)}{2}}_{\text{razplovimo } T(h)} + \underbrace{\frac{h}{2}\sum_{i=1}^{n} f(a + (i - 1/2)h),}_{\text{računamo samo ta del}} \quad n = (b-a)/h.$$

Algoritem:

1. Izračunamo $T(b-a) = (b-a)\frac{f(a)+f(b)}{2}$.
2. Izračunamo $T((b-a)/2) = \frac{T(b-a)}{2} + \frac{b-a}{2}f((a+b)/2)$.
3. Izračunamo $\frac{1}{3}(T(b-a) - T((b-a)/2))$. Če je to dovolj majhno po absolutni vrednosti, končamo, približek za integral pa je $T((b-a)/2)$. Sicer ponovimo postopek z razpolovljenim h.

Algoritem: klik

Adaptivno trapezno pravilo

Motivacija: Če uporabimo trapezno pravilo s kontrolo koraka, potem dolžine koraka h ne rabimo sami določiti, vendar pa je h enak na celotnem integracijskem intervalu. Želeli bi, da na nekaterih delih intervala uporabimo večje h, manjše pa le tam, kjer je to res potrebno.

Zgornji cilj lahko dosežemo z uporabo rekurzivnega računanja integrala:

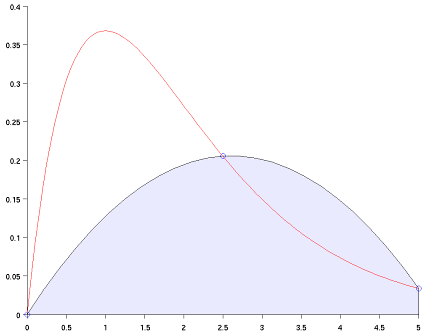
- Najprej izračunamo $T(b-a)$ in $T((b-a)/2)$.
- Če je podobno kot pri kontroli koraka zgoraj ocena napake $e := \frac{T(b-a)/2 - T(b-a)}{3}$ dovolj majhna, vrnemo $T((b-a)/2) + e$ in končamo.
- Če je e prevelik, ponovimo zgornji postopek ločeno za podintervala $[a, (a+b)/2]$ in $[(a+b)/2, b]$, pri čemer naj bo napaka na vsakem največ polovica začetne tolerance.
- Rekurzivno nadaljujemo zgornji postopek in dobimo oceno integrala, pri čemer delilne točke ne bodo enakomerno razporejene po intervalu $[a, b]$.

Algoritem: klik

Enostavno Simpsonovo pravilo

Naj bo p_2 polinom stopnje 2, s katerim interpoliramo točke

$$\big(a, f(a)\big), \quad \Big(\frac{a+b}{2}, \quad f\Big(\frac{a+b}{2}\Big)\Big), \quad \big(b, f(b)\big):$$



$$\boxed{p_2(x) = C_0 + C_1 \cdot (x-a) + C_2 \cdot (x-a)\left(x - \frac{a+b}{2}\right)}.$$

Označimo $h := \frac{b-a}{2}$. Rešujemo sistem:

$$p_2(a) = f(a), \quad p_2\Big(\frac{a+b}{2}\Big) = f\Big(\frac{a+b}{2}\Big), \quad p_2(b) = f(b).$$

Dobimo

$$C_0 = f[a] = f(a), \qquad C_1 = f\Big[a, \frac{a+b}{2}\Big] = \frac{f(a+h)-f(a)}{h},$$
$$C_2 = f\Big[a, \frac{a+b}{2}, b\Big] = \frac{f(a+2h)-2f(a+h)+f(a)}{2h^2}.$$

Računamo $\int_a^b p_2(x)dx$ (naredimo subsitucijo $x = a+t$);

$$\begin{aligned} &\int_a^{a+2h} p_2(x)\, dx = \int_0^{2h} p_2(a+t)\, dt \\ = \quad & f(a)\cdot 2h + \frac{f(a+h)-f(a)}{h}\cdot 2h^2 + \frac{f(a+2h)-2f(a+h)+f(a)}{2h^2} \\ = \quad & \boxed{\frac{h}{3}\big(f(a)+4f(a+h)+f(a+2h)\big)}. \end{aligned}$$

Izkaže se, da je napaka približno:

$$\boxed{-\frac{1}{90}h^5 f^{(4)}(\xi)}, \quad \xi \in [a, b]$$

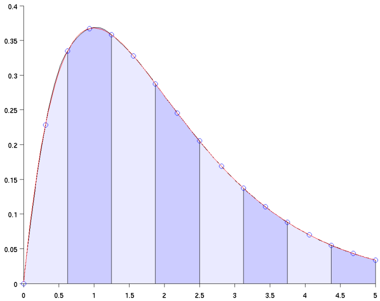
Algoritem: klik

Sestavljeno Simpsonovo pravilo in napaka

Vzemimo ekvidistantno particijo $P = \{x_0 = a < \cdots < x_n = b\}$ intervala $[a, b]$ na sodo število enako dolgih intervalov in na zaporednih trojicah točk uporabimo osnovno Simpsonovo pravilo ($h = x_{i+1} - x_i$):

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{\frac{n}{2}-1} \frac{h}{3}[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})]$$

$$= \boxed{\frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + f(x_n)\right]}.$$



Napaka E_i na intervalu $[x_{2i}, x_{2i+2}]$ je enaka

$$E_i = -\frac{h^5 f^{(4)}(\eta_i)}{90}$$

za nek $\eta_i \in [x_{2i}, x_{2i+2}]$. Torej je skupna napaka

$$E = \sum_{i=0}^{\frac{n}{2}-1} E_i = \sum_{i=0}^{\frac{n}{2}-1} -\frac{h^5 f^{(4)}(\eta_i)}{90} = -\frac{n}{2}\frac{h^5 f^{(4)}(\eta)}{90}$$

$$= \boxed{-\frac{(b-a)h^4 f''(\eta)}{180}},$$

kjer je $\eta \in [a, b]$ in smo v tretji enakosti uporabili izrek o srednji vrednosti.

Adaptivno Simpsonovo pravilo

Motivacija. Ideja je povsem enaka kot pri adaptivnem trapeznem pravilu, tj. radi bi uporabili čim večji h povsod, kjer je to mogoče. Če s $S(h)$ označimo vrednost sestavljenega Simpsonovega pravila s korakom dolžine h, potem napako E ocenimo iz $S(h)$ in $S(h/2)$.

Postopek:

- Najprej izračunamo $S(b-a)$ in $S((b-a)/2)$.
- Iz $\int_a^b f(x)dx = S(h) + C_1 h^4 = S(h/2) + C_1 (\frac{h}{2})^4$ izrazimo
$$C_1 \left(\frac{h}{2}\right)^4 = \frac{S(b-a)/2 - S(b-a)}{15},$$
kar je naša ocena napake E. Če je E dovolj majhna, vrnemo $S((b-a)/2) + E$ in končamo.
- Če je E prevelik, ponovimo zgornji postopek ločeno za podintervala $[a, (a+b)/2]$ in $[(a+b)/2, b]$, pri čemer naj bo napaka na vsakem največ polovica začetne tolerance.
- Rekurzivno nadaljujemo zgornji postopek in dobimo oceno integrala, pri čemer delilne točke ne bodo enakomerno razporejene po intervalu $[a, b]$.

Algoritem: klik

Osnovna Newton–Cotesova pravila

Newton–Cotesova (NC) pravila interval $[a, b]$ razdelijo z $n + 1$ ekvidistantnimi točkami in $\int_a^b f(x)dx$ aproksimirajo z $\int_a^b p_n(x)dx$, kjer je p_n interpolacijski polinom stopnje n na teh točkah.

Pravila:

ime	n	formula
trapezno	1	$\frac{(b-a)}{2}\left[f(a) + f(b)\right]$
Simp. $1/3$	2	$\frac{(b-a)}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]$
Simp. $3/8$	3	$\frac{(b-a)}{8}\left[f(a) + 3f(a+h) + 3f(b-h) + f(b)\right]$
Boolovo	4	$\frac{(b-a)}{90}\left[7f(a) + 32f(a+h) + 12f(\frac{a+b}{2}) + 32f(b-h) + 7f(b)\right]$

Ocene napak:

ime	n	napaka	h
trapezno	1	$-\frac{(b-a)h^2}{12}f''(\xi)$	$h = b - a$
Simpsonovo $1/3$	2	$-\frac{(b-a)h^4}{180}f^{(4)}(\xi)$	$h = (b-a)/2$
Simpsonovo $3/8$	3	$-\frac{(b-a)h^4}{80}f^{(4)}(\xi)$	$h = (b-a)/3$
Booleovo	4	$-\frac{2(b-a)h^6}{945}f^{(6)}(\xi)$	$h = (b-a)/4$

Izpeljava NC pravil z metodo nedoločenih koeficientov

Izpeljati želimo integracijsko formulo na danih (ekvidistantnih) $n+1$ točkah, vozlih, ki bo točna za polinome stopnje največ n:

$$\int_a^{a+nh} f(x)\,dx = \sum_{i=0}^{n} a_i f(a+ih) + R(f(x)),$$

kjer so $a_0, \ldots, a_n$ iskani koeficienti, $R(f(x))$ pa napaka. Izpeljimo Simpsonovo $3/8$ pravilo.

$$\int_0^{3h} f(x)\,dx = a_0 f(0) + a_1 f(h) + a_2 f(2h) + a_3 f(3h) + R(f(x)),$$

Želimo

$$R(1) = R(x) = R(x^2) = R(x^3) = 0:$$

$$\begin{aligned}
\int_0^{3h} 1\,dx &= 3h = a_0 + a_1 + a_2 + a_3, \\
\int_0^{3h} x\,dx &= \frac{9h}{2} = a_1 + 2a_2 + 3a_3, \\
\int_0^{3h} x^2\,dx &= \frac{27h}{3} = a_1 + 4a_2 + 9a_3, \\
\int_0^{3h} x^3\,dx &= \frac{81h}{4} = a_1 + 8a_2 + 27a_3.
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
a_0 &= \frac{3}{8}h, \\
a_1 &= \frac{9}{8}h, \\
a_2 &= \frac{9}{8}h, \\
a_3 &= \frac{3}{8}h.
\end{aligned}$$

Predvidevamo, da je napaka $R(f(x))$ oblike

$$D \cdot f^{(4)}(\xi),$$

kjer je $\xi \in [a,b]$. Za $f(x) = x^4$ dobimo

$$\int_0^{3h} x^4\, dx = \frac{3^5}{5} h^5 = \frac{3}{8} h(3h^4 + 3 \cdot 2^4 \cdot h^4 + 3^4 h^4) + 24D \qquad \Rightarrow \qquad D = -\frac{3}{80} h^5.$$

Torej:

$$\boxed{\int_a^b f(x)dx = \frac{(b-a)}{8}\left[f(a) + 3f(a+h) + 3f(b-h) + f(b)\right] - \frac{(b-a)h^4}{80} f^{(4)}(\xi)}.$$

Gaussove kvadraturne formule

- NC pravila za integriranje so oblike

$$\int_a^b f(x)\,dx \approx \sum_{j=0}^{n} w_j f(x_j), \tag{23}$$

  kjer so točke x_j enakomerno razporejeni vozli, w_j pa uteži.

- Vemo pa že iz poglavja o interpolacijskih polinomih, da ekvidistantne točke niso vedno najboljša izbira.

- Rešili se bomo ekvidistantnih vozlov v kvadraturnih formulah.

- V formuli (23) bomo izbirali vozle in koeficiente na optimalen način, tako da maksimiziramo stopnjo natančnosti, tj. integracijsko pravilo bo točno za polinome najvišjih možnih stopenj.

- Imamo $n+1$ prostih točk $x_j \in [a,b]$,

$$a \leqslant x_0 < x_1 < \cdots < x_{n-1} < x_n \leqslant b.$$

  in $n+1$ realnih koeficientov w_j, tj. skupaj $2n+2$ neznank.

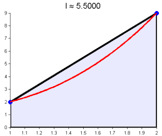
Primer najboljših vozlov za interval $[-1, 1]$

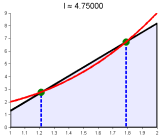
Oglejmo si primer $n = 1$ (tj. 2 točki) na primeru intervala $[-1, 1]$. Poiščimo w_0, w_1, x_0, x_1, tako da velja

$$\int_{-1}^{1} f(x)\,dx \approx w_0 f(x_0) + w_1 f(x_1),$$

pri čemer je aproksimacija kar se da točna.

$$\int_{1}^{2} (x^3 + 1)\,dx = \left[\frac{x^4}{4} + x\right]_1^2 = 4.75.$$





Cilj: poišči w_0, w_1, x_0, x_1 tako da bi aproksimacija točna za polinome stopnje največ 3:

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3.$$

To pomeni, da mora za vsak $c_0, c_1, c_2, c_3 \in \mathbb{R}$ veljati:

$$\begin{aligned}\int_{-1}^{1} f(x)\,dx &= \int_{-1}^{1} \left(c_0 + c_1 x + c_2 x^2 + c_3 x^3\right)\,dx \\ &= w_0 \left(c_0 + c_1 x_0 + c_2 x_0^2 + c_3 x_0^3\right) + w_1 \left(c_0 + c_1 x_1 + c_2 x_1^2 + c_3 x_1^3\right).\end{aligned}$$

Desno stran preuredimo na konstantne, linearne, kvadratične in kubične člene, ter dobimo, da je naslednji izraz

$$c_0\left(w_0+w_1-\int_{-1}^{1} 1\,dx\right)+c_1\left(w_0x_0+w_1x_1-\int_{-1}^{1} x\,dx\right)$$
$$+c_2\left(w_0x_0^2+w_1x_1^2-\int_{-1}^{1} x^2\,dx\right)+c_3\left(w_0x_0^3+w_1x_1^3-\int_{-1}^{1} x^3\,dx\right).$$

ničelen. Ker so koeficienti c_0, c_1, c_2 in c_3 poljubni, morajo biti koeficienti pri njih ničelni.

Od tod sledi:

$$w_0+w_1=\int_{-1}^{1} 1\,dx=2 \qquad w_0x_0+w_1x_1=\int_{-1}^{1} x\,dx=0$$
$$w_0x_0^2+w_1x_1^2=\int_{-1}^{1} x^2\,dx=\frac{2}{3} \qquad w_0x_0^3+w_1x_1^3=\int_{-1}^{1} x^3\,dx=0$$

Z nekaj algebre pridemo do:

$$w_0=1 \quad w_1=1 \quad x_0=-\frac{\sqrt{3}}{3} \quad x_1=\frac{\sqrt{3}}{3}$$

Zato:

$$\boxed{\int_{-1}^{1} f(x)\,dx \approx f\left(-\frac{\sqrt{3}}{3}\right)+f\left(\frac{\sqrt{3}}{3}\right)}.$$

Posplošitev na interval $[a, b]$

Z linearno substitucijo

$$t = a_0 + a_1 x, \quad t(a) = -1, \ t(b) = 1,$$

preslikamo interval $[a, b]$ na $[-1, 1]$.

Velja $a_0 = -\frac{b+a}{b-a}$ in $a_1 = \frac{2}{b-a}$ ter

$$x = \frac{b-a}{2} t + \frac{b+a}{2}, \quad dx = \frac{b-a}{2} dt.$$

Sledi:

$$\boxed{\int_a^b f(x)\, dx = \int_{-1}^1 f\left(\frac{(b-a)t + b + a}{2}\right) \frac{b-a}{2}\, dt}$$

in lahko uporabimo kvadraturno formulo nad $[-1, 1]$.

Z uporabo dveh točk, $n = 1$, smo dobili točen integral za polinome stopnje največ $2 \cdot 1 + 1 = 3$.

Razširitev Gaussovih kvadraturnih formul

Sedaj je naš cilj razširiti zgornje pravilo tako, da bo delovalo za polinome višje stopnje, tj. z vsaki dodanim parom vozla in uteži želimo povečati točnost za dve stopnji.

Velja:

- Smiselno kvadraturno pravilo za integracijo nad intervalom $[-1, 1]$ na enem vozlu bi uporabilo $x = 0$. To pa je ničla funkcije $\boxed{\phi(x) = x}$.
- Kvadraturo pravilo na dveh točkah $\pm\frac{1}{\sqrt{3}}$ smo dobili za ničli funkcije

$$\boxed{\phi(x) = 3x^2 - 1}.$$

- Kako nadaljevati?

Izrek (Gauss)

Naj bo $q(x)$ netrivialen polinom stopnje $n+1$, tako da je

$$\int_a^b x^k q(x)dx = 0 \qquad \text{za vsak } k = 0, 1, \ldots, n$$

in naj bodo $x_0, x_1, \ldots, x_n$ ničle funkcije $q(x)$. Potem velja

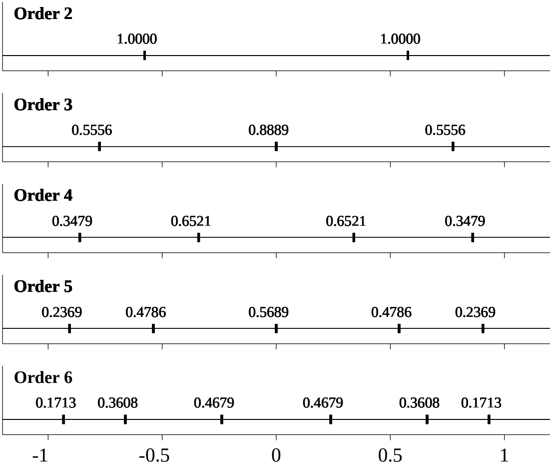
$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i),$$

kjer je

$$A_i = \int_a^b \ell_i(x)dx \text{ za } i = 0, \ldots, n,$$

pri čemer ℓ_i označuje i-ti Lagrangeov bazni polinom na točkah $x_0, \ldots, x_n$, pravilo pa je točno za polinome stopnje največ $2n+1$.

Izgled vozlov



Algoritem: klik Primer: klik

Integracija v več dimenzijah

Ločili bomo primera:

1. Zanima nas $\int_\Omega f(x,y)\,dx\,dy$, kjer je $\Omega = [a,b]\times[c,d] \subseteq \mathbb{R}^2$ pravokotnik.
2. Zanima nas $\int_\Omega f(\underline{x})\,d\Omega$, kjer je $\Omega \subseteq \mathbb{R}^d$ poljubno območje v $\mathbb{R}^d$.

V prvem primeru lahko uporabimo dve sestavljeni pravili za vsako spremenljivko posebej. Naj bosta

$$a = x_0 < x_1 < x_2 < \ldots < x_n = b$$

in

$$c = y_0 < y_1 < y_2 < \ldots < y_n = d$$

delitvi intervalov $[a,b]$ in $[c,d]$ na n enakih delov in $h = \frac{b-a}{n}$, $k = \frac{d-c}{n}$. Če uporabimo sestavljeni trapezni pravili dobimo:

$$\int_\Omega f(x,y)\,dx\,dy = \int_c^d \int_a^b f(x,y)\,dx\,dy = \int_c^d \left(\sum_{i=0}^{n-1} \frac{h}{2}(f(x_i,y) + f(x_{i+1},y)) \right) dy$$

$$= \frac{hk}{4} \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} (f(x_i,y_j) + f(x_i,y_{j+1}) + f(x_{i+1},y_j) + f(x_{i+1},y_{j+1}))$$

Izkaže se, da pomnožimo istoležne koeficiente v tabeli funkcijskih vrednosti

$$\left(\begin{array}{cccccc} f(x_n,y_0) & f(x_n,y_1) & f(x_n,y_2) & \cdots & f(x_n,y_{n-1}) & f(x_n,y_n) \\ f(x_{n-1},y_0) & f(x_{n-1},y_1) & f(x_{n-1},y_2) & \cdots & f(x_{n-1},y_{n-1}) & f(x_{n-1},y_n) \\ \vdots & \cdots & \cdots & \cdots & \vdots & \\ f(x_1,y_0) & f(x_1,y_1) & f(x_1,y_2) & \cdots & f(x_1,y_{n-1}) & f(x_1,y_n) \\ f(x_0,y_0) & f(x_0,y_1) & f(x_0,y_2) & \cdots & f(x_1,y_{n-1}) & f(x_0,y_n) \end{array}\right),$$

s tabelo koeficientov

$$\frac{hk}{4}\cdot\left(\begin{array}{cccccc} 1 & 2 & 2 & \cdots & 2 & 1 \\ 2 & 4 & 4 & \cdots & 4 & 2 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 2 & 4 & 4 & \cdots & 4 & 2 \\ 1 & 2 & 2 & \cdots & 2 & 1 \end{array}\right) = \frac{hk}{4}\left(\begin{array}{c} 1 \\ 2 \\ \vdots \\ \vdots \\ 2 \\ 1 \end{array}\right)\left(\begin{array}{cccccc} 1 & 2 & \cdots & \cdots & 2 & 1 \end{array}\right)$$

in seštejemo vse vrednosti v dobljeni matriki.

Če bi namesto sestavljenega trapeznega pravila uporabili Simpsonovega, bi morali za tabelo koeficientov vzeti

$$\frac{hk}{9}\cdot uu^T,$$

V drugem primeru uporabimo Monte Carlo metode, ki temeljijo na dejstvu, da velja

$$\int_\Omega f(x_1, \ldots, x_d)\, d\Omega = \mathrm{Vol}(\Omega) \cdot E_\Omega(f(X_1, \ldots, X_d)),$$

kjer je $\mathrm{Vol}(\Omega) = \int_\Omega 1\, d\Omega$ volumen območja Ω, $X = (X_1, \ldots, X_d) : \Omega \to \Omega$ slučajni vektor, E_Ω pa pričakovana vrednost slučajne spremenljivke $f(X_1, \ldots, X_d)$.

Naključno moramo torej vzorčiti na območju Ω, nato pa izračunati povprečje funkcijskih vrednosti.

Za dovolj veliko naključnih točk bo povprečje dobra ocena za vrednost integrala.

Reševanje diferencialnih enačb

$$y' = f(x, y), \quad y(x_0) = y_0$$

- Eulerjeva metoda
- Runge-Kutta metode
- Adaptivne metode: DOPRI5, Cash-Fehlberg

Diferencialna enačba

Diferencialna enačba (DE) je enačba oblike:

$$F(t, x, \dot{x}, \ddot{x}, \ldots, x^{(n)}) = 0, \tag{25}$$

kjer je $x = x(t)$ odvisna spremenljivka, t neodvisna spremenljivka, $\dot{x}$ pa označuje odvod x po t.

Če je $y = y(x)$ odvisna spremenljivka, x pa neodvisna, potem je DE oblike

$$F(x, y', y'', \ldots, y^{(n)}) = 0. \tag{26}$$

Ključna lasnost DE je ta, da poleg neodvisne spremenljvike t (oz. x) in odvisne spremenljivke x (oz. y) nastopajo še odvodi odvisne spremenljivke $\dot{x}$, …, $x^{(n)}$ (oz. y', …, $y^{(n)}$).

Rešitev DE je (dovoljkrat odvedljiva) funkcija, ki zadošča enačbi (25) oz. (26) na definicijskem območju $\mathcal{D}$ neodvisne spremenljivke.

Red DE je stopnja najvišjega odvoda, ki nastopa v DE.

Primeri. $y' = y$, $y' + 5xy = 3x^2$, $\dot{x} + x = 0$, $\ddot{x} + a\dot{x} + bx = A\cos\omega t$.

Splošna rešitev diferencialne enačbe reda n je družina funkcij, odvisna od n parametrov, ki so vse rešitve diferencialne enačbe.

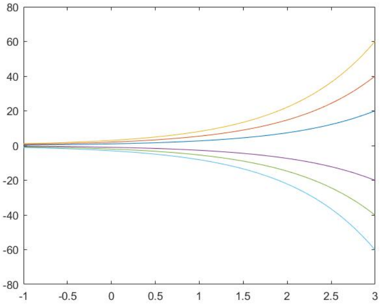
Primer

Rešimo DE $y' = y$.

$$\frac{dy}{dx} = y \quad \Rightarrow \quad \frac{dy}{y} = dx \quad \Rightarrow \quad \int \frac{dy}{y} = \int dx$$

$$\Rightarrow \quad \log(|y|) = x + C, \quad C \in \mathbb{R}$$

$$\Rightarrow \quad y = Ke^x, \quad K \in \mathbb{R}$$



Partikularna rešitev je posamezna rešitev iz te družine.
Določena je z n dodatnimi pogoji, na primer z začetnimi pogoji:

$$x(t_0) = a_0, \quad \dot{x}(t_0) = a_1, \ldots, \quad x^{(n-1)}(t_0) = a_{n-1}$$

Zelo malo DE je analitično rešljivih. Mednje sodijo:

- DE z ločljivima spremenljivkama
- Linearne DE
- DE zelo posebne oblike

Večina DE ni analitično rešljivih. Te rešujemo numerično.

Diferencialna enačba 1. reda z ločljivima spremenljivkama

$$\dot{x} = f(t)g(x)$$

Enačbo rešimo tako, da vpeljemo $\dot{x} = \dfrac{dx}{dt}$ in ločimo spremenljivki:

$$\frac{dx}{dt} = f(t)g(x), \quad \frac{dx}{g(x)} = f(t)dt$$

in potem integriramo

$$\int \frac{dx}{g(x)} = \int f(t)dt.$$

Linearna diferencialna enačba

$$y' + f(x)y = g(x) \tag{27}$$

Pravimo, da je enačba homogena, če je $g(x) = 0$ in nehomogena, če je $g(x) \neq 0$.

1. Rešimo homogeni del $y' + f(x)y = 0$ s pomočjo ločitve spremenljivk. Dobimo rešitev

$$y = Ce^{-\int f(x)dx} = C\, z(x)$$

2. Metoda variacije konstante
   - V (27) vstavimo $y = C(x)\, z(x)$ in rešimo na $C(x)$.
   - Tako dobljeni C vstavimo v rešitev homogenega dela.

Numerično reševanje DE

Na intervalu $[a, b]$ rešujemo DE prvega reda

$$y' = f(x, y), \quad y(a) = y_0. \tag{28}$$

Interval $[a, b]$ razdelimo z zaporedjem točk

$$a = x_0 < x_1 < x_2 < \ldots < x_n = b.$$

Z y_i označimo približek za rešitev (28) v točki x_i. Označimo dolžino koraka z $h_i := x_{i+1} - x_i$.

Razliko med približkom in točno rešitvijo v x_i pišemo z $g_i = y_i - y(x_i)$ in jo imenujemo globalna napaka v x_i.

Razliko med približkom in točno rešitvijo DE

$$z' = f(x, z), \quad z(x_{i-1}) = y_{i-1} \tag{29}$$

v x_i pišemo z $\ell_i = y_i - z(x_i)$ in jo imenujemo lokalna napaka v x_i.

Globalno napako lahko ocenimo s pomočjo lokalnih napak:

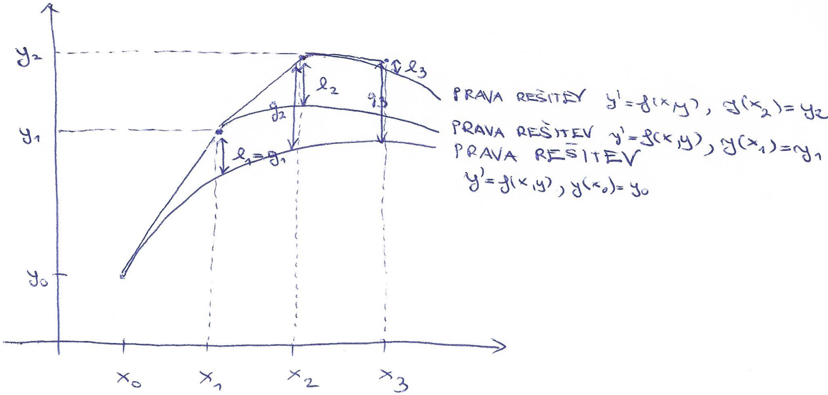
$$|g_i| \leqslant |\ell_1| + |\ell_2| + \ldots + |\ell_i|.$$

Red metode je število $p \in \mathbb{N}$, ki zadošča $\boxed{\ell_i = Ch_i^{p+1} + \mathcal{O}(h_i^{p+2})}$

Eulerjeva metoda

Pri tej metodi v vsaki točki x_i uporabimo linearno aproksimacijo funkcije. Rešitev na intervalu $[x_i, x_{i+1}]$ nadomestimo z odsekom tangente na graf rešitve v točki x_i:

$$y_{i+1} = y_i + h_i \cdot f(x_i, y_i).$$



```
y = y_0
x = x_0
h = (b - a)/n
for i = 1, ..., n-1
    y = y + h · f(x, y)
    x = x + h
end
```

Ker je

$$y(x+h) = \underbrace{y(x) + hy'(x)}_{\text{upoštevamo}} + \underbrace{\frac{h^2}{2} y''(\xi)}_{\text{napaka}}, \quad \xi \in [x, x+h],$$

je red Eulerjeve metode 1.

Algoritem: klik