Numerična matematika

izročki predavanj

Aljaž Zalar

Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Viri

Viri v slovenščini:

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

Tuji viri - numerična linearna algebra:

- G.H. Golub, C.F. Van Loan: Matrix Computations, 3rd edition, Johns Hopkins Univ. Press, Baltimore, 1996.
- L.N. Trefethen, D. Bau: Numerical Linear Algebra, SIAM, Philadelphia, 1997.
- J.W. Demmel: Applied Numerical Linear Algebra, SIAM, 1997.

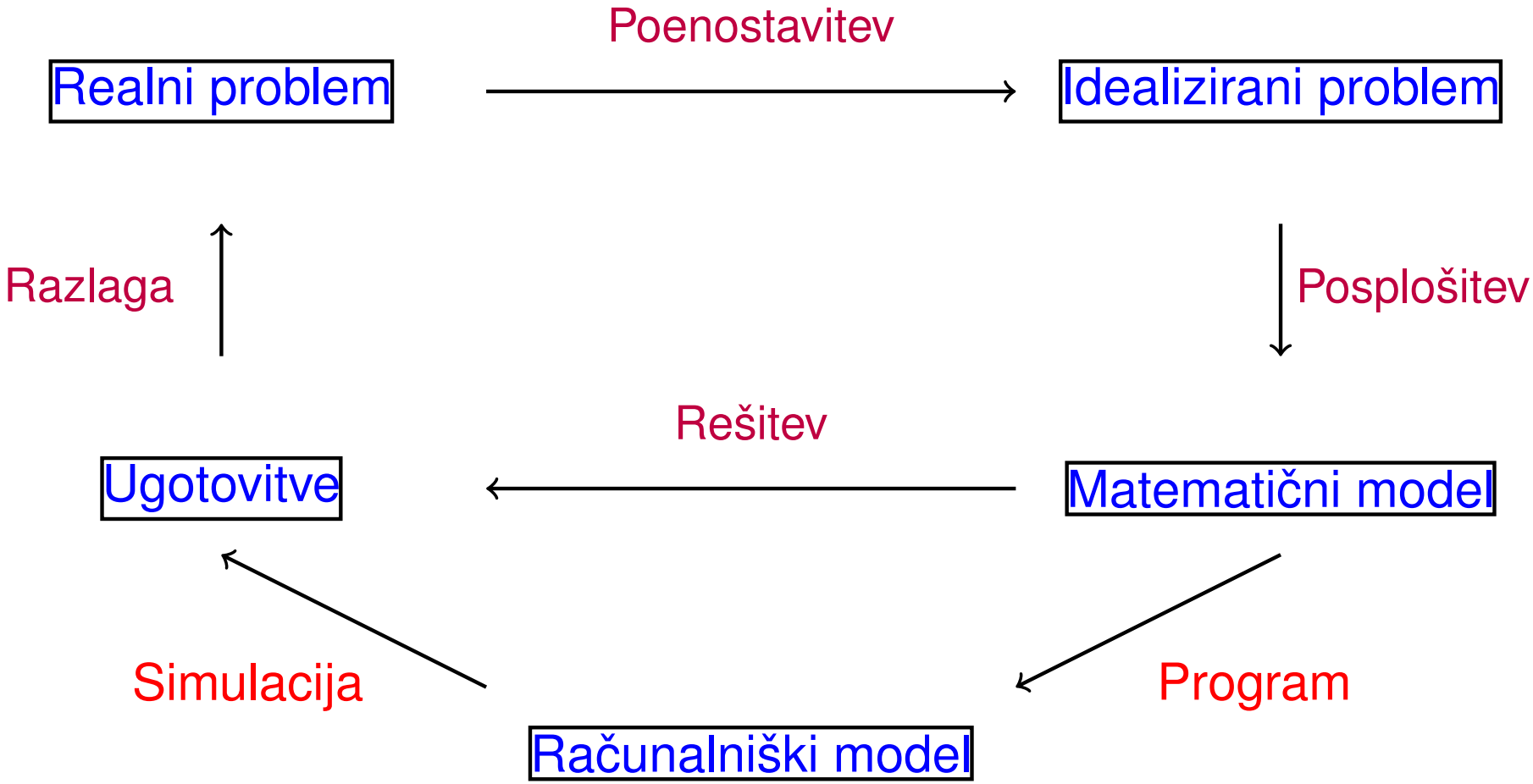
Tuji viri - numerična analiza:

- 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.
- D.R. Kincaid, E.W. Cheney: Numerical Analysis, Mathematics of Scientific Computing, 3rd edition, Brooks/Cole, Pacific Grove, 2002.

Obveznosti

- Predavanja: 3 ure na teden
- Vaje: 2 uri na teden
- 3 domače naloge: 50% ocene
- Izpit iz teorije: 50% ocene

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
   - QR razcep za predoločene sisteme
   - Iterativne metode - Jacobijeva in Gauss-Seidlova iteracija
3. Iskanje lastnih vrednosti matrik
   - Potenčna metoda
   - PageRank algoritem
   - QR iteracija in njene izboljšave
4. Reševanje (sistemov) nelinearnih enačb in optimizacija
   - Tangentna oz. Newtonova metoda
   - Metoda fiksne točke
5. Aproksimacija in interpolacija
   - Lagrangeov in Newtonov interpolacijski polinom
   - Aproksimacija po metodi najmanjših kvadratov

6. Numerično odvajanje in integriranje
   - Trapezna metoda
   - Simpsonova metoda
   - Rombergova metoda
   - Avtomatsko odvajanje
7. 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 računanjem 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=1.3877787807814446e-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

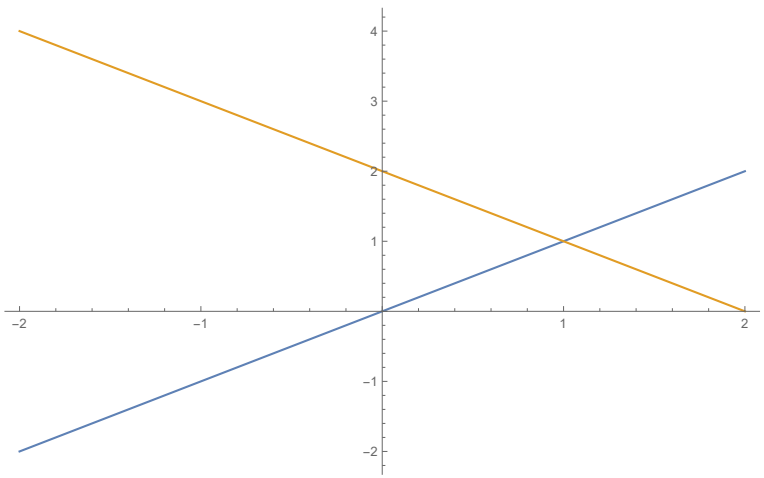
$$\begin{aligned} x + y = 2 \quad &\rightarrow \quad x + y = 1.9999 \\ x - y = 0 \quad &\rightarrow \quad x - y = 0.0002 \end{aligned}$$

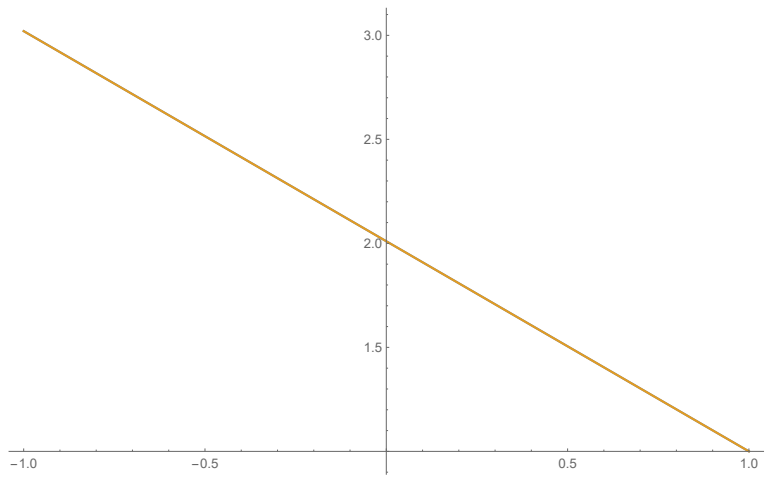
ima rešitvi $x = y = 1$ *oz.* $x = 1.00005$ *in* $y = 0.99985$. *Problem je neobčutljiv, saj je šlo za spremembo za isti velikostni razred.*

Sistem in njegova perturbacija

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

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 $9.5 \cdot 10^{-8}\ \mathrm{s}$. Po 100 urah računanja je bila napaka $9.5 \cdot 10^{-8}\ \mathrm{s} \cdot 100 \cdot 60 \cdot 60 \cdot 10 = 0.34\ \mathrm{s}$. Ker je hitrost rakete $1.676\ \mathrm{m/s}$, je bila pozicija rakete za več kot $500\ \mathrm{m}$ 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

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

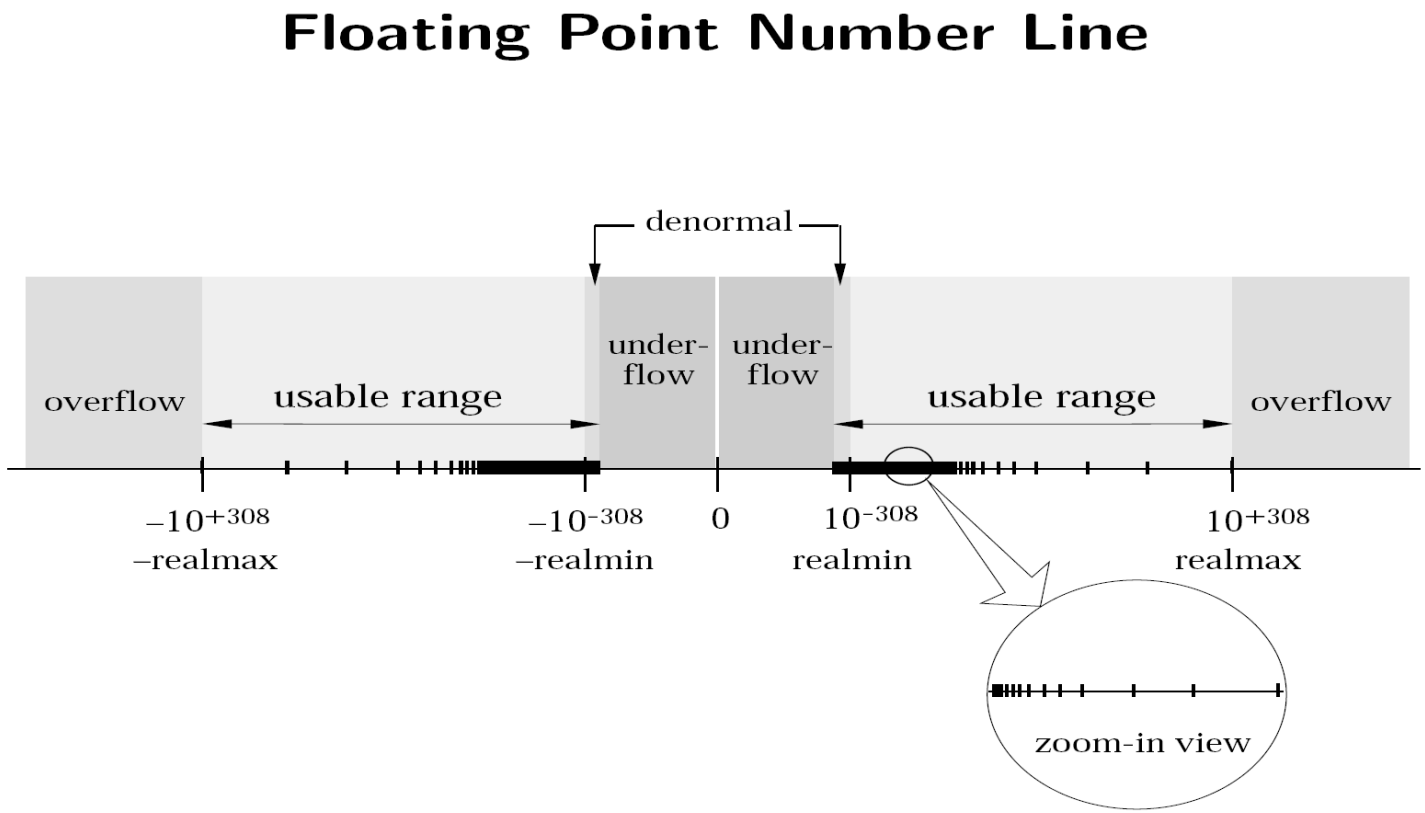
kjer je

- 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* $+(0.812345)_{10} \times 10^{-3}$.

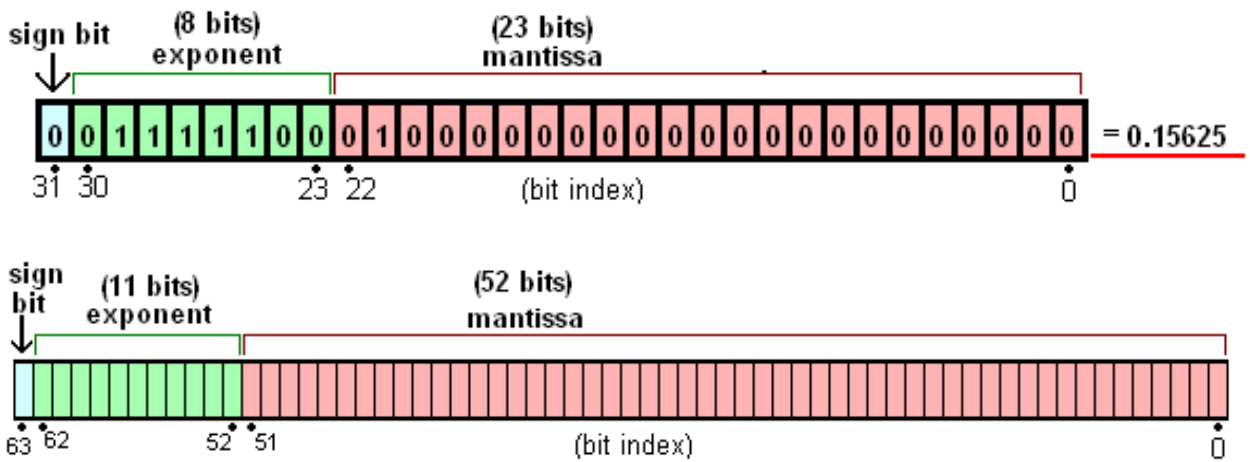
Prekoračitev in podkoračitev



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

IEEE standard

- IEEE Enojna natančnost: števila so predstavljena z 32 biti.
- IEEE Dvojna natančnost: števila so predstavljena z 64 biti.



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_-:

$$\begin{aligned} x &= (0.1 b_2 b_3 \ldots b_m b_{m+1})_2 \times 2^e, \\ x_- &= (0.1 b_2 b_3 \ldots b_m)_2 \times 2^e, \\ x_+ &= \left((0.1 b_2 b_3 \ldots b_m)_2 + 2^{-m}\right) \times 2^e, \end{aligned}$$

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

Absolutna napaka:

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

Relativna napaka:

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

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

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

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

$$(a+b)+c = (a+b+c)(1+\epsilon_3) \quad \text{kjer je} \quad \epsilon_3 \approx \frac{a+b}{a+b+c}\delta_1+\delta_2,$$

$$a+(b+c) = (a+b+c)(1+\epsilon_4) \quad \text{kjer je} \quad \epsilon_4 \approx \frac{b+c}{a+b+c}\delta_3+\delta_4.$$

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

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

Primer ($\sin \frac{\pi}{10}$ računamo v desetiškem sistemu z $m = 4$)

- D_n: $fl(\frac{\pi}{10}) = 0.3142 \cdot 10^0$. *Ocenimo:* $|D_n| \approx \sin'(\frac{\pi}{10})|x - fl(x)| \leqslant \frac{1}{2} \cdot 10^{-4}$.
- D_m : $\sin x \approx x - x^3/6$. *Ocenimo:* $|D_m|| \leqslant x^5/120 = 2.6 \cdot 10^{-5}$.
- D_z: $fl(x - fl(fl(fl(x \cdot x) \cdot x)/6))$. *Ocenimo:* $|D_z| \leqslant 3.0 \cdot 10^{-5}$.

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 = -1.110223024625157e-16
```

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 kar koli. 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 $|\delta_i| \leqslant u$. 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 $|\delta_i| \leqslant u$. 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

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

$Ax = b$

- Vektorske in matrične norme
- Pogojenostno število $\kappa(A)$
- Direktne metode za reševanje
  - *LU* razcep
  - Pivotna rast $\rho(A)$
  - Razcep Choleskega
- Iterativne metode za reševanje
  - Jacobi, Gauss-Seidel, SOR, SSOR, konjugirani gradienti
- Predoločeni sistemi
  - *QR* razcep
  - Householderjeva zrcaljenja
  - *SVD* razcep

Vektorska norma je preslikava $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$, ki zadošča:

1. Pozitivna definitnost: $\|x\| \geqslant 0$ za vsak $x \in \mathbb{C}^n$ in $\|x\| = 0 \Leftrightarrow x = 0$.
2. Homogenost: $\|\alpha x\| = |\alpha| \, \|x\|$ za vsaka $\alpha \in \mathbb{C}$ in $x \in \mathbb{C}^n$
3. Trikotniška neenakost: $\|x + y\| \leqslant \|x\| + \|y\|$ za vsaka $x, y \in \mathbb{C}^n$.

Primer

Naj bo $x = (x_1, \ldots, x_n) \in \mathbb{C}^n$.

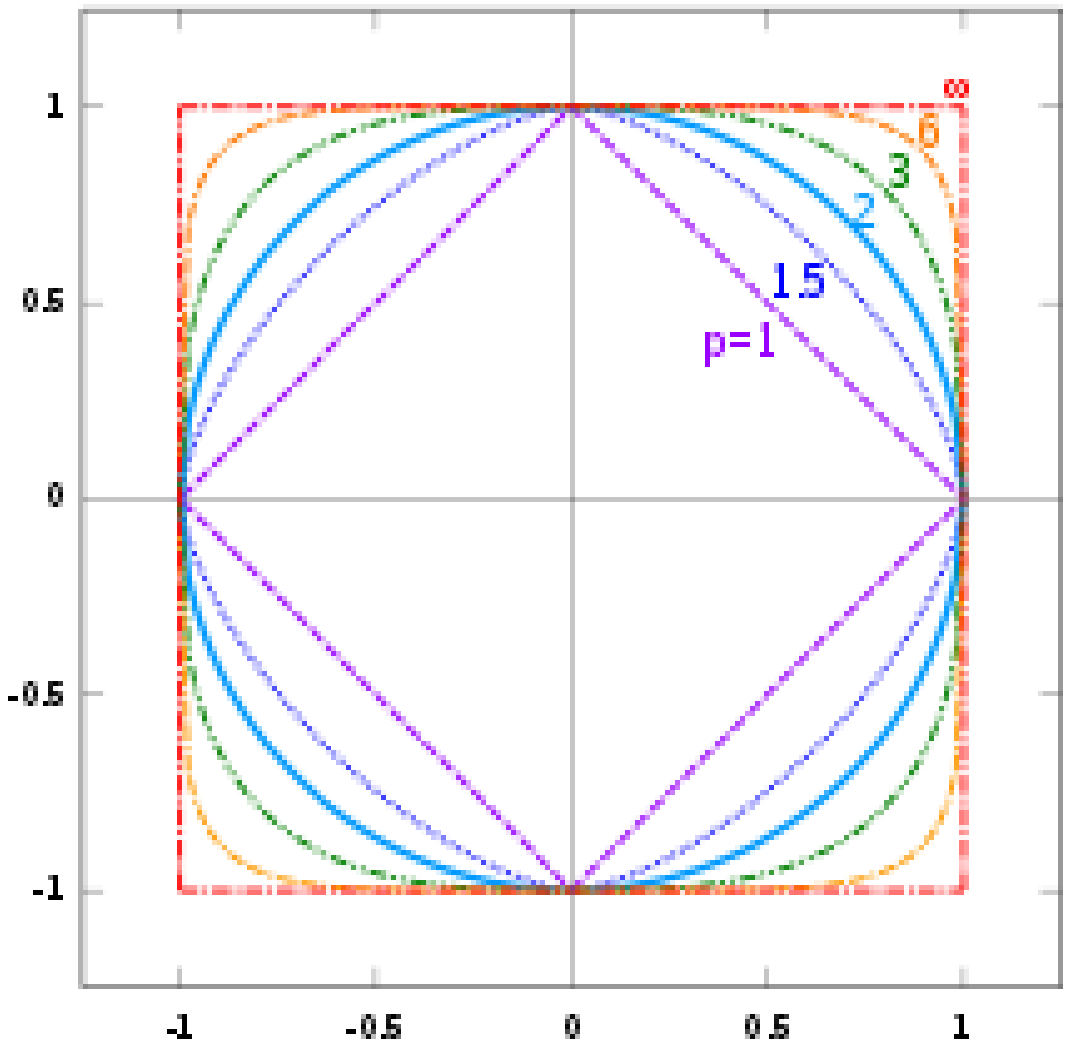
- *p–norma,* $p \in \mathbb{N}$*:*

$$\|x\|_p := \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p}.$$

- *Supremum norma:*

$$\|x\|_\infty = \max(|x_1|, \ldots, |x_n|).$$

Enotske krožnice v različnih normah



Matrična norma je preslikava $\|\cdot\| : \mathbb{C}^{n\times n} \to \mathbb{R}$, ki zadošča:

1. Pozitivna definitnost: $\|A\| \geqslant 0$ za vsak $A \in \mathbb{C}^{n\times n}$ in $\|A\| = 0 \Leftrightarrow A = 0$.
2. Homogenost: $\|\alpha A\| = |\alpha| \, \|A\|$ za vsaka $\alpha \in \mathbb{C}$ in $A \in \mathbb{C}^{n\times n}$.
3. Trikotniška neenakost: $\|A + B\| \leqslant \|A\| + \|B\|$ za vsaka $A, B \in \mathbb{C}^{n\times n}$.
4. Submultiplikativnost: $\|AB\| \leqslant \|A\| \, \|B\|$ za vsaka $A, B \in \mathbb{C}^{n\times n}$.

Trditev

Naj bo $\|\cdot\|_*$ vektorska norma na $\mathbb{C}^n$. Potem predpis

$$\|A\|_* := \max_{\|x\|=1} \|Ax\|_* = \max_{x\neq 0} \frac{\|Ax\|_*}{\|x\|_*}.$$

določa matrično normo na $\mathbb{C}^{n\times n}$.

Dokaz: klik

Naj bo $A = [a_{ij}]_{i,j=1,\ldots,n} \in \mathbb{C}^{n\times n}$ matrika. Nekaj matričnih norm:

1. 1–norma:

$$\|A\|_1 = \max_{j=1,\ldots,n} \Big(\sum_{i=1}^{n} |a_{ij}| \Big). \quad \text{Dokaz: klik}$$

2. Spektralna norma: Tu $\lambda_j(X)$ označuje j–to lastno vrednost matrike X.

$$\|A\|_2 = \sqrt{\max_{j=1,\ldots,n} \lambda_j(A^T A)}.$$

3. Frobeniusova norma:

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2}.$$

4. Supremum norma:

$$\|A\|_\infty = \max_{i=1,\ldots,n} \Big(\sum_{j=1}^{n} |a_{ij}| \Big).$$

Zakaj imeti več matričnih norm?

Nekatere norme je bistveno zahtevneje izračunati od ostalih. Zahtevno je npr. določanje spektralne norme $\|\cdot\|_2$, saj je računanje lastnih vrednosti zahtevna naloga. Poceni pa je izračunati 1–normo, ∞–normo in F–normo. Iz različnih ocen, kot so

$$\begin{aligned}
\frac{1}{\sqrt{n}}\|A\|_F &\leqslant \|A\|_2 \leqslant \|A\|_F, \\
\frac{1}{\sqrt{n}}\|A\|_1 &\leqslant \|A\|_2 \leqslant \sqrt{n}\|A\|_1, \\
\frac{1}{\sqrt{n}}\|A\|_\infty &\leqslant \|A\|_2 \leqslant \sqrt{n}\|A\|_\infty,
\end{aligned}$$

pa lahko dobro ocenimo $\|A\|_2$.

Občutljivost sistema $Ax = b$

Zanima nas, kako na spremembo rešitve x vpliva napaka v začetnih podatkih, tj. napaka v A in b.

Zanima nas torej, kako velik je Δx v primeru majhnih perturbacij ΔA in Δb v rešitvi

$$(A + \Delta A)(x + \Delta x) = b + \Delta b. \tag{1}$$

Radi bi ocenili relativno napako $\frac{\|\Delta x\|}{\|x\|}$, kjer je $\|\cdot\|$ neka vektorska norma.

Izberimo vektorsko normo $\|\cdot\|$. Definirajmo občutljivost oz. pogojenostno število obrnljive matrike A v normi $\|\cdot\|$:

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|.$$

$\kappa(A)$ meri občutljivost sistema $Ax = b$

Izrek

Naj bo A v (1) *obrnljiva matrika.*

1. *Privzemimo, da je $\Delta A = 0$. Potem velja:*

$$\frac{\|\Delta x\|}{\|x\|} \leqslant \kappa(A)\frac{\|\Delta b\|}{\|b\|}.$$

2. *Naj bo $\Delta A \neq 0$, naj za identično matriko I velja $\|I\| = 1$ in naj bo še $\left\|A^{-1}\right\| \|\Delta A\| < 1$. Potem velja:*

$$\frac{\|\Delta x\|}{\|x\|} \leqslant \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|}\right).$$

Dokaz: klik

Primer

1. *Če se spomnimo primera računanja prečišča dveh premic iz prvih predavanj, lahko vidimo, da je vprašanje občutljivosti glede na začetne podatke v resnici vprašanje občutljivosti matrik*

$$A_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad in \quad A_2 = \begin{pmatrix} 1.00 & 0.99 \\ 0.99 & 0.98 \end{pmatrix}.$$

Za njiju velja:

$$\kappa_1(A_1) = \kappa_F(A_1) = \kappa_\infty(A_1) = 2, \quad \kappa_2(A_1) = 1,$$
$$\kappa_1(A_2) = \kappa_\infty(A_2) = 3.96 \cdot 10^4, \quad \kappa_2(A_2) = 2, \quad \kappa_F(A_2) = 3.92 \cdot 10^4,$$

kjer κ_* označuje občutljivost v matrični normi $*$. Kot smo se geometrijsko prepričali, je drugi sistem res občutljiv, prvi pa ne.

2. *Primer zelo občutljive matrike je Hilbertova matrika*

$$H_n = \left[\frac{1}{i+j-1} \right]_{i,j} \in \mathbb{R}^{n \times n}.$$

Ta se pojavi pri iskanju polinoma, ki se v normi $\|f\| = \sqrt{\int_0^1 f^2 dx}$ najbolje prilega dani funkciji, saj je $\int_0^1 x^{i+j} dx = \frac{1}{i+j+1}$. Velja $\kappa_2(H_5) \approx 4.8 \cdot 10^5$, za naključno 5×5 matriko pa velja $\kappa_2 \approx 100$.

Primer

1. *Če z Matlabom z ukazom* $\backslash$ *rešimo sistem* $H_{15}x = v$*, kjer je*

$$v = H_{15} \cdot [1, \ldots, 1]^T,$$

bi morali dobiti za rezultat $x = [1, \ldots, 1]$*. Toda*

$$\|x - [1, \ldots, 1]^T\|_2 = 5.3 \cdot 10^{-3}.$$

Koda primera: klik

Direktne metode

$Ax = b$

- Gaussova-eliminacija
- LU razcep
- Pivotiranje
- Pivotna rast
- Razcep Choleskega

Reševanje kvadratnih linearnih sistemov

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

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

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

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

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

Zanima nas, ali obstaja linearna kombinacija (2), 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?

  Vemo že, da so slabe matrike z velikim $\kappa(A)$.

  Kaj pa, če $\kappa(A)$ *ni velik?*

- 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

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

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

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

Rešimo z obratno substitucijo

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

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

```
1   -n × n matrika A = [a_ij]_ij in n × 1 vektor b = [b_i]_i
2   -preoblikujemo [A|b] v zgornjetrikotno z GE
3
4   for k = 1 ... n − 1
5     for i = k + 1 ... n
6       xmult = a_ik / a_kk
7       a_ik = 0
8       for j = k + 1 ... n
9         a_ij = a_ij − (xmult) a_kj
10      end
11      b_i = b_i − (xmult) b_k
12    end
13  end
```

Izrek

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

$$\frac{2}{3}n^3 + \mathcal{O}(n^2). \quad \textit{Dokaz: klik}$$

Obratna substitucija in število operacij

```
1   -zgornjetrikotna n × n matrika U = [u_ij]_{i,j}, vektor
        c = [c_i]_i
2   -resimo sistem Ux = c
3
4   x_n = c_n/u_nn
5   for i = n − 1 . . . 1
6      s = c_i
7      for j = i + 1 . . . n
8         s = s − u_ij x_j
9      end
10     x_i = s/u_ii
11  end
```

Izrek

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

n^2. *Dokaz: klik*

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.

Eliminacijske matrike

Kako v matrični obliki zapišemo prištevanje večkratnika neke vrstice matrike k drugi?

Trditev

Prištejmo z $\alpha \in \mathbb{R}$ pomnoženo i–to vrstico matrike A njeni j–ti vrstici, kjer je $i \neq j$, in dobljeno matriko označimo z $\widetilde{A}$. Velja:

$$\widetilde{A} = A + \alpha \cdot E_{ij}A = (I_n + \alpha E_{ij})A,$$

kjer je E_{ij} matrika, ki ima v i–ti vrstici in j–tem stolpce 1, drugje pa 0, I_n pa identična matrika.

V prvem stolpcu pridelamo 0 pod diagonalo med GE na naslednji način:

$$A \mapsto (I_n - \frac{a_{21}}{a_{11}}E_{21})A \mapsto (I_n - \frac{a_{31}}{a_{11}}E_{31})(I_n - \frac{a_{21}}{a_{11}}E_{21})A \mapsto \cdots$$
$$\mapsto \underbrace{(I_n - \frac{a_{n1}}{a_{11}}E_{n1}) \cdots (I_n - \frac{a_{21}}{a_{11}}E_{21})}_{L_1} A =: A^{(1)}.$$

Eliminacijske matrike

Če odpravimo oklepaje v matriki L_1, in upoštevamo $E_{i1}E_{j1} = 0$, saj je $i, j > 1$, dobimo spodnjetrikotno matriko

$$L_1 = I_n - \frac{a_{21}}{a_{11}}E_{21} - \ldots - \frac{a_{n1}}{a_{11}}E_{n1}.$$

Z istim premislekom v drugem stolpcu nove matrike pridelamo 0 z množenjem $A^{(1)} = [a_{ij}^{(1)}]_{i,j}$ s spodnjetrikotno matriko

$$L_2 = I_n - \frac{a_{32}^{(1)}}{a_{22}^{(1)}}E_{32} - \ldots - \frac{a_{n2}^{(1)}}{a_{22}^{(1)}}E_{n2}.$$

Dobimo:

$$A^{(2)} = L_2A^{(1)} = L_2L_1A.$$

Na koncu GE dobimo s tem postopkom zgornjetrikotno matriko

$$U = L_{n-1}\cdots L_1A. \tag{3}$$

Eliminacijske matrike

Iz (3) bi radi izrazili A. Preprost račun pokaže, da inverze L_i^{-1} eliminacijskih matrik dobimo tako, da vse minuse v defincji L_i spremenimo v pluse:

$$L_1^{-1} = I_n + \frac{a_{21}}{a_{11}} E_{21} + \ldots + \frac{a_{n1}}{a_{11}} E_{n1},$$

$$L_2^{-1} = I_n + \frac{a_{32}^{(1)}}{a_{22}^{(1)}} E_{32} + \ldots + \frac{a_{n2}^{(1)}}{a_{22}^{(1)}} E_{n2},$$

$$\vdots$$

$$L_{n-1}^{-1} = I_n + \frac{a_{n,n-1}^{(n-2)}}{a_{n-1,n-1}^{(n-2)}} E_{n,n-1}$$

Iz (3) dobimo A na naslednji način:

$$A = \underbrace{L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}}_{L} U.$$

Preprost račun pokaže (kjer upoštevamo $E_{ij}E_{k\ell} = 0$ za $j \neq k$), da velja

$$L = I_n + \frac{a_{21}}{a_{11}} E_{21} + \ldots + \frac{a_{n1}}{a_{11}} E_{n1} + \frac{a_{32}^{(1)}}{a_{22}^{(1)}} E_{32} + \ldots + \frac{a_{n,n-1}^{(n-2)}}{a_{n-1,n-1}^{(n-2)}} E_{n,n-1}$$

LU razcep matrike A

```
1   -Vhod: A = [a_ij]_{i,j} n × n matrika.
2   -Izhod: Spodnja trikotna matrika L in zgornja
        trikotna matrika U, da je A = LU
3   -ℓ_ik v spodnjem algoritmu so elementi pod
        diagonalo v L, na diagonali so same 1
4   -preostali elementi a_ij v zgornjem trikotniku so
        elementi matrike U
5
6   for k = 1, ..., n − 1
7     for i = k + 1, ..., n
8       ℓ_ik = a_ik / a_kk
9       for j = k + 1, ..., n
10        a_ij = a_ij − ℓ_ik a_kj
11      end
12    end
13  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

```
1   -Vhod: spodnja trikotna n × n matrika L = [ℓ_ij]_{i,j} in
        vektor b = [b_i]_i
2   -Izhod: resitev y sistema Ly = b
3
4   y_1 = b_1/ℓ_11
5   for i = 2...n
6      s = b_i
7      for j = 1...i − 1
8         s = s − ℓ_ij y_j
9      end
10     y_i = s/ℓ_ii
11  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 razcep 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 $k = 1, \ldots, n$.*

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

Izrek (LU razcep z delnim pivotiranjem)

Če izvedemo Gaussovo eliminacijo z delnim pivotiranjem matrike $A \in \mathbb{R}^{(n-1)\times(n-1)}$ do zgornje trikotne matrike U z zaporedjem transformacij:

$$\begin{aligned} A &\mapsto P_1 A \mapsto L_1 P_1 A_1 \mapsto P_2 L_1 P_1 A_1 \mapsto \cdots \\ &\mapsto L_{n-1} P_{n-1} \cdots L_1 P_1 A =: U, \end{aligned}$$

kjer so P_i permutacijske, L_i pa eliminacijske matrike uporabljene na i–tem koraku postopka. Potem velja

$$PA = LU$$

kjer so

$$P = P_{n-1} P_{n-2} \cdots P_1, \qquad L = \widehat{L}_1^{-1} \widehat{L}_2^{-1} \cdots \widehat{L}_{n-1}^{-1},$$

$$\widehat{L}_i^{-1} = P^{(i)} L_i^{-1} (P^{(i)})^T, \qquad P^{(i)} = P_{n-1} \cdots P_{i+1}.$$

Dokaz: klik

LU razcep z delnim pivotiranjem - algoritem

```
1   -Vhod: A = [a_ij]_{i,j}  n × n matrika
2   -Izhod: permutacijska matrika P, spodnja in
        zgornja trikotna matrika L in U, da je
        PA = LU
3
4   P in L identicni n × n matriki
5   for k = 1,...,n−1
6     poisci q-to in k-to vrstico, ki zadosca
          |a_qk| = max_{k≤p≤n} |a_pk|
7     zamenjaj q-to in k-to vrstico v matrikah A, P
          in strogem spodnjem trikotniku L
8     for i = k+1,...,n
9       ℓ_ik = a_ik / a_kk
10      for j = k+1,...,n
11        a_ij = a_ij − ℓ_ik a_kj
12        end
13    end
14   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

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

Stabilnost LU razcepa matrike A

Iz zgornjega izreka sledi

$$\begin{aligned}\|E\|_\infty &\leqslant 3(n-1)u \cdot \left(\|A\|_\infty + (\|\widehat{L}\|\|\widehat{U}\|)_\infty\right) + \mathcal{O}(u^2)\\ &\leqslant 3(n-1)u \cdot \left(\|A\|_\infty + \|\widehat{L}\|_\infty\|\widehat{U}\|_\infty\right) + \mathcal{O}(u^2)\\ &\leqslant 3(n-1)u\|A\|_\infty + 3(n-1)n\|\widehat{U}\|_\infty + \mathcal{O}(u^2),\end{aligned}$$

kjer smo v drugi neenakosti upoštevali submultiplikativnost, v tretji neenakosti pa to, da so pri LU razcepu z delnim pivotiranjem vsi elementi matrike L navzgor omejeni z 1. Zato velja $\|L\|_\infty \leqslant n$. Torej je relativna napaka v supremum normi navzgor omejena z

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

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

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

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

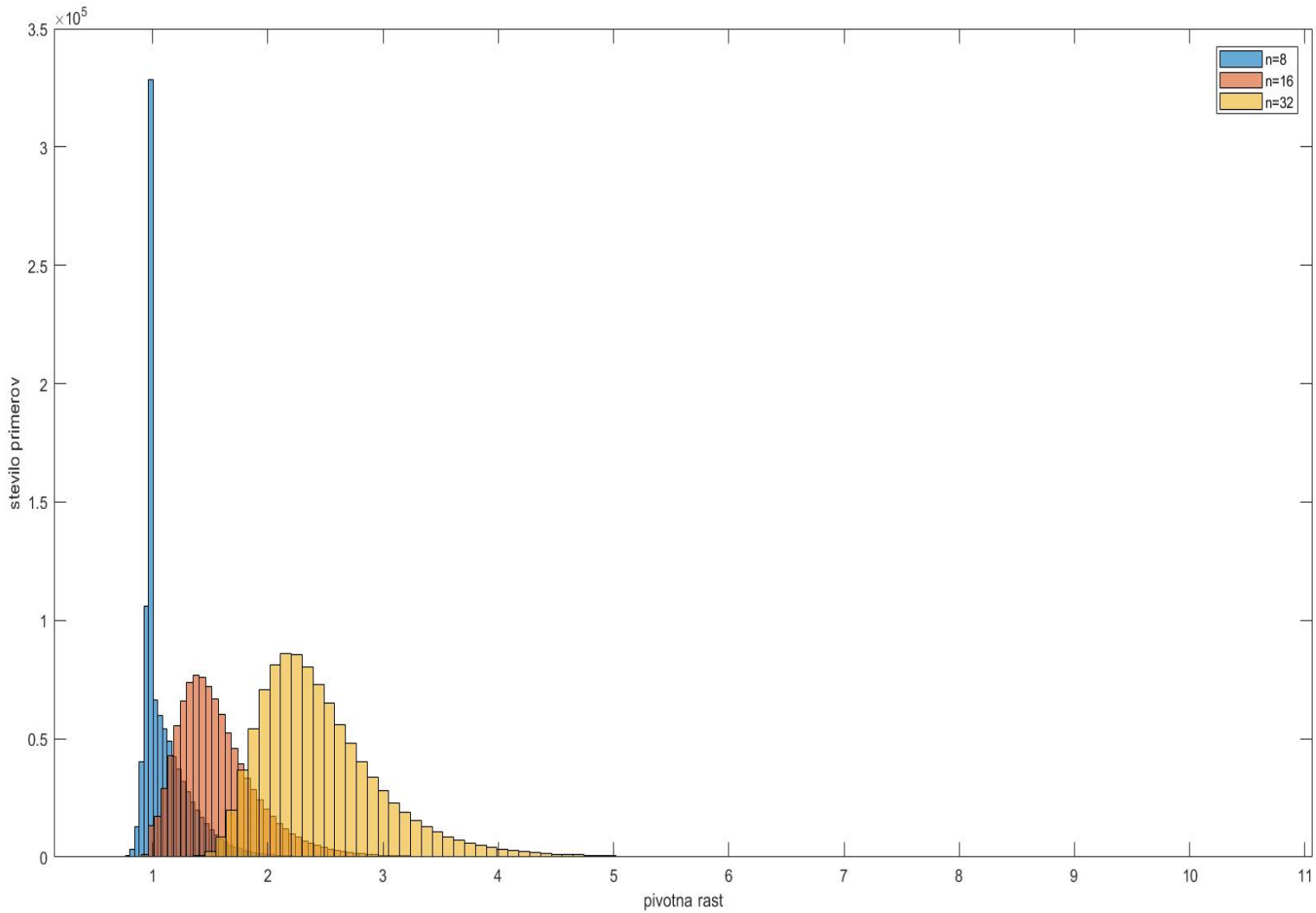
Matrika

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

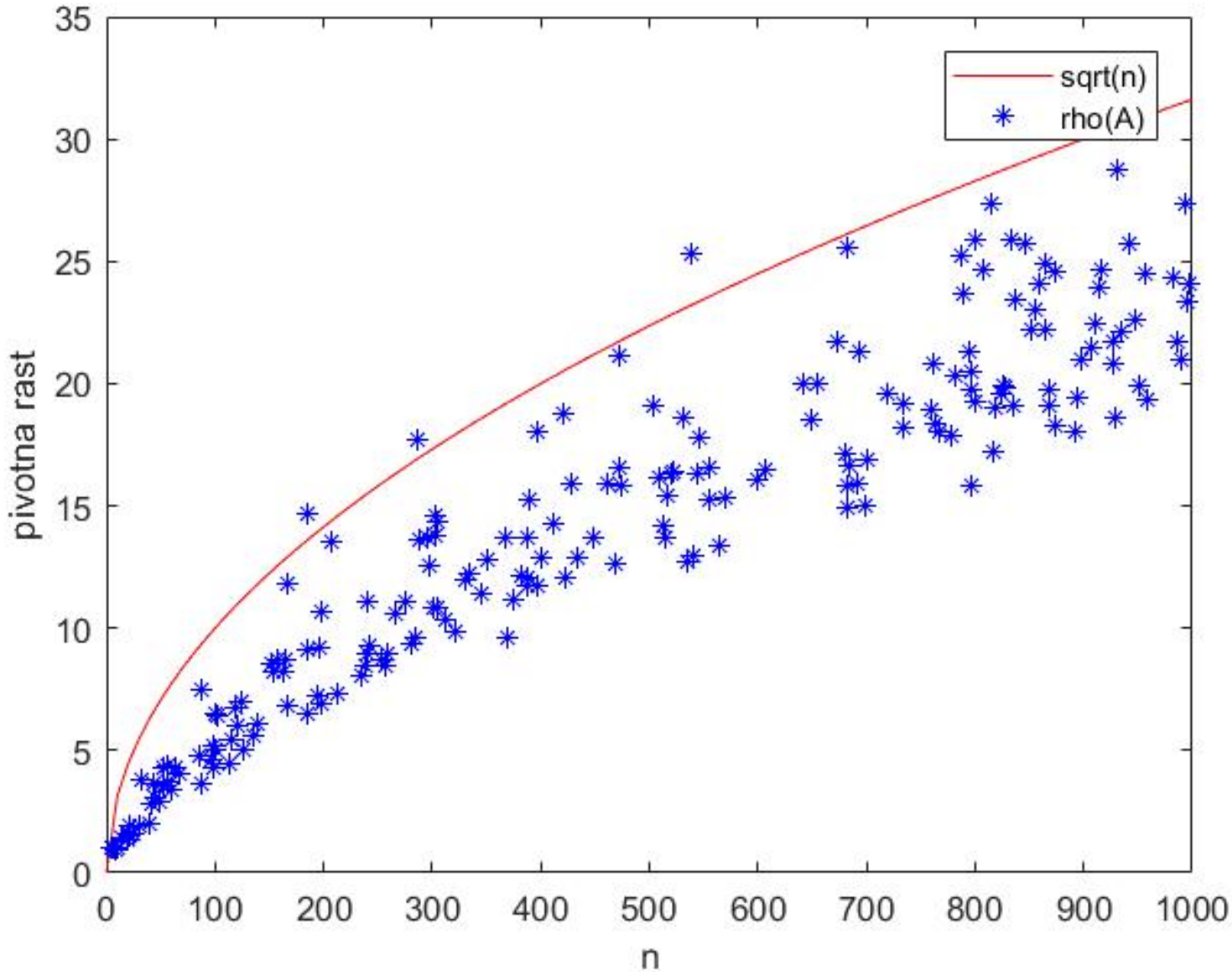
ima pivotno rast 2^{n-1}.

Statistično pa velja, da je pričakovana vrednost pivotne rasti $\mathcal{O}(n^{1/2})$, 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 $[0, 1]$):



LDL razcep simetrične matrike *A*

Trditev

Naj bo

$$A = A^T$$

$n \times n$ simetrična matrika in $A = LU$ njen LU razcep. Če je D diagonalna matrika, katere diagonala se ujema z diagonalo U-ja, potem je $U = DL^T$ in

$$A = LDL^T.$$

Dokaz. Velja

$$LU = A = A^T = (LU)^T = U^T L^T.$$

Z množenjem te verige enakosti z leve z L^{-1} in z desne z $(L^T)^{-1}$ dobimo

$$U(L^T)^{-1} = L^{-1}U^T =: D.$$

Ker je leva stran zgornja trikotna, desna pa spodnja trikotna, je D diagonalna matrika. Torej velja

$$A = LU = LDL^T.$$

Razcep Choleskega pozitivno definitne matrike A

Izrek (Razcep Choleskega)

Naj bo A simetrična in pozitivno definitna matrika, tj. za vsak $x \neq 0$ velja $x^T Ax > 0$. Potem obstaja spodnjetrikotna matrika V, da velja

$$A = VV^T.$$

Temu razcepu pravimo razcep Choleskega matrike A. Matrika V je enaka $V := LD^{1/2}$, kjer sta L in D matriki iz LDL razcepa matrike A.

Dokaz. Za obstoj je potrebno preveriti samo to, da $D^{1/2}$ res lahko izračunamo, tj. da so diagonalni elementi matrike D pozitivni. Da dobimo i-ti element na diagonali D, izračunamo $x^T Ax$ za vektor $x = (L^T)^{-1} e_i$, kjer je e_i i-ti stolpec identitete. Ker je A pozitivno definitna, je $x^T Ax > 0$.

Razcep Choleskega - algoritem

```
1   A = [a_ij]_{i,j} je dana n × n matrika
2   ce se razcep v celoti izvede, je rezultat
        spodnjetrikotna matrika V iz A = VV^T
3
4   for k = 1, ..., n
5       v_kk = sqrt(a_kk − Σ_{i=1}^{k−1} v_ki^2)
6       for j = k + 1, ..., n
7           for i = 1, ..., k − 1
8               a_jk = a_jk − v_ji v_ki
9           end
10          v_jk = a_jk / v_kk
11      end
12  end
```

Razcep Choleskega

Izrek (Cena razcepa Choleskega)

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

Dokaz: klik

Razcep Choleskega tako zahteva samo pol toliko operacij kot LU razcep in je najcenejši numerični način za ugotavljanje pozitivne definitnosti simetrične matrike.

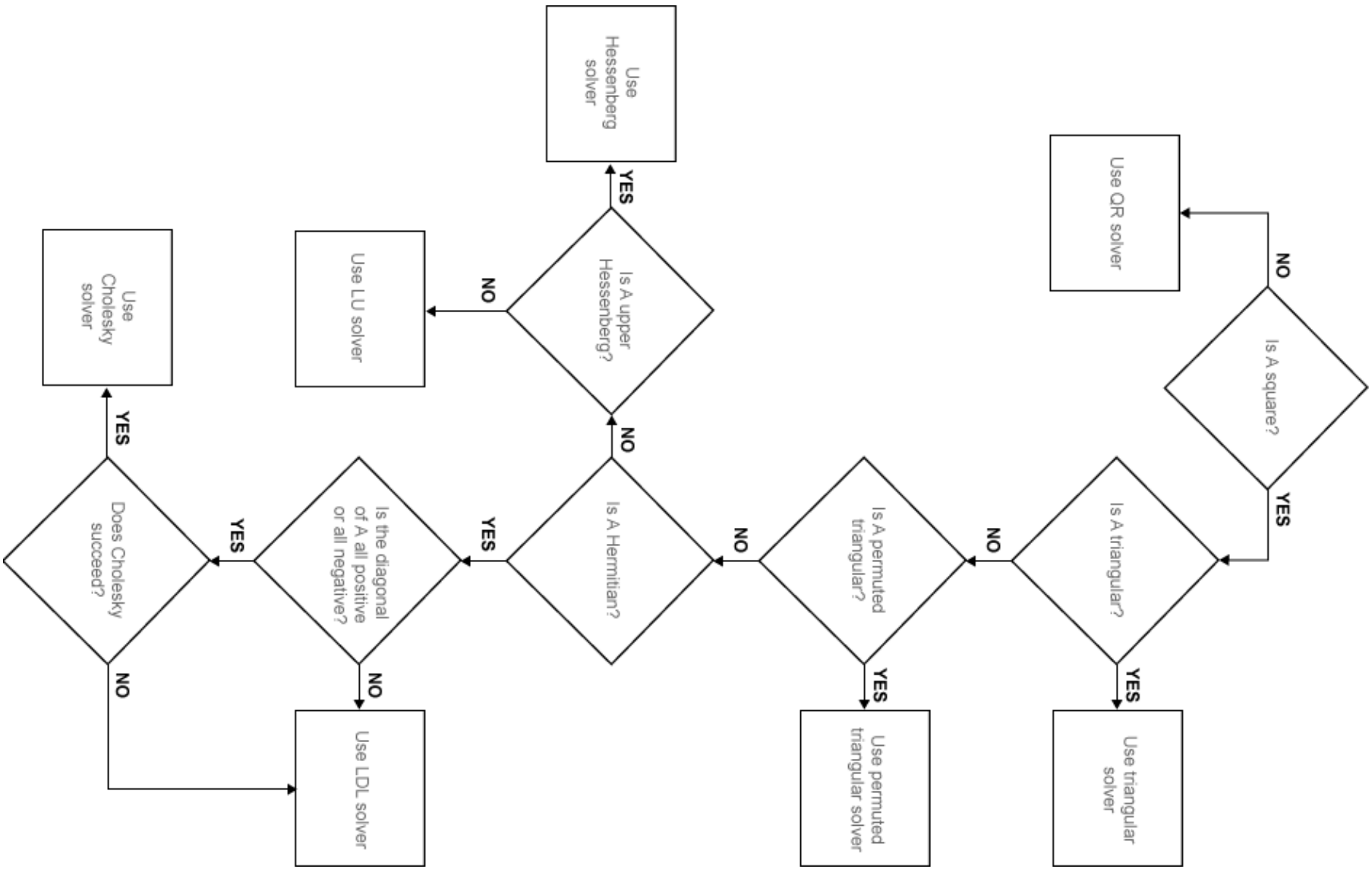
Reševanje sistema $Ax = b$ prek razcepa Choleskega:

1. Izračunamo $A = VV^T$. Cena: $\frac{1}{3}n^3 + \mathcal{O}(n^2)$.
2. Rešimo $Vy = b$ s premo subsitucijo. Cena: $n^2 + \mathcal{O}(n)$.
3. Rešimo $V^T x = y$ z obratno subsitucijo. Cena: $n^2 + \mathcal{O}(n)$.

Izrek (Stabilnost računanja razcepa Choleskega)

Računanje razcepa Choleskega je numerično stabilna metoda.

Backslash \ operator v Matlabu



Iterativne metode

$$Ax = b$$

- Jacobijeva iteracija
- Gauss-Seidlova iteracija
- SOR iteracija

- Pospešitev Čebiševa, SSOR, Metode podprostorov Krilova, Metoda konjugiranih gradientov, Hitra Fourierova transformacija, Ciklična redukcija, Večmrežna metoda.

Iterativne metode

Doslej smo iskali točne rešitve x^* sistema

$$Ax = b.$$

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

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

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

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.

Osnovna ideja

Sistem linearnih enačb $Ax = b$ preuredimo v obliko $x = Rx + c$, ki omogoča rekurzivno definicijo zaporedja približkov

$$x^{n+1} = Rx^n + c.$$

Če zaporedje konvergira

$$x_{m+1} = Rx_m + c,$$

je, limita zaporedja, rešitev sistema enačb.
Vprašanje je, kako izbrati R in c, da zaporedje x_{m+1} konvergira.

Osnovna ideja

Matriko A zapišemo kot

$$A = M - K,$$

kjer je M nesingularna matrika. Velja

$$Ax = Mx - Kx = b \quad \text{oz.} \quad x = \underbrace{M^{-1}K}_{R:=} x + \underbrace{M^{-1}b}_{c:=}.$$

Sedaj tvorimo zaporedje

$$x_{m+1} = Rx_m + c.$$

Glavno vprašanje je, kdaj zaporedje x_{m+1} konvergira (saj bo limita rešitev $Ax = b$).

Trditev

Naj bo $\|\cdot\|$ neka matrična norma, za katero je $\|R\| < 1$. Potem zaporedje $x_{m+1} = Rx_m + c$ konverigra proti rešitvi sistema $Ax = b$ za poljuben začetni približek x_0. *Dokaz: klik*

Spektralni radij matrike $R \in \mathbb{R}^{n\times n}$ je $\rho(R) = \max_j |\lambda_j(R)|$, kjer je $\lambda_j(R)$ j–ta lastna vrednost matrike R.

Trditev

1. *Za vsako matrično normo* $\|\cdot\|$ *velja* $\rho(R) \leqslant \|R\|$.
2. *Za vsak* $\epsilon > 0$ *obstaja matrična norma* $\|\cdot\|_*$ *za katero velja* $\|R\|_* \leqslant \rho(R) + \epsilon$.

Dokaz: klik

Posledica

Zaporedje $x_{m+1} = Rx_m + c$ *konvergira proti rešitvi sistema* $Ax = b$ *za poljuben začetni približek* x_0 *natanko tedaj, ko je* $\rho(R) < 1$.

Hitrost konvergence zaporedja $R \in \mathbb{R}^{n\times n}$ je

$$r(R) := -\log_{10} \rho(R).$$

Število $r(R)$ pove, najmanj za koliko se na vsakem koraku iteracije poveča število pravih desetiških števk.

Cilj pri iskanju razcepa $A = M - K$:

1. $Rx = M^{-1}Kx$ in $M^{-1}b$ je enostavno izračunati.
2. $\rho(R)$ je čim manjši.

Zgornja cilja sta nasprotujoča. Najenostavneje je za M vzeti I, vendar je potem $\rho(R)$ lahko velik. Obratno je lahko izbira $M = A$ in $K = 0$ dobra za drugi cilj, vendar slaba za prvega.

Odslej naprej pišimo:

$$A = D - \widetilde{L} - \widetilde{U},$$

kjer je D diagonalna matrike A, $-\widetilde{L}$ njen spodnje, $-\widetilde{U}$ pa njen zgornjetrikotni del.

Jacobijeva iteracija

Pri Jacobijevi iteraciji za M vzamemo D za K pa $\widetilde{L} + \widetilde{U}$. Tako sta matrika R in vektor c enaka

$$R_J := D^{-1}(\widetilde{L} + \widetilde{U}) =: L + U, \qquad c_J := D^{-1}b.$$

Če označimo j–ti približek rešitve z $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)})$, potem po komponentah to pomeni

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

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

Pri Gauss-Seidlovi iteraciji za M vzamemo $D - \widetilde{L}$ za K pa $\widetilde{U}$. Tako sta matrika R in vektor c enaka

$$R_{GS} := (D - \widetilde{L})^{-1}\widetilde{U}, \qquad c_{GS} := (D - \widetilde{L})^{-1}b.$$

Če označimo j–ti približek rešitve z $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)})$, potem po komponentah to pomeni

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

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 $x_i^{(m+1)}$ na desni strani enačbe izračunamo iz predpisa za GS–iteracijo:

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

Če to enačbo preuredimo in zapišemo v matrični obliki dobimo

$$R_{SOR(w)} = (D - w\widetilde{L})^{-1}\big((1-w)D + w\widetilde{U}\big),$$
$$c_{SOR(w)} = w(D - w\widetilde{L})^{-1}b.$$

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. 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 določimo optimalen w v $SOR(w)$.

Trditev (Spektralni radij R_{GS}, R_J)

Če je A kVDD, potem velja $\|R_{GS}\|_\infty \leqslant \|R_J\|_\infty < 1$. *Zato Jacobijeva in GS–iteracija konvergirata.* *Dokaz: klik*

- Ocena iz zgornje trditve ne zagotavlja, da bo GS–iteracija vedno hitrejša od Jacobijeve. V primeru najneugodnejše izbire začetnega približka, bo Jacobijeva iteracija gotovo počasnejše, sicer pa ne vemo.
- Ker je $\|A\|_\infty = \|A^T\|_1$, tudi za krepko stolpično diagonalno dominantne matrike obe iteraciji konvergirata.

Zgornjo trditev lahko pri dodatnih lastnosti matrike A še malo posplošimo.

Matrika je:

- vrstično diagonalno dominantna (VDD), če za vsak i velja $|a_{ii}| \geqslant \sum_{j=1, j\neq i}^{n} |a_{ij}|$, pri čemer vsaj za en i velja strogi neenačaj.
- nerazcepna, če ne obstaja permutacijska matrika P, tako da bi veljalo $A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$, kjer sta A_{11} in A_{22} kvadratni matriki.

Trditev (Spektralni radij R_{GS}, R_J)

Če je A VDD in nerazcepna, potem velja $\rho(R_{GS}) < \rho(R_J) < 1$. Zato Jacobijeva in GS–iteracija konvergirata.

Nerazcepnost matrike lahko opišemo z grafi:
Graf $G(A)$ matrike $A \in \mathbb{R}^{n\times n}$ je usmerjen graf z vozlišči $1, \ldots, n$, pri čemer povezava od i do j obstaja natanko tedaj, ko je $a_{ij} \neq 0$.

Usmerjen graf je krepko povezan, če obstaja usmerjena pot med poljubnima vozliščema $i \neq j$.

Trditev. A je nerazcepna natanko tedaj, ko je $G(A)$ krepko povezan.

Trditev (Spektralni radij $R_{SOR(w)}$)

Velja $\rho(R_{SOR(w)}) \geqslant |w-1|$. Torej je potreben pogoj za konvergenco $0 < w < 2$. *Dokaz: klik*

Izrek (Spektralni radij $R_{SOR(w)}$)

Če je A pozitivno definitna matrika, potem je $\rho(R_{SOR(w)}) < 1$ za vsak $0 < w < 2$. Za te w iteracija $SOR(w)$ konvergira.

Izrek (Primerjava spektralnih radijev)

Če je graf matrike A dvodelen, potem velja:

1. $\rho(R_{GS}) = \rho(R_J)^2$.
2. *Če ima R_J realne lastne vrednosti in je $\mu := \rho(R_J) < 1$, potem za $SOR(w)$ velja:*

$$w_{opt} = \frac{2}{1 + \sqrt{1 - \mu^2}},$$

$$\rho(R_{SOR(w_{opt})}) = w_{opt} - 1,$$

$$\rho(R_{SOR(w)}) = \begin{cases} w - 1, & w_{opt} \geqslant w < 2, \\ f(w, \mu), & 0 < w < w_{opt}, \end{cases}$$

kjer je $f(w, \mu) := 1 - w + \frac{1}{2}w^2\mu^2 + w\mu\sqrt{1 - w + \frac{1}{4}w^2\mu^2}$.

Metoda konjugiranih gradientov

Naj bo matrika sistema A pozitivno definitna.

$$f(x) = \frac{1}{2} x^T A x - b^T x$$

doseže minimum natanko tedaj, ko je x rešitev sistema:

$$\nabla f(x) = Ax - b = 0.$$

- Uporabimo gradientni spust.
- Smer spusta premaknemo, da je konjugirana prejšnji smeri.
- Vektorja sta konjugirana, če je $x^T A y = 0$ (pravokotna za skalarni produkt $\langle x, y \rangle_A = x^T A y$).

Metoda konjugiranih gradientov

Ideja algoritma

1. izberemo začetni približek x_0,
2. smer spusta je enaka $b - Ax_0$,
3. popravimo smer, da je konjugirana prejšnji
4. iskanje po premici (line search) da naslednji približek
5. ponovimo postopek

Algoritem

1. $r_0 = b - Ax$, $p_0 = r_0$
2. ponavljaj dokler ni $\|r_k\|^2$ majhna
   - 2.1 $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$ (iskanje po premici)
   - 2.2 $x_{k+1} = x_k - \alpha_k p_k$ (naslednji približek)
   - 2.3 $r_{k+1} = r_k - \alpha_k A p_k$ (ostanek - gradient)
   - 2.4 $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$; $p_{k+1} = r_{k+1} + \beta_k p_k$ (konjugirani gradient)

Metoda konjugiranih gradientov

Lastnosti in prednosti:

- Konvergira, če je A simetrična in pozitivno definitna.
- V neskončni natančnosti se zagotovo konča po n korakih.
- Željeno natančnost pogosto doseže prej.
- Potrebujemo le operacijo množenja.
- Konstantna prostorska zahtevnost (ni napolnitve razpršenih matrik).

Predoločeni sistemi

$$Ax = b$$

- *SVD* razcep
- Moore-Penroseov inverz
- Občutljivost sistema
- Normalni sistem
- *QR* razcep pred Gram-Schmidtove ortogonalizacije
- *QR* razcep prek Householderjevih zrcaljenj

Predoločeni sistemi

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:

$$Ax = b. \tag{4}$$

Kadar je $n > m$, sistemu (4) pravimo predoločen, točna rešitev pa najverjetneje ne obstaja. Zato nas navadno zanima rešitev, ki v izbrani vektorski normi $\|\cdot\|$ minimizira ostanek, tj. želimo, da je $\|Ax - b\|$ čim manjše. Če za $\|\cdot\|$ izberemo $\|\cdot\|_2$, potem se problemu reče linearni problem najmanjših kvadratov:

$$\text{Poišči } x \in \mathbb{R}^m, \text{ ki minimizira } \|Ax - b\|_2. \tag{5}$$

Primeri uporabe so:

- prilagajanje krivulj,
- statistično modeliranje podatkov,
- geodetsko modeliranje.

Primer (Kubična interpolacija)

Dane so točke $(x_1, y_1), \ldots, (x_n, y_n)$. *Iščemo pa kubični polinom*

$$p(x) = a_3x^3 + a_2x^2 + a_1x + a_0,$$

ki se podatkom najbolje prilega po metodi najmanjših kvadratov. Torej iščemo koeficiente $a_0, \ldots, a_3 \in \mathbb{R}$, *tako da je vsota*

$$\sum_{i=1}^{n} (a_3x_i^3 + a_2x_i^2 + a_1x_i + a_0 - y_i)^2$$

najmanjša možna.

To lahko napišemo kot sistem

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n, \end{pmatrix}$$

ki ga rešujemo po metodi najmanjših kvadratov.

Singularni razcep matrike

Izrek (Obstoj SVD)

Naj bo $A \in \mathbb{R}^{n\times m}$ matrika. Potem obstajajo

- *ortogonalni matriki $U \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{m\times m}$ ($UU^T = I_n$, $V^TV = I_m$),*
- *diagonalna matrika*

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min(n,m)}),$$

kjer je $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \sigma_{\min(n,m)} \geqslant 0$,

da velja

$$A = U\Sigma V^T.$$

Stolpcem V pravimo desni, stolpcem U pa levi singularni vektorji.

Geometrijsko singularni razcep ortonormirano bazo $v_1, \ldots, v_m$ preslika v večkratnike ortonormirane baze $u_1, \ldots, u_m$ in matrika A je 'diagonalna'.

Trditev (Spektralna norma matrike)

Velja $\|A\|_2 = \sigma_1$.

MP inverz

Moore-Penroseov posplošen inverz (MP inverz), matrike $A \in \mathbb{R}^{n\times m}$ je matrika $A^+ \in \mathbb{R}^{m\times n}$, ki zadošča naslednjim štirim pogojem:

1. $AA^+A = A$.
2. $A^+AA^+ = A^+$.
3. $(AA^+)^T = AA^+$.
4. $(A^+A)^T = A^+A$.

Trditev

Naj bo $A = U\Sigma V^T$ SVD razcep matrike A. Potem je MP inverz matrike A enak

$$A^+ = V\Sigma^+ U^T,$$

kjer je

$$\Sigma^+ = \operatorname{diag}(\sigma_1^+, \sigma_2^+, \ldots, \sigma_{\min(n,m)}^+)$$

in

$$\sigma_i^+ = \begin{cases} \sigma_i^{-1}, & \text{če je } \sigma_i \neq 0, \\ 0, & \text{sicer} \end{cases}.$$

Trditev

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

Občutljivost sistema $Ax = b$

Naj bo $A \in \mathbb{R}^{n \times m}$ pravokotna matrika. Pogojenostno število matrike A je

$$\kappa_2(A) := \|A\|_2 \|A^+\|_2.$$

Trditev

$\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$, *kjer sta* $\sigma_{\max}(A)$ *in* $\sigma_{\min}(A)$ *največja in najmanjša singularna vrednost* A.

Izrek (Občutljivost sistema $Ax = b$)

Naj ima matrika $A \in \mathbb{R}^{n \times m}$ *poln rang, tj.* $\operatorname{rank}(A) = m$. *Naj:*

- x_0 *minimizira* $\|b - Ax\|^2$,
- $r := b - Ax_0$ *in*
- x' *minimizira* $\|(b + \Delta b) - (A + \Delta A)x\|^2$,
- $\epsilon := \max\left(\frac{\|\Delta A\|_2}{\|A\|_2}, \frac{\|\Delta b\|_2}{\|b\|_2}\right)$.

Potem je

$$\frac{\|x' - x_0\|}{\|x_0\|} \geqslant \epsilon \underbrace{\left(\frac{2\kappa_2(A)}{\cos\theta} + \tan\theta \cdot \kappa_2^2(A)\right)}_{\kappa_{LS}(A)} + \mathcal{O}(\epsilon^2),$$

kjer je $\sin\theta = \frac{\|r\|_2}{\|b\|_2}$. *Število* κ_{LS} *je pogojenostno število za problem najmanjših kvadratov.*

Reševanje prek normalnega sistema

Trditev

Naj bo $\operatorname{rank}(A) = m$. *Rešitev sistema* (4) *po metodi najmanjših kvadratov je* $x \in \mathbb{R}^m$, *ki reši t.i. normalni sistem:*

$$A^T A x = A^T b. \tag{6}$$

Dokaz: klik

Težava, ki se pojavi pri reševanju (6), je numerična stabilnost. Računanje skalarnega produkta v splošnem ni relativno stabilna operacija. Vhodi matrike $A^T A$ pa so ravno skalarni produkti stolpcev A.

Primer - linearna regresija

Iščemo premico, ki se najbolje prilega podatkom

$$(x_1, y_1), \ldots, (x_n, y_n)$$

po metodi najmanjših kvadratov. Premica je oblike

$$y = a + bx.$$

Torej sta spremenljivki a in b. Sistem lahko zapišemo v obliki

$$\underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\vec{b}}. \tag{7}$$

Po zgoraj napisanem je rešitev (7) enaka

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

Reševanje normalnega sistema - razcep Choleskega

Če je matrika $A \in \mathbb{R}^{n \times m}$ ranga m, potem je matrika $A^T A$ pozitivno definitna in sistem $A^T Ax = A^T b$ lahko rešimo s pomočjo razcepa Choleskega:

1. Izračunaj $B = A^T A$ in $c = A^T b$.

   Cena: $nm^2 + \mathcal{O}(mn)$. (Zadosti je računati zgornjetrikotni del AA^T.)
2. Izračunaj razcep Choleskega $B = VV^T$. Cena: $\frac{1}{3}m^3 + \mathcal{O}(m^2)$.
3. Reši $Vy = A^T b$. Cena: $\mathcal{O}(m^2)$.
4. Reši $V^T x = y$. Cena: $\mathcal{O}(m^2)$.

Skupna cena: $nm^2 + \frac{1}{3}m^3 + \mathcal{O}(nm)$.

QR razcep

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

$$A = QR. \tag{8}$$

Iz (8) sledi, da sta stolpična prostora matrik A in Q enaka. 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 $R = [r_{ij}]_{i,j}$, potem veljajo zveze:

$$\begin{aligned} q_1 &= \frac{1}{r_{11}} a_1, \\ q_2 &= \frac{1}{r_{22}} (a_2 - r_{12} q_1), \\ &\vdots, \\ q_m &= \frac{1}{r_{mm}} (a_m - r_{1m} q_1 - \ldots - r_{m-1,m} q_{m-1}). \end{aligned} \tag{9}$$

Iz (9) lahko izpeljemo enega od načinov za izračun QR razcepa, tj. z uporabo Gram-Schmidtove ortogonalizacije (GSO) oz. modificirane Gram-Schmidtove ortogonalizacije (mGSO):

1 $A = [a_1, \ldots, a_m]$ je $n \times m$ matrika s stolpci $a_1, \ldots, a_m$
2 Rezultat sta matriki $q = [q_1, \ldots, q_m]$ in $R = [r_{ij}]$, da je
$A = QR$
3
4 $r_{11} = \|a_1\|_2$
5 $q_1 = \frac{1}{r_{11}} a_1$
6 for $j = 2, \ldots, m$
7 $q_j = a_j$
8 for $i = 1, \ldots, j$
9 $r_{ij} = q_i^T a_j$ $r_{ij} = q_i^T q_j$
10 $q_j = q_j - r_{ij} q_i$
11 end
12 $r_{jj} = \|q_j\|_2$
13 $q_j = \frac{1}{r_{jj}} q_j$
14 end

Uporaba QR razcepa za reševanje sistema (4)

Izrek (Računska zahtevnost CGS/MGS)

Število računskih operacij $(+, -, \cdot, :)$ *za izračun QR razcepa z GSO je*

$$\approx 2nm^2.$$

Dokaz: klik

Trditev (Reševanje sistema prek QR razcepa)

Naj bo $A \in \mathbb{R}^{n \times m}$ *in* $\operatorname{rank}(A) = m$. *Rešitev sistema* (4) *po metodi najmanjših kvadratov je enaka rešitvi zgornjetrikotnega sistema*

$$Rx = Q^T b, \tag{10}$$

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

Dokaz: klik

Householderjeva zrcaljenja (HZ)

GS postopek je lahko zelo nestabilen, saj izračunana matrika Q v primeru ne dovolj linearno neodvisnih stolpcev A ni ortogonalna. Zato želimo QR razcep narediti na drugačen način z uporabo ortogonalnih matrik.

Zrcaljenje preko hiperravnine, ki je pravokotna na vektor $w \in \mathbb{R}^m$, imenovano Householderjevo zrcaljenje (HZ), je predstavljeno z matriko

$$P_w = I_m - \frac{2}{\|w\|_2^2} ww^T.$$

Trditev (Lastnosti HZ)

- *P_w je ortogonalna: $P_w^T = P_w$ in $P_w^2 = I$.*
- *Za $x = \alpha w + u$, kjer je $u \perp w$, velja $P_w x = -\alpha w + u$.*

Matrike P_w ne izračunamo, ampak zadošča hraniti le vektor w, saj je

$$P_w x = x - \frac{2}{\|w\|_2^2} (w^T x) w.$$

Cena izračuna je P_w je $4m + \mathcal{O}(1)$ ($\|w\|_2$ smo že predhodno izračunali).

Trditev

Naj bosta $x, y \in \mathbb{R}^m$ *z* $\|x\|_2 = \|y\|_2$. *Potem za* $w = x - y$ *velja* $P_w x = y$.

Po zgornji trditvi lahko vektor x preslikamo v $\|x\|_2 e_1$ in $-\|x\|_2 e_1$ s HZ.

Lema

Velja $\|w_\pm\|_2 = 2\|x\|_2^2 (\|x\|_2 \mp x_1)$.

Za numerično stabilnost je bolje vzeti predznak, za katerega je $\|x\|_2 \mp x_1$ večje. Če je $x_1 < 0$ vzamemo $-$, sicer pa $+$.

Izračun QR razcepa prek HZ

Matriko A preoblikujemo v zgornjetrikotno R z množenjem z leve z $m-1$ HZ-ji:

1. S H_1 preslikamo prvi stolpec A v večkratnik e_1.
2. Drugi stolpec $H_1 A$ od diagonale navzdol v večkratnik e_2 z $\widetilde{H}_2 = 1 \oplus H_2$.
3. Tretji stolpec $\widetilde{H}_2 H_1 A$ od diagonale navzdol v večkratnik e_3 z $\widetilde{H}_3 = I_2 \oplus H_2$.
4. Nadaljujemo ta postopek.

Cena izračuna matrike R je $\approx 2nm^2 - \frac{2}{3}n^3$. Cena izračuna Q je še dodatno $4mn^2 - 2nm^2$.

Uporabljene vektorje zrcaljenja lahko hranimo v spodnjem trikotniku matrike A.

Izračun QR razcepa prek HZ - grafično

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \underbrace{\rightarrow}_{H_1} A_1 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix}$$

$$\underbrace{\rightarrow}_{\begin{pmatrix} 1 & 0 \\ 0 & H_2 \end{pmatrix}} A_2 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix} \underbrace{\rightarrow}_{\begin{pmatrix} I_2 & 0 \\ 0 & H_3 \end{pmatrix}} A_3 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

$$\underbrace{\rightarrow}_{\begin{pmatrix} 1_3 & 0 \\ 0 & H_4 \end{pmatrix}} A_4 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{pmatrix}.$$

Problem lastnih vrednosti matrik

$$Ax = \lambda x$$

- Potenčna metoda
- Ortogonalna iteracija
- *QR* iteracija
- *QR* iteracija s premiki

Lastne vrednosti in lastni vektorji

Za kvadratno matriko $A \in \mathbb{R}^{n\times n}$ je število $\lambda \in \mathbb{C}$ lastna vrednost, če obstaja tak neničelni vektor $v \in \mathbb{R}^n$, ki zadošča

$$Av = \lambda v. \tag{11}$$

Takemu vektorju rečemo lastni vektor, pripadajoč λ.

Matriki $A, B \in \mathbb{R}^{n\times n}$ sta podobni, če obstaja obrnljiva matrika $S \in \mathbb{R}^{n\times n}$, da velja

$$B = SAS^{-1}.$$

Trditev

Lastne vrednosti matrike A so vse različne ničle karakterističnega polinoma

$$p_A(\lambda) := \det(A - \lambda I).$$

Dokaz: klik

Ločimo med geometrijsko $g_A(\lambda)$ in algebraično vektratnostjo $a_A(\lambda)$ lastne vrednosti λ. Prva je enaka $\dim \ker(A - \lambda I)$, druga pa stopnji λ kot ničli polinoma $p_A(\lambda)$. Velja $1 = g_A(\lambda) \leqslant a_A(\lambda)$.

Podobnost matrik

Primer

Matrika $A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ *ima lastno vrednost 2, za katero velja* $g_A(2) = 1 < a_A(2) = 2$.

Definicija

Matriki $A, B \in \mathbb{R}^{n \times n}$ *sta podobni, če obstaja obrnljiva matrika* $S \in \mathbb{R}^{n \times n}$, *da velja*

$$B = SAS^{-1}.$$

Trditev

Če sta matriki $A, B \in \mathbb{R}^{n \times n}$ *podobni, potem imate iste lastne vrednosti. Če je* S *podobnosta matrika in* v *lastni vektor* A, *ki pripada lastni vrednosti* λ, *potem je* Sv *lastni vektor matrike* B. *Dokaz: klik*

Trditev

Če ima matrika $A \in \mathbb{R}^{n \times n}$ *same različne lastne vrednosti* $\lambda_1, \ldots, \lambda_n$, *potem pripadajoči lastni vektorji* $x_1, \ldots, x_n$ *tvorijo bazo prostora* $\mathbb{R}^n$.

Diagonalizacija

Če poljuben vektor x razvijemo po bazi lastnih vektorjev x_i, $i = 1, \ldots, n$,

$$x = \sum_{i=1}^{n} \beta_i x_i,$$

potem je

$$Ax = A\Big(\sum_{i=1}^{n} \beta_i x_i\Big) = \sum_{i=1}^{n} \beta_i A x_i = \sum_{i=1}^{n} \beta_i \lambda_i x_i.$$

V tem primeru se torej matrika A obnaša kot diagonalna matrika

$$D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix}$$

in velja

$$A = SDS^{-1},$$

kjer je $S = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times n}$.

Jordanska in Schurova forma

Nima vsaka matrika A n različnih lastnih vektorjev. Najbolj splošna oblika matrike je Jordanova forma $J(A)$, tj. matrika A je podobna matriki direktni vsoti Jordanskih kletk

$$J(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & 0 \\ \vdots & & \ddots & \lambda & 1 \\ 0 & 0 & \cdots & 0 & \lambda \end{pmatrix}.$$

Toda izračun $J(A)$ ni numerično stabilen, zato se za velike matrike v praksi ne uporablja, pač pa se uporablja Schurova forma.

Trditev (Schurova forma)

Naj bo $A \in \mathbb{R}^{n\times n}$ matrika. Potem obstajata:

1. *Unitarna matrika $Q \in \mathbb{C}^{n\times n}$ $(QQ^* = I_n)$, da je Q^*AQ zgornje trikotna matrika.*
2. *Ortogonalna matrika $U \in \mathbb{R}^{n\times n}$ $(UU^T = I_n)$, da je U^TAU bločno zgornje trikotna matrika z diagonalnimi bloki velikosti 2×2 oz. 1×1.*

Dokaz: klik

Iskanje lastnih vrednosti

V nadaljevanju si bomo pogledali več metod za iskanje lastnih vrednosti:

1. Potenčna metoda: Namenjena iskanju lastnega vektorja, ki pripada po absolutni vrednosti največji lastni vrednosti.

2. Inverzna iteracija: Namenjena iskanju lastnega vektorja, ki pripada lastni vrednosti, ki je najbližje številu $\sigma \in \mathbb{R}$.
3. Ortogonalna iteracija: Namenjena hkratnemu iskanju celotnega lastnega podprostora neke lastne vrednosti λ.

4. *QR* iteracija: Verzija ortogonalne iteracije, ki deluje tudi za premike $A - \sigma I$.

5. *QR* iteracija s pohitritvami: Da pohitrimo *QR* iteracijo sprva matriko preoblikujemo v podobno matriko usrezne oblike, za katero bo konvergenca *QR* iteracije bistveno hitrejša.

Iskanje dominantne lastne vrednosti

Definicija

Lastna vrednost λ_1 je dominantna, če za vse ostale lastne vrednosti $\lambda_2, \ldots, \lambda_n$ velja

$$|\lambda_1| > |\lambda_i|, \quad i = 2, \ldots, n.$$

V mnogih problemih nas zanima samo dominantna lastna vrednost, npr. v Googlovem PageRank algoritmu, kar bomo kmalu videli.

Dominantno lastno vrednost pa lahko enostavno poiščemo z uporabo potenčne metode.

Potenčna metoda

Vhod: matrika $A \in \mathbb{R}^n$, zacetni priblizek $x_0 \in \mathbb{R}^n$
Izhod: dominantni lastni par $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}$

$i = 0; \quad \widetilde{\lambda}_0 = x_0^T A x_0$
while $\|Ax_i - \widetilde{\lambda}_i x_i\| > tol$
    $y_{i+1} = Ax_i$
    $x_{i+1} = \frac{y_{i+1}}{\|y_{i+1}\|_2}$
    $\widetilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$
    $i = i + 1$
end
$x = x_i; \quad \lambda = \widetilde{\lambda}_i$

Trditev (Konvergenca potenčne metode)

Naj ima matrika $A \in \mathbb{R}^{n \times n}$ n lastnih vrednosti, pri čemer je ena dominantna. Zaporedje vektorjev $(x_i, \widetilde{\lambda}_i)$ iz potenčne metode konvergira proti lastnemu paru, ki pripada dominantni lastni vrednosti.

Če je λ_1 dominantna lastna vrednost, λ_2 pa po absolutni vrednosti druga največja lastna vrednost, potem je hitrost konvergence odvisna od razmerja $\frac{|\lambda_2|}{|\lambda_1|}$. *Dokaz: klik*

Uporaba potenčne metode - PageRank algoritem

Spletni iskalnik Google sta leta 1998 zagnala Sergey Brin in Larry Page. Iskalnik spletne strani oštevilči glede na njihovo pomembnost. Ko vpiše v iskalno polje neko besedno zvezo, potem iskalnik poišče vse spletne strani, kjer se pojavi ta besedna zveza, rezultate pa razvrsti glede na ta vrstni red pomembnosti, pri čemer višje izpiše tiste spletne strani, ki imajo nižjo številko.

Algoritem za številčenje spletnih strani se imenuje PageRank algoritem. Izkaže se, da algoritem temelji na iskanju lastnega vektorja dominantne lastne vrednosti Googlove matrike sosednosti.

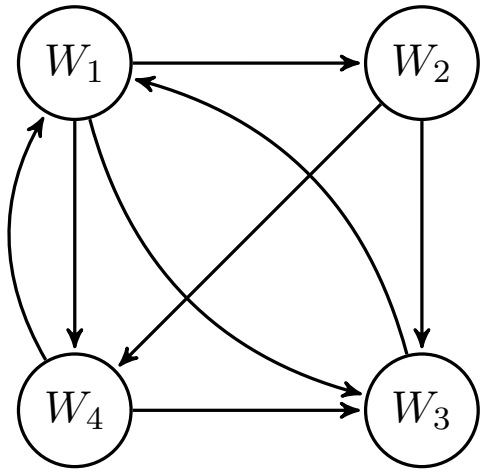
Splet pa lahko predstavimo kot velik usmerjen graf.

Vsaki spletni strani priredimo vozlišče grafa. Dve vozlišči u, v povežemo z usmerjeno povezavo, če iz spletne strani u vodi povezava na stran v.

Število usmerjenih povezav iz strani v_i označimo z O_i.

Matrika sosednosti H grafa ima stolpce/vrstice označene z vozlišči v_i grafa. Na mestu ij pa stoji $\frac{1}{O_j}$. Torej so v j-tem stolpcu neničelni vhodi v vrsticah strani v_i, v katere vodi usmerjena povezava iz v_j.

Primer



Matrika sosednosti grafa na sliki je enaka

$$H = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{pmatrix}.$$

Teža oz. pomembnost x_k spletne strani v_k pa je enaka

$$x_k = \sum_{i \in I_k} \frac{x_i}{O_i}, \tag{12}$$

pri čemer je I_k množica vseh spletnih strani, ki vodijo na stran v_k.

Enačbe (12) zapišemo s pomočjo matrike sosednosti kot

$$x = Hx,$$

kjer je x vektor pomembnosti. Torej moramo najti lastni vektor matrike H, ki pripada lastni vrednosti 1.

Vprašanje: Ali tak vektor sploh obstaja? Tj. ali je 1 gotovo lastna vrednost matrike H?

Ne nujno. Če malo popravimo matriko H s smiselnim oponašanjem naključnega uporabnika spleta, pa je odgovor da, kot bomo kmalu videli.

Nekaj definicij

Matrika $A = [a_{ij}]_{i,j} \in \mathbb{R}^{n\times n}$ je:

1. pozitivna, če ja $a_{ij} \geqslant 0$ za vsaka $i, j = 1, \ldots, n$.

2. razcepna, če obstaja neka permutacija stolpcev/vrstic, da je permutirana matrika oblike

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

   kjer sta A_{11}, A_{22} kvadratni matriki, 0 je ničelni blok in A_{12} je pravokotni blok. Če matrika ni razcepna, ji pravimo nerazcepna.

3. vrstično stohastična, če je $\sum_{j=1}^{n} a_{ij} = 1$ za vsak $i = 1, \ldots, n$. Z drugimi besedami, vsota vhodov v vsaki vrstici je enaka 1.

4. stolpično stohastična, če je vsota vhodov v vsakem stolpcu enaka 1.

Izrek (Perron-Frobeniusov izrek, 1912)

Naj bo A pozitivna nerazcepna matrika, ki je vrstično ali stolpično stohastična. Potem je 1 dominantna lastna vrednost matrike A, obstaja pa pripadajoč lastni vektor s samimi nenegativnimi elementi.

Uporaba za PageRank algoritem: Matrika sosednosti H svetovnega spleta je pozitivna, ni pa nujno nerazcepna in stolpično stohastična. Graf namreč lahko razpade na več komponent za povezanost, ki dajo razcepnost matrike. Poleg tega lahko obstajajo 'slepe' spletne strani, iz katerih ni nobene izhodne povezave.

Rešitev problema razcepnosti in stohastičnosti matrike H: Matriko H malo popravimo:

$$\widetilde{H} = \alpha H + (1-\alpha)\frac{1}{N}S,$$

kjer je $\alpha \in (0,1)$ utež, N je število spletnih strani, S pa matrika samih 1. Matrika $\frac{1}{N}S$ oponaša korak uporabnika spleta, ko na nekem koraku iz trenutne strani skočimo naključno z verjetnostjo $\frac{1}{N}$ na poljubno drugo stran.

Matrika $\widetilde{H}$ je očitno nerazcepna in stolpično stohastična. Po Perron-Frobeniusovem izreku obstaja x s samimi nenegativnimi vhodi, da velja

$$\widetilde{H}x = x.$$

Glede na vhode v x Google rangira spletne strani po pomembnosti. Vektor x lahko poiščemo s potenčno metodo.

Hitrost konvergence potenčne metode v odvisnosti od α. Glede na α je konvergenca metode različno hitra.

Za $\alpha = 0.5$ je konvergenca zagotovljena že po nekaj korakih potenčne metode, vendar ima naključni skok v tem primeru prevelik vpliv in matrika $\widetilde{H}$ ne oponaša dobro uporabnika spleta, ki naključno brska po spletu, dokler se ne naveliča in zamenja tok brskanja.

Za $\alpha = 0.99$ je konvergenca prepočasna za praktično uporabo.

Navadno se navaja, da Google uporablja $\alpha \approx 0.8$, ki je nek kompromis med hitrostjo in smiselnostjo.

Več informacij o vplivu α je opisano v knjigi:

Amy Langville, Carl Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, 2006.

Inverzna iteracija

Če poznamo približno vrednost σ neke lastne vrednosti, lahko naredimo to lastno vrednost dominantno tako, da od A odštejemo σI in invertiramo matriko, tj. $B := (A - \sigma I)^{-1}$. Če so λ_i lastne vrednosti A, potem so $(\lambda_i - \sigma)^{-1}$ lastne vrednosti B. Pripadajoči lastni vektorji pa so enaki.

Tako lahko uporabimo potenčno metodo za B. Metodo imenujemo inverzna iteracija.

1 Vhod: matrika $A \in \mathbb{R}^{n \times n}$, zacetni priblizek $x_0 \in \mathbb{R}^n$,
    premik $\sigma \in \mathbb{R}$
2 Izhod: Lastni par $(x, \lambda) \in \mathbb{R}^n \times \mathbb{R}$, λ blizu σ
3
4 $i = 0$; $\widetilde{\lambda}_0 = x_0^T A x_0$
5 while $\|Ax_i - \widetilde{\lambda}_i x_i\| > tol$
6   Resi $(A - \sigma I) y_{i+1} = x_i$ na y_{i+1}
7     $x_{i+1} = \frac{y_{i+1}}{\|y_{i+1}\|_2}$
8     $\widetilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$
9     $i = i + 1$
10 end
11 $x = x_i$; $\lambda = \widetilde{\lambda}_i$

Ortogonalna iteracija

Posplošitev potenčne metode do metode, ki istočasno išče p–dimenzionalni podprostor lastnih vektorjev se imenuje ortogonalna iteracija.

```
1  Vhod: matrika A ∈ R^{n×n}, ortogonalna matrika Z_0 ∈ R^{n×p}
2  Izhod: prvih p absolutno max l.vr. λ_1,...,λ_p od A
3
4  i = 0;   A_0 = Z_0^T A Z_0;   L_0...spodnje trikotni del A_0
5
6  while ||L_i||_F / ||A_i||_F > tol
7      Y_{i+1} = A Z_i
8      Izracunaj QR razcep matrike Y_{i+1}
9      Y_{i+1} = Z_{i+1} R_{i+1}  (Z_{i+1} je ortogonalna matrika)
10     A_{i+1} = Z_{i+1}^T A Z_{i+1};   L_{i+1}...spodnje trikotni del A_{i+1}
11     i = i + 1
12 end
13
14 diagonala matrike A_i so λ_1,...,λ_p
```

Trditev

Naj bodo lastne vrednosti A enake

$$|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_p| > |\lambda_{p+1}| \geqslant \ldots \geqslant |\lambda_n|$$

in $v_1, \ldots, v_n$ pripadajoči lastni vektorji. Potem je $\mathrm{Lin}(Z_i) = \mathrm{Lin}\{v_1, \ldots, v_p\}$.

Dokaz: klik

Trditev

Naj ima A same realne lastne vrednosti in naj velja

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|.$$

1. *Če za Z_0 vzamemo I_n, potem zaporedje $Z_i^T A Z_i$ konvergira proti Schurovi formi matrike A, kjer so na diagonali lastne vrednosti v padajočem vrstnem redu.*
2. *Hitrost konvergence j–tega diagonalnega vhoda matrike A je odvisna od*

$$\min\left(\frac{|\lambda_j|}{|\lambda_{j-1}|}, \frac{|\lambda_{j+1}|}{|\lambda_j|}\right).$$

Dokaz: klik

QR iteracija

V ortogonalno iteracijo bi radi vključili še premike, da lahko pospešimo konvergenco. To lahko naredimo s *QR* iteracijo.

Vhod: matrika $A \in \mathbb{R}^{n \times n}$
Izhod: l.vr. $\lambda_1, \ldots, \lambda_n$ od A

$i = 0$; $A_0 = A$; L_0...spodnje trikotni del A_0

while $\frac{\|L_i\|_F}{\|A_i\|_F} > tol$
    Izracunaj QR razcep matrike A_i:
    $A_i = Q_i R_i$
    Pomnozi Q_i in R_i v obratnem vrstnem redu:
    $A_{i+1} = R_i Q_i$; L_{i+1}..spodnje trikotni del A_0
    $i = i + 1$
end

Vse matrike A_i v QR iteraciji imajo iste lastne vrednosti in konvergirajo proti Schurovi formi matrike A.

Trditev

1. *A_{i+1} je ortogonalno podobna A_i*
2. *A_i izračunan s QR iteracijo je enak $Z_i^T A Z_i$, kjer je Z_i matrika dobljena z ortogonalno iteracijo z $Z_0 = I_n$. Torej A_i konvergirajo proti Schurovi formi za A.*

Dokaz: klik

Primer

Na spodnjih povezavah sta MATLAB kodi za simulacijo hitrosti konvergence QR iteracije brez premikov.

https://people.eecs.berkeley.edu/~demmel/ma221_Fall04/Matlab/qriter.m

https://people.eecs.berkeley.edu/~demmel/ma221_Fall04/Matlab/qrplt.m

QR iteracija z enojnim premikom

QR iteracijo lahko pospešimo s pomočjo premikov.

```
1  Vhod: matrika A ∈ R^{n×n}
2  Izhod: Schurova forma matrike A
3
4  i = 0;   A_0 = A
5
6  for i = 0, 1, ...
7      Izberi premik σ_i ∈ R blizu lastne vrednosti A
8      Izracunaj QR razcep matrike A_i − σ_i I_n:
9      A_i − σ_i I = Q_i R_i
10     Pomnozi Q_i in R_i v obratnem vrstnem redu in
           pristej premik:
11     A_{i+1} = R_i Q_i + σ_i I_n
12     i = i + 1
13 end
```

Vse matrike A_i v QR iteraciji imajo iste lastne vrednosti.

Trditev

A_{i+1} je ortogonalno podobna A_i. *Dokaz: klik*

QR iteracija z enojnim premikom

Zaustavitveni kriterij: Kdaj zadnji diagonalni element matrike A_i proglasimo za lastno vrednost? Smiselna zahteva je računati normo vrstice $A_i(n, 1:n-1)$. Ko je ta manjša od

$$\left(|A_i(n,n)| + A_i(n-1,n-1)|\right) \cdot tol,$$

kjer je *tol* izbrana toleranca, končamo. Nato nadaljujemo iteracijo na podmatriki $A_i(1:n-1, 1:n-1)$.

Hitrost konvergence diagonalnih elementov matrike A proti lastnim vrednostim pri premiku σ_i je sorazmerna z

$$\frac{|\lambda_k - \sigma_i|}{\min_{j \neq k} |\lambda_j - \sigma_i|},$$

kjer je $|\lambda_k - \sigma_i| = \min_j |\lambda_j - \sigma_i|$.

Cena *QR* iteracije:

1. Cena izračuna *QR* razcepa na vsakem koraku *QR* iteracije je $\mathcal{O}(n^3)$.
2. Tudi če na vsakem koraku pridobimo eno lastno vrednost, je skupna cena $\mathcal{O}(n^4)$.

Pocenitev QR iteracije z enojnim premikom

Vprašanje. Ali se da zmanjšati število računskih operacij za izvedbo QR iteracije?

Matriko bi pred vstopom v QR iteracijo radi preoblikovali v enostavnejšo obliko, za katero bo veljalo:

1. QR razcep je poceni izračunati.
2. Oblika matrike se bo ohranjala med QR iteracijo.

Matrika H je zgornja Hessenbergova, če ima pod diagonalo neničelno kvečjemu prvo poddiagonalo.

Primer

$$H = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}.$$

Preoblikavanje matrike v zgornjo Hessenbergovo matriko

Matriko A lahko preoblikujemo v ortogonalno podobno zgornjo Hessenbergovo matriko H za naslednjim postopkom:

1. Poiščemo Householderjevo zrcaljenje H_1, ki stolpec $A(2:n,1)$ (izpustimo prvo vrstico stolpca) prezrcali v večkratnik stolpca $[1,0,\ldots,0]^T \in \mathbb{R}^{n-1}$. Potem ima matrika

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix} A \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix}^T$$

   v prvem stolpcu pravo obliko.

2. Poiščemo Householderjevo zrcaljenje H_2, ki stolpec $A_1(3:n,2)$ (izpustimo prvi dve vrstici stolpca) prezrcali v večkratnik stolpca $[1,0,\ldots,0]^T \in \mathbb{R}^{n-2}$. Potem ima matrika

$$A_2 = \begin{pmatrix} I_2 & 0 \\ 0 & H_2 \end{pmatrix} A_1 \begin{pmatrix} I_2 & 0 \\ 0 & H_2 \end{pmatrix}^T$$

   v prvih dveh stolpcih pravo obliko.

3. Nadaljujemo ta postopek do predzadnjega stolpca $n-2$.

Primer

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \rightarrow A_1 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix}$$

$$\rightarrow A_2 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix} \rightarrow A_3 = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}.$$

Cena prevedbe v zgornjo Hessenbergovo obliko je $\frac{10}{3}n^3 + \mathcal{O}(n^2)$ operacij.

Trditev

1. *Vse matrike $A, A_1, A_2, \ldots, A_{n-2}$ v zgornjem postopku imajo iste lastne vrednosti.*

2. *Med QR iteracijo s premikom se zgornja Hessenbergova oblika ohranja, tj. če ima A_i to obliko, potem ima tudi A_{i+1} to obliko.*

Givensove rotacije

Vprašanje: Kako poceni izračunati QR razcep zgornje Hessenbergove matrike H?

V ta namen uporabimo Givensove rotacije. Cilj je poiskati rotacijo R v $\mathbb{R}^2$, ki vektor

$$v := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

zavrti v vektor z drugo komponento 0. Ker se velikost ohranja, mora biti to

$$u := \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{pmatrix}.$$

Trditev

Matrika $R \in \mathbb{R}^{2\times 2}$, ki predstavlja rotacijo v $\mathbb{R}^2$ in zavrti vektor v v u, je

$$R = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix},$$

kjer je

$$\cos\varphi = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \quad \sin\varphi = \frac{-x_2}{\sqrt{x_1^2 + x_2^2}}.$$

Dokaz: klik

QR razcep Hessenbergove matrike

QR razcep zgornje Hessenbergove matrike A izračunamo z naslednjim postopkom:

1. Poiščemo Givensovo rotacijo $\widetilde{R}_{21}$, ki vektor $A(1:2,1)$ zavrti v večkratnik stolpca $[1,0]^T \in \mathbb{R}^2$. Potem ima matrika

$$A_1 = \begin{pmatrix} \widetilde{R}_{21} & 0 \\ 0 & I_{n-2} \end{pmatrix} A := R_{21} A$$

   v prvem stolpcu pod diagonalo same 0.

2. Poiščemo Givensovo rotacijo $\widetilde{R}_{32}$, ki vektor $A_1(2:3,2)$ zavrti v večkratnik stolpca $[1,0]^T \in \mathbb{R}^2$. Potem ima matrika

$$A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \widetilde{R}_{23} & 0 \\ 0 & 0 & I_{n-3} \end{pmatrix} A := R_{32} A$$

   v prvih dveh stolpcih pod diagonalo same 0.

3. Nadaljujemo ta postopek do predzadnjega stolpca $n-1$ in dobimo zgornjetrikotno matriko R. Velja še

$$Q = R_{21}^T \cdots R_{n-2,n-1}^T R_{n-1,n}^T.$$

Primer

$$A = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \underbrace{\to}_{R_{21}} A_1 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

$$\underbrace{\to}_{R_{23}} A_2 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix} \underbrace{\to}_{R_{43}} A_3 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

$$\underbrace{\to}_{R_{54}} A_4 = \begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{pmatrix}.$$

Cena izračuna QR razcepa zgornje Hessenbergove matrike je $\mathcal{O}(n^2)$, saj ima množenje z R_{ij} ceno $\mathcal{O}(n)$. QR iteracija se tako res poceni iz $\mathcal{O}(n^4)$ na $\mathcal{O}(n^3)$.

QR iteracija z dvojnim premikom

V primeru, da imamo par kompleksnih konjugiranih lastnih vrednosti, je smiselno na vsakem koraku QR iteracije narediti dvojni premik:

1. Izberi premik $\sigma \in \mathbb{C}$ blizu lastne vrednosti A
2. Izracunaj QR razcep matrike $A - \sigma I_n$:
3. $A - \sigma I = Q_1 R_1$
4. Pomnozi Q_1 in R_1 v obratnem vrstnem redu in pristej premik:
5. $A' = R_1 Q_1 + \sigma I_n$
6. Izracunaj QR razcep matrike $A' - \overline{\sigma} I_n$:
7. $A' - \overline{\sigma} I = Q_2 R_2$
8. Pomnozi Q_2 in R_2 v obratnem vrstnem redu in pristej premik:
9. $A'' = R_2 Q_2 + \overline{\sigma} I_n$

Primerno izbiro za σ dobimo iz lastnih vrednosti spodnjega desnega 2×2 vogala matrike A.

Radi pa bi se izognili kompleksni aritmetiki. To pa se da doseči, kot je prikazano na naslednji strani.

Da se izognemu kompleksni aritmetiki, lahko oba koraka združimo in izračunamo QR razcep realne matrike

$$N = (A - \overline{\sigma}I)(A - \sigma I) = A^2 - (\sigma + \overline{\sigma})A + \sigma\overline{\sigma}I_n.$$

Velja namreč

$$N = Q_1 Q_2 R_2 R_1,$$

kjer so Q_1, Q_2, R_2, R_1 kot na prejšnji strani. Velja

$$A'' = Q_2^T Q_1^T A Q_1 Q_2.$$

Vsota $\sigma + \overline{\sigma}$ je kar sled spodnjega 2×2 vogala, produkt $\sigma\overline{\sigma}$ pa njegova determinanta.

Reševanje nelinearnih enačb in optimizacija

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

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

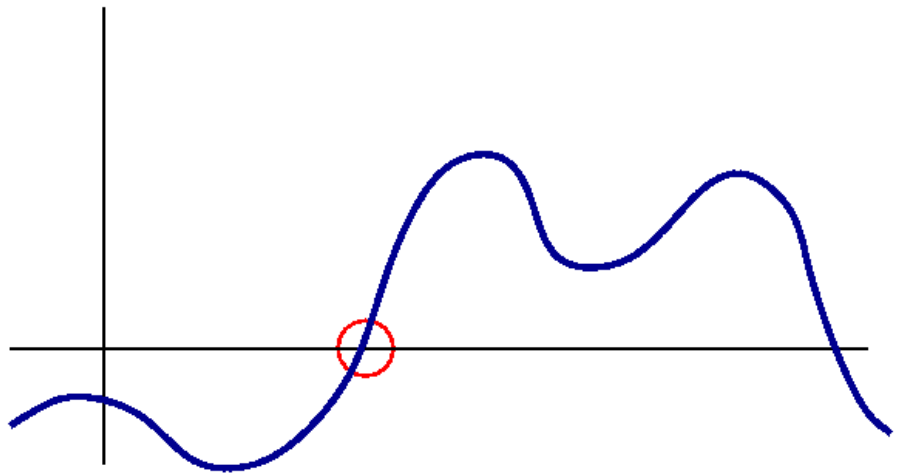
$$* \; \min\{f(x) : x \in K \subseteq \mathbb{R}^n\}$$

- 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

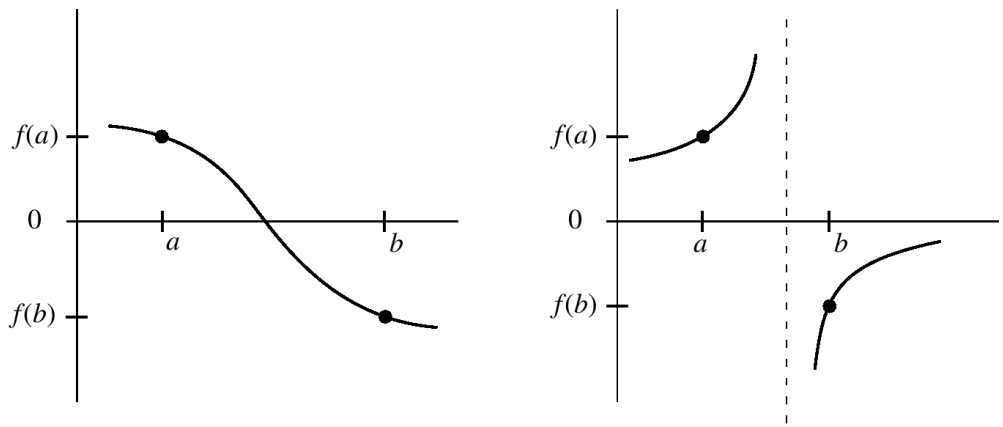
$$f(x) = 0.$$



- 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:

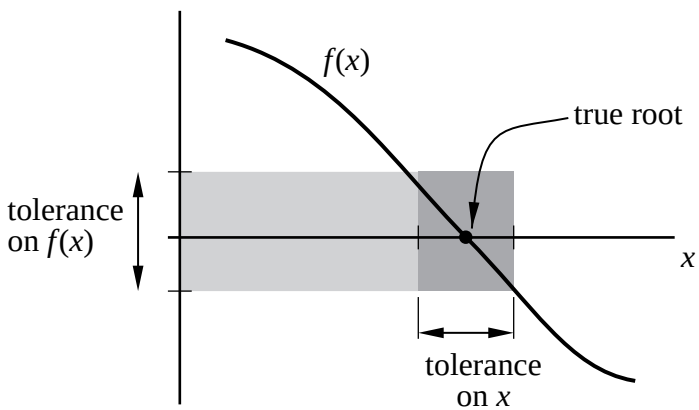
- Lahko nas zanima, kdaj velja

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

- Lahko pa nas zanima, kdaj velja

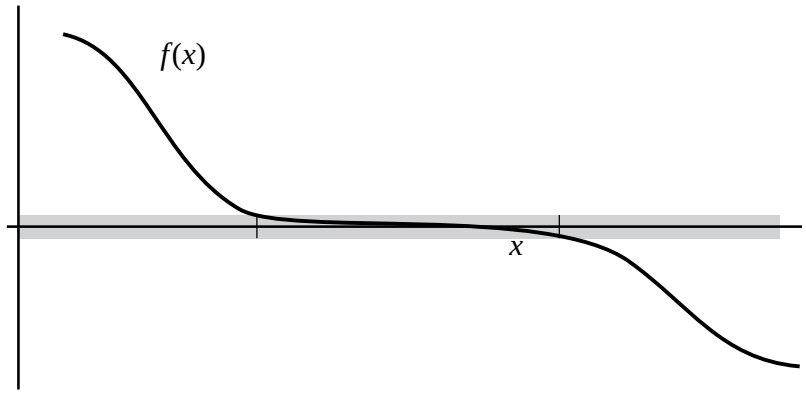
$$|f(x_k)| < \text{toleranca}.$$

- Še najbolje pa je zahtevati izpolnjenost obeh pogojev hkrati.

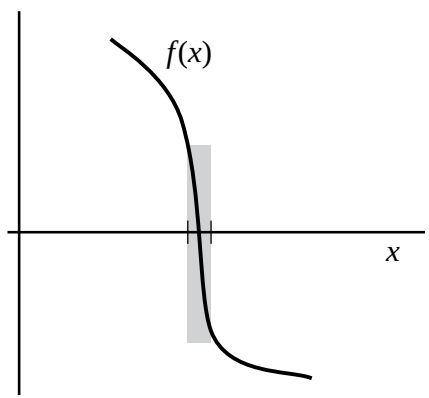


Primerjava obeh konvergenčnih kriterijev

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.

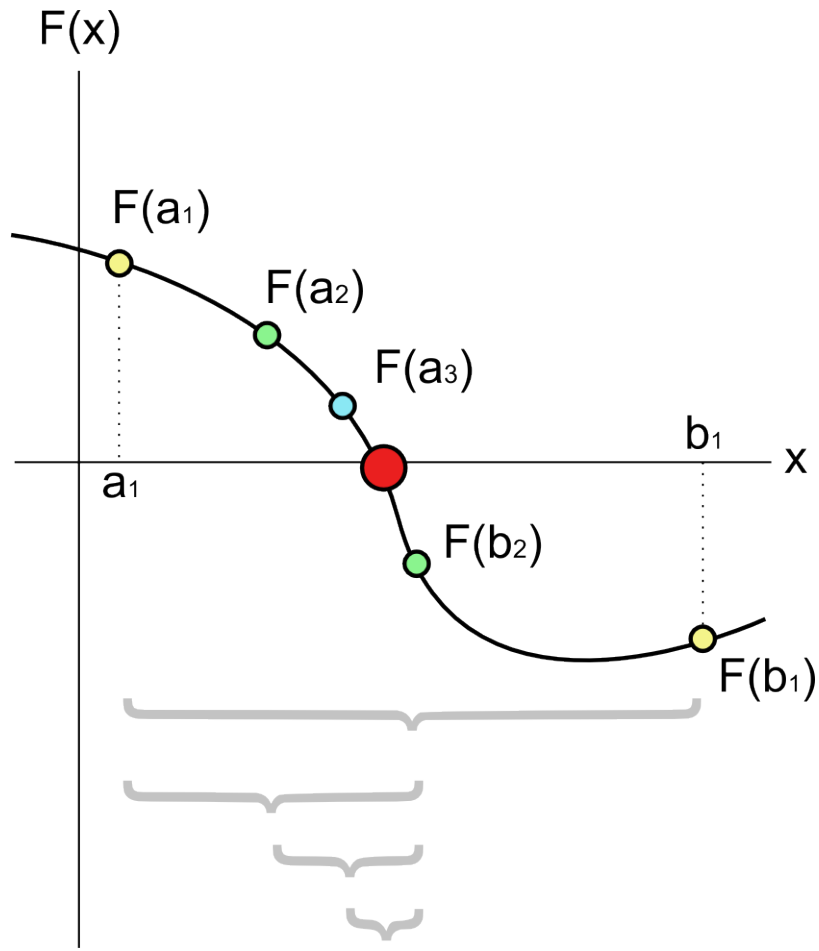
Bisekcija

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

```
1   Vhod: funkcija f, krajisci a, b, toleranca tol
2   Izhod: priblizek x* za f(x) = 0 z |x - x*| < tol
3
4   for k = 1, 2, ...
5      x_m = a + (b - a)/2
6      if sign(f(x_m)) = sign(f(a))
7         a = x_m
8      else
9         b = x_m
10     end
11     if |b-a|<tol, stop
12  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

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

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

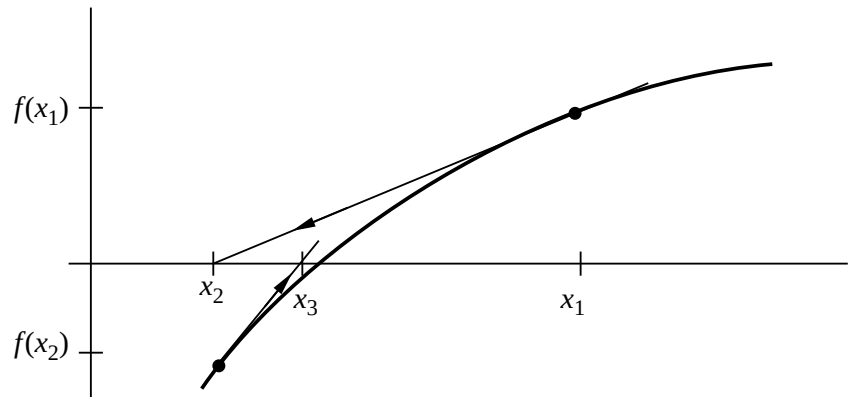
n	$\frac{\delta_n}{\delta_0}$	število izračunov funkcijskih vrednosti
5	3.1×10^{-2}	7
10	9.8×10^{-4}	12
20	9.5×10^{-7}	22
30	9.3×10^{-10}	32
40	9.1×10^{-13}	42
50	8.9×10^{-16}	52

Tangentna metoda

Iteracija:

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}. \tag{13}$$

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

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

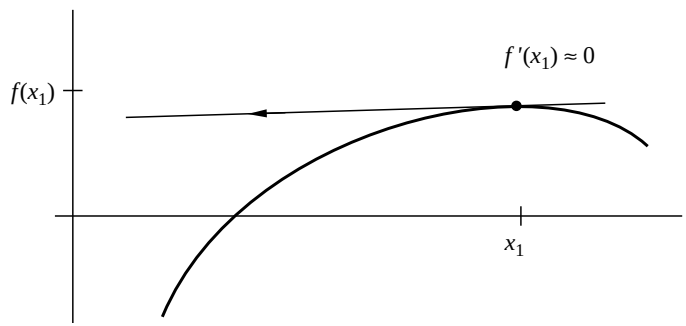
in izrazimo x_{k+1}.

```
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:



Hitrost konvergence tangentne metode

Spomnimo se

Metoda ima hitrost konvergence reda r, če obstaja konstanta $C > 0$, tako da je

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

Izrek

Tangentna metoda ima hitrost konvergence reda 2 v okolici rešitve x^* vedno, ko velja $f'(x_*) \neq 0$.

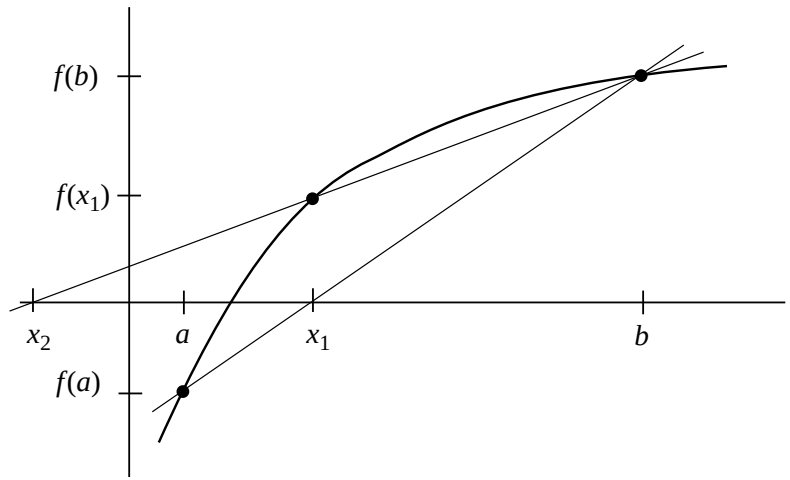
Dokaz: klik

Sekantna metoda

Iteracija:

$$x_{k+1} = x_k - f(x_k)\left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right] \tag{14}$$

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

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

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

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

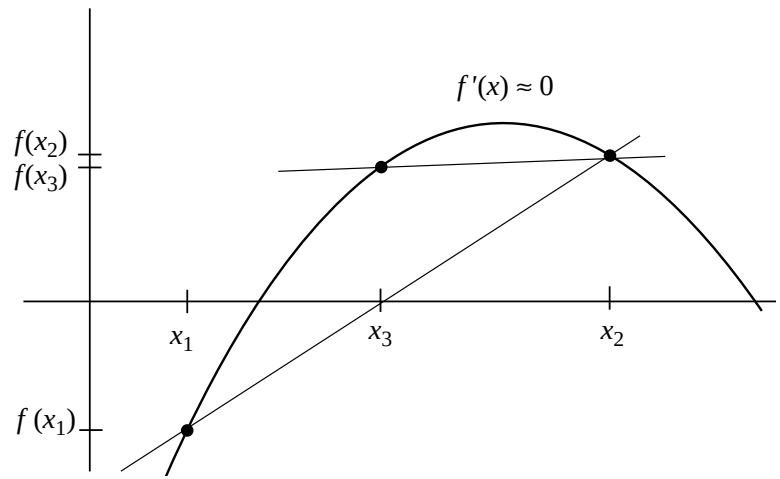
Vstavimo to aproksimacijo v (13) in dobimo (14).

```
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

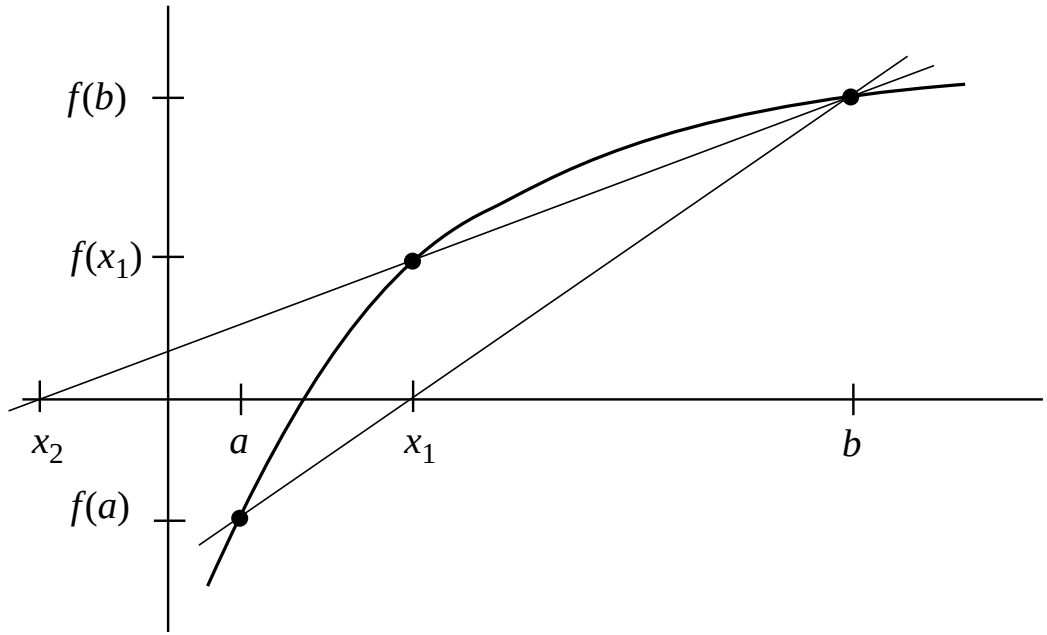
Lastnosti sekantne metode:

- 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 oz. navadno iteracijo 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

Izrek (Konvergenčni izrek za navadno iteracijo)

Naj bo g zvezno odvedljiva na intervalu $I = [a, b]$ in naj velja $g(I) \subseteq I$. Naj bo še $\sup_{x \in I} |g'(x)| = m < 1$. Velja:

- *$g(x) = x$ ima enolično rešitev ξ na I.*
- *Za vsak $x_0 \in I$ zaporedje $x_n = g(x_{n-1})$ konvergira proti ξ, pri čemer je*

$$|x_{n+1} - \xi| \leqslant \min \left\{ m^{n+1}|x_0 - \xi|, \frac{m}{1-m}|x_{n+1} - x_n|, \frac{m^{n+1}}{1-m}|x_1 - x_0| \right\}.$$

Dokaz: klik

Izrek (Hitrost konvergence navadne iteracije)

Naj bo g v okolici negibne točke ξ p-krat zvezno odvedljiva in velja

$$g'(\xi) = g''(\xi) = \cdots = g^{(p-1)}(\xi) = 0 \qquad \textit{in} \qquad g^{(p)}(\xi) \neq 0.$$

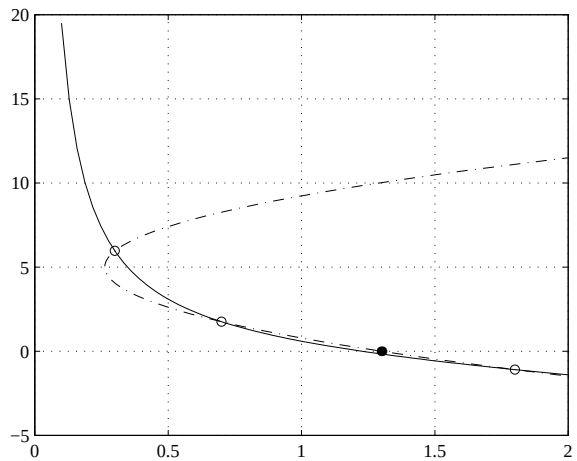
Potem je red konvergence zaporedja $x_{n+1} = g(x_n)$ enak p.

Dokaz: 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.




```
1 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

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

Izpeljava:

1. Funkcije f_i razvijemo v Taylorjevo vrsto:

$$f_i(\underline{x} + \Delta\underline{x}) = f_i(\underline{x}) + \sum_{k=1}^{n} \frac{\partial f_i}{\partial x_k}(\underline{x})\Delta x_k + \ldots, \quad i = 1, \ldots, n.$$

2. Zanemarimo člene višjega reda in enačimo $f_i(\underline{x} + \Delta\underline{x}) = 0$.
3. Dobimo zgornji sistem.

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

Izrek (Prvi konvergenčni izrek Jacobijeve iteracije)

Naj $\underline{g} : \mathbb{R}^n \to \mathbb{R}^n$ *na nekem območju* $\Omega \subseteq \mathbb{R}^n$ *zadošča:*

1. $\underline{g}(\Omega) \subseteq \Omega$.
2. $\|\underline{g}(\underline{x}) - \underline{g}(\underline{y})\| \leqslant m\|\underline{x} - \underline{y}\|$ *za vsaka* $\underline{x}, \underline{y} \in \Omega$ *nek* $0 \leqslant m < 1$.

Enačba

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

ima na območju Ω *eno samo rešitev* $\underline{\xi}$ *in zaporedje* $\underline{x}^{(r+1)}$ *konvergira proti* $\underline{\xi}$ *za poljuben začetni približek* $\underline{x}^{(0)} \in \Omega$. *Velja še*

$$\|\underline{x}^{(r+1)} - \underline{\xi}\| \leqslant \frac{m^r}{1-m}\|\underline{x}^{(1)} - \underline{x}^{(0)}\|.$$

Izrek (Drugi konvergenčni izrek Jacobijeve iteracije)

Naj $\underline{g} : \mathbb{R}^n \to \mathbb{R}^n$ *zvezno odvedljiva v negibni točki* $\underline{\xi}$ *in naj bo* $\|J_{\underline{g}}(\underline{\xi})\| < 1$.
Potem obstaja zaprta okolica $\Omega \subseteq \mathbb{R}^n$ *fiksne točke* $\underline{\xi}$, *tako da zaporedje* $\underline{x}^{(r+1)}$ *konvergira proti* $\underline{\xi}$ *za poljuben začetni približek* $\underline{x}^{(0)} \in \Omega$.

Kvazi-Newtonove metode: Broydenova metoda

- Pri velikim številu enačb je Newtonova metoda zelo zahtevna, saj potrebujemo na vsakem koraku n^2 parcialnih odvodov in $\mathcal{O}(n^3)$ računskih operacij za reševanje linearnega sistema.
- Kot se pri navadni tangentni metodi izognemo računanja odvodov z uporabo sekantne metode, se tudi pri kvazi-Newtonovih metodah izognemu računanju parcialnih odvodov. Najbolj znana je Broydenova metoda.

Naj bo B_r približek za $J_{\underline{f}}(\underline{x}^{(r)})$. Korak kvazi-Newtonove metode je:

1. reši $B_r \Delta \underline{x}^{(r)} = -\underline{f}(\underline{x}^{(r)})$,
2. $\underline{x}^{(r+1)} = \underline{x}^{(r)} + \Delta \underline{x}^{(r)}$,
3. določi B_{r+1}.

Pri Broydenovi metodi za B_{r+1} matriko, ki zadošča t.i. sekantnemu pogoju

$$B_{r+1}(\underline{x}^{(r+1)} - \underline{x}^{(r)}) = \underline{f}(\underline{x}^{(r+1)}) - \underline{f}(\underline{x}^{(r)})$$

in je v spektralni normi najbližje B_r (tj. največja lastna vrednost razlike $B_{r+1} - B_r$ je najmanjša možna).

Iščemo torej $\Delta B_r = B_{r+1} - B_r$ z minimalno spektralno normo, ki zadošča $\Delta B_r \Delta \underline{x}^{(r)} = f(\underline{x}^{(r+1)})$.

Trditev (Približek za Jacobijevo matriko v Broydenovi metodi)

$$B_{r+1} = B_r + \frac{\underline{f}(\underline{x}^{(r+1)})(\Delta \underline{x}^{(r)})^T}{\|\Delta \underline{x}^{(r)}\|_2^2}.$$

Dokaz: klik

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:

$$\underline{x}^{(r+1)} = \underline{x}^{(r)} + \lambda_r v_r, \tag{15}$$

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 (15):

- 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 (15): 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

$$\nabla g(x) = \left[\begin{array}{ccc} \frac{\partial g(x)}{\partial x_1} & \cdots & \frac{\partial g(x)}{\partial x_n} \end{array} \right] = 0.$$

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

$$H_g(\underline{x}) = \left[\begin{array}{ccc} \frac{\partial^2 g(\underline{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 g(\underline{x})}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 g(\underline{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 g(\underline{x})}{\partial x_n^2} \end{array} \right].$$

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

Gauss-Newtonova iteracija

Rešujemo predoločen sistem

$$f : \mathbb{R}^n \to \mathbb{R}^m, \quad f(x) = (0, \ldots, 0) \tag{16}$$

m nelinearnih enačb v n neznankah, kjer je $m > n$.

Sistem (16) v splošnem nima rešitve. Radi bi našli rešitev (16) po metodi najmanjših kvadratov, tj. iščemo $\alpha \in \mathbb{R}^n$, ki minimizira

$$\|f(\alpha)\|^2 = \min\{\|f(x)\|^2\}.$$

Gauss-Newtonova metoda je posplošitev Newtone metode, kjer na vsakem koraku iteracije namesto inverza Jacobijeve matrike računamo psevdoinverz:

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

Na vsakem koraku je $\underline{x}^{(r+1)}$ rešitev predoločenega linearnega sistema

$$J_{\underline{f}}(\underline{x}^{(r)})\Delta\underline{x}^{(r)} = \underline{f}(\underline{x}^{(r)})$$

po metodi najmanjših kvadratov.

Konvergenca ni zagotovljena, pa tudi rešitev v primeru konvergence je lahko zgolj lokalni, ne pa nujno globalni minimum funkcije $\|\underline{f}(\underline{x})\|^2$.

Primer

Dane so točke $(x_i, y_i) \in \mathbb{R}^2$ *for* $i = 1, \ldots, m$. *Iščemo funkcijo*

$$f(x, a, b) = ae^{bx},$$

ki se najbolje prilega podatkom po metodi najmanjših kvadratov.

Predoločen sistem, ki ga rešujemo, je $F(a, b) = 0$, *kjer je*

$$F : \mathbb{R}^2 \to \mathbb{R}^m, \quad F(a, b) = (y_1 - ae^{bx_1}, \ldots, y_m - ae^{bx_m}).$$

Jacobijeva matrika za F je

$$DF(a, b) = \begin{bmatrix} -e^{bx_1} & ax_1 e^{bx_1} \\ \vdots & \\ -e^{bx_m} & ax_m e^{bx_m} \end{bmatrix}.$$

Gauss-Newtonova metoda:

- *Izberi začetni približek* (a_0, b_0),
- *Iteriraj:*

$$\begin{bmatrix} a_{r+1} \\ b_{r+1} \end{bmatrix} = \begin{bmatrix} a_r \\ b_r \end{bmatrix} - DF(a_r, b_r)^+ F(a_r, b_r)^T.$$

Algoritem: klik Primer: 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, ortogonalni polinomi

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:

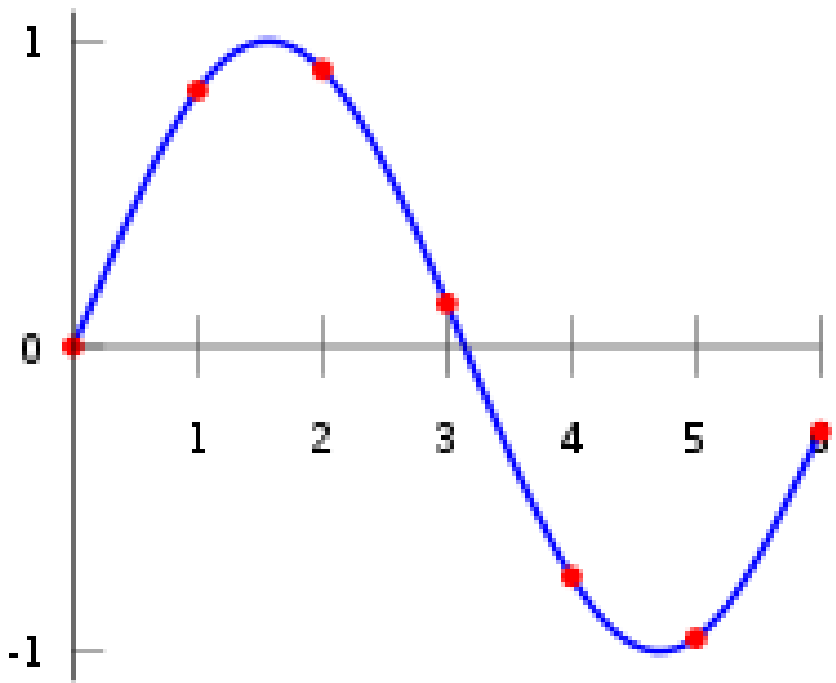
$$n+1 \quad \text{točk} \quad x_0, \ldots, x_n \quad \text{in vrednosti} \quad y_0, \ldots, y_n.$$

Iščemo polinom

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

stopnje n, ki zadošča

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad \ldots, \quad p(x_n) = y_n. \tag{17}$$



Dobimo sistem

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0, \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1, \\ &\vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n. \end{aligned} \tag{18}$$

Polinomu $p(x)$ pravimo interpolacijski polinom.

V matrični obliki lahko sistem (18) 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

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

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 (18)?
2. Ali je sistem (18) 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

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

je bolje uporabiti eno od naslednjih baz:

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

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

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:

$$\frac{f(x_j) - f(x_i)}{x_j - x_i}. \tag{19}$$

Če izraz (19) 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*

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

Dokaz: klik

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	$f[x_3]$			

Koda: klik klik klik

Primeri: klik klik

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

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

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

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

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:

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

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

$$e_n(t) = p_n(t) - f(t) = -f[x_0, x_1, \ldots, x_n, t] \prod_{i=0}^{n} (t - x_i).$$

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

Izrek (O deljenih diferencah)

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

Dokaz: *klik*

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.

Aproksimacija po metodi najmanjših kvadratov

Do ekvivalentnega sistema bi prišli tudi z zapisamo prvotnega sistema v obliki $Ax = b$ in mu priredili normalni sistem $A^T Ax = A^T b$:

$$
\begin{aligned}
a_0 + a_1 \sum_{i=1}^{m} x_i + \ldots + a_n \sum_{i=1}^{m} x_i^n &= \sum_{i=1}^{m} f(x_i), \\
a_0 \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2 + \ldots + a_n \sum_{i=1}^{m} x_i^{n+1} &= \sum_{i=1}^{m} f(x_i) x_i, \\
a_0 \sum_{i=1}^{m} x_i^2 + a_1 \sum_{i=1}^{m} x_i^3 + \ldots + a_n \sum_{i=1}^{m} x_i^{n+2} &= \sum_{i=1}^{m} f(x_i) x_i^2, \\
&\vdots \\
a_0 \sum_{i=1}^{m} x_i^n + a_1 \sum_{i=1}^{m} x_i^{n+1} + \ldots + a_k \sum_{i=1}^{m} x_i^{2n} &= \sum_{i=1}^{m} f(x_i) x_i^n.
\end{aligned}
$$

Ta sistem lahko rešimo z Gaussovo eliminacijo, vendar je lahko pri veliko točkah slabo pogojen. Cilj je problem preoblikovati v ekvivalentnega, vendar bolje pogojenega. Rešitev leži v uporabi baze ortogonalnih polinomov.

Zamenjava baze prostora polinomov

Če bi namesto baze $\{1, x, \ldots, x^n\}$ vzeli novo bazo polinomov

$$\{g_0(x), g_1(x), \ldots, g_n(x)\},$$

bi dobili sistem linearnih enačb:

$$a_0 \sum_{i=1}^{m} g_0^2(x_i) + a_1 \sum_{i=1}^{m} g_0(x_i)g_1(x_i) + \ldots + a_n \sum_{i=1}^{m} g_0(x_i)g_n(x_i) = \sum_{i=1}^{m} f(x_i)g_0(x_i),$$

$$a_0 \sum_{i=1}^{m} g_1(x_i)g_0(x_i) + a_1 \sum_{i=1}^{m} g_1^2(x_i) + \ldots + a_n \sum_{i=1}^{m} g_1(x_i)g_n(x_i) = \sum_{i=1}^{m} f(x_i)g_1(x_i),$$

$$a_0 \sum_{i=1}^{m} g_2(x_i)g_0(x_i) + a_1 \sum_{i=1}^{m} g_2(x_i)g_1(x_i) + \ldots + a_n \sum_{i=1}^{m} g_2(x_i)g_n(x_i) = \sum_{i=1}^{m} f(x_i)g_2(x_i),$$

$$\vdots$$

$$a_0 \sum_{i=1}^{m} g_n(x_i)g_0(x_i) + a_1 \sum_{i=1}^{m} g_n(x_i)g_1(x_i) + \ldots + a_n \sum_{i=1}^{m} g_n^2(x_i) = \sum_{i=1}^{m} f(x_i)g_n(x_i).$$

Želeli bi izbrati bazo, v kateri bo ta sistem diagonalen. Iščemo torej polinome g_j, za katere velja

$$\sum_{i=1}^{m} g_j(x_i)g_k(x_i) = 0 \qquad \text{za } j \neq k.$$

Zgornji sistem lahko zapišemo tudi v matrični obliki:

$$A = \begin{pmatrix} \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle & \dots & \langle g_0, g_n \rangle \\ \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle & \dots & \langle g_1, g_n \rangle \\ \vdots & & & \vdots \\ \langle g_0, g_n \rangle & \langle g_1, g_n \rangle & \dots & \langle g_n, g_n \rangle \end{pmatrix}, \qquad x = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}, \qquad b = \begin{pmatrix} \langle f, g_0 \rangle \\ \langle f, g_1 \rangle \\ \vdots \\ \langle f, g_n \rangle \end{pmatrix},$$

kjer je $\langle f, g \rangle := \sum_{i=1}^{m} f(x_i) g(x_i)$. Iščemo torej polinome g_j, za katere velja

$$\langle g_j, g_k \rangle = 0 \qquad \text{za } j \neq k. \tag{20}$$

Zaporedju polinomov $g_0, \dots, g_n$, za katere velja $\deg g_j = j$, vodilni koeficient g_j je 1 in velja (20), pravimo zaporedje ortogonalnih polinomov (ZOP).

Rešitev našega problema je ZOP g_j polinom p:

$$p(x) = \sum_{i=0}^{n} a_i g_i(x), \qquad a_i = \frac{\langle f, g_i \rangle}{\langle g_i, g_i \rangle}.$$

V nadaljevanju se bomo ukvarjali s tem, kako konstruirati ZOP.

Ortogonalni polinomi

Izrek (Ortogonalnost)

Naj bo $g_0, \ldots, g_n$ zaporedje ortogonalnih polinomov glede na $\langle \cdot, \cdot \rangle$. Potem je

$$\langle g_k, p \rangle = 0$$

za vsak polinom stopnje p največ $k - 1$. *Dokaz: klik*

Izrek (Tričlenska rekurzivna zveza)

Zaporedje ortogonalnih polinomov glede na $\langle \cdot, \cdot \rangle$ zadošča tričlenski rekurzivni relaciji

$$g_{i+1}(x) = (x - \alpha_{i+1}) g_i(x) - \beta_i g_{i-1}(x),$$

kjer so koeficienti α_i, β_i, $i \in \mathbb{N}$ določeni z enačbami

$$\alpha_{i+1} = \frac{\langle x g_i, g_i \rangle}{\langle g_i, g_i \rangle}, \quad \beta_0 = 0, \quad \beta_i = \frac{\langle x g_i, g_{i-1} \rangle}{\langle g_{i-1}, g_{i-1} \rangle}.$$

Dokaz: klik

Gaussove kvadraturne formule

Izrek (Ničle ortogonalnih polinomov)

Naj bo $g_0, \ldots, g_n$ zaporedje ortogonalnih polinomov glede na

$$\langle f, g \rangle_w = \int_a^b fgw \, dx,$$

kjer je w pozitivna utež. Potem ima g_i i različnih ničel na intervalu $[a, b]$.

Dokaz: klik

Gaussova kvadraturna formula (GKF) reda n je predpis, s katerim ocenimo $\int_a^b f(x)w(x) \, dx$:

$$\int_a^b f(x)w(x) \, dx \approx \sum_{i=1}^{n} \rho_i f(x_i),$$

kjer so $x_1, \ldots, x_n \in [a, b]$ vozli, $q_1, \ldots, q_n$ pa uteži.

Cilj GKF je, da je točna za integracijo polinomov čim višjih stopenj.

Ker imamo $2n$ prostih parametrov, lahko dobimo z ustrezno izbiro vozlov in uteži točno formulo do stopnje $2n - 1$.

Enačbe, ki jih moramo rešiti za določitev x_i, q_i so nelinearne in težko analitično rešljive. Velja pa naslednji izrek:

Izrek (O vozlih in utežeh v GKF reda n)

Naj bo q_n n–ti ortogonalni polinom glede na $\langle \cdot, \cdot \rangle$. Potem so $x_1, \ldots, x_n$ njegove ničle, uteži ρ_i pa so enake $\int_a^b \ell_i w \, dx$, kjer je ℓ_i i–ta Lagrangeova bazna funkcija na točkah $x_1, \ldots, x_n$.

Dokaz: klik

Numerična integracija

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

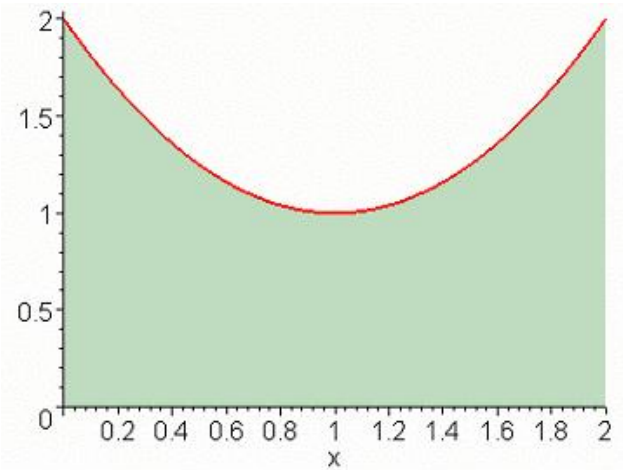
- Newton–Cotesova (NC) pravila: trapezno, Simpsonovo pravilo
- Izbira koraka v NC pravilih
- Adaptivna NC pravila
- Rombergova metoda
- Gaussove kvadraturne formule
- Integracija v več spremenljivkah

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.

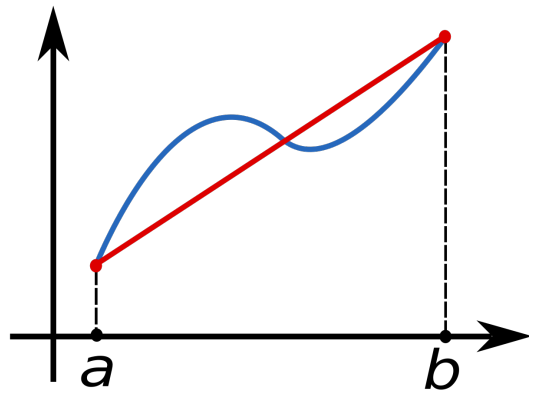


Če ne znamo izračunati nedoločenega integrala F, smo v težavah. Npr. za $f(x) = e^{-x^2}$, $g(x) = \frac{\sin x}{x}$, $h(x) = x \tan x$.

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:

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

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.

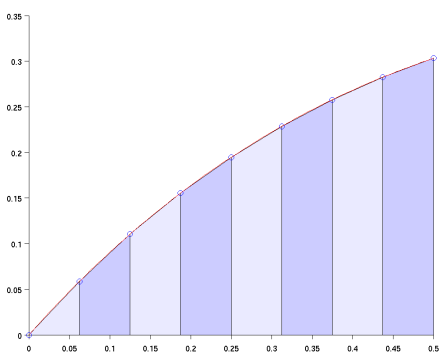
Če interval $[a, b]$ razdelimo z ekvidistantnimi točkami $x_0, x_1, \ldots, x_n$, tj.

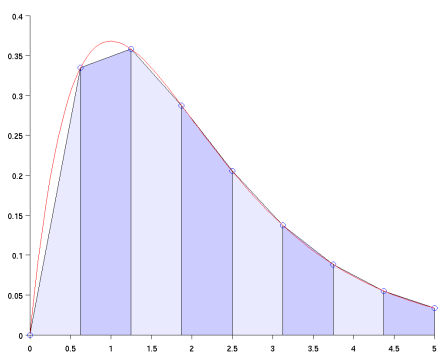
$$h := h_i = x_{i+1} - x_i$$

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

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

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:

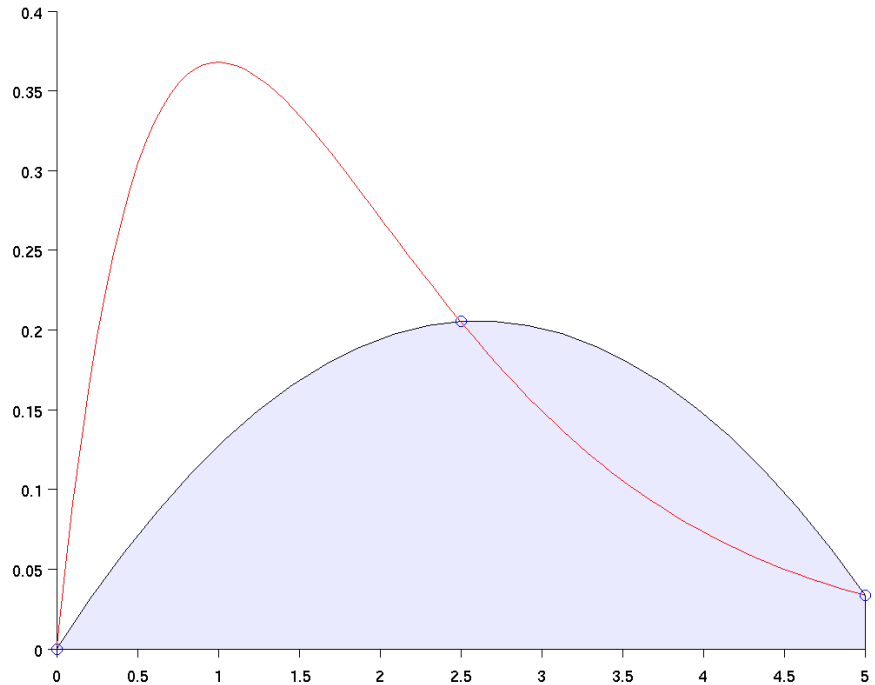
- Najprej izračunamo $T(b-a)$ in $T((b-a)/2)$.
- Če je podobno kot pri kontroli koraka zgoraj ocena napake $e := \frac{T(b-a)/2 - T(b-a)}{3}$ dovolj majhna, vrnemo $T((b-a)/2) + e$ in končamo.
- Če je e prevelik, ponovimo zgornji postopek ločeno za podintervala $[a, (a+b)/2]$ in $[(a+b)/2, b]$, pri čemer 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

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



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

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 $x = a + t$);

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

Izkaže se, da je napaka približno:

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

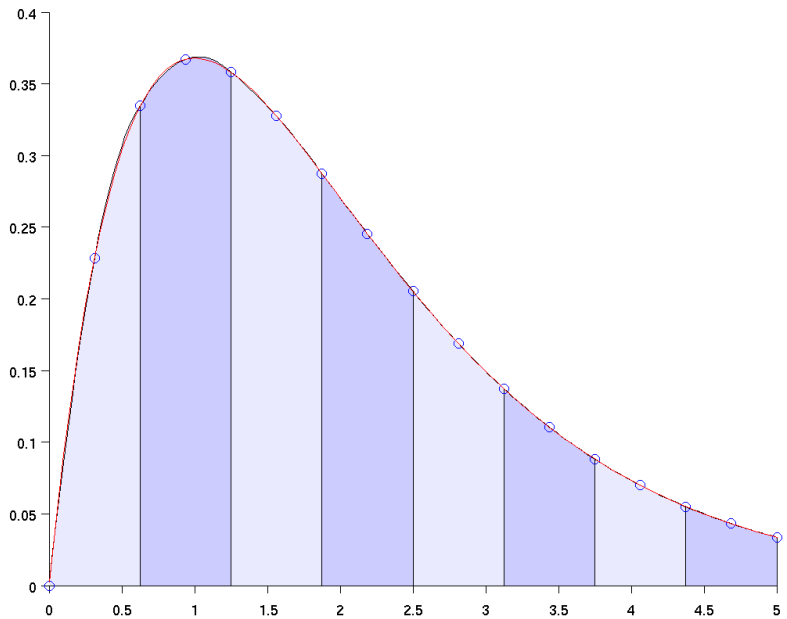
Algoritem: klik

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

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

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

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:

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

Rombergova metoda

Naj bo $T(h)$ aproksimacija integrala $I = \int_a^b f(x)\,dx$ z uporabo sestavljenega trapeznega pravila s korakom h.

Če je naša funkcija $(2k+2)$-krat odvedljiva, lahko napako trapezne formule (z uporabo Euler-Maclaurinove vrste) razvijemo v konvergentno vrsto po sodih potencah h:

$$\boxed{E(h) := T(h) - I = C_1 h^2 + C_2 h^4 + \ldots + C_k h^{2k} + \mathcal{O}(h^{2k+2})}, \tag{21}$$

pri čemer so konstante $C_1, \ldots, C_k$ neodvisne! od h.

V (21) vnesemo h, $h/2$, $h/4$, $\ldots$, $h/2^k$ in dobimo:

$$I = T(h) + C_1 h^2 + C_2 h^4 + \ldots + C_\ell h^{2\ell} + \mathcal{O}(h^{2\ell+2}), \tag{22}$$

$$I = T(h/2) + C_1 \left(\frac{h}{2}\right)^2 + C_2 \left(\frac{h}{2}\right)^4 + \ldots + C_\ell \left(\frac{h}{2}\right)^{2\ell} + \mathcal{O}\left(\left(\frac{h}{2}\right)^{2\ell+2}\right), \tag{23}$$

$$I = T(h/4) + C_1 \left(\frac{h}{4}\right)^2 + C_2 \left(\frac{h}{4}\right)^4 + \ldots + C_\ell \left(\frac{h}{4}\right)^{2\ell} + \mathcal{O}\left(\left(\frac{h}{4}\right)^{2\ell+2}\right), \tag{24}$$

Z uporabo (22) in (23) bi se radi znebili člena pri h^2. Pomnožimo (23) s 4, da dobimo

$$4I = 4T(h/2) + C_1 h^2 + 4C_2 \left(\frac{h}{2}\right)^4 + \ldots + 4C_\ell \left(\frac{h}{2}\right)^{2\ell} + \mathcal{O}\left(\left(\frac{h}{2}\right)^{2\ell+2}\right), \tag{25}$$

Sedaj od (25) odštejemo (22) in dobimo

$$3I = 4T(h/2) - T(h) + C_2 \left(\frac{1}{2^2} - 1\right) h^4 + \ldots + C_\ell \left(\frac{1}{2^{2\ell-2}} - 1\right) h^{2\ell} + \mathcal{O}(h^{2\ell+2}),$$

oziroma

$$I = \underbrace{\frac{4T(h/2) - T(h)}{3}}_{T_1(h/2)} + \frac{C_2}{3} \left(\frac{1}{2^2} - 1\right) h^4 + \ldots + \frac{C_\ell}{3} \left(\frac{1}{2^{2\ell-2}} - 1\right) h^{2\ell} + \mathcal{O}(h^{2\ell+2}).$$

Podobno z uporabo (23) in (24) pridemo do enakosti

$$I = \underbrace{\frac{4T(h/4) - T(h/2)}{3}}_{T_1(h/4)} + \frac{C_2}{3} \left(\frac{1}{2^2} - 1\right) \left(\frac{h}{2}\right)^4 + \ldots + \frac{C_\ell}{3} \left(\frac{1}{2^{2\ell-2}} - 1\right) \left(\frac{h}{2}\right)^{2\ell} + \ldots.$$

V splošnem dobimo:

$$I = \underbrace{\frac{4T\left(\frac{h}{2^k}\right) - T\left(\frac{h}{2^{k-1}}\right)}{3}}_{T_1(h/2^k)} + \frac{C_2}{3}\left(\frac{1}{2^2} - 1\right)\left(\frac{h}{2^{k-1}}\right)^4 + \ldots + \frac{C_\ell}{3}\left(\frac{1}{2^{2\ell-2}} - 1\right)\left(\frac{h}{2^{k-1}}\right)^{2\ell} + \mathcal{O}\left(\left(\frac{h}{2^{k-1}}\right)^{2\ell+2}\right).$$

Torej so

$$\boxed{T_1(h/2^\ell) = \frac{4T(h/2^\ell) - T(h/2^{\ell-1})}{3}}, \quad \ell = 1, 2, 3, \ldots, k$$

boljši približki za I, saj je

$$I = T_1(h/2^\ell) + \mathcal{O}\left(\left(\frac{h}{2^{\ell-1}}\right)^4\right).$$

Lahko bi nadaljevali zgornji postopek in dobili;

$$\boxed{T_2(h/2^\ell) = \frac{4^2 T_1(h/2^\ell) - T_1(h/2^{\ell-1})}{4^2 - 1}}, \quad \ell = 2, 3, 4, \ldots, k.$$

Na m-tem koraku, kjer je $m = 1, \ldots, k$, dobimo

$$\boxed{T_m(h/2^\ell) = \frac{4^m T_{m-1}(h/2^\ell) - T_{m-1}(h/2^{\ell-1})}{4^m - 1}}, \quad \ell = m, m+1, m+2, \ldots, k$$

in

$$I = T_m(h/2^\ell) + \mathcal{O}\left(\left(\frac{h}{2^{\ell-1}}\right)^{2(m+1)}\right).$$

Vrednosti $T_m(h/2^\ell)$ v kompaktni obliki računamo s pomočjo trikotne tabele:

$$\begin{array}{lllll}
T(h) & & & & \\
T(h/2) & T_1(h/2) & & & \\
T(h/2^2) & T_1(h/2^2) & T_2(h/2^2) & & \\
T(h/2^3) & T_1(h/2^3) & T_2(h/2^3) & T_3(h/2^3) & \\
T(h/2^4) & T_1(h/2^4) & T_2(h/2^4) & T_3(h/2^4) & T_4(h/2^4),
\end{array}$$

pri čemer $T_m(h/2^\ell)$ izračunamo tako, da:

- pomnožimo element levo od njega z utežjo $\dfrac{4^m}{4^m - 1}$
- odštejemo element za eno levo in eno navzgor pomnožen z utežjo $\dfrac{1}{4^m - 1}$.

Algoritem: klik

Gaussove kvadraturne formule

- NC pravila za integriranje so oblike

$$\int_a^b f(x)\,dx \approx \sum_{j=0}^{n} w_j f(x_j), \tag{26}$$

  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 (26) 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.

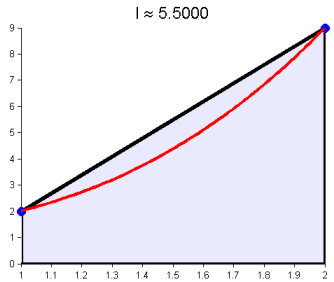
Primer najboljših vozlov za interval $[-1, 1]$

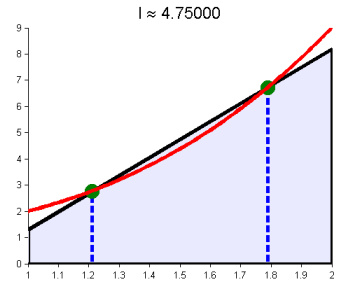
Oglejmo si primer $n = 1$ (tj. 2 točki) na primeru intervala $[-1, 1]$. Poiščimo w_0, w_1, x_0, x_1, tako da velja

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

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

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

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

Od tod sledi:

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

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

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

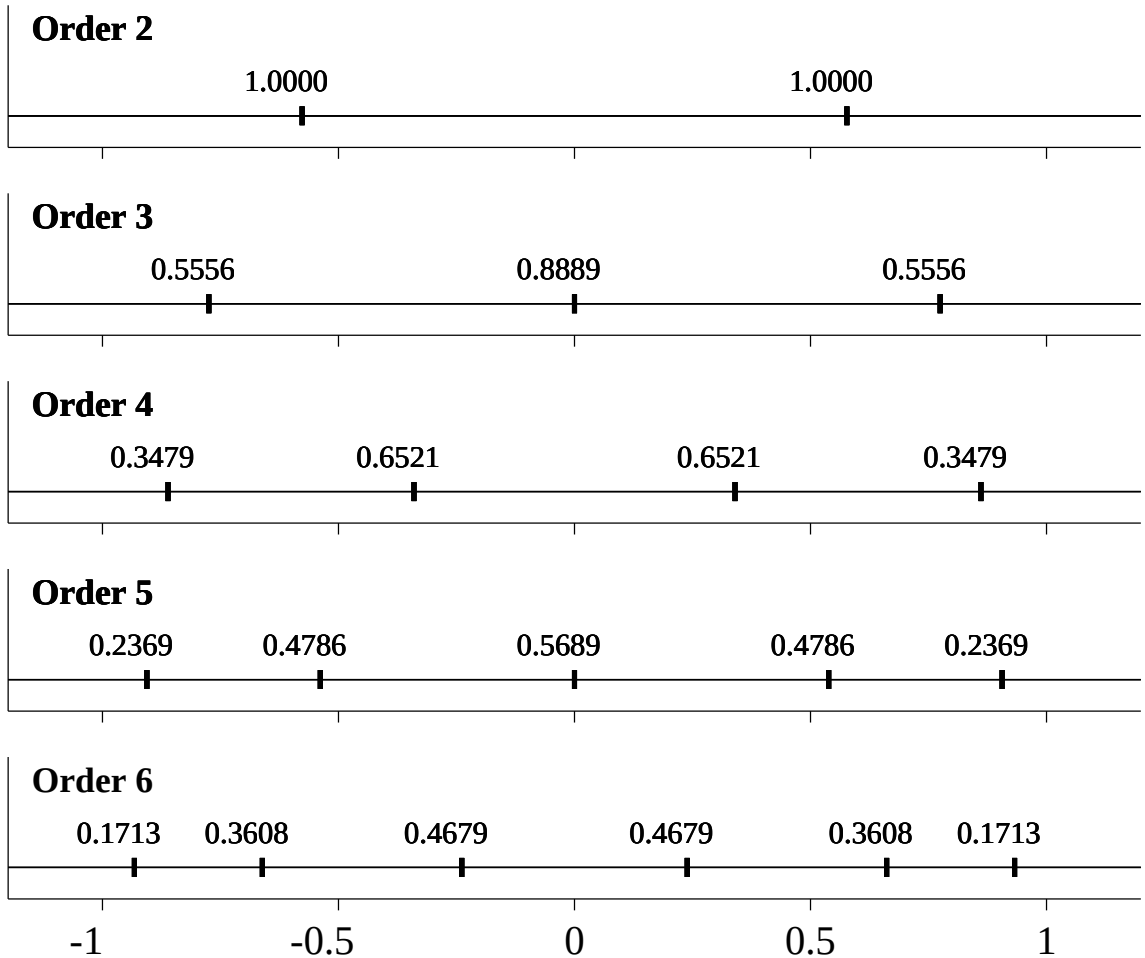
kjer je

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

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 $2n+1$.

Dokaz: klik

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

$$\Omega = [a,b] \times [c,d] \subseteq \mathbb{R}^2.$$

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:

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

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

Kompaktneje lahko to povemo tako, da pomnožimo istoležne koeficiente v tabeli funkcijskih vrednosti

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

s tabelo koeficientov

$$\frac{hk}{4}\cdot\begin{pmatrix} 1 & 2 & 2 & \cdots & 2 & 1 \\ 2 & 4 & 4 & \cdots & 4 & 2 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 2 & 4 & 4 & \cdots & 4 & 2 \\ 1 & 2 & 2 & \cdots & 2 & 1 \end{pmatrix} = \frac{hk}{4}\begin{pmatrix} 1 \\ 2 \\ \vdots \\ \vdots \\ 2 \\ 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & \cdots & \cdots & 2 & 1 \end{pmatrix}$$

in seštejemo vse vrednosti v dobljeni matriki.

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

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

kjer je $h = \frac{b-a}{2n}$, $k = \frac{d-c}{2n}$ in $u^T = \begin{pmatrix} 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 1 \end{pmatrix}$.

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:

$$F(t, x, \dot{x}, \ddot{x}, \ldots, x^{(n)}) = 0, \tag{28}$$

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

$$F(x, y', y'', \ldots, y^{(n)}) = 0. \tag{29}$$

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 (28) oz. (29) na definicijskem območju $\mathcal{D}$ neodvisne spremenljivke.

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

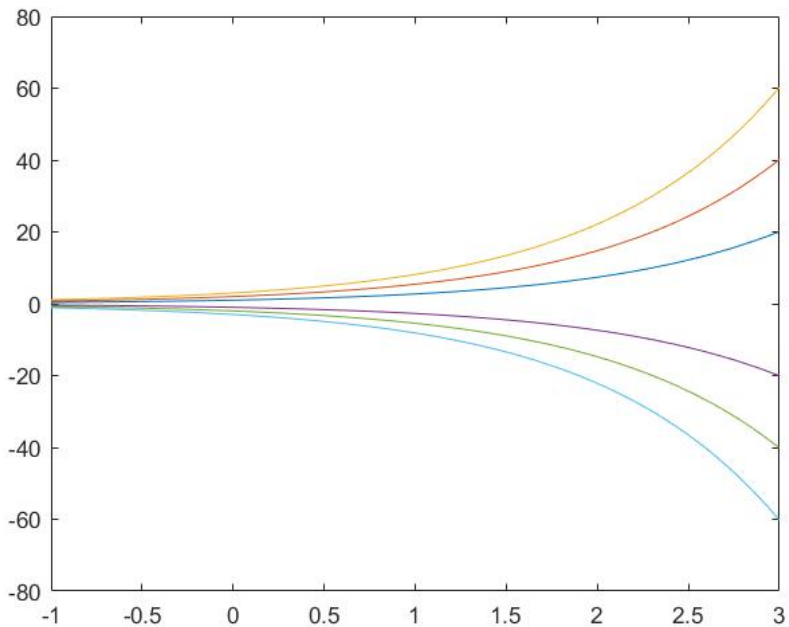
Primeri. $y' = y$, $y' + 5xy = 3x^2$, $\dot{x} + x = 0$, $\ddot{x} + a\dot{x} + bx = A\cos\omega t$.

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

Primer

Rešimo DE $y' = y$.

$$\begin{aligned} \frac{dy}{dx} = y \quad &\Rightarrow \quad \frac{dy}{y} = dx \quad \Rightarrow \quad \int \frac{dy}{y} = \int dx \\ &\Rightarrow \quad \log(|y|) = x + C, \quad C \in \mathbb{R} \\ &\Rightarrow \quad y = Ke^x, \quad K \in \mathbb{R} \end{aligned}$$



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 $\dot{x} = \frac{dx}{dt}$ 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

$$y' + f(x)y = g(x) \tag{30}$$

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 (30) 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

$$y' = f(x, y), \quad y(a) = y_0. \tag{31}$$

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 (31) 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

$$z' = f(x, z), \quad z(x_{i-1}) = y_{i-1} \tag{32}$$

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:

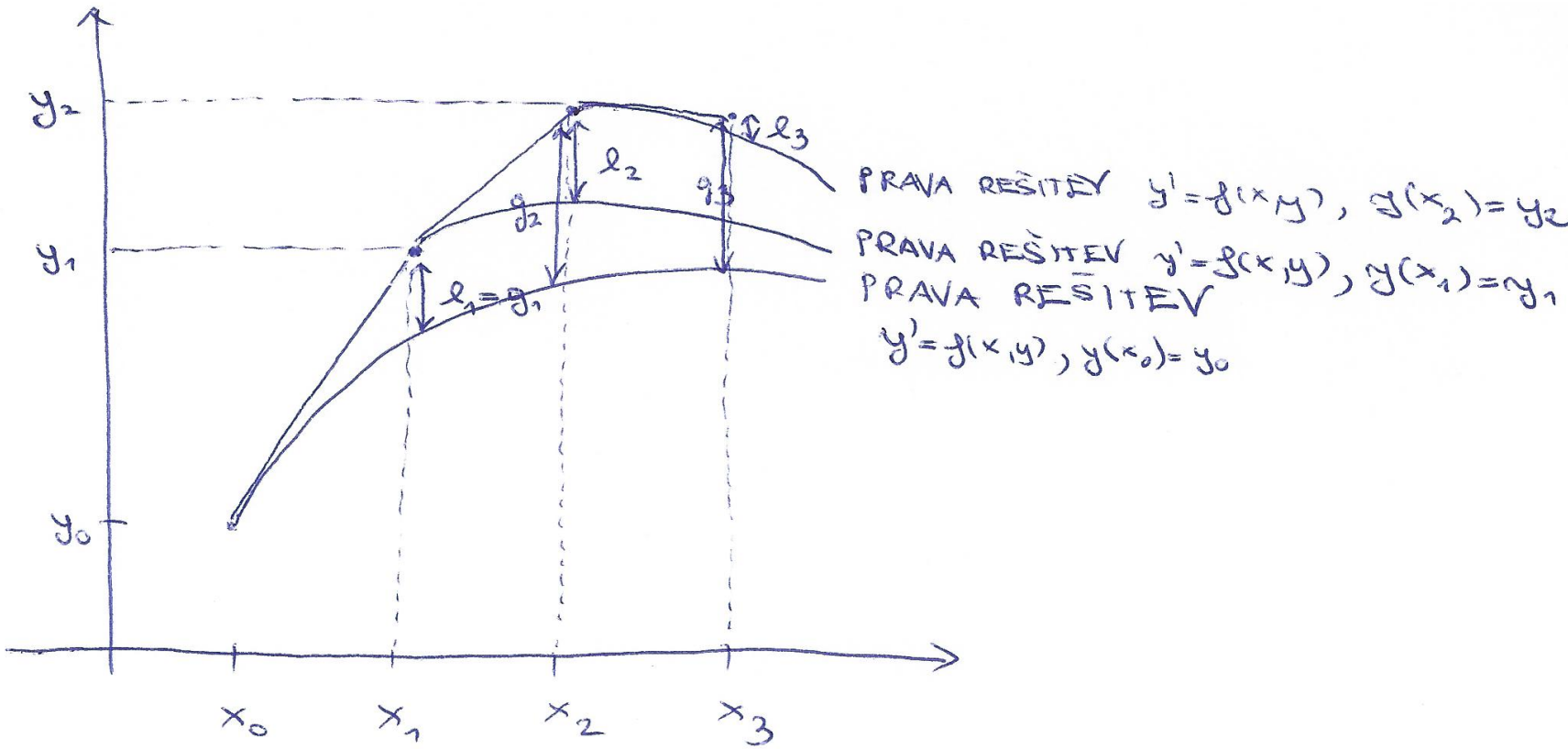
$$|g_i| \leqslant |\ell_1| + |\ell_2| + \ldots + |\ell_i|.$$

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

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:

Algoritem: klik

Metode Runge-Kutta

Ideja teh metod je, da za aproksimacijo odvoda na intervalu $[x_n, x_{n+1}]$ ne upoštevamo odvoda le v točki x_n, temveč neko uteženo povprečje odvodov na $[x_n, x_{n+1}]$.

Primer (Metode Runge-Kutta (RK) reda 2)

Upoštevamo odvoda v točki x_n in $x_n + ch \in [x_n, x_{n+1}]$, kjer je $h = x_{n+1} - x_n$ in $c \in [0, 1]$. Približek y_{n+1} izračunamo tako, da se premaknemo za uteženo povprečje premikov po tangentah v točkah x_n in $x_n + ch$:

$$y_{n+1} = y_n + \underbrace{b_1}_{\text{utež}} \cdot \underbrace{(h \cdot f(x_n, y_n))}_{\text{tangenta v } x_n} + \underbrace{b_2}_{\text{utež}} \cdot \underbrace{(h \cdot f(x_n + ch, y(x_n + ch)))}_{\text{tangenta v } x_n + ch} \quad (33)$$

Upoštevamo

$$y(x_n + ch) \approx y_n + chy'(x_n) = y_n + chf(x_n, y_n) \approx y_n + ahf(x_n, y_n), \quad (34)$$

kjer je a postal prost parameter.

Primer (Metode Runge-Kutta (RK) reda 2)

Upoštevamo (34) *v* (33) *in dobimo*

$$y_{n+1} = y_n + b_1 \cdot \underbrace{(h \cdot f(x_n, y_n))}_{k_1} + b_2 \cdot \underbrace{(h \cdot f(x_n + ch, y_n + a \cdot k_1))}_{k_2}. \quad (35)$$

Z razvojem funkcije $y(x_n + h)$ *in* $f(x_n + ch, y_n + ak_1)$ *v Taylorjevi vrsti in primerjavo koeficientov pri* h *in* h^2 *v* (35) *dobimo pogoja*

$$\begin{aligned} 1 &= b_1 + b_2, \\ \frac{1}{2}(f_x + f_y f)_n &= b_2 c (f_x)_n + b_2 a (f f_y)_n, \end{aligned} \quad (36)$$

kjer f_n, $(f_x)_n$, $(f_y)_n$ *pomenijo* $f(x_n, y_n)$, $f_x(x_n, y_n)$, $f_y(x_n, y_n)$. *Enačbi* (36) *imata veliko rešitev, npr.:*

- $b_1 = b_2 = \frac{1}{2}$ *in* $c = a = 1$. *RK metoda je:*

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2), \\ k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + h, y_n + k_1). \end{aligned}$$

Primer (Metode Runge-Kutta (RK) reda 2)

- $b_1 = 1$, $b_2 = 0$ *in* $c = a = \frac{1}{2}$. *RK metoda je:*

$$\begin{aligned} y_{n+1} &= y_n + k_2, \\ k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1). \end{aligned}$$

Splošna RK metoda je oblike

$$\begin{aligned} y_{n+1} &= y_n + b_1 k_1 + b_2 k_2 + \ldots + b_s k_s, \\ k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + c_2 h, y_n + a_{2,1} k_1), \\ k_3 &= hf(x_n + c_3 h, y_n + a_{3,1} k_1 + a_{3,2} k_2), \\ k_s &= hf(x_n + c_s h, y_n + a_{s,1} k_1 + \ldots + a_{s,s-1} k_{s-1}). \end{aligned} \tag{37}$$

Butcherjeva tabela

RK metode (37) v kompaktni obliki shranjujemo v Butcherjevi tabeli:

$$
\begin{array}{c|ccccccc}
0 & 0 & & & & & \\
c_2 & a_{2,1} & 0 & & & & \\
c_3 & a_{3,1} & a_{3,2} & 0 & & & \\
\vdots & \vdots & & & & & \\
c_s & a_{s_1} & a_{s_2} & a_{s,3} & \cdots & a_{s,s-1} & 0 \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & b_s,
\end{array}
$$

kjer je še

$$
\begin{aligned}
c_2 &= a_{2,1}, \\
c_3 &= a_{3,1} + a_{3,2}, \\
&\vdots \\
c_s &= a_{s,1} + a_{s,2} + \ldots + a_{s,s-1}.
\end{aligned}
$$

Metoda Runge-Kutta reda 4

Butcherjeva tabela:

0	0			
$\frac{1}{2}$	$\frac{1}{2}$	0		
$\frac{1}{2}$	0	$\frac{1}{2}$	0	
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Metoda je

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4,$$

$$k_1 = hf(x_n, y_n), \qquad k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1),$$

$$k_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2), \qquad k_4 = hf(x_n + h, y_n + k_3).$$

Algoritem in primer: klik klik

Ocenjevanje napake in kontrola koraka

1. Pri računanju nas zanima velikost globalne napake.
2. Med izvajanjem metode ocenjujemo velikost lokalnih napak.
3. Na velikost lokalnih napak ključno vpliva izbira dolžine koraka.

Naj bo M metoda reda p, s katero izračunamo $y(x_{n+1})$ z dolžino koraka h. Približek označimo z $y_{n+1,h}$. Velja:

$$\ell_{n+1} := y_{n+1,h} - z(x_{n+1}) \approx Ch^{p+1}, \tag{38}$$

kjer je $z(x)$ rešitev začetnega problema

$$y' = f(x, y), \quad y(x_n) = y_n. \tag{39}$$

Podobno velja:

$$\ell_{n+1} = y_{n+1,h/2} - z(x_{n+1}) \approx C(h/2)^{p+1} + C(h/2)^{p+1} = 2^{-p}Ch^{p+1}, \tag{40}$$

saj smo pri koraku $h/2$ naredili dva koraka metode.

Odštejemo (40) od (38) in dobimo

$$y_{n+1,h} - y_{n+1,h/2} \approx Ch^{p+1}(1 - 2^{-p}). \tag{41}$$

Iz (41) izrazimo Ch^{p+1} in dobimo

$$Ch^{p+1} \approx \frac{y_{n+1,h} - y_{n+1,h/2}}{1 - 2^{-p}}. \tag{42}$$

1. Če je $|\ell_{n+1}| < \epsilon h$, potem $y_{n+1,h}$ sprejmemo.
   V vsaki točki namreč omejimo napako na ϵ. Na celem intervalu integriramo torej napako največ ϵ in dobimo mejo ϵh.

2. Če je $|\ell_{n+1}| \geqslant \epsilon h$, potem ponovimo računanje približka $y(x_{n+1})$ s krajšim korakom.

3. Če je $|\ell_{n+1}|$ bistveno manjši od ϵh, lahko v nadaljevanju uporabimo daljši korak.

Spreminjanje dolžine koraka

Recimo, da rešujemo DE z metodo reda p, Torej je lokalna napaka

$$\ell_n \approx C h_n^{p+1}. \tag{43}$$

Na naslednjem koraku želimo, da je napaka sorazmerna dolžini koraka:

$$C h_{n+1}^{p+1} \approx \epsilon h_{n+1}. \tag{44}$$

Izrazimo C iz (43), vstavimo v (44) in dobimo

$$\frac{h_{n+1}^{p+1}}{h_n^{p+1}} \approx \frac{\epsilon h_{n+1}}{|\ell_n|}.$$

Dolžina naslednjega koraka naj bo zato

$$h_{n+1} = h_n \sqrt[p]{\frac{\epsilon h_n}{|\ell_n|}}.$$

Zaradi zaokroževanja desno stran pomnožimo še s $\sigma \approx 1$, npr. $\sigma = 0.9$.

Metoda vgnezdenih parov za oceno ℓ_n

Naj bosta M_1, M_2 dve metodi Runge-Kutta z istima matrikama koeficientov $a_{i,j}$ (in zato c_i), vendar različnima vektorjema uteži b_i in b_i^*. Naj bo prva metoda reda p, druga pa $p+1$.

Primer

Metodo vgnezdenih parov uporabimo za Butcherjevi tabeli:

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & 1 & 0 \\ & \frac{1}{2} & \frac{1}{2} \end{array}.$$

Prva metoda je Eulerjeva, reda 1*, druga pa reda RK reda 2. Velja*

$$y_{n+1} = y_n + k_1,$$
$$y_{n+1}^* = y_n + \frac{1}{2}(k_1 + k_2).$$

Ocena lokalne napake je

$$\ell_{n+1} \approx y_{n+1}^* - y_{n+1} = (-k_1 + k_2)/2.$$

DOPRI5, Fehlberg, Cash-Karp

Zelo uporabne metode za praktično računanje so metode DOPRI5 (1980, avtorja Dormand in Prince), Fehlberg (1969), Cash-Karp, ki z metodo gnezdenih parov združi dve RK metodi, eno reda 4 in eno reda 5:

https://en.wikipedia.org/wiki/Dormand%E2%80%93Prince_method

https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta%E2%80%93Fehlberg_method

https://en.wikipedia.org/wiki/Cash%E2%80%93Karp_method

Algoritem in primer: klik klik

Linearne večkoračne metode

Za nekatere funkcije (npr. zelo strme ali hitro oscilirajoče funkcije na intervalu $[x_n, x_{n+1}]$) Runge-Kutta metode ne dajo dobrih približkov oz. ne konvergirajo dovolj hitro.

Možna rešitev je uporaba veččlenskih metod, ki poleg vrednosti $y(x_n)$, $f(x_n, y(x_n))$, uporabijo še približke za vrednosti

$$y(x_{n-1}), \ldots, y(x_{n-i})$$

in

$$f(x_{n-1}, y(x_{n-1})), \ldots, f(x_{n-i}, y(x_{n-i})).$$

Izračun $y(x_{n+1})$ temelji na osnovnem izreku integralskega računa:

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))dx. \tag{48}$$

V nadaljevanju naj bo h dolžina koraka, y_i približek za $y(x_i)$, f_i pa oznaka za $f(x_i, y_i)$.

Adams-Bashforthove metode oz. AB metode

Naj bo $p_{k-1}(x)$ interpolacijski polinom stopnje $k-1$ skozi točke

$$(x_n, f_n),\ (x_{n-1}, f_{n-1}),\ \ldots,\ (x_{n-k+1}, f_{n-k+1}).$$

Potem (48) izračunamo kot

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} p_{k-1}(x)dx. \tag{49}$$

Izkaže se, da je

$$y(x_{n+1}) - y(x_n) = h(\beta_1 f_n + \beta_2 f_{n-1} + \ldots + \beta_k f_{n-k+1}), \tag{50}$$

kjer koeficiente β_i preberemo iz naslednje tabele:

k	i	1	2	3	4	C_{k+1}
1	β_i	1				$\frac{1}{2}$
2	$2\beta_i$	3	-1			$\frac{5}{12}$
3	$12\beta_i$	23	-16	5		$\frac{3}{8}$
4	$24\beta_i$	55	-59	37	-9	$\frac{251}{720}$

.

Pri tem je lokalna napaka ocene y_{n+1} enaka

$$C_{k+1} h^{k+1} y^{(k+1)}(\xi_n), \quad \xi_n \in (x_n, x_{n+1}).$$

Vrstico 3 tabele preberemo kot

$$h\Big(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\Big),$$

lokalna napaka ocene pa je

$$\frac{251}{720} h^4 y^{(4)}(\xi_n), \quad \xi_n \in (x_n, x_{n+1}).$$

Algoritem:

klik

Adams-Moultonove metode oz. AM metode

Naj bo $q_k(x)$ interpolacijski polinom stopnje k skozi točke

$$(x_{n+1}, f_{n+1}),\ (x_n, f_n),\ \ldots,\ (x_{n-k+1}, f_{n-k+1}).$$

Potem (48) izračunamo kot

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} q_k(x)dx. \tag{51}$$

Izkaže se, da je

$$y(x_{n+1}) - y(x_n) = h(\beta_0^* f_{n+1} + \beta_1^* f_n + \ldots + \beta_k^* f_{n-k+1}), \tag{52}$$

kjer koeficiente β_i^* preberemo iz naslednje tabele:

k	i	1	2	3	4	C_{k+1}
0	β_i^*	1				$-\frac{1}{2}$
1	$2\beta_i^*$	1	1			$-\frac{1}{12}$
2	$12\beta_i^*$	5	8	-1		$-\frac{1}{24}$
3	$24\beta_i^*$	9	19	-5	1	$-\frac{19}{720}$

.

Pri tem je lokalna napaka ocene y_{n+1} enaka

$$C_{k+1} h^{k+2} y^{(k+1)}(\xi_n), \quad \xi_n \in (x_n, x_{n+1}).$$

Vrstico 3 tabele preberemo kot

$$h\Big(\frac{5}{12} f_{n+1} + \frac{8}{12} f_n - \frac{1}{12} f_{n-1}\Big),$$

lokalna napaka ocene pa je

$$-\frac{1}{24} h^4 y^{(3)}(\xi_n), \quad \xi_n \in (x_n, x_{n+1}).$$

Opazimo, da pri AB metodah y_{n+1} eksplicitno izračunamo, pri AM metodah pa y_{n+1} nastopa na obeh straneh (52). Tako ga moramo izračunati s pomočjo ene od metod za reševanje nelinearnih enačb. V praksi se uporabi navadno interacijo. Če za začetni približek uporabimo AB metodo, nato pa naredimo korak AM metode, dobimo metodo prediktor-korektor.

Algoritem:

klik klik

Sistemi diferencialnih enačb

Sistem DE je oblike:

$$\begin{aligned} y_1' &= f_1(x, y_1, \ldots, y_m), \\ y_2' &= f_2(x, y_1, \ldots, y_m), \\ &\vdots \\ y_m' &= f_m(x, y_1, \ldots, y_m), \end{aligned} \tag{53}$$

kjer so $y_1(x), \ldots, y_m(x)$ neznane funkcije. Imamo še m začetnih pogojev $y_i(x_0) = y_{i,0}$ za $i = 1, \ldots, m$. Sistem (53) lahko zapišemo v vektorski obliki:

$$\vec{y} = \vec{f}(x, \vec{y}), \quad \vec{y}(x_0) = \vec{y}_0, \tag{54}$$

kjer so

$$\begin{aligned} \vec{y} &= (y_1, \ldots, y_m), \quad \vec{f} = (f_1, \ldots, f_m), \\ \vec{y}(x_0) &= (y_1(x_0), \ldots, y_m(x_0)). \end{aligned}$$

Sistem (54) lahko rešujemo z Runge-Kutta metodami, le da vse funkcije podamo kot vektorske funkcije, točke pa kot vektorje.

Algoritmi: klik klik klik

Robni problem - strelska metoda

Robni sistem DE v dveh spremenljivkah je oblike:

$$\begin{aligned} y' &= f(x, y, z), \\ z' &= g(x, y, z), \end{aligned} \tag{55}$$

kjer sta $y(x)$ in $z(x)$ neznani funkciji, $x \in [a, b]$, dana pa sta še pogoja

$$y(a) = y_a \in \mathbb{R}, \qquad z(b) = z_b \in \mathbb{R}.$$

Sistem (55) na intervalu rešujemo s strelsko metodo, tako da ugibamo vrednost $z(a) = \alpha_1$, rešimo začetni problem z eno od numeričnih metod in pogledamo, ali je v rešitvi res $z(b) = z_b$. To skoraj gotovo ne bo izpolnjeno.

Zato uvedemo funkcijo napake

$$F : \mathbb{R} \to \mathbb{R}, \quad \alpha \mapsto z_b - z(b).$$

Radi bi našli α, tako, da je $F(\alpha) = 0$. Iščemo torej ničlo funkcije F. Ker F ni eksplicitno podana, tangentne metode za iskanje ničle F ne moremo uporabiti. Lahko pa uporabimo sekantno metodo, pri čemer sprva izračunamo $F(\alpha_1)$ in $F(\alpha_2)$ za dva začetna približka α_1, α_2.

Algoritem in primer: klik klik