

# Numerične metode

izročki predavanj

Aljaž Zalar

Fakulteta za računalništvo in informatiko  
Univerza v Ljubljani

Verzija 4.11.2022

# Literatura

## Osnovna vira:

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

## Tuji viri:

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

# Obveznosti

Potek predmeta:

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

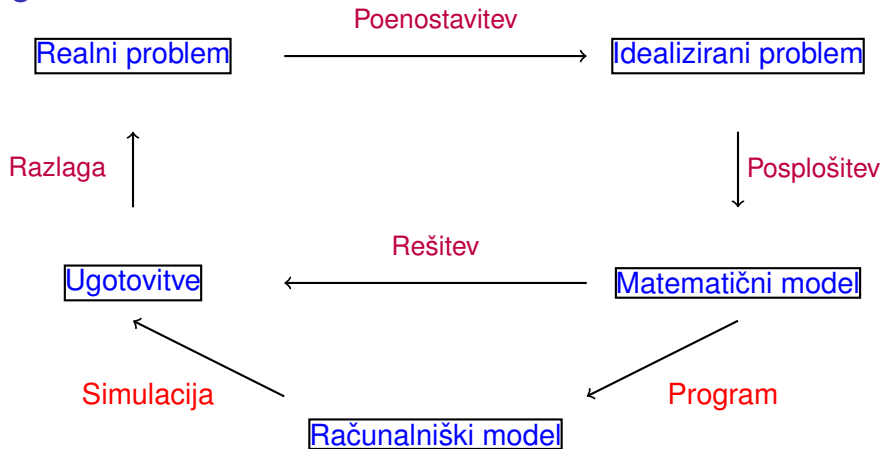
Ocena:

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

Programska oprema:

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

# Vloga numerične matematike



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

# Vsebina predmeta

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

## 5. Numerično odvajanje in integriranje

- ▶ Trapezna metoda
- ▶ Simpsonova metoda
- ▶ Rombergova metoda

## 6. Numerično reševanje diferencialnih enačb

- ▶ Eulerjeva metoda
- ▶ Runge-Kutta metode

Prvo poglavje:

# Uvod v numerično računanje

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

# Numerično in simbolno računanje

## Numerično računanje:

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

## Simbolno računanje:

- ▶ **simboli** predstavljajo števila
- ▶ izraz preoblikujemo s simbolnim računanjem do novega simbolnega izraza - **analitična rešitev**

## Primer

- ▶ *Numerično:*

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

- ▶ *Simbolno:*

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



# Numerično in simbolno računanje

## Primer

```
1 >> x=rand; (x^2-1)/(x+1) - (x-1)
2
3 ans=1.387778780781446e-17
```

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

# Kaj zanima numerično matematiko?

**Metoda** . . . matematična konstrukcija, s katero rešujemo problem

**Algoritem** . . . koraki metode

**Implementacija** . . . zapis algoritma v izbranem jeziku

**Kaj pomeni 'biti numerično dober'?**

majhna sprememba podatkov  $\Rightarrow$  majhna napaka rezultata

**Tipična vprašanja numerične matematike:**

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

# Občutljivih problemov NM ne more rešiti

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

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

## Primer (presečišča premic)

*Sistem in njegova perturbacija*

$$x + y = 2 \quad \rightarrow \quad x + y = 1.9999$$

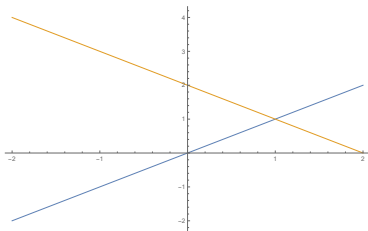
$$x - y = 0 \quad \rightarrow \quad x - y = 0.0002$$

*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

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

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 računat na vsako desetino sekunde v 24-bitnem sistemu. Ker velja

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

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

- ▶ *Eksplozija rakete Ariana 5, Francoska Gvajana, 1996:*  
*posledica prekoračitve obsega števil.*

[https://www.youtube.com/watch?v=PK\\_yguLapgA](https://www.youtube.com/watch?v=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

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

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

- ▶ *Potop naftne ploščadi Sleipner A, Stavanger, Norveška, 1991,* milijarda dolarjev škode: *nenatančna obdelava obremenitev pri reševanju PDE-jev.*

<https://www.youtube.com/watch?v=eGdiPs4THW8>

# Ponovitev predstavljivih števil

Števila shranjujemo v obliki

$$x = \pm 0.d_1d_2d_3 \dots d_m \times \beta^e,$$

kjer je

- ▶  $\beta$  naravno število (v računalništvu  $\beta = 2$ ),
- ▶  $d_1d_2d_3 \dots d_m$  mantisa,  $e$  eksponent.

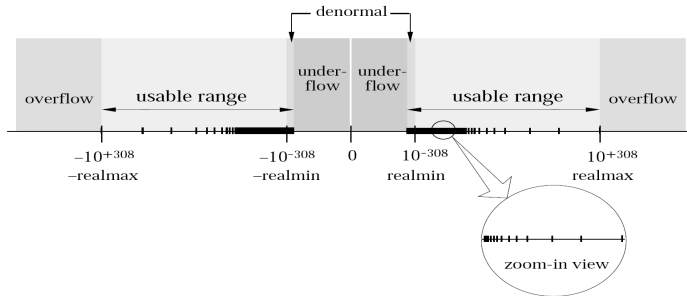
Primer (baza 10)

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



# Prekoračitev in podkoračitev

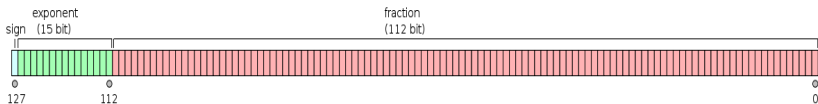
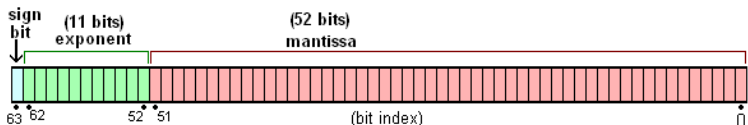
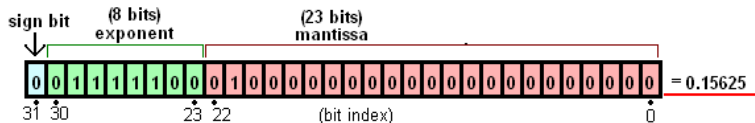
## Floating Point Number Line



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

# Različne natančnosti

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



## Kaj so zaokrožitvene napake?

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

$$x_- \leq x \leq x_+$$

najbližji predstavljivi števili števila  $x$ . Potem je

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

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

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

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

**Absolutna napaka:**

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

**Relativna napaka:**

$$\frac{x - x_-}{x} \leq \frac{2^{e-m-1}}{1/2 \times 2^e} \leq \underbrace{2^{-m}}_u \dots \text{osnovna zaokrožitvena napaka}$$

Torej je

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

Podobno

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

Sledi

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

## Kako računamo s predstavljenimi števili?

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

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

**Seštevanje** numerično **ni asociativna operacija**, tj.

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

### Primer

```
1 >> a=rand;b=rand;c=rand;((a+b)+c)-(a+(b+c))
2
3 ans=-2.220446049250313e-16
```

# Seštevamo od manjših k večjim številom

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

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

Celotna napaka  $D$  je

$$D = D_n + D_m + D_z.$$

# Stabilnost meri kakovost metode

Stabilnost metode preverimo z **analizo zaokrožitvenih napak**.

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

▶ Prva delitev:

▶ **Absolutna napaka**:  $\bar{x} - x$ .

▶ **Relativna napaka**:  $\frac{\bar{x} - x}{x}$ .

▶ Druga delitev:

▶ **Direktna napaka**: Numerična napaka rezultata.

▶ **Obratna napaka**: Koliko je potrebno spremeniti začetne podatke, da dobimo izračunan rezultat.

Velja

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

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

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

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

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

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:

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

```
1  $k_1 := a^2$   
2  $k_2 := k_1 - b$   
3  $k_3 := \sqrt{k_2}$   
4  $k_4 := a + k_3$   
5  $k_5 := \frac{-b}{k_4}$ 
```

```
1 >> a = 10000; b=-1;  
2 >> x = -a+sqrt(a^2 - b)  
3 x = 5.0000000055588316e-05  
4  
5 >> x^2 + 2 * a * x +b  
6 ans = 1.361766321927860e-08  
7  
8 >> x = -b/(a+sqrt(a^2-b))  
9 x = 4.999999987500000e-05  
10  
11 >> x^2 + 2 * a * x +b  
12 ans = -9.011402890989895e-17
```

Koda primera: [klik](#)

# Računanje s stabilnejšo obliko

- ▶ Izračun vrednosti funkcije

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

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

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

Koda primera: [klik](#)

- ▶ Vrsto

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

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

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

Koda primera: [klik](#)

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

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

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

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

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

Koda primera: [klik](#)

## Seštevanje in odštevanje v splošnem nista relativno direktno stabilni operaciji

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

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

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

$$\frac{|\bar{p} - p|}{|p|} \leq \frac{|x(\delta_1 + \delta_3 + \delta_1\delta_3) + y(\delta_2 + \delta_3 + \delta_2\delta_3)|}{|x + y|}.$$

Torej:

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

# Množenje (in deljenje) je relativno direktno stabilna operacija

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

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

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

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

Torej:

Relativna napaka  $\frac{|\bar{p} - p|}{|p|}$  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.
- ▶ :



Drugo poglavje:

# Linearni sistemi

$$Ax = b$$

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

# Direktne metode

$$Ax = b$$

- ▶ Gaussova-eliminacija
- ▶  $LU$  razcep
- ▶ Pivotiranje
- ▶ Pivotna rast

# Reševanje kvadratnih linearnih sistemov

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

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

kjer so  $a_{ij}, b_j$  realna števila.

V matrični obliki ga zapišemo kot

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \dots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_x = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_b.$$

## Geometrijski pomen sistema $Ax = b$

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

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

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

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

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

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

# Sistem $Ax = b$ z vidika numerične matematike

- ▶ Kako **drago** je reševanje sistema  $Ax = b$ ?  
cena=število osnovnih računskih operacij (+, -, ·, :).
- ▶ Kateri **problemi** in **napake** se pojavijo med reševanjem  $Ax = b$ ?  
Ali obstajajo slabe matrike? Kako take matrike identificirati?
- ▶ Za katere matrike se da **enostavno** in **poceni** rešiti tak sistem?

# Ponovitev Gaussove eliminacije (GE)

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

## Primer

Rešujemo  $Ax = b$ , kjer sta

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

Tvorimo *razširjen sistem*

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

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

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

## Primer

Odštejemo 1-kratnik druge vrstice od tretje

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

Rešimo z *obratno substitucijo*

$$x_3 = \frac{2}{-2} = -1,$$

$$x_2 = \frac{1}{-2} (-9 - 5x_3) = 2,$$

$$x_1 = \frac{1}{-3} (-1 - 2x_2 + x_3) = 2.$$

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   $-n \times n$  matrika  $A = [a_{ij}]_{ij}$  in  $n \times 1$  vektor  $b = [b_i]_i$ 
2  -preoblikujemo  $[A|b]$  v zgornjetrikotno z GE
3
4  for  $k = 1 \dots n - 1$ 
5      for  $i = k + 1 \dots n$ 
6           $xmult = a_{ik} / a_{kk}$ 
7           $a_{ik} = 0$ 
8          for  $j = k + 1 \dots n$ 
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 (+, −, ·, :) za prevedbo matrike  $A$  in razširjene matrike  $[A|b]$  v zgornjetrikotno obliko je

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



## Obratna substitucija in število operacij

```
1  -zgornjetrikotna  $n \times n$  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  $i = n - 1 \dots 1$   
6      $s = c_i$   
7     for  $j = i + 1 \dots n$   
8          $s = s - u_{ij}x_j$   
9     end  
10     $x_i = s / u_{ii}$   
11 end
```

### Izrek

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

$$n^2.$$

## Motivacija za zapis GE v matrični obliki

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

Če bomo v nekem postoku reševali sisteme  $Ax = b$  pri **fixni 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 preoblikovati vektor  $b$** , ne da bi delali GE na razširjenem sistemu.

# LU razcep matrike $A$

```
1  -Vhod:  $A = [a_{ij}]_{i,j}$   $n \times n$  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  $k = 1, \dots, n-1$   
7    for  $i = k+1, \dots, n$   
8       $l_{ik} = a_{ik}/a_{kk}$   
9      for  $j = k+1, \dots, n$   
10        $a_{ij} = a_{ij} - l_{ik}a_{kj}$   
11     end  
12   end  
13 end
```

## Izrek

Število računskih operacij (+, -, ·, :) 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  $n \times n$  matrika  $L = [\ell_{ij}]_{i,j}$  in  
   vektor  $b = [b_i]_i$   
2  -Izhod: resitev  $y$  sistema  $Ly = b$   
3  
4   $y_1 = b_1/\ell_{11}$   
5  for  $i = 2 \dots n$   
6      $s = b_i$   
7     for  $j = 1 \dots i - 1$   
8          $s = s - \ell_{ij}y_j$   
9     end  
10     $y_i = s/\ell_{ii}$   
11 end
```

### Izrek

Število računskih operacij (+, −, ·, :) 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 substitucijo, tj. od  $y_1$  proti  $y_n$ .  
Cena:  $n^2 - n$ .
3. Rešimo  $Ux = y$  z obratno substitucijo, t. od  $x_n$  proti  $x_1$ .  
Cena:  $n^2$ .

Cena preme substitucije je za  $n$  operacij manjša kot cena obratne substitucije, saj imamo na diagonalni  $L$  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  $y = (8 \quad 2 \quad -5 \quad -1)^T.$

3. Rešimo  $Ux = y$  in dobimo  $x = (1 \quad -1 \quad 1 \quad -1)^T.$

LU razcep brez pivotiranja: [koda](#)

Prema substitucija: [koda](#)

Obratna substitucija: [koda](#)

Primer: [koda](#)

# Obstoj LU razcepa matrike

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

Problematična sta npr. matriki

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

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

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

## Izrek (Obstoj LU razcepa)

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

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

## LU razcep z delnim pivotiranjem

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

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

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

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

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

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



# LU razcep z delnim pivotiranjem - algoritem

```
1  -Vhod:  $A = [a_{ij}]_{i,j}$   $n \times n$  matrika
2  -Izhod: permutacijska matrika  $P$ , spodnja in
      zgornja trikotna matrika  $L$  in  $U$ , da je
       $PA = LU$ 
3
4   $P$  in  $L$  identicni  $n \times n$  matriki
5  for  $k = 1, \dots, n-1$ 
6      poisci  $q$ -to in  $k$ -to vrstico, ki zadosca
           $|a_{qk}| = \max_{k \leq p \leq n} |a_{pk}|$ 
7       $q$ -to in  $k$ -to vrstico v matrikah  $A, P$  in strogem
          spodnjem trikotniku  $L$ 
8      for  $i = k+1, \dots, n$ 
9           $l_{ik} = a_{ik}/a_{kk}$ 
10         for  $j = k+1, \dots, n$ 
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 (+, −, ·, :) 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  $\mathcal{O}(n^2)$  primerjani in menjavi.

**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 substituicijo. Cena:  $n^2 - n$ .
3. Rešimo  $Ux = y$  z obratno substituicijo. 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.

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

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

2. Rešimo  $Ly = Pb$  in dobimo  $y = (-14 \quad 0 \quad -9 \quad -\frac{1}{8})^T$ .

3. Rešimo  $Ux = y$  in dobimo  $x = (1 \quad -1 \quad 1 \quad -1)^T$ .

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  $\hat{x}$ . Računali smo v treh korakih:

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

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  $\hat{L}$ ,  $\hat{U}$  velja  $A = \hat{L}\hat{U} + E$ , kjer je

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

# Stabilnost LU razcepa matrike A

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

**Izrek** ( Ocena relativne napake pri izračunu LU razcepa )

*Pri LU razcepu z delnim pivotiranjem velja ocena relativne napake:*

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

# Pivotna rast

**Pivotna rast** matrike  $A$  je definirana kot

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

Velja

$$\|\hat{U}\|_{\infty} \leq n\rho(A)\|A\|_{\infty}.$$

## Trditev

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

**Dokaz.** Velja namreč  $|\ell_{ij}| \leq 1$ ,  $a_{ij}$  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 matrice s pivotno rastjo  $2^{n-1}$ , tako da LU razcep z delnim pivotiranjem **teoretično ni stabilen**.

## Primer

*Matrika*

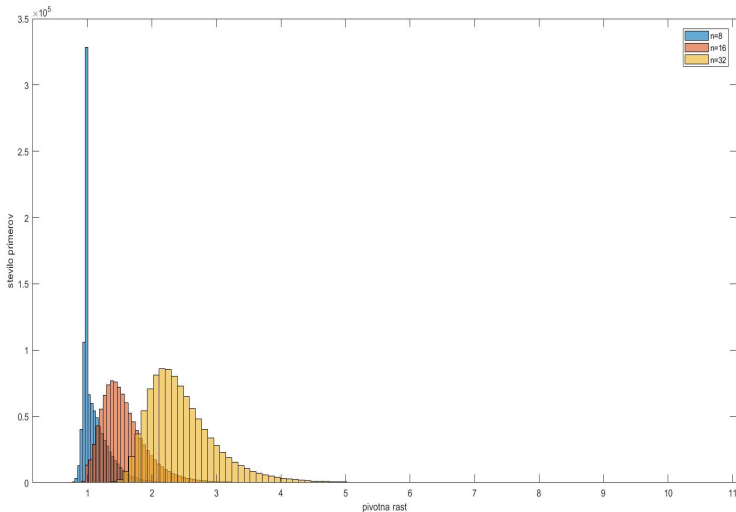
$$A_n = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & 1 & \ddots & \vdots & 1 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ -1 & \cdots & \cdots & -1 & 1 \end{pmatrix}$$

*ima pivotno rast  $2^{n-1}$ .*

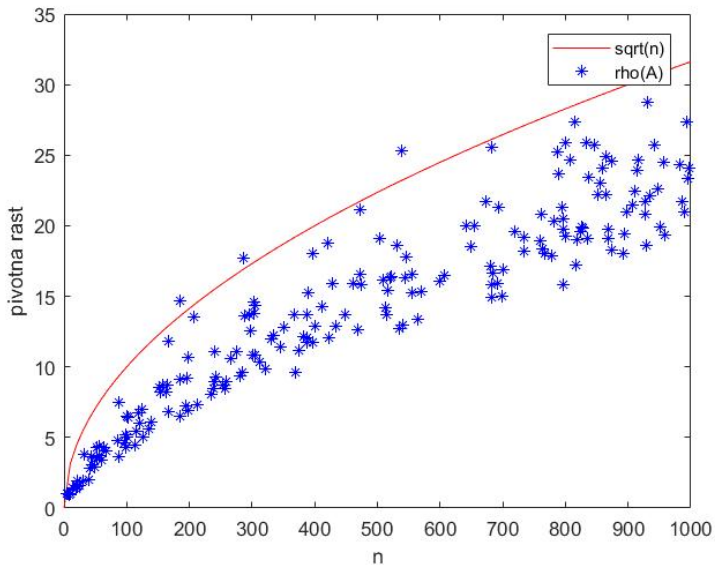
Statistično pa velja, da je pričakovana vrednost pivotne rasti  $\mathcal{O}(n^{2/3})$ , tako da LU razcep z delnim pivotiranjem **v praksi je obratno stabilen**.



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



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



# Iterativne metode

$$Ax = b$$

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

# Iterativne metode za reševanje $Ax = b$

Doslej smo iskali **točno rešitev**  $x^*$  sistema

$$Ax = b. \quad (2)$$

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

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

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

Prednosti iterativnih metod pred direktnimi:

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

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

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

*Kako izboljšati  $x^{(0)}$  ?*

Idealno bi prišteli pravi razliko:

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

kar lahko drugače zapišemo kot

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

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

*Kaj pa, če bi znali aproksimirati  $A^{-1}$ ?*

Recimo, da je približek

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

poceni za izračunati. Potem izračunamo

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

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

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

# Algoritem iterativnih metod

```
1  $A$  je dana  $n \times n$  matrika, ki jo aproksimiramo z  
   matriko  $Q$ , in  $n \times 1$  vektor  $b$ .  
2 Izberi zacetni priblizek  $x = x^{(0)}$ , toleranco  
   dovoljene relativne napake  $tol$  in maksimalno  
   stevilo  $k_{max}$  korakov iteracije.  
3  
4  $x^{(nov)} = \infty$   
5 for  $k = 1$  to  $k_{max}$   
6    $r = b - Ax$   
7   if  $\frac{\|x^{(nov)} - x\|}{\|x\|} \leq tol$ , stop  
8   else  
9      $x = x^{(nov)}$   
10     $x^{(nov)} = x + Q^{-1}r$   
11 end  
12  $x = x^{(nov)}$ 
```

## Jacobijeva iteracija

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

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

Če pišemo

$$A = S + D + Z, \quad (4)$$

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

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



Pišimo

$$x^{(j)} = \left( x_1^{(j)} \quad x_2^{(j)} \quad \dots \quad x_n^{(j)} \right)^T$$

Po komponentah (5) pomeni

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

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

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

[Koda algoritma: klik](#)

[Primer 1: klik](#)

[Primer 2: klik](#)

## Gauss-Seidlova iteracija

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

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

Potem (3) postane

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

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

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

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

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

OZ.

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

Po komponentah (9) pomeni

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

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

[Koda algoritma: klik](#)     [Primer 1: klik](#)     [Primer 2: klik](#)

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

# SOR iteracija

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

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

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

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

Koda algoritma: [klik](#)

Primer 1: [klik](#)

Primer 2: [klik](#)

# Konvergenčni kriteriji

Matrika je **krepro vrstično diagonalno dominantna (kVDD)**, če za vsak  $i$  velja

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

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

## Nekaj konvergenčnih rezultatov:

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

# Reševanje nelinearnih enačb in optimizacija

$$* f(x) = 0$$

$$* f_i(x_1, x_2, \dots, x_n) = 0, \quad i=1, \dots, n$$

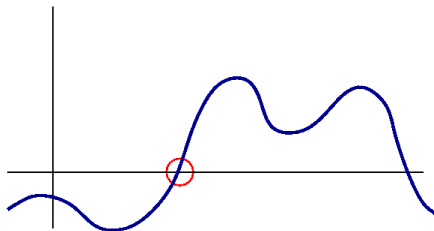
$$* \min\{f(x) : x \in K \subseteq \mathbb{R}^n\}$$

- ▶ Ena enačba v eni spremenljivki: Bisekcija, tangentsna 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

$$f(x) = 0.$$

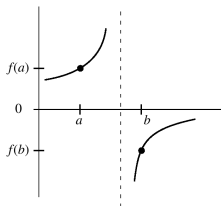
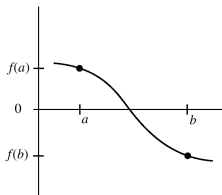


- ▶ **Nelinearni sistemi niso tako enostavno rešljivi** kot sistemi linearnih enačb.
- ▶ Ničel polinoma stopnje 5 ne moremo zapisati analitično.
- ▶ Kako reševati take probleme? Z **iterativnim postopkom**, pri čemer se rešitvam čim bolj približamo.

# 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:

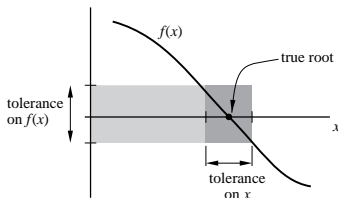
- ▶ Lahko nas zanima, kdaj velja

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

- ▶ Lahko pa nas zanima, kdaj velja

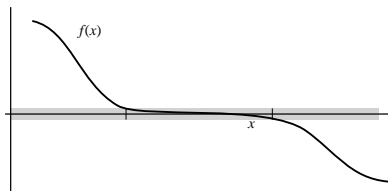
$$|f(x_k)| < \text{toleranca.}$$

- ▶ Še najboljše pa je zahtevati izpolnjenost **obeh pogojev** hkrati.

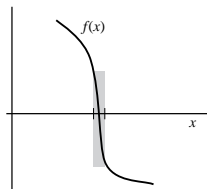


# Primerjava obeh konvergenčnih kriterijev

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.

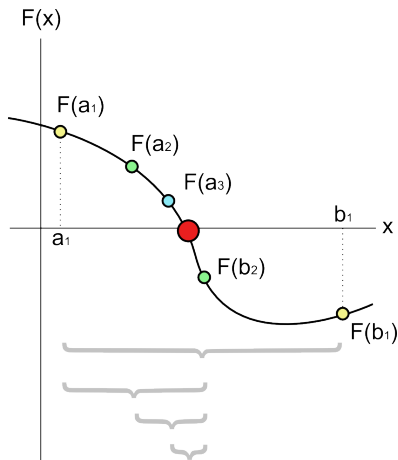
# Bisekcija

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  zacetni podatki:  $f$ ,  $a$ ,  $b$ ,  $tol$ 
2  for  $k = 1, 2, \dots$ 
3       $x_m = a + (b - a)/2$ 
4      if  $\text{sign}(f(x_m)) = \text{sign}(f(a))$ 
5           $a = x_m$ 
6      else
7           $b = x_m$ 
8      end
9      if  $|b - a| < tol$ , stop
10 end
```

[Algoritem: klik](#)

[Primer 1: klik](#)

[Primer 2: klik](#)

## Hitrost konvergence in računska zahtevnost

Naj bo  $\delta_n$  velikost intervala po  $n$ -tem koraku bisekcije. Potem velja

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

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

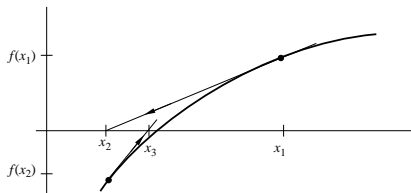
$n$	$\frac{\delta_n}{\delta_0}$	število izračunov funkcijskih vrednosti
5	$3.1 \times 10^{-2}$	7
10	$9.8 \times 10^{-4}$	12
20	$9.5 \times 10^{-7}$	22
30	$9.3 \times 10^{-10}$	32
40	$9.1 \times 10^{-13}$	42
50	$8.9 \times 10^{-16}$	52

# Tangentna metoda

Iteracija:

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

Izpeljava:



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

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

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

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

in izrazimo  $x_{k+1}$ .

```

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

```

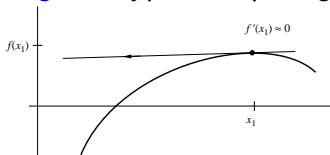
Algoritem: [klik](#)

Primer 1: [klik](#)

Primer 2: [klik](#)

### Lastnosti tangentne metode:

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



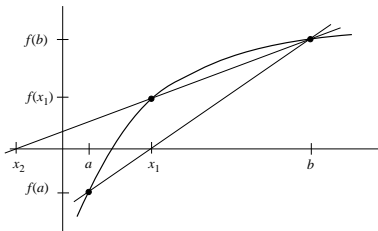


# Sekantna metoda

Iteracija:

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

Izpeljava:



S pomočjo dveh zaporednih približkov  $x_{k-1}$  in  $x_k$ , za nov približek vzamemo  $x$ -koordinato presečišča sekantne 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,  $x_{k-1}$  = 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 (11) in dobimo (12).

```
1  zacetni podatki: f, x1, x2
2  for k = 2, 3...
3      xk+1 = xk - f(xk)(xk - xk-1)/(f(xk) - f(xk-1))
4      if je izpolnjen tolerancni pogoj, stop
5  end
```

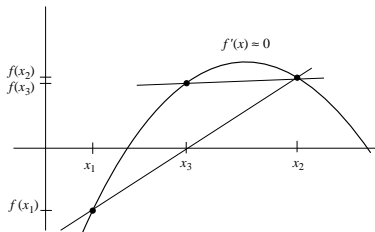
[Algoitem: klik](#)

[Primer 1: klik](#)

[Primer 2: klik](#)

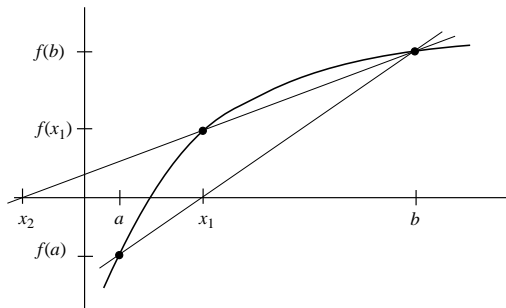
## Lastnosti sekantne metode:

- ▶ Konvergenca je podobna tisti pri tangentni metodi. **Red je  $\approx 1.62$** , 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 sekantne skozi točki  $(a, f(a))$  in  $(b, f(b))$  z abscisno osjo.
- ▶ Za nova približka  $a, b$  vzamemo interval, kjer je funkcija različno predznačena.

```

1  zacetni podatki:  a,b
2  for k = 2,3...
3      c = b - f(b)(b - a)/(f(b) - f(a))
4      if f(a)f(c) < 0
5          b = c
6      else
7          a = c
8      if je izpolnjen tolerancni pogoj, stop
9  end

```

Algoritem: klik

Primer 1: klik

Primer 2: klik

### Lastnosti metode regula falsi:

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

# Metode fiksne točke

Metodo fiksne točke dobimo tako, da enačbo

$$f(x) = 0$$

preoblikujemo v ekvivalentno enačbo

$$g(x) = x.$$

Točki  $x$  pravimo **negibna točka** funkcije  $g$ .

**Algoritem:**

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

Algoritem: klik

Primer 3: klik

Primer 1: klik

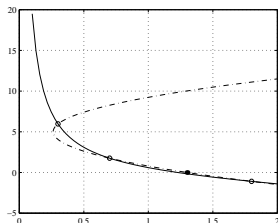
Primer 4: klik

Primer 2: 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:

$$f_1(x_1, \dots, x_n) = 0,$$

$$f_2(x_1, \dots, x_n) = 0,$$

$$\vdots$$

$$f_n(x_1, \dots, x_n) = 0.$$

Če definiramo

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

potem lahko sistem na kratko zapišemo kot

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

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



# Newtonova iteracija

Pri Newtonovi iteraciji tvorimo zaporedje približkov

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

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

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

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

$$\begin{aligned} \underline{J}_f(\underline{x}^{(r)}) \Delta \underline{x}^{(r)} &= -\underline{f}(\underline{x}^{(r)}), \\ \underline{x}^{(r+1)} &= \underline{x}^{(r)} + \Delta \underline{x}^{(r)}. \end{aligned}$$

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  $\underline{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

2. Izberemo začetni približek

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

3. Računamo zaporedje približkov

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

Algoritem: [klik](#)

Primer: [klik](#)

## Variacijske metode

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

**Trditev** ( Pretvorba iskanja ničel funkcije na iskanje globalnih minimumov )

*Ničle funkcije  $f(\underline{x})$  so globalni minimumi funkcije*

$$g : \mathbb{R}^n \rightarrow \mathbb{R}, \quad g(\underline{x}) = \|\mathbf{f}(\underline{x})\|^2 = \sum_{i=1}^n (f_i(\underline{x}))^2.$$

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

## Iskanje lokalnih ekstremov $g$

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

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

kjer je  $\lambda_r$  neko realno število. Veljalo bo:

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

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

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

Po izbiri smeri moramo najti še  $\lambda_r$  v (13): Definiramo

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

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  $\alpha$ , nato pa čez točke  $(0, q(0))$ ,  $(\alpha/2, q(\alpha/2))$ ,  $(\alpha, q(\alpha))$  potegnemo parabolo in za  $\lambda$  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

$$\nabla g(\underline{x}) = \left[ \frac{\partial g(\underline{x})}{\partial x_1} \quad \dots \quad \frac{\partial g(\underline{x})}{\partial x_n} \right] = 0.$$

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

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

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

Algoritem in primeri: klik