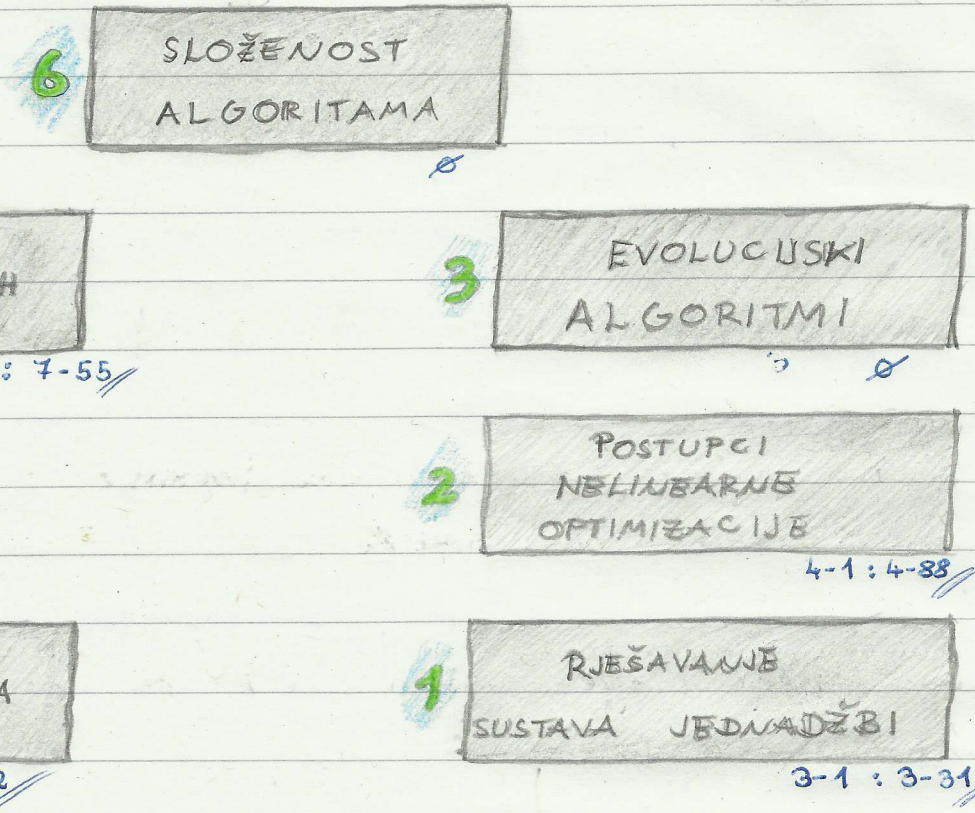
ANALIZA I PROJEKTIRANJE RAČUNALOM



RJEŠAVANJE SUSTAVA LINEARNIH ALGEBARSKIH JEDNADŽBI

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1m}x_m = b_1$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nm}x_m = b_m$$

podvlaka označava vektor / matricu

⇒ kraći zapis (matrični): $\underline{A} \cdot \underline{x} = \underline{b}$

$\underline{A}$ ⇒ matrica sustava $(n \times m)$

$\underline{b}$ ⇒ slobodni vektor

⇒ matrično rješavanje sustava (to je onaka kako smo radili na matematici):

$$\underline{A} \cdot \underline{x} = \underline{b} \quad / \cdot \underline{A}^{-1}$$
$$\underline{E} \cdot \underline{x} = \underline{A}^{-1} \cdot \underline{b}$$
$$\underline{x} = \underline{A}^{-1} \cdot \underline{b}$$

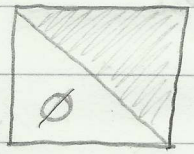
⇒ ako je matrica $\underline{A}$ nesingularna (ima inverznu matricu) tada postoji jednoznačno rješenje sustava za bilo koji vektor $\underline{b}$

↳ u slučaju da ne postoji inverz matrice sustava, broj rješenja ovisi o vektoru $\underline{b}$

⇒ OSTALE METODE RJEŠAVANJA:

a) Gaussova eliminacija:

• matricu sustava svodimo na gornju trokutastu ; onda se rješavamo varijabl prema gore supstitucijom



$\cdot \underline{x} = \underline{b}'$

b) LU dekomporucija (BITNO!)

• rastavljamo matricu sustava na:

$$\underline{A} = \underline{L} \cdot \underline{U}$$

• matrica $\underline{L}$:

$$\underline{L} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \emptyset \\ l_{31} & l_{32} & 1 & \\ \vdots & \vdots & \vdots & \ddots \\ l_{n1} & l_{n2} & l_{n3} & \cdots 1 \end{bmatrix}$$

(donja trokutasta s jedinicnom dijagonalom)

• matrica $\underline{U}$:

$$\underline{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ \emptyset & & & u_{nn} \end{bmatrix}$$

(gornja trokutasta matrica)

↳ RJEŠAVANJE:

$$\underline{A} \cdot \underline{x} = \underline{L} \cdot (\underline{U} \cdot \underline{x}) = \underline{b} = \underline{L} \cdot \underline{y} = \underline{b}$$

$\underline{U} \cdot \underline{x} \rightarrow \underline{y}$

↳ **supstitucija unaprijed:**

• metoda rješavanja (1. dio)

$$\underline{L} \cdot \underline{y} = \underline{b} \quad \Rightarrow \quad 1 \cdot y_1 = b_1$$

$$l_{21} \cdot y_1 + 1 \cdot y_2 = b_2 \quad \Rightarrow \quad y_2 = b_2 - l_{21} \cdot y_1$$

$$\vdots$$

OPĆENITO: $$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} \cdot y_j \quad ; \quad i = 1, \dots, n$$

↳ **supstitucija unatrag:**

• metoda rješavanja (2. dio)

$$\underline{U} \cdot \underline{x} = \underline{y} \quad \Rightarrow \quad u_{nn} \cdot x_n = y_n \quad \Rightarrow \quad x_n = \frac{1}{u_{nn}} y_n$$

OPĆENITO: $$x_i = \frac{1}{x_{ii}} \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right)$$

⇒ sada znamo postupak rješavanja, pa samo još treba pokazat kako rastavit matricu $\underline{A}$ na matrice $\underline{L}$ i $\underline{U}$ potrebne za rješavanje

⇒ pretpostavimo da znamo rastaviti matricu A_{n-1} na $\underline{L}'$ i $\underline{U}'$, pa možemo probat napisat A_n kao umnožak $\underline{L}$ i $\underline{U}$:

$$A = \begin{bmatrix} a_{11} & \underline{w}^T \\ \underline{v} & \underline{A}' \end{bmatrix} = \begin{bmatrix} 1 & \underline{0}_{n-1}^T \\ \frac{1}{a_{11}}\underline{v} & \underline{E}_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & \underline{w}^T \\ \underline{0}_{n-1} & \underline{A}' - \underline{v}\underline{w}^T \cdot \frac{1}{a_{nn}} \end{bmatrix}$$

SHUROV KOMPLEMENT

⋮

⇒ no, ovo je komplicirano te nećemo postoj jednostavniji algoritam:

↳ KORISTI ISTI MEMORIJSKI PROSTOR jer možemo sve smjestit tamo gdje je bila matrica A (ne treba pamtit dijagonale i one nule jer su one trivijalni element)

↳ dodatno, prostor za vektor $\underline{b}$ može se koristit za vektore $\underline{y}$ i $\underline{x}$ (to mi nećemo raditi, al treba znat za tu mogućnost)

ALGORITAM LU DEKOMPOZICIJE:

```
ZA i = 1 DO n-1
    ZA j = i+1 DO n
        A[j, i] /= A[i, i]
        ZA k = i+1 DO n
            A[j, k] -= A[j, i] * A[i, k]
```

PR: LU dekomponiraj sljedeću matricu!

→ ne mijenja se!

→ „pilot element" (onaj s kojim dijelimo stupac ispod)

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 4 & 6 & 1 & -1 \\ -8 & 3 & -5 & -6 \\ 12 & 12 & 7 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & & \\ -2 & & & \\ 3 & & & \end{bmatrix}$$

a) $6 - 1 \cdot 3 = 3$

↳ „ono što je bilo prije minus umnožak onog u istom stupcu na vrhu i onog u istom retku krajnje lijevo"

↳ nakon jedne iteracije algoritma dobijemo:

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & -1 & -2 \\ -2 & 9 & -1 & -4 \\ 3 & 3 & 1 & 1 \end{bmatrix}$$

obrnuti!

→ ne mijenja se (pomaknulo se dolje i u desno)

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & -1 & -2 \\ -2 & 9 & -1 & -4 \\ 3 & 3 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & -1 & -2 \\ -2 & 3 & b)/2 & 2 \\ 3 & 1 & & \end{bmatrix}$$

b) $-1 - (1)(3) = 2$

↳ nakon druge iteracije:

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & -1 & -2 \\ -2 & 3 & 2 & 2 \\ 3 & 1 & 2 & 3 \end{bmatrix}$$

↳ nakon zadnje (treće) iteracije:

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 3 & -1 & -2 \\ -2 & 3 & 2 & 2 \\ 3 & 1 & 1 & 1 \end{bmatrix} = \underline{L}\,\underline{U}$$

ALGORITAM SUPSTITUCIJE UNAPRIJED (uta memorija):

```
ZA i = 1 DO n-1
    ZA j = i+1 DO n
        b[j] -= A[j, i] * b[i]
```

ALGORITAM SUPSTITUCIJE UNATRAG:

```
ZA i = n DO 1
    b[i] /= A[i, i]
    ZA j = 1 DO i-1
        b[j] -= A[j, i] * b[i]
```

ZAD: Riješi zadani sustav:

$$\begin{bmatrix} 2 & 2 & 3 \\ 2 & 3 & 0 \\ 0 & 4 & 2 \end{bmatrix} \cdot \underline{X} = \begin{bmatrix} 25 \\ 10 \\ 10 \end{bmatrix}$$

↳ provodimo LU dekomporciju:

$$\begin{bmatrix} 2 & 2 & 3 \\ 1 & 1 & -3 \\ 0 & 4 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & 2 & 3 \\ 1 & 1 & -3 \\ 0 & 4 & 14 \end{bmatrix} = \underline{L} \setminus \underline{U}$$

↳ provodimo supstituciju unaprijed:

$$\underline{L} \cdot \underline{y} = \underline{b} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 25 \\ 10 \\ 10 \end{bmatrix}$$

• odavde je trivijalno izračunati:

$$y_1 = 25$$

$$y_2 = -15$$

$$y_3 = 70$$

↳ na kraju treba provesti supstituciju unatrag:

$$\underline{U} \cdot \underline{x} = \underline{y} \Rightarrow \begin{bmatrix} 2 & 2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 14 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 25 \\ -15 \\ 70 \end{bmatrix}$$

• odavde je tvrdnja vidjeti:

$$x_3 = 5$$

$$x_2 = 0$$

$$x_1 = 5$$

$$\underline{x} = \begin{bmatrix} 5 \\ 0 \\ 5 \end{bmatrix}$$

⇒ NAPOMENA: kraj je rješenje će uvijek biti cijeli brojevi

ZAD: Riješi sljedeći sustav!

$$\begin{bmatrix} 6 & 2 & 10 \\ 2 & 3 & 0 \\ 0 & 4 & 2 \end{bmatrix} \underline{x} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

↳ dekompozicija na L i U:

$$\left[\begin{array}{ccc} 6 & 2 & 10 \\ \hline 2 & 3 & 0 \\ 0 & 4 & 2 \end{array}\right] \Rightarrow \begin{bmatrix} 6 & 2 & 10 \\ \frac{1}{3} & \frac{7}{3} & -\frac{10}{3} \\ 0 & 4 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 6 & 2 & 10 \\ \frac{1}{3} & \frac{7}{3} & -\frac{10}{3} \\ 0 & \frac{12}{7} & \frac{54}{7} \end{bmatrix}$$

↳ povratni sustav $\underline{L} \cdot \underline{y} = \underline{b}$:

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ 0 & \frac{12}{7} & 1 \end{bmatrix} \cdot \underline{y} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \underline{y} = \begin{bmatrix} 2 \\ \frac{7}{3} \\ 0 \end{bmatrix}$$

↳ radni sustav $\underline{U} \cdot \underline{x} = \underline{y}$:

$$\begin{bmatrix} 6 & 2 & 10 \\ 0 & \frac{7}{3} & -\frac{10}{3} \\ 0 & 0 & \frac{54}{7} \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 2 \\ \frac{7}{3} \\ 0 \end{bmatrix} \Rightarrow \underline{x} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

PR: Riješi zadan sustav LU dekompozicijom

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 3 \\ 1 & 3 & 3 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 1/2 \\ 2 \\ 1 \end{bmatrix}$$

↳ dekompozicija na L i U:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 3 \\ 1 & 3 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 2 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & & \end{bmatrix}$$

stožerni element je nula, pa ovo ne možemo riješiti LU dekompozicijom

PRS Riješi zadan sustav LU dekompozicijom!

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \underline{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow$$ odmah vidimo:

$$\underline{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

↳ realni problem rješenja može umjesto nule dati:

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

↳ radimo LU dekompoziciju:

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 10^{-20} & 1 \\ 10^{20} & 1-10^{-20} \end{bmatrix}$$

↳ pomoćni sustav $L \cdot \underline{y} = \underline{b}$:

$$\begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \cdot \underline{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \underline{y} = \begin{bmatrix} 1 \\ -10^{20} \end{bmatrix}$$

↳ sustav na kraju $U \cdot \underline{x} = \underline{y}$:

$$\begin{bmatrix} 10^{-20} & 1 \\ 0 & 1-10^{20} \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 1 \\ -10^{20} \end{bmatrix} \Rightarrow \underline{x} = \begin{bmatrix} -\frac{10^{20}}{10^{20}-1} \\ \frac{10^{20}}{10^{20}-1} \end{bmatrix} \approx \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

↓

ovo je dobro na papiru, ali u računalu to neće biti tako!

↳ ako koristimo **double** (preciznost na 16 znamenki), stvari će izgledati ovako:

$$\underline{U}' \cdot \underline{x} = \underline{y} \quad \Rightarrow \quad \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix} \underline{x} = \begin{bmatrix} 1 \\ -10^{20} \end{bmatrix}$$

$$\Downarrow$$

$$x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

↓

skroz krivo ispadne, pa to popravljamo PIVOTIRANJEM ▽

PIVOTIRANJE ILI STOŽERNI RAZVOJ

⇒ element na glavnoj dijagonali matrice sustava tijekom dekompozicije nazivaju se STOŽERNI ili PIVOT ELEMENTI

⇒ može se dogoditi da:

a) pivot element postane jednak nuli

b) pivot element je jako mal što dovodi do velikih pogrešaka

⇒ PIVOTIRANJE: zamjena redaka i/il stupaca matrice kako bi za pivot dobili što veću apsolutnu vrijednost

⇒ <u>varijante pivotiranja:</u>

a) djelomično pivotiranje:

- po retcima ILI po stupcima
- mi ćemo raditi po stupcima:
  - ↳ zamjenjujemo retke tako da uvijek dobijemo maksimalni pivot
  - ↳ zamjena „k"-tog i „r"-tog retka:

$$|a_{rk}| = \max |a_{jk}| \; ; \; j = k, \ldots, n$$

b) potpuno pivotiranje:

- po retcima i po stupcima

LUP DEKOMPOZICIJA

⇒ LU dekompozicija uz pivotiranje po stupcima (radi zamjenu redaka)

↳ pokazuje se da je ovo dovoljno ukoliko sustav ima rješenja

⇒ NAPOMENA: S obzirom da je zamjena redaka matrice sustava zapravo zamjena redoslijeda jednadžbi, treba promijeniti i redoslijed redaka u vektoru $\underline{b}$

⇒ ALGORITAM LUP:

$\underline{P} = \underline{E}$; // permutacijska matrica za evidenciju zamjena

ZA $i = 1$ DO $m-1$

ODABERI PIVOT ELEMENT: $a_{R,i} = \max |a_{j,i}|$, $j = i, \dots, m$;

AKO ($A[R, i] == 0$)

KRAJ;

ZAMIJENI (R-ti, i-ti REDAK MATRICE A) ;

ZAMIJENI (R-ti, i-ti REDAK MATRICE P) ;

ZA $j = i+1$ DO m

$A[j, i] \mathrel{/}= A[i, i]$;

ZA $k = i+1$ DO m

$A[j, k] \mathrel{-}= A[j, i] * A[i, k]$;

⇒ NAPOMENE:

a) rješenje ra provjeru jednakosti double varijabli:

$$A[R, i] == \emptyset \Rightarrow |A[R, i]| < \varepsilon \quad (\varepsilon \text{ je mali})$$

b) svaka kvadratna matrica može se svest na L i U ur barem jednu permutaciju redaka il stupaca (no, to i dalje ne znači da sustav ima rješenje - matrica može još uvijek biti singularna)

⇒ RASTAV MATRICE SUSTAVA:

$$\underline{P} \cdot \underline{A} = \underline{L} \cdot \underline{U}$$

⇒ supstitucija unaprijed: $\underline{L} \cdot \underline{y} = \underline{P} \cdot \underline{b}$

⇒ supstitucija unatrag: $\underline{U} \cdot \underline{x} = \underline{y}$

Pr: Matricu rastavi LUP dekompozicijom!

$$\begin{bmatrix} 4 & 10 & 5 & -3 \\ -4 & 6 & 6 & 3 \\ \textcircled{-8} & 4 & 2 & 6 \\ -4 & 8 & 1 & -3 \end{bmatrix} \Rightarrow \begin{bmatrix} -8 & 4 & 2 & 6 \\ -4 & 6 & 6 & 3 \\ 4 & 10 & 5 & -9 \\ -4 & 8 & 1 & -3 \end{bmatrix} \Rightarrow$$

→ sad se ne mijenja!!

$$\Rightarrow \begin{bmatrix} -8 & 4 & 2 & 6 \\ \frac{1}{2} & \textcircled{4} & 5 & 0 \\ -\frac{1}{2} & 12 & 6 & -6 \\ \frac{1}{2} & 6 & 0 & -6 \end{bmatrix} \Rightarrow \begin{bmatrix} -8 & 4 & 2 & 6 \\ -\frac{1}{2} & 12 & 6 & -6 \\ \frac{1}{2} & 4 & 5 & 0 \\ \frac{1}{2} & 6 & 0 & -6 \end{bmatrix} \Rightarrow \begin{bmatrix} -8 & 4 & 2 & 6 \\ -\frac{1}{2} & 12 & 6 & -6 \\ \frac{1}{2} & \frac{1}{3} & \textcircled{3} & 2 \\ \frac{1}{2} & \frac{1}{2} & -3 & -3 \end{bmatrix} \Rightarrow$$

↓ kroj prve iteracije, pa tražimo najveći u redu ovima

↓ kroj druge iteracije pa tražimo opet najvećeg

$$\Rightarrow \begin{bmatrix} -8 & 4 & 2 & 6 \\ -\frac{1}{2} & 12 & 6 & -6 \\ \frac{1}{2} & \frac{1}{3} & 3 & 2 \\ \frac{1}{2} & \frac{1}{2} & -1 & -1 \end{bmatrix}$$

↳ izgled permutacijske matrice:

$$\underline{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ZAD: Riješi zadan sustav LUP dekompozicijom!

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 2 \\ 4 & 3 & 4 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 4 \\ 3 \\ 8 \end{bmatrix}$$

↳ dekompozicija LUP metodom:

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 2 \\ 4 & 3 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 3 & 4 \\ 1 & 2 & 2 \\ 3 & 2 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 3 & 4 \\ \frac{1}{4} & \frac{5}{4} & 1 \\ \frac{3}{4} & -\frac{1}{4} & -2 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 3 & 4 \\ \frac{1}{4} & \frac{5}{4} & 1 \\ \frac{3}{4} & -\frac{1}{5} & -\frac{9}{5} \end{bmatrix} = \underline{L} \setminus \underline{U}$$

↳ permutacijska matrica:

$$\underline{P} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

↳ supstitucija unaprijed:

$$\underline{L} \cdot \underline{y} = \underline{P} \cdot \underline{b}$$

↗ ispermutiran vektor $\underline{b}$ pomoću $\underline{P}$

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{3}{4} & -\frac{1}{5} & 1 \end{bmatrix} \cdot \underline{y} = \begin{bmatrix} 8 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \underline{y} = \begin{bmatrix} 8 \\ 1 \\ -\frac{9}{5} \end{bmatrix}$$

↳ supstitucija unatrag:

↳ dobijeno rješenje:

$$\underline{x} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

ZAD: Riješi zadani sustav LUP dekompozicijom!

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

↳ rastav i permutacijska matrica:

$$\ldots\ldots \Rightarrow \begin{bmatrix} 1 & 1 \\ 10^{-20} & 1-10^{-20} \end{bmatrix} \qquad \underline{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

↳ supstitucija unaprijed:

$$\begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix} \cdot \underline{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow\Rightarrow \underline{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

↳ supstitucija unatrag:

$$\ldots\ldots \Rightarrow\Rightarrow \underline{x} = \begin{bmatrix} -\frac{1}{1-10^{-20}} \\ \frac{1}{1-10^{-20}} \end{bmatrix} \approx \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

↳ ono što se desi na računalu:

$$\underline{U}' \cdot \underline{x} = \underline{Y} \Rightarrow \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \underline{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow\Rightarrow \underline{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

dobro rješenje, pa možemo zaključiti da je LUP puno robusnija

INVERZIJA MATRICE

⇒ ako baš trebamo inverz matrice, a imamo dekomporicju, onda uz pomoć dekomporicje možemo brzo dobiti inverz

⇒ skica problema: $\underline{A} \cdot \underline{x} = \underline{b}$

$$\underline{A} \cdot \underline{X} = \underline{E}$$

$$\underline{A} \cdot [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n] = [\underline{e}_1, \underline{e}_2, \ldots, \underline{e}_n]$$

↓

dobijemo „n" jednadžbi:

$$\underline{A} \cdot \underline{x}_1 = \underline{e}_1$$
$$\underline{A} \cdot \underline{x}_2 = \underline{e}_2$$
$$\vdots$$
$$\underline{A} \cdot \underline{x}_n = \underline{e}_n$$

⇒ INVERZIJA: „n" puta LUP: 1 x LUP dekomporicija

$n \times$ { supstitucija unaprijed; supstitucija unatrag }

↳ u obliku jednadžbi:

$$\underline{L} \cdot \underline{U} = \underline{P} \cdot \underline{A}$$

$$\underline{L} \cdot \underline{U} \cdot \underline{X} = \underline{P} \cdot \underline{E} = \underline{P}$$

↓

A^{-1}

⇒ na supstituciju unaprijed služi se stupci permutirane jedinične matrice

↳ jednom računamo: $\underline{P} \cdot \underline{A} = \underline{L} \cdot \underline{U}$

↳ „m" puta računamo:

$$\begin{cases} \underline{L} \cdot \underline{y}_i = \underline{P} \cdot \underline{e}_i \\ \underline{U} \cdot \underline{x}_i = \underline{y}_i \end{cases}$$

⇒ NAPOMENA:

↳ rješavanje sustava $\underline{A} \cdot \underline{x} = \underline{b}$ u MatLab-u radi se lijevim dijeljenjem:

$$\underline{x} = \underline{A} \backslash \underline{b}$$

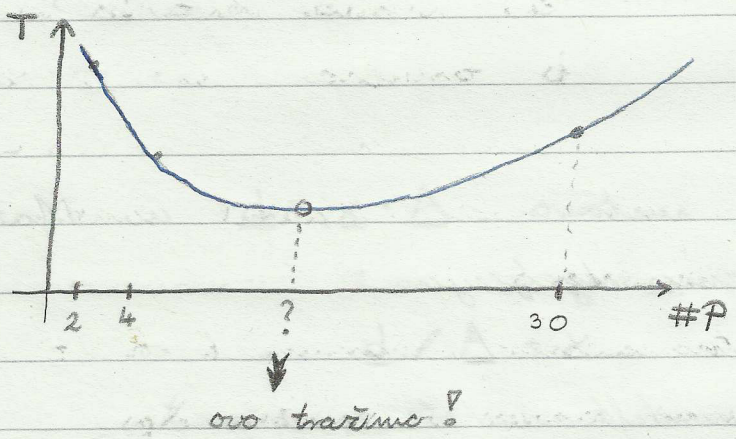
(„\" je left-divide operator)

↳ MatLab ispod haube koristi upravo LUP dekompoziciju za rješavanje ovog sustava

↳ MatLab koristi „double" po default-u, pa rješenje iz MatLab-a mora odgovarati našoj implementaciji

POSTUPCI NELINEARNE OPTIMIZACIJE

⇒ na primjer, trajanje u ovisnosti na broj procesora na kojima se program izvršava:



⇒ problem je u što manje pokušaja doći do što boljeg rješenja

⇒ ovo su problemi koji se ne mogu riješit analitički

⇒ OPIS PROBLEMA OPTIMIRANJA:

- **varijable** (parametri):
  - ↳ označavamo sa: $\vec{X}$ (il' ovako $\underline{X}$)
- **funkcija cilja**:
  - ↳ označavamo sa: $F(\vec{X})$
  - ↳ mi ćemo se bavit jednokriterijskom optimizacijom što znači da funkcija cilja vraća jedan skalar

↳ mi možemo imati više kriterija, al ih stavljamo u jedan skalar koj funkcija cilja vraća

↳ konstrukcija funkcije cilja za više parcijalnih funkcija:

- postoj puno načina:
  a) suma kvadrata
  b) težinska suma kvadrata
  c) maksimalna apsolutna vrijednost neke od parcijalnih funkcija cilja

- **optimum funkcije:**

↳ pretpostavimo minimum funkcije u točki $\underline{X_m}$

↳ radimo razvoj $\underline{X}$ oko $\underline{X_m}$

↳ što znači da trebamo gradijent:

∇F (1. derivacija)

↳ trebamo i drugu derivaciju što je matrica koja se zove HESSEOVA MATRICA:

$\underline{H}$

(matrica drugih parcijalnih derivacija)

↳ izraz funkcije cilja oko $\underline{X_m}$ razvijen u Taylorov red:

$$F(\underline{X}) = F(\underline{X_m}) + \underbrace{\nabla F^T \cdot \Delta \underline{X} + \frac{1}{2} \Delta \underline{X}^T \cdot \underline{H} \cdot \Delta \underline{X}}$$

ovo treba biti pozitivno ako je $\underline{X_m}$ minimum

↳ da bi: $\nabla F^T \cdot \Delta X + \frac{1}{2} \Delta X^T \cdot H \cdot \Delta X$ bilo veće od nule, mora biti:

$\Delta F^T \cdot \Delta X \geq 0 \Rightarrow \nabla F = \vec{0}$ (jer je ΔX varijabilan u + i − smjeru)

$\frac{1}{2} \Delta X^T \cdot H \cdot \Delta X \geq 0$

⇒ **UVJETI DA JE X_m MINIMUM:**

1) ∇F = je nul-vektor
2) H je pozitivno definitna

⇒ svojstvo promatrane točke ovisi o matrici H:

a) H je pozitivno definitna ⇒ minimum
b) H je negativno definitna ⇒ maksimum
c) H je indefinitna ⇒ sedlasta točka

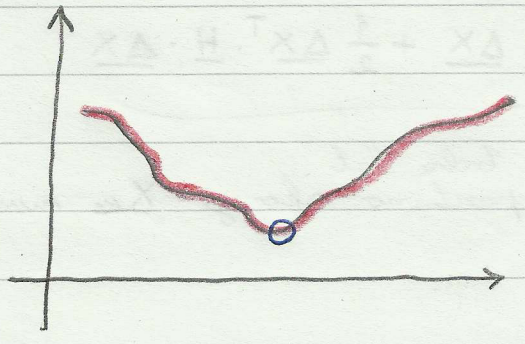
⇒ vrste funkcija:

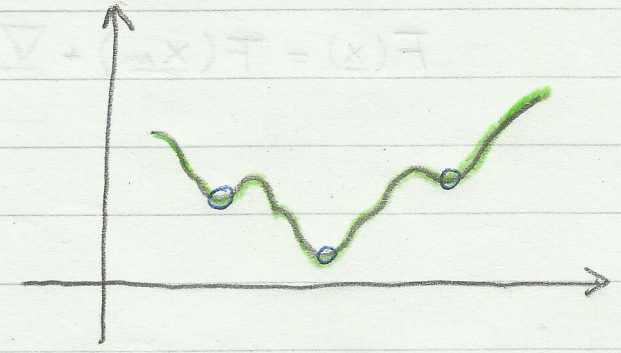
UNIMODALNE

• imaju samo jedan ekstrem na cijelom području definicije

VIŠEMODALNE

• više lokalnih optimuma od kojih je barem jedan globalni





⇒ NAPOMENA: BITNO!

↳ za višemodalne funkcije nema jamstva pronalaska globalnog optimuma

⇒ pretpostavka za ovo poglavlje:

↳ mi promatramo funkcije (il područje) koje su unimodalne

↳ pitanje je kako što brže pronaći traženi ekstrem unimodalne funkcije

PRONALAŽENJE OPTIMUMA FUNKCIJE JEDNE VARIJABLE

⇒ rješenje je interval, a ne neki specificirani X!

↳ oblika: $X \in [\_\_, \_\_]$

↳ ni samo možemo zahtjevati jako uzak interval

⇒ **ZADANO:**

- početna točka (neko rješenje)
- početni interval

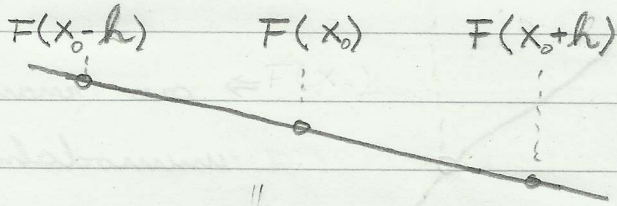
↳ često ovo ipak nemamo, pa prvo trebamo odrediti u kojem se intervalu uopće nalazi ekstrem dane unimodalne funkcije

⇒ PRONALAŽENJE UNIMODALNOG INTERVALA FUNKCIJE JEDNE VARIJABLE:

↳ zadano:

- početna točka: „x_0"
- parametar: korak „h"

↳ pogledamo izgled funkcije oko x_0:



PADAJUĆA NA PROMATRANOM INTERVALU

$\Rightarrow F(x_0 + 2^i \cdot h)$

sve do:

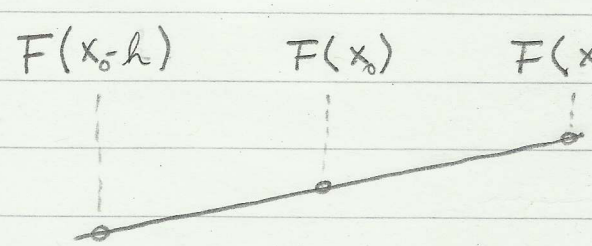
$F(x_0+2^k \cdot h) > F(x_0+2^{k-1} \cdot h)$

↓

tada znamo:

- interval:

$$\left[x_0 + 2^{k-2} \cdot h,\ x + 2^k \cdot h\right]$$



RASTUĆA NA PROMATRANOM INTERVALU

$\Rightarrow F(x_0 - 2^i \cdot h)$

sve do:

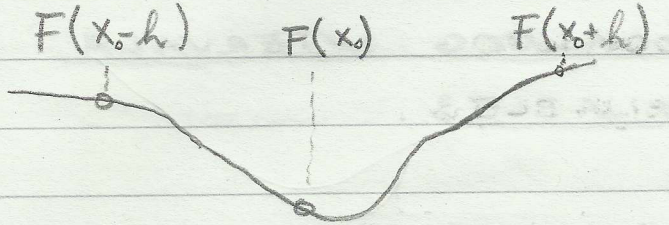
$F(x_0-2^k \cdot h) > F(x_0 - 2^{k-1} \cdot h)$

↓

tada znamo:

- interval:

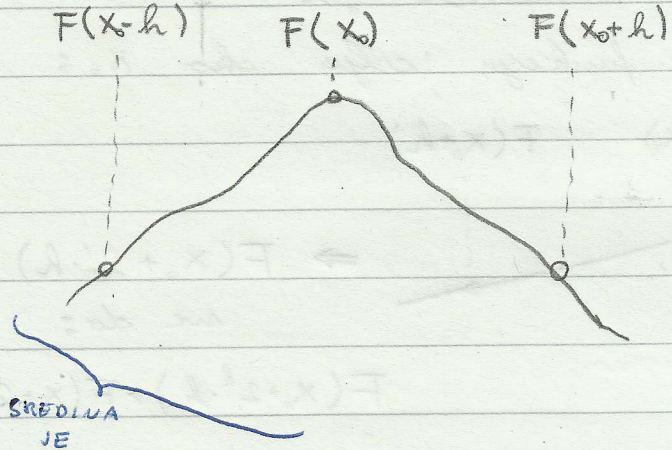
$$\left[x_0 - 2^k \cdot h,\ x_0 - 2^{k-2} \cdot h\right]$$



⇒ odmah znamo interval:

$$[x_0 - h, x_0 + h]$$

SREDINA JE NAJMANJA



⇒ ovo znamo da nije unimodalna funkcija, pa možemo na krivo koju stranu jer ne znamo koji od tih lokalnih minimuma je bolji

PR: Nadi unimodalni interval od:

$$f(x) = x^2 - 2 \; ; \quad x_0 = 100 \; ; \quad h = 1$$

	$x_0 - h$	$x_0 - 2h$	$x_0 - 4h$	....	$x_0 - 64h$	$x_0 - 128h$	$x_0 - 256$
X	99	98	96	....	36	-28	-156
f(X)				....			
	>	>	>		>	>	<

↳ dobijemo interval:

$$[-156, 36]$$

ZAD: Nad funkcijom $g(x)$ dolazi u uzastopni interval $[6, 18]$, uz početne uvjete: $x_0 = 2$, $h = 1$?

Odredi istinitost sljedećih tvrdnji:

a) $g(1) < g(2)$

b) $g(5) > g(10)$

c) $g(8) > g(18)$

d) $g(11) < g(18)$

↳ pogledajmo što je algoritam radio:

x	1	2	3	4	6	10	18
f(x)		>	>	>	>	>	<

a) nije istina - vidi se iz tablice

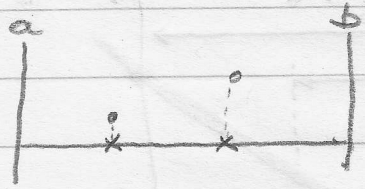
b) istina jer cijelo funkcija pada

c) ne znamo jer ne znamo o odnosima unutar intervala

d) istina

⇒ SMANJENJE UNIMODALNOG INTERVALA:

↳ imamo unimodalni interval, pa ga sada trebamo smanjit



↳ da bi ga mogli smanjiti trebamo odrediti dvije točke unutar intervala za uvrštavanje

↳ ne znamo koju granicu odbacit: gornju il donju?

↳ UVJETI ALGORITMA:

1) simetrija - brzina jednaka za odbacivanje gornje il donje granice

2) jednu točku želimo iskoristit

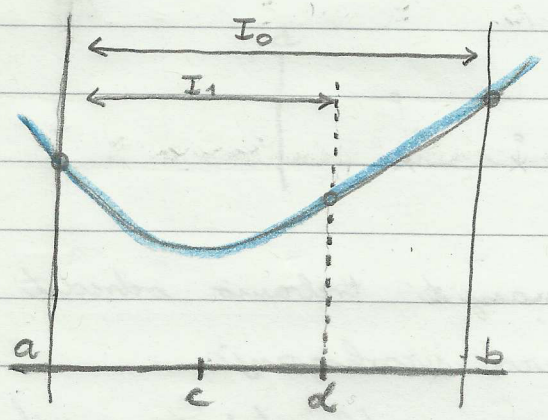
3) jednaka smanjenja intervala u svakoj iteraciji

↳ takav algoritam postoj:

• postupak zlatnog reza

("golden section search")

POSTUPAK ZLATNOG REZA



c, d ⇒ simetrični su

$|ca| = |db|$

$$\frac{d-a}{b-a} = \frac{b-c}{b-a} = k$$

↓

faktor smanjenja intervala

↳ računanje granica:

$$d = a + (b-a)\cdot k$$

$$c = b - (b-a)\cdot k$$

↳ prinjet sljedeće:

$$\underbrace{\frac{I_1}{I_0} = k \;,\; \frac{I_2}{I_1} = k}$$

$$\frac{I_2}{I_0} = k^2$$

↳ također vrijedi: $I_0 = I_1 + I_2 \;/ : I_0$

$$k^2 + k - 1 = 0$$

$$\vdots$$

$$k = \frac{\sqrt{5}-1}{2} \approx 0.6180\,g$$

GOLDEN RATIO !

↳ za „k" dodatno vrijedi: $k = \frac{1}{1+k}$

ZAD: Zadana je funkcija oblja s početnom točkom i intervalom: $f(x) = (x-4)^2$, $x_0 = 0$, $h = 1$.

Reducaraj do širine intervala $\varepsilon \leq 1$!

↳ prvo, treba pronaći unimodalni interval: I

x	-1	0	1	2	4	8
f(x)						
	>	>	>	>	<	

• unimodalni interval: $[2, 8]$

↳ sada slijedi sužavanje intervala: $c = b - k(b-a)$

$d = a + k(b-a)$

	a_i	c_i	d_i	b_i	
0:	2	4.292	5.708	8	$f(c_i) \leq f(d_i)$? TRUE
					↳ odbaci desnu stranu!
1:	2	3.416	4.292	5.708	$f(c_i) \leq f(d_i)$? FALSE
					↳ odbaci lijevu stranu!
2:	3.416	4.292	4.832	5.708	" analogno "
3:	3.416	3.957	4.292	4.832	" analogno "
4:	(3.416)		(4.292)		

manje od 1 pa smo gotovi:

$[3.416, 4.292]$

ZAD: Početni interval je $[-100, 100]$. Koliko iteracija zlatnog reza je potrebno da se interval smanji na $\varepsilon \leq 0.001$?

$$I_0 = 200$$

$$I_0 \cdot k^n \leq \varepsilon$$

$$k^n \leq \frac{\varepsilon}{I_0} \quad / \log$$

$$n \log k \leq \log \frac{\varepsilon}{I_0}$$

$$n \geq \frac{\log \frac{\varepsilon}{I_0}}{\log k}$$

$n \geq 25.36 \Rightarrow$ „n" mora biti barem 26

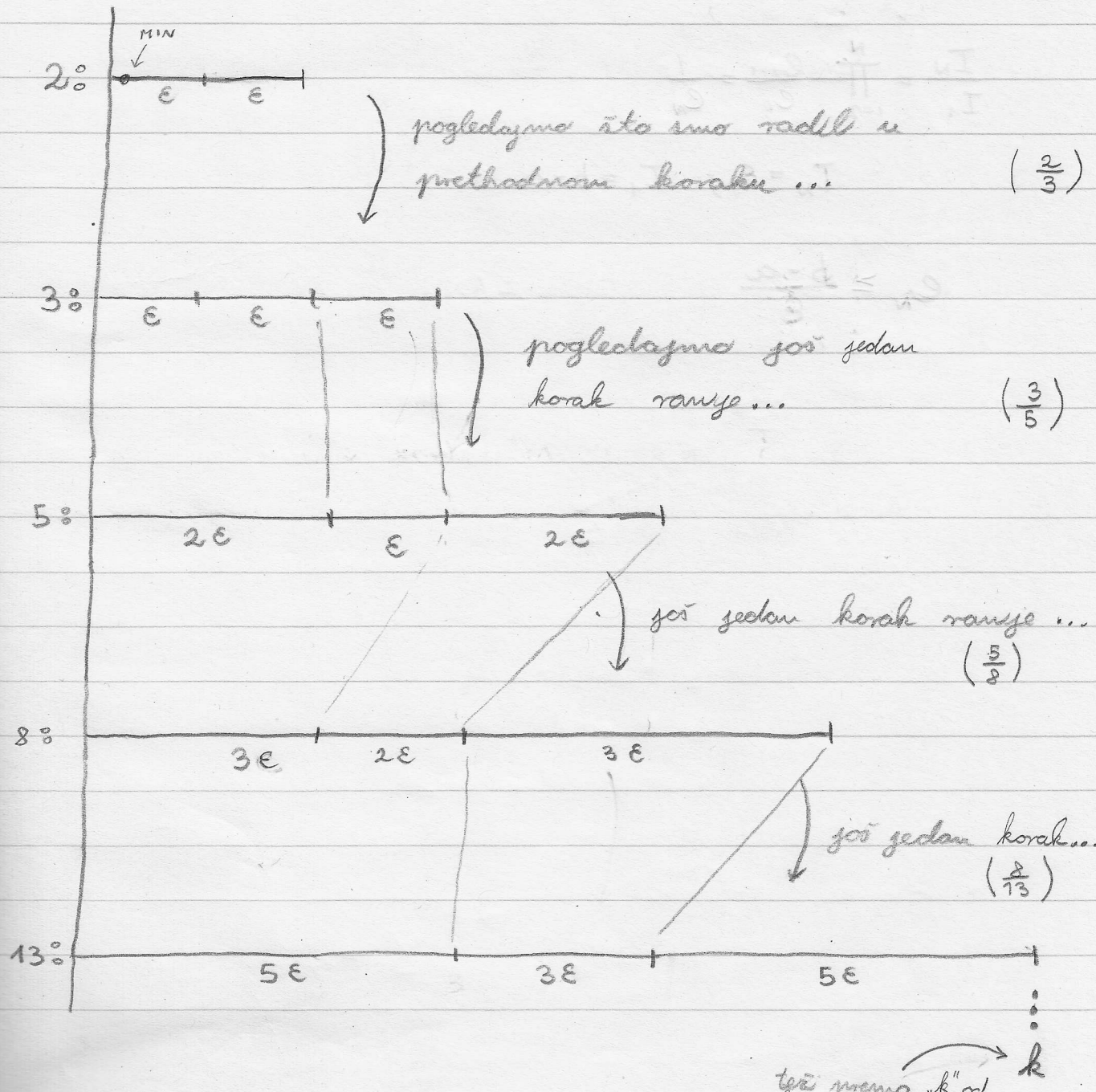
$(n \geq 26)$ //

FIBONACCIJEV POSTUPAK

⇒ zlatni rez je najbolja metoda smanjenja intervala u konstantnom vremenu

↳ pogledajmo drugi postupak:

· fibonaccijev postupak!



⇒ računanje granica: $d = a + \frac{e_{i-1}}{e_i}(b-a)$

$$c = b - \frac{e_{i-1}}{e_i}(b-a)$$

⇒ e sad, što je omjer $\frac{e_{i-1}}{e_i}$?

↳ to je faktor smanjenja u svakom koraku i on konvergira prema „k" u zlatnom rezu

$$\frac{I_N}{I_1} = \prod_{i=2}^{N} \frac{e_{i-1}}{e_i} = \frac{1}{e_N}$$

$$I_N = \varepsilon \; ; \quad I_1 = b-a$$

$$e_N \geq \frac{b-a}{\varepsilon}$$

PR: Funkcija cilja je dana: $f(x) = (x-4)^2$

Reducirat interval $[-2, 6]$ do $\varepsilon \leq 1$ sa filozofskom postupkom!

↳ traži prvi element koji je veći il jednak od trenutne širine intervala što je jednako 8:

NIZ: 1, 1, 2, 3, 5, **8**

	a_i	c_i	d_i	b_i
0:	-2	1	3	6
1:	1	3	4	6
2:	3	4	5	6
3:	3		4	5

u zadnjem koraku usputno računamo vrijednost funkcije cilja u točki „lijevo" sredine intervala

" npr. 4.01

stoga, 4.01 je bolje od 5, pa uzimamo za rezultat:

$[3, 4.01]$

OPTIMIRANJE FUNKCIJA VIŠE VARIJABLI BEZ UPORABE DERIVACIJA

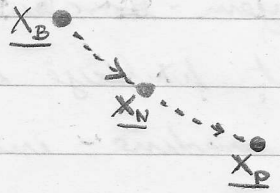
⇒ pogledat ćemo neke nove algoritme

POSTUPAK HOOKE-JEEVES: (pseudokod je na webu)

- ulazni podaci:
  - $\underline{X}_0$ ⇒ početna točka
  - $\Delta\underline{X}$ ⇒ vektor pomaka
  - $\underline{\varepsilon}$ ⇒ preciznost

  } ovo će nama biti skalar jer će biti isto za sve dimenzije

- postupak:
  - a) ISTRAŽIVANJE: $\pm\Delta X_i$, $\forall i$
  - b) KRETANJE:



⇒ naći bolju točku od bazne točke i skočiti za istu udaljenost u istom smjeru (PREBACIVANJE)

↳ podaci postupka:
- $\underline{X}_B$ ⇒ bazna točka
- $\underline{X}_N$ ⇒ točka dobivena pretraživanjem
- $\underline{X}_P$ ⇒ početna točka pretraživanja

- formula prebacivanja:

$$\underline{X}'_P = 2\underline{X}_N - \underline{X}_B$$

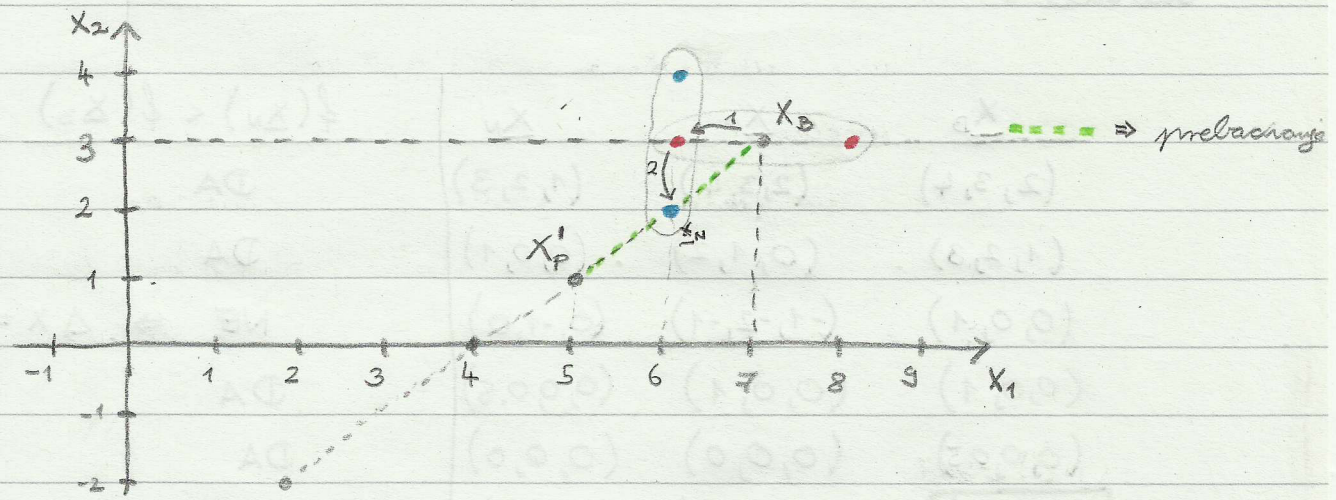
- rješenje: bazna točka posljednje iteracije

PR: Optimiraj sljedeću funkciju algoritmom Hooke-Jeeves:

$$f(\underline{x}) = x_1^2 + 4x_2^2$$

$$x_0 = (7, 3), \quad \Delta x = 1, \quad \varepsilon = 0.25$$

⇒ napravimo skicu za praćenje postupka:



⇒ tablica za praćenje bitnih varijabli:

$\underline{x}_B$	$\underline{x}_P$	$\underline{x}_N$	UVJET NAPREDOVANJA: $f(\underline{x}_N) < f(\underline{x}_B)$
(7, 3)	(7, 3)	(6, 2)	DA
(6, 2)	(5, 1)	(4, 0)	DA
(4, 0)	(2, -2)	(1, -1)	DA
(1, -1)	(-2, -2)	(-1, -1)	NE ⇒ sada povećaj rezoluciju: $\Delta x = 0.5$ ⇒ resetiraj se (zanemari trenutni $\underline{x}_P$ i $\underline{x}_N$)
(1, -1)	(1, -1)	(0.5, -0.5)	DA
(0.5, -0.5)	(0.0)	(0.0)	DA
$\boxed{(0,0)}$	(-0.5, 0.5)	(0, 0)	NE ⇒ $\Delta x = 0.25$ $(= \varepsilon)$

KRAJ!

↳ rješenje: (0, 0)

ZAD: Primjen postupak Hooke-Jeeves na funkciju:

$$f(\underline{x}) = x_1^2 + x_2^2 + x_3^2$$

$X_0 = (2, 3, 4)$, $\Delta X = 1$, $\varepsilon = 0.25$

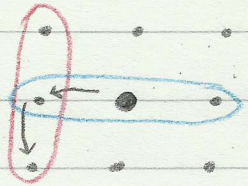
⇒ sada nećemo skicirat, već samo crtamo tablicu:

$\underline{X}_B$	$\underline{X}_P$	$\underline{X}_N$	$f(\underline{X}_N) < f(\underline{X}_B)$
(2, 3, 4)	(2, 3, 4)	(1, 2, 3)	DA
(1, 2, 3)	(0, 1, 2)	(0, 0, 1)	DA
(0, 0, 1)	(-1, -2, -1)	(0, -1, 0)	NE ⇒ $\Delta X = 0.5$
(0, 0, 1)	(0, 0, 1)	(0, 0, 0.5)	DA
(0, 0, 0.5)	(0, 0, 0)	(0, 0, 0)	DA
(0, 0, 0)	(0, 0, -0.5)	(0, 0, 0)	NE ⇒ $\Delta X = 0.25$ ($< \varepsilon$)

KRAJ!

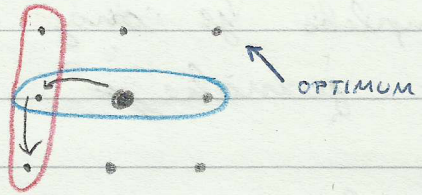
↳ rješenje: barna točka zadnje iteracije jest (0, 0, 0)

⇒ NAPOMENA: složenost algoritma je linearna (u ovisnosti o broju varijabli) jer svaku dimenziju ispitujemo dva puta



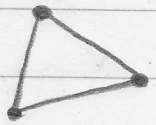
$O(2m)$

⇒ NAPOMENA: algoritam može izabrati krivi smjer jer ne ispituje cijelu okolinu barem točke

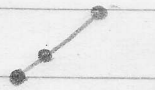


SIMPLEKS POSTUPAK (NELDER & MEAD):

↳ SIMPLEKS: skup od „N+1" točke koj u „N" dimenzijskom prostoru tvori „N" - dimenzijsku tvorevinu



u dvodimenzionalnom prostoru, ovo je simpleks



u dvodimenzionalnom prostoru, ovo nije simpleks

↳ pravilan simpleks je onaj koj ima udaljenosti među točkama jednake //

- **podaci postupka:**
  1) $\underline{X}[0], \underline{X}[1], \ldots, \underline{X}[N] \Rightarrow$ točke simpleksa
  2) $\underline{X}[l] \Rightarrow$ točka simpleksa s najnižom vrijednosti funkcije cilja (najmanja vrijednost)
  3) $\underline{X}[h] \Rightarrow$ točka simpleksa s najvećom vrijednosti funkcije cilja (najveća vrijednost)
  4) $\underline{X}_c \Rightarrow$ aritmetička sredina svih točaka osim najlošije (CENTROID)

- **operacije postupka:**
  1) REFLEKSIJA $\Rightarrow$ preslikavanje u odnosu na dani centroid

$$\underline{X}_R = (1+\lambda)\underline{X}_c - \lambda \cdot \underline{X}[R]$$

$\lambda \Rightarrow$ koeficijent refleksije (DEFAULT: $\lambda = 1$)

2) EKSPANZIJA ⇒ dodatno udaljavanje u smjeru dobivene točke refleksije

$$\underline{x}_E = (1-\gamma)\,\underline{x}_c + \gamma \cdot \underline{x}_R$$

γ ⇒ koeficijent ekspanzije
(DEFAULT: $\gamma = 2$)

3) KONTRAKCIJA ⇒ smanjivanje udaljenosti jer smo „nanjušili" da smo blizu optimuma

$$\underline{x}_K = (1-\beta)\,\underline{x}_c + \beta \cdot \underline{x}[h]$$

β ⇒ 0.5

4) POMAK PREMA $\underline{x}[l]$ ⇒ pomak prema najboljem rješenju simpleksa (i to svih točaka koje imamo)

$$\underline{x}[i] = \frac{1}{2}\left(\underline{x}[i] + \underline{x}[l]\right), \quad i = 1, \ldots, N+1, \quad i \neq l$$

ovo isto može biti parametar

⇒ NAPOMENA: ne treba znati kod algoritma, ali treba znati redoslijed i imena svih operacija i podataka

ZAD: Primjeni simplex postupak, ali samo nađi centroid na podatke: $f(x) = x_1^2 + x_2^2 + x_3^2$

$x_1 = (1, 2, 3)$

$x_2 = (0, 2, 4)$

$x_3 = (-2, 0, 3)$

$x_4 = (-4, 0, 1)$

$f(x_1) = 14$

$f(x_2) = 20$

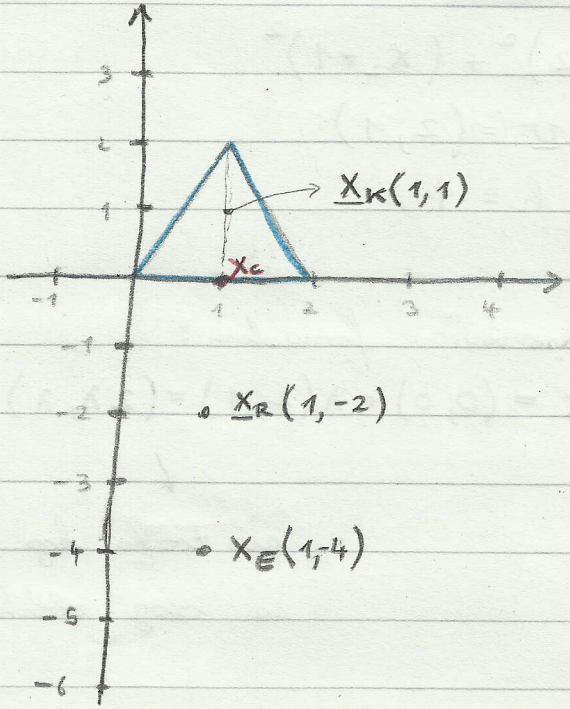
$f(x_3) = 13$

$f(x_4) = 17$

$$x_c = \frac{1}{3}\begin{bmatrix} 1+(-2)+(-4) \\ 2+0+0 \\ 3+3+1 \end{bmatrix} = \begin{bmatrix} -\frac{5}{3} \\ \frac{2}{3} \\ \frac{7}{3} \end{bmatrix}$$

ZAD: Zadan je skup točaka koje tvore simpleks: $(0,0)$, $(2,0)$, $(1,2)$ i funkcija čija je radna sa: $f(\underline{x}) = x_1^2 + x_2^2$

Skicirat centroid kontrakciju i ekspanziju!



PR: Uvod u odredivanje minimuma funkcije više varijabli korištenjem postupaka za nalaženje minimuma funkcije jedne varijable!

↳ nadi minimum funkcije tako da nađeš unimodalni interval i koristenjem zlatnog reza ur dane podatke:

$$f(\underline{x}) = (x_1 - 2)^2 + (x_2 + 1)^2$$

$$\underline{x}_0 = (0, 0) \; ; \quad \underline{v} = (2, 1)$$

$$\varepsilon \leq 0.5 \; ; \quad h = 1$$

⇒ krenemo s „magičnom" formulom:

$$\underline{x} = \underline{x}_0 + \lambda \underline{v} = (0, 0) + \lambda(2, 1) = (2\lambda, \lambda)$$

↓

sve točke koje razmatramo su ovog oblika!

$$f(\underline{x}) = (2\lambda - 2)^2 + (\lambda + 1)^2 = f(\lambda)$$

↓

pomoću vektora $\underline{v}$ smo se ograničili na jedan pravac i sada korištenjem nalaženja unimodalnog intervala i zlatnim rezom možemo naći funkciju

↳ NAPOMENA: varijacije algoritma se razlikuju samo po izboru pravca (izbor vektora $\underline{v}$) po kojem se traži unimodalni interval

⇒ unimodalni interval:

λ	−1	0	1	2
$f(\lambda)$	16	5	4	13
	>	>	>	<

$\Rightarrow \lambda \in [0, 2]$

⇒ relativ rez:

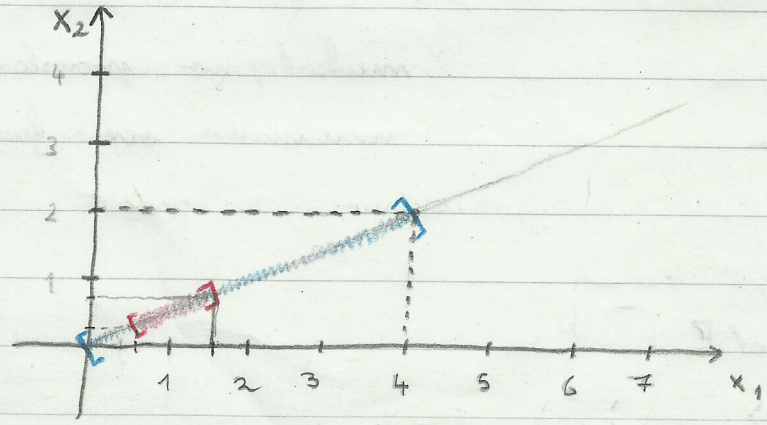
a	c	d	b	
0	0.764	1.236	2	
0	0.472	0.764	1.236	
0	0.292	0.472	0.764	
0.292	0.472	//	0.764	$\Rightarrow \lambda \in [0.292, 0.764]$

λ_1 λ_2

⇒ vraćamo se u vektor $\underline{X}$ sa dobivenim intervalom u kojem je λ smješten:

$$\underline{X} \in [X_0 + \lambda_1 \underline{v}, X_0 + \lambda_2 \underline{v}]$$

$$\underline{X} \in \left[\begin{matrix} 0.584 \\ 0.292 \end{matrix}, \begin{matrix} 1.528 \\ 0.764 \end{matrix} \right]$$

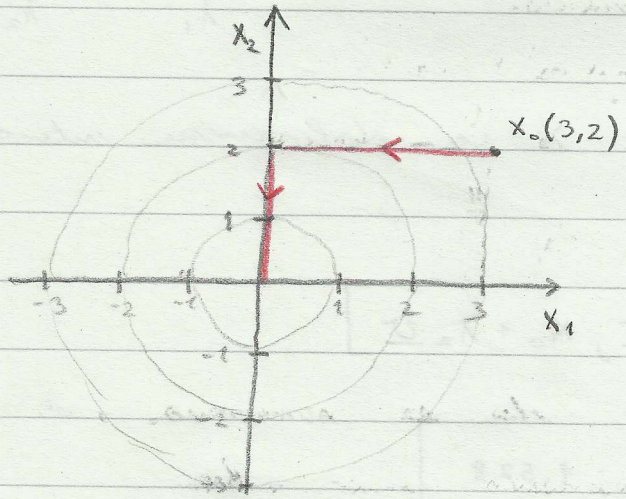


[ ] ⇒ početni interval

[ ] ⇒ redukcijski interval

TRAŽENJE U SMJEROVIMA KOORDINATNIH OSI$

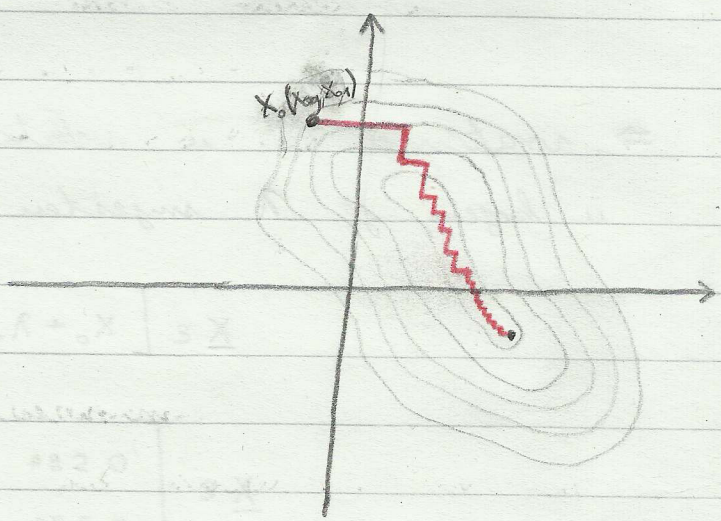
- traži najprije po prvoj varijabli, zatim po drugoj varijabli i tako po svima
  - ↳ nakon toga vrati se ponovno na prvu varijablu, to radi do mjesta zaustavljanja
- problem ove metode:
  - ↳ ako je „dolina" (minimum) orijentirana tako da nije paralelna s koordinatnim osima, sporo napredujemo kroz prostor pretraživanja



$$f(\underline{x}) = x_1^2 + x_2^2$$

⇓

nema problema sa ovakvom funkcijom cilja



⇓

mukotrpno pronalaženje minimuma ove funkcije

⇒ PROBLEM: ostar brod!?

POSTUPAK PO POWELLU:

↳ kako bi ispravili problem traženja u smjerovima koordinatnih osi trebamo adaptirati postupak (na, još uvijek bez derivacija)

- koristit će svojstva kvadratne funkcije:
  - ↳ opći oblik: („kanonski oblik")

$$F(\underline{x}) = c + \underline{b}^T \underline{x} + \frac{1}{2} \underline{x}^T \underline{A}\, \underline{x}$$

PR: Pogledajmo sljedeću kvadratnu funkciju:

$$F(\underline{x}) = (x_1 - 2)^2 + (x_2 + 1)^2$$

↳ sredimo gornju funkciju:

$$F(\underline{x}) = x_1^2 - 4x_1 + 4 + x_2^2 + 2x_2 + 1$$

$$F(\underline{x}) = 5 + \begin{bmatrix} -4 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

⇓ zapis u kanonskom obliku

- ideja je u pretpostavci da se promatrana funkcija u okolini minimuma ponaša kao neka kvadratna funkcija
- uvjet za ekstrem kvadratne funkcije:

$$\nabla F = \underline{A}\, \underline{x} + \underline{b} \quad ; \quad \underline{H} = \underline{A}$$

↑ Hesseova matrica (konstantna za kvadratnu funkciju)

$$\nabla F = \underline{0} \Rightarrow \underline{A}\, \underline{x} + \underline{b} = \underline{0}$$

$$\underline{A}\, \underline{x} = -\underline{b}$$

$$\boxed{\underline{x}_{min} = -\underline{A}^{-1} \cdot \underline{b}}$$

• kako nam ovo sad pomaže na postupak pronalaska minimuma?

↳ pa, trebamo još jedan pojam:

⇒ KONJUGIRANI SMJEROVI:

$$\underline{v}_i, \underline{v}_j \iff \underline{v}_i^T \cdot \underline{A} \cdot \underline{v}_j = 0$$

$$\left(\underline{v}_i^T \cdot \underline{A} \cdot \underline{v}_i > 0\right)$$

• **dodatna svojstva** („pretpostavke"):

1) minimum kvadratne funkcije dobiva se optimiranjem u smjeru konjugiranih smjerova s obzirom na matricu A

2) novi konjugirani smjer dobiva se spajanjem minimuma na dva paralelna pravca zadanog smjera:

↳ **objašnjenje dodatnih svojstava:**

1) tražimo minimum na pravcu $\underline{v}_1$ od početne točke X_0, dolijemo (za neki lambda) najbolji mogući X_1:

$$\underline{X}_1 = \underline{X}_0 + \lambda_1 \underline{v}_1$$

↳ uzmemo konjugirani $\underline{v}_2$ (konjugirani smjer) i dolazimo do X_2, i to radimo „n" puta!?

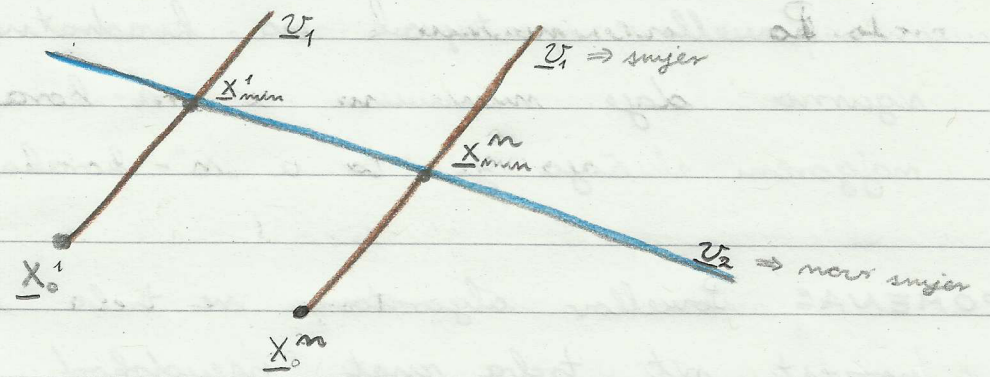
$$\underline{X}_2 = X_1 + \lambda_2 \underline{v}_2$$

$$\vdots$$

$$\boxed{\underline{X}_n = \underline{X}_0 + \sum_{j=1}^{n} \lambda_j \cdot \underline{v}_j}$$

→ $\underline{X}_n$ je minimum (postoj dokaz za to u skripti (ne treba znati))

2) trebamo znati naći drugi smjer!



↳ govori o tome kako sagraditi novi smjer

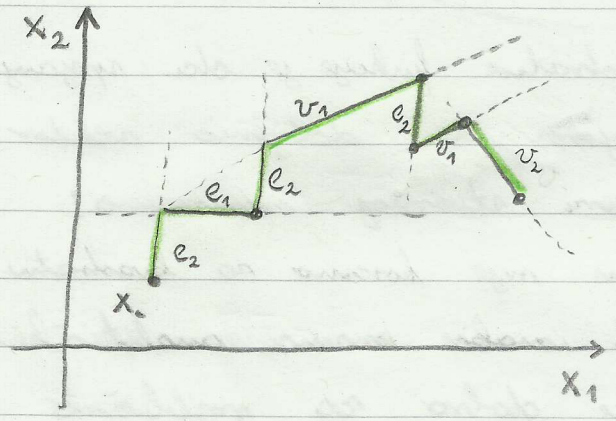
⇒ POWELLOV ALGORITAM:

- pretpostavlja da funkcija koja se optimira je kvadratna (približno kvadratna)
- time može koristiti spomenuta svojstva

↳ na početku traži po koordinatnim osima!

↳ dalje, konjugirano za novi smjer!

Pr: Ilustracija algoritma!



⇒ smjerovi:

e_1 e_2 v_1

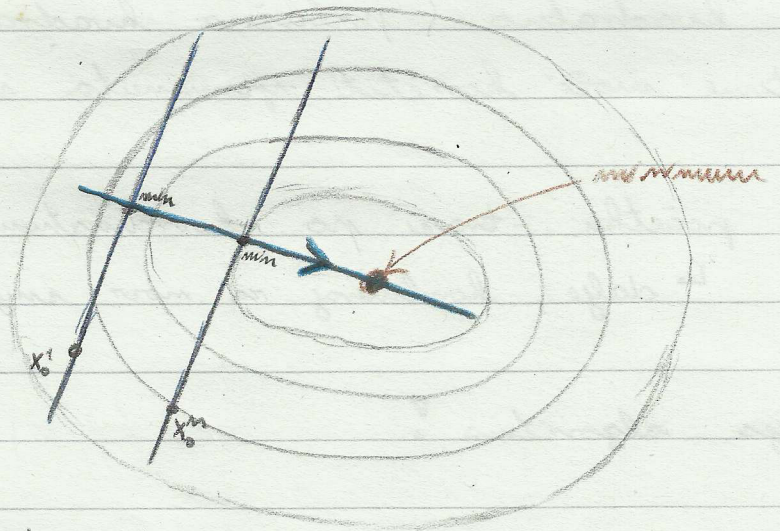
e_2 v_1 v_2

⇒ bitne stvari za znat:

↳ Powellov postupak na kvadratnoj funkciji sigurno daje minimum u „n"- koraka (u najgorem slučaju je to u „n"- koraka)

⇒ NAPOMENA: Powellov algoritam ne treba znat provest nit treba znat pseudokod, al treba znat ilustrirat algoritam i osnovne ideje algoritma

⇒ ilustracija korištenih svojstava:



↳ svojstvo kvadratne funkcije je da spojanjem minimuma po smjerovima dobijemo vektor koji gledo u smjeru stvarnog minimuma

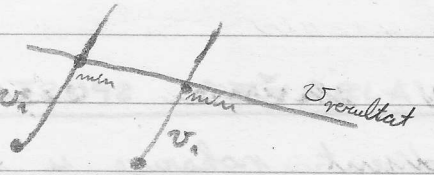
↳ no, to nam nije korisno za kvadratne funkcije jer njih izraze znamo analitički riješiti (to je dobro za približno kvadratne funkcije)

ZAD: Zadana je funkcija $F(\underline{x}) = (x_1-2)^2 + (x_2+1)^2$. Za zadanu funkciju odredi konjugirane smjer smjeru $\underline{v}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

↳ može se rješit:

a) analitički: $\underline{v}_i^T \underline{A} \underline{v}_j = 0$

b) naći minimum u svom smjeru i spoj ih (gotova grafička metoda)



OPTIMIRANJE FUNKCIJA VIŠE VARIJABLI GRADIJENTNIM POSTUPCIMA

⇒ sada ćemo koristiti gradijent (derivacije) u pronalasku minimuma funkcija

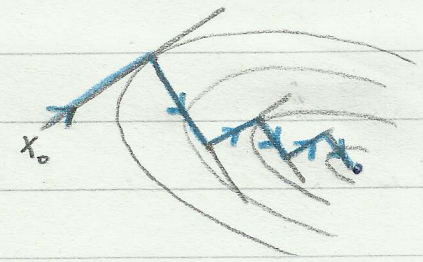
⇒ **METODA NAJBRŽEG SPUSTA:**

- gradijent pokazuje u smjeru najbržeg rasta funkcije, pa mi želimo ići u suprotnom smjeru

$$\underline{v} = -\frac{\nabla F}{\|\nabla F\|}$$

- loše strane:

↳ svaki novi smjer je okomit na prethodni, pa ponovno radimo „cik-cak" (dokaz za ovo imamo u Powellovom postupku)



Pr: Zadana je funkcija $F(\underline{x}) = (x_1-4)^2 + 4(x_2-2)^2$, $x_0 = (0,0)$.
Provedi jednu iteraciju metode najbrzeg spusta!
(Minimum na pravcu odredi analitički!)

$$\frac{\partial F}{\partial x_1} = 2(x_1-4) = 2x_1 - 8$$

$$\frac{\partial F}{\partial x_2} = 8(x_2-2) = 8x_2 - 16$$

$$\nabla F = \begin{bmatrix} 2x_1 - 8 \\ 8x_2 - 16 \end{bmatrix}$$

↳ početni gradijent:

$$\nabla F / \underline{x}_0 = \begin{bmatrix} -8 \\ -16 \end{bmatrix}$$

$$\underline{v}_0 = -\frac{\nabla F}{\|\nabla F\|} = \frac{1}{\sqrt{320}} \begin{bmatrix} 8 \\ 16 \end{bmatrix} \Rightarrow$$ no, mi radimo na papiru, pa ne trebamo normirani vektor smjera

↳ nama je samo bitan smjer, pa za v_0 možemo uzeti:

$$\underline{v}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

↳ svodimo na funkciju jedne varijable:

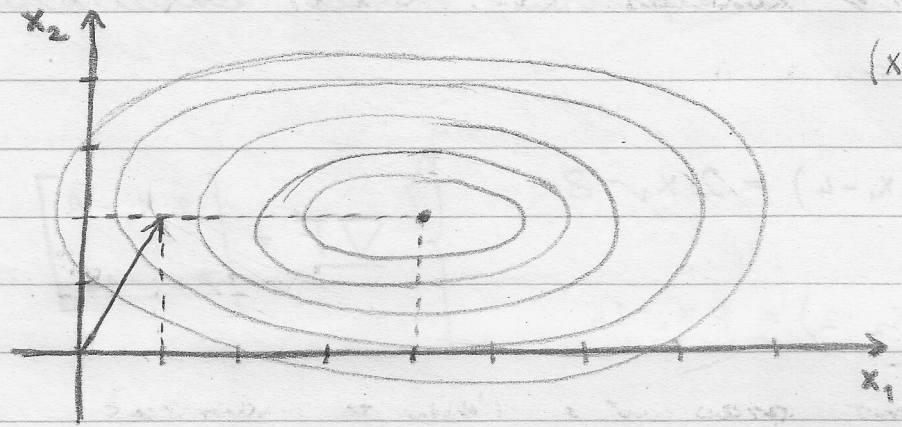
$$F(\underline{x}_0 + \lambda \underline{v}_0) = (\lambda - 4)^2 + 4(2\lambda - 2)^2$$

$$f(\lambda) = 17\lambda^2 - 40\lambda + 32$$

↳ analitički minimum: $\frac{df(\lambda)}{d\lambda} = 0 = 34\lambda - 40 \Rightarrow \lambda = \frac{40}{34} = 1.176$

$$\underline{x}_1 = \underline{x}_0 + \lambda \underline{v}_0 = \begin{bmatrix} 1.176 \\ 2.352 \end{bmatrix}$$

↳ no, pokazuje li dobiveni gradijent prema minimumu?



$(x_1-4)^2+4(x_2-2)^2$

↓

ova četvorka je kružnica za elipse

⇓

ne pokazuje prema minimumu jer su konture elipse, a mi smo krenuli od neke osi od elipse

↳ da smo krenuli od savršenog paraboloida (npr. $F(\underline{x}) = (x_1-4)^2 + (x_2-2)^2$) onda od svake početne točke gradijent gleda prema minimumu

NEWTON - RAPHSONOV POSTUPAK:

PR: Pogledajmo funkciju $F(\underline{x}) = (x_1-4)^2 + 4(x_2-2)^2$ te je optimiramo $\overset{P}{\circ}$

↳ nađimo gradijent i druge derivacije:

$$\frac{\partial F}{\partial x_1} = 2x_1 - 8 \; ; \quad \frac{\partial F}{\partial x_2} = 8x_2 - 16 \; ; \quad \frac{\partial F}{\partial x_1^2} = 2 \; ; \quad \frac{\partial F}{\partial x_2^2} = 8$$

↳ napišimo gradijent i Hesseovu matricu:

$$\underline{\nabla F} = \begin{bmatrix} 2x_1 - 8 \\ 8x_2 - 16 \end{bmatrix} ; \quad \underline{H} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$

↳ trebamo i inverz Hesseove matrice:

$$H^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{8} \end{bmatrix}$$

↳ krenimo u optimiranje od početne točke $\underline{x}_0 = (0,0)$:

$$\underline{\nabla F}/_{\underline{x}_0} = \begin{bmatrix} -8 \\ -16 \end{bmatrix} ; \quad \underline{\Delta x} = -\underline{H}^{-1} \cdot \underline{\nabla F}/_{\underline{x}_0} = -\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{8} \end{bmatrix} \cdot \begin{bmatrix} -8 \\ -16 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

ovo je stvarno minimum!

↳ vidi da za $\underline{x}_0 = (1,1)$ također dobivamo minimum u jednom skoku

↳ minimum u jednom skoku dobivamo zato jer je polazna funkcija kvadratna

↳ za približno kvadratne funkcije, trebamo mnogo više koraka!

↳ **loše strane:**

- NESTABILNOST POSTUPKA (divergencija)
  ↳ skok može udaljiti od minimuma funkcije i divergirat za nekvadratne funkcije

Pr: Dokaži da postupak divergira za jednu dimenziju (analogno za više dimenzija 🙂).

$$f(x) = f(\underbrace{x_0 + \Delta x}_{x_{min}}) = f(x_0) + f'(x_0)\cdot\Delta x + \frac{1}{2} f''(x_0)\Delta x^2$$

$$\frac{\partial f(x_0+\Delta x)}{\partial \Delta x} = 0 = f'(x_0) + f''(x_0)\Delta x$$

$$\Delta x = -\frac{f'(x_0)}{f''(x_0)}$$

PR: Pokaži da postupak Newton-Raphson divergira za funkciju $f(x) = \frac{x^2}{2} - \cos x$.

↳ nađemo dvije derivacije:

$$f'(x) = x + \sin x \qquad f''(x) = 1 + \cos x$$

↳ krenemo od $x_0 = 1$:

$$f'(x_0) = 1.841 \qquad f''(x_0) = 1.54$$

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} = -0.195$$

↳ ponovno računamo i vidimo da postupak ide prema nuli (jer smo dovoljno blizu minimumu da se funkcija ponaša kvadratno)

↳ da smo uzeli npr $x_0 = 3$, postupak će divergirati jer funkcija ne sliči kvadratnoj

DZ Proveđi Newton-Raphsonov postupak na

$$F(\underline{x}) = \frac{x_1^4}{4} - x_1^2 + 2x_1 + (x_2 - 1)^2 \; ; \quad \underline{x}_0 = (0, 2)$$

(postupak bi trebao oscilirati)

↳ TEHNIKE POVEĆANJA STABILNOSTI

a) minimizacija na pravcu

b) smanjivanje koraka Δx

c) ako postupak počne divergirati, koristi najbrži spust

⇒ dodatne loše strane Newton-Raphsonovog postupka

- potrebna je druga parcijalna derivacija u svakom koraku kako bi dobili cijelu Hesseovu matricu
- potrebno je u svakom koraku izračunati rješenje sustava $\underline{H}\underline{x} = \underline{b}$ (ili LUP dekompozicijom ili strukturnim nalaženjem inverza

⇒ ovaj Newton-Raphsonov postupak je dobar jer koristi sve informacije na raspolaganju (i prvu i drugu derivaciju)

↳ dodatno, algoritam je robusan ako se koriste tehnike povećanja stabilnosti

↳ no svejedno je jako računalno zahtjevan zbog drugih derivacija

FLETCHER - POWELL POSTUPAK:

- predstavnik kvazi - Newtonovih metoda
- Hesseova matrica (tj njena inverzija) se aproksimira tijekom provedbe postupka te se tako smanjuje računalna zahtjevnost:

$$\underline{G}_j \approx \underline{H}_j^{-1}$$

- funkcija se optimira u zadanom smjeru:

$$\underline{v}_{j+1} = -\underline{G}_j \cdot \nabla F$$

- ako je funkcija približno kvadratna, aproksimacija Hesseove matrice je dobra!
  - ↳ nakon dovoljno iteracija, aproksimacija konvergira prema Hesseovoj matrici

- na početku, krećemo u smjeru suprotnom od gradijenta:

$$\underline{G}_0 = \underline{E} \Rightarrow \underline{v}_1 = -\nabla F$$

  - ↳ znači, u najgorem slučaju je ova metoda gradijentni spust

- ova metoda se danas koristi za nalaženje optimuma
- formula Fletchera i Powella jamči da će za kvadratnu funkciju matrica $\underline{G}$ postati jednaka $\underline{H}^{-1}$ za „n" iteracija

OPTIMIRANJE S OGRANIČENJIMA

⇒ uz zadanu funkciju cilja $F(\underline{x})$ imamo i zadana ograničenja

⇒ VRSTE OGRANIČENJA:

a) EKSPLICITNA

↳ gornja i donja granica pojedine varijable

$$\underline{x}_D \leq \underline{x} \leq \underline{x}_G$$

↳ ova ograničenja je lagano raditi, te nam ne predstavljaju velike probleme

b) IMPLICITNA

↳ dolaze u dva oblika

1) nejednadžbe: $g_i(\underline{x}) \geq 0, \forall i$

2) jednadžbe: $h_j(\underline{x}) = 0, \forall j$

⇒ svako ograničenje u obliku jednadžbe smanjuje stupanj slobode za jedan i tako smanjuje prostor pretraživanja

⇒ no, ograničenja jednadžbi možemo aproksimirati s dvije nejednadžbe:

$$|h(\underline{x})| \leq \varepsilon \quad \Rightarrow \quad \begin{matrix} -h(\underline{x}) + \varepsilon \geq 0 \\ h(\underline{x}) + \varepsilon \geq 0 \end{matrix}$$

⇒ ne želimo nužno uvijek jednadžbe raspisivati kao nejednadžbe jer neki postupci mogu raditi s ograničenjima u obliku jednadžbi

↳ no, neki postupci rabe isključivo ograničenja u obliku nejednadžbi

⇒ NAPOMENA: uvjetna ograničenja, dozvoljeno područje ne mora niti postojati

↳ da bi stvar bila teža, ne možemo jednostavno reći postoji li uopće dozvoljeno područje (ovo može biti toliko teško reći kao i sam problem optimiranja)

⇒ OPĆI OBLIK PROBLEMA OPTIMIRANJA:

$F(\underline{x})$ ⇒ funkcija cilja

$\underline{x}_D \leq \underline{x} \leq \underline{x}_G$ ⇒ eksplicitna ograničenja

$\underline{g}(\underline{x}) > \vec{0}$ ⇒ ograničenja nejednakosti

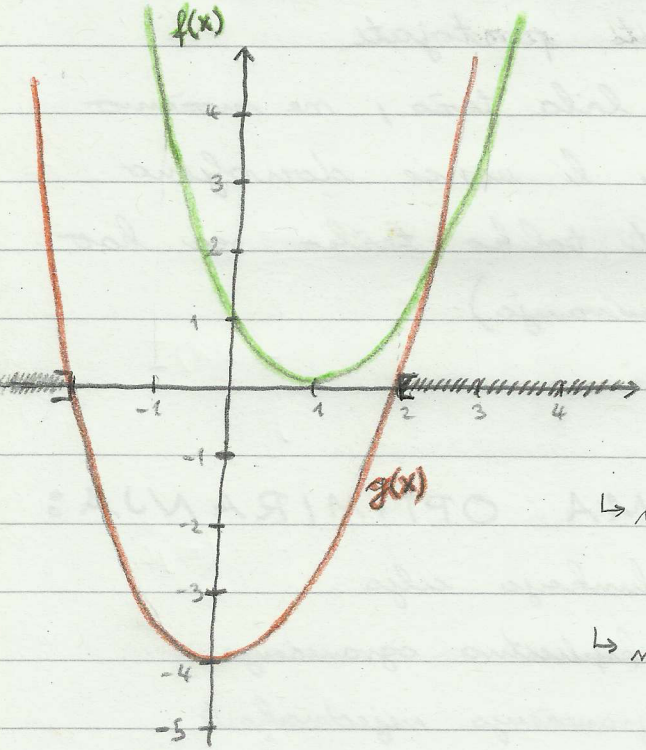
$\underline{h}_j(\underline{x}) = \vec{0}$ ⇒ ograničenja jednadžbi

⇒ BITNA SPOZNAJA:

↳ za unimodalne funkcije vrijedi:

- uz dodavanje ograničenja, minimum ili ostaje isti ili leži na granici dozvoljenog područja

⇒ još jedna spoznaja nam je bitna, pa pogledajmo za primjer funkciju $f(x) = (x-1)^2$:



- imamo ograničenje:

$$g(x) = x^2 - 4 \geq 0$$

$$x_{MIN} = 2 \Rightarrow f(x_{MIN}) = 1$$

↳ minimizacija od $x_0 = 10$ dolazimo:

$$x'_{MIN} = 2 \Rightarrow f(x'_{MIN}) = 1$$

↳ minimizacija od $x_0 = -10$ dolazimo:

$$x''_{MIN} = -2 \Rightarrow f(x''_{MIN}) = 9$$

⇓

ovisnost o početnoj točki!
(VIŠEMODALNOST!)

⇒ DRUGA BITNA SPOZNAJA:

↳ dodavanjem ograničenja unimodalnoj funkciji, optimizacija može postati višemodalna

⇒ RJEŠAVANJE PROBLEMA S OGRANIČENJIMA:

↳ dva načina:

1) transformacija originalnog problema u problem bez ograničenja i tada primjenimo postojeće postupke optimiranja

↳ matematičkim trikovima se rješavamo ograničenja

2) korištenje prilagođenih postupaka koji uzimaju ograničenja u obzir

TRANSFORMACIJA ORIGINALNOG PROBLEMA U PROBLEM BEZ OGRANIČENJA

⇒ ODSTRANJIVANJE EKSPLICITNIH OGRANIČENJA:

↳ uvodi se pomoćna neograničena varijabla i definira se transformacija u dozvoljeno područje

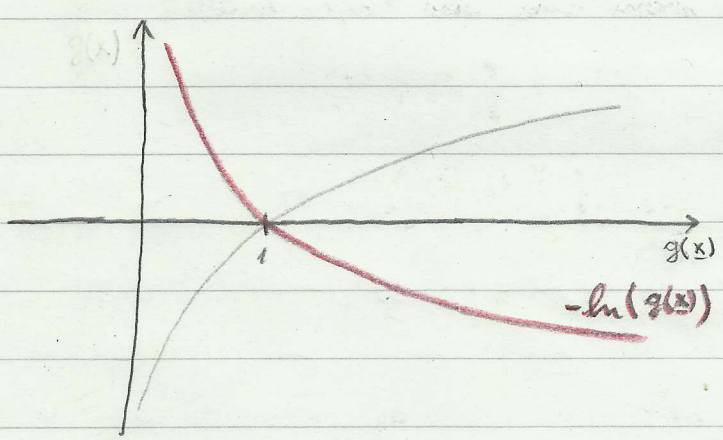
↳ drugim riječima, uvodimo mapiranje ▽: POGLEDAJ NEKE FUNKCIJE U SKRIPTI! ▽:

⇒ transformiranje možemo radit na nekoliko načina:

a) TRANSFORMIRANJE S UNUTARNJOM POČETNOM TOČKOM:

$I(x)$ { • definiramo pomoćnu funkciju koja raste prema grani dozvoljenog područja, a van dozvoljenog područja je beskonačno

$s(T)$ { • e sad, trebamo i dodatnu funkciju koja definira koliko nas prva početna funkcija tjera da ostanemo unutar dozvoljenog područja



⇒ kako izabrati $I(x)$?

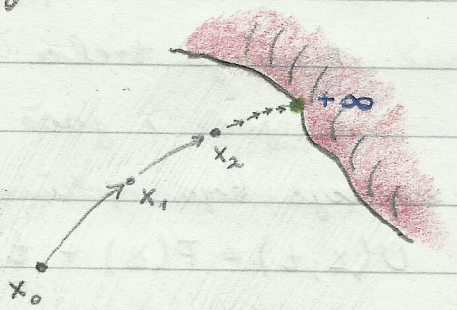
↳ prirodan odabir je negativan logaritam funkcije ograničenja:

$$I(x) = -\ln(g(x))$$

↳ naša funkcija koju optimiramo je sada:

$$U(\underline{x}, r) = F(x) - r \sum \ln(g(\underline{x}))$$

↳ „r" je početno proizvoljan, a svaki puta ga smanjujemo !



minimum je najčešće na granici dozvoljenog područja

↳ napomena: pogledaj primjer u skripti za funkciju $F(\underline{x}) = x_1 + x_2$ uz ograničenja:

$$g_1(\underline{x}) = x_2 \geq 0$$

$$g_2(\underline{x}) = x_1 - x_2^2 \geq 0$$

↳ NUŽAN UVJET: postojanje početne točke koja zadovoljava ograničenja, tj. točka koja je unutar dozvoljenog područja

↳ ako nemamo točku unutar dozvoljenog područja, postoj metoda za <u>pronalaženje unutarnje točke</u>:

• definiraj pomoćnu funkciju oblika:

$$G(\underline{x}) = -\sum_i t_i g_i(\underline{x})$$

$$t_i = \begin{cases} 0, & g_i(\underline{x}) \geq 0 \\ 1, & \text{inače} \end{cases}$$

ova funkcija govori koliko $\underline{x}$ narušava ograničenja, pa to minimiziramo da nađemo nulu (ne je zadovoljeno)

ovo minimiziraj nekim ranijim postupkom!

↳ tamo gdje je ovo **nula** imamo $\underline{x}$ unutar dozvoljenog područja

b) TRANSFORMIRANJE S VANJSKOM POČETNOM TOČKOM:

- sada je pomoćna funkcija takva da nas tjera unutra
- ona dodatna funkcija koja definira jačinu tjeranja treba rast, tako da nas sve jače i jače tjera unutra

↳ naša funkcija koju optimiramo je sada:

$$U(\underline{x}, t) = F(\underline{x}) + t \cdot \sum \left(\frac{g_i(\underline{x}) - |g_i(\underline{x})|}{2} \right)^2$$

↳ „t" povećavamo

↳ ova transformacija se super ponaša za ograničenja u obliku jednadžbi (VIDI SKRIPTU!)

c) TRANSFORMIRANJE NA MJEŠOVIT NAČIN:

- ograničenja nejednadžbi rješi na način „a)", a ograničenja nejednadžbi na način „b)"
- mora vrijediti uvijet iz „a)"
- za „r" možemo pisati:

$$r = \frac{1}{t}$$

↳ naša funkcija koju optimiramo:

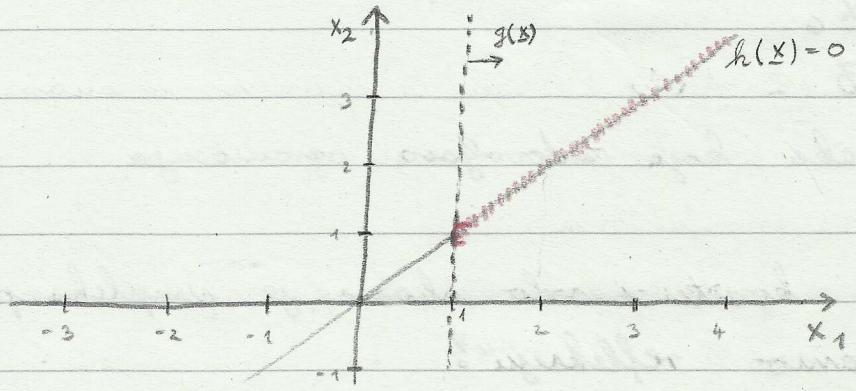
$$U(\underline{x}, t) = F(\underline{x}) - \frac{1}{t} \sum \ln(g_i(\underline{x})) + t \sum h_i^2(\underline{x})$$

PRS Pogledajmo funkciju $f(\underline{x}) = x_1^2 + (x_2 - 1)^2$ uz ograničenja:

$$g(\underline{x}) = x_1 - 1 \geq 0$$

$$h(\underline{x}) = x_2 - x_1 = 0$$

↳ skicirajmo funkciju i ograničenja:



↳ nadomjesna funkcija čija se transformacija može na mješovit način:

$$U(\underline{x}, t) = x_1^2 + (x_2 - 1)^2 - \frac{1}{t} \ln(x_1 - 1) + t(x_2 - x_1)^2$$

POSTUPAK PO BOXU

⇒ početno:

problem $\begin{cases} F(\underline{x}) \\ \underline{x}_0 \leq \underline{x} \leq \underline{x}_G \\ g(\underline{x}) \geq \vec{0} \end{cases}$

x_0: točka koja zadovoljava ograničenja

⇒ ovaj postupak koristi i vrlo sličan je simpleks postupku, al koristi samo **refleksiju**!

⇒ <u>napomena</u>: dozvoljeno područje mora biti konveksno da bi se mogao generirat početni skup točaka (mora postojat spojnica za točke unutar dozvoljenog područja)

⇒ RJEŠENJE ALGORITMA:

- centroid radnje iteracije

⇒ UVJET ZAUSTAVLJANJA:

↳ kada udaljenost između najlošije točke i centroida padne ispod neke granice:

$$|\underline{x}_h - \underline{x}_c| < \varepsilon$$

PR: Proved postupak po Boxu na sljedećoj funkciji uz dana ograničenja:

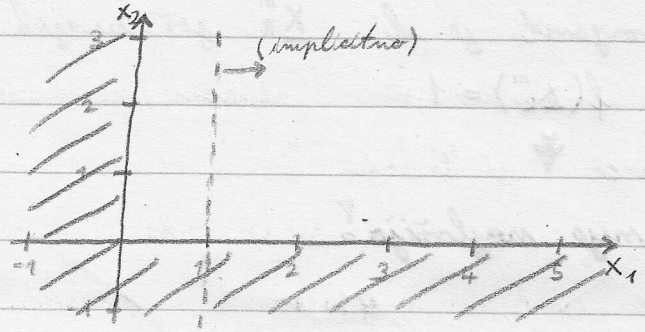
$$f(\underline{x}) = x_1^2 + (x_2 - 1)^2$$

$$x_1, x_2 \in [0, \infty\rangle$$

$$x_1 - 1 \geq 0 \quad \Rightarrow \text{ovo je implicitno!}$$

(tako je zadano!)

↳ napravimo sliku dozvoljenog područja:



↳ neka su zadane i početne točke i faktor refleksije:

$$(1,0),\ (2,1),\ (2,3),\ (1,3) \qquad \alpha = 2$$

↳ nacrtamo tablicu:

$\underline{x}$	(1,0)	(2,1)	(2,3)	(1,3)
$f(\underline{x})$	2	4	8	5

najlošija, pa je centroid:

$$\underline{x}_c = \left(\frac{1+2+1}{3}, \frac{0+1+3}{3}\right) = \left(\frac{4}{3}, \frac{4}{3}\right)$$

↳ radimo refleksiju:

$$\underline{x}_R = (1+\alpha)\underline{x}_c - \alpha\,\underline{x}_h = (4,4) - (4,6) = (0,-2)$$

↳ kako b zadovoljil eksplicitno, pomičemo je k najbližoj unutar dozvoljenog područja:

$$\underline{x}_R' = (0,0)$$

↳ ne zadovoljava eksplicitna, pa pomičemo na pola prema centroidu sve dok ne zadovolji eksplicitna ograničenja:

$$\underline{x}_R'' = \left(\frac{2}{3}, \frac{2}{3}\right)$$

↓ opet ?

$$\underline{x}_R''' = (1, 1) \Rightarrow \text{ova zadovoljava!}$$

↳ treći provjerit je li $\underline{x}_R'''$ još uvijek najlošija:

$$f(\underline{x}_R''') = 1$$

↓

nije najlošija!

>>> KRAJ PRVE ITERACIJE <<<

↳ **druga iteracija:**

$\underline{x}$	$(1,0)$	$(2,1)$	$(1,1)$	$(1,3)$
$f(\underline{x})$	2	4	1	5

najlošija: $\underline{x}_h$

$$\underline{x}_C = \left(\frac{4}{3}, \frac{2}{3}\right) \; ; \quad \underline{x}_R = (2, -4)$$

↳ zbog eksplicitnih ograničenja: $\underline{x}_R' = (2, 0)$

↳ implicitna zadovoljava

↳ $\underline{x}_R'$ je najlošija jer je $f(\underline{x}_R') = 5$, pa računamo:

$$\underline{x}_R'' = \frac{1}{2}(\underline{x}_R + \underline{x}_C) = \left(\frac{5}{3}, \frac{1}{3}\right)$$

>>> KRAJ DRUGE ITERACIJE <<<

PR: Kako bi pokazal zašto se uzima „2n" točaka, a ne „n+1" pogledajmo postupak po Boxu na sljedećoj funkciji uz ograničenja i početni skup točaka:

$$f(\underline{x}) = x_1^2 + x_2^2$$

$$x_1, x_2 \in [1, \infty\rangle \qquad (1,4),\ (1,2),$$

$$x_1 + x_2 - 3 \geq 0 \text{ (implicitno)} \qquad (3,3),\ (4,3)$$

↳ pišemo tablicu:

$\underline{x}$	(1, 4)	(1, 2)	(3, 3)	(4, 3)
$f(\underline{x})$	17	5	18	25

najlošija

$$\underline{x}_C = \left(\tfrac{5}{3}, 3\right)$$

$$\underline{x}_R = (-3, 3)$$

↳ zbog eksplicitnih: $x_R' = (1, 3)$

↳ implicitna su OK i x_R' nije najlošija, pa je iteracija gotova

↳ druga iteracija:

$$\underline{x}_C = (1, 3)$$

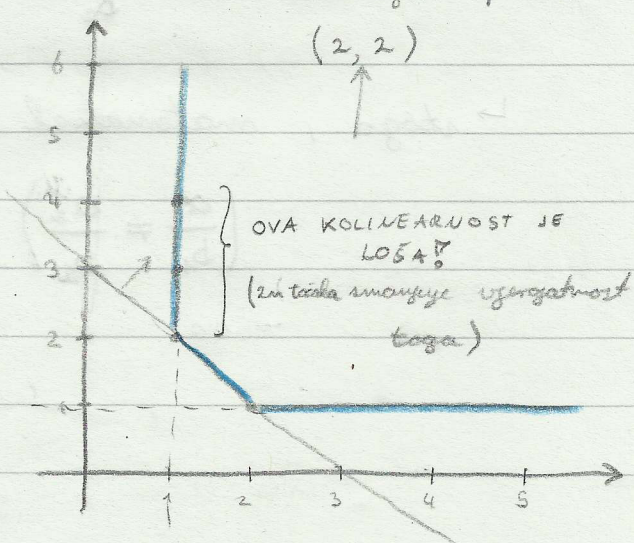
$$\underline{x}_R = (-3, 3)$$

⋮ ekspl.
⋮ impl.

$$\underline{x}_R' = (1, 3)$$

dobili smo isto što nije nužno loše, ali ovdje je, pa nacrtajmo graf:

zbog toga postupak konvergira prema (2, 2)



ZAD: Zadana je funkcija cilja $F(\underline{x})$ uz dva ograničenja:

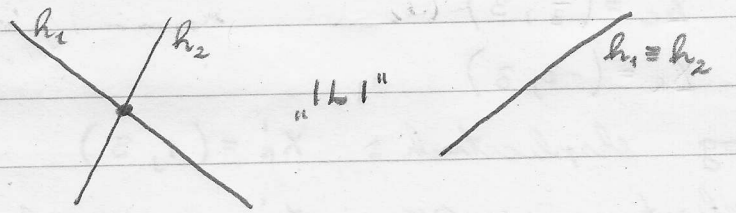
$$a_1 x_1 + b_1 x_2 = c_1$$

$$a_2 x_1 + b_2 x_2 = c_2$$

Navedi uvjet postojanja rješenja, definiraj novu funkciju cilja koja uzima ograničenja u obzir!

↳ pretpostavimo da su zadana ograničenja implicitna ograničenja u obliku jednakosti:

- ograničenja su pravci, pa ako se oni sijeku ili poklapaju, onda rješenje postoji:



↳ napišimo ograničenja malo lepše:

$$h_1: \quad x_2 = -\frac{a_1}{b_1} x_1 + \frac{c_1}{b_1}$$

$$h_2: \quad x_2 = -\frac{a_2}{b_2} x_1 + \frac{c_2}{b_2}$$

↳ stoga, matematički zapisano, rješenje postoji za:

$$\left(\frac{a_1}{b_1} \neq \frac{a_2}{b_2}\right) \quad \text{"ILI"} \quad \left(\frac{a_1}{b_1} = \frac{a_2}{b_2} \quad \text{"I"} \quad \frac{c_1}{b_1} = \frac{c_2}{b_2}\right)$$

↳ za drugi dio radatka treba definirat novu funkciju oblika:

- koristimo se „vanjskom početnom točkom" jer se ne nadamo da početna točka zadovoljava ograničenja:

$$U(\underline{x}, t) = F(\underline{x}) + t \cdot \left(h_1^2(\underline{x}) + h_2^2(\underline{x})\right)$$

$$U(\underline{x}, t) = F(\underline{x}) + t \cdot \left[(a_1 x_1 + b_1 x_2 - c_1)^2 + (a_2 x_1 + b_2 x_2 - c_2)^2\right]$$

ovo smo dobili tako da smo rekli: $h_i(\underline{x}) = 0$

RJEŠAVANJE SUSTAVA NELINEARNIH JEDNADŽBI

PR: Rješavati ćemo ovakve sustave:

$$x_1^2 + x_2^2 - 1 = 0$$

$$x_1 - 2x_2 = 0$$

⇒ uvedimo neke pojmove, oznake:

↳ oznaka pojedine jednadžbe: $g_i(\underline{x})$

↳ oznaka sustava: $\underline{G}(\underline{x}) = \underline{0}$

↳ za početak ćemo gledat sustave $n \times n$:

$$\underline{G}(\underline{x}) = \begin{bmatrix} g_1(\underline{x}) \\ g_2(\underline{x}) \\ \vdots \\ g_n(\underline{x}) \end{bmatrix}$$

⇒ IDEJA RJEŠAVANJA KOJA NAM PRVA PADNE NAPAMET:

↳ nadomjesna funkcija cilja:

$$F(\underline{x}) = \frac{1}{2} \sum g_i^2(\underline{x})$$

↓

nadalje tražimo minimum ove sume

↳ no ovo zna bit pre-kompleksno za rješit, pa trebamo drugi pristup

⇒ za bolje metode rješavanja, trebat ćemo matricu Jakobijan:

$$\underline{J} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix}$$

⇒ **GAUSS-NEWTON POSTUPAK:**

↳ ideja: razvijemo u Taylorov red oko početne točke x_0 i mičemo se u x za koj se nadamo da je minimum:

x_0 → početna točka

x ⇒ minimum?

$$g_i(\underline{x}) = g_i(\underline{x}_0) + \nabla^T g_i(\underline{x}_0) \cdot (\underline{x} - \underline{x}_0) + \ldots$$

⇓ uzmemo pojedinačne funkcije i stavimo u sustav

$$\underline{G}(\underline{x}) = \underline{G}(\underline{x}_0) + \underline{J}(x_0) \cdot (\underline{x} - \underline{x}_0)$$

↳ želimo da je ovo jednako nul: $\underline{G}(\underline{x}) = \underline{0}$

↳ zatim izrazimo x:

$$\underline{x} = \underline{x}_0 - \underline{J}^{-1}(x_0) \cdot \underline{G}(\underline{x}_0)$$

$\underline{x}$ je minimum ako je sustav linearan, a ako nije, ovo će biti iterativan postupak

OKRENI →

↳ iterativan dio Gauss-Newtonovog postupka:

$$\underline{X}_{k+1} = \underline{X}_k - \underline{J}^{-1}(\underline{X}_k) \cdot \underline{G}(\underline{X}_k)$$

↳ no J^{-1} je kompliciran, pa se pojednostavljuje sa sređivanjem:

$$\underline{\Delta X} = \underline{X}_{k+1} - \underline{X}_k$$

↳ stoga, stvar se rješava kao:

$$\boxed{\underline{J}(\underline{X}_k)\,\underline{\Delta X} = -\underline{G}(\underline{X}_k)}$$

⇒ **ALGORITAM:**

Gauss-Newton (x_0, ε)

- $\underline{X} = \underline{X}_0$
- PONAVLJAJ
  - izračunaj $\underline{G}(\underline{x})$, $\underline{J}(\underline{x})$;
  - riješi $\underline{J}(\underline{x})\,\underline{\Delta X} = -\underline{G}(\underline{x})$
  - $\underline{X} = \underline{X} + \underline{\Delta X}$
- DOK JE ($\underline{\Delta X} > \varepsilon$)

PR: Riješ dan sustav Gauss-Newtonovim postupkom:

$$X_1^2 + X_2^2 - 1 = 0 \quad \Rightarrow \quad g_1(\underline{X})$$

$$X_1 - 2X_2 = 0 \quad \Rightarrow \quad g_2(\underline{X})$$

Početna točka je $\underline{X}_0 = (1, 1)$!

↳ izračunajmo Jakobijan matricu:

$$\underline{J} = \begin{bmatrix} 2X_1 & 2X_2 \\ 1 & -2 \end{bmatrix}$$

primjeti da je sustav linearan, unutra bi bile konstante

↳ rješavamo: $\underline{J} \cdot \underline{\Delta X} = -G$

↳ za X_0:

$$\begin{bmatrix} 2 & -2 \\ 1 & -2 \end{bmatrix} \underline{\Delta X} = \underbrace{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}_{\text{„njera nezadovoljenja"}}$$

⇓ dobivamo:

$$\Delta X_1 = 0 \qquad \Delta X_2 = -\frac{1}{2}$$

↳ nova iteracija je za $\underline{X}_1 = \underline{X}_0 + \underline{\Delta X} = (1, \frac{1}{2})$

↳ za X_1:

$$\begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix} \underline{\Delta X} = \underbrace{\begin{bmatrix} -\frac{1}{4} \\ 0 \end{bmatrix}}_{\text{njera nezadovoljenja je manja!}}$$

⇓ dobivamo:

$$\Delta X_1 = -\frac{1}{10} \qquad \Delta X_2 = -\frac{1}{20}$$

↳ daljnje iteracije: $\underline{X}_2 = (0.9, 0.45)$ $\quad G(\underline{X}_2) = [0.0125 \quad 0]^T$

$\underline{X}_3 = (0.894, 0.4472)$ $\quad G(\underline{X}_3) = [3 \cdot 10^{-5} \quad 0]^T$

↳ za vježbu možemo provesti isti postupak, al iz početne točke $\underline{x}_0 = (-1, 0)$ za isti sustav

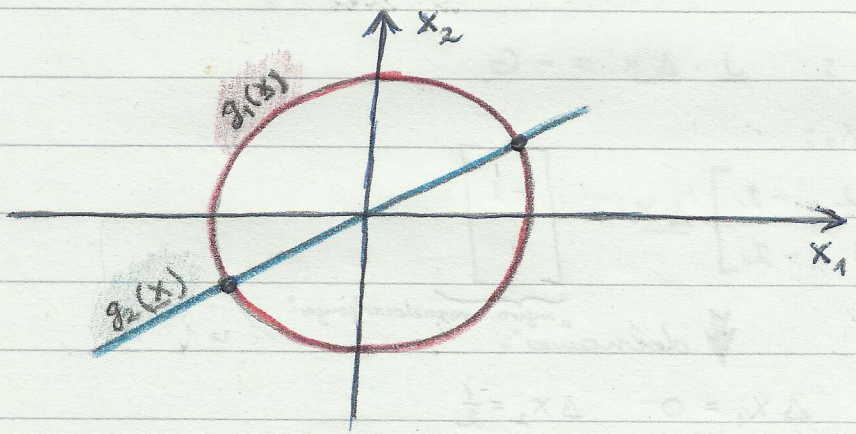
↳ dobivamo:

$$\underline{x}_1 = (-1, -\tfrac{1}{2})$$

$$\underline{x}_2 = (-0.9, -0.45)$$

⬇

slične, al suprotne točke jer se sada gibamo prema drugom rješenju (početni sustav ima dva rješenja)



↳ stoga, optimizacija koju u ovoj skripti radimo nije unimodalna, al su srećom oba rješenja jednako dobra

PR: Pogledajmo Rosenbrockovu funkciju:

$$f(\underline{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

↳ nju želimo pikazat kao sumu parcijalnih funkcija kako bi je mogli rješit Gauss-Newtonovom postupkom:

↳ parcijalne funkcije su kvadratne:

$$f_1(\underline{x}) = 10(x_2 - x_1^2)$$

$$f_2(\underline{x}) = 1 - x_1$$

↳ prvo, nađemo Jakobijan matricu:

$$\underline{J} = \begin{bmatrix} -20x_1 & 10 \\ -1 & 0 \end{bmatrix}$$

↳ krenemo s početnom točkom $\underline{x}_0 = (0, 0)$:

$$\underline{J} \cdot \Delta\underline{x} = -\underline{G}$$

$$\begin{bmatrix} 0 & 10 \\ -1 & 0 \end{bmatrix} \Delta\underline{x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \Rightarrow \begin{matrix} \Delta x_1 = 1 \\ \Delta x_2 = 0 \end{matrix}$$

$$\underline{x}_1 = \underline{x}_0 + \Delta\underline{x} = (1, 0)$$

↳ druga iteracija za x_1:

$$\begin{bmatrix} -20 & 10 \\ -1 & 0 \end{bmatrix} \Delta\underline{x} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \Rightarrow \begin{matrix} \Delta x_1 = 0 \\ \Delta x_2 = 1 \end{matrix}$$

$$\underline{x}_2 = \underline{x}_1 + \Delta\underline{x} = (1, 1)$$

(puno bolje od spusta po gradijentu il po osima)

↳ u dvije iteracije smo našli minimum, no trebamo znati rastavit početnu funkciju!

PRIMJENA GAUSS-NEWTONOVOG POSTUPKA NA OPĆENITE SUSTAVE

⇒ općeniti sustavi nemaju jednak broj varijabli i jednadžbi:

- broj varijabli: „n"
- broj jednadžbi: „m"

⇒ da bi bilo rješivo, mora biti ispunjeno:

$$m \geq n$$

⇒ matrica Jakobijan:

$$\underline{J}_{m \times n} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix}$$

⇒ postupak svodimo na prethodno pokazani za $n \times n$:

$$\boxed{\underline{J}^T \cdot \underline{J} \cdot \underline{\Delta x} = -\underline{J}^T \underline{G}}$$

⇒ primjena ovog postupka:

↳ interpolacija niza točaka po modelu (LEAST SQARES)

- imamo radarske točke:

(t_i, Y_i): $(1,2), (0,3), (-1,2), \ldots$

- npr. uzmemo sljedeći model:

$$Y = f(\underline{X}) = X_1 t^2 + X_2 t + X_3$$

- dobijemo sustav jednadžbi za svaku točku:

$$\begin{cases} X_1 t_i^2 + X_2 t_i + X_3 - Y_i = 0 \end{cases}$$

⬇

ovo radi GPS za određivanje pozicije ☺

PR: Riješi raden sustav u početnu točku $x_0 = 3$?

$$x + 1 = 0$$
$$2x^2 + x - 1 = 0$$

↳ računamo Jakobijan matricu:

$$\underline{J} = \begin{bmatrix} 1 \\ 4x + 1 \end{bmatrix}$$

↳ za početnu točku x_0: $\underline{J}(\underline{x}_0) = \begin{bmatrix} 1 \\ 13 \end{bmatrix}$; $\underline{G}(\underline{x}_0) = \begin{bmatrix} 4 \\ 20 \end{bmatrix}$

$$\underline{J}^T \underline{J} \cdot \Delta x = -\underline{J}^T \cdot \underline{G}$$

$$\begin{bmatrix} 1 & 13 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 13 \end{bmatrix} \cdot \Delta x = -\begin{bmatrix} 1 & 13 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 20 \end{bmatrix}$$

$$\Delta x = -1.553$$

↳ dobivamo sljedeću točku: $x_1 = x_0 + \Delta x = 1.447$

↳ sljedeće iteracije nas vode na 0.25:

$(x_2 = 0.727,\ x_3 = 0.432,\ x_4 = 0.325, \dots) \Rightarrow 0.25$

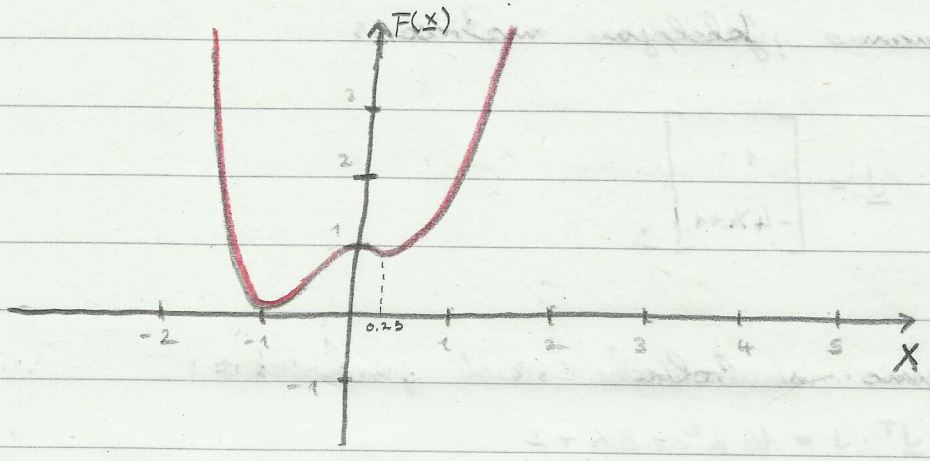
↳ da smo krenuli od početne točke $x_0 = -2$, ravnali bi u točku -1:

↳ rješto je tome tako?

↳ pa, pogledajmo nadomjesnu funkciju:

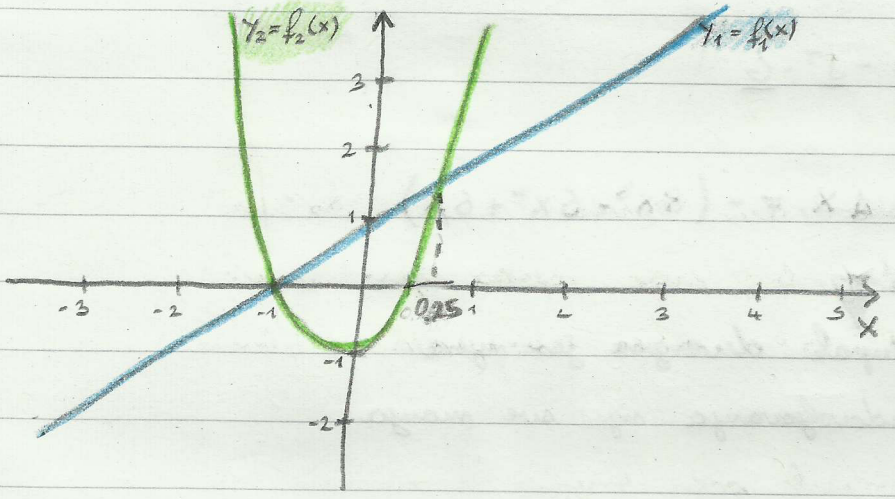
$$F(x) = \frac{1}{2}(x+1)^2 + \frac{1}{2}(2x^2+x-1)^2$$

↳ ona izgleda ovako:



↳ pomaže i drugačija vizualizacija:

• prikaz svake funkcije posebno:



PR: Pogledajmo malo modificirani prethodni primjer:

$$x + 1 = 0$$

$$-2x^2 + x - 1 = 0$$

↳ izračunamo Jakobijan matricu:

$$\underline{J} = \begin{bmatrix} 1 \\ -4x+1 \end{bmatrix}$$

↳ računajmo simbolički radi prikaza:

$$\underline{J}^T \cdot J = 16x^2 - 8x + 2$$

$$\underline{J}^T \cdot \underline{G} = 8x^3 - 6x^2 + 6x$$

$$x_{k+1} = x_k - \frac{8x^3 - 6x^2 + 6x}{16x^2 - 8x + 2}$$

$$\underline{J}^T \cdot \underline{J} \cdot \Delta x = -\underline{J}^T \cdot \underline{G}$$

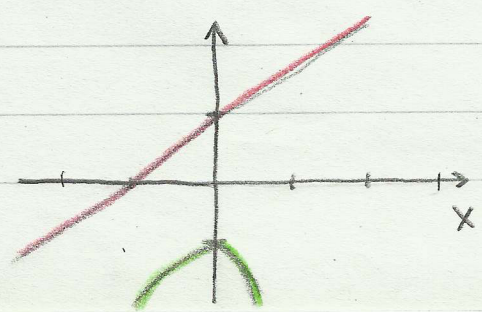
$$(16x^2 - 8x + 2)\,\Delta x = -(8x^3 - 6x^2 + 6x)$$

$\Downarrow$

postupak divergira jer nijera nezadovoljavanja nije sve manja

$x_0 = 0.1 \Rightarrow -0.303,\ 0.197,\ -0.468, \ldots$

$x_0 = 1 \Rightarrow 0.2,\ 0.985,\ 0.093,\ -0.275, \ldots$



funkcije se ne sjeku, pa je nemoguće zadovoljiti obje

⇒ pretpostavka da Gauss-Newtonov postupak ne garantira konvergenciju osim ako je Jakobijan punog ranga u svakom koraku

↳ to nije teorijska pretpostavka, nego se pristupa drugačije:

- korak algoritma $\underline{x}_{k+1} = \underline{x}_k + \underline{\Delta x}$ se mijenja pronalaženjem minimuma na pravcu u smjeru vektora $\underline{\Delta x}$

⇓

takav pristup uvelike poboljšava performanse algoritma, no još uvijek ne daje potpunu garanciju konvergencije

(stoga, konstrukt ćemo malo popravljenu i unaprijeđenu verziju ovog postupka)

POSTUPAK LEVENBERG - MARQUARDT

⇒ Gauss-Newton postupak divergira ako nadomjesna funkcija nije bljepa kvadratna nego je nekto drugo kompliciranije

⇒ IDEJA:

- gledamo nadomjesnu funkciju: $F(\underline{x}) = \frac{1}{2}\sum_{m} g_i^2(\underline{x})$
- nadomjesnu funkciju rješavamo najbržim spustom
  ↳ najbrži spust može biti spor, al je siguran (konvergira)

$$\frac{\partial F}{\partial x_j} = \sum_{i=1}^{m} g_i(\underline{x}) \cdot \frac{\partial g_i}{\partial x_j} = \underline{J}_j^T \cdot \underline{G}(\underline{x})$$

↓ po jednoj varijabli

ovo je samo jedan stupac Jakobijan matrice

↳ stoga, derivacija po svim varijablama (gradijent) je:

$$\nabla F = \underline{J}^T \cdot \underline{G}$$

⇓

smjer najbržeg spusta:

↳ označimo $\underline{v} = -\underline{J}^T \cdot \underline{G}$

⇒ uspoređimo sada najbrži spust i Gauss-Newton postupak:

$$\underline{\Delta x} = -\underline{J}^T \cdot \underline{G}$$

$$\underbrace{(\underline{J}^T \cdot \underline{J})}_{\underline{A} = \underline{J}^T \underline{J}} \cdot \underline{\Delta x} = -\underline{J}^T \cdot \underline{G}$$

⇒ konačna formula postupka:

$$\boxed{(\underline{A} + \mu \cdot \underline{I}) \cdot \underline{\Delta x} = -\underline{J}^T \cdot \underline{G}}$$

$\mu \rightarrow$ faktor prigušenja

$\mu \approx 0 \Rightarrow$ radimo Gauss-Newton postupak

$\mu >> 0 \Rightarrow$ radimo najbrži spust

↳ **IDEJA:** kada smo daleko od minimuma, radi najbrži spust (to je na početku), a kada smo blizu minimuma, nadamo se da se funkcija ponaša kvadratno, pa će Gauss-Newton postupak puno bže konvergirati od najbržeg spusta

⇒ prijedlog za početni „μ": $\mu_0 = \max\{a_{ii}\}$

⇒ **ALGORITAM:**

Levenberg - Marquardt ($\underline{x}_0$, ε)

- $\underline{x} = \underline{x}_0$; $\nu = 2$; $\mu = \max\{a_{ii}\}$;
- PONAVLJAJ
  - riješi $(\underline{J}^T \cdot \underline{J} + \mu I)\,\Delta\underline{x} = -\underline{J}^T \cdot \underline{G}$
  - $\underline{x}' = \underline{x} + \Delta\underline{x}$
  - AKO $F(\underline{x}') < F(\underline{x})$
    - $\underline{x} = \underline{x}'$;
    - $\mu \mathrel{/}= \nu$;
  - INAČE
    - $\mu = \mu * \nu$
- DOK JE ($|\Delta\underline{x}| > \varepsilon$)

⇒ na ovaj postupak, za svaki $\mu > 0$ je matrica sustava pozitivno definitna što znači da sustav ima rješenje koje daje smanjenje funkcije cilja

↳ garancija konvergencije ∇!

⇒ primjer da nije potrebno tražiti minimum na pravcu ∇!

PR: Napravi jednu iteraciju Levenberg-Marquardt postupka na Rosenbrockovoj banani funkciji!

$X_0 = (0, 0)$

$$\underline{J}(X_0) = \begin{bmatrix} 0 & 10 \\ -1 & 0 \end{bmatrix} \Rightarrow \underline{J}^T\underline{J} = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix}$$

$$\mu_0 = \max\{a_{ii}\} = 100$$

$$\underline{G}(\underline{x}_0) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \quad \underline{J}^T\underline{G} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

↳ rješimo glavnu jednadžbu postupka:

$$(\underline{J}^T\underline{J} + \mu I)\,\Delta\underline{x} = -\underline{J}^T\underline{G}$$

$$\begin{bmatrix} 101 & 0 \\ 0 & 200 \end{bmatrix} \Delta\underline{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \Delta\underline{x} = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}$$

$$\underline{x}' = \underline{x}_0 + \Delta\underline{x} = (0.01 \quad 0)$$

$F(\underline{x}') < F(\underline{x}) \Rightarrow$ ustvno, pa se korak prihvaća te se „μ" dijeli sa dva:

$\mu = 50$

>>> KRAJ PRVE ITERACIJE <<<

GENETSKI ALGORITMI

⇒ radi cemo pogledati višemodalne funkcije za koje nemamo garancije pronalaska globalnog minimuma

⇒ <u>možemo postupiti onako:</u>

a) random pokrenuti deterministicke postupke (naučene do sada) iz različitih početnih točaka

b) koristimo stohastičke algoritme (npr. genetski algoritmi)

⇒ opcenit evolucijski algoritam:

EVOLUCIJSKI ALGORITAM
- STVORI POPULACIJU
- PONAVLJAJ
  - PRIMJENI GENETSKE OPERATORE
  - ODABERI NOVU POPULACIJU
- DOK (NIJE DOVOLJNO DOBRO)

⇒ **FUNKCIJA DOBROTE:**

↳ oznaka: F

↳ ne mora biti jednaka funkciji cilja „f"

↳ općenit pristup:

WINDOWING:

↳ preslikavanje u radnom intervalu

$$F_i = a + (b-a) \cdot \frac{f_i - f_{worst}}{f_{best} - f_{worst}}$$

⇓

najbolja jedinka je sada vrijednosti „a" (na numeraciji), a najgora je tada vrijednost „b"

↳ time je dobrota u intervalu $[a, b]$

⇒ **PRIKAZ RJEŠENJA:**

↳ kako imamo ograničen prikaz brojeva u računalu, trebamo se ograničiti na određeni prostor pretraživanja (stoga, obično imamo definirana **eksplicitna ograničenja** $[dg, gg]$)

a) **BINARNI PRIKAZ**

$x \rightarrow$ domena

$b \rightarrow$ binarna vrijednost

$n \rightarrow$ broj bitova

KROMOSOM	b	x
000...00	$\emptyset$	dg
111...11	$2^n - 1$	gg

↳ preslikavanje iz binarne u realnu domenu je definirano sljedećom formulom:

$$x = dg + \frac{b}{2^n - 1}(gg - dg)$$

↳ preslikavanje iz realne u binarnu domenu je definirano sljedećom formulom:

$$b = \frac{x - dg}{gg - dg}(2^n - 1)$$

↳ kako izračunati broj bitova za binarni prikaz?

- pa, to ovisi o željenoj preciznosti, a mi uvijek želimo što manju preciznost koja zadovoljava naša očekivanja kako bi smanjili prostor pretraživanja:

$$n \geq \frac{\log \lfloor 1 + (gg - dg) \cdot 10^r \rfloor}{\log 2}$$

⇓

"r" ⇒ BROJ PRECIZNIH DECIMALNIH MJESTA

intuitivno si ovo objasni na sljedećim primjerima:

$$\left.\begin{array}{l} dg = -100 \\ gg = 100 \\ r = 2 \end{array}\right\} n = 15$$

$$\left.\begin{array}{l} dg = 0 \\ gg = 100 \\ r = 1 \end{array}\right\} n = 10$$

b) PRIKAZ GRAYEVIM KODOM

↳ motivacija je da udaljenost kromosoma bude jednaka u problemskoj (realnoj) i genotipskoj (binarnoj) domeni

c) PRIKAZ KAO VEKTOR REALNIH BROJEVA

↳ prevladavaju danas

↳ brži nego što nema binarnog kodiranja i dekodiranja

⇒ **POČETNA POPULACIJA:**

↳ najčešće, uzimamo neko slučajno rješenje

↳ ako ne možemo dobit neko rješenje koje zadovoljava ograničenja, treba preispitat odabir genetskog algoritma

⇒ **SELEKCIJA:** (odabir)

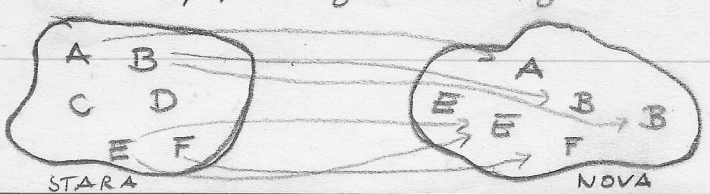
↳ svaka jedinka ima vjerojatnost preživljavanja koja je proporcionalna njenoj dobroti

↳ obično imamo konstantnu veličinu populacije (N)

↳ dva načina selekcije:

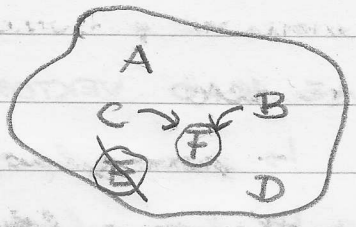
a) GENERACIJSKI

↳ na temelju trenutne populacije, gradimo novu populaciju (u skrivenom memorijskom prostoru)



b) ELIMINACIJSKI

↳ briramo one koje eliminiramo i onoga tko će ramijenit onog tko će ramijenit eliminiranog



↳ ovo su bil tipovi postupka selekcije

↳ OPERATORI SELEKCIJE/ODABIRA:

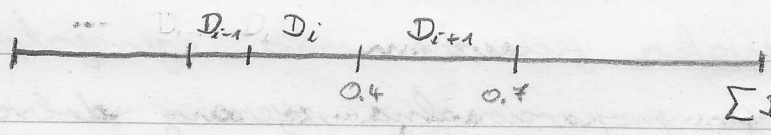
• operator koj bira jednu jedinku iz postojeće populacije na temelju dobrote jedinki

↳ pogledajmo neke primjere:

1) jednostavni generacijski odabir

↳ generacijskog tipa

↳ „roulette-wheel selection"



⇓

generiraj slučajan broj iz [0,1], odaberi jedan kromosom za sljedeću generaciju (ponavlja se N puta, jer treba popuniti cijelu generaciju)

2) jednostavan eliminacijski odabir

↳ eliminacijskog tipa

↳ „roulette-wheel selection" onoga koji se eliminira, no onda treba revidirati brojevni pravac jer smo jednu jedinku izbacili

↳ ponavljamo dok nismo izbacili onoliko koliko smo htjeli

3) turnirski odabir

↳ koristi se OPERATOR TURNIRA (k)

↳ odabere se slučajno k-jedinki i uzme se najbolja ili najgora (ovisno o tipu, je li eliminacijski ili generacijski)

↳ preporuka: koristi eliminacijski tip uz 3-turnirski odabir jer koristi istu memoriju, a ne treba dvije populacije čuvati u isto vrijeme, a ima inherentno ugrađen elitizam

↳ elitizam je svojstvo da se najbolja jedinka sigurno prebacuje u novu populaciju

⇒ KRIŽANJE:

↳ dvije jedinke su roditelji

↳ jedna nova jedinka je dijete

↳ parametar:

- vjerojatnost križanja (p_c)
  - ↳ obično oko 70% ili 80%

a) kod binarnog prikaza

1) križanje s jednom točkom prekida

R1: 001 | 0110

R2: 011 | 1010

D: 001 | 1010

↳ može biti spor u istraživanju velikog prostora pretraživanja

2) križanje s dvije ili više točaka prekida

R1: 10 | 101 | 01

R2: 01 | 010 | 10

D: 10 | 010 | 01

3) uniformno križanje

↳ generira slučajan vektor bitova

↳ tamo gdje su roditelji isti, uzmi tu vrijednost, a tamo gdje su različiti, uzmi vrijednost iz slučajnog vektora

R1: 0 1 0 1 0 1

R2: 1 1 1 1 0 0

S: 1 1 1 0 0 0

D: 1 1 1 1 0 0

$$D = AB + R(A \oplus B)$$

b) kod prikaza s pomičnim zarezom

1) aritmetičko križanje

↳ dijete je linearna kombinacija dva roditelja

$$D = a \cdot R_1 + (1-a) \cdot R_2$$

$$a \in [0, 1]$$

2) heurističko križanje

↳ koristi se informacija o tome koj je roditelj bolji

$$D = a(R_2 - R_1) + R_2$$

$$a \in [0, 1]$$

$R_2 \rightarrow$ bolje rješenje

↳ treba paziti da dijete zadovoljava ograničenja

⇒ MUTACIJA:

↳ služi za izbjegavanje lokalnih optimuma i obnavljanje izgubljenog genetskog materijala (npr. ako sve jedinke imaju jedinicu na 42. mjestu, a križanjem se to ne može promijeniti - tada smo izgubili jednu dimenziju i smanjili prostor pretraživanja na pola što nam je pre-radikalno)

a) kod binarnog prikaza

• parametri:

a) vjerojatnost permutacije jednog bita $\Rightarrow p_m$

b) vjerojatnost mutacije jedinke $\Rightarrow p_M$

$$1-(1-p_m)^m \approx p_M$$

1) jednostavna mutacija

↳ s nekom vjerojatnošću „flippaj" jedan bit

↳ stoga, radaje se „p_m"

b) kod prikaza s pomičnim zarezom

↳ tu ima jedino smisla pričati o vjerojatnosti mutacije jedne jedinke p_m

1) Gaussovska mutacija

↳ izaberi neku vrijednost iz Gaussove razdiobe sa središtem u jedinki koja se mutira

⇒ UVJET ZAUSTAVLJANJA:

↳ imamo nekoliko načina:

- broj iteracija (generacija)
- broj evaluacija funkcije cilja ⇒ najbolje?
- dostignuta vrijednost funkcije cilja
- vremensko ograničenje
- broj iteracija bez poboljšanja

ZAD: Genetskim algoritmom optimira se funkcija dvije varijable gdje je prva varijabla u intervalu $[-5, 5]$, a druga $[0, 1]$, a željena preciznost je na dvije decimale. Koliko je bitova potrebno za obje varijable u binarnom prikazu i koji je broj mogućih rješenja? Najbliže jedinke koje najbolje predstavljaju točke $(-2, 0.2)$ i $(0, 0.99)$!? Provedi križanje (jednolike) na slučajni kromosom kao na jedinica i dekodiraj rezultat!? Ako je $p_{mutacije} = 0.01$ za bit, koja je vjerojatnost da će biti barem jedan bit u kromosomu biti mutiran?

a) $X_1 \Rightarrow$ min. 1000 vrijednosti $\Rightarrow n_1 = 10$

$X_2 \Rightarrow$ min. 100 vrijednosti $\Rightarrow n_2 = 7$

↳ broj mogućih rješenja je 2^{17} jer imamo ukupno 17 bitova za željenu preciznost //

b) $X \rightarrow b \Rightarrow$ formula: $b = \frac{x - d_g}{g_g - d_g}(2^n - 1)$

prvi kromosom

$X_1 = -2 \Rightarrow b_1 = \frac{-2+5}{10}(2^{10} - 1) = 306.9$

↳ najbolje moguće dobivena do zaokruživanja na 307

$X_2 = 0.2 \Rightarrow b_2 = \frac{0.2 - 0}{1}(2^7 - 1) = 25.4$

↳ najbolje moguće dobivena do zaokruživanja na 25

drugi kromosom

$x_1 = 0 \Rightarrow b_1 = 511.5$

↳ zaokruživanje: 512

$x_2 = 0.99 \Rightarrow b_2 = 125.7$

↳ zaokruživanje: 126

↳ binarni prikaz kromosoma:

$(-2, 0.2) \Rightarrow 0100110011 \,|\, 00110001$

$(0, 0.99) \Rightarrow 1000000000 \,|\, 11111110$

c) radimo križanje nad gornjim rezultatom:

↳ slučajni kromosom su sve jedinice, pa tamo gdje se roditelji razlikuju, uvrsti jedinicu:

DIJETE: $1100110011 \,|\, 11111111$

↳ dekodiranje djeteta:

$b_1 = 819$ ⇓ $x_1 = 3.006$

$b_2 = 127$ ⇓ $x_2 = 1$

d) $p_m = 0.01$

↳ kako je šetanje po svima neefikasno za veće jedinke, dobro je izračunati vjerojatnost mutacije jedinke, umjesto da šećemo po svim genima

$$p_M = 1 - (1 - 0.01)^m$$

$$p_M = 1 - (1 - 0.01)^{17} = 15.7\%$$

ANALIZA PRIJELAZNIH POJAVA

⇒ na temelju stanja sustava i na temelju fizikalnih zakonitosti pokušavamo izračunati (predvidjeti) ponašanje sustava

↳ pokušavamo predvidjeti budućnost sustava!

⇒ OPIS STANJA SUSTAVA:

↳ temelji se na varijablama stanja

(npr. na opis jabuke koja pada sa stabla trebamo dvije varijable stanja: visina i brzina)

↳ varijable stanja obično opisuju količinu pohranjene energije u sustavu

(te varijable opisuju memorijske elemente sustava)

⇒ PROMJENE STANJA SUSTAVA:

↳ promjene varijabli stanja u vremenu opisane su diferencijalnim jednadžbama

⇓

tu je problem jer se one rješavaju integriranjem, a to nije uvijek moguće analitički, pa je srž problema rješavanje NUMERIČKA INTEGRACIJA

⇒ NAPOMENA:

- kod sustava gdje imamo diferencijalne jednadžbe višeg reda (drugog il više), uvijek te jednadžbe možemo prebacit u sustav diferencijalnih jednadžbi prvog reda

⇒ za ilustraciju, pogledaj matematičko njihalo iz skripte i opis njegovog sustava

↳ npr. za njihalo s prigušenjem, postupak opisa sustava i pretvorba u sustav diferencijalnih jednadžbi bi izgledao ovako:

$$G \cdot \varphi = -m \ddot{\varphi} L - k \cdot \dot{\varphi}$$

$$\ddot{\varphi} + \underbrace{\frac{k}{m \cdot L}}_{K} \dot{\varphi} + \frac{g}{L} \cdot \varphi = 0$$

$$\left.\begin{aligned} X_1 &= \varphi \\ \dot{X}_1 &= X_2 \\ \dot{X}_2 &= \ddot{\varphi} \end{aligned}\right\} \quad \left.\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= -K X_2 - \frac{g}{L} \cdot X_1 \end{aligned}\right\}$$

primjet da s lijeve strane imamo prve derivacije

iz toga dobivamo sljedeći sustav:

$$\underline{\dot{X}} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & -\frac{g}{L} \cdot X_1 \end{bmatrix}$$

⇒ OSNOVNI POJMOVI:

$\underline{X}$ → varijabla stanja

$\underline{X}_0, t_0$ → početni uvjet

$\dot{\underline{X}} = f(\underline{X}, t)$ → prva derivacija

} ZADANO!

$\underline{X}(t) = ?$

} OVO SE TRAŽI!

↳ do rješenja se dolazi integriranjem, no budući da analitički to u općem slučaju nije moguće, radimo numeričku integraciju

⇒ mi ćemo se većinom baviti linearnim sustavima prvog reda:

$$\dot{\underline{X}} = \underline{A}\,\underline{x} + \underline{B}\,\underline{r}(t)$$

$\underline{A}$ → matrica sustava (kvadratna)

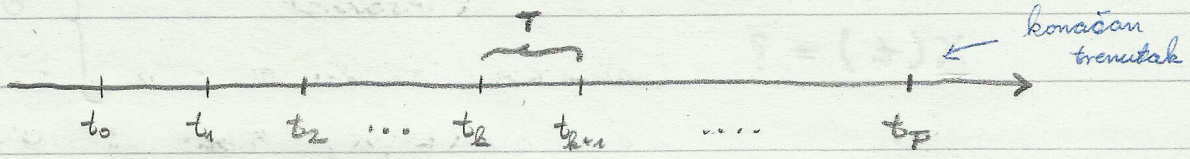
$\underline{r}$ → vanjska pobuda

$\underline{B}$ → općenita matrica „raširena" za pobudu

⇒ POSTUPAK RJEŠAVANJA:

↳ pretpostavljamo radnu početnu vrijednost $X(t_0)$

↳ trebamo diskretizirat domenu jer ne možemo uzet beskonačno mnogo točaka za numeričku integraciju



T ⇒ period integracije (obično konstantan, ali ne mora biti)

↳ kada znamo „X_k", da bi dobili „X_{k+1}" možemo koristit „X_k"

↳ možemo koristit i proizvoljno mnogo prethodnih koraka:

$$X_{k+1} = F(\underbrace{X_k, X_{k-1}, \dots, X_{k-p+1}}_{p\text{-koraka}})$$

↳ KORAČNOST POSTUPKA:

- koliko se prethodnih koraka koristi za izračunavanje sljedeće vrijednosti

⇒ ODABIR POSTUPKA:

↳ ovisi o:

- količini računanja (složenosti)
- točnosti / preciznosti
- numeričkoj stabilnosti

(stabilni postupci imaju svojstvo da s povećanjem broja koraka ne divergiraju, nego se ili približavaju analitičkom rješenju ili ostaju na istoj udaljenosti)

⇓

numerička stabilnost je najbitnije svojstvo i moramo ga znati provjeriti za svaki postupak!

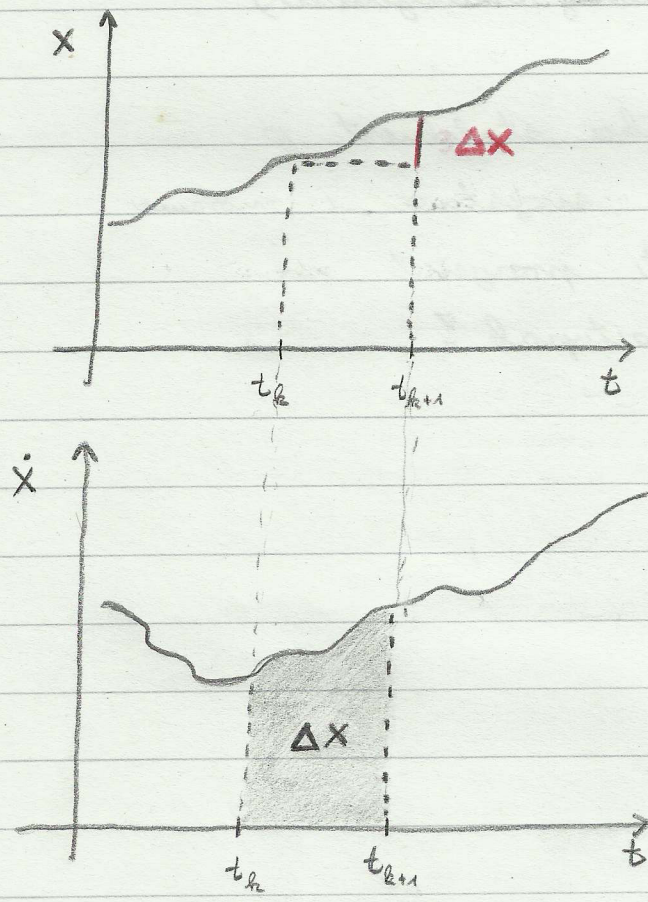
EULEROV POSTUPAK

⇒ jednostavan, ali nije jako točan (dovoljno dobar za studentske radove)

⇒ ZADANO: a) $\dot{X} = f(X, t)$ (npr. $\dot{X} = \frac{X}{5} + \sin X + \frac{t}{1}$)

b) period integracije T

↳ poslijedično tome znamo: $X(t_k)$ i $\dot{X}(t_k)$

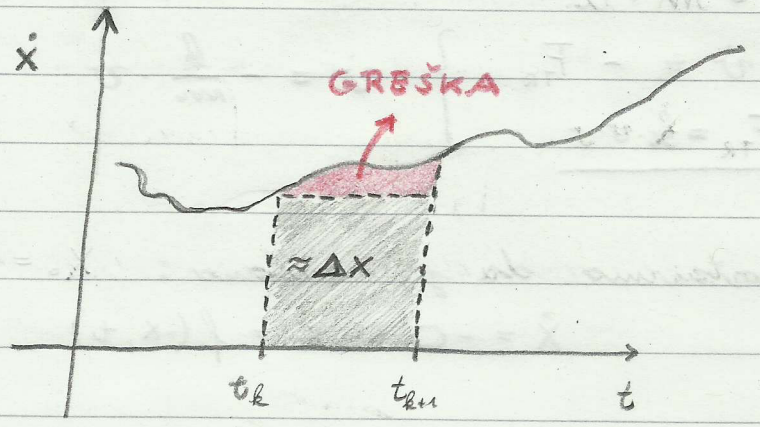


znat da ∫ ora površinu jednaka ΔX, pa kada bi znali derivaciju, znali bi i promjenu varijable

↳ stoga, ako nam neka čapne derivaciju, možemo dobit promjenu varijable!

↳ kako ne možemo dovnat tu derivaciju, radimo aproksimaciju konstantom:

· pretpostavimo da se derivacija nije mijenjala od „t_k" do „t_{k+1}":



↳ ako znamo ovo, tada je:

$$\Delta X = \dot{X}_k \cdot T \quad \Rightarrow \text{time znamo } X_{k+1}$$

$$X_{k+1} = X_k + \Delta X$$

$$\boxed{X_{k+1} = X_k + \dot{X}_k \cdot T}$$

$$X_{k+1} = X_k + f(x_k, t_k) \cdot T$$

↳ sada izračunamo prvu: $\dot{X}_{k+1} = f(X_{k+1})$

Pr: Tijelo usporava zbog trenja kretanjem kroz viskozni tekućinu u beztežinskom prostoru (sila trenja je proporcionalna brzini)

$$F = m \cdot a$$

$$\left.\begin{aligned} m \cdot \dot{v} &= -F_{TR} \\ F_{TR} &= k \cdot v \end{aligned}\right\} \quad \dot{v} = -\frac{k}{m} \cdot v$$

↳ pretpostavimo da je zadano: $X_0 = 1$

$$\dot{X} = -0.1 \cdot X = f(x,t)$$

↳ rješimo analitički za početak:

$$\frac{dx}{dt} = -0.1\,x$$

$$\frac{dx}{x} = -0.1\,dt \;/\; \int$$

$$\ln x = -0.1\,t$$

$$x = e^{-0.1t}$$

$$\Downarrow$$

$$X(t) = X_0\, e^{-0.1(t-t_0)}$$

↳ *projektirano sada na Eulerov postupak:*

$$\dot{x} = -0.1x$$

$$x_{k+1} = x_k + T \cdot \dot{x}_k = x_k + T(-0.1 \cdot x_k)$$

↳ *sada trebamo ovo izračunati, a da bi vidjeli kako se rezultat ponaša, izračunajmo to za različite ponuke:*

x_k	ANALITIČKI: $e^{-0.1t}$	EULER: $T=1$	EULER: $T=1.5$	EULER: $T=2.0$	EULER $T=2.5$
x_0	1	1	1	1	1
x_1	0.3048	0.9	-0.5	-1	-1.5
x_2	0.8187	0.81	0.25	1	2.25
x_3	0.7408	0.729	-0.125	-1	-3.375
x_4	0.6703	0.6561	-0.0625	1	5.0625
⋮	⋮	⋮	⋮	⋮	⋮
x_{30}	0.0498	0.0424	⋮	⋮	DIVERGIRA

stabilan postupak
(KONVERGENCIJA ili NE-DIVERGENCIJA)

↳ *upoznaj se sa stabilnosti Eulerovog postupka na sljedeće dvije stranice!*

$$\lambda = \sigma + j\omega = -0.1 + j \cdot 0 = -0.1$$

$$T \leq -\frac{2 \cdot (-0.1)}{0.01}$$

$$T \leq 20$$

⇒ ODREĐIVANJE STABILNOSTI:

↳ radi se pomoću ispitne jednadžbe:

$$\dot{x} = \lambda \cdot x$$

$\lambda \Rightarrow$ kompleksna konstanta koja predstavlja svojstvenu vrijednost sustava

↳ primjer do ispitne jednadžbe zapravo maksimalno pojednostavljuje sustav (linearizira ga!)

↳ objasnimo kako nam ispitna jednadžba pomaže:

- u svakom koraku radimo neku grešku δ_{k+1} pa ako ona raste kroz korake, tada postupak divergira:

$$|\delta_{k+1}| > |\delta_k|$$

↓

ako je ovo istina, postupak divergira!

⇒ odredimo koliko smijemo povećati „T" na Eulerov postupak da bi postupak ostao stabilan:

$$x_{k+1} = x_k + T(\lambda \cdot x_k)$$

⇒ uzjed: $x_k + \delta_k$; $x_{k+1} + \delta_{k+1}$ } rad se greška

$$x_{k+1} + \delta_{k+1} = (1 + \lambda T)(x_k + \delta_k)$$

$$\delta_{k+1} = (1 + \lambda T)\,\delta_k \quad / \,|\cdot|$$

$$|\delta_{k+1}| = |1 + \lambda T| \cdot |\delta_k|$$

↳ uzjet stabilnost: $|\delta_{k+1}| \leq |\delta_k|$

$$|1 + \lambda T| \leq 1$$

↳ ranima nas T, pa rospisujemo:

$$\lambda = \sigma + j\omega$$

$$|1 + \sigma T + j\omega T| \leq 1$$

$$(1 + \sigma T)^2 + (\omega T)^2 \leq 1$$

$$\vdots$$

$$\boxed{T \leq -\frac{2\sigma}{|\lambda|^2}}$$

stabilnost na Eulerov postupak

PONAVLJANJE

⇒ što treba znati za ispit od formula postupaka:

a) EULEROVA METODA:

$$x_{k+1} = x_k + T \cdot \dot{x}_k = x_k + T \cdot \underbrace{f(x_k, t_k)}_{\dot{x}_k}$$

↳ uvjet stabilnost:

$$|1 + \lambda T| \leq 1 \quad \Rightarrow \quad T \leq -\frac{2\sigma}{|\lambda|^2}$$

b) OBRNUTA EULEROVA METODA:

$$x_{k+1} = x_k + T \cdot \dot{x}_k$$

↳ uvjet stabilnost:

$$|1 - \lambda T| \leq 1 \quad \Rightarrow \quad T \geq -\frac{2\sigma}{|\lambda|^2}$$

⇓

implicitan postupak, pa ima puno veće područje stab-lnost

c) TRAPEZNI POSTUPAK:

$$x_{k+1} = x_k + \frac{T}{2}\left(\dot{x}_k + \dot{x}_{k+1}\right)$$

⇓

postupak drugog reda

↳ uvjet stabilnost:

$$\underbrace{\left(\left|\frac{1 + \frac{T}{2}\lambda}{1 - \frac{T}{2}\lambda}\right| \leq 1\right)}_{\text{ne treba znati}} \quad \Rightarrow \quad \sigma \leq 0$$

d) HEUNOV POSTUPAK:

$$\widetilde{x}_{k+1} = x_k + T \cdot \dot{x}_k$$

$$x_{k+1} = x_k + \frac{T}{2}\left(f(x_k, t_k) + f(\widetilde{x}_{k+1}, t_{k+1})\right)$$

↳ mjet stabilnost

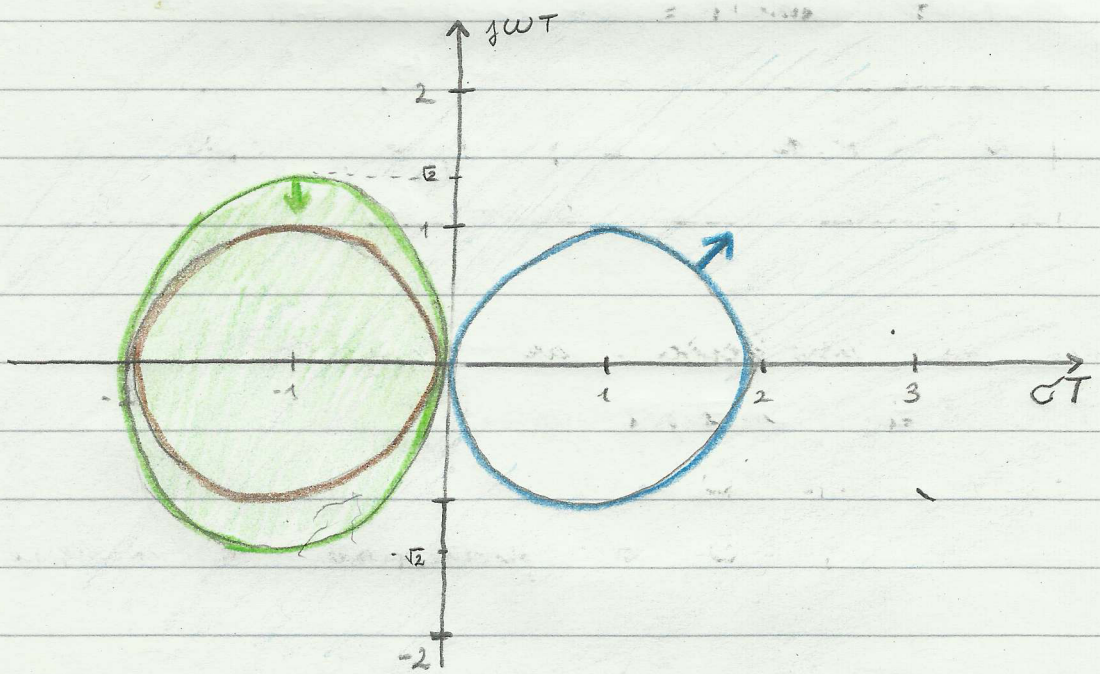
ZAD: Za radan sustav $\dot{\underline{x}} = \begin{bmatrix} -2 & 0 \\ 0 & 4 \end{bmatrix} \underline{x}$ predložite postupak numeričke integracije po pitanju stabilnost ?!

↳ trebamo svojstvene vrijednosti:

$$|\lambda \underline{E} - \underline{A}| = (\lambda+2)(\lambda-4) = 0$$

$$\lambda_1 = -2 \qquad \lambda_2 = 4$$

↳ prikažemo sada ovo u ravnini kako bi vidjeli veličine područja stabilnost:



↳ odgovara nam samo obrnuti Euler jer jedino on sadrži obje svojstvene vrijednosti:

$$T \geq \frac{2\sigma}{|\lambda|^2} \Rightarrow \quad T \geq -1 \quad (\lambda = -2)$$

$$T \geq 0.5 \quad (\lambda = 4)$$

LINEARNI VIŠEKORAČNI POSTUPCI

⇒ **koračnost**:

- „r"-koračni postupci koriste rješenja u „r" prethodnih koraka

⇒ linearnom kombinacijom prethodnih „r" vrijednosti varijabli stanja i prethodnih „r" vrijednosti derivacija varijabli stanja:

- OPĆI OBLIK:

$$x_{k+1} = T \cdot \beta_0 \cdot \dot{x}_{k+1} + \sum_{j=1}^{r} \left(\alpha_j \cdot x_{k-j+1} + T \cdot \beta_j \cdot \dot{x}_{k-j+1} \right)$$

⇒ postupak je implicitan ako ima uključenu derivaciju trenutka kojeg računamo:

$\beta_0 \neq 0 \Leftrightarrow$ postupak je implicitan

$\beta_0 = 0 \Leftrightarrow$ postupak je eksplicitan

⇒ ako je postupak implicitan, $\dot{x}_{k+1}$ treba prebaciti lijevo što će biti moguće jedino ako je funkcija „f" dovoljno dobra

↳ npr. za linearnu funkciju „f" će to uvijek biti moguće

(TO RADIMO ZA PRIPREMU ZA SLJEDEĆI KORAK POSTUPKA)

↳ za kompleksne funkcije „f", radimo iteraturno te pokazujemo da konvergiramo tako da je varijka u sljedećem koraku manja od varijke u prethodnom koraku:

$$X_{k+1}^{(m+1)} - X_{k+1}^{(m)} \approx T\beta_0 \frac{\partial f(x_j, t_{k+1})}{\partial X} \Big/ \left(X_{k+1}^{(m)} - X_{k+1}^{(m-1)}\right)$$

↳ iz ovoga dobivamo uvjet stabilnost koji nije potrebno eksplicitno znati, ali vidi u skripti da znati okvirno otprilike kako doći do njega i zato trebamo birati $X_{k+1}^{(0)}$ ∇

↳ **kako odabrati $X_{k+1}^{(0)}$?**

· PREDIKTORSKO-KOREKTORSKI POSTUPCI:

(P) ↳ PREDIKTOR: eksplicitni postupak za procjenu $X_{k+1}^{(0)}$

(C) ↳ KOREKTOR: implicitni postupak koji se ponavlja u barem jednoj iteraciji

↳ opći oblik:

$$P(EC)^S E$$

jednom primjeni prediktor, a nekoliko puta korektor

⇒ NAPOMENA: primjeti da je Heunov postupak zapravo prediktorsko-korektorski s jednom primjenom korektora (pokušaj shvatiti što je u tom postupku prediktor (Eulerova formula), a što je korektor (Trapezni postupak))

ZAD: Zadan je sustav $\underline{\dot{X}} = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} \underline{X}$. Proveđi jednu iteraciju PECE postupka uz početne vrijednosti varijabli stanja jednake 1 uz period integracije jednak 0.01. Kao prediktor upotrebi Eulerov postupak, a za korektor obrnuti Eulerov postupak!

↳ prediktor: (Euler)

$$X^{\circ}_{1,1} = X_{1,0} + T(X_{2,0}) = 1 + 0.01 \cdot 1 = 1.01$$

$$X^{\circ}_{2,1} = X_{2,0} + T(-5X_{1,0} - 2X_{2,0}) = 1 + 0.01(-5-2) = 0.93$$

↳ korektor: (obrnuti Euler)

$$X_{1,1} = X_{1,0} + T(X^{\circ}_{2,1}) = 1.0093$$

$$X_{2,1} = X_{2,0} + T(-5X^{\circ}_{1,1} - 2X^{\circ}_{2,1}) = 0.9309$$

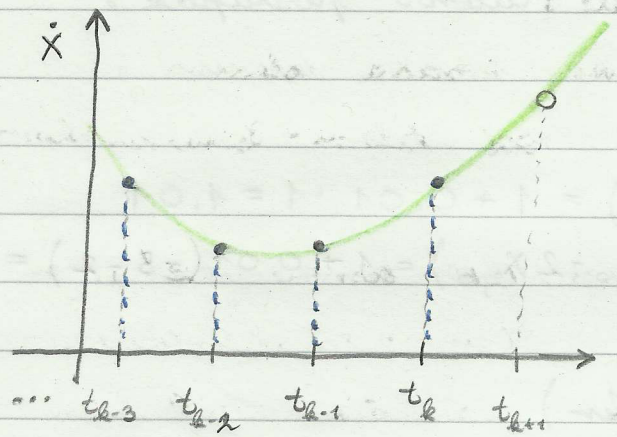
KRAJ PRVOG KORAKA!

⇒ NAPOMENA: u PECE postupcima korektor treba biti višeg reda od prediktora (treba biti precizniji od prediktora)

↳ pretpostavljam da gornji nadodatak istoga nema smisla jer su prediktor i korektor istog reda?

⇒ LAGRANGEOVA INTERPOLACIJA:

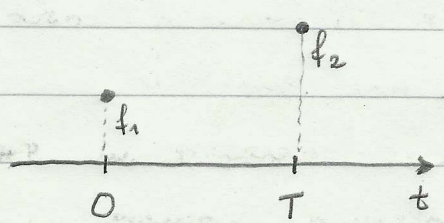
↳ matematički alat koji će nam poslužit ra procjenu $\dot{X}_{k+1}$ kod implicitnih postupaka (služi nam ra izgradnju prediktora)



konstruiramo polinom kroz poznate točke i pomoću njega predviđamo nepoznatu

↳ super nam je LaGrangeova interpolacija jer ona daje polinom što možemo lagano integrirati (to nam treba!) - da konstruiramo neku trigonometrijsku interpolaciju, to bi bilo puno teži ra integrirati

PR: Pogledajmo interpolaciju na samo dvije točke
(očekujemo pravac)



(PRVA TOČKA) $L_1(t) = \dfrac{(t-T)}{(0-T)}$ $\quad t=0 \Rightarrow L_1=1$ $\quad t=T \Rightarrow L_2=0$

(DRUGA TOČKA) $L_2(t) = \dfrac{(t-0)}{(T-0)} = \dfrac{t}{T}$

↳ iz tih barem polinoma radimo:

$$P(t) = \frac{T-t}{T}\cdot f_1 + \frac{t}{T}\cdot f_2 = \frac{f_2-f_1}{T}t + f_1$$

⇒ ADAMS-BASHFORTHOVI POSTUPCI:

↳ primjenjuju LaGrangeovu interpolaciju na izgradnju prediktora koristeći željeni broj točaka iz prošlosti

↳ vrijedi sljedeće:

- koliko koraka idemo u prošlost, toliki je red Adams-Bashforthovog postupka

↳ no, NO-FREE-LUNCH teorem dyeluje te kako povećavamo red Adams-Bashforthovog postupka tako smanjujemo područje stabilnosti

⇓

stoga, A-B postupci se ne koriste sami već samo kao prediktori

⇒ ADAMS-MOLTONOVI POSTUPCI:

↳ implicitni postupci jer koriste i jednu točku iz budućnosti da bi izgradili polinom korištenjem LaGrangeove interpolacije

↳ primjeti: „p" - koračni postupak je „p+1" reda

↳ dodatno, valja napomenuti da ako je A-M prvog reda, to je obrnuti Euler, a ako je A-M drugog reda, to je poprilično trapezni postupak

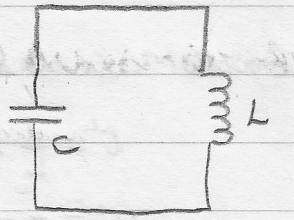
↳ stoga, Adams-Moltonovi postupci se koriste kao korektori jer su točniji jednog reda više od Adams-Bashforthovih postupaka / prediktora

⇒ NAPOMENA: područje stabilnosti korektora vrijedi samo ako se on pusti do konvergencije

⇓

u praksi se to nikad ne može postići, ali se može doći vrlo blizu, pa može izgledati jako slično ovim postupcima odprije poznatim (Euler, Trapez, ...)

ZAD: Imamo zadan sljedeći C-L krug:



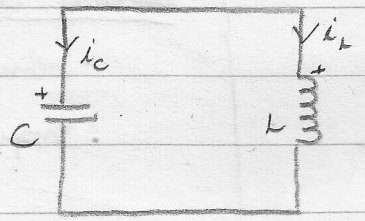
Za zadanu mrežu dobivamo sustav diferencijalnih jednadžbi i predložit odgovarajući način rješavanja!

Zadano je: $L = 0.1\,H$ $C = 1\,mF$

↳ treba identificirat varijable stanja:

- memorijski element su oni koji mogu pohranit energiju
- za naš primjer su memorijski elementi kondenzator i zavojnica

↳ uvedimo oznake na slici kruga:



$$i_C = C \cdot \frac{du}{dt} = C \cdot u_C'$$

$$u_L = L \cdot \frac{di}{dt} = L \cdot i_L'$$

↳ trebamo primjenit i Kirchoffova pravila:

$$u_C = u_L$$

$$i_C = -i_L$$

↳ uvrštavamo prve dvije jednadžbe u Kirchoffova pravila:

$$u_C = L \cdot i_L'$$

$$C \cdot u_C' = i_C$$

↳ postoj jedan napon i jedna struja, pa možemo

izbaciti indekse C i L:

$$i' = \frac{1}{L} u$$

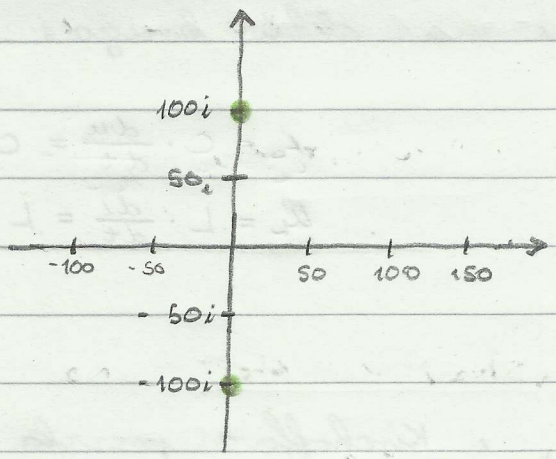
$$u' = -\frac{1}{C} i$$

↳ pišemo sustav: $\begin{bmatrix} i' \\ u' \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & 0 \end{bmatrix} \begin{bmatrix} i \\ u \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ -1000 & 0 \end{bmatrix} \begin{bmatrix} i \\ u \end{bmatrix}$

↳ izračunavamo svojstvene vrijednosti:

$$|\lambda \underline{E} - \underline{A}| = \begin{vmatrix} \lambda & -10 \\ 1000 & \lambda \end{vmatrix}$$

$$\lambda_1 = 100i \qquad \lambda_2 = -100i$$

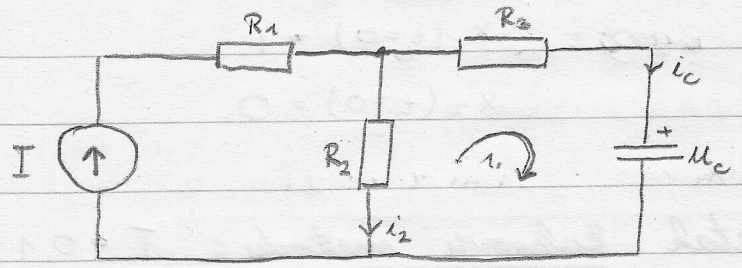


dolaze u obzir:
- obrnuti Euler
- trapezni postupak

↳ u stvarnosti, uvijek postoji otpor, pa se svojstvene vrijednosti ne nalaze na Y-osi nego lijevo od nje, pa u obzir dolaze i drugi postupci

↳ NAPOMENA: ovo je harmonički oscilator, pa je to sustav ekvivalentan matematičkom njihalu

ZAD: Opiši sljedeću mrežu diferencijalnim jednadžbama i odredi najveći T za Eulerov postupak. Provedi dvije iteracije Eulerovog postupka za $T = 0.01$!



$u_c(t=0) = 0$

$I = 1\,mA$

$R_1 = R_2 = 100\,\Omega$

$R_3 = 50\,\Omega$

$C = 100\,\mu F$

↳ imamo samo jedan memorijski element, pa će biti samo jedna diferencijalna jednadžba:

$$I = i_c + i_2 \qquad (1)\ i_2 \cdot R_2 = i_c \cdot R_3 + u_c$$

$$(I - i_c) R_2 = i_c R_3 + u_c$$

$$\Big\downarrow\ i_c = C \cdot u_c'$$

$$u_c' (-R_2 - R_3) C = u_c - I \cdot R_2$$

$$u_c' = \frac{1}{C(-R_2 - R_3)} u_c - \frac{I \cdot R_2}{C(-R_2 - R_3)} = -66.66\, u_c + 6.66 = f(u, t)$$

$$\lambda = -66.66 \Rightarrow \text{za Eulera: } T \leq -\frac{2\sigma}{|\lambda|^2} = 0.03$$

↳ provedimo dvije iteracije:

$t = 0.01$: $u_1 = u_0 + T \cdot f(u_0, t_0) = 0.0\dot{6}\,V$

$t = 0.02$: $u_2 = u_1 + T \cdot f(u_1, t_1) = 0.0\dot{8}\,V$

PR: Pokažimo sada rješavanje nekog nelinearnog sustava ovim numeričkim metodama integracije:

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -\sin X_1 - 0.5 X_2$$

↳ početni uvjet: $X_1(t=0) = 1$

$X_2(t=0) = 0$

⇒ krenimo za početak Eulerovu metodu: $T = 0.1$

$$X_{1,1} = X_{1,0} + T \cdot \dot{X}_{1,0} = 1 + 0.1 \cdot 0 = 1$$

$$X_{2,1} = X_{2,0} + T \cdot \dot{X}_{2,0} = 0 + 0.1\,(-\sin(1) - 0.5 \cdot 0) = -0.084$$

$$X_{1,2} = X_{1,1} + T \cdot \dot{X}_{1,1} = 0.991$$

$$X_{22} = X_{2,1} + T \cdot \dot{X}_{2,1} = -0.164$$

↳ vidimo da je postupak ist, jedino se sustav ne može prikazati u matričnom obliku jer sustav nije linearan

⇒ pokažimo sada Trapezni postupak: $T = 0.1$

$$X_{1,1} = X_{1,0} + \frac{T}{2}(X_{2,0} + X_{2,1})$$

$$X_{2,1} = X_{2,0} + \frac{T}{2}(-\sin(X_{1,0}) - 0.5 X_{2,0} - \sin X_{1,1} - 0.5 X_{2,1})$$

↳ sada, trebamo procjenu za:

$$X_{1,1}^{(0)} = 1 \; ; \; X_{2,1}^{(0)} = 0$$

↳ uzeli smo samo trenutne vrijednosti koje su zapravo početne jer je ovo prvi korak (da smo koristili Eulera kao procjenitelja, dobili bi Heunov algoritam)

↳ sada tu procjenu popravljamo:

$$X_{1,1}^{(1)} = 1 + 0.05(0+0) = 1 \quad ; \quad X_{2,1}^{(1)} = 0 + 0.05 \cdot (\ldots) = -0.084$$

$$X_{1,1}^{(2)} = 1 + 0.05(0 - 0.084) = 0.996 \quad ; \quad X_{2,1}^{(1)} = -0.082$$

$$X_{1,1}^{(3)} = 0.9959 \quad ; \quad X_{2,1}^{(3)} = -0.0819$$

$\Downarrow$

OVO KONVERGIRA!

↳ ako ne konvergira, procjena nije dobra, pa je bolje konstit neki eksplicitni postupak kao procjenitelj kako bi bili sigurni u konvergenciju

ZAD: Navedenim jednadžbama dan je proizvoljan postupak numeričke integracije. Odredi područje stabilnosti ∇! Ako se postupkom rješava sustav $\dot{X} = -0.1 \cdot X$; $T = 1$, hoće li rješenje biti stabilno?

$$m_1 = x_k + T \cdot f(x_k, t_k)$$

$$m_2 = x_k + T \cdot f(x_{k+1}, t_{k+1})$$

$$x_{k+1} = x_k + \frac{T}{2} \cdot [\, f(m_1, t_k) + f(m_2, t_{k+1}) \,]$$

↳ pa, odredimo uvjet stabilnost pomoću ispitne jednadžbe:

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

$$\dot{x} = \lambda \cdot x \quad \Rightarrow \quad m_1 = x_k + T \cdot \lambda \cdot x_k$$

$$m_2 = x_k + T \cdot \lambda \cdot x_{k+1}$$

$$x_{k+1} = x_k + \frac{T}{2} \left[\underbrace{\lambda (x_k + T \lambda x_k)}_{m_1} + \underbrace{\lambda (x_k + T \lambda x_{k+1})}_{m_2} \right]$$

↓

sada je dalje samo sređivanje da dobijemo grešku u „k"-tom i „k+1"-om koraku

$$x_{k+1} = x_k + \frac{T}{2} \lambda x_k + \frac{T^2 \lambda^2}{2} x_k + \frac{T}{2} \lambda x_k + \frac{T^2 \lambda^2}{2} x_{k+1}$$

$$x_{k+1} \left(1 - \frac{(\lambda T)^2}{2} \right) = x_k \left(1 + \lambda T + \frac{(\lambda T)^2}{2} \right)$$

δ_{k+1} $\qquad$ δ_k

$$\left| \frac{1 + \lambda T + \frac{(\lambda T)^2}{2}}{1 - \frac{(\lambda T)^2}{2}} \right| \leq 1 \quad \Rightarrow \text{ za dani sustav } \lambda = -0.1$$

$$\left| \frac{1 - 0.1 + \frac{0.01}{2}}{1 - \frac{0.01}{2}} \right| = 0.909 \leq 1$$

STABILNO!

POGREŠKE U POSTUPCIMA ANALIZE

⇒ zbog ograničene preciznosti prikaza brojeva u računalu te ograničene preciznosti operacija, javljaju se pogreške

⇒ UZROCI POGREŠAKA:

a) netočan / približni model

↳ npr. sinus kuta aproksimiramo (sjećate se?!) iznosom samog kuta: $\sin\varphi \approx \varphi$

b) aproksimacija u algoritmu

↳ npr. koristimo numeričku integraciju Eulerovim postupkom - veća greška od nekog postupka višeg reda (npr. Runge-Kutta)

c) pogreške zaokruživanja

↳ nastaju zbog ograničenog prikaza brojeva u računalu

↳ nisu uvijek očite te ih često nismo niti svjesni

↳ o njima ćemo se više baviti u sljedećem poglavlju //

⇒ PRIKAZ S POMIČNOM TOČKOM:

↳ za točno množenje dva „n"-bitna broja trebamo $2n$ bitova

↳ za točno brojanje dva „n"-bitna broja trebamo $n + (e_{MAX} - e_{MIN})$

↳ uvedimo neke oznake:

- BAZA: β (u računalu: $\beta = 2$)
- BROJ ZNAMENKI: p
- EKSPONENT: $e_{MIN} \leftrightarrow e_{MAX}$

↳ danas se koristi IEEE Standard 754:



z: predznak

e: pomaknut eksponent

f: frakcija / mantisa (nije cijeli dio signifikanda)

k: broj bitova za „e"

p': broj bitova za „f"

- signifikand: 1. f ili 0. f

p: broj bitova za signifikand

(vrijedi: $p = p' + 1$)

- eksponent: pomaknut za $\boxed{a = 2^{(k-1)} - 1}$

• dekodiranje broja zapisanog u IEEE 754 standardu:

Z	e	f	VRIJEDNOST
0 ili 1	0	0	$(-1)^z \cdot 0 = \pm 0$ (pozitivna ili negativna nula)
0 ili 1	0	$1 \le f \le 2^{p'}-1$	$(-1)^z \cdot 2^{-a+1} \cdot (0.f)$ (denormirani significand)
0 ili 1	$1 \le e \le 2^k-2$	$0 \le f \le 2^{p'}-1$	$(-1)^z \cdot 2^{e-a} \cdot (1.f)$
0 ili 1	2^k-1 (sve jedinice)	0	$(-1)^z \cdot \infty$
0 ili 1	2^k-1 (sve jedinice)	$f \neq 0$	NaN

(ovu tablicu treba znati)

Pr: Pogledajmo zapis s manjim brojem bitova:

$k=3$; $p'=5$

$a = 2^{(3-1)}-1 = 3$ $\quad e \in [1, \dots, 6] \Rightarrow$ EKSP $\in [-2, 3]$

↳ najmanji broj: $000|00001 \Rightarrow 2^{1-a} \cdot 0.f = 2^{-2} \cdot 0.00001 =$

$= 2^{-2} \cdot 0.03125$

$= 0.0078125$

↳ najveći broj: $110|11111 \Rightarrow 2^{6-3} \cdot 1.f = 2^3 \cdot 1.11111 =$

$= 2^3 \cdot 1.96875 =$

$= 15.75$

⇒ NAPOMENA: broj bitova eksponenta određuje raspon vrijednosti koje možemo prikazati

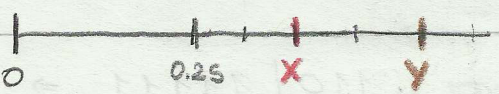
⇒ NAPOMENA: broj bitova significanda / frakcije određuje preciznost (rezoluciju) jer u istom rasponu možemo prikazati više vrijednosti

⇒ NAPOMENA: najpreciznij smo oko nule, a sve dalje možemo zapisati sve rjeđe brojeve (apsolutna preciznost pada, a relativna ostaje ista)

⇒ zašto denormirano područje?

↳ da nemamo denormirano područje, neki mogli prikazati ništa manje od 0.25, pa bi se desilo sljedeće čudno ponašanje:

- ako su X i Y malo veći od 0.25 i različiti su, X == Y bi bio „false", a X - Y bi davalo nulu!



X - Y < 0.25, pa bi u računalu to bila nula!

ZAD: Po uzoru na IEEE 754 definiran je prikaz sa
1 bitom za predznak, 4 bita za eksponent i
4 bita za frakciju. Ako je broj u tom prikazu
zapisan ovako: 100001100, odredi njegovu vrijednost!

$z = 1 \Rightarrow$ broj je negativan

$e = 0 \Rightarrow$ denormirani signifikand

$a = 2^{k-1} - 1 = 2^3 - 1 = 7$

$e \in [1, \ldots, 14] \Rightarrow EKSP \in [-6, \ldots, 7]$

↳ dekodiramo našu vrijednost:

$1|0000|1100 \rightarrow v = -2^{-6} \cdot 0.1100 = -2^{-6} \cdot 0.75 = \ldots$

ZAD: Definiran je pribor s jednim bitom za predznak i nepoznatim brojem za eksponent i frakciju. Ako je 5.625 u tom pribaru 0100101101, odredi te nepoznate brojeve bitova?

↳ zapišimo 5.625 u binarnom zapisu:

$5.625 = 101.101 =$

$= 1.01101 \cdot 2^{(2)}$

ovo treba biti negdje vidljivo u zapisu broja

0|1001|01101.

ovo je traženi broj, pa ostaje 4 bita za eksponent

↳ treba provjeriti je li 2 stvarno pomaknuti 1001:

$k = 4 \Rightarrow a = 7$

$EKSP \in 2^{9-7} = 2^{(2)}$

↳ dobili smo dobro, pa je:

$p' = 5$ //

PR: Pogledajmo preciznost nekih raspona za funkcije $\sin(x)$ i za funkciju $10000 + \sin(x)$?

a) $\sin(x) \longrightarrow [-1, 1]$ ⇒ raspon tražimo preciznost na intervalu $[-1, 1]$

- FLOAT: $(1|8|23)$; $EKSP \in [-126, 127]$
  - ↳ broj vrijednosti: $2 \cdot 127 \cdot 2^{23} \approx 2.13 \cdot 10^9$
  (reći, preciznost je oko 10^9)
- DOUBLE: $(1|11|52)$; $EKSP \in [-1022, 1023]$
  - ↳ broj vrijednosti: $2 \cdot 1023 \cdot 2^{52} \approx 9 \cdot 10^{18}$
  (reći, preciznost je oko 10^{18})

b) $10000 + \sin(x) \rightarrow [9999, 10001]$ ⇒ raspon tražimo preciznost na intervalu $[9999, 10001]$

- FLOAT: $(1|8|23)$; $EKSP \in [-126, 127]$
  - ↳ broj vrijednosti: $\frac{2^{23}}{8192} \cdot 2 = 2048 \approx 2 \cdot 10^2$
  ← KAKO DOBITI OVAJ BROJ?
  (reći, preciznost je oko 10^2)

↳ na double je analogno te je double puno bolji izbor ako ovako nešto treba računati

↳ **naklyučak**: preciznost ovisi o brojevnom području u kojem računamo

⇒ POGREŠKE ZAOKRUŽIVANJA:

↳ pogledajmo koliku pogrešku radimo kada zapisujemo neku vrijednost u računalu

↳ **APSOLUTNA POGREŠKA:**

$$AP = |X - rd(X)|$$

↳ maksimalna je kada je prva znamenka koju odbacujemo upravo $\beta/2$ (=1 u računalu)

↳ ULP ⇒ „units-in-the-last-place"

⇒ težina znamenke na zadnjem mjestu

$$\boxed{ULP = \beta^{-(p-1)} \cdot \beta^{e}}$$

↳ maksimalna apsolutna pogreška:

$$MAP = \frac{1}{2} ULP$$

$\Downarrow$ u računalu

$$MAP = 2^{a-p}$$

PR: Pogledajmo pogrešku kod zapisa 1|3|5 !

$p' = 5$; $p = 6$

$$MAP = 2^{a-p} = 2^{-3} = 0.125$$

↓

• vizualiziramo s dvije najveće susjedne vrijednosti:

$$\left.\begin{aligned} MAX_1 &= 2^3 \cdot 1.11111 = 15.75 \\ MAX_2 &= 2^3 \cdot 1.11110 = 15.5 \end{aligned}\right\}$$

$$MAP = \frac{MAX_2 - MAX_1}{2} = 0.125$$

↳ **RELATIVNA POGREŠKA:**

$$RP = \left| \frac{X - rd(x)}{X} \right|$$

↳ najveća pri dnu raspona eksponenta

↳ EPS: „strogna međa"

$$EPS = \frac{1}{2} \beta^{-(p-1)}$$

⇓

↳ u računalu je $EPS = 2^{-p}$

↳ pmnjeti da je relativna pogreška uvijek manja od EPS:

$$|\varepsilon| \leq EPS$$

(stoga, često relativnu pogrešku izražavamo u broju EPS-ova)

↳ nadalje, vrijedi izraz:

$$rd(x) = x(1+\varepsilon)$$

⇒ ARITMETIKA S POMIČNOM TOČKOM:

↳ za početak, pogledajmo primjer:

PR: Imamo double varijable:

$$p' = 52, \; p = 53 \Rightarrow EPS = 2^{-53}$$

$$EPS \approx 1.102 \cdot 10^{-16}$$

↳ ili ove vrijednosti EPSa, vidimo da je $1 + 10^{-16}$ uvijek 1 u računalu

↳ prva veća vrijednost od 1 je:

$$2^{0} \cdot (1 + 2^{-52}) = 1 + 2EPS$$

↳ prva manja vrijednost od 1 je:

$$2^{-1} (2 - 2^{-52}) = 1 - EPS$$

⇓

ovo je objašnjenje zašto je:

$1 + 10^{-16} - 1 = 0$ (krivi rezultat ☹)

$1 - 10^{-16} - 1 = -EPS$ (dobar rezultat ☺)

ZADAĆA: Za zapis brojeva 1/4/5 odredi pogrešku za brojeve -249.5 i 25 pomičnom zbrajanja:

$-249.5 \Rightarrow 2^7 \cdot 1.11111 = 252$

$2^7 \cdot 1.11110 = \boxed{248}$ → uzimamo ovog jer je bliže

$25 \Rightarrow 2^4 \cdot 1.10010 = \boxed{25}$ → točno je pa ga odmah možemo uzet

$(248): \quad 2^7 \cdot 1.11110$

$-(25): \quad -2^7 \cdot 0.00110\,01$

$(-)\ 2^7 \cdot 1.11000$

OVO IZGUBIMO JER MOŽEMO PAMTITI SAMO $p'=5$ BITOVA

$-2^7 \cdot 1.11000 \Rightarrow -224$

(stvarni rezultat je $25 - 249.5 = -224.5$)

↳ izračunajmo **apsolutnu pogrešku**:

$AP = 0.5$ (odmah se vidi)

↳ odredimo preko ULP-ova:

$ULP = 2^{-(p-1)} \cdot 2^e = 2^{-5} \cdot 2^7 = 4$

$AP = 0.5 = \frac{1}{8} ULP$

↳ izračunajmo **relativnu pogrešku**:

$EPS = 2^{-p} = 2^{-6}$

$\varepsilon = \frac{0.5}{224.5} = 0.0022 \approx 0.14\,EPS$

PR: Podsjetimo se malo standardnog double raspa ▽

double 1/11/52

↳ vrijednost unutar intervala $[1, 2\rangle$ iznosi:

2^{52} jer se mijenja samo zadnjih 52 bita

↳ vrijednost unutar intervala $[2, 3\rangle$ iznosi:

$\frac{1}{2} \cdot 2^{52}$ jer sada imamo 51 bita koji se mijenjaju

↳ koja je prva vrijednost veća od 1?

• pa j to je $1 + 2\text{EPS} = 1 + 2^{-52}$

ZAD: Definirajte prikaz po uzoru na IEEE 754 tako da u području [-100, 100] najveća apsolutna pogreška ne smije biti veća od 0.5 !! Prikažite brojeve -99 i 4.75 u tom prikazu, provedi operaciju zbrajanja i odredi pogrešku u rezultatu.

↳ koliki eksponent trebamo za brojeve do 100:

· 2^7 je previše jer se prikazuje kao 1. mjesto, pa trebamo 6 bita !!

· kako je eksponent pomaknut, sa 3 bita možemo do 2^3, a sa 4 bita možemo do 2^7 ($a = 7$)

· stoga, trebamo 4 bita

↳ koliki frakcijski trebamo?

· odredujemo na temelju željene pogreške u danom intervalu

· prikažimo 100:

$100 = 2^6 \cdot 1.1001$
$1.1000 \cdot 2^6 = 96$

} razlika je 4, pa ovo nije dobro

↳ mi trebamo 99 jer je to upravo $2 \cdot 0.5$ do 100 :)

$-99 = -2^6 \cdot \underline{1.00011}$

ukupno trebamo 6 bita

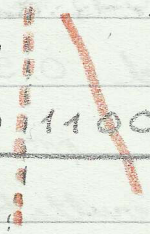
↳ sada smo dobili prikaz:

1 | 4 | 6

OKRENI →

↳ sada možemo provest zbrajanje:

$$
\begin{array}{rr}
(99) & 2^6 \cdot 1.100011 \\
(4.75) & -\ 2^6 \cdot 0.000100\,1100 \\
\hline
 & 2^6 \cdot 1.011111
\end{array}
$$



• oduzimamo manjeg od većeg i naštelovamo predznak rezultata

↳ dekodiranje rezultata:

$$-2^6 \cdot 1.011111 = 95$$

↳ apsolutna pogreška: 0.75

↳ relativna pogreška:

$$\varepsilon = \frac{0.75}{94.25} = 0.78\%$$

• izrazimo relativnu pogrešku u EPSovima, pa dobivamo:

$$EPS = 2^{-p} = 2^{-7}$$

$$\varepsilon = 1\,EPS$$

PR: Vidimo primjer iz skripte koji pokazuje da rezultat aritmetičkih operacija može drastično ovisiti o redoslijedu njihovog provođenja ▽ $EPS = 5 \cdot 10^{-8}$

a) $a + (b + c) \Rightarrow$ A.P. $= 2 \cdot 10^{-12}$; $\varepsilon = 3.1 \cdot 10^{-3} = 0.06 EPS$

b) $(a + b) + c \Rightarrow$ A.P. $= 3.7 \cdot 10^{-7}$; $\varepsilon = 5.8 \cdot 10^{-4} = 11580 EPS$

↓ ▽ ovo je puno lošije jer smo više od 10^4 EPSa relativne greške

c) $(a + c) + b \Rightarrow$ dođe iste rezultate kao što smo dobili pod b) //

↳ samo 1 put nas vodi k dobrom rezultatu !

⇒ TEOREM: na odusimanju sa „p" znamenki, relativna greška može biti („β"−1) - u najgorem slučaju

PR: Pokažimo teorem na sljedećem primjeru:

$\beta = 10$, $p = 4$

↳ greška se događa s brojevima koji su na granici područja eksponenta:

• uzmimo za primjer sljedeće brojeve:

$X = 10000 = 10^4 \cdot 1.000$

$-Y = 9999 = 10^3 \cdot 9.999$

$(X-Y) = 10^4 = 10$

$10^4 \cdot 1.000!$

$- 10^4 \cdot 0.9999$

$10^4 \cdot 0.001$

⇓

10 je rezultat ?!?!?

(nije baš dobro jer smo očekivali 1)

PR: Prikažimo najveću pogrešku za DOUBLE prikaz!

$$EPS = 2^{-53}$$

$$\left.\begin{array}{l} 2 = 2^1 \cdot 1.000\ldots00 \\ 2 - 2^{52} = 2^0 \cdot 1.111\ldots11 = 2 - EPS \end{array}\right\} \text{ovo oduzmi}$$

$$\begin{array}{r} 2^1 \cdot 1.000\ldots00 \\ -\ 2^1 \cdot 0.111\ldots11 \\ \hline 2^1 \cdot 0.000\ldots01 \end{array} = 2^1 \cdot 2^{-52} = 2^{-51} = 4\,EPS$$

$$\varepsilon = 100\%$$

⇒ NAPOMENA: sklopovske implementacije zbog takvih grešaka koriste zaštitne znamenke koje pomažu kod aritmetičkih operacija (GUARD DIGITS)

⇒ TEOREM: za oduzimanje uz „p+1" znamenku, relativna greška može biti 2EPS u najgorem slučaju

⇒ NAPOMENA: množenje i dijeljenje nemaju ovih problema jer za to ionako trebamo duplo bitova //

RASPROSTIRANJE POGREŠKE

⇒ pitanje: kako razlikit algoritmi uz približne operacije utječu na konačnu pogrešku

⇒ NEKI POJMOVI:

x_i ⇒ ulazni argumenti

$f(\underline{x})$ ⇒ matematička funkcija koju računamo (to je onaj točan rezultat)

Y ⇒ dobiveni rezultat

⇒ za svaku proizvoljnu operaciju $\oplus$ vrijedi:

$$\boxed{a \oplus b = (a \oplus b)(1+\varepsilon)}$$

↓

znači, točan rezultat je približan rezultat plus pogreška

- ovo koristimo u analizi algoritama

PR: Pogledajmo jednostavno zbrajanje sa 3 pribrojnika te uvidimo razlike ovisno o redoslijedu zbrajanja!

↳ računamo: $a+b+c$

$a = 2.2 \cdot 10^{-5}$

$b = 3.3 \cdot 10^{1}$

$c = -3.3 \cdot 10^{1}$

↳ pogledajmo ograničenje greške:

$$\varepsilon_y = \frac{a+b}{a+b+c}\varepsilon_1 + \varepsilon_2$$

$$\downarrow \quad |\varepsilon_1|, |\varepsilon_2| \leq 2EPS$$

$$|\varepsilon_y| \leq \left| 1 + \frac{a+b}{a+b+c} \right| 2EPS$$

} ovo je općenito za zbrajanje dva broja

1) redoslijed zbrajanja: $a+(b+c)$

$b+c \approx 6.18 \cdot 10^{-4}$

↳ ovo je dobro! (samo zaokruženo)

2) redoslijed zbrajanja: $(a+b)+c$

$a+b = 33.68$

$a+b+c \approx 10^{-4}$

↳ ovo nije dobro! (samo zaokruženo)

PR: Istražimo pogrešku na računanju razlike dva kvadrata: $a^2 - b^2 = (a+b)(a-b)$

$$\left.\begin{aligned} \mu_1 &= (a+b)(1+\varepsilon_1) \\ \mu_2 &= (a-b)(1+\varepsilon_2) \end{aligned}\right\} \; |\varepsilon_1|, |\varepsilon_2| \leq 2\,EPS$$

$$\tilde{Y} = \mu_1 \odot \mu_2 = (\mu_1 \cdot \mu_2)(1+\varepsilon_3) \; ; \quad |\varepsilon_3| \leq EPS$$

$$\tilde{Y} = (a+b)(1+\varepsilon_1)(a-b)(1+\varepsilon_2)(1+\varepsilon_3)$$

$$\varepsilon_Y = \frac{\tilde{Y} - Y}{Y} = \ldots = 2\varepsilon_1 + \varepsilon_3 \leq 5\,EPS$$

↳ e, da smo ovo proveli tako da prvo kvadriramo svaki član i onda oduzmemo, dobivamo puno lošiji slučaj jer greška ovisi o argumentu:

$$\varepsilon_Y = \ldots + \underbrace{(\varepsilon_1 + \varepsilon_2)\,b^2}$$

ovisi o argumentu, pa ne možemo ograničiti odozgo

⇒ FAKTOR POJAČANJA POGREŠKE:

- sada se pitamo koliko složena računska operacija pojačava pogrešku u argumentima
- stoga, imamo grešku u argumentima, a ne u operacijama (njih promatramo kao složene)
- on iznosi:

$$\varepsilon_y = \sum_{i=1}^{n} \frac{x_i}{f_j(\underline{x})} \cdot \frac{\partial f_j(\underline{x})}{\partial x_i} \cdot \varepsilon_{x_i}$$

PR: Vratimo se na primjer a^2-b^2 i pogledajmo faktor pojačanja pogreške!

$$\varepsilon_{a^2-b^2} = \frac{a^2}{a^2-b^2}\varepsilon_{a^2} + \frac{b^2}{a^2-b^2}\varepsilon_{b^2}$$

} prosuvišno povećanje pogreške (ako je a^2-b^2 malo, pogreška se množi s velikim brojem)

$$\varepsilon_{(a+b)(a-b)} = \varepsilon_{a+b} + \varepsilon_{a-b}$$

} ovo je dobro jer ne ovisi o argumentu

⇒ NAPOMENA: iz prethodnih primjera zaključujemo da prvo valja vršiti operacije koje imaju veći faktor povećanja pogreške

⇒ ZAKLJUČNO IZ OVOG POGLAVLJA:

↳ množenje „good" zbrajanje „bad"

↳ za velike sume brojeva ($\sum_{i=1}^{n} x_i$) koristimo Kahanov algoritam

↳ koristi ugrađene funkcije kad god je moguće za obavljanje nekih operacija

↳ za testiranje vlastitih implementacija:

- napravi „float" i „double" inačicu te ako ima neke veće razlike, možemo zaključiti da je implementacija ranjiva na pogreške u zaključivanju

↳ pazi kako uspoređuješ brojeve

SLOŽENOST ALGORITAMA

⇒ **OCJENA SLOŽENOSTI:**

a) A posteriori

↳ nakon implementacije

↳ svodi se na mjerenje, pa podaci o mjerenju

da imamo gotovu implementaciju

b) A priori

↳ radi se prije implementacije

↳ ovim se mi bavimo!

⇒ trajanje jedne operacije proglašava se jedinicom mjere, a ostale se uspoređuju s nekim faktorom u odnosu na jedinične operacije

⇒ jedinična mjera može biti:

+, -, *, /, slanje poruke, broj evaluacija, ...

↳ po default-u se misli na broj iteracija

⇒ mjerimo broj operacija u ovisnosti o n:

↳ „n" je parametar

$n \in \mathbb{N}_0$

↳ broj operacija je $f(n)$

⇒ MJERE SLOŽENOSTI

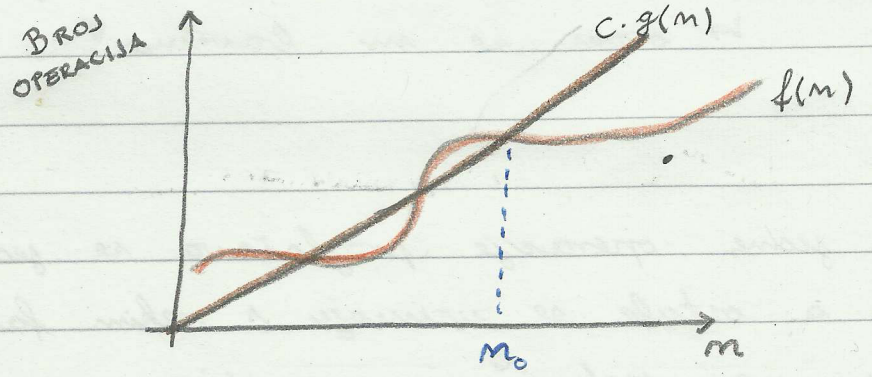
a) O - NOTACIJA

↳ $f(n) = O(g(n)) \Leftrightarrow \exists\, c, n_0 > 0$ za koje vrijedi:

$$|f(n)| \leq c\,|g(n)| \quad \forall n \geq n_0$$

↳ drugi način definiranja je traženje funkcije $g(n)$ za koju vrijedi:

$$0 \leq \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty .$$



↳ ova je asimptotska ograda

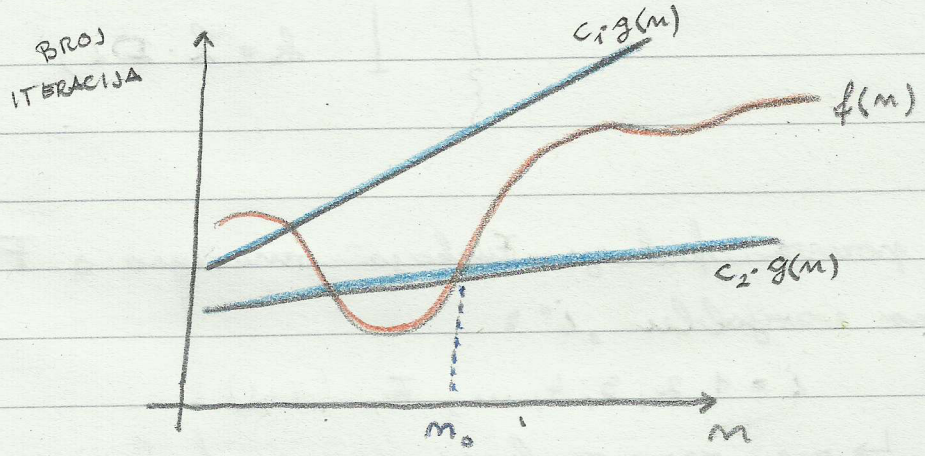
b) Θ - NOTACIJA

↳ $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_0 > 0$ za koje vrijedi:

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n > n_0$$

↳ drugi način definiranja je traženje funkcije $g(n)$ za koju vrijedi:

$$0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$



↳ ovo je donja ograda

c) σ - NOTACIJA

↳ definirana je ovako pa želimo točnost:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$$

ZAD: Procjeni složenost danog algoritma i broj operacija množenja i dijeljenja u ovisnosti o ulaznom parametru „n"?

```
Početak(n) {
    i = 1 ; k = n²
    Funkcija(n, k, i);
}
```

```
Funkcija(n, k, i) {
    k = k - i ; i = i + 1 ;
    AKO ( k > 0 )
        Funkcija(n, k, i);
    ZA i = 1 do n
        k = k · 2 · i ;
}
```

↳ broj poziva funkcije Funkcija označimo s F

↳ pratimo varijablu „i":

$$i = 1, 2, 3, 4, \ldots, F, (F+1)$$

↳ ona zapravo broj poziva funkcije

$$\sum i = k \quad \Rightarrow \text{skuži zašto to vrijedi}$$

$$\sum i = k = n^2$$

$$\frac{F(F+1)}{2} = k = n^2 \quad \Rightarrow \quad F^2 + F - 2n^2 = 0$$

$$F = \frac{-1 \pm \sqrt{1+8}}{2}$$

$$F = -\frac{1}{2} + \frac{\sqrt{1+8n^2}}{2}$$

↳ za procjenu složenosti, broj poziva funkcije procjenjujemo (aproksimiramo), ovako:

$$F = -\frac{1}{2} + \frac{\sqrt{1+8n^2}}{2} \approx \frac{\sqrt{8n^2}}{2} = \sqrt{2}\,n$$

↳ svaki put kad se funkcija poziva, radimo $2n$ operacija množenja (u „for" petlji):

$$f(n) = 2n\,F$$

$$f(n) = 2n \cdot \sqrt{2}\,n$$

$$f(n) = 2\sqrt{2}\,n^2$$

↳ složenost u O notacij: $O(n^2)$

↳ složenost u o notacij: $o(2\sqrt{2}\,n^2)$ //

ZAD: Funkcija RadNesto ima $2n^2$ broj operacija!
Odredi broj operacija danog algoritma!

```
Pocetak (n) {
    i = n;
    DOKJE (n) {
        RadNesto ();
        ZA  j = n DO 1 {
            RadNesto ();
        i = i/2;
}
```

↳ ponavljanja DOKJE petlje: $\underbrace{n, \frac{n}{2}, \frac{n}{4}, \ldots, 1}$

↳ broj ponavljanja je:
$\lfloor \log_2(n) \rfloor$

↳ ponavljanja ZA petlje: „n" puta

$$f(n) = \log_2 n \cdot RN() + n \cdot \log_2 n \cdot RN()$$

$$f(n) = 2n^2 \log_2 n + 2n^3 \log_2 n$$

↳ stoga, vrijedi složenost:

$$O(n^3 \log_2 n); \; o(2n^3 \log_2 n)$$