OBSAH

doc. Ing. Josef Daněk, Ph.D.

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Plzeň 2024

Motto: "Hodíme to do počítače a dostaneme výsledek"

Otázky:

Jaký je výsledek?

Dobrý?

Špatný?

Proč?

Úvodní motivační příklady

1) $$\sum_{k=1}^{100000} \frac{1}{10} = 10000$$

```
%-------------------
  s=0;
  h=single(1/10);
  for i=1:100000
     s=s+h;
  end;
  s
%-------------------

s =

    9.9985566e+03
```

```
%-------------------
  s=0;
  for i=1:100000
    s=s+1/10;
  end;
  s
%-------------------

s =

    1.000000000001885e+04
```

Poznámka: Aritmetika s n-platnými ciframi s užitím zaokrouhlování nebo řezání.

2) Vyčíslení hodnoty funkce zadané ve dvou tvarech

$$f(x) = x(\sqrt{x+1} - \sqrt{x}) \text{ a } \quad g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

Platí: $\quad x(\sqrt{x+1} - \sqrt{x}) \cdot \dfrac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} = \dfrac{x(x+1-x)}{\sqrt{x+1} + \sqrt{x}} = \dfrac{x}{\sqrt{x+1} + \sqrt{x}}$

Určete $f(300)$ a $g(300)$ pomocí aritmetiky se 6-ti platnými ciframi a užitím zaokrouhlování.

$$\begin{aligned} f(300) &\approx 300(\sqrt{301} - \sqrt{300}) \\ &= 300(17{,}3494 \text{ - } 17{,}3205) = 300 \cdot 0{,}0289 \\ &= \mathbf{8{,}67000} \\ g(300) &\approx \frac{300}{\sqrt{301} + \sqrt{300}} \\ &= \frac{300}{17{,}3494 + 17{,}3205} = \frac{300}{34{,}6699} \\ &= \mathbf{8{,}65304} \end{aligned}$$

Přesný výsledek je **8,653049162609 ...**

Chyba při vyčíslení $g(300)$ je mnohem menší než u $f(300)$.

3) Vyčíslení hodnoty polynomu zadaného ve dvou tvarech

$$P(x) = x^3 - 3x^2 + 3x - 1 \text{ a } \quad Q(x) = ((x-3)x + 3)x - 1$$

Platí: $\quad ((x-3)x+3)x - 1 = ((x^2 - 3x + 3)x - 1 = x^3 - 3x^2 + 3x - 1$

Určete $P(2{,}19)$ a $Q(2{,}19)$ pomocí aritmetiky se 3-mi platnými ciframi a zaokrouhlování.

$$\begin{aligned} P(2{,}19) &\approx 2{,}19^3 \text{ - } 3 \cdot 2{,}19^2 + 3 \cdot 2{,}19 \text{ - } 1 \\ &= 10{,}5 \text{ - } 3 \cdot 4{,}80 + 3 \cdot 2{,}19 \text{ - } 1 \\ &= 10{,}5 \text{ - } 14{,}4 + 6{,}57 \text{ - } 1 = \mathbf{1{,}67} \\ Q(2{,}19) &\approx ((2{,}19 \text{ - } 3)2{,}19 + 3)2{,}19 \text{ - } 1 \\ &= (\text{- } 0{,}81 \cdot 2{,}19 + 3)2{,}19 \text{ - } 1 = (\text{- } 1{,}77 + 3)2{,}19 \text{ - } 1 \\ &= 1{,}23 \cdot 2{,}19 \text{ - } 1 = 2{,}69 \text{ - } 1 = \mathbf{1{,}69} \end{aligned}$$

Přesný výsledek je **1,685159 ...**

pro $P(2{,}19)$ je chyba 0,015159

pro $Q(2{,}19)$ je chyba -0,004841

4) Hledání kořenů polynomu $P(x) = (x-1)^n$

Pro polynom stupně $n \geq 5$ $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$ obecně nelze řešení spočítat pomocí konečného počtu kroků pomocí operací $+, -, \cdot, /$ a odmocniny $\sqrt[r]{\ }, r \in \mathbb{N}$. Paolo Ruffini (nekompletní důkaz - 1813), Niels Henrik Abel (1824) .

Implementace v Matlabu (například pro $n = 4$)

```
%------------------------------
  polynom=poly([ 1 1 1 1 ])
  koreny=roots(polynom)
%------------------------------


polynom =

     1    -4     6    -4     1


koreny =

     1.000223371630863e+00
     9.999999706778071e-01 + 2.233423015147889e-04i
     9.999999706778071e-01 - 2.233423015147889e-04i
     9.997766870135252e-01
```

5) Určete prvních 15 členů posloupnosti, která je zadána rekurentní formulí

$$\begin{aligned} x_1 &= 5 \\ x_2 &= 1 \\ x_n &= 20{,}2\ x_{n-1} - 4\ x_{n-2}, \qquad n \geq 3 \end{aligned}$$

Přesné řešení je $x_n = \left(\frac{1}{5}\right)^{n-2}$

Ověření:

$$x_1 = \left(\frac{1}{5}\right)^{1-2} = 5, \quad x_2 = \left(\frac{1}{5}\right)^{2-2} = 1$$

$$\left(\frac{1}{5}\right)^{n-2} = 20{,}2\left(\frac{1}{5}\right)^{n-3} - 4\left(\frac{1}{5}\right)^{n-4} \qquad /\cdot\left(\frac{1}{5}\right)^{4-n}$$

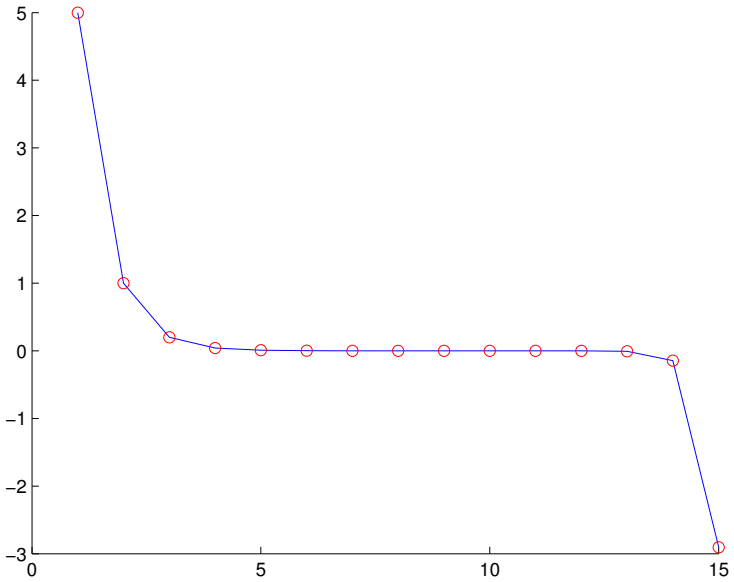
$$\left(\frac{1}{5}\right)^{2} = 20{,}2 \cdot\frac{1}{5} - 4$$

$$0{,}04 = 4{,}04 - 4$$

Výsledky z Matlabu:

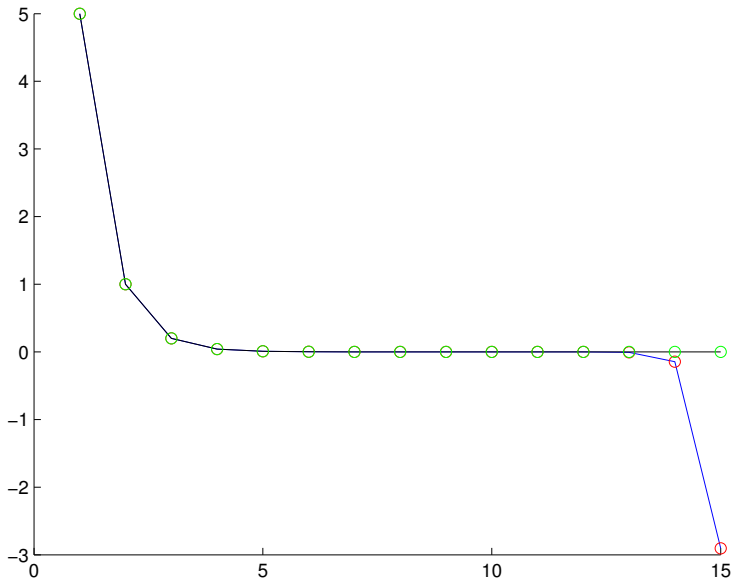
```
%-----------------------------------------
  x(1)=5;
  x(2)=1;

  for n=3:15
    x(n) = 20.2 * x(n-1) - 4 * x(n-2);
  end;
%-----------------------------------------
```

n	x(n)
1	5.000000
2	1.000000
3	0.200000
4	0.040000
5	0.008000
6	0.001600
7	0.000320
8	0.000064
9	0.000013
10	0.000002
11	-0.000018
12	-0.000363
13	-0.007259
14	-0.145175
15	-2.903496



```
%-----------------------------------------
  x(1)=5;
  x(2)=1;
  presne_x(1)=x(1);
  presne_x(2)=x(2);

  for n=3:15
    x(n) = 20.2 * x(n-1) - 4 * x(n-2);
    presne_x(n) = 0.2^(n-2);
  end;
%-----------------------------------------
```

n	x(n)	presne_x(n)	chyba
1	5.000000	5.000000	0
2	1.000000	1.000000	0
3	0.200000	0.200000	-7.21645e-16
4	0.040000	0.040000	-1.41831e-14
5	0.008000	0.008000	-2.83546e-13
6	0.001600	0.001600	-5.67089e-12
7	0.000320	0.000320	-1.13418e-10
8	0.000064	0.000064	-2.26836e-09
9	0.000013	0.000013	-4.53671e-08
10	0.000002	0.000003	-9.07342e-07
11	-0.000018	0.000001	-1.81468e-05
12	-0.000363	0.000000	-0.000362937
13	-0.007259	0.000000	-0.00725874
14	-0.145175	0.000000	-0.145175
15	-2.903496	0.000000	-2.9035



Obecné řešení dif. rovnice

$x_n = 20,2\, x_{n-1} - 4x_{n-2}$

Řešení hledáme ve tvaru $x_n = A^n$

$$\begin{aligned} A^n &= 20{,}2\ A^{n-1} \text{ - } 4\ A^{n-2} \qquad /\cdot A^{2-n} \\ A^2 &= 20{,}2\ A \text{ - } 4 \\ 0 &= A^2 \text{ - } 20{,}2\ A + 4 \end{aligned}$$

$$\sqrt{D} = \sqrt{20,2^2 - 4 \cdot 1 \cdot 4} = \sqrt{392,04} = 19{,}8$$

$$A_{1,2} = \frac{20,2 \pm 19,8}{2} = \begin{cases} 20 \\ 0,2 \end{cases}$$

Obecné řešení: $\boxed{x_n = c_1 \cdot 20^n + c_2 \cdot \left(\frac{1}{5}\right)^n}$

V našem případě $c_1 = 0$.

Vlivem zaokrouhlovacích chyb se stane koeficient c_1 nenulový $\Rightarrow$ nestabilní rekurze.

6) Řešení kvadratické rovnice $ax^2 + bx + c = 0,\ a \neq 0$

Pro nezáporný diskriminant má rovnice řešení $x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Pro některé hodnoty koeficientů může dojít v aritmetice s pohyblivou desetinnou čárkou k přetečení, podtečení a k selhání. Pokud jsou koeficienty velmi malé nebo velmi velké, potom může b^2 nebo $4ac$ podtéct nebo přetéct. Abychom tomu předešli, vydělíme rovnici hodnotou největšího z parametrů. Potom je největší koeficient 1 nebo -1. Ztrátě přesnosti pro $b \doteq \sqrt{b^2 - 4ac}$ zabráníme použitím alternativní formule (pro $c \neq 0$).

Řešení kvadratické rovnice lze určit z formule $x_{1,2} = \dfrac{2c}{-b \mp \sqrt{b^2 - 4ac}}$.

$$\begin{aligned} \frac{2c}{-b \mp \sqrt{b^2 - 4ac}} &= \frac{2c}{-b \mp \sqrt{b^2 - 4ac}} \cdot \frac{-b \pm \sqrt{b^2 - 4ac}}{-b \pm \sqrt{b^2 - 4ac}} \cdot \frac{\frac{1}{2c}}{\frac{1}{2c}} = \\ &= \frac{-b \pm \sqrt{b^2 - 4ac}}{[b^2 - (b^2 - 4ac)] \cdot \frac{1}{2c}} = \\ &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \end{aligned}$$

Uvažujeme rovnici s koeficienty a = 0,05010, b = -98,78, c = 5,015.

```
%-----------------------------------------
% koreny kvadraticke rovnice s koeficienty
    a=0.0501; b=-98.78; c=5.015;
    x=roots([a b c]);
%-----------------------------------------

  x1 = 1971.605916
  x2 = 0.050771
```

Použijeme-li aritmetiku se 4 ciframi, dostaneme

$$\begin{aligned} b^2 - 4ac &= 98{,}78^2 \text{ - } 4{\cdot}0{,}05010{\cdot}5{,}015 \\ &= 9757 \text{ - } 0{,}2004{\cdot}5{,}015 \\ &= 9757 \text{ - } 1{,}005 = 9756 \\ \sqrt{b^2 - 4ac} &= 98{,}77 \end{aligned}$$

Standardní formule:

$$x_{1,2} = \frac{98,78 \pm 98,77}{0,1002} = \begin{cases} \mathbf{1972} \\ 0,0998 \quad !!! \end{cases}$$

Modifikovaná formule:

$$x_{1,2} = \frac{10,03}{98,78 \mp 98,77} = \begin{cases} 1003 \quad !!! \\ \mathbf{0{,}05077} \end{cases}$$

Shrnutí

Ke ztrátě přesnosti může docházet při

- odčítání skoro stejných hodnot
- sčítání nesrovnatelných hodnot
- násobení velkých čísel (přetečení)
- násobení malých čísel (podtečení)
- dělení malým číslem blízkým nule

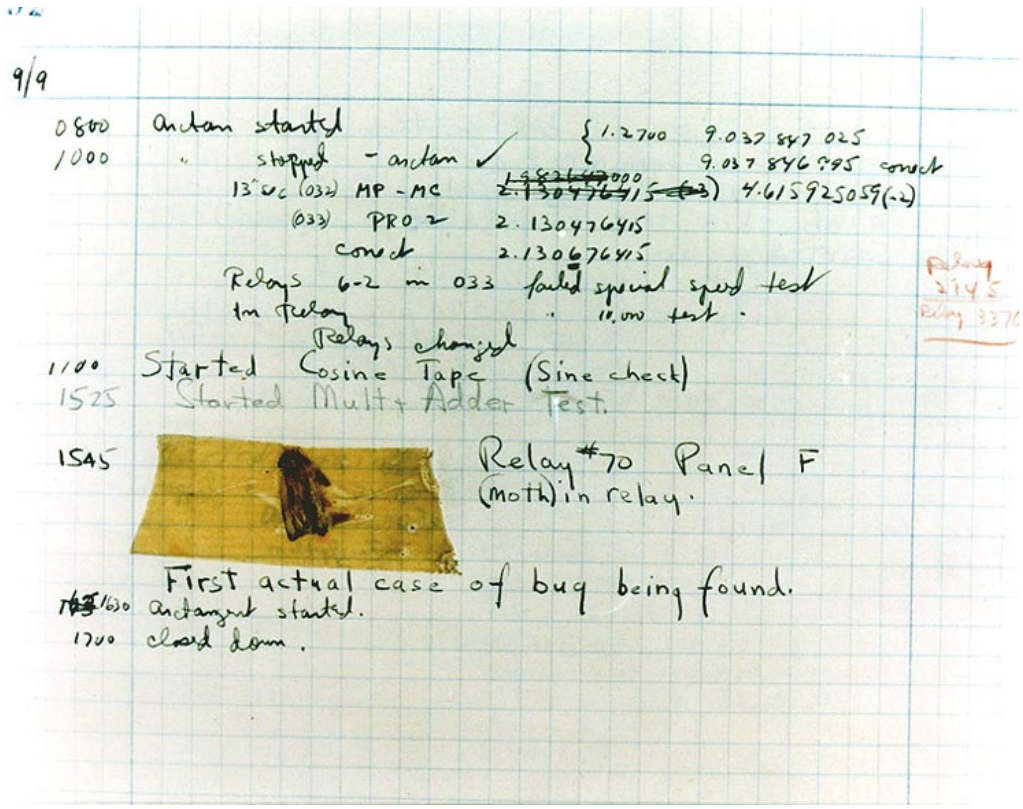
Závažné počítačové chyby (příklady z nedávné historie)

Význam slova **bug**

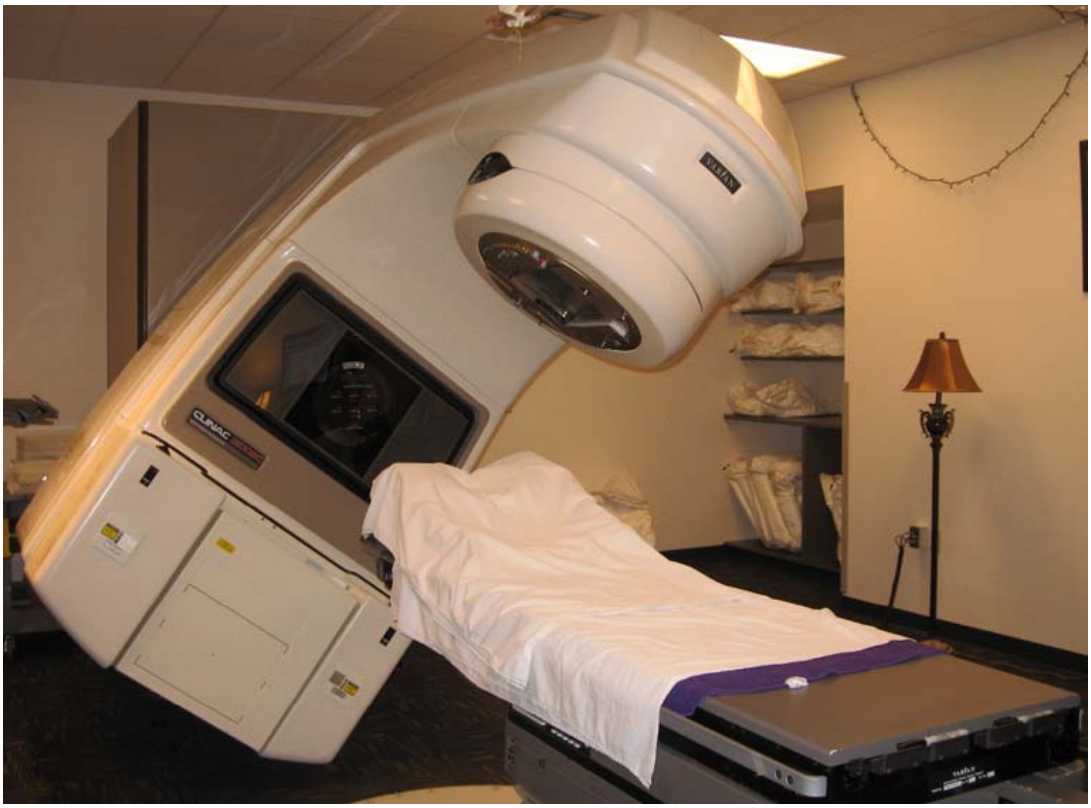
Programátorská chyba se často i v češtině označuje anglickým výrazem bug a proces jejího odstraňování ladění (debugování).

Bug znamená doslova moucha, štěnice nebo obecně brouk. V angličtině se ve významu chyba (například konstruktérská) používá už velmi dlouho – použil ho například Thomas Edison roku 1878, když mluvil o svých vynálezech. S počítači pak pronikl do mnoha dalších jazyků.

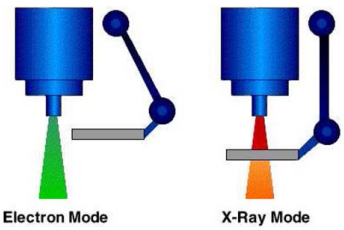
Traduje se, že původem tohoto významu je problém způsobený skutečným hmyzem. Známá je třeba historka o molu zachyceném na relé počítače Mark II dne 9. září 1947. Mol byl pečlivě vyproštěn a nalepen do záznamu s poznámkou "první skutečný případ nalezeného bugu".



Therac-25

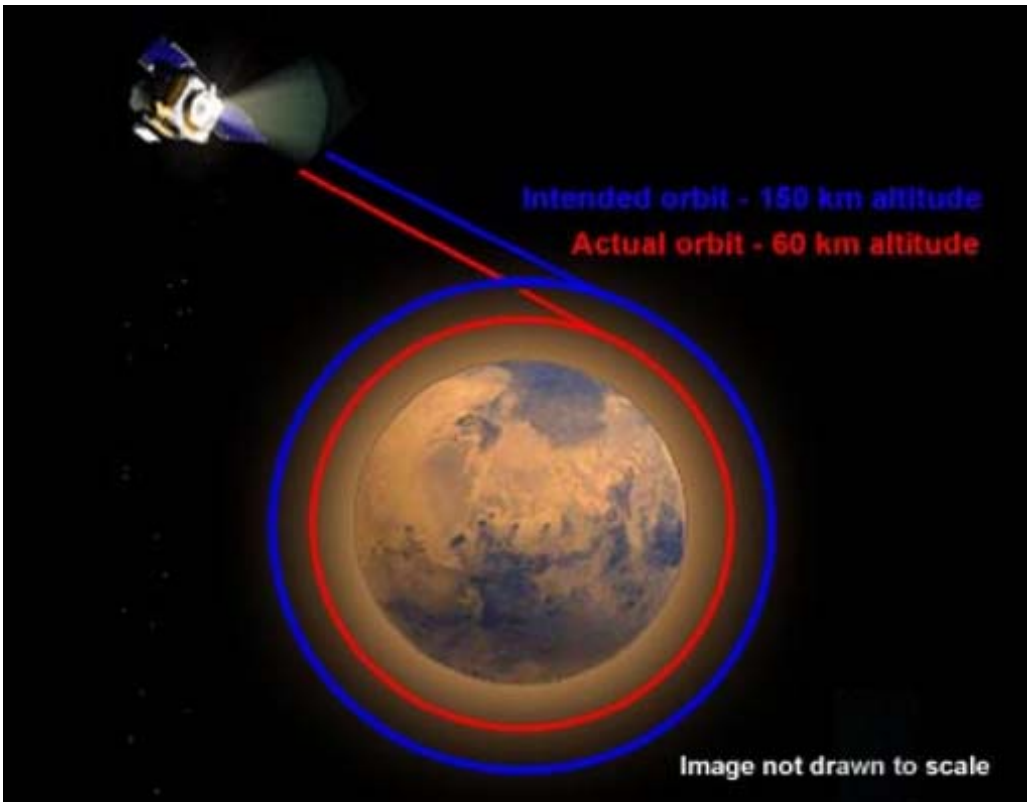


- radiologický přístroj pro léčbu nádorů ozařováním
- červen 1985 - leden 1987 prokázáno min. **6 případů úmrtí**
- ozařoval ve dvou režimech (x-paprsky, elektrony)



- chyby v programu - nedostatečně ošetřené situace
  1. **race condition** (souběh)
     chyba při nastavení parametrů operátorem a jejich rychlé opravě
  2. další problém
     jedna z bezpečnostních funkcí přístroje kontrolovala nastavené parametry a správný stav indikovala 0 v registru, při chybném nastavení hodnotu registru zvětšila o 1
     co se stalo po 256. neúspěšné kontrole? (pro čítač byl vyhrazen 1 Byte)

Mars Climate Orbiter

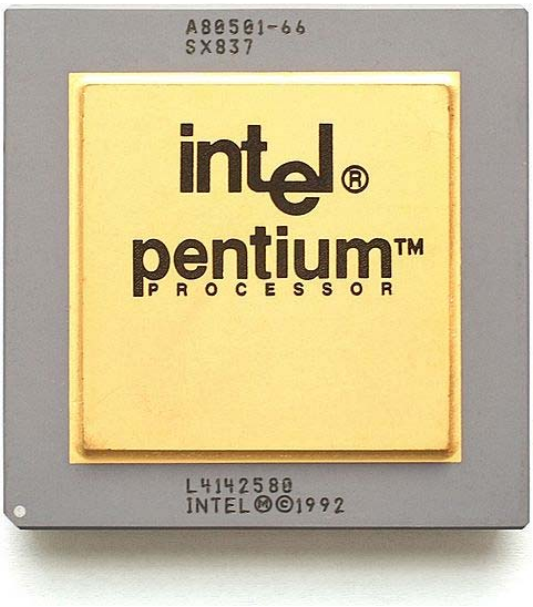


- 11.12.1998 z kosmodromu Eastern Test Rande startuje raketa Delta 7425
- 23.9.1999 dorazil k Marsu satelit Mars Climate Orbiter
- nepodařilo se "zakotvit" jej na plánované oběžné dráze (cca 150 km)
- ve skutečnosti se dostal na jinou oběžnou dráhu (cca 60 km)
  https://cs.wikipedia.org/wiki/Mars_Climate_Orbiter
- příčina
  - tým v řídícím středisku v Coloradu používal anglické měrné jednotky
  - navigační tým v Kalifornii používali metrické
- satelit shoří v atmosféře **cca 300 000 000 USD**

Ariane 5

- nosná raketa ESA, v roce 1996 byla zničena 40 sekud po startu
  https://www.youtube.com/watch?v=N6PWATvLQCY
- následník Ariane 4, používala její sw, avšak Ariane 5 dosahovala při startu 5x většího zrychlení hodnoty se dostaly mimo očekávaný rozsah a při konverzi 64-bitového desetinného čísla na 16-bitové celé číslo došlo **přetečení**
- rutina, která měla situaci řešit, byla z důvodu efektivity vypnuta
- Ariane 5 se sebezničila **cca 370 000 000 USD**

Pentium FDIV bug



- 1994, Professor Thomas Nicely (Lynchburg College, Virginia, USA)
  https://faculty.lynchburg.edu/~nicely/pentbug/pentbug.html
- chyba procesoru Intel P5 Pentium (in floating point unit)
- nesprávné výsledky při dělení desetinných čísel
  https://www.ipsr.ku.edu/stafffil/hoyle/pentium_fdiv
- ruční ověření například pomocí zadání výpočtu

  1. příklad

     správná hodnota $\dfrac{4195835}{3145727} = 1.333820449136241002$

     vypočtená hodnota $\dfrac{4195835}{3145727} = 1.333739068902037589$

  2. příklad

     správná hodnota $\dfrac{3145727 \times 4195835}{3145727} = 4195835$

     vypočtená hodnota $\dfrac{3145727 \times 4195835}{3145727} = 4195579$

- způsobeno nevyplněnou tabulkou v matematickém koprocesoru
- celková škoda **cca 500 000 000 USD**

Aproximace ve výpočtové matematice

Příklad:

Pokud pro výpočet povrchu Země použijeme vzorec $S = 4\pi r^2$, kde r je poloměr Země, použili jsme několik aproximací:

- povrch Země není přesně sféra
- hodnota poloměru je vypočtená
- π má nekonečný rozvoj
- při výpočtu jsou data i výsledky operací zaokrouhlovány

Datová a výpočtová chyba

Typický problém: Určit hodnotu funkce $f : \mathbb{R} \to \mathbb{R}$ v daném bodě x.

přesný vstup: x	přesný výstup: $f(x)$
nepřesný vstup: $\widehat{x}$	nepřesný výstup: $\widehat{f}(\widehat{x})$

Celková chyba: $\widehat{f}(\widehat{x}) - f(x) = \underbrace{(\widehat{f}(\widehat{x}) - f(\widehat{x}))}_{\text{výpočtová chyba}} + \underbrace{(f(\widehat{x}) - f(x))}_{\text{datová chyba}}$

Příklad: Přibližně vyčíslete hodnotu $\sin\frac{\pi}{8}$.

přesný vstup: $x = \frac{\pi}{8} \doteq 0,39270$	přesný výstup: $f(x) = \sin\frac{\pi}{8} \doteq 0,38268$
nepřesný vstup: $\widehat{x} = \frac{3,14}{8} = 0,39250$	nepřesný výstup: $\sin\frac{3,14}{8} \doteq 0,38250 = \widehat{f}(\widehat{x})$

Celková chyba: $0,38250 - 0,38268 = \underbrace{(0,38250 - \sin\frac{3,14}{8})}_{\text{výpočtová chyba}} + \underbrace{(\sin\frac{3,14}{8} - \sin\frac{\pi}{8})}_{\text{datová chyba}}$

Výpočtová chyba:	Chyba vyčíslení funkční hodnoty pro pevný argument
Datová chyba:	Rozdíl funkčních hodnot přesného a přibližného vstupu

Výpočtová chyba má 2 složky

- chyba metody (aproximace)
- zaokrouhlovací chyba

Absolutní a relativní chyba

x ... teoretická (přesná) hodnota

$\widehat{x}$ ... aproximace

Absolutní chyba: $\varepsilon = \widehat{x} - x \quad \Rightarrow \quad \widehat{x} = x + \varepsilon$

$\varepsilon = |\widehat{x} - x|$

Relativní chyba $(x \neq 0)$: $\delta = \dfrac{|(\widehat{x} - x)|}{|x|} = \dfrac{\text{absolutní chyba}}{\text{přesná hodnota}}$

Př.:

$\lvert 3,1418 - 3,1415\rvert = 0,0003$	$\dfrac{0,0003}{3,1415} \doteq 0,0001$
$\lvert 31418 - 31415\rvert = 3$	$\dfrac{3}{31415} \doteq 0,0001$
(Absolutní chyby)	(Relativní chyby)

Platí: $\delta = \dfrac{\widehat{x} - x}{x} \qquad \Rightarrow \qquad \widehat{x} = x(1 + \delta)$

Poznámka: V praxi potřebujeme odhadnout chybu

estimate $|chyba| \approx \ldots$

bound $|chyba| \leq \ldots$

Příklad: Výpočet hodnoty funkce kosinus pro argument blízký $\dfrac{\pi}{2}$.

$x \approx \dfrac{\pi}{2} \qquad h > 0$ ... malá perturbace (chyba)

absolutní chyba:

$$\cos(x + h) - \cos(x) = -2 \sin \frac{2x + h}{2} \sin \frac{h}{2} \approx -2 \sin x \cdot \frac{h}{2} = -h \cdot \sin x \approx -h$$

relativní chyba:

$$\left| \frac{\cos(x + h) - \cos x}{\cos x} \right| \approx \left| \frac{-h \sin x}{\cos x} \right| = | - h \tan x| \approx \infty$$

$\Rightarrow$ malá změna x na vstupu vyvolá velkou relativní změnu na výstupu a to nezávisle na použité metodě výpočtu.

```
Vypocet hodnoty funkce f(x)=cos(x) pro hodnotu argumentu
x blizkou pi/2. Uvazujeme, ze se ve vyjadreni x dopoustime
male chyby (perturbace) h>0.

Zadej hodnotu argumentu x (blizke pi/2) = 1.57078
Zadej hodnotu male zmeny (perturbace) h = 0.00001
-----------------------------------------------------------

Absolutni chyba (na vystupu)
  cos(x+h) = 0.00000633
  cos(x) = 0.00001633
  cos(x+h) - cos(x) = -0.00001000

Absolutni chyba (na vstupu)
 (x+h) - x = h = 0.00001000

-----------------------------------------------------------

Relativni chyba (na vystupu)
  ( cos(x+h) - cos(x) ) / cos(x) = -0.612490

Relativni chyba (na vstupu)
 ( (x+h) - x ) / x  = 0.00000637

-----------------------------------------------------------

Pomer Absolutni chyby na vystupu / Absolutni chyby na vstupu
       je 1.00000000

Pomer Relativni chyby na vystupu / Relativni chyby na vstupu
       je 96208.717628

Pozn.:  Posledni pomer nazyvame cislo podminenosti.
```

$$\textbf{číslo podmíněnosti} = \frac{|\text{relativní chyba na výstupu}|}{|\text{relativní chyba na vstupu}|} = \frac{0,61249}{0,637 \cdot 10^{-5}} \doteq \mathbf{0,96 \cdot 10^5}$$

Předchozí problém byl špatně podmíněný, neboť číslo podmíněnosti bylo $\gg 1$, tj. malá relativní změna vstupních dat způsobila velkou relativní změnu výstupních dat.

Podmíněnost úlohy řešit SLAR

Uvažujeme soustavu $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A}$ ... $n \times n$ regulární, $\mathbf{b} \in \mathbb{R}^n$.

Označení:

$\Delta\mathbf{A}$ ... malá změna matice $\mathbf{A}$

$\Delta\mathbf{b}$ ... malá změna vektoru $\mathbf{b}$

$\Delta\mathbf{x}$ ... odpovídající změna vektoru neznámých

$\mathbf{x}^*$ ... přesné řešení soustavy $\mathbf{Ax} = \mathbf{b}$

Platí:

$$(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x}^* + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$$

Uvažujme situaci $\underline{\Delta\mathbf{A} = \mathbf{0}}$, tj. $\mathbf{A}$ je zadána přesně

Otázka: Jakou změnu řešení vyvolá změna pravé strany?

$$\underline{\mathbf{Ax}^*} + \mathbf{A}\Delta\mathbf{x} = \underline{\mathbf{b}} + \Delta\mathbf{b}$$

$$\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$$

$$\Delta\mathbf{x} = \mathbf{A}^{-1}\Delta\mathbf{b}$$

Z vlastností maticové normy plyne:

$$\mathbf{Ax}^* = \mathbf{b} \;\Rightarrow\; \|\mathbf{b}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}^*\| \;\Rightarrow\; \frac{1}{\|\mathbf{x}^*\|} \leq \frac{\|\mathbf{A}\|}{\|\mathbf{b}\|}$$

$$\Delta\mathbf{x} = \mathbf{A}^{-1}\Delta\mathbf{b} \;\Rightarrow\; \|\Delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{b}\|$$

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}^*\|} \leq \frac{\|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{b}\| \cdot \|\mathbf{A}\|}{\|\mathbf{b}\|}$$

$$C_p = \frac{\dfrac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}^*\|}}{\dfrac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}} \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

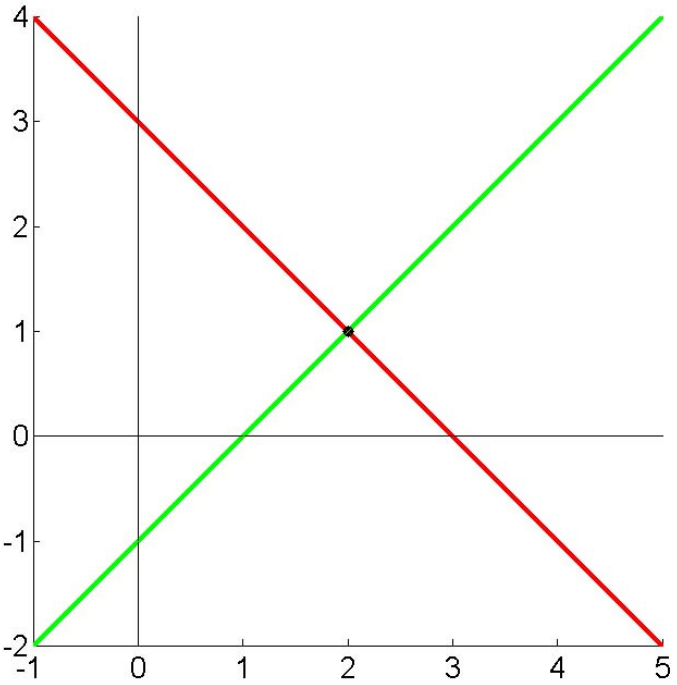
Případ, kdy $\underline{\Delta\mathbf{b} = \mathbf{0}}$, tj. $\mathbf{b}$ je zadána přesně, i obecný případ $\Delta\mathbf{b} \neq \mathbf{0}$, $\Delta\mathbf{A} \neq \mathbf{0}$ vede na stejné vyjádření čísla podmíněnosti.

Poznámka: Pro symetrické matice je číslo podmíněnosti podíl největší a nejmenší absolutní hodnoty vlastního čísla (pro symetrické matice platí, že mají reálná vlastní čísla).

$$C_p = \underbrace{\|\mathbf{A}^{-1}\|}_{\frac{1}{\lambda_{min}}} \cdot \underbrace{\|\mathbf{A}\|}_{\lambda_{max}} = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

Geometrická interpretace - dobře podmíněná úloha
(malá relativní změna vstupních dat vyvolá malou relativní změnu výstupních dat)

$$\boxed{\begin{aligned} x+y&=3\\ x-y&=1\end{aligned}} \quad \Rightarrow \quad \left.\begin{aligned} y&=3-x\\ y&=x-1\end{aligned}\right\} \quad \text{řešení} \quad \begin{aligned} x^*&=2\\ y^*&=1\end{aligned}$$



$$\mathbf{A}=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad \mathbf{b}=\begin{bmatrix}3\\ 1\end{bmatrix}$$

$$C_p = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

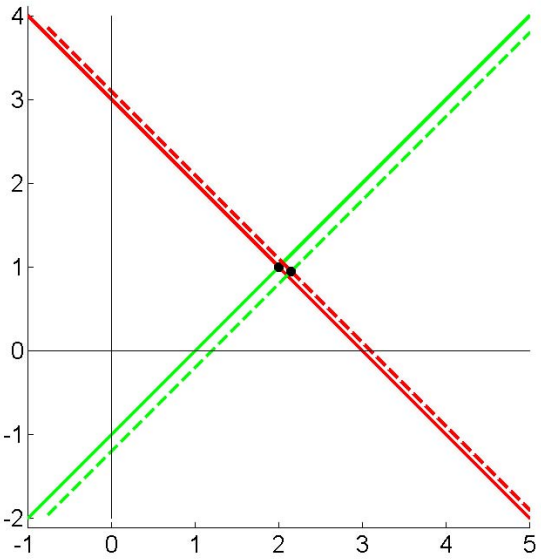
$$\mathbf{A}^{-1}=\begin{bmatrix}0,5 & 0,5\\ 0,5 & -0,5\end{bmatrix}=0,5\,\mathbf{A}$$

$C_p = 1 \cdot 2 = 2$ (řádková, sloupcová norma); $\quad C_p = \dfrac{1}{\sqrt{2}}\sqrt{2} = 1$ (spektrální norma)

$$\left(\mathbf{A}^T\mathbf{A}=\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix} \quad \|\mathbf{A}\|_{SP}=\sqrt{2}, \quad \|\mathbf{A}^{-1}\|_{SP}=\frac{1}{\sqrt{2}}\right)$$

1. změna pravé strany (změna matice $\Delta\mathbf{A} = \mathbf{0}$)

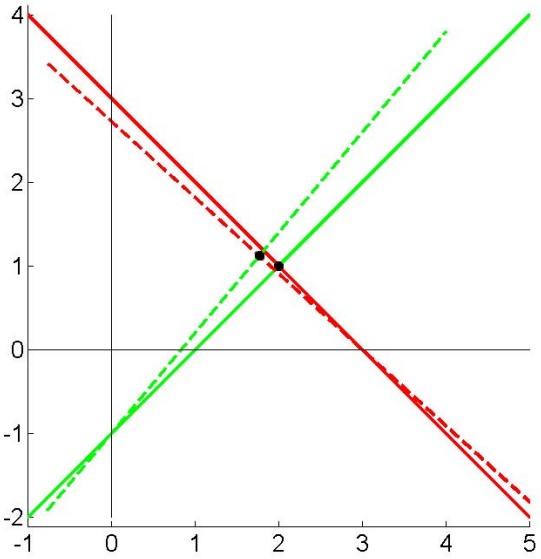
$$\Delta\mathbf{b} = \begin{bmatrix} 0,1 \\ 0,2 \end{bmatrix}, \quad \mathbf{b} + \Delta\mathbf{b} = \begin{bmatrix} 3,1 \\ 1,2 \end{bmatrix}$$



změněná soustava $y = 3,1 - x$; $y = x - 1,2$

2. změna matice soustavy (změna pravé strany $\Delta\mathbf{b} = \mathbf{0}$)

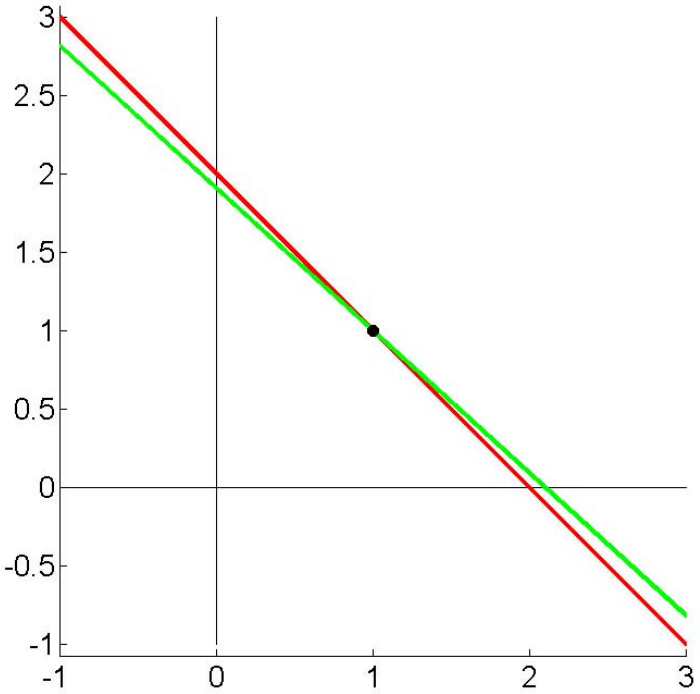
$$\Delta\mathbf{A} = \begin{bmatrix} 0 & 0,1 \\ 0,2 & 0 \end{bmatrix}, \quad \mathbf{A} + \Delta\mathbf{A} = \begin{bmatrix} 1 & 1,1 \\ 1,2 & -1 \end{bmatrix}$$



změněná soustava $y = \dfrac{1}{1,1}(3 - x)$; $y = 1,2\,x - 1$

Geometrická interpretace - špatně podmíněná úloha
(malá relativní změna vstupních dat vyvolá velkou relativní změnu výstupních dat)

$$\boxed{\begin{matrix} x+y=2 \\ x+1,1\,y=2,1 \end{matrix}} \quad \Rightarrow \quad \left.\begin{matrix} y=2-x \\ y=\dfrac{1}{1,1}(2,1-x) \end{matrix}\right\} \quad \text{řešení} \quad \begin{matrix} x^*=1 \\ y^*=1 \end{matrix}$$



$$\mathbf{A}=\begin{bmatrix} 1 & 1 \\ 1 & 1,1 \end{bmatrix}, \quad \mathbf{b}=\begin{bmatrix} 2 \\ 2,1 \end{bmatrix}$$

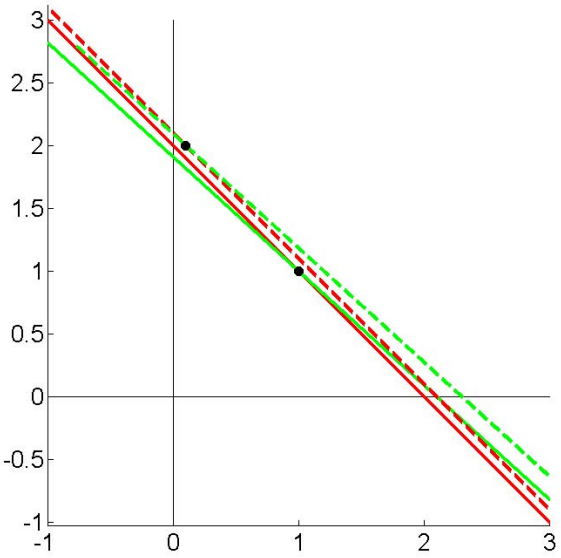
$$C_p = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

$$\mathbf{A}^{-1}=\begin{bmatrix} 11 & -10 \\ -10 & 10 \end{bmatrix}$$

$C_p = 2,1 \cdot 21 = 44,1$ (řádková, sloupcová norma);
$C_p = 2,0512 \cdot 20,5125 = 42,07$ (spektrální norma)

1. <u>změna pravé strany</u> (změna matice $\Delta\mathbf{A} = \mathbf{0}$)

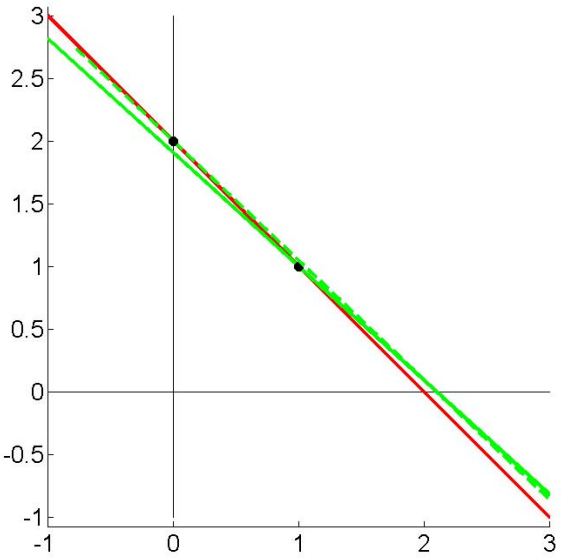
$$\Delta\mathbf{b} = \begin{bmatrix} 0,1 \\ 0,2 \end{bmatrix}, \quad \mathbf{b} + \Delta\mathbf{b} = \begin{bmatrix} 2,1 \\ 2,3 \end{bmatrix}$$



změněná soustava $y = 2,1 - x$; $y = \dfrac{1}{1,1}(2,3 - x)$

2. <u>změna matice soustavy</u> (změna pravé strany $\Delta\mathbf{b} = \mathbf{0}$)

$$\Delta\mathbf{A} = \begin{bmatrix} 0 & 0 \\ 0 & -0,05 \end{bmatrix}, \quad \mathbf{A} + \Delta\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1,05 \end{bmatrix}$$



změněná soustava $y = 2 - x$; $y = \dfrac{1}{1,05}(2,1 - x)$

Praktický výpočet čísla podmíněnosti

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

změna na vstupu ... $\Delta\mathbf{A}$

vyvolá změnu na výstupu ... $\Delta\mathbf{x}$

$$(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b}$$

$$C_p = \frac{\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|}}{\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}}$$

Příklad

$$\mathbf{A} = \begin{bmatrix} 50 & -100 \\ 50 & -101 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

změna matice soustavy

$$\Delta\mathbf{A} = \begin{bmatrix} 0 & 0,1 \\ 0,2 & 0 \end{bmatrix} \quad \Rightarrow \quad \widetilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A} = \begin{bmatrix} 50 & -99,9 \\ 50,2 & -101 \end{bmatrix}$$

$$\widetilde{\mathbf{x}} = \widetilde{\mathbf{A}}^{-1}\mathbf{b} = \begin{bmatrix} 2,8527 \\ 1,4278 \end{bmatrix}$$

$$\Delta\mathbf{x} = \mathbf{x} - \widetilde{\mathbf{x}} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 2,8527 \\ 1,4278 \end{bmatrix} = \begin{bmatrix} -0,8527 \\ -0,4278 \end{bmatrix}$$

$\|.\|$	1.vektorová/sloupcová	maximová/řádková	euklidovská/spektrální
$\Delta\mathbf{x} = \begin{bmatrix} -0,8527 \\ -0,4278 \end{bmatrix}$	1,2805	0,8527	0,9539
$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$	3	2	2,2361
$\Delta\mathbf{A} = \begin{bmatrix} 0 & 0,1 \\ 0,2 & 0 \end{bmatrix}$	0,2	0,2	0,2
$\mathbf{A} = \begin{bmatrix} 50 & -100 \\ 50 & -101 \end{bmatrix}$	201	151	158,7479
C_p	428,97	321,89	338,6

$$\|\mathbf{x}\|_1 = \sum_i |x_i|\,, \quad \|\mathbf{x}\|_\infty = \max_i |x_i|\,, \quad \|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$$

$$\|\mathbf{A}\|_S = \max_k \sum_i |a_{ik}|\,, \quad \|\mathbf{A}\|_R = \max_i \sum_k |a_{ik}|\,, \quad \|\mathbf{A}\|_{SP} = \max_k \lambda_k^{\frac{1}{2}}(A^H A)$$

(Při použití různých norem dostáváme obecně různá čísla podmíněnosti.)

Teoretické číslo podmíněnosti

$$C_p = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

$$\mathbf{A}^{-1} = \begin{bmatrix} 2,02 & -2 \\ 1 & -1 \end{bmatrix}$$

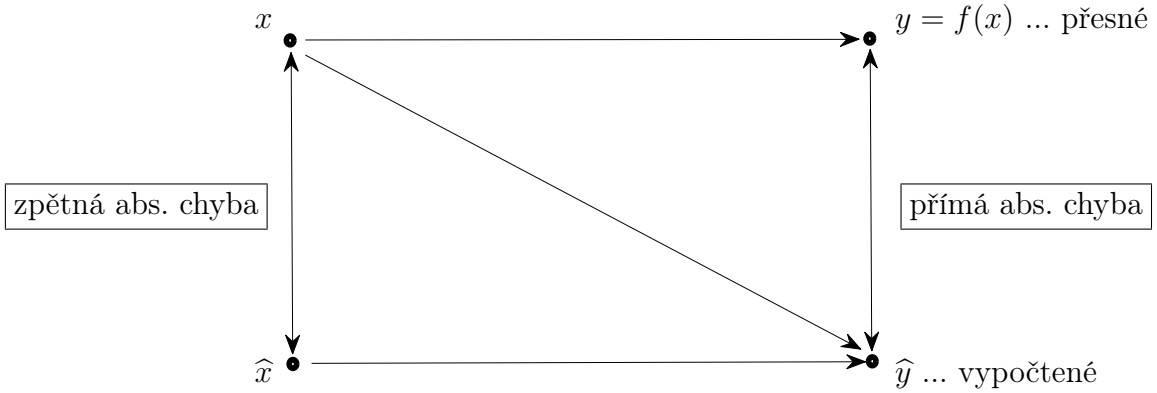
$$C_p = 151 \cdot 4,02 = 607,02 \quad \text{(řádková } \|.\|\text{)};$$

$$C_p = 201 \cdot 3,02 = 607,02 \quad \text{(sloupcová } \|.\|\text{)};$$

$$C_p = 158,7479 \cdot 3,1750 \doteq 504,03 \quad \text{(spektrální } \|.\|\text{)}$$

Přímá a zpětná chyba

$x \to f(x) = y \qquad\qquad f : \mathbb{R} \to \mathbb{R}$



Otázka: $\widehat{y}$ je přesným obrazem f jakého bodu $\widehat{x}$?

přímá chyba (abs) $\qquad \triangle y = \widehat{y} - y$

zpětná chyba (abs) $\qquad \triangle x = \widehat{x} - x \qquad (f(x) = y)$

Příklad:

Jaká je přímá a zpětná chyba pro tato data:

$x = 2,\ f(x) = \sqrt{x},\ y = \sqrt{2},\ \widehat{y} = 1,4$?

přímá chyba:

$$|\triangle y| = |\widehat{y} - y| = |1,4 - 1,41421| \approx 0,01421$$

$$\frac{|\triangle y|}{|y|} \approx 1\%$$

zpětná chyba:

$$\widehat{x} = ?, \sqrt{\widehat{x}} = 1,4, \widehat{x} = 1,96$$

$$|\triangle x| = |\widehat{x} - x| = |1,96 - 2| = 0,04$$

$$\frac{|\triangle x|}{|x|} = 2\%$$

```
Vysvetleni pojmu prima a zpetna chyba na uloze
vypoctu hodnoty funkce f(x)=sqrt(x) pro x = 2.

Zadej, na kolik platnych cifer n budeme zaokrouhlovat
      n = 2
------------------------------------------------------------

 Presne x = 2.000000

 Presne y=f(x) = 1.414214

 Nepresne (zaokrouhlene) Y = 1.400000

   PRIMA absolutni chyba, tj |Y - y| = 0.014214

   PRIMA relativni chyba, tj |Y - y| / |y| = 0.010051

 Jake X se presne zobrazi na Y ?

 X = Y^2 = 1.960000

   ZPETNA absolutni chyba, tj |X - x| = 0.040000

   ZPETNA relativni chyba, tj |X - x| / |x| = 0.020000
```

Podmíněnost a stabilita

$$cond = \frac{|\text{rel. změna řešení}|}{|\text{ rel. změna vstup. dat}|} = \frac{\left|\frac{f(\widehat{x})-f(x)}{f(x)}\right|}{\left|\frac{\widehat{x}-x}{x}\right|} = \frac{\text{rel. přímá chyba}}{\text{rel. zpětná chyba}}$$

- Stabilita algoritmu je analogická s podmíněností. Algoritmus je stabilní, je-li výsledek relativně málo citlivý na chyby vzniklé během výpočtu.
- Přesnost výsledku je blízkost vypočteného a přímého řešení
- Stabilita algoritmu sama o sobě negarantuje přesnost.
- Přesnost závisí na stabilitě algoritmu stejně tak jako na podmíněnosti problému.

stabilita algoritmu + dobrá podmíněnost úlohy $\Rightarrow$ přesnost výsledku

Vraťme se k příkladu, ve kterém jsme řešili kvadratickou rovnici $ax^2 + bx + c = 0$ s koeficienty a = 0,05010, b = -98,78, c = 5,015.

```
-----------------------------------------
 koreny kvadraticke rovnice s koeficienty
 a=0.0501; b=-98.78; c=5.015;
 x=roots([a b c])
 x1 = 1971.605916
 x2 = 0.050771
-----------------------------------------
 aa=a*1.001
 xa=roots([aa b c])
  xa1 = 1969.636229
  xa2 = 0.050771
 rel_zmena_vystup=(xa-x)./x
  -9.9903e-04
   2.5752e-08
-----------------------------------------
 bb=b*1.001
 xb=roots([a bb c])
  xb1 = 1973.577623
  xb2 = 0.050720
 rel_zmena_vystup=(xb-x)./x
   1.0001e-03
  -9.9905e-04
 -----------------------------------------
 cc=c*1.001
 xc=roots([a b cc])
  xc1 = 1971.605865
  xc2 = 0.050821
 rel_zmena_vystup=(xc-x)./x
  -2.5752e-08
   1.0000e-03
-----------------------------------------
```

Úloha najít kořeny této kvadratické rovnice **je dobře podmíněná**, neboť pro relativní chybu na vstupu 0.001 došlo k relativní chybě na výstupu nejvýše ve stejných řádech, tj. `1e-3` (číslo podmíněnosti ≈ 1).

Připomeňme výsledky dvou různých vzorečků (algoritmů) v aritmetice se 4 ciframi.

Standardní formule $x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \dfrac{98,78 \pm 98,77}{0,1002} = \begin{cases} \mathbf{1972} \\ 0,0998 \quad \textbf{!!!} \end{cases}$

Modifikovaná formule $x_{1,2} = \dfrac{2c}{-b \mp \sqrt{b^2 - 4ac}} = \dfrac{10,03}{98,78 \mp 98,77} = \begin{cases} 1003 \quad \textbf{!!!} \\ \mathbf{0,05077} \end{cases}$

Každý z obou algoritmů byl nestabilní pro výpočet jednoho ze dvou kořenů.

Číselné soustavy

Motivace: Pro zobrazování čísel v počítači není vhodná desítková soustava. používá se dvojková soustava. Připomeňme si tedy, jak se převádí čísla mezi různými soustavami.

- zápis v desítkové soustavě:

  $$1563 = (1 \cdot 10^3) + (5 \cdot 10^2) + (6 \cdot 10^1) + (3 \cdot 10^0)$$

  obecně

  $$N = (a_k \cdot 10^k) + (a_{k-1} \cdot 10^{k-1}) + \ldots + (a_1 \cdot 10^1) + (a_0 \cdot 10^0)$$

  $$N \in \mathbb{N}, \quad a_k \in \{0, 1, \ldots, 8, 9\}$$

- zápis ve dvojkové soustavě:

  $$1563 = (1 \cdot 2^{10}) + (1 \cdot 2^9) + (0 \cdot 2^8) + (0 \cdot 2^7) + (0 \cdot 2^6) + (0 \cdot 2^5) + (1 \cdot 2^4) + (1 \cdot 2^3) + \\ + (0 \cdot 2^2) + (1 \cdot 2^1) + (1 \cdot 2^0)$$

  $$(\mathbf{1563})_{\mathbf{10}} = (\mathbf{11000011011})_{\mathbf{2}}$$

Příklad: Převeďte z desítkové soustavy číslo 139 do dvojkové soustavy.

```
------------------------------------------------------------------------
Prevod cisla 139.000000 z 10-soustavy do 2-soustavy na 8 desetinnych mist
Znamenko ............ +
Cela cast ........... 139
Desetinna cast ...... 0.000000
------------------------------------------------------------------------
  prevod_cele_casti =

   139 : 2 = 69 : 2 = 34 : 2 = 17 : 2 = 8 : 2 = 4 : 2 = 2 : 2 = 1
     1        1        0        1       0       0       0

   Cislo 139 v 10-soustave prevedeno do 2-soustavy je 10001011.
```

Zbytky zapíšeme odzadu: $(139)_{10} = (10001011)_2$.

Příklad: Převeďte z desítkové soustavy číslo $\frac{1}{10}$ do dvojkové soustavy.

```
------------------------------------------------------------------------
Prevod cisla 0.100000 z 10-soustavy do 2-soustavy na 6 desetinnych mist
Znamenko ............ +
Cela cast ........... 0
Desetinna cast ...... 0.100000
------------------------------------------------------------------------
  prevod_desetinne_casti =

   0.1 * 2 = 0.2 * 2 = 0.4 * 2 = 0.8 * 2 = 0.6 * 2 = 0.2 * 2 = 0.4
     0         0         0         1         1         0

   Cislo 1.000000e-01 v 10-soustave prevedeno do 2-soustavy je 0.000110.
```

Pokud výsledek byl větší než 1, zapsali jsme jedničku, číslo ořízli a pokračovali dál.

Znovu se opakuje $0{,}2 \cdot 2 \Rightarrow$ ve dvojkové soustavě jde o periodické číslo (má nekonečný rozvoj).

$$(0,1)_{10} = (0,0\overline{0011})_2$$

Poznámka: Mnoho reálných čísel, které lze v desítkové soustavě zapsat pomocí konečného počtu cifer, pro zápis ve dvojkové soustavě vyžaduje cifer nekonečně mnoho.

Do dvojkové soustavy převeďte z desítkové soustavy např. číslo $0,7$.

$$\begin{aligned}
(0,7)_{10} &= (0,1\overline{0110})_2 = 1 \cdot 2^{-1} + \sum_{k=0}^{\infty} 1 \cdot 2^{-(3+4k)} + \sum_{k=0}^{\infty} 1 \cdot 2^{-(4k)} = \\
&= 0,5 + \frac{1}{8} \cdot \sum_{k=0}^{\infty} \left(2^{-4}\right)^k + \frac{1}{16} \cdot \sum_{k=0}^{\infty} \left(2^{-4}\right)^k = \\
&= 0,5 + \frac{1}{8} \cdot \frac{1}{1-\frac{1}{16}} + \frac{1}{16} \cdot \frac{1}{1-\frac{1}{16}} = 0,5 + \frac{2}{15} + \frac{1}{15} = \\
&= 0,7
\end{aligned}$$

Příklad: Převeďte z desítkové soustavy číslo $\frac{4321}{3}$ $(= 1440,\overline{3})$ do sedmičkové soustavy.

```
------------------------------------------------------------------------
Prevod cisla 1440.333333 z 10-soustavy do 7-soustavy na 4 desetinne mista
Znamenko ............ +
Cela cast ........... 1440
Desetinna cast ...... 0.333333
------------------------------------------------------------------------
  prevod_cele_casti =

   1440 : 7 = 205 : 7 = 29 : 7 = 4
      5         2        1

  Cislo 1440 v 10-soustave prevedeno do 7-soustavy je 4125.
------------------------------------------------------------------------
  prevod_desetinne_casti =

   0.33333 * 7 = 0.33333 * 7 = 0.33333 * 7 = 0.33333 * 7 = 0.33333
         2             2             2             2

  Cislo 3.333333e-01 v 10-soustave prevedeno do 7-soustavy je 0.2222.
------------------------------------------------------------------------

  Cislo 1.440333e+03 v 10-soustave prevedeno do 7-soustavy je
  na 4 desetinne mista 4125.2222.
```

Pokud výsledek byl větší než 1, zapsali jsme celou část, číslo ořízli a pokračovali dál.

Znovu se opakuje $0{,}33333 \cdot 7 \Rightarrow$ v sedmičkové soustavě jde také o periodické číslo (má nekonečný rozvoj).

$$(1440,\overline{3})_{10} = (4125,\overline{2})_7$$

Do desítkové soustavy převedeme číslo ze soustavy se základem N velmi jednoduše.

Příklad: Z dvojkové soustavy převeďte do desítkové soustavy číslo 10110010.1001101.

```
-----------------------------------------------------------------
Prevod cisla 10110010.1001101 z 2-soustavy do 10-soustavy.
Cela cast ........... 10110010
Desetinna cast ...... 1001101
-----------------------------------------------------------------

Vysledek = 1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 +
+ 1 * 2^1 + 0 * 2^0 + 1 * 2^(-1) + 0 * 2^(-2) + 0 * 2^(-3) + 1 * 2^(-4) +
+ 1 * 2^(-5) + 0 * 2^(-6) + 1 * 2^(-7)

Vysledek je 178.601562
```

Příklad: Ze čtyřkové soustavy převeďte do desítkové soustavy číslo 2130.211.

```
-----------------------------------------------------------------
Prevod cisla 2130.211 z 4-soustavy do 10-soustavy.
Cela cast ........... 2130
Desetinna cast ...... 211
-----------------------------------------------------------------

Vysledek = 2 * 4^3 + 1 * 4^2 + 3 * 4^1 + 0 * 4^0 + 2 * 4^(-1) +
         + 1 * 4^(-2) + 1 * 4^(-3)

Vysledek je 156.578125
```

Příklad: Z šestkové soustavy převeďte do desítkové soustavy číslo 12345.54321.

```
-----------------------------------------------------------------
Prevod cisla 12345.54321 z 6-soustavy do 10-soustavy.
Cela cast ........... 12345
Desetinna cast ...... 54321
-----------------------------------------------------------------

Vysledek =
1 * 6^4 + 2 * 6^3 + 3 * 6^2 + 4 * 6^1 + 5 * 6^0 + 5 * 6^(-1) +
+ 4 * 6^(-2) + 3 * 6^(-3) + 2 * 6^(-4) + 1 * 6^(-5)

Vysledek je 1865.960005
```

Zobrazení čísel, strojová čísla

Motivace: $\sum_{k=1}^{100000} \frac{1}{10} = 9.998,55664$

- Lidé používají desítkovou soustavu.
- Počítače dvojkovou.

Komunikace s počítačem

- Zadání v 10-soustavě.
- Převod do 2-soustavy (počítač).
- Výpočet (počítač).
- Zpětný převod do 10-soustavy (provádí počítač).
- Výsledek v 10-soustavě.

Binární zlomky

lze vyjádřit jako sumu se zápornými mocninami dvou

$R \in \mathbb{R} \qquad 0 < R < 1 \qquad d_j \in \{0,1\}$

$R = (d_1 \cdot 2^{-1}) + (d_2 \cdot 2^{-2}) + \ldots + (d_n \cdot 2^{-n}) + \ldots$

$R = (0, d_1\, d_2\, \ldots\, d_n\, \ldots)_2$

Zápis čísel

- V desítkové soustavě (vědecká notace)

  $0,000747 = 7,47 \cdot 10^{-4}$

  $313,815 = 3,13815 \cdot 10^{2}$

- Strojová čísla

  normalizovaná pohyblivá řádová čárka (REAL)

  $\mathbf{x} = \pm\, \mathbf{q} \cdot \mathbf{2^n} \qquad \frac{1}{2} \leq q < 1 \ldots$ mantisa, $\quad n \ldots$ exponent

Příklad:

Sestrojte všechna strojová čísla s mantisou délky 4 a exponentem v rozsahu od -3 do 4, tj.

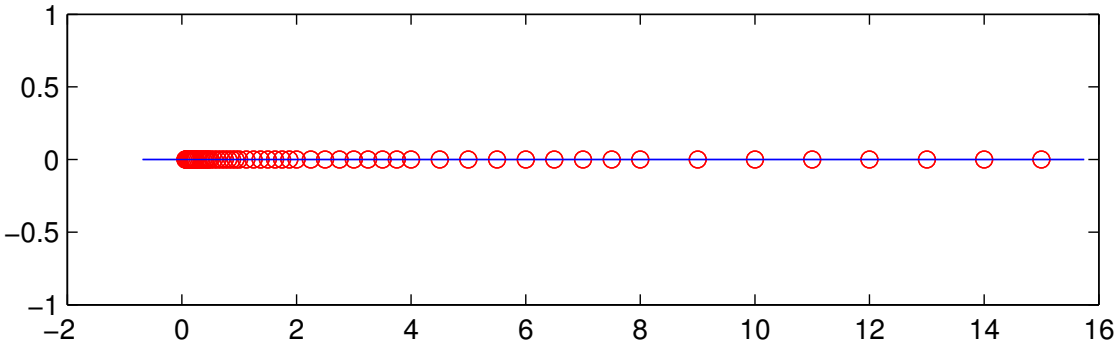
$x = q \cdot 2^n$, kde $q = 0, d_1\, d_2\, d_3\, d_4$, $n \in \{-3, -2, -1, 0, 1, 2, 3, 4\}$

```
------------------------------------------------------------------
 Vygenerovani mnoziny strojovych cisel s mantisou delky 4 a
 exponentem v rozsahu od -3 do 4.
------------------------------------------------------------------
vypis_vsech_cisel =

  Columns 1 through 4

      0.0625       0.125        0.25         0.5
   0.0703125    0.140625     0.28125       0.5625
    0.078125     0.15625      0.3125        0.625
   0.0859375    0.171875     0.34375       0.6875
     0.09375      0.1875       0.375         0.75
   0.1015625    0.203125     0.40625       0.8125
     0.10938     0.21875      0.4375        0.875
   0.1171875    0.234375     0.46875       0.9375

  Columns 5 through 8

           1            2            4            8
       1.125         2.25          4.5            9
        1.25          2.5            5           10
       1.375         2.75          5.5           11
         1.5            3            6           12
       1.625         3.25          6.5           13
        1.75          3.5            7           14
       1.875         3.75          7.5           15
```

Abychom si lépe uvědomili jakou mantisou a jakým exponentem je určeno získané číslo, uvedeme si je v následující tabulce.

q \ n	-3	-2	-1	0	1	2	3	4
0.1000_2	0,0625	0,125	0,25	0,5	1	2	4	8
0.1001_2	0.0703125	0,140625	0,28125	0,5625	1,125	2,25	4,5	9
0.1010_2	0.078125	0,15625	0,3125	0,625	1,25	2,5	5	10
0.1011_2	0.0859375	0,171875	0,34375	0,6875	1,375	2,75	5,5	11
0.1100_2	0.09375	0,1875	0,375	0,75	1,5	3	6	12
0.1101_2	0.1015625	0,203125	0,40625	0,8125	1,625	3,25	6,5	13
0.1110_2	0.109375	0,21875	0,4375	0,875	1,75	3,5	7	14
0.1111_2	0.1171875	0,234375	0,46875	0,9375	1,875	3,75	7,5	15

Získaná čísla si je také vhodné vykreslit na číselnou osu, získáme tak přehled o jejich rozložení. Snadno zjistíme, že čísla nejsou rozložena rovnoměrně.



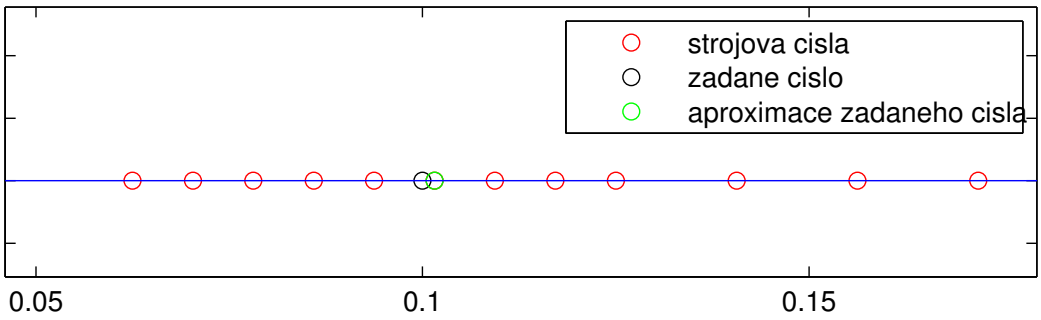
Příklad:

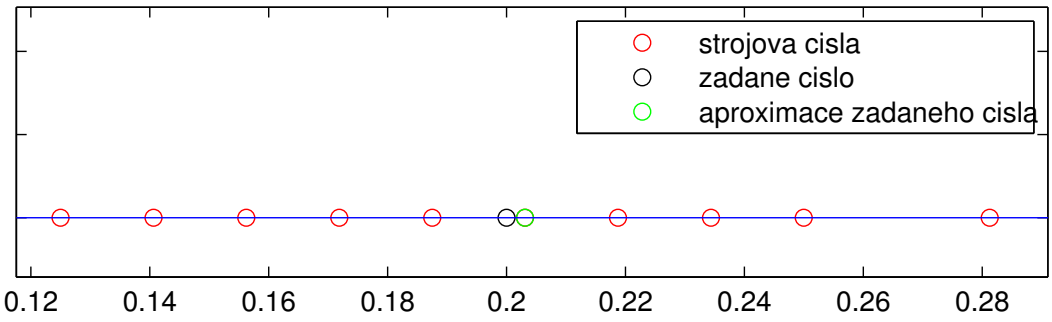
Uvažujme množinu strojových čísel vygenerovanou v předchozím příkladu (tj. strojová čísla s mantisou délky 4 a exponentem v rozsahu od -3 do 4).

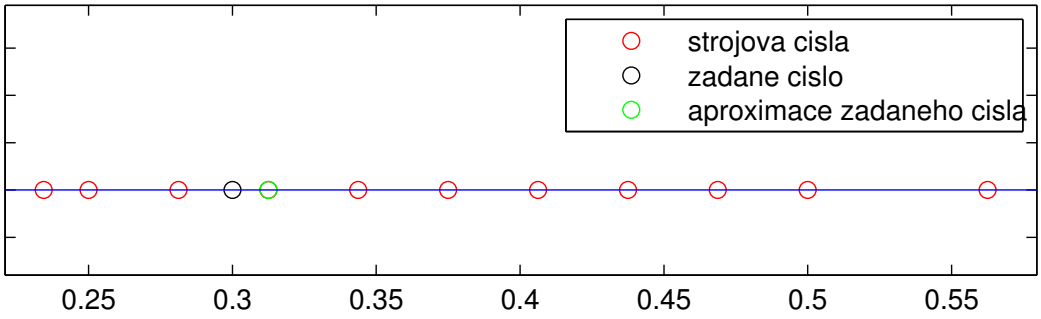
Na několika příkladech si ukažme, na která čísla této množiny se zobrazí zadaná čísla, např. $\frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{1}{6}, \frac{7}{15}$.

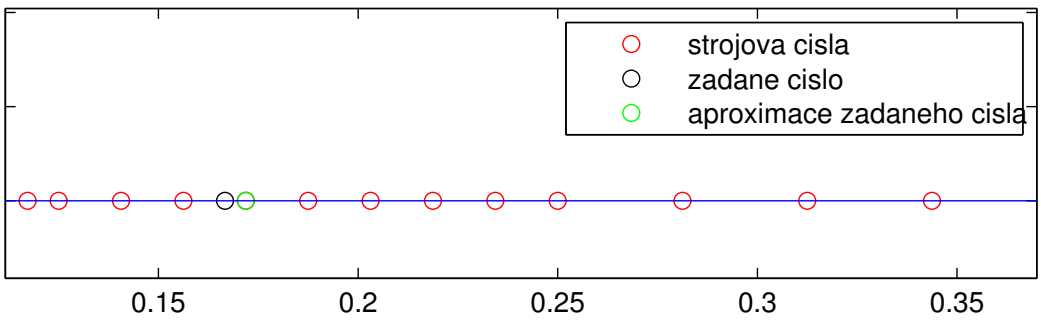
Předpokládáme, že počítač zobrazí číslo na nejbližší číslo, které lze zobrazit, v případě shody na větší.

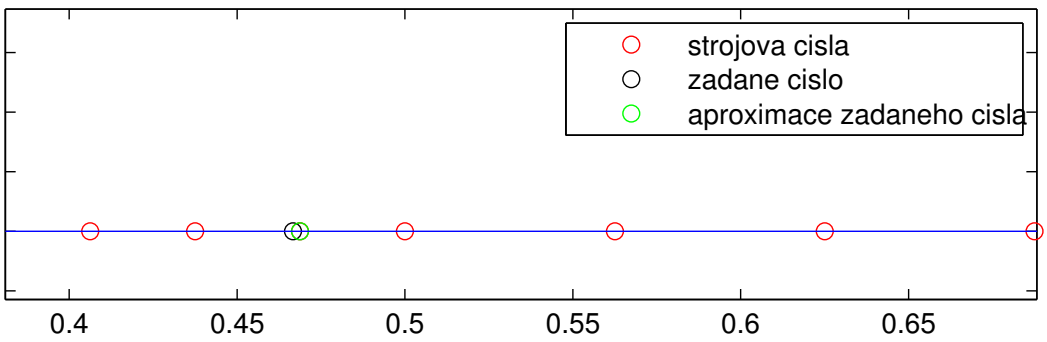
```
------------------------------------------------------------------
 Zobrazeni zadanych cisel do mnoziny strojovych cisel
 s mantisou delky 4 a exponentem v rozsahu od -3 do 4.
------------------------------------------------------------------

 Zobrazuje cisla

cisla =

       1/10
       1/5
       3/10
       1/6
       7/15

 zadane cislo    zapis obrazu        obraz          chyba
------------------------------------------------------------------
  0.10000000     0.1101 x 2^-3     0.10156250    -0.00156250
  0.20000000     0.1101 x 2^-2     0.20312500    -0.00312500
  0.30000000     0.1010 x 2^-1     0.31250000    -0.01250000
  0.16666667     0.1011 x 2^-2     0.17187500    -0.00520833
  0.46666667     0.1111 x 2^-1     0.46875000    -0.00208333
```


strojova cisla
zadane cislo
aproximace zadaneho cisla
0.05
0.1
0.15


strojova cisla
zadane cislo
aproximace zadaneho cisla
0.12
0.14
0.16
0.18
0.2
0.22
0.24
0.26
0.28


strojova cisla
zadane cislo
aproximace zadaneho cisla
0.25
0.3
0.35
0.4
0.45
0.5
0.55


strojova cisla
zadane cislo
aproximace zadaneho cisla
0.15
0.2
0.25
0.3
0.35

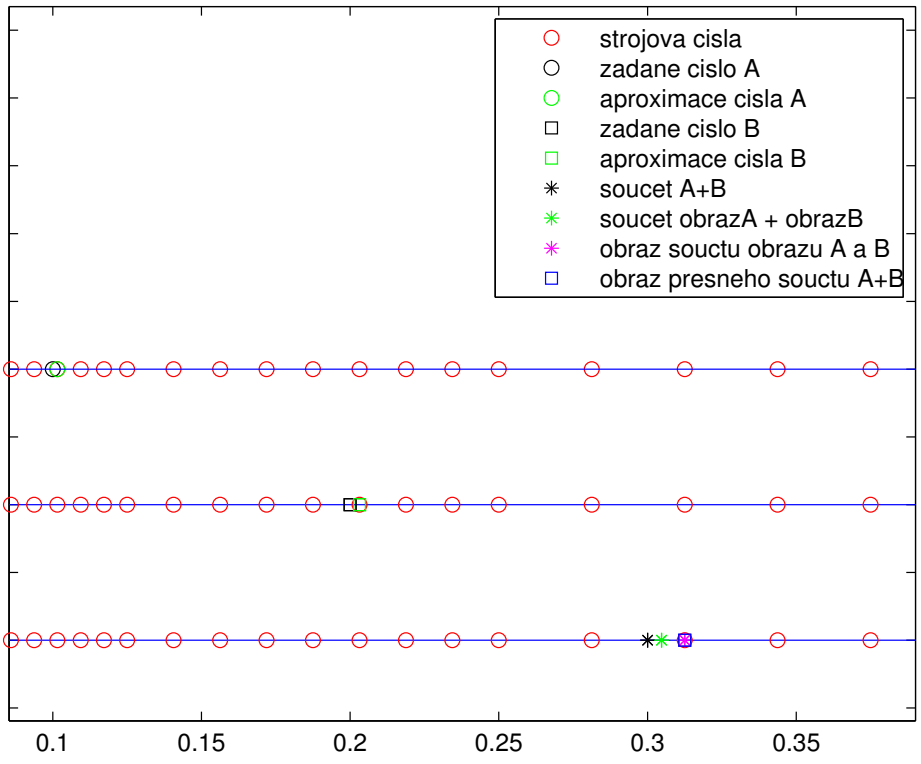

strojova cisla
zadane cislo
aproximace zadaneho cisla
0.4
0.45
0.5
0.55
0.6
0.65

Příklad:

Uvažujme množinu strojových čísel vygenerovanou v předchozím příkladu (tj. strojová čísla s mantisou délky 4 a exponentem v rozsahu od -3 do 4).

Ukažme si, jak se v tomto stroji sečtou čísla $\frac{1}{10}$ a $\frac{1}{5}$.

```
-------------------------------------------------------------
 Zobrazeni souctu cisel A a B v zadane mnozine strojovych cisel
 s mantisou delky M a exponentem v rozsahu od Exp_min do Exp_max
-------------------------------------------------------------
 Cislo A = 0.100000
 Cislo B = 0.200000
 Pocet cisel mantisy M = 4
 Rozsah pro exponent:  od -3 do 4

  cislo              zapis obrazu       obraz
-------------------------------------------------------------
A =  0.10000000      0.1101 x 2^-3      0.10156250
B =  0.20000000      0.1101 x 2^-2      0.20312500
obrazA+obrazB=
     0.30468750      0.1010 x 2^-1      0.31250000
A+B= 0.30000000      0.1010 x 2^-1      0.31250000
```

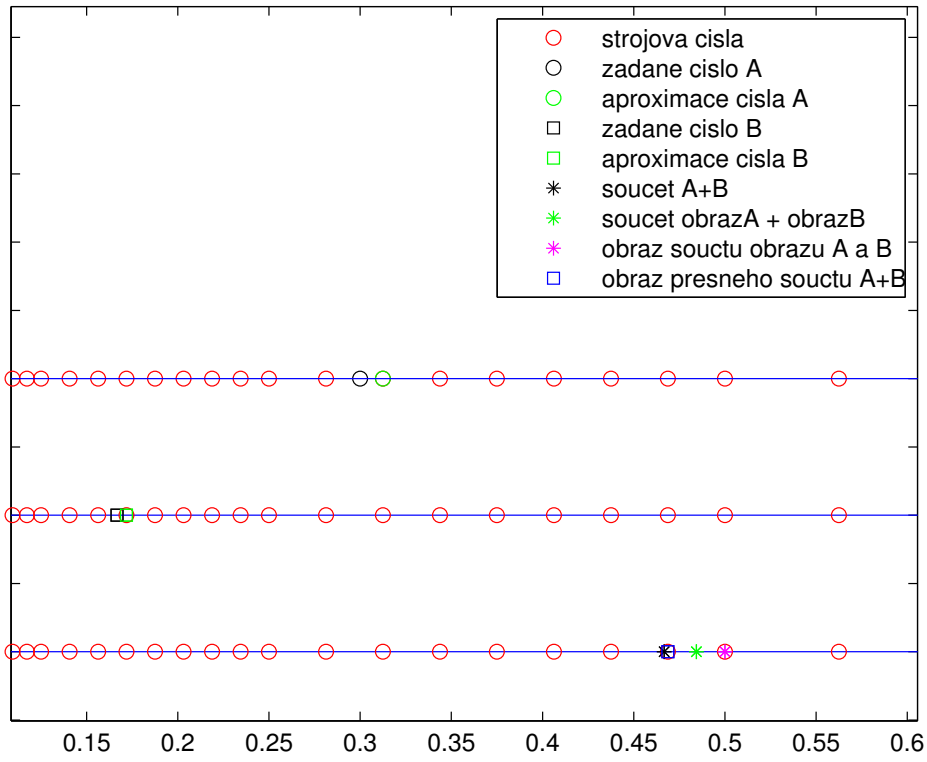


Příklad:

Uvažujme množinu strojových čísel vygenerovanou v předchozím příkladu (tj. strojová čísla s mantisou délky 4 a exponentem v rozsahu od -3 do 4).

Ukažme si, jak se v tomto stroji sečtou čísla $\frac{3}{10}$ a $\frac{1}{6}$.

```
-----------------------------------------------------------------
 Zobrazeni souctu cisel A a B v zadane mnozine strojovych cisel
 s mantisou delky M a exponentem v rozsahu od Exp_min do Exp_max
-----------------------------------------------------------------
 Cislo A = 0.300000
 Cislo B = 0.166667
 Pocet cisel mantisy M = 4
 Rozsah pro exponent:  od -3 do 4

  cislo              zapis obrazu       obraz
-----------------------------------------------------------------
A =  0.30000000      0.1010 x 2^-1      0.31250000
B =  0.16666667      0.1011 x 2^-2      0.17187500
obrazA+obrazB=
     0.48437500      0.1000 x 2^0       0.50000000
A+B= 0.46666667      0.1111 x 2^-1      0.46875000
```



Poznámka:

V předposledním příkladě se shodovali obraz přesného výsledku s obrazem součtu obrazů jednotlivých sčítanců.

V posledním příkladě se obraz přesného výsledku s obrazem součtu obrazů jednotlivých sčítanců neshodoval !

Chyba výpočtu v posledním příkladě:

$$\frac{7}{15} - 0,1000_2 \cdot 2^0 = \frac{14-15}{30} = -\frac{1}{30} = -0,0\overline{3}$$

Relativně:

$$\frac{\frac{1}{30}}{\frac{7}{15}} = \frac{1}{14} = \mathbf{7,14}\,\% \qquad !!!$$

Přesnost počítače

- Vymezíme-li pro mantisu 24 bitů, získáme 7 desetinných míst ($2^{24} = 16777216$).
- Vymezíme-li pro mantisu 53 bitů, získáme 16 desetinných míst ($2^{53} = 9007199254740992$).

Základní formáty:

Formát	Bytes	Bitů pro mantisu	Bitů pro exponent
Single	4	24	8
Double	8	53	11

https://en.wikipedia.org/wiki/Single-precision_floating-point_format

https://en.wikipedia.org/wiki/Double-precision_floating-point_format

Příklad:

- Uvážíme formát SINGLE , tj. 24 bitů pro mantisu.

  $\frac{1}{10} = 0,0\overline{0011}_2 \approx 0,1100\,1100\,1100\,1100\,1100\,1100_2 \cdot 2^{-3}$.

  Chyba zobrazení je $0,\overline{1100}_2 \cdot 2^{-27} (= \frac{1}{10} \cdot 2^{-24}) \approx 5,96 \cdot 10^{-9}$.

- Máme-li počítat $\sum_{k=1}^{100000} \frac{1}{10} = 9.998,55664$.

  Musí být chyba větší než $100000 \cdot 5,96 \cdot 10^{-9} = \mathbf{5,96 \cdot 10^{-4}}$.

Ve skutečnosti je chyba ještě větší, neboť se v průběhu výpočtu musí částečně suma zaokrouhlovat dolů nebo nahoru, jak suma roste, později přičítaná čísla $\frac{1}{10}$ jsou oproti sumě menší a jsou tedy počítány s menší přesností (viz následující příklad).

Příklad: Ve formátu SINGLE sečtěte čísla 10000 a 0,1.

```
------------------------------------------------------------------------
Prevod cisla 10000 z 10-soustavy do 2-soustavy na 0 desetinnych mist
Cela cast ........... 10000
Desetinna cast ...... 0.000000
------------------------------------------------------------------------
prevod_cele_casti =

10000 : 2 = 5000 : 2 = 2500 : 2 = 1250 : 2 = 625 : 2 = 312 : 2 =
    0          0          0          0         1         0


= 156 : 2 = 78 : 2 = 39 : 2 = 19 : 2 = 9 : 2 = 4 : 2 = 2 : 2 = 1
    0        0        1        1       1       0       0

Cislo 10000 v 10-soustave prevedeno do 2-soustavy je 10011100010000.
```

$$(10000)_{10} = (10011100010000)_2 = 0,1001110001 \cdot 2^{14}$$

$$(2^{14} = 16384)$$

$$\begin{aligned}
10000 \quad &\ldots \quad 0,1001\,1100\,0100\,0000\,0000\,0000 \cdot 2^{14} \\
0{,}1 \quad &\ldots \quad 0,1100\,1100\,1100\,1100\,1100\,1100 \cdot 2^{-3} \\
0{,}1 \text{ po SHIFTu} \quad &\ldots \quad 0,0000\,0000\,0000\,0000\,0110\,0110 \cdot 2^{14} \\
&= (01100110)_2 \cdot 2^{-24} \cdot 2^{14} = (64+32+4+2) \cdot 2^{-10} = \\
&= \frac{102}{1024} \doteq 0,099609375
\end{aligned}$$

$$10000 + 0{,}1 \quad \ldots \quad 0,1001\,1100\,0100\,0000\,0110\,0110 \cdot 2^{14}$$

Číslo 10000 je zobrazeno přesně.

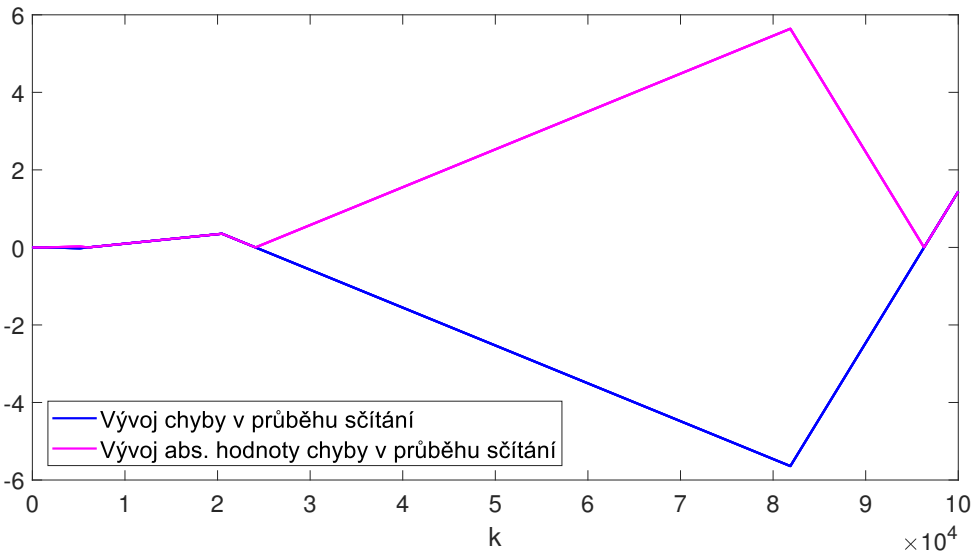
Chyba zobrazení 0,1 po SHIFTU je $\frac{1}{10} - \frac{102}{1024} \doteq 0{,}1 - 0,099609375 = 3,90625 \cdot 10^{-4}$

Shrnutí:

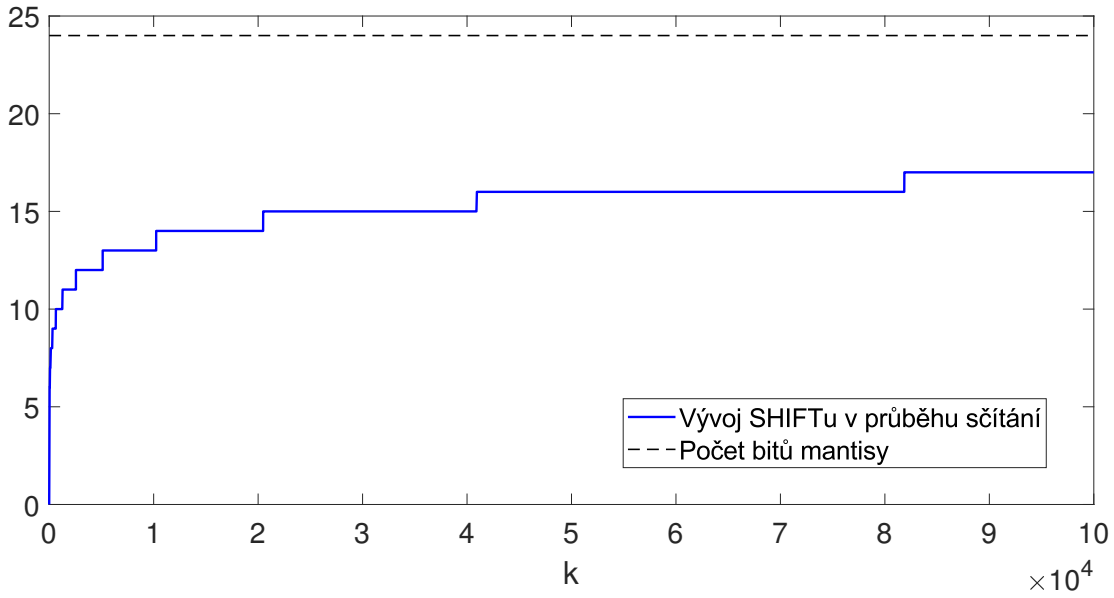
$$10000 + 0{,}1 \;\rightarrow\; \text{výsledek s chybou } 3,90625 \cdot 10^{-4}$$

(v sumě z motivačního příkladu jde o jeden krok)

Podrobnější analýzou získáme graf s vyčíslenou chybou, resp. absolutní hodnotou chyby.



Následující obrázek ukazuje SHIFT exponentu menšího sčítance v průběhu sumace.

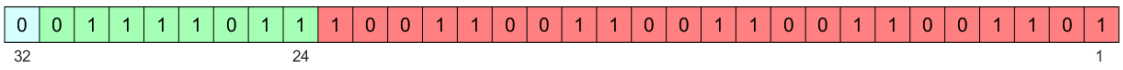


Poznámka: Pokud použijeme formát DOUBLE, dostaneme přesnější výsledky, ovšem v principu bude situace obdobná, pouze se nepřesnosti projeví později.

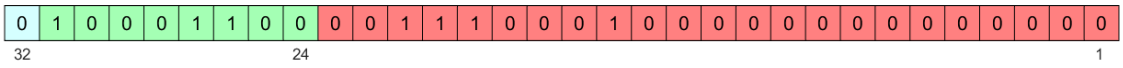
Poznámka: Existují postupy, jak eliminovat chybu při sumaci, např. https://en.wikipedia.org/wiki/Kahan_summation_algorithm

Znázornění čísel ve formátu SINGLE

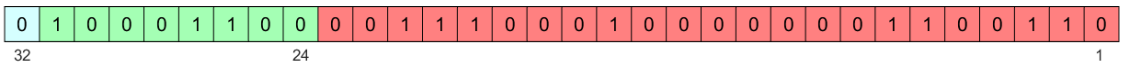
0,1



10 000



10 000,1



Iterační principy

Na iteračním principu je založeno velké množství metod ať z oblasti numerické matematiky, tak z dalších oblastí matematiky.

Příklad: Najděte kladné řešení rovnice $x^2 + x - 2 = 0$.

1. možnost

Úlohu lze řešit použitím vzorce pro řešení kvadratické rovnice $ax^2 + bx + c = 0$, tj.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

v našem případě vyjde:

$$x_{1,2} = \frac{-1 \pm \sqrt{1 + 4 \cdot 2}}{2 \cdot 1} = \frac{-1 \pm 3}{2} = \begin{cases} \mathbf{1} \\ -2 \end{cases}$$

Odpověď: Kladné řešení rovnice je $\alpha = 1$.

Tento výpočet patří mezi přímé metody, tj. teoreticky po konečném počtu operací jsme nalezli přesné řešení. Je třeba si uvědomit, že existuje jen málo problémů, pro které máme k dispozici vzorec a lze je tedy řešit přímo.

2. možnost

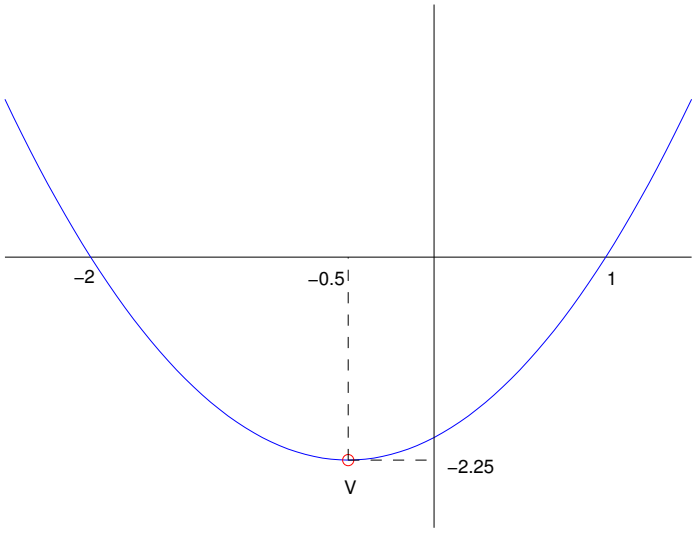
Pro řešení můžeme použít nějakou iterační metodu.

Jak vypadá graf funkce $y = x^2 + x - 2$?

Grafem je parabola, její vrchol určíme pomocí doplnění na čtverec.

$$y = \underbrace{\left(x + \frac{1}{2}\right)^2}_{x^2 + x + \frac{1}{4}} - \frac{1}{4} - 2$$

Protože je koeficient u x^2 kladný, je parabola otevřená nahoru a vrchol odpovídá bodu, ve kterém je funkční hodnota minimální, tj. pro $x = -\frac{1}{2}$ je funkční hodnota $f\left(-\frac{1}{2}\right) = -2,25$.



Princip iteračních metod spočívá v konstrukci posloupnosti přibližných řešení tak, aby přesné řešení bylo limitním případem. Samozřejmě nesmíme zapomenout, že je nutné zadat počáteční aproximaci, a že je nutné iterační postup nějakým způsobem ukončit (po konečném počtu kroků).

Pro řešení našeho příkladu použijeme třeba **metodu půlení intervalu (bisekci)**.

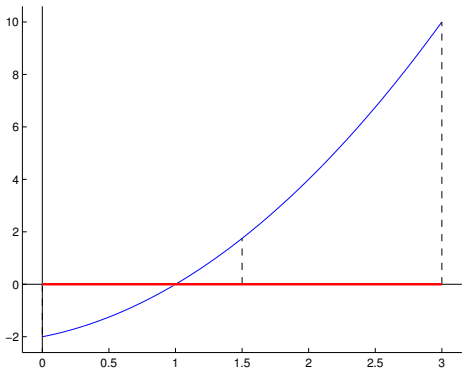
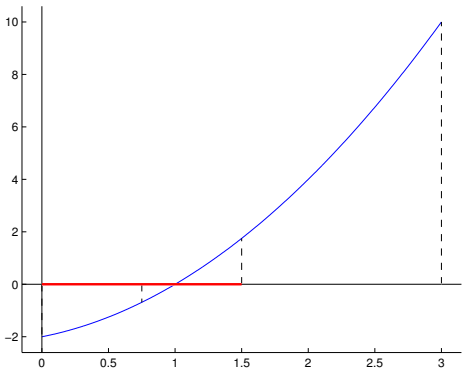
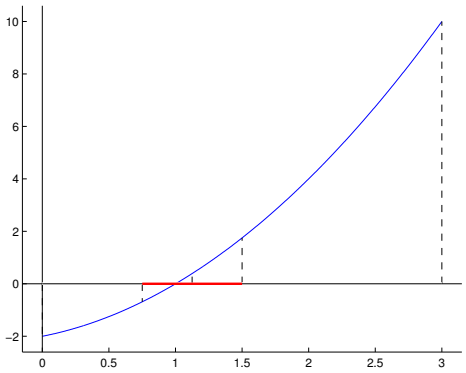
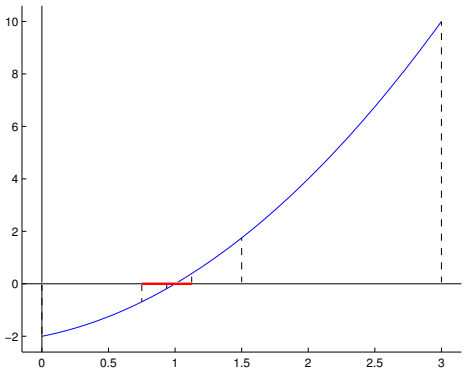
Na začátku musíme zadat interval, na kterém řešení hledáme, tj. vezmeme třeba $\langle 0, 3\rangle$. Musí platit, že funkční hodnoty v krajních bodech intervalu mají opačná znaménka, a že je funkce spojitá. Pokud jsou tyto předpoklady splněny, musí existovat alespoň 1 řešení naší úlohy. V každém kroku vezmeme pouze polovinu původního intervalu tak, aby opět funkční hodnoty v krajních bodech měly opačná znaménka.

Dále musíme výpočet nějakým způsobem ukončit. Můžeme použít některý z následujících způsobů:

- délka intervalu v daném kroku $< \varepsilon$
- absolutní hodnota funkční hodnoty ve středu intervalu $< \varepsilon$
- počet kroků metody $< N$
- ...

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.01

Funkce f(x) je zadana v souboru fce01.m takto:

function out=fce01(x);
out=x^2+x-2;
```

k	a	b	s	f(a)	f(b)	f(s)
0	0.0000	3.0000	1.5000	-2.0000	10.0000	1.7500
1	0.0000	1.5000	0.7500	-2.0000	1.7500	-0.6875
2	0.7500	1.5000	1.1250	-0.6875	1.7500	0.3906
3	0.7500	1.1250	0.9375	-0.6875	0.3906	-0.1836
4	0.9375	1.1250	1.0312	-0.1836	0.3906	0.0947
5	0.9375	1.0312	0.9844	-0.1836	0.0947	-0.0466
6	0.9844	1.0312	1.0078	-0.0466	0.0947	0.0235
7	0.9844	1.0078	0.9961	-0.0466	0.0235	-0.0117
8	0.9961	1.0078	1.0020	-0.0117	0.0235	0.0059
9	0.9961	1.0020	0.9990	-0.0117	0.0059	-0.0029

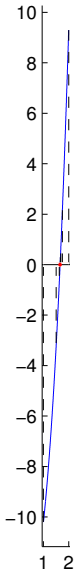





Poznámka: Použitím různých podmínek zastavíme algoritmus obecně v různých krocích.

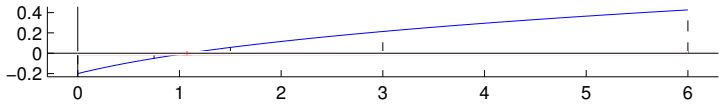
- Pokud bychom použili podmínku, že velikost intervalu v daném kroce je menší než 0,1, pak skončíme v 5. kroku ... $1,0312 - 0,9375 < 0,1$.
- Pokud bychom použili podmínku, že funkční hodnota uprostřed intervalu je v abs. hodnotě menší než 0,1, pak skončíme ve 4. kroku ... $f(1,0312) = 0,0947 < 0,1$.

Otázka: Je lepší použít podmínku na délku intervalu nebo na funkční hodnotu?

Pokud je funkce monotónní a hodně strmá (má velkou absolutní hodnotu derivace), pak se algoritmus zastaví při použití podmínky na velikost intervalu dříve než při použití podmínky na funkční hodnotu, tj. přesnější výsledek získáme použitím podmínky na funkční hodnotu.



Pokud je funkce monotónní a velmi pomalu roste nebo klesá (abs. hodnota derivace je blízká 0), pak se algoritmus zastaví při použití podmínky na velikost intervalu později než při použití podmínky na abs. hodnotu funkce uprostřed intervalu.



```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <1,2> a zastavovaci podminku b-a<0.01
a soucasne pro zastavovaci podminku abs(f(s))<0.01

Funkce f(x) je zadana v souboru fce02.m takto:

function out=fce02(x);
out=0.3*(x+1)^4-15;
```

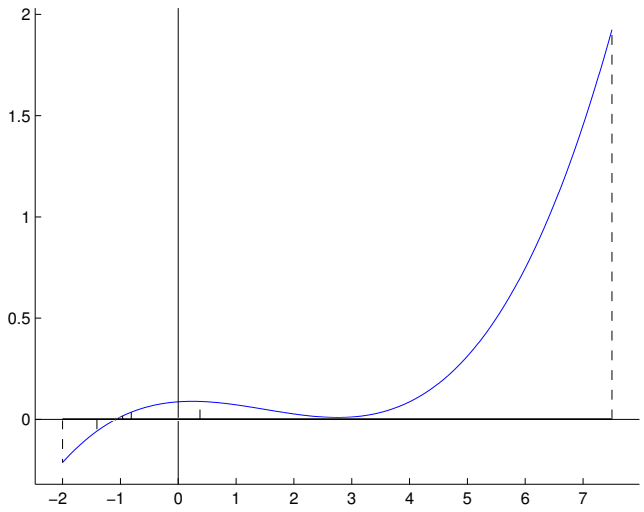

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <1,2> a zastavovaci podminku b-a<0.01
a soucasne pro zastavovaci podminku abs(f(s))<0.01

Funkce f(x) je zadana v souboru fce02.m takto:

function out=fce02(x);
out=0.3*(x+1)^4-15;
```


```
| k |    a    |    b    |    s    |   f(a)  |   f(b)  |   f(s)  |
-------------------------------------------------------------------
| 0 |  1.0000 |  2.0000 |  1.5000 |-10.2000 |  9.3000 | -3.2812 |
| 1 |  1.5000 |  2.0000 |  1.7500 | -3.2812 |  9.3000 |  2.1574 |
| 2 |  1.5000 |  1.7500 |  1.6250 | -3.2812 |  2.1574 | -0.7558 |
| 3 |  1.6250 |  1.7500 |  1.6875 | -0.7558 |  2.1574 |  0.6500 |
| 4 |  1.6250 |  1.6875 |  1.6562 | -0.7558 |  0.6500 | -0.0653 |
| 5 |  1.6562 |  1.6875 |  1.6719 | -0.0653 |  0.6500 |  0.2892 |
| 6 |  1.6562 |  1.6719 |  1.6641 | -0.0653 |  0.2892 |  0.1112 |
| 7 |  1.6562 |  1.6641 |  1.6602 | -0.0653 |  0.1112 |  0.0228 |

 Zastaveni vypoctu pro podminku na     b-a < 0.010000

| 8 |  1.6562 |  1.6602 |  1.6582 | -0.0653 |  0.0228 | -0.0213 |
| 9 |  1.6582 |  1.6602 |  1.6592 | -0.0213 |  0.0228 |  0.0007 |

 Zastaveni vypoctu pro podminku na  abs(f(s)) < 0.010000

 Delka posledniho intervalu je               b-a = 0.001953
 Abs. hodnota funkcni hodnoty v bode s je    f(s)= 0.000716
```

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,6> a zastavovaci podminku b-a<0.01
a soucasne pro zastavovaci podminku abs(f(s))<0.01

Funkce f(x) je zadana v souboru fce03.m takto:

function out=fce03(x);
out=(x+1)^(1/4)-1.2;

| k |    a    |    b    |    s    |   f(a)  |   f(b)  |   f(s)  |
-------------------------------------------------------------------
| 0 |  0.0000 |  6.0000 |  3.0000 | -0.2000 |  0.4266 |  0.2142 |
| 1 |  0.0000 |  3.0000 |  1.5000 | -0.2000 |  0.2142 |  0.0574 |
| 2 |  0.0000 |  1.5000 |  0.7500 | -0.2000 |  0.0574 | -0.0498 |
| 3 |  0.7500 |  1.5000 |  1.1250 | -0.0498 |  0.0574 |  0.0074 |

 Zastaveni vypoctu pro podminku na  abs(f(s)) < 0.010000

| 4 |  0.7500 |  1.1250 |  0.9375 | -0.0498 |  0.0074 | -0.0202 |
| 5 |  0.9375 |  1.1250 |  1.0312 | -0.0202 |  0.0074 | -0.0062 |
| 6 |  1.0312 |  1.1250 |  1.0781 | -0.0062 |  0.0074 |  0.0007 |
| 7 |  1.0312 |  1.0781 |  1.0547 | -0.0062 |  0.0007 | -0.0027 |
| 8 |  1.0547 |  1.0781 |  1.0664 | -0.0027 |  0.0007 | -0.0010 |
| 9 |  1.0664 |  1.0781 |  1.0723 | -0.0010 |  0.0007 | -0.0002 |
|10 |  1.0723 |  1.0781 |  1.0752 | -0.0002 |  0.0007 |  0.0002 |

 Zastaveni vypoctu pro podminku na     b-a < 0.010000

 Delka posledniho intervalu je               b-a = 0.005859
 Abs. hodnota funkcni hodnoty v bode s je    f(s)= 0.000231
```

Pokud funkce není monotónní, nelze dopředu říci, použitím které zastavovací podmínky získáme přesnější aproximaci řešení.

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <-2,7.5> a zastavovaci podminku b-a<0.01
a soucasne pro zastavovaci podminku abs(f(s))<0.01

Funkce f(x) je zadana v souboru fce04.m takto:

function out=fce04(x);
out=0.01*(x+1)*(x-2.5)*(x-3)+0.012;

| k |    a    |    b    |    s    |   f(a)  |   f(b)  |   f(s)  |
-------------------------------------------------------------------
| 0 | -2.0000 |  7.5000 |  2.7500 | -0.2130 |  1.9245 |  0.0097 |

 Zastaveni vypoctu pro podminku na  abs(f(s)) < 0.010000

| 1 | -2.0000 |  2.7500 |  0.3750 | -0.2130 |  0.0097 |  0.0887 |
| 2 | -2.0000 |  0.3750 | -0.8125 | -0.2130 |  0.0887 |  0.0357 |
| 3 | -2.0000 | -0.8125 | -1.4062 | -0.2130 |  0.0357 | -0.0579 |
| 4 | -1.4062 | -0.8125 | -1.1094 | -0.0579 |  0.0357 | -0.0042 |
| 5 | -1.1094 | -0.8125 | -0.9609 | -0.0042 |  0.0357 |  0.0174 |
| 6 | -1.1094 | -0.9609 | -1.0352 | -0.0042 |  0.0174 |  0.0070 |
| 7 | -1.1094 | -1.0352 | -1.0723 | -0.0042 |  0.0070 |  0.0015 |
| 8 | -1.1094 | -1.0723 | -1.0908 | -0.0042 |  0.0015 | -0.0013 |
| 9 | -1.0908 | -1.0723 | -1.0815 | -0.0013 |  0.0015 |  0.0001 |
|10 | -1.0908 | -1.0815 | -1.0862 | -0.0013 |  0.0001 | -0.0006 |

 Zastaveni vypoctu pro podminku na     b-a < 0.010000
```



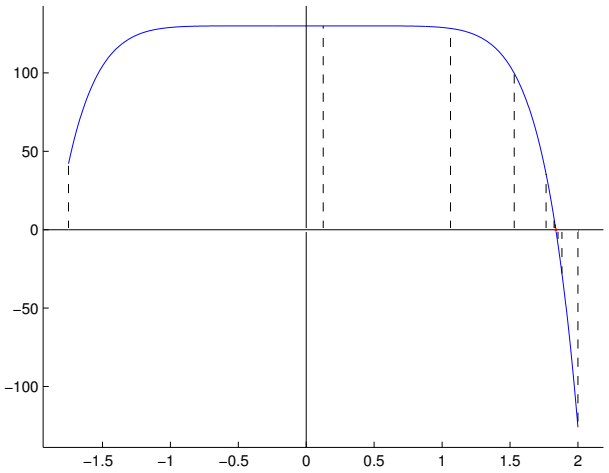
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <-1.75,2> a zastavovaci podminku b-a<0.01
a soucasne pro zastavovaci podminku abs(f(s))<0.01

Funkce f(x) je zadana v souboru fce05.m takto:

function out=fce05(x);
out=130-x^8;
```

k	a	b	s	f(a)	f(b)	f(s)
0	-1.7500	2.0000	0.1250	42.0361	-126.0000	130.0000
1	0.1250	2.0000	1.0625	130.0000	-126.0000	128.3758
2	1.0625	2.0000	1.5312	128.3758	-126.0000	99.7748
3	1.5312	2.0000	1.7656	99.7748	-126.0000	35.5531
4	1.7656	2.0000	1.8828	35.5531	126.0000	-27.9271
5	1.7656	1.8828	1.8242	35.5531	-27.9271	7.3647
6	1.8242	1.8828	1.8535	7.3647	-27.9271	-9.3061
7	1.8242	1.8535	1.8389	7.3647	-9.3061	-0.7383
8	1.8242	1.8389	1.8315	7.3647	-0.7383	3.3699
9	1.8315	1.8389	1.8352	3.3699	-0.7383	1.3301

```
 Zastaveni vypoctu pro podminku na     b-a < 0.010000
```

10	1.8352	1.8389	1.8370	1.3301	-0.7383	0.2995
11	1.8370	1.8389	1.8380	0.2995	-0.7383	-0.2185
12	1.8370	1.8380	1.8375	0.2995	-0.2185	0.0407
13	1.8375	1.8380	1.8377	0.0407	-0.2185	-0.0888
14	1.8375	1.8377	1.8376	0.0407	-0.0888	-0.0240
15	1.8375	1.8376	1.8376	0.0407	-0.0240	0.0084

```
 Zastaveni vypoctu pro podminku na  abs(f(s)) < 0.010000
```

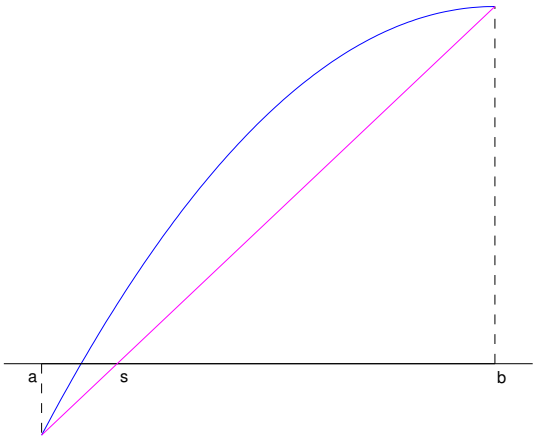


Poznámka:

Z obrázků je patrné, že chyba našeho přibližného řešení může být daleko menší než použité ε, ale také mnohem větší než použité ε.

Chyba řešení se obecně nerovná číslu ε, které jsme použili v zastavovací podmínce !

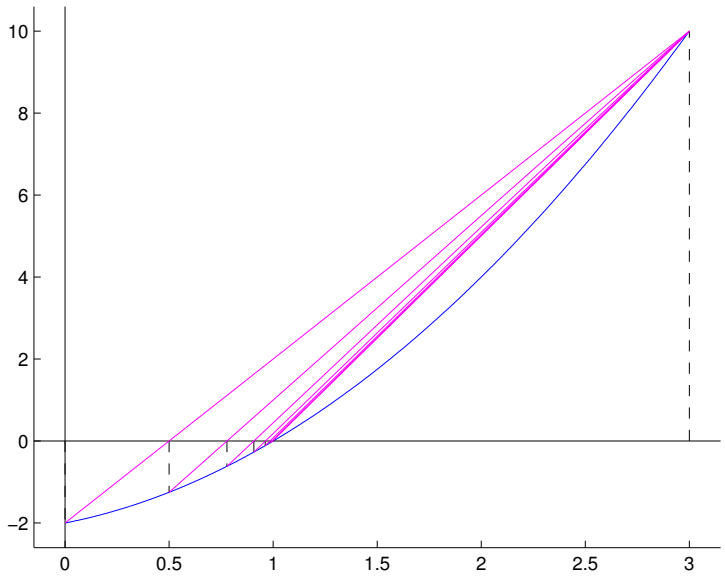
Ukážeme si jinou metodu na řešení. Naším cílem je rychleji zmenšovat interval, viz obrázek:



Metoda se jmenuje **regula falsi** a algoritmus je stejný jako u bisekce, ovšem jako bod s nebereme střed intervalu, ale průsečík tětivy s osou x.

Pokud bychom použili metodu regula falsi na zadání z úvodního příkladu, dostali bychom:

```
Metoda regula falsi pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku |f(s)|<0.01

Funkce f(x) je zadana v souboru fce01.m takto:

function out=fce01(x);
out=x^2+x-2;

| k |    a    |    b    |    s    |   f(a)   |   f(b)   |   f(s)   |
-------------------------------------------------------------------
| 0 |  0.0000 |  3.0000 |  0.5000 | -2.0000 | 10.0000 | -1.2500 |
| 1 |  0.5000 |  3.0000 |  0.7778 | -1.2500 | 10.0000 | -0.6173 |
| 2 |  0.7778 |  3.0000 |  0.9070 | -0.6173 | 10.0000 | -0.2704 |
| 3 |  0.9070 |  3.0000 |  0.9621 | -0.2704 | 10.0000 | -0.1123 |
| 4 |  0.9621 |  3.0000 |  0.9847 | -0.1123 | 10.0000 | -0.0456 |
| 5 |  0.9847 |  3.0000 |  0.9939 | -0.0456 | 10.0000 | -0.0184 |
| 6 |  0.9939 |  3.0000 |  0.9975 | -0.0184 | 10.0000 | -0.0074 |
```

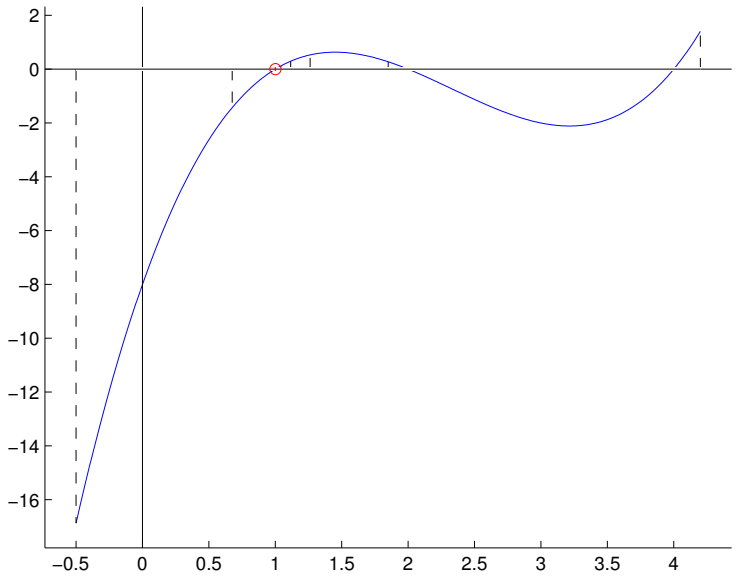


Z obrázku je patrné, že pro zastavení nelze použít podmínku na délku intervalu v daném kroku, protože tato délka pro počet kroků jdoucí do ∞ neklesá k nule. Tj. musíme použít podmínku na abs. hodnotu funkční hodnoty v bodě s.

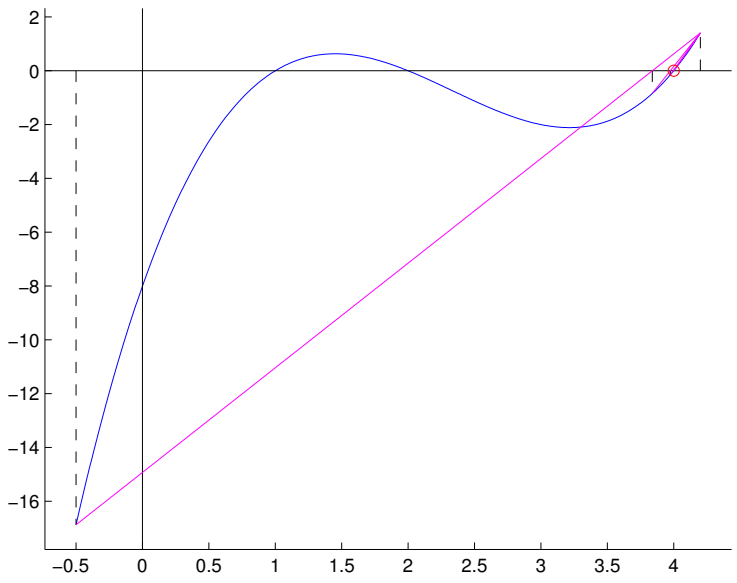
Otázka: Pokud existuje více řešení, tak které najdu?
(Ať už metodou bisekce nebo metodou regula falsi).

- Nelze dopředu říci bez dalších informací.
- Každá z metod může nalézt jiný z nich pro stejná vstupní data.

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <-0.5,4.2> a zastavovaci podminku b-a<0.01

Funkce f(x) je zadana v souboru fce06.m takto:

function out=fce06(x);
out=(x-1)*(x-2)*(x-4);

| k |    a    |    b    |    s    |   f(a)   |   f(b)   |   f(s)   |
-------------------------------------------------------------------
| 0 | -0.5000 |  4.2000 |  1.8500 |-16.8750 |  1.4080 |  0.2741 |
| 1 | -0.5000 |  1.8500 |  0.6750 |-16.8750 |  0.2741 | -1.4318 |
| 2 |  0.6750 |  1.8500 |  1.2625 | -1.4318 |  0.2741 |  0.5300 |
| 3 |  0.6750 |  1.2625 |  0.9688 | -1.4318 |  0.5300 | -0.0977 |
| 4 |  0.9688 |  1.2625 |  1.1156 | -0.0977 |  0.5300 |  0.2949 |
| 5 |  0.9688 |  1.1156 |  1.0422 | -0.0977 |  0.2949 |  0.1195 |
| 6 |  0.9688 |  1.0422 |  1.0055 | -0.0977 |  0.1195 |  0.0163 |
| 7 |  0.9688 |  1.0055 |  0.9871 | -0.0977 |  0.0163 | -0.0393 |
| 8 |  0.9871 |  1.0055 |  0.9963 | -0.0393 |  0.0163 | -0.0112 |
| 9 |  0.9963 |  1.0055 |  1.0009 | -0.0112 |  0.0163 |  0.0026 |
```



```
Metoda regula falsi pro reseni nelinearni rovnice f(x)=0
na intervalu <-0.5,4.2> a zastavovaci podminku |f(s)|<0.01

Funkce f(x) je zadana v souboru fce06.m takto:

function out=fce06(x);
out=(x-1)*(x-2)*(x-4);

| k |    a    |    b    |    s    |   f(a)   |   f(b)   |   f(s)   |
-------------------------------------------------------------------
| 0 | -0.5000 |  4.2000 |  3.8380 |-16.8750 |  1.4080 | -0.8448 |
| 1 |  3.8380 |  4.2000 |  3.9738 | -0.8448 |  1.4080 | -0.1539 |
| 2 |  3.9738 |  4.2000 |  3.9961 | -0.1539 |  1.4080 | -0.0235 |
| 3 |  3.9961 |  4.2000 |  3.9994 | -0.0235 |  1.4080 | -0.0035 |
```

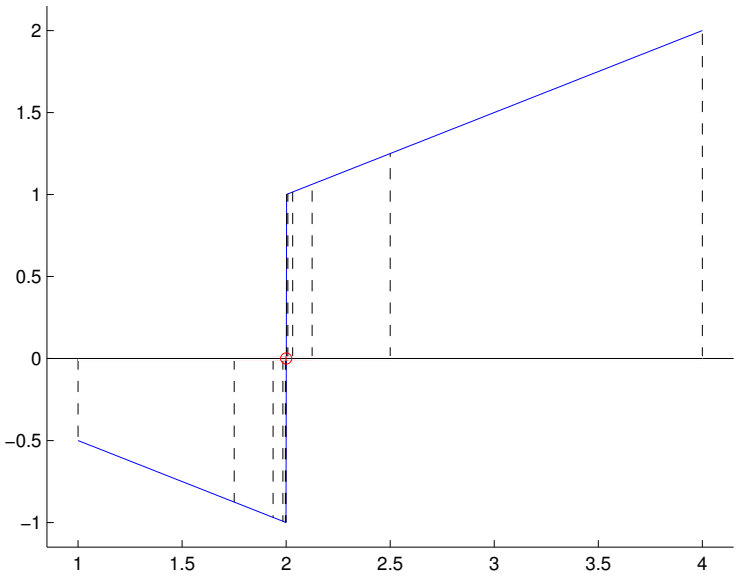


Příklad:

Najděte řešení rovnice $\dfrac{|x^2-2x|}{2x-4}=0$ na intervalu $\langle 1;4\rangle$ pomocí metody půlení intervalu a metody regula falsi.

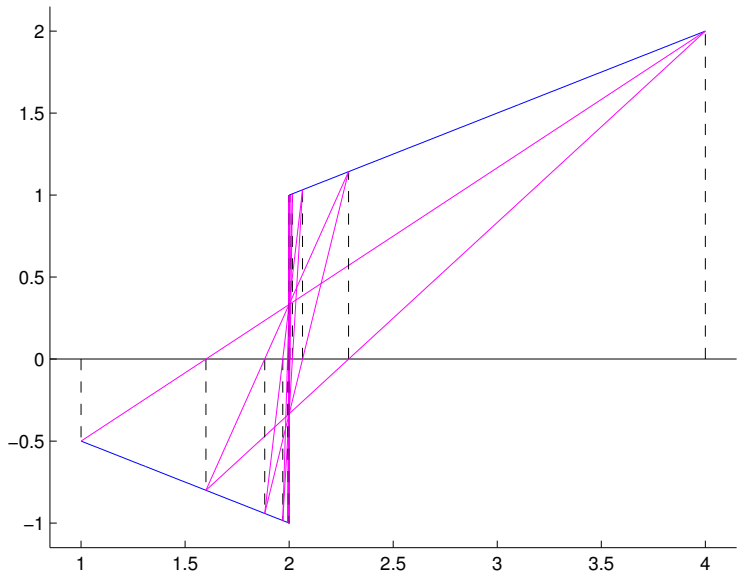
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <1,4> a zastavovaci podminku b-a<0.01

Funkce f(x) je zadana v souboru fce07.m takto:

function out=fce07(x);
out=abs(x^2-2*x)/(2*x-4);
```

k	a	b	s	f(a)	f(b)	f(s)
0	1.0000	4.0000	2.5000	-0.5000	2.0000	1.2500
1	1.0000	2.5000	1.7500	-0.5000	1.2500	-0.8750
2	1.7500	2.5000	2.1250	-0.8750	1.2500	1.0625
3	1.7500	2.1250	1.9375	-0.8750	1.0625	-0.9688
4	1.9375	2.1250	2.0312	-0.9688	1.0625	1.0156
5	1.9375	2.0312	1.9844	-0.9688	1.0156	-0.9922
6	1.9844	2.0312	2.0078	-0.9922	1.0156	1.0039
7	1.9844	2.0078	1.9961	-0.9922	1.0039	-0.9980
8	1.9961	2.0078	2.0020	-0.9980	1.0039	1.0010
9	1.9961	2.0020	1.9990	-0.9980	1.0010	-0.9995



```
Metoda regula falsi pro reseni nelinearni rovnice f(x)=0
na intervalu <1,4> a zastavovaci podminku |f(s)|<0.01

Funkce f(x) je zadana v souboru fce07.m takto:

function out=fce07(x);
out=abs(x^2-2*x)/(2*x-4);
```

k	a	b	s	f(a)	f(b)	f(s)
0	1.0000	4.0000	1.6000	-0.5000	2.0000	-0.8000
1	1.6000	4.0000	2.2857	-0.8000	2.0000	1.1429
2	1.6000	2.2857	1.8824	-0.8000	1.1429	-0.9412
3	1.8824	2.2857	2.0645	-0.9412	1.1429	1.0323
4	1.8824	2.0645	1.9692	-0.9412	1.0323	-0.9846
5	1.9692	2.0645	2.0157	-0.9846	1.0323	1.0079
6	1.9692	2.0157	1.9922	-0.9846	1.0079	-0.9961
7	1.9922	2.0157	2.0039	-0.9961	1.0079	1.0020
8	1.9922	2.0039	1.9980	-0.9961	1.0020	-0.9990
9	1.9980	2.0039	2.0010	-0.9990	1.0020	1.0005
10	1.9980	2.0010	1.9995	-0.9990	1.0005	-0.9998
11	1.9995	2.0010	2.0002	-0.9998	1.0005	1.0001
12	1.9995	2.0002	1.9999	-0.9998	1.0001	-0.9999
13	1.9999	2.0002	2.0001	-0.9999	1.0001	1.0000
14	1.9999	2.0001	2.0000	-0.9999	1.0000	-1.0000
15	2.0000	2.0001	2.0000	-1.0000	1.0000	1.0000
16	2.0000	2.0000	2.0000	-1.0000	1.0000	-1.0000
17	2.0000	2.0000	2.0000	-1.0000	1.0000	1.0000



Shrnutí:

- Bisekce nalezla jako přibližné řešení číslo 1.9990.
- Metoda regula falsi nekonvergovala.

Nebyl splněn předpoklad spojitosti funkce f na $\langle a, b\rangle$ a rovnice neměla řešení.

Příklad z úvodu přednášky zkusíme řešit jiným způsobem.

Pokud má řešení splňovat rovnici

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

musí taky splňovat rovnici

$$x = 2 - x^2.$$

Zvolíme počáteční aproximaci $x_0 = 2,5$ a počítáme další iterace podle vzorce

$$x_{k+1} = 2 - x_k^2.$$

Dostaneme:

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=2.5 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce08.m takto:

function out=fce08(x);
out=2-x^2;

|  k  |            x(k)            |
-------------------------------------
|  0  |     2.500000               |
|  1  |    -4.250000               |
|  2  |    -16.062500              |
|  3  |    -256.003906             |
|  4  |    -65536.000015           |
|  5  |    -4294967296.000000      |
|  6  |    -18446744073709551616   |
   .                 .
   .                 .
   .                 .
|  10 |            -Inf            |
|  11 |            -Inf            |
```

Posloupnost bohužel nekonverguje.

Zkusíme jiný předpis, například

$$x^2 = 2 - x,$$
$$x = +\sqrt{2 - x} \quad \text{(hledáme kladné řešení).}$$

Pokud bychom volili $x_0 = 2,5$, pak bychom dostali pod odmocninou záporné číslo. Zvolíme proto například $x_0 = 1,5$ a další iterace získáme pomocí předpisu

$$x_{k+1} = \sqrt{2 - x_k}.$$

Dostaneme:

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce09.m takto:

function out=fce09(x);
out=sqrt(2-x);

|  k  |       x(k)       |
---------------------------
|  0  |     1.500000     |
|  1  |     0.707107     |
|  2  |     1.137055     |
|  3  |     0.928949     |
|  4  |     1.034916     |
|  5  |     0.982387     |
|  6  |     1.008768     |
|  7  |     0.995606     |
|  8  |     1.002194     |
```

Vidíme, že posloupnost konverguje k přesnému řešení $\alpha = 1$. Opět je třeba zvolit zastavovací podmínku. Výpočet ukončíme, pokud rozdíl dvou po sobě jdoucích iterací bude dostatečně malý, tj.

$$|x_{k+1} - x_k| < \varepsilon.$$

$$|x_8 - x_7| < 0,006588 \leq 0,01 = \varepsilon$$

Pro volbu $\varepsilon = 0,01$ výpočet ukončíme v osmém kroku a za přibližné řešení prohlásíme $x_8 = 1,002194$.

Poslední použitá metoda se nazývá **metoda prosté iterace**.

Metoda prosté iterace

Uvažujeme tzv. úlohu o pevném bodě

$$x = \varphi(x).$$

Řešení úlohy se nazývá pevným bodem zobrazení (funkce) φ.

Pro nalezení řešení jsme zvolili iterační postup, kde jsme na začátku zvolili počáteční aproximaci x_0 a následující iterace počítali podle vzorce

$$x_{k+1} = \varphi(x_k).$$

Položme si otázku, kdy bude proces konvergovat?

Ukážeme si několik příkladů:

Příklad 1: $x_{k+1} = 3x_k - 2,\ x_0 = 1,1$

Řešení musí splňovat rovnici $x = 3x - 2$, tj. $2x = 2 \Rightarrow x = 1$.

S počáteční volbou jsme velmi blízko, vypočteme několik iterací:

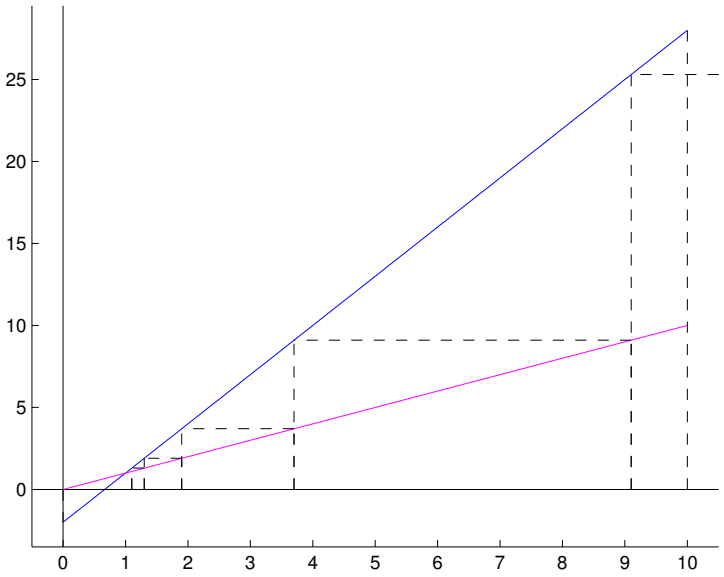
```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.1 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce10.m takto:

function out=fce10(x);
out=3*x-2;
```

k	x(k)	dx(k)=x(k)-x(k-1)	dx(k)/dx(k-1)
0	1.100000		
1	1.300000	0.200000	
2	1.900000	0.600000	3.000000
3	3.700000	1.800000	3.000000
4	9.100000	5.400000	3.000000
5	25.300000	16.200000	3.000000
6	73.900000	48.600000	3.000000
7	219.700000	145.800000	3.000000
8	657.100000	437.400000	3.000000
9	1969.300000	1312.200000	3.000000
10	5905.900000	3936.600000	3.000000

Vidíme, že se od přesného řešení vzdalujeme a pomocí tohoto vztahu jej neurčíme.

Grafické zobrazení: Má platit rovnost $x = \varphi(x)$, tj. přesné řešení odpovídá průsečíku grafů funkcí $y = x$ a $y = \varphi(x)$.

Nejen, že se iterace vzdalují od přesného řešení, ale také se zvětšují rozdíly mezi nimi !



Příklad 2:

Pokud chceme řešit rovnici $x = 3x - 2$, můžeme zkusit jiný tvar přepisu $x = \varphi(x)$.

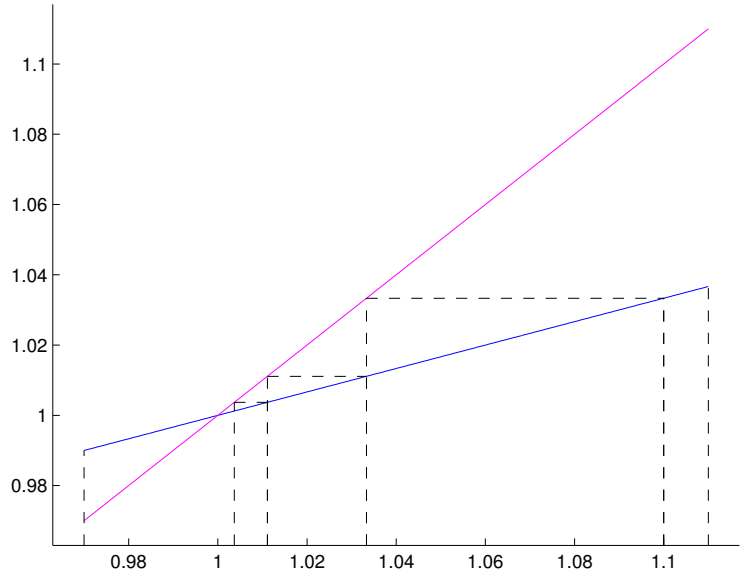
např. $x_{k+1} = \frac{1}{3}x_k + \frac{2}{3},\ x_0 = 1,1$

$$x = 3 \cdot x - 2 \qquad / \cdot \frac{1}{3}$$

$$\frac{1}{3}x = x - \frac{2}{3} \qquad / + \frac{2}{3}$$

$$x = \frac{1}{3}x + \frac{2}{3}$$

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.1 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce11.m takto:

function out=fce11(x);
out=1/3*x+2/3;

| k |       x(k)       | dx(k)=x(k)-x(k-1) |   dx(k)/dx(k-1)    |
-------------------------------------------------------------------
| 0 |     1.100000     |                   |                   |
| 1 |     1.033333     |     -0.066667     |                   |
| 2 |     1.011111     |     -0.022222     |      0.333333     |
| 3 |     1.003704     |     -0.007407     |      0.333333     |
```



Vidíme, že posloupnost klesá k přesnému řešení (blíží se k 1).

Dále je vidět, že se iterace zahušťují a absolutní hodnota rozdílu $|x_k - x_{k-1}|$ s rostoucím k klesá k nule, tj. pro $\forall k\ (> k_0)$ by mělo platit:

$$|x_{k+2} - x_{k+1}| < |x_{k+1} - x_k|.$$

Pokud využijeme iterační předpis, můžeme výraz na levé straně nahradit:

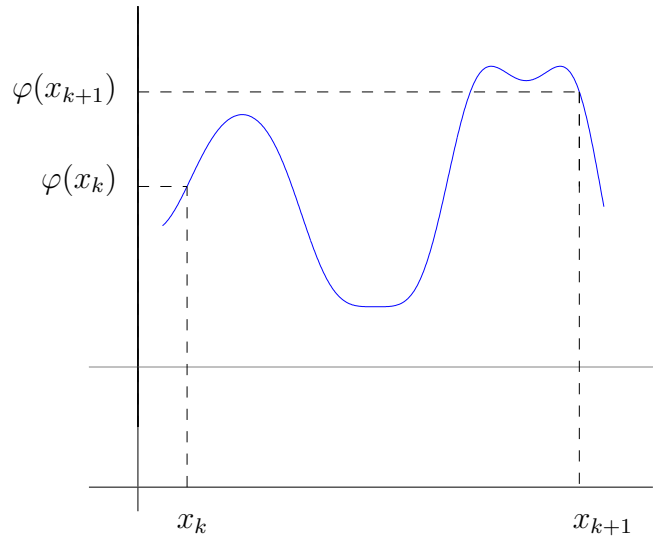
$$|\varphi(x_{k+1}) - \varphi(x_k)| < |x_{k+1} - x_k|.$$

Nerovnici vydělíme $|x_{k+1} - x_k| \neq 0$ (pokud by $x_{k+1} - x_k = 0$, pak by $x_{k+1} = x_k$ a jednalo by se o přesné řešení $x_k = x_{k+1} = \varphi(x_k)$) a dostaneme

$$\frac{|\varphi(x_{k+1}) - \varphi(x_k)|}{|x_{k+1} - x_k|} < 1, \quad \text{resp.} \ \frac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|} < 1 \tag{$\star$}$$

Aby proces konvergoval, je vhodné zajistit splnění podmínky ($\star$).

Geometrický význam:



Pokud bude pro funkci φ platit následující podmínka ($\star\star$) máme splnění podmínky ($\star$) zaručeno:

$$\forall x, y \in I: \quad |\varphi(x) - \varphi(y)| < |x - y|, \tag{$\star\star$}$$

kde I je nějaký interval obsahující řešení, na kterém ho hledáme.

Analogický zápis:

$$\forall x, y \in I: \quad \frac{|\varphi(x) - \varphi(y)|}{|x - y|} < 1.$$

Pokud funkce φ splňuje ($\star\star$), říkáme o ní, že je kontrahující nebo že jde o kontrakci.

Vraťme se k příkladu 1 a 2 a ověřme, zda byla příslušná funkce φ kontrahující (pro jednoduchost na $\mathbb{R}$).

ad 1) $\varphi(x) = 3x - 2$

Pokud vezmeme libovolná 2 různá čísla $x, y \in \mathbb{R}$ dostaneme:

$|\varphi(x) - \varphi(y)| = |(3x - 2) - (3y - 2)| = |3x - 3y - 2 + 2| = |3x - 3y| = \mathbf{3}|x - y|$

$\Rightarrow$ φ není kontrakce.

ad 2) $\varphi(x) = \frac{1}{3}x + \frac{2}{3}$

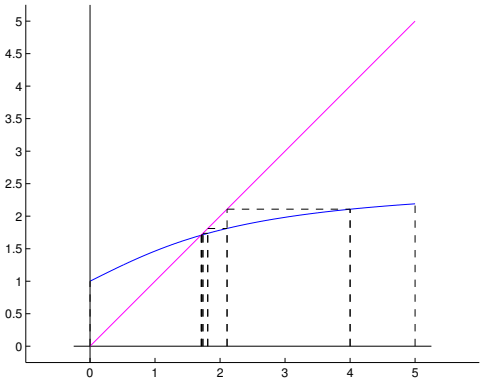
Opět pro libovolná 2 různá čísla $x, y \in \mathbb{R}$ platí:

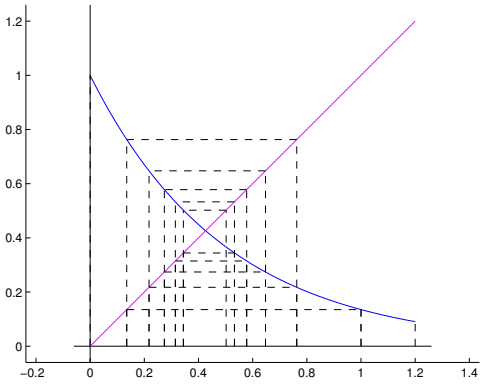
$$|\varphi(x)-\varphi(y)| = \left|(\frac{1}{3}x + \frac{2}{3}) - (\frac{1}{3}y + \frac{2}{3})\right| = \left|\frac{1}{3}x - \frac{1}{3}y - \frac{2}{3} + \frac{2}{3}\right| = \left|\frac{1}{3}x - \frac{1}{3}y\right| = \mathbf{\frac{1}{3}}|x-y|$$

$\Rightarrow$ φ je kontrakce.

Geometricky to znamená, že změna funkce φ z jednoho bodu do druhého musí být menší než vzdálenost těchto bodů, tj. funkce φ nemůže mít velké "růsty" a "poklesy" (změny). Ve srovnání s přímkou $x = y$ musí mít menší "růst" v absolutní hodnotě.

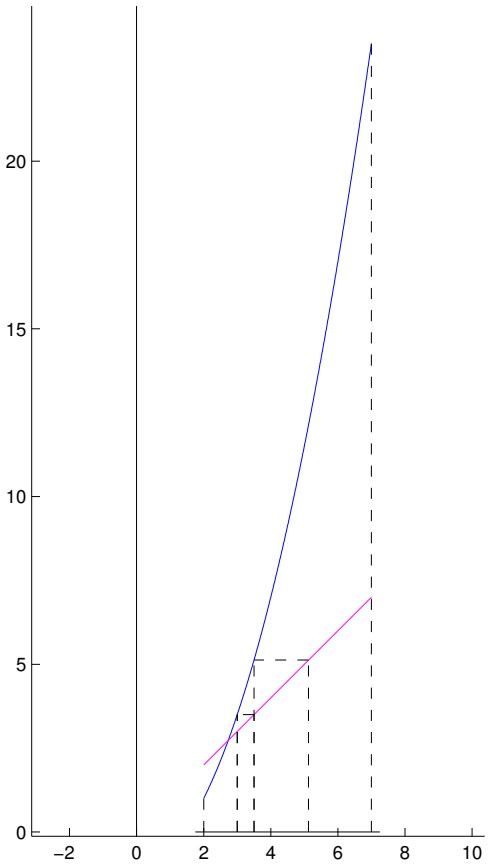
Příklady funkcí $\varphi = \varphi(x)$, které **jsou kontrakcí**:



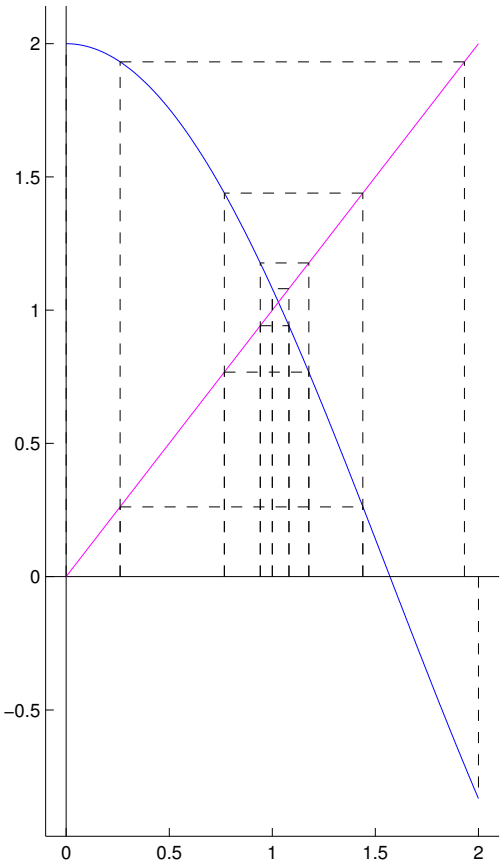


$\varphi(x) = \operatorname{arctg}(x/2) + 1$ na intervalu $\langle 1, 5\rangle$ $\varphi(x) = \mathrm{e}^{(-x/2)}$ na intervalu $\langle 0, 1\rangle$

Příklady funkcí $\varphi = \varphi(x)$, které **nejsou kontrakcí**:



$\varphi(x) = x^2/2 - 1$ na intervalu $\langle 2, 7 \rangle$



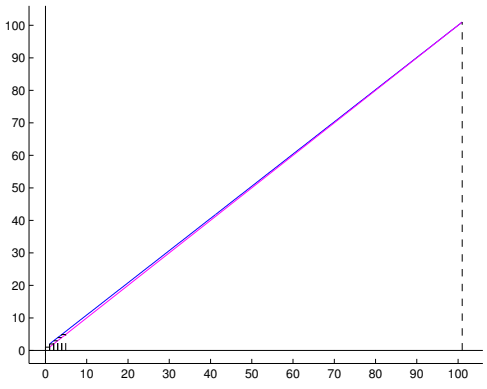
$\varphi(x) = 2\cos x$ na intervalu $\langle 0, 2 \rangle$

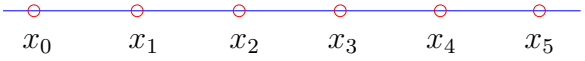
Otázka: Co ovlivňuje rychlost konvergence (zhušťování bodů)?

Příklad: Uvažujeme předpis $x_{k+1} = 0,99\,x_k + 1,\ x_0 = 0$. Posuďte zhušťování iterací.

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=0 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce16.m takto:


function out=fce16(x);
out=0.99*x+1;
```

k	x(k)	dx(k)=x(k)-x(k-1)	dx(k)/dx(k-1)
0	0.000000		
1	1.000000	1.000000	
2	1.990000	0.990000	0.990000
3	2.970100	0.980100	0.990000
4	3.940399	0.970299	0.990000
5	4.900995	0.960596	0.990000





$x_1 - x_0 = 1,\ \ x_2 - x_1 = 0,99,\ \ x_3 - x_2 = 0,9801$

$|x_2 - x_1|$ se oproti $|x_1 - x_0|$ moc nezměnilo

$|x_3 - x_2|$ se oproti $|x_2 - x_1|$ moc nezměnilo

tj. čím víc bude $\dfrac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|}$ blíže k 1 (zdola), tím pomaleji bude proces konvergovat.

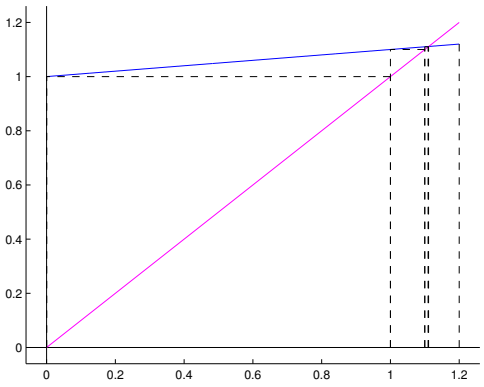
Pokud by byl tento podíl roven 1, pak proces nekonverguje (např.: $x_{k+1} = x_k + 1,\ x_0 = 0$).

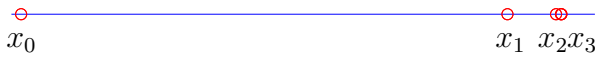
Označme podíl: $q_k = \dfrac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|}$ ... v tomto případě platí, že $q_k = 0,99$.

Příklad: Uvažujeme předpis $x_{k+1} = 0,1\,x_k + 1,\ x_0 = 0$. Posuďte zhušťování iterací.

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=0 a zastavovaci podminku
|x(k)-x(k-1)|<0.01

Funkce funkce_phi(x) je zadana v souboru fce17.m takto:


function out=fce17(x);
out=0.1*x+1;
```

k	x(k)	dx(k)=x(k)-x(k-1)	dx(k)/dx(k-1)
0	0.000000		
1	1.000000	1.000000	
2	1.100000	0.100000	0.100000
3	1.110000	0.010000	0.100000
4	1.111000	0.001000	0.100000





$x_1 - x_0 = 1, \quad x_2 - x_1 = 0,1, \quad x_3 - x_2 = 0,01$

$|x_2 - x_1|$ je 10 x menší než $|x_1 - x_0|$

$|x_3 - x_2|$ je 10 x menší než $|x_2 - x_1|$

tj. čím víc bude $\dfrac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|}$ blíže k 0, tím rychleji bude proces konvergovat.

Pokud by byl tento podíl roven 0, pak máme přesné řešení, protože musí platit, že

$x_{k+2} - x_{k+1} = 0$, tj. $x_{k+2} = x_{k+1}$, tj. $\varphi(x_{k+1}) = x_{k+1}$ a x_{k+1} je přesné řešení.

Podíl q_k je v tomto případě roven 0,1.

Vraťme se ke vztahu $(\star\star)$.

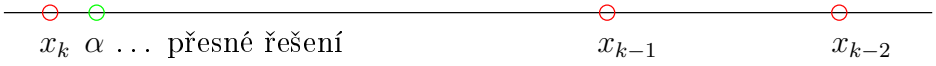
Definice:

Nechť $f : I \to \mathbb{R}$. Potom platí, že existuje q > 0: $|f(x) - f(y)| < q \cdot |x - y| \quad \forall x, y \in I$.

Říkáme, že funkce f je **lipschitzovská funkce**.

Otázka: Jaká je souvislost chyby přibližného řešení a velikosti rozdílu dvou po sobě jdoucích iterací?

$$|x_k - \alpha| \gg |x_k - x_{k-1}|$$



$$|x_k - \alpha| \ll |x_k - x_{k-1}|$$

Odpověď: Mohou nastat obě situace, tj. kdy je chyba $|x_k - \alpha|$ mnohem větší než vzdálenost dvou posledních iterací $|x_k - x_{k-1}|$, ale i naopak mnohem menší.

Odhad chyby metody prosté iterace

- Máme konvergentní proces $x_k = \varphi(x_{k-1}), \quad k = 1, 2, \ldots$
- Přesné řešení α splňuje vztah $\alpha = \varphi(\alpha), \quad \lim_{k\to\infty} x_k = \alpha$
- Po odečtení dostaneme $x_k - \alpha = \varphi(x_{k-1}) - \varphi(\alpha)$

  tj. $|x_k - \alpha| = |\varphi(x_{k-1}) - \varphi(\alpha)|$ $\leftarrow$
- Předpokládáme, že φ je lipchitzovská s konstantou $q \in (0, 1)$, tj. musí platit:

$$|\varphi(x_{k-1}) - \varphi(\alpha)| \leq q \cdot |x_{k-1} - \alpha| \quad \leftarrow$$

- Dále použijeme $\triangle$ nerovnost:

$$|x_{k-1} - \alpha| = |x_{k-1} - x_k + x_k - \alpha| \leq |x_{k-1} - x_k| + |x_k - \alpha| \quad \leftarrow$$

- Dostaneme:

$$|x_k - \alpha| \leq q \cdot |x_{k-1} - x_k| + q \cdot |x_k - \alpha|$$

$$(1 - q) \cdot |x_k - \alpha| \leq q \cdot |x_{k-1} - x_k| \qquad / \cdot \frac{1}{1-q} > 0$$

$$|x_k - \alpha| \leq \frac{q}{1-q} \cdot |x_{k-1} - x_k|$$

Poznámka: Hodnota konstanty q velmi ovlivňuje odhad chyby, ale také, jak jsme viděli dříve, i rychlost "zhušťování" iterací.

Příklad: Opět $x_{k+1} = 0,99\, x_k + 1,\ x_0 = 0$

platí také: $x_{k+2} = 0,99\, x_{k+1} + 1$

po odečtení získáme: $|x_{k+2} - x_{k+1}| = 0,99\, |x_{k+1} - x_k| \quad \Rightarrow \quad q = 0,99$

odhad chyby: $|x_k - \alpha| \leq \frac{0,99}{1 - 0,99} |x_{k-1} - x_k| = 99\, |x_{k-1} - x_k|.$

Tj. chyba může být 99x větší než tolerance ε použitá pro zastavení procesu: $|x_{k-1} - x_k| < \varepsilon$.

Příklad: Opět $x_{k+1} = 0,1 x_k + 1,\ x_0 = 0$

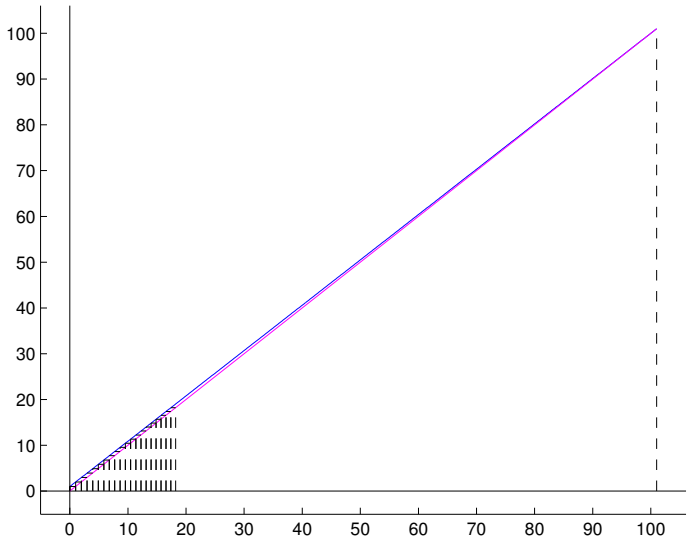
platí také: $x_{k+2} = 0,1\, x_{k+1} + 1$

po odečtení: $|x_{k+2} - x_{k+1}| = 0,1\, |x_{k+1} - x_k| \quad \Rightarrow \quad q = 0,1$

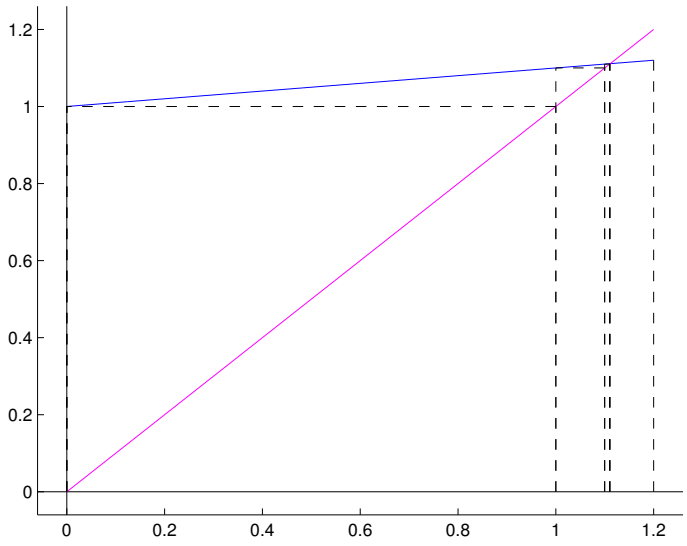
odhad chyby: $|x_k - \alpha| \leq \frac{0,1}{1 - 0,1} |x_{k-1} - x_k| = \frac{1}{9} |x_{k-1} - x_k|$

Tj. chyba je 9x menší než tolerance ε použitá pro zastavení procesu: $|x_{k-1} - x_k| < \varepsilon$.

```
Odhad chyby metody proste iterace pro reseni nelinearni rovnice
                  x = phi(x) = 0.99*x+1

|  k |       x(k)       | dx(k)=x(k)-x(k-1) |   dx(k)/dx(k-1)    |
-----------------------------------------------------------------------
|  0 |     0.000000     |                   |                    |
|  1 |     1.000000     |      1.000000     |                    |
|  2 |     1.990000     |      0.990000     |      0.990000      |
|  3 |     2.970100     |      0.980100     |      0.990000      |
|  4 |     3.940399     |      0.970299     |      0.990000      |
|  5 |     4.900995     |      0.960596     |      0.990000      |
|  6 |     5.851985     |      0.950990     |      0.990000      |
|  7 |     6.793465     |      0.941480     |      0.990000      |
|  8 |     7.725531     |      0.932065     |      0.990000      |
|  9 |     8.648275     |      0.922745     |      0.990000      |
| 10 |     9.561792     |      0.913517     |      0.990000      |
| 11 |    10.466175     |      0.904382     |      0.990000      |
| 12 |    11.361513     |      0.895338     |      0.990000      |
| 13 |    12.247898     |      0.886385     |      0.990000      |
| 14 |    13.125419     |      0.877521     |      0.990000      |
| 15 |    13.994165     |      0.868746     |      0.990000      |
| 16 |    14.854223     |      0.860058     |      0.990000      |
| 17 |    15.705681     |      0.851458     |      0.990000      |
| 18 |    16.548624     |      0.842943     |      0.990000      |
| 19 |    17.383138     |      0.834514     |      0.990000      |
| 20 |    18.209306     |      0.826169     |      0.990000      |

 Odhad chyby :

| x_20 - x_presne | <= q / (1-q) *  | x_20 - x_19 |

| 18.209306 - x_presne | <= 0.99 / (1-0.99) * 0.826169 = 81.790694
```



```
Odhad chyby metody proste iterace pro reseni nelinearni rovnice
                    x = phi(x) = 0.1*x+1
pro pocatecni aproximaci x0=0 a zast. podminku |x(k)-x(k-1)|<0.01

| k |       x(k)       | dx(k)=x(k)-x(k-1) |   dx(k)/dx(k-1)   |
-----------------------------------------------------------------
| 0 |     0.000000     |                   |                   |
| 1 |     1.000000     |      1.000000     |                   |
| 2 |     1.100000     |      0.100000     |      0.100000     |
| 3 |     1.110000     |      0.010000     |      0.100000     |
| 4 |     1.111000     |      0.001000     |      0.100000     |

 Odhad chyby :

| x_4 - x_presne | <= q / (1-q) *  | x_4 - x_3 |

| 1.111000 - x_presne | <= 0.1 / (1-0.1) * 0.001000 = 0.000111
```



Chaos ($\chi\alpha o\varsigma$)

Deterministický chaos - na první pohled slova s protichůdným obsahem.

Edward Norton Lorenz (23.5.1917 - 16.4.2008) (*otec chaosu*)

Americký matematik (Harvard) a meteorolog (MIT). Při studiu modelů počasí Lorenz objevil, že počasí se ne vždy chová podle předpovědi. Malá odchylka v počátečních hodnotách proměnných v jeho primitivním počítačovém modelu počasí měla za následek veliké rozdíly v chování počasí. Tato citlivá závislost na počátečních podmínkách se stala známou jako motýlí efekt.

Vylučuje prediktabilitu (dlouhodobější)

Deterministické chování Ve zdánlivě chaotických systémech (turbulence, stoupající kouř) existuje skrytý řád. Je možné stanovit, jaké příčiny vedly k danému pozorovanému následku (většinou je to však možné pouze dodatečně - *a* posteriori).

$$\text{PŘÍČINA} \quad \Rightarrow \quad \text{NÁSLEDEK}$$

Soběpodobnost Určitá vlastnost se zachovává nezávisle na měřítku. Objevuje se určitý řád ve zdánlivém chaosu. Graficky znázorněná vlastnost soběpodobnosti ... fraktál.

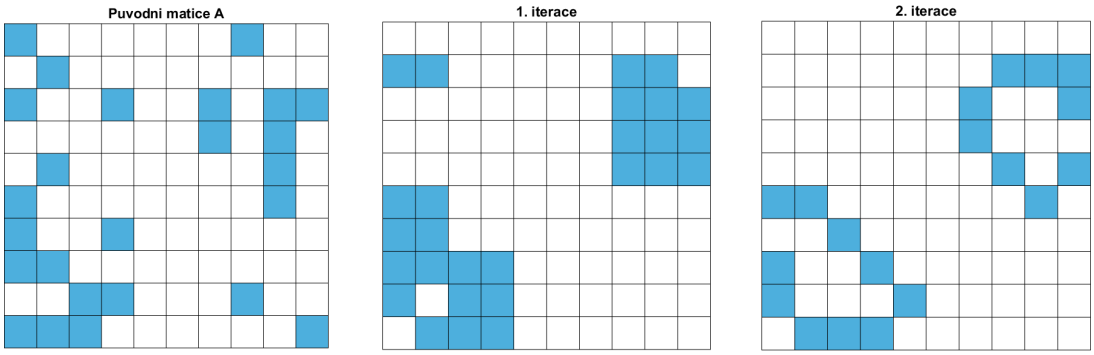
Samoorganizace Například pohyb ryb v hejnu.

Věrohodný model - Každá ryba se řídí třemi jednoduchými lokálními pravidly:

soudržnost hejna, zařazení a oddělení.

Je překvapující, jak komplikované kolektivní chování z těchto 3 jednoduchých pravidel vychází. Neexistuje žádný globální plán pohybu nebo perspektiva a neexistuje žádný vůdce hejna (podobné chování mravenců).

Hra život



Zkusme sestavit jednoduchý model růstu populace.

Pro jednoduchost uvažujeme kolonii baktérií, která se vždy po určitém čase dělením rozmnoží.

Přírůstek za čas $\triangle t$: $\dfrac{y(n+1)-y(n)}{\triangle t} = k \cdot y(n)$ (diferenční rovnice)

$y(n)$ ... počet jedinců (baktérií) v čase n

k ... koeficient růstu (tj. frekvence dělení baktérií za určitý čas)

$\triangle t$ ... změna času (z času n do času $n+1$) ... časový krok

$$y(n+1) - y(n) = k \cdot y(n) \cdot \triangle t$$
$$y(n+1) = (k\triangle t + 1) \cdot y(n)$$
$$y(n+1) = b \cdot y(n)$$

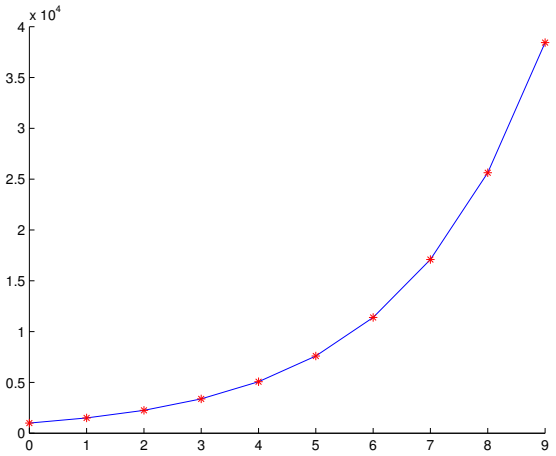
Získali jsme rekurentně zadanou posloupnost.

Poznámka: Stejný model lze použít i pro modelování počtu lidí nebo zvířat atd.

Příklad: Jaké je chování systému $x_{n+1} = b \cdot x_n$ pro různé hodnoty parametrů ?

Pro zvolené $b = 1,5$ a $x_0 = 1000$ dostaneme:

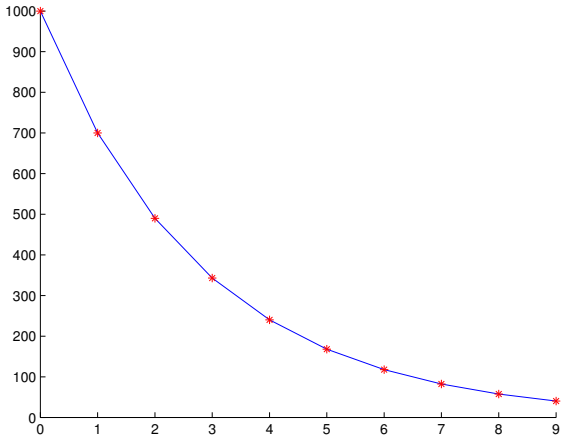
```
Nejjednodussi model
rustu populace:

x_0     = 1000  (zadano)
x_(n+1) = b * x_(n)
parametr b = 1.500000

   n |  hodnota(n)
----------------------
   1 |  1000.000000
   2 |  1500.000000
   3 |  2250.000000
   4 |  3375.000000
   5 |  5062.500000
   6 |  7593.750000
   7 |  11390.625000
   8 |  17085.937500
   9 |  25628.906250
  10 |  38443.359375
```



Exponenciální chování

Pro zvolené $b = 0,7$ a $x_0 = 1000$ dostaneme:

```
Nejjednodussi model
rustu populace:

x_0     = 1000  (zadano)
x_(n+1) = b * x_(n)
parametr b = 0.700000

   n |  hodnota(n)
----------------------
   1 |  1000.000000
   2 |  700.000000
   3 |  490.000000
   4 |  343.000000
   5 |  240.100000
   6 |  168.070000
   7 |  117.649000
   8 |  82.354300
   9 |  57.648010
  10 |  40.353607
```



Opět exponenciální chování

Pokud vezmeme v úvahu omezený prostor nebo omezený zdroj potravy, nemůže tento model fungovat (chování je exponenciální).

⇒ Musíme vybrat model, ve kterém se počet jedinců bude regulovat tak, aby nemohl být překročen určitý maximální počet.

Uvažujeme tento model:

$$X_{n+1} \;=\; b \cdot X_n \cdot (10^4 - X_n)/10^4 \;=\; b \cdot X_n \cdot \left(1 - \frac{X_n}{10^4}\right)$$

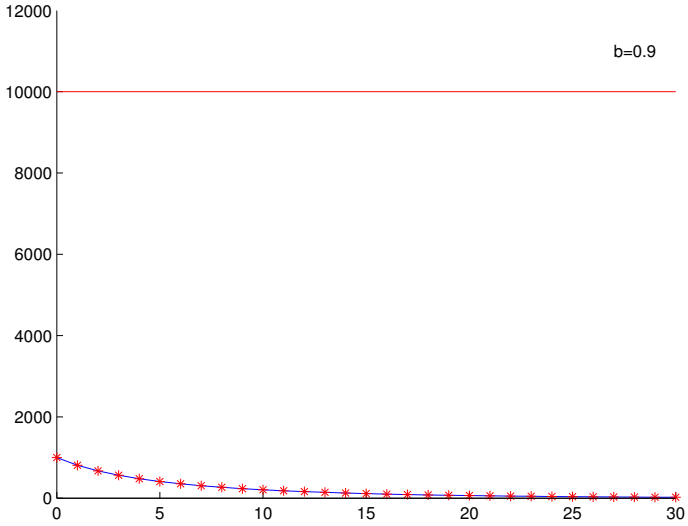
10^4 ... maximální možný počet jedinců v populaci.

Pokud je závorka $(10^4 - X_n)$ malé číslo, tj. populace je již početná, bude počet v dalším čase $(n+1)$ regulován a dojde k redukci v populaci.

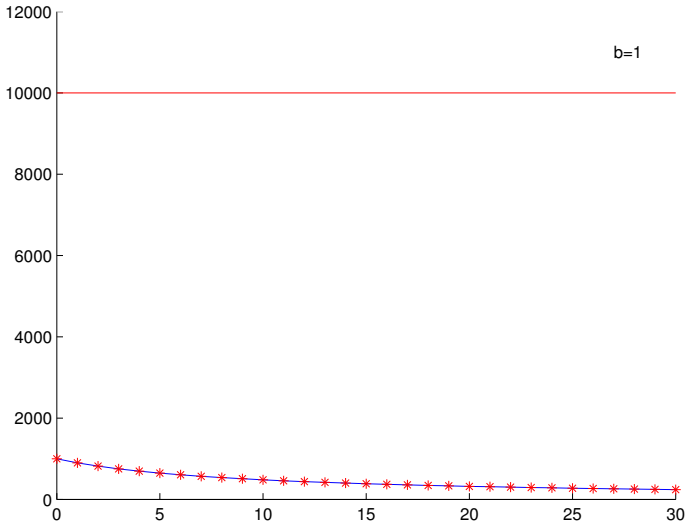
Naopak, pokud je malý počet jedinců v populaci, je závorka $(10^4 - X_n)$ velké číslo a nárůst počtu jedinců v dalším čase $(n+1)$ se podpoří.

Nyní budeme sledovat chování modelu po různé hodnoty parametru b. Ve všech uvažujeme počáteční počet jedinců populace $X_0 = 1000$, tj. 10% z možné kapacity prostředí.

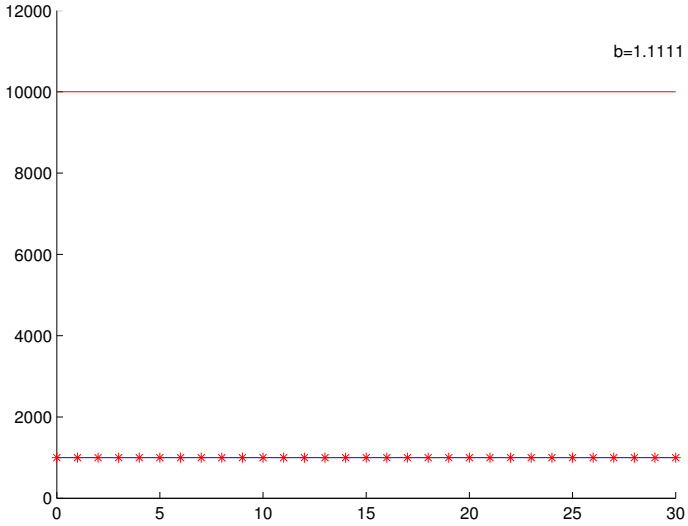
1. případ $b = 0,9$



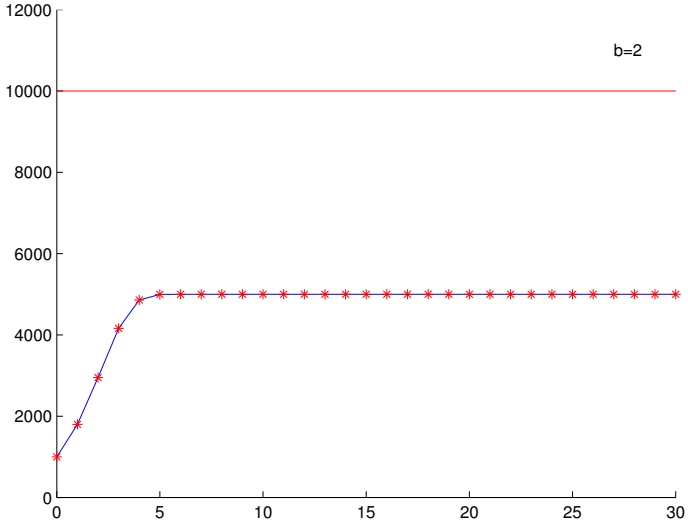
2. případ $b = 1$



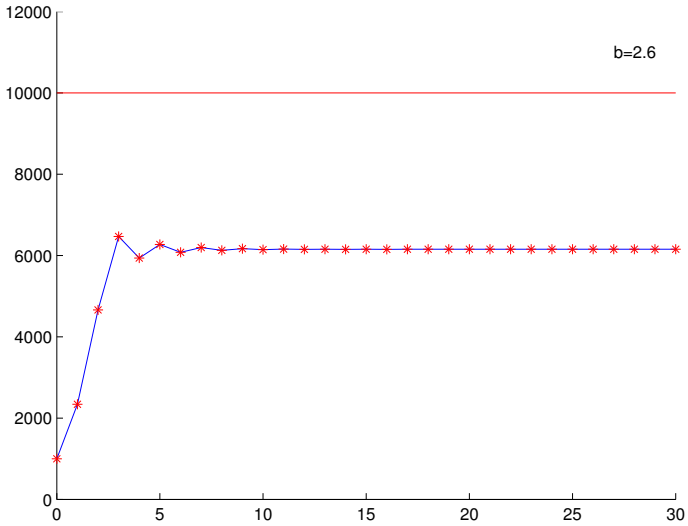
3. případ $b = \frac{10}{9}$



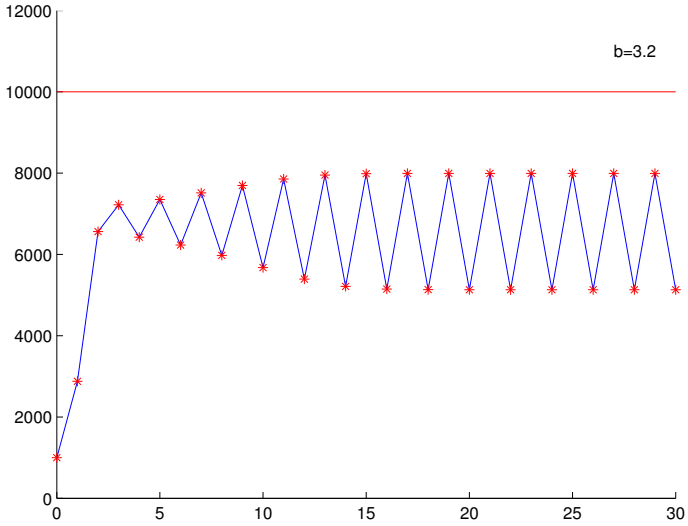
4. případ $b = 2$



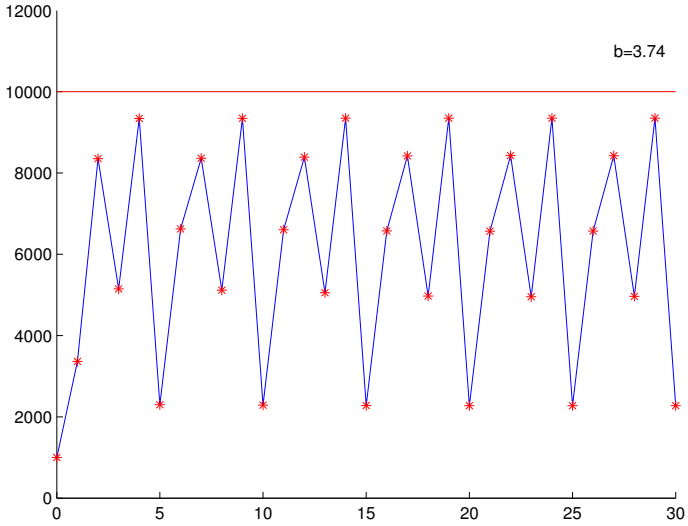
5. případ $b = 2,6$



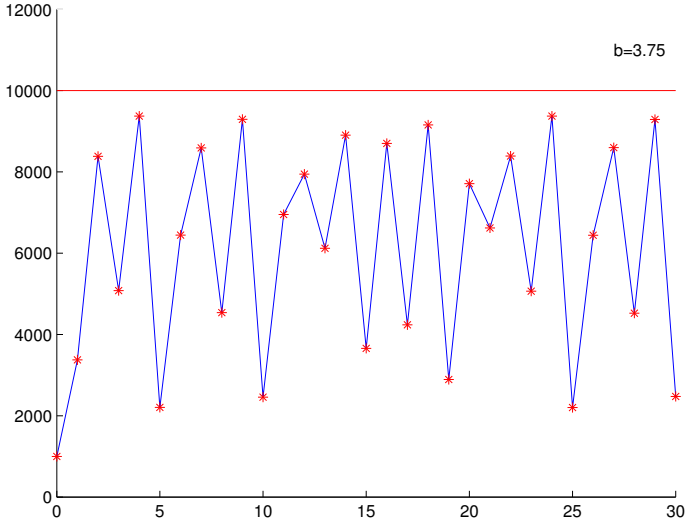
6. případ $b = 3,2$



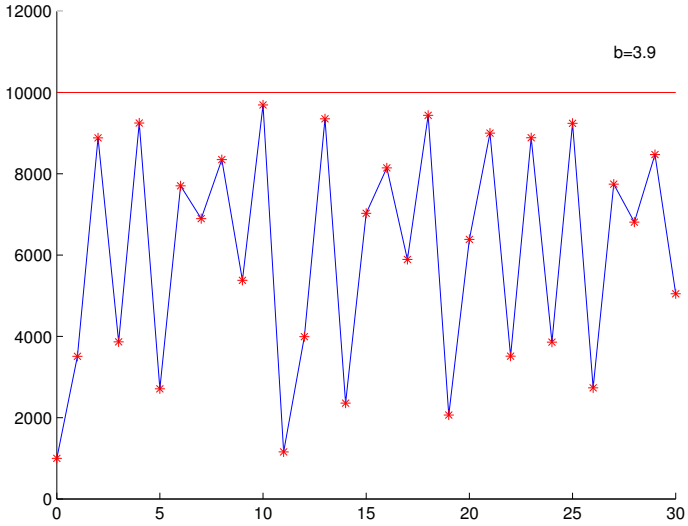
7. případ $b = 3,74$



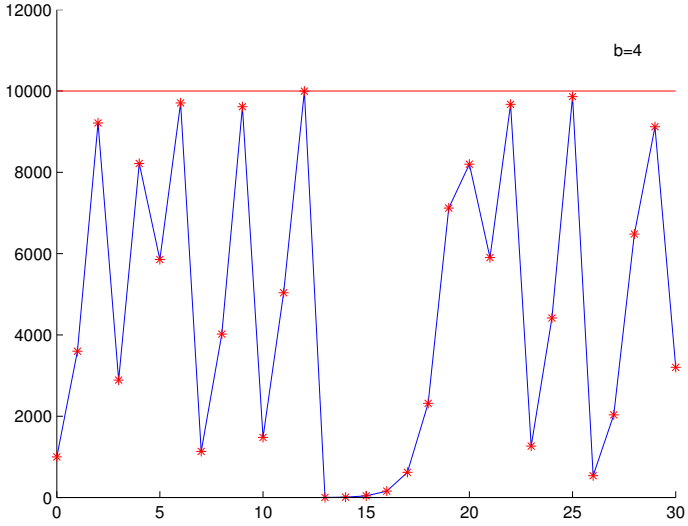
8. případ $b = 3,75$



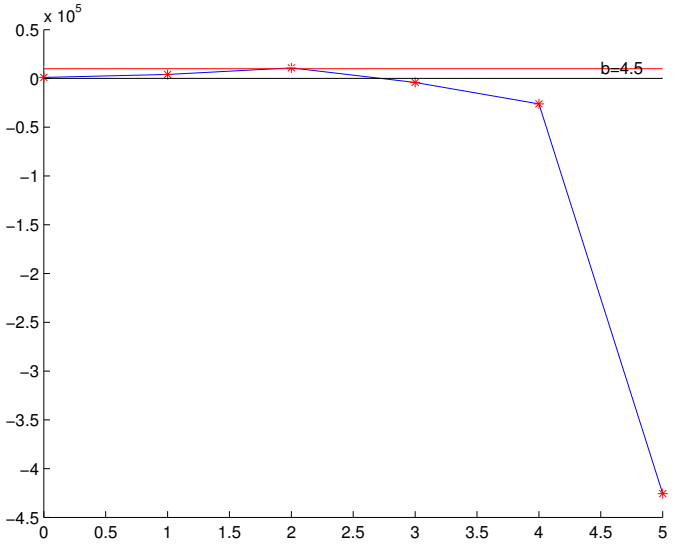
9. případ $b = 3,9$



10. případ $b = 4$



11. případ $b = 4,5$



Komentáře:

- Pro parametr $0 \leq b < \frac{10}{9}$ populace vymizí, tj. počet se ustálí na hodnotě 0 jedinců.
- Pro parametr $b = \frac{10}{9}$ se populace nemění, tj. zůstává na počáteční hodnotě 1000 jedinců.
- Pro parametr $b = 2$, resp. $b = 2,6$ se počet se ustálil na hodnotě 5000, resp. 6153,84.
- Pro parametr $b = 3,2$ proces nekonverguje, ale osciluje mezi dvěma hodnotami 5130,4 a 7994,5 ("vzestup" a "pád", perioda je 2).
- Výsledky pro parametr $b = 3,74$ a $b = 3,75$ ilustrují fakt, že malá změna v parametru b způsobí dramatickou změnu v chování modelu.

  Pro $b = 3,74$ se zdá, že se opakuje pět hodnot, tj. perioda je 5.

  Pro $b = 3,75$ se periodicita poruší.
- Pro parametr $b = 3,\check{9}$. Zde se zdá, že systém ztratil regulární charakter a získává charakter "chaosu". Členy posloupnosti nejsou náhodné - model je deterministický.
- Pro parametr $b = 4$ dochází v modelu k velikým výkyvům, kdy populace skoro vymizí a stavem, kdy zcela naplní kapacitu prostředí.
- Pro parametr $b = 4.5$ dojde k překročení meze pro prostředí a systém se zhroutí.

Věnujeme se případům, kdy posloupnost konverguje, tj. počet se ustálí na jediné hodnotě, kterou označme X. Potom musí platit rovnost, pokud do rekurentní formule dosadíme za X_n i X_{n+1} hodnotu X, tj.

$$X = b\,X\,\frac{10^4 - X}{10^4} \qquad / \cdot \frac{10^4}{X}$$

$$10^4 = b\,(10^4 - X)$$

$$10^4 = b\,10^4 - b\,X$$

$$X = \frac{10^4\,(b-1)}{b}$$

$$X = 10^4\,(1 - \frac{1}{b})$$

Hledání ustálené hodnoty pro přepis $X_{n+1} = \phi(X_{n+1})$ je myšlenkou tzv. metody prosté iterace, resp. věty o pevném bodě.

Zkusme se zamyslet, kdy bude předpis $X_{n+1} = \phi(X_{n+1})$ konvergovat k jedné hodnotě.

V případě $b = 2$ dostaneme po dosazení, že $X = 10^4\,(1 - \frac{1}{2}) = 5000$.

V případě $b = 2,6$ dostaneme po dosazení, že $X = 10^4(1 - \frac{1}{2,6}) = 6153,84$.

V případě $b = 3,2$ dostaneme po dosazení, že $X = 10^4(1 - \frac{1}{3,2}) = 6875$

Tady ovšem proces osciloval mezi $5130,4$ a $7994,5$. Tj. neplatilo, že $\lim_{n\to\infty} X_n = X$.

Pokud se proces ustálí ve stavu, kdy se střídá hodnota X a Y musí platit:

$$X = \lim_{n\to\infty} X_{2n} \quad \text{a} \quad Y = \lim_{n\to\infty} X_{2n+1}$$

$$Y = b\,X\,\frac{10^4 - X}{10^4}$$

$$X = b\,Y\,\frac{10^4 - Y}{10^4}$$

$$X = b\,(b\,X\,\frac{10^4 - X}{10^4})\,(10^4 - (b\,X\,\frac{10^4 - X}{10^4}))\,10^{-4}$$

$$X = \frac{b^2 X}{10^8}(10^4 - X)(10^4 - b\,X\,(1 - \frac{X}{10^4}))$$

$$X = (\frac{b^2 X}{10^4} - \frac{b^2 X^2}{10^8})(10^4 - b\,X + \frac{bX^2}{10^4}) \qquad / : X$$

$$1 = (1 - \frac{X}{10^4})(\frac{b^2}{10^4})(10^4 - b\,X + \frac{b\,X^2}{10^4})$$

Jedná se o kubickou rovnici, jejíž jedním z kořenů musí být $X = 10^4(1 - \frac{1}{b})$, tj. musí být zahrnut případ, že proces konverguje.

Rovnici lze tedy vydělit členem $(X - 10^4\,(1 - \frac{1}{b}))$, získáme tak kvadratickou rovnici s kořeny X a Y.

$$-X^3(\frac{b^3}{10^{12}}) + X^2(\frac{b^3}{10^8} + \frac{b^3}{10^8}) + X(-\frac{b^3}{10^4} - \frac{b^2}{10^8}) + b^2 - 1 = 0$$

Rovnice má pro $b = 3,2$ kořeny:

$X = 7994,55$, $Y = 5130,445$ a $Z = 6875$.

```
 Zkoumame, mezi kterymi 2 hodnotami osciluje reseni
 v modelu rustu populace s omezenim:

 MEZ     = 10000
 x_0     = 1000
 x_(n+1) = b * x_(n) *(MEZ - x_(n)) / MEZ
 b       = 3.200000

Predpokladame, ze
- liche cleny posloupnosti se ustali na hodnote X
- sude cleny posloupnosti se ustali na hodnote Y

Limitne tedy musi platit, ze
  X = b*Y*(10^4-Y)/10^4
  Y = b*X*(10^4-X)/10^4

Po dosazeni za Y do prvni rovnice dostaneme rovnici
  X = b*(b*X*(10^4-X)/10^4)*(10^4-(b*X*(10^4-X)/10^4))/10^4
resp. po faktorizaci

 0 =

  1/1000000000000 X (b X + 10000 - 10000 b)

           2  2          2
         (b  X  - 10000 b  X - 10000 b X + 100000000 b + 100000000)

Reseni pro parametr b=3.200000 jsou

  X(1) = 0.000000
  X(2) = 6875.000000
  X(3) = 7994.554905
  X(4) = 5130.445095
```

Rovnice má pro $b = 3,3$ kořeny:

$X = 4794.270198$, $Y = 8236.032832$ a $Z = 6969.69697$.

```
 Zkoumame, mezi kterymi 2 hodnotami osciluje reseni
 v modelu rustu populace s omezenim:

 MEZ     = 10000
 x_0     = 1000
 x_(n+1) = b * x_(n) *(MEZ - x_(n)) / MEZ
 b       = 3.300000

Predpokladame, ze
- liche cleny posloupnosti se ustali na hodnote X
- sude cleny posloupnosti se ustali na hodnote Y

Limitne tedy musi platit, ze
  X = b*Y*(10^4-Y)/10^4
  Y = b*X*(10^4-X)/10^4

Po dosazeni za Y do prvni rovnice dostaneme rovnici
  X = b*(b*X*(10^4-X)/10^4)*(10^4-(b*X*(10^4-X)/10^4))/10^4
resp. po faktorizaci

 0 =

  1/1000000000000 X (b X + 10000 - 10000 b)

           2  2          2
         (b  X  - 10000 b  X - 10000 b X + 100000000 b + 100000000)

Reseni pro parametr b=3.300000 jsou

  X(1) = 0.000000
  X(2) = 6969.696970
  X(3) = 8236.032832
  X(4) = 4794.270198
```

Zkusme vypočítat kořeny rovnice pro $b = 2$.

V tomto případě se ovšem řešení ustálilo na jedné hodnotě.

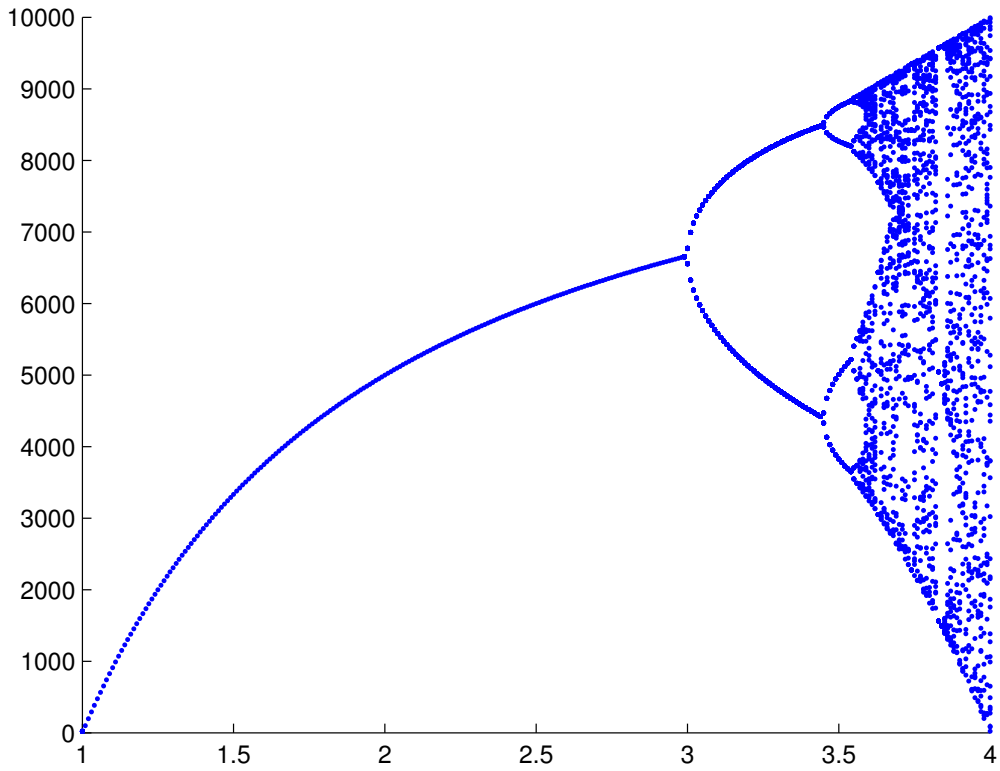
```
 Zkoumame, mezi kterymi 2 hodnotami osciluje reseni
 v modelu rustu populace s omezenim:

 MEZ     = 10000
 x_0     = 1000
 x_(n+1) = b * x_(n) *(MEZ - x_(n)) / MEZ
 b       = 2.000000

Predpokladame, ze
- liche cleny posloupnosti se ustali na hodnote X
- sude cleny posloupnosti se ustali na hodnote Y

Limitne tedy musi platit, ze
  X = b*Y*(10^4-Y)/10^4
  Y = b*X*(10^4-X)/10^4

Po dosazeni za Y do prvni rovnice dostaneme rovnici
  X = b*(b*X*(10^4-X)/10^4)*(10^4-(b*X*(10^4-X)/10^4))/10^4
resp. po faktorizaci

 0 =

  1/1000000000000 X (b X + 10000 - 10000 b)

           2  2          2
         (b  X  - 10000 b  X - 10000 b X + 100000000 b + 100000000)

Reseni pro parametr b=2.000000 jsou

  X(1) = 0.000000
  X(2) = 5000.000000
  X(3) = 7500.000000 + 4330.127019 * i
  X(4) = 7500.000000 - 4330.127019 * i

 Koreny X(3) a X(4) nejsou realne. =>
 V tomto pripade nebude reseni oscilovat mezi dvema hodnotami.
```

Zkusme vypočítat kořeny rovnice pro $b = 3,6$.

V tomto případě se ovšem řešení neustálilo na jedné hodnotě ani neoscilovalo mezi dvěma hodnotami.

```
 Zkoumame, mezi kterymi 2 hodnotami osciluje reseni
 v modelu rustu populace s omezenim:

 MEZ     = 10000
 x_0     = 1000
 x_(n+1) = b * x_(n) *(MEZ - x_(n)) / MEZ
 b       = 3.600000

Predpokladame, ze
- liche cleny posloupnosti se ustali na hodnote X
- sude cleny posloupnosti se ustali na hodnote Y

Limitne tedy musi platit, ze
  X = b*Y*(10^4-Y)/10^4
  Y = b*X*(10^4-X)/10^4

Po dosazeni za Y do prvni rovnice dostaneme rovnici
  X = b*(b*X*(10^4-X)/10^4)*(10^4-(b*X*(10^4-X)/10^4))/10^4
resp. po faktorizaci

 0 =

  1/1000000000000 X (b X + 10000 - 10000 b)

           2  2          2
         (b  X  - 10000 b  X - 10000 b X + 100000000 b + 100000000)

Reseni pro parametr b=3.600000 jsou

  X(1) = 0.000000
  X(2) = 7222.222222
  X(3) = 8696.284406
  X(4) = 4081.493371


Pro overeni vypiseme nasledujici iterace:

  X_27 = 8881.699173
  X_28 = 3575.668151
  X_29 = 8269.660362
  X_30 = 5151.355603
```

Pokud pro různé hodnoty parametru b vyneseme do grafu ustálená řešení, resp. několik vyšších členů posloupnosti, získáme tento obrázek:



Uvažujeme podobný příklad, tentokrát v komplexním oboru.

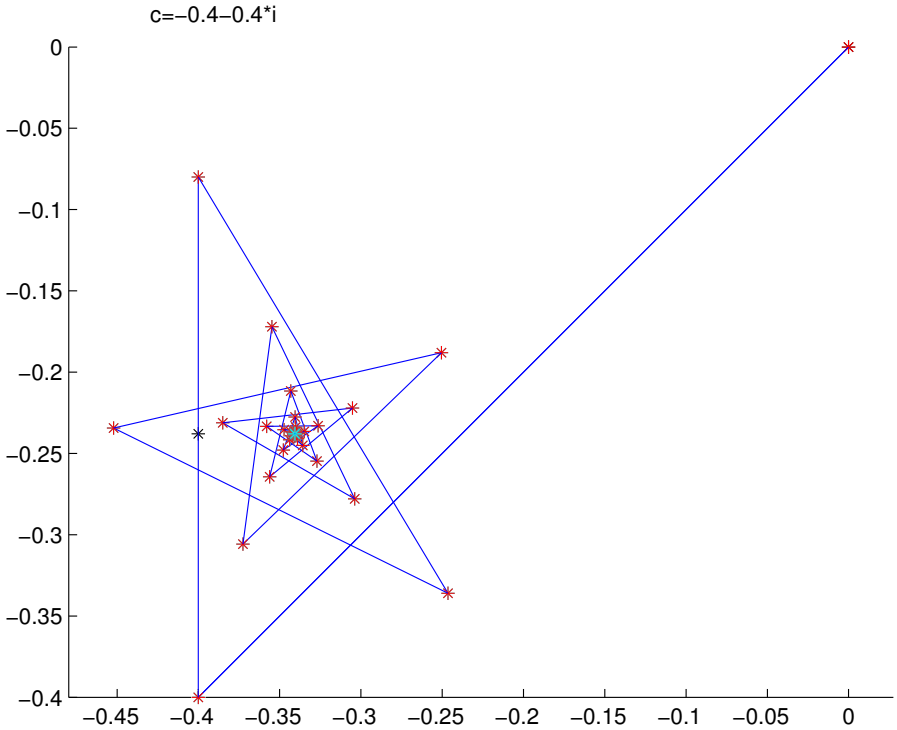
Máme iterační formuli:

$$X_{n+1} = X_n^2 + c$$

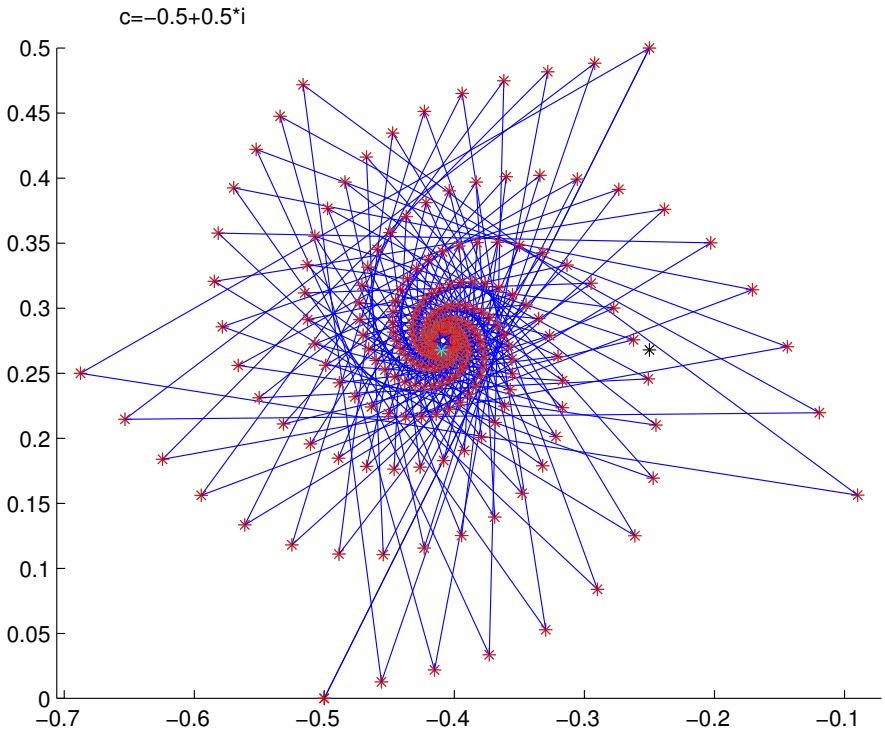
a volíme počáteční iteraci X_0 a hodnotu c.

Ukažme si vykreslené členy posloupnosti pro několik případů volby X_0 a c.

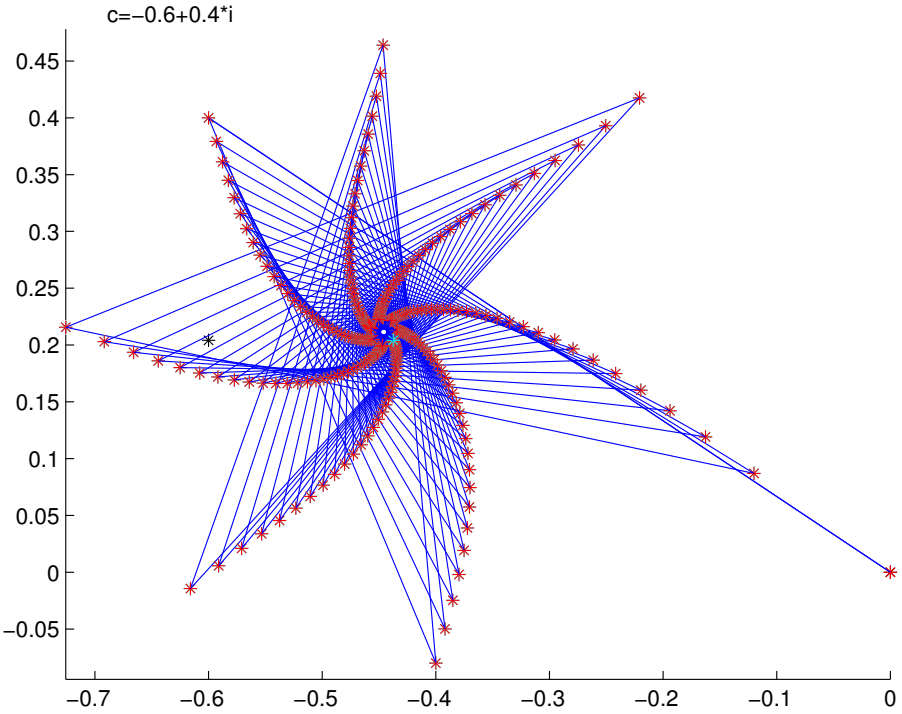
1. příklad $x_0 = 0;\ \ c = -0,4 - 0,4i$



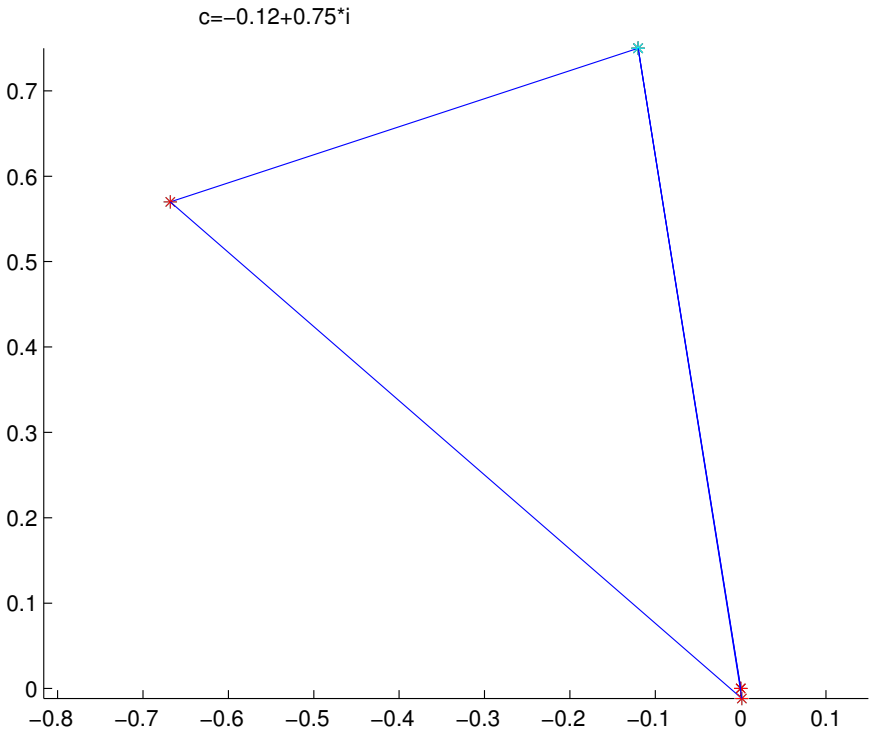
2. příklad $x_0 = -0,5;\ \ c = -0,5 + 0,5i$



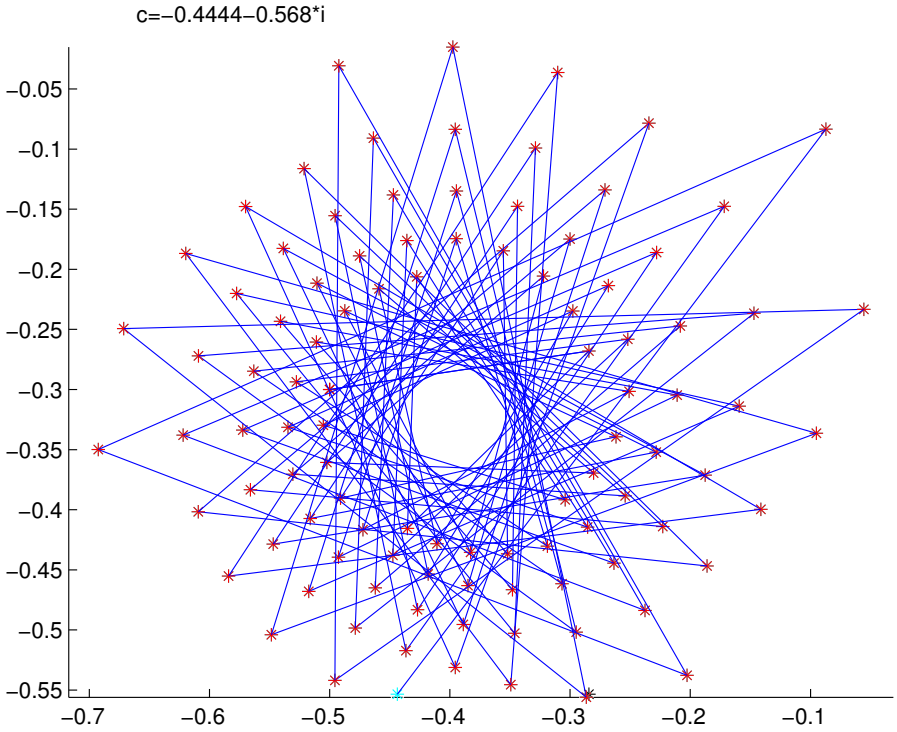
3. příklad $x_0 = 0;\ \ c = -0,6 + 0,4i$



4. příklad $x_0 = 0;\ \ c = -0,12 + 0,75i$



5. příklad $x_0 = -0,5 - 0,3i; \;\; c = -0,4444 - 0,568i$



Fraktály

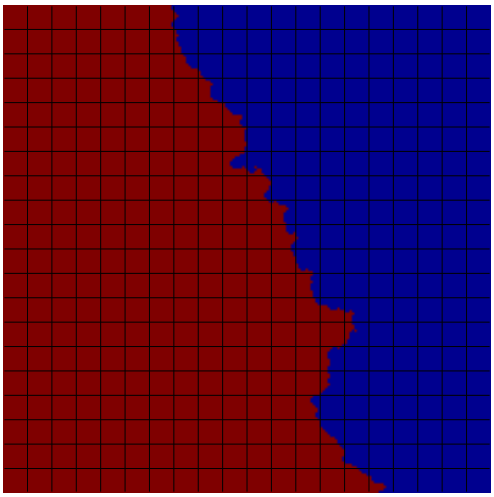
Měření délky pobřeží

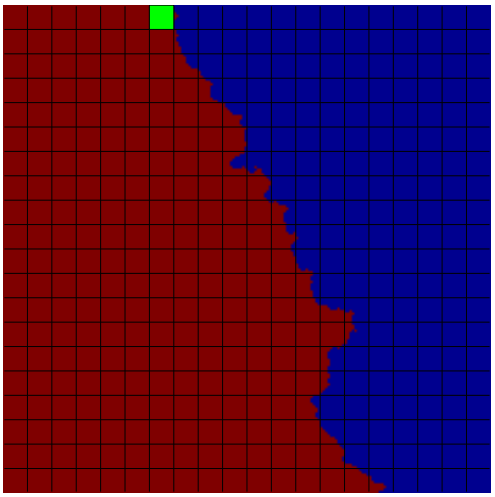
Jak mám změřit délku pobřeží?

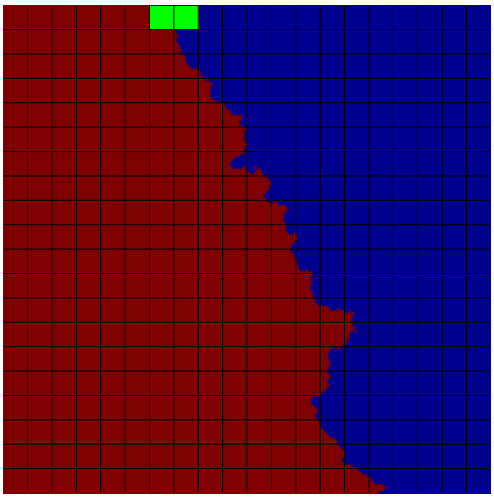
Zkusíme 2 možnosti:

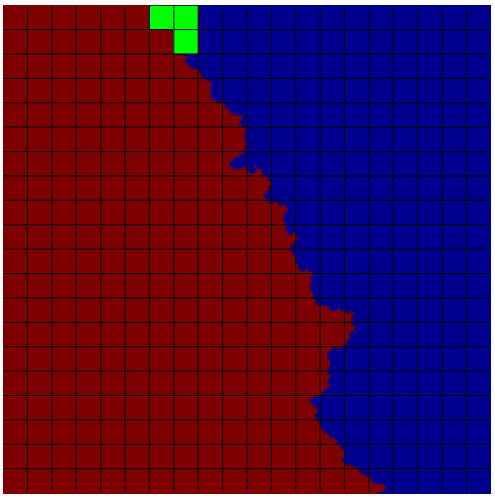
1) - Vytvoříme mřížku (čtvercovou) s určitou velikostí čtverce.
   - Obarvíme ty čtverce, ve kterých leží jak pevná zem, tak moře.
   - Výsledná délka je dána počtem obarvených čtverců x jejich počet (strana, úhlopříčka).
2) - Určíme si délku měřidla (kružítkem).
   - Začneme v jednom krajním bodě.
   - Určujeme další bod pobřeží, který leží právě tak daleko od předchozího.
   - Výsledná délka je součtem vzdáleností získaných bodů, tj. jejich počet x délka měřidla.

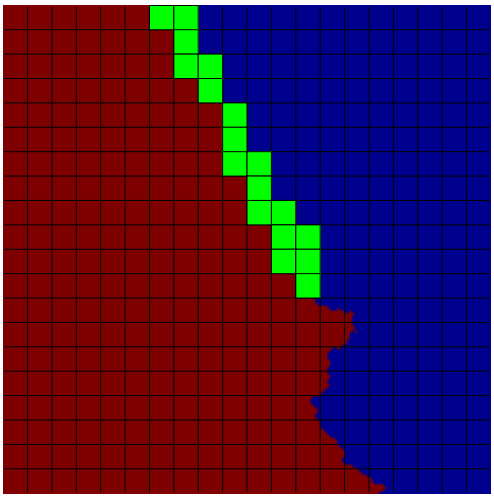
add 1)

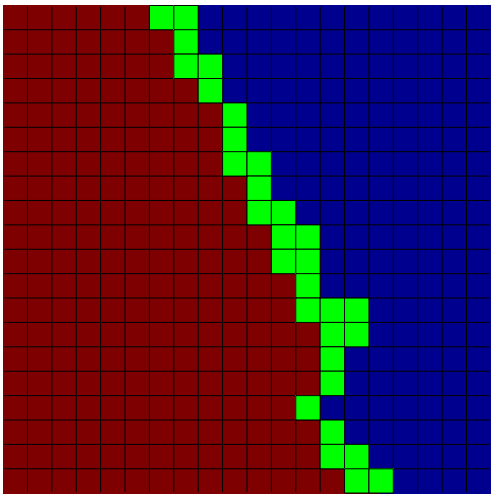




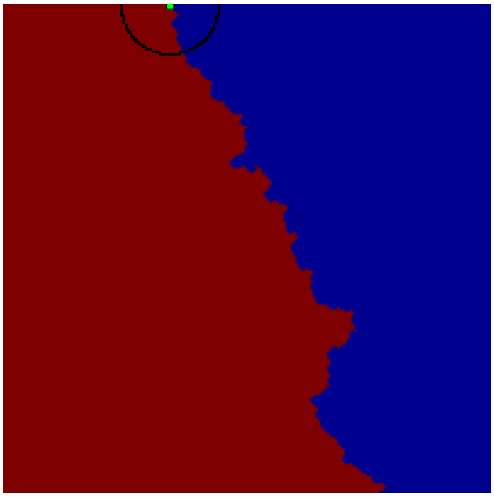


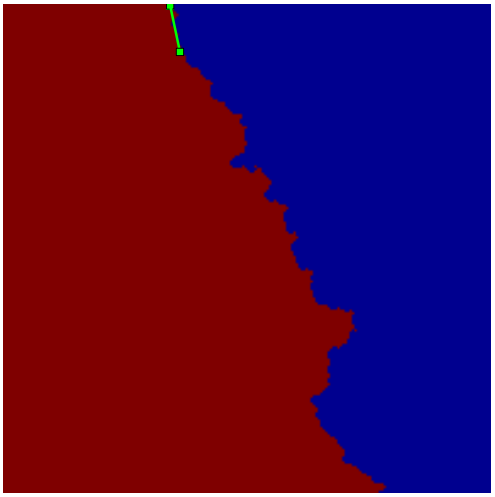


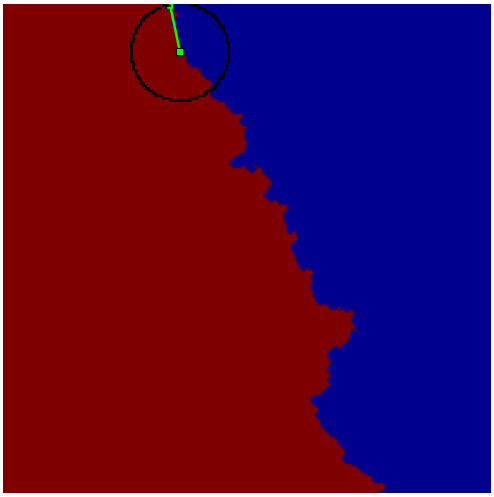


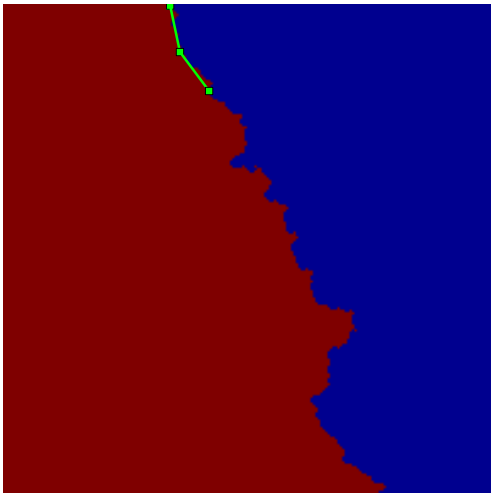


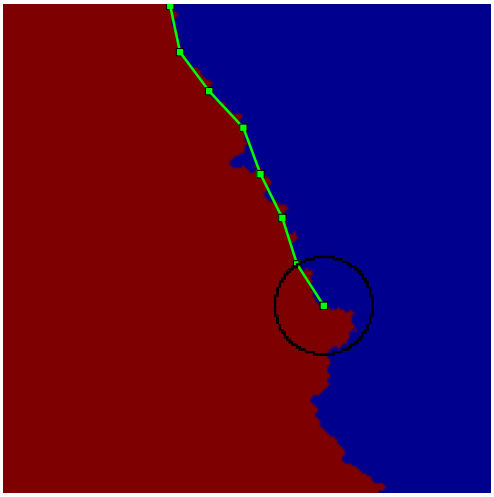
add 2)

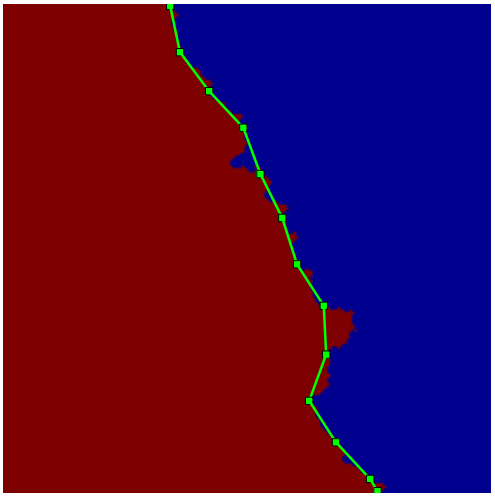










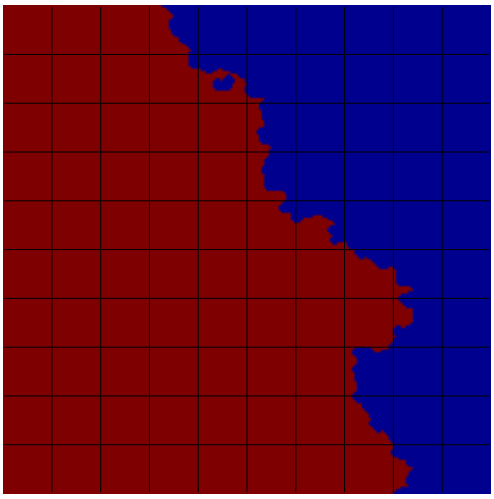


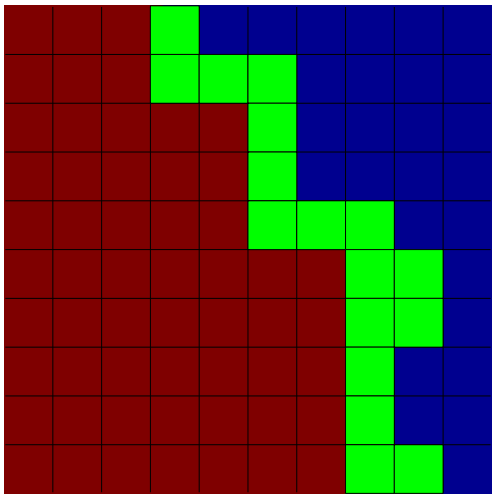
Pozorování:

Čím menší budu mít měřidlo tím větší naměříme délku pobřeží ! (Richardsonův efekt)

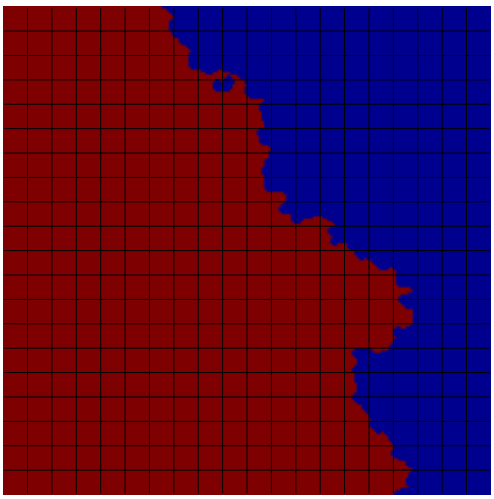
Členitost pobřeží charakterizuje neceločíselná dimenze.

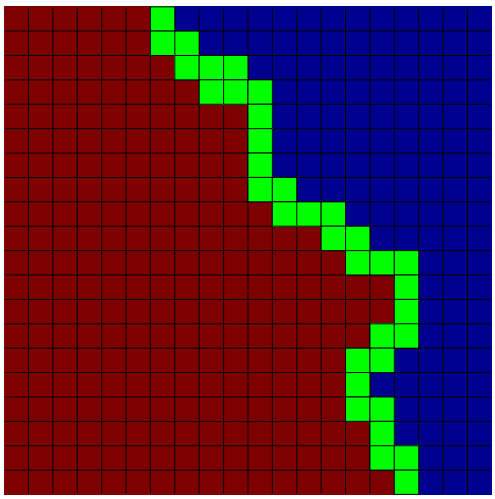
add 1)



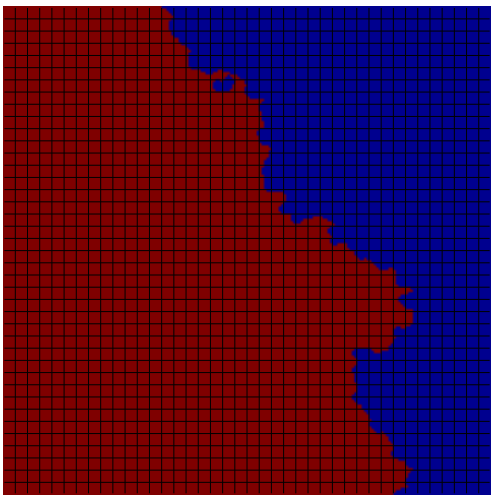


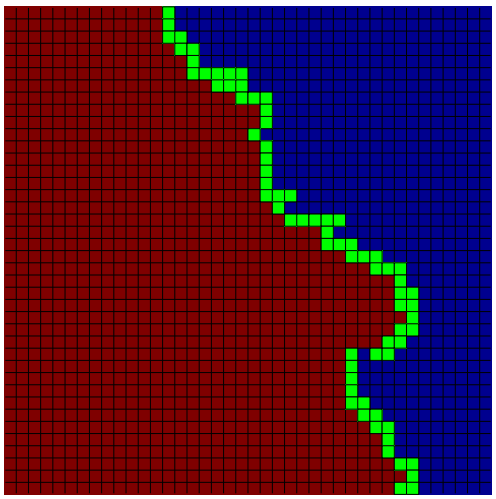
```
Ukazka mereni delky pobrezi pomoci ctvercove site.

 Zadej pocet ctvercu na stranu
  (doporuceno 5,10,20,40) = 10

 Pokud by mapa predstavovala uzemi 100 x 100 km,
 zvoleny ctverec by mel stranu delky 10.000000 km.

 Pocet oznacenych ctvercu je 17 .

 Celkova delka pobrezi je tedy soucinem poctu ctvercu a
 delky strany ctverce, tj. 17 x 10.000000 =  170.000000.
```





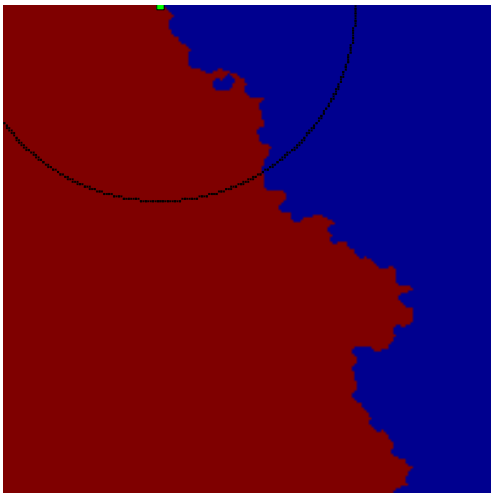
```
Ukazka mereni delky pobrezi pomoci ctvercove site.

 Zadej pocet ctvercu na stranu
  (doporuceno 5,10,20,40) = 20

 Pokud by mapa predstavovala uzemi 100 x 100 km,
 zvoleny ctverec by mel stranu delky 5.000000 km.

 Pocet oznacenych ctvercu je 35 .

 Celkova delka pobrezi je tedy soucinem poctu ctvercu a
 delky strany ctverce, tj. 35 x 5.000000 =  175.000000.
```

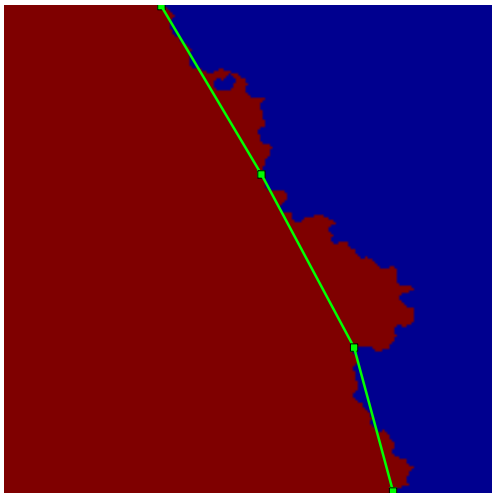




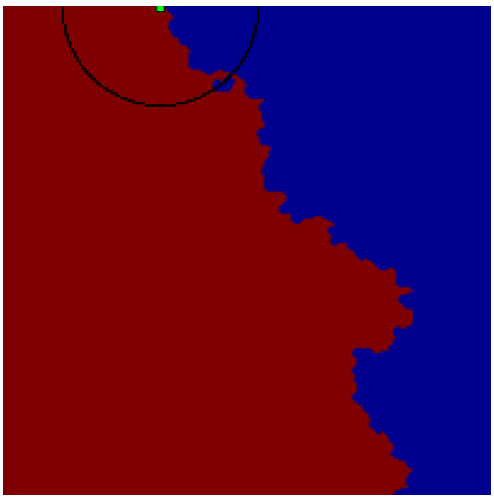
```
Ukazka mereni delky pobrezi pomoci ctvercove site.

 Zadej pocet ctvercu na stranu
  (doporuceno 5,10,20,40) = 40

 Pokud by mapa predstavovala uzemi 100 x 100 km,
 zvoleny ctverec by mel stranu delky 2.500000 km.

 Pocet oznacenych ctvercu je 73 .

 Celkova delka pobrezi je tedy soucinem poctu ctvercu a
 delky strany ctverce, tj. 73 x 2.500000 =  182.500000 .
```

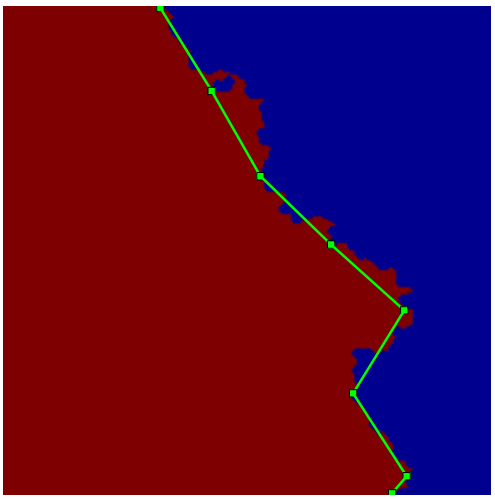
add 2)



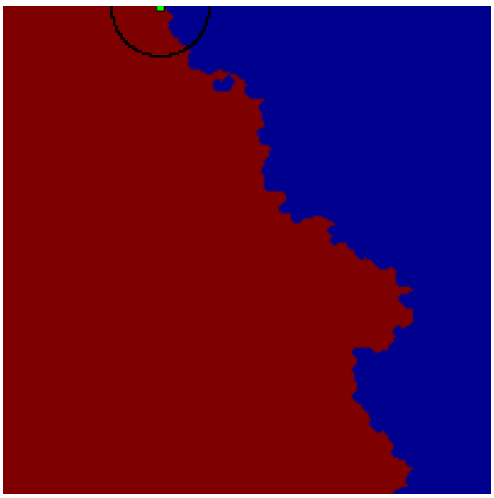


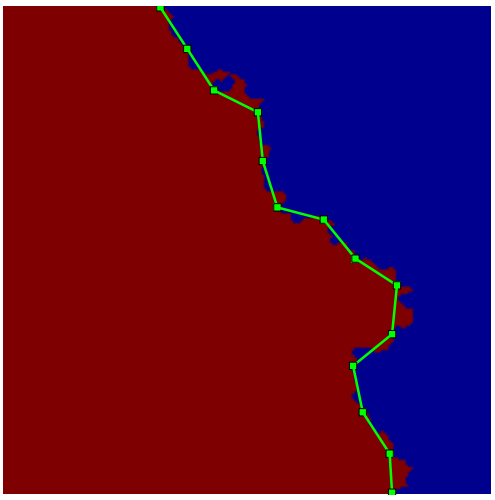
```
Ukazka mereni delky pobrezi pomoci kruzitka.

 Predpokladame, ze mapa predstavuje uzemi 100 x 100 km,
 Zadej delku, kterou nastavime na kruzitku
  (doporuceno 5 az 50 km) = 40

 Pocet namerenych usecek je 2.764138 .

 Celkova delka pobrezi je tedy soucinem poctu usecek a
 jejich delky, tj. 2.764138 x 40.000000 =  110.565503.
```





```
Ukazka mereni delky pobrezi pomoci kruzitka.

 Predpokladame, ze mapa predstavuje uzemi 100 x 100 km,
 Zadej delku, kterou nastavime na kruzitku
  (doporuceno 5 az 50 km) = 20

 Pocet namerenych usecek je 6.215058 .

 Celkova delka pobrezi je tedy soucinem poctu usecek a
 jejich delky, tj. 6.215058 x 20.000000 =  124.301163.
```

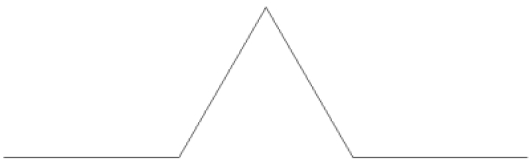




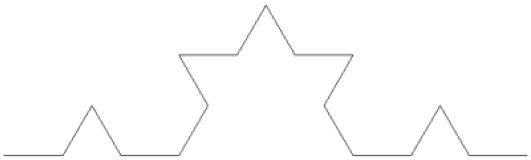
```
Ukazka mereni delky pobrezi pomoci kruzitka.

 Predpokladame, ze mapa predstavuje uzemi 100 x 100 km,
 Zadej delku, kterou nastavime na kruzitku
  (doporuceno 5 az 50 km) = 10

 Pocet namerenych usecek je 12.801561 .

 Celkova delka pobrezi je tedy soucinem poctu usecek a
 jejich delky, tj. 12.801561 x 10.000000 =  128.015610.
```

Uvažujeme následující fraktálovou strukturu (**Kochova křivka**).

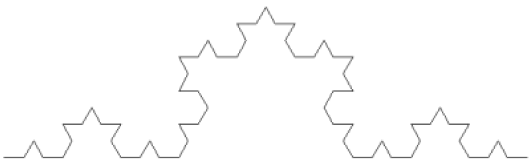
- Nejprve uvažujeme úsečku délky jedna.



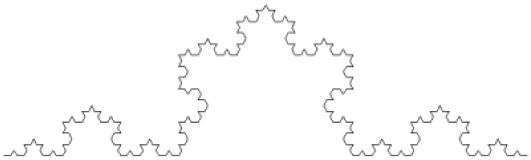
- Prostřední třetinu úsečky nahradíme dvěma stejně velkými úsečkami, takže s původní částí tvoří rovnostranný trojúhelník.



- Nad každou stranou této lomené čáry provedeme stejnou operaci a neustále opakujeme.



atd.



Otázky: Jak dlouhá je lomená čára ?
Jak velký obsah nad původní úsečkou vymezí tato lomená čára ?

Pro délku v jednotlivých krocích dostáváme:

1) Na počátku: 1 úsečka o délce jedna
2) Po 1. kroku: 4 úsečky o délce 1/3
3) Po 2. kroku: 16 úseček o délce 1/9
4) Po 3. kroku: 64 úseček o délce 1/27

$\vdots$

n) Po n krocích: 4^n úseček o délce $1/3^n$

Limitní přechod:

$$\text{Celková délka lomené čáry je} \quad \lim_{n\to\infty} 4^n \cdot \left(\frac{1}{3}\right)^n = \lim_{n\to\infty} \left(\frac{4}{3}\right)^n = \infty$$

Pro obsah plochy v jednotlivých krocích dostáváme:

1) Na počátku žádný obsah
2) Po 1. kroku přibude 1 trojúhelník o straně 1/3
3) Po 2. kroku přibude 4 trojúhelníky o straně 1/9
4) Po 3. kroku přibude 16 trojúhelníků o straně 1/27

$\vdots$

n) Po n krocích přibude 4^n trojúhelníků o straně délky $1/3^n$

(obsah rovnostranného trojúhelníka o straně a je $S_\triangle = \frac{1}{2}a \cdot v = \frac{1}{2}a \cdot \frac{\sqrt{3}}{2}a = \frac{\sqrt{3}}{4}a^2$)

Limitní přechod pro celkový obsah plochy pod lomenou čarou:

$$\sum_{n=1}^{\infty} 4^{n-1} \cdot \frac{\sqrt{3}}{4} \cdot (\frac{1}{3})^{2n} = \sum_{n=1}^{\infty} \frac{\sqrt{3}}{16} \cdot 4^n \cdot (\frac{1}{9})^n = \sum_{n=1}^{\infty} \frac{\sqrt{3}}{16} \cdot (\frac{4}{9})^n$$

= (součet nekonečné geometrické řady s koeficientem $q = \frac{4}{9}$ a prvním členem $\frac{\sqrt{3}}{36}$) =

$$= \frac{\sqrt{3}}{36} \cdot \frac{1}{1-\frac{4}{9}} = \frac{\sqrt{3}}{36} \cdot \frac{9}{5} = \frac{\sqrt{3}}{20} \doteq 0,08660254.$$

Závěr: Objekt má konečný obsah, ale nekonečný povrch!

Poznámka: Podobně je tomu v případě měření délky pobřeží.

Definice dimenze křivky

- Uvažujeme měření délky pomocí měřidla délky r.
- Aproximace délky křivky je dána počtem kroků měřidla podél křivky (N) vynásobené délkou měřidla $E(r) = N \cdot r$
- Pro hladké křivky konverguje $E(r)$ pro $r \to 0_+$ ke konečnému číslu ... délce křivky ... dimenze 1 (stačí popsat pouze vzdálenost od nějakého pevného bodu).
- Pro fraktály $E(r)$ diverguje k $+\infty \Rightarrow$ dimenze je <u>větší než 1</u>.
- Chceme najít hodnotu D_H takovou, že $N \cdot r^{D_H}$ je konečné nenulové číslo.

D_H ... <u>fraktální dimenze křivky</u> (Hausdorfova dimenze).

Pro $r \to 0_+$ a $N \to \infty$ má platit $(0 < c < \infty)$:

$$N \cdot r^{D_H} = c \qquad / \log$$

$$\log(N \cdot r^{D_H}) = \log c$$

$$\log N + D_H \log r = \log c$$

$$D_H = \frac{\log c - \log N}{\log r}$$

pro $r \to 0_+$ rovno nule

$$D_H = \frac{\log N - \log c}{\log \frac{1}{r}} = \frac{\log N}{\log \frac{1}{r}} + \underbrace{\frac{\log c}{\log r}}_{= 0 \text{ pro } r \to 0_+}$$

$$D_H = \lim_{r \to 0_+,\ N \to \infty} \frac{\log N}{\log \frac{1}{r}}$$

Příklady:

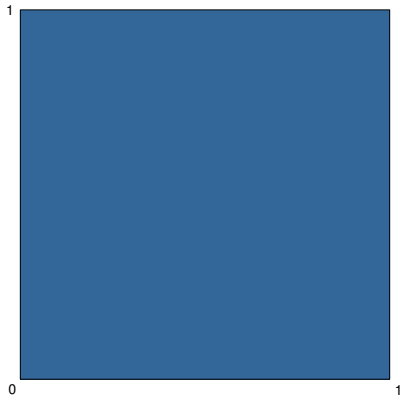
1) **Úsečka délky 1**

měřidlo délky r	počet kroků měření N
$r = 1$	$N = 1$
$r = 1/2$	$N = 2$
$r = 1/4$	$N = 4$
$\vdots$	$\vdots$
$r = 1/2^n$	$N = 2^n$

$$\frac{\log N}{\log \frac{1}{r}} = \frac{\log 2^n}{\log 2^n} = 1 \ldots \text{(dimenze je 1)}$$

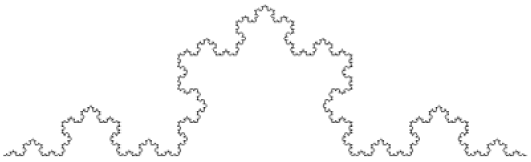
2) **Vyplněný čtverec o straně délky 1**

měřidlo délky r	počet kroků měření
$r = 1$	$N = 1$
$r = 1/2$	$N = 4$
$r = 1/4$	$N = 16$
$\vdots$	$\vdots$
$r = 1/2^n$	$N = 4^n$



$$\frac{\log N}{\log \frac{1}{r}} = \frac{\log 2^{2n}}{\log 2^n} = 2 \ldots \text{(dimenze je 2)}$$

3) **Kochova křivka**



v $n-$tém kroku jsme měli $\underbrace{4^n}_{=N}$ úseček o délce $\underbrace{\left(\frac{1}{3}\right)^n}_{=r}$

$$D_H = \lim_{N\to\infty r\to 0_+} \frac{\log N}{\log \frac{1}{r}} = \lim_{n\to\infty} \frac{\log 4^n}{\log 3^n} = \frac{\log 4}{\log 3} \doteq \underline{1,2619}$$

(neceločíselná hodnota dimenze)

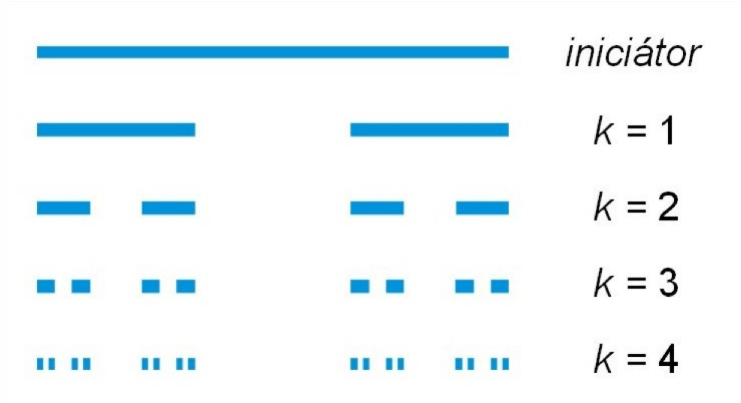
Jiný příklad jednoduché fraktálové struktury:

4) **Cantorovo diskontinuum**

Na počátku vezmeme úsečku.

V 1. kroku vypustíme prostřední $\frac{1}{3}$.

V každém dalším kroku vypouštíme prostřední $\frac{1}{3}$ z každé úsečky.



	délka	počet úseček	délka úsečky
na počátku	1	1	1
po 1. kroku	2/3	2	1/3
po 2. kroku	$(2/3)^2$	4	1/9
po 3. kroku	$(2/3)^3$	8	1/27
$\vdots$			
po $n-$tém kroku	$(2/3)^n$	2^n	$(1/3)^n$

$$\lim_{n\to\infty}\left(\frac{2}{3}\right)^n = 0 \quad \Rightarrow \quad \text{délka je } 0$$

Otázka: Jaká je fraktálová dimenze?

$$D_H = \lim_{n\to\infty}\frac{\log 2^n}{\log 3^n} = \frac{\log 2}{\log 3} \doteq 0,6309$$

Juliovy množiny

Vytvářeny pomocí iteračního předpisu pro $z_n \in \mathbb{C}$:

$$z_{n+1} = z_n^2 + c, \qquad z_n, c \in \mathbb{C}$$

z_n ... reprezentuje pozici bodu v komplexní rovině

c ... lib. zvolená konstanta, která je pro konkrétní obrazec stejná

$$\text{Juliova množina} \qquad J = \{z_0 \in \mathbb{C} : \lim_{n\to\infty} z_n \neq \infty\}$$

Prahový poloměr divergence z_0

Při praktickém počítání máme pouze konečný počet iterací, ze kterých musíme usoudit, zda posloupnost konverguje či nikoliv.

Platí: Pokud $\exists n \in N : |z_n| > r(c) = \max\{|c|, 2\}$. Potom posloupnost diverguje.

Důkaz:

$$|z_n| > 2 \;\text{ a }\; |z_n| > |c| \quad \Rightarrow \quad \exists \varepsilon > 0 : |z_n| = 2 + \varepsilon$$

Dále použijeme $\triangle$ nerovnost:

$$|z_n^2| = |z_n^2 + c - c| \leq |z_n^2 + c| + |c| \quad \Rightarrow \quad |z_n^2 + c| \geq |z_n^2| - |c|$$

Potom

$$|z_{n+1}| = |z_n^2 + c| \geq |z_n^2| - |c| \geq |z_n|^2 - |c| \geq |z_n|^2 - |z_n| = |z_n|(|z_n| - 1) > |z_n| \cdot (1 + \varepsilon)$$

$$|z_{n+1}| > 2 + 2\varepsilon$$

v dalším kroku se abs. hodnota z_n více zvětšila ($>$ než ε), tj. po dosazení dostaneme

$$|z_n| > (1 + \varepsilon)^n \cdot |z_0| \quad \Rightarrow \quad z_n \text{ diverguje} .$$

Vykreslení

- určíme oblast v komplexní rovině

$$a \leq \operatorname{Re} z \leq b, \;\; c \leq \operatorname{Im} z \leq d$$

- zvolíme diskretizační krok a pro zadanou konstantu c a postupně pro každý bod z_0 počítáme podle iterační formule $z_{n+1} = z_n^2 + c$. Jestliže $|z_n|$ překročí hodnotu $\max\{|c|, 2\}$ řekneme, že posloupnost diverguje. Pokud po zadaném počtu iterací nepřekročí $|z_n|$ hodnotu $\max\{|c|, 2\}$, rozhodneme, že posloupnost konverguje. Body v komplexní rovině obarvíme podle toho zda posloupnost konvergovala či divergovala.

Příklady:

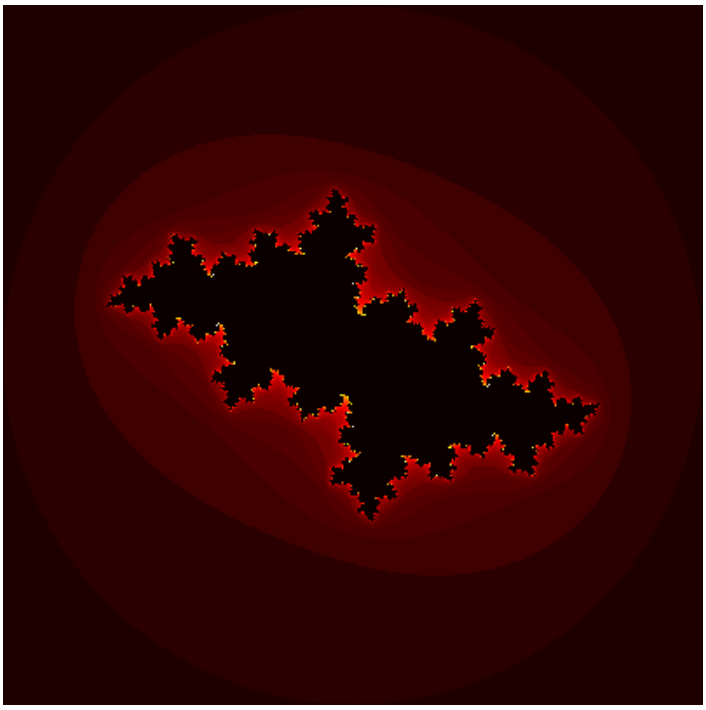
Juliovy množiny pro různé hodnoty c
na oblasti $-2 \leq \mathrm{Re}\, z \leq 2,\ -2 \leq \mathrm{Im}\, z \leq 2.$

1)

Juliova mnozina pro c=–0.097–0.64872i

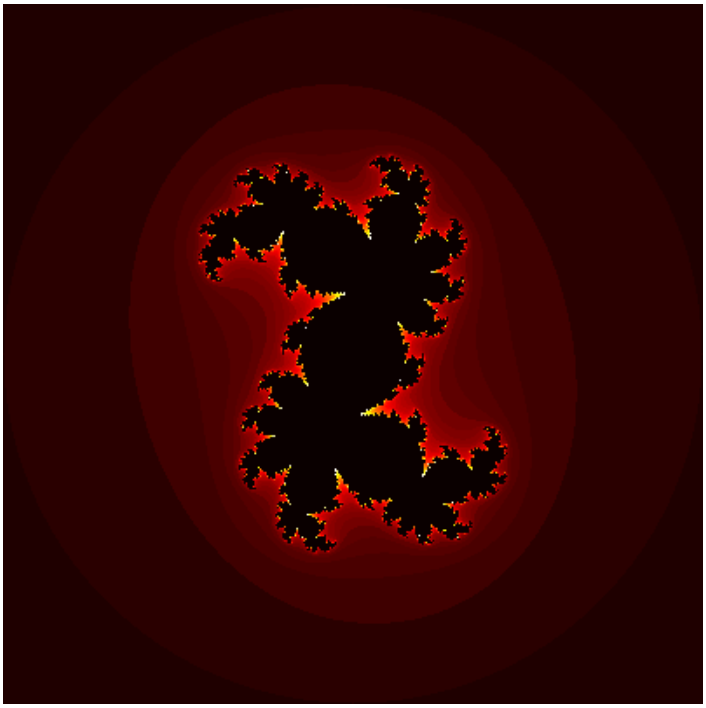
2)

Juliova mnozina pro c=–0.504+0.50197i



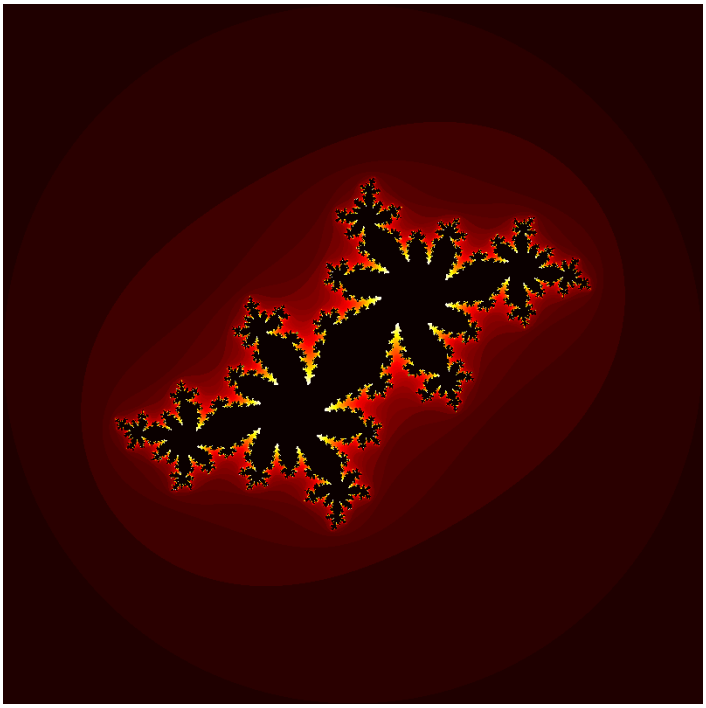
3)

Juliova mnozina pro c=0.37688+0.21646i



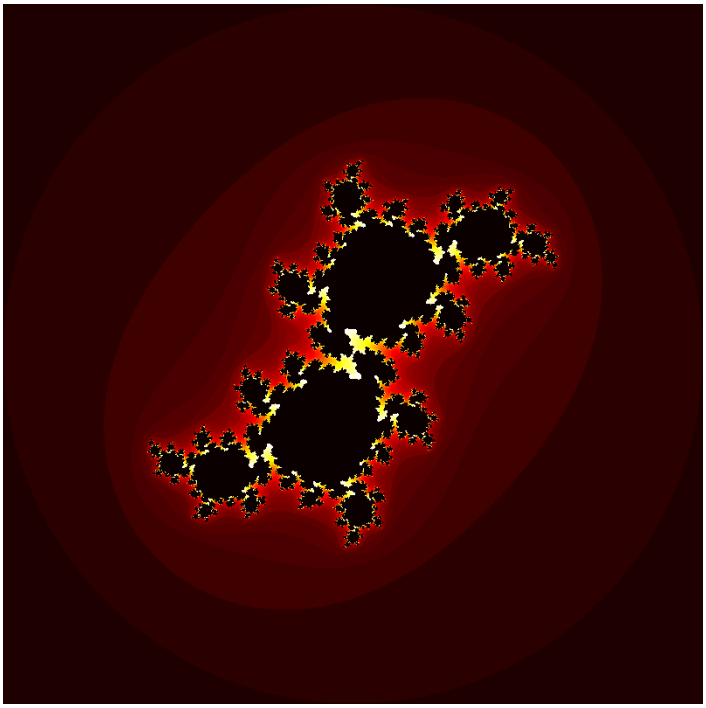
4)

Juliova mnozina pro c=−0.3555−0.62008i



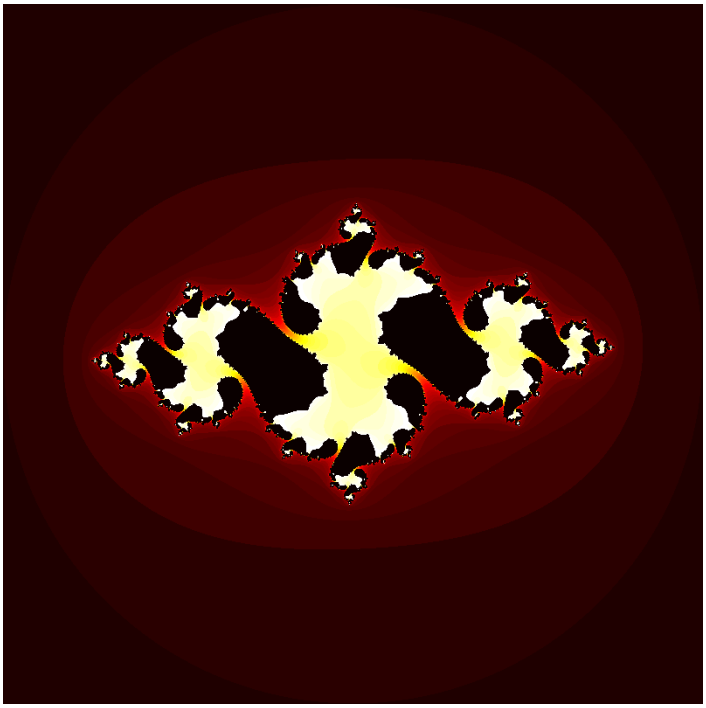
5)

Juliova mnozina pro c=0.060124−0.62423i



6)

Juliova mnozina pro c=−0.7515−0.072i



Mandelbrotova množina

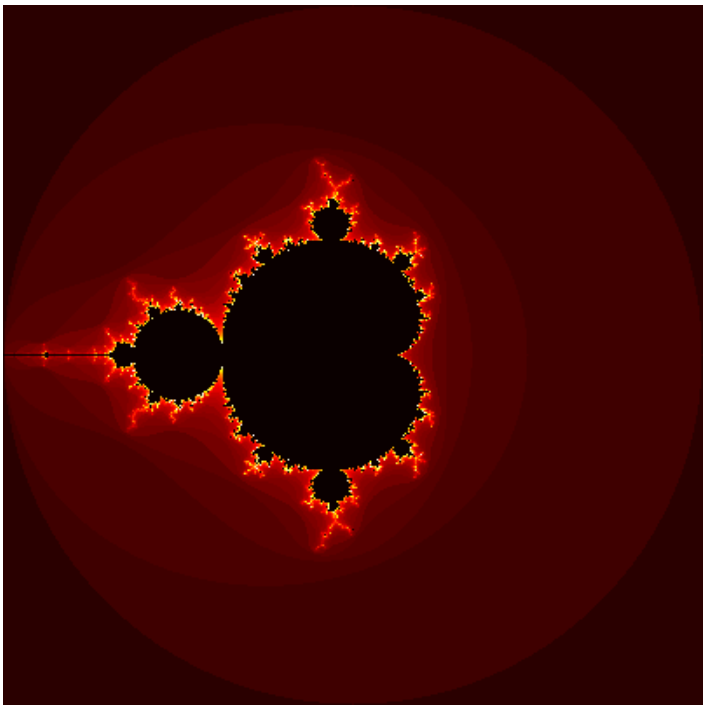
Opět využíváme iterační předpis

$$z_{n+1} = z_n^2 + c, \qquad z_n, c \in \mathbb{C}$$

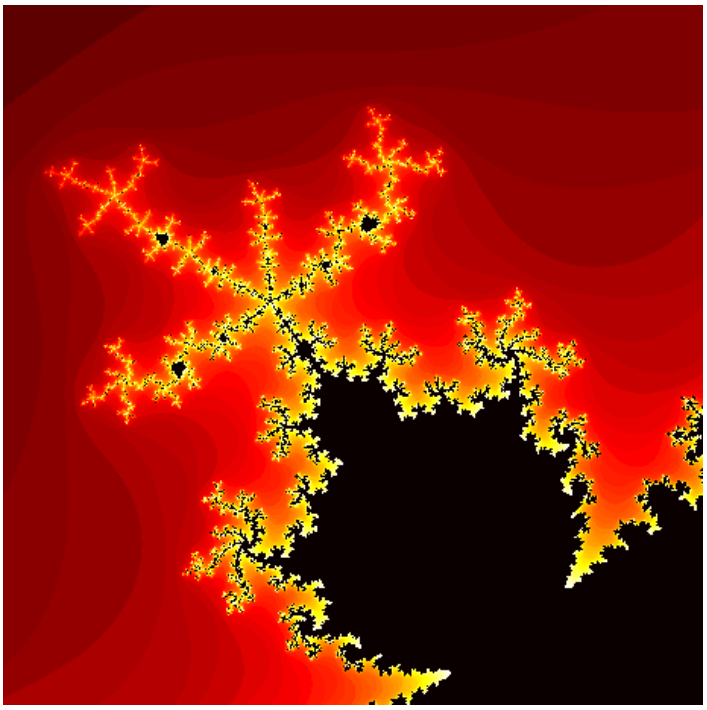
ovšem zde volíme $z_0 = 0$ a konstanta c reprezentuje pozici bodu v komplexní rovině

$$\text{Mandelbrotova množina} \qquad M = \{c \in \mathbb{C} : \lim_{n\to\infty} z_n \neq \infty\}$$

Příklad: Mandelbrotova množina na oblasti $-2 \leq \operatorname{Re} z \leq 2,\ -2 \leq \operatorname{Im} z \leq 2.$



Příklad: Mandelbrotova množina na oblasti $-0.65 \leq \operatorname{Re} z \leq -0.42,\ 0.51 \leq \operatorname{Im} z \leq 0.74$.



Motto: "Matematika je schovaná všude,
ovšem ne každý ji vidí"

Cílem přednášky je ukázat několik abstraktních matematických výsledků, které se používají při řešení praktických problémů. Některé matematické věty ovšem našly uplatnění až mnoho desítek či stovek let po svém vzniku.

Samoopravný kód (diskrétní matematika)

Slouží k detekci a opravování chyb vzniklých při přenosu dat.

Představme si situaci, že potřebujeme poslat data po telekomunikační lince, která není příliš spolehlivá, takže při přenosu může vlivem šumu dojít k řadě chyb. Pro nás je důležité zajistit, aby příjemce obdržel data bez chyb, a proto jsme ochotní poslat i více informací, pokud to pomůže případné chyby eliminovat. Protože je přenos drahý, chceme, aby celkový objem dat byl co možná nejmenší.

Data, která odesíláme ... posloupnost n bitů.

Pravděpodobnost chyby v přenosu jednoho bitu ... p (chyby jsou navzájem nezávislé).

1. přístup:

Data pošleme bez jakékoliv úpravy. Pravděpodobnost, že budou přijata bez chyby, tj. u všech bitů nedojde k chybě, je $(1-p)^n$. Pokud bude $n = 100$ a $p = 0,01$ (1%) vychází pravděpodobnost bezchybného příjmu 37%.

```
          Testovani posilani zprav bez uprav
 ------------------------------------------------------
  Zprava o 100 bitech je opakovane 100 krat poslana,
  s pravdepodobnosti 0.010000 nastane chyba pri prenosu 1 bitu.

  Prumerny pocet chyb pri prenosu jedne zpravy je 1.020000
  Z 100 prenosu bylo 33 bezchybnych,
  tj. 33.0 procent prenosu bylo bezchybnych.

 ------------------------------------------------------
  Teoreticky plati, ze pri posilani zpravy o n bitech,
  s pravdepodobnosti chyby pri prenosu 1 bitu p je
  pravdepodobnost bezchybneho prenosu cele zpravy:

    (1-p)^n = (1-0.010000)^100 = 0.366032
```

2. přístup:

Každý bit pošleme vícekrát (lichý počet), např. $3x$, a při příjmu zvolíme většinovou hodnotu.

Bit 0 odešleme jako 000 Bit 1 odešleme jako 111

Při příjmu dekódujeme trojice podle pravidel

000, 001, 010, 100 $\rightarrow$ 0 111, 110, 101, 011 $\rightarrow$ 1

Spočteme pravděpodobnost správného dekódování. Jednotlivý bit se dekóduje právě tehdy, když při přenosu příslušné trojice nastává nejvýše jedna chyba:

$$\underbrace{(1-p)^3}_{\text{bezchybný přenos}} + 3 \cdot \underbrace{p(1-p)^2}_{\text{přenos i-té složky s chybou}} = (1-p)^2(1+2p)$$

Pravděpodobnost správného dekódování n bitů získáme umocněním předchozího vztahu na n. Pro $n = 100$ a $p = 0,01$ vychází pravděpodobnost bezchybného příjmu 97%.

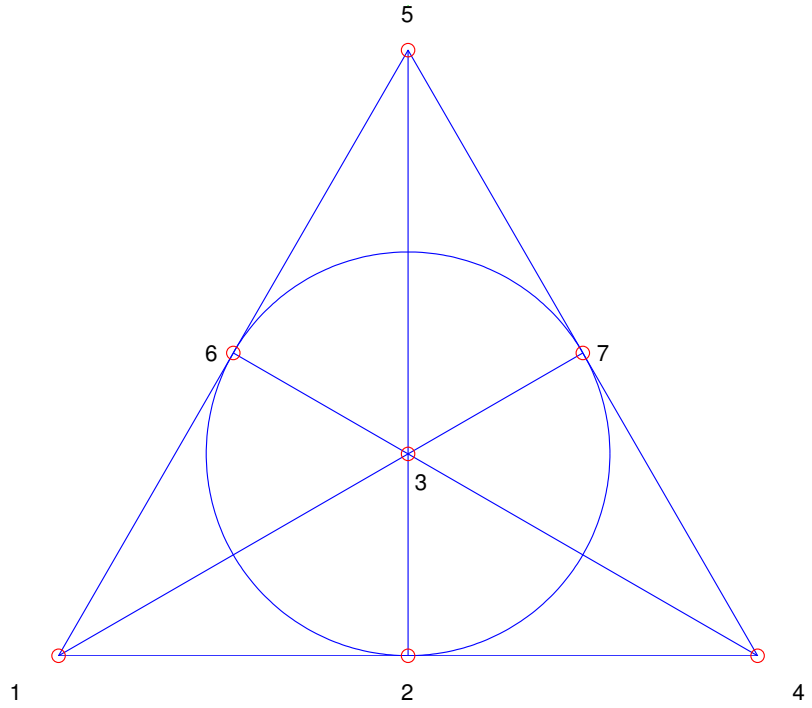
```
          Testovani posilani duplicitnich zprav
-------------------------------------------------------
 Puvodni zprava ma 100 bitu.
 Kazdy bit zpravy je poslan 3 krat za sebou.
 Pravdepodobnost chyby pri prenosu jednoho bitu je 0.01.
 Pri testovani cely proces opakujeme 100 krat.

 Prumerny pocet chyb pri prenosu jedne zpravy je 0.040000
 Z 100 prenosu bylo 96 bezchybnych,
 tj. 96.0 procent prenosu bylo bezchybnych.
 -------------------------------------------------------
 Teoreticky plati, ze pri posilani zpravy o n bitech,
 s pravdepodobnosti chyby pri prenosu 1 bitu p, pricemz
 kazdy bit posilame o krat, je pravdepodobnost bezchybneho
 prenosu opakovane poslaneho bitu dano souctem:

  (1-p)^o  ...  v o-tici zadna chyba
    = (1-0.010000)^3 = 0.970299

  (o nad 1) p^1 (1-p)^(o-1)  ...   v o-tici 1 chyb
    = (3 nad 1) 0.010000^1 (1-0.010000)^(3-1) = 0.029403

 v souctu pro jeden bit:  0.999702

 Pravdepodobnost bezchybneho prenosu cele zpravy:
  0.999702^n = 0.999702^100 = 0.970635
```

3. přístup: **Hammingův kód**

Uvažujeme tzv. hypergraf Fanovy roviny:

7 vrcholů (bodů) a 7 hran

Každá hrana obsahuje 3 body a každé dvě hrany se protínají právě v jednom bodě.



Body: 1, 2, 3, 4, 5, 6, 7.

Hrany: $\{1,2,4\}$, $\{1,3,7\}$, $\{1,5,6\}$, $\{2,3,5\}$, $\{2,6,7\}$, $\{3,4,6\}$, $\{4,5,7\}$.

Každé hraně přiřadíme tzv. charakteristický vektor, tj. vektor o sedmi složkách tak, že je-li $i-$tý bod bodem hrany bude na $i-$té pozici charakteristického vektoru 1 jinak 0.

$$\{1,2,4\} \to [1101000]$$
$$\{1,3,7\} \to [1010001]$$
$$\{1,5,6\} \to [1000110]$$
$$\{2,3,5\} \to [0110100]$$
$$\{2,6,7\} \to [0100011]$$
$$\{3,4,6\} \to [0011010]$$
$$\{4,5,7\} \to [0001101]$$

Těchto 7 charakteristických vektorů doplníme jejich 7 doplňky:

$$[0010111], [0101110], [0111001], [1001011], [1011100], [1100101], [1110010]$$

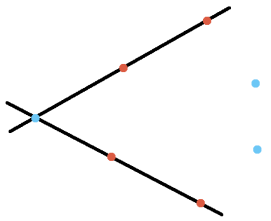
a dále vektory [1111111], [0000000].

Dostaneme 16 vektorů, tzv. kódová slova. Platí zajímavá vlastnost:

Věta: Každá 2 různá kódová slova se liší alespoň ve 3 souřadnicích.

Poznámka:

A Vezmu-li si libovolné 2 hrany v původním hypergrafu, mají vždy společný právě jeden bod, a tudíž zbylé 2 body na každé hraně jsou navzájem různé, tj. zbývají dva body, které nejsou ani v jedné ze zadaných hran. ⇒ Pro libovolné 2 různé hrany původního hypergrafu platí, že se jejich charakteristické vektory shodují právě ve 3 souřadnicích a zbylé 4 mají odlišné.



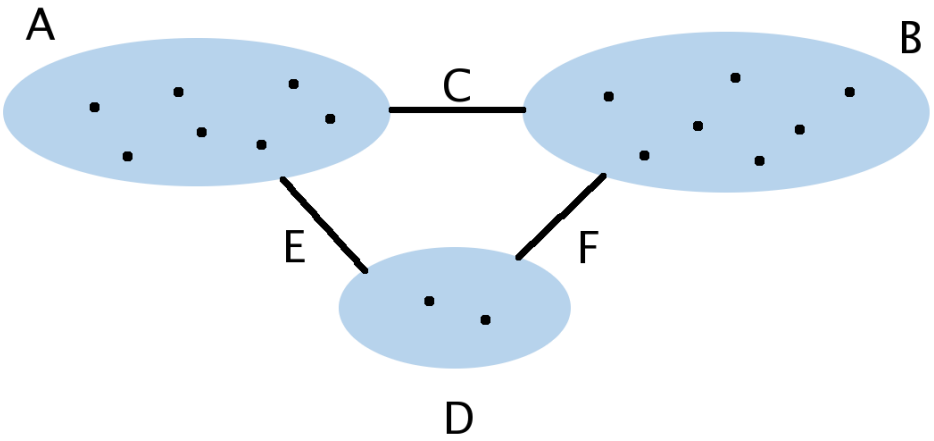
B Pokud budeme uvažovat skupinu 7 doplňkových vektorů, bude pro ně platit stejná vlastnost jako pro skupinu původních vektorů.

C Pokud vezmeme libovolný vektor z původních 7 a přidáme libovolný vektor z doplňkových, potom můžeme určit vzor doplňkového vektoru a pro něj a pro vybraný vektor z původních opět musí platit, že mají shodné 3 souřadnice a zbylé 4 mají odlišné, nebo jde o stejný vektor. Pokud se opět vrátíme k doplňkovému vektoru, potom bude platit, že na začátku zvolené vektory se budou shodovat na 4 pozicích a lišit se budou na 3 pozicích nebo se liší na všech pozicích.

D Pokud uvážíme 2 poslední doplněné vektory [1111111] a [0000000], ty se liší na všech pozicích.

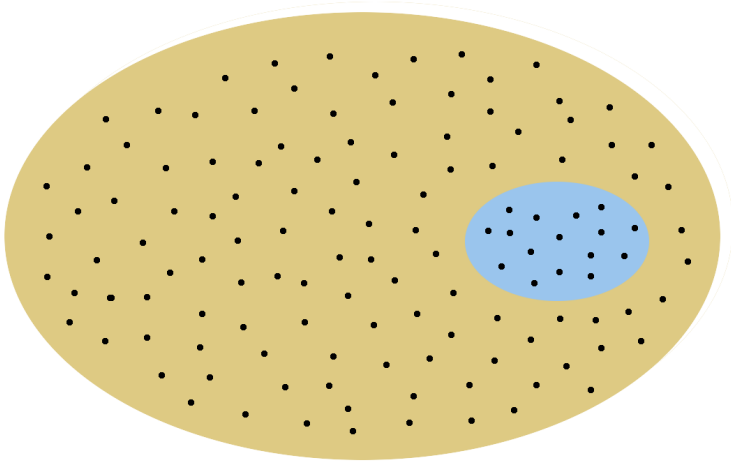
E Pokud budu srovnávat lib. vektor z původních 7 vektorů s vektorem [1111111], budou se lišit ve 4 pozicích, protože každá hrana měla právě tři body (tj. vektor měl vždy 3 jedničky); analogicky pokud budu srovnávat lib. vektor z původních 7 s vektorem [0000000], budou se lišit ve třech pozicích.

F Pro srovnání skupiny doplňkových vektorů s [1111111] a [0000000] platí analogicky jako v E.



Ukázali jsme, že lib. dvojice vektorů ze všech 16 vektorů splňuje to, že se liší minimálně ve 3 souřadnicích.

Podívejme se na problém z druhé strany: Uvažujeme lib. vektor o 7 složkách, kde na každé pozici může být 0 nebo 1. Těchto vektorů je $2^7 = 128$ a mezi nimi i těch 16 našich.



Předchozí věta se dá upřesnit takto:

Věta: Ve skupině našich 16 vektorů se lib. 2 z nich liší buď na 3, 4 nebo 7 pozicích.
(Dokážeme, projdeme-li body A, B, ..., F.)

Důsledek: Pokud ke každému z našich 16 vektorů přiřadíme 7 vektorů tak, že se tyto budou lišit právě v jedné ze 7 pozic, dostaneme celkem $8 \cdot 16 = 128$ vektorů, to jsou ovšem všechny možnosti, které máme, protože ke každým 2 různým vektorům ze 16 původních jsou přiřazeny různé "modifikované vektory".

A, B ... vektory ze "16"

A ... modifikujeme na $\widetilde{A}$ (v 1 pozici)

B ... modifikujeme na $\widetilde{B}$ (v 1 pozici)

A a B se liší minimálně ve třech pozicích $\Rightarrow$ $\widetilde{A}$ a $\widetilde{B}$ se liší minimálně v 1 pozici.

Dostaneme silné tvrzení:

Pro každý vektor délky 7 (obsahující 0 nebo 1) existuje právě jeden vektor z naší "16" tak, že se buď shodují nebo se liší právě na jedné pozici.

$\Rightarrow$ Pokud se při přenosu poruší nejvýše jeden bit, lze ho opravit.

Využití: Vrátíme se k problému přenést $n-$tici bitů. Tu rozdělíme na bloky o čtyřech bitech. Potom každý blok je dvojkovým zápisem jednoho čísla od 0 do 15 (16 možností). Místo, abychom posílali tyto bloky, budeme posílat naše kódové vektory ze "16", které jednoznačně přiřadíme číslům 0 a 15.

Příklad:

Chceme poslat posloupnost 0100 1101 1111 0011 0101.

Získáme bloky 0100 1101 1111 0011 0101.

Místo abychom posílali tyto bloky, pošleme příslušné kódové vektory, tj.

$$0100011 \mid 1101000 \mid 1111111 \mid 0011010 \mid 0101110\ .$$

Všimněme si, že pro každou čtveřici číslic existuje právě jeden vektor z naší "16", který takto začíná.

```
       Zprava vyuzivajici Hamminguv kod
-------------------------------------------------

 Zadej zpravu v binarnim tvaru  = 01001101111100110101

 Zpravu rozdelime na bloky po 4 bitech:
  0100   |1101   |1111   |0011   |0101   |

 Bloky doplnime na kodova slova:
  0100011|1101000|1111111|0011010|0101110|
```

Přenášená posloupnost je delší než n bitů, konkrétně $n+\frac{3}{4}n = \frac{7}{4}n$ bitů. Pravděpodobnost, že při přenosu jednoho bloku (kódového vektoru) nastane nejvýše jedna chyba je:

$$(1-p)^7 + 7 \cdot p \cdot (1-p)^6 = (1-p)^6(1+6p).$$

Kódových slov je $\frac{n}{4}$, a proto celková pravděpodobnost, že se data správně dekódují je

$$\left((1-p)^6(1+6p)\right)^{\frac{n}{4}} = (1-p)^{\frac{3n}{2}} \cdot (1+6p)^{\frac{n}{4}}.$$

Pro náš případ $n = 100$ a $p = 0,01$ vyjde $\approx 95\%$.

To je skoro stejně vysoká pravděpodobnost jako, když jsme informaci posílali 3x za sebou. Objem přenášených dat ale není $3n$, ale pouze $\frac{7}{4}n$, tj. stačí přenášet $\approx 58\%$ dat.

```
        Testovani posilani kodovanych zprav
-------------------------------------------------------
 Puvodni zprava ma 100 bitu.
 Pravdepodobnost chyby pri prenosu jednoho bitu je 0.010000.
 Pri testovani cely proces opakujeme 100 krat.

 Prumerny pocet chybnych bloku pri prenosu jedne zpravy je 0.040000
 Z 100 prenosu bylo 96 bezchybnych,
 tj. 96.0 procent prenosu bylo bezchybnych.

-------------------------------------------------------
 Teoreticky plati, ze pri posilani zpravy o n bitech,
 s pravdepodobnosti chyby pri prenosu 1 bitu p, pricemz
 posilame n + 3/4 n bitu (doplnili jsme kodova slova),
 je pravdepodobnost bezchybneho prenosu jednoho bloku
 (mohla nastat nejvyse jedna chyba):

  (1-p)^7 + 7p(1-p)^6  = (1+6p)(1-p)^6  =
    = (1+6*0.010000)(1-0.010000)^6  = 0.997969

 Kodovych slov je posilano n/4, proto
 pravdepodobnost bezchybneho prenosu cele zpravy je

  ( (1+6p)(1-p)^6 )^(n/4)  = 0.950442
```

Poznámky: Kontrolní mechanismy se používají v řadě případů, například při přidělování **rodného čísla** (v jiných státech se používají různé alternativy identifikačního čísla).
https://cs.wikipedia.org/wiki/Rodné_číslo

850916/1573

Dalším příkladem je konstrukce **ISBN**, tj. identifikačního čísla pro knihy,
https://cs.wikipedia.org/wiki/International_Standard_Book_Number

podobné zabezpečení má v sobě i **čárový kód**, který je dnes vlastně na každém výrobku,
https://cs.wikipedia.org/wiki/Čárový_kód

ISBN 978-0-7334-2609-4

9780733426094

dále **QR kód**, atd. atd.
https://cs.wikipedia.org/wiki/QR_kód
https://ipadvetride.cz/tag/qr-kod
https://www.qrgenerator.cz/

Teorie grafů

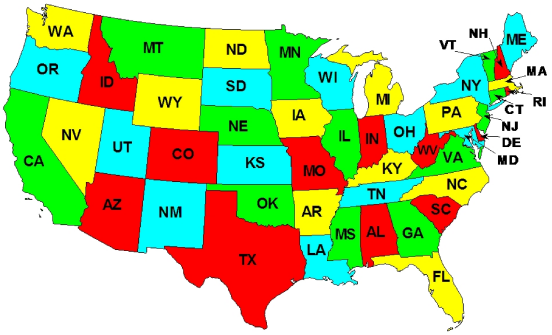
Obyčejný graf je dvojice (V, E), kde V je množina vrcholů (uzlů) a E je množina hran a každé 2 vrcholy spojuje nejvýše 1 hrana.

Graf je rovinný, jestliže existuje takové jeho nakreslení, že se žádné 2 hrany nekříží. Je-li navíc "souvislý" (pro každé 2 vrcholy existuje spojení), pak platí tzv. **Eulerův vztah**:

$$v + u = e + 2,$$

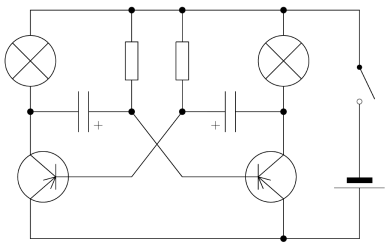
v ... počet vrcholů, e ... počet hran, u ... počet stěn.

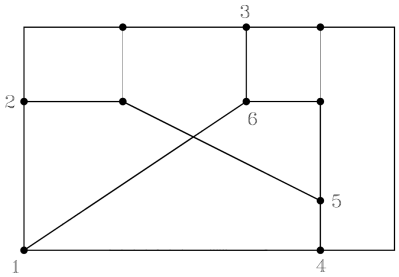
- V roce 1976 byl vyřešen **problém 4 barev**, tj. dokázalo se, že každou mapu lze obarvit nejvýše 4 barvami tak, že žádné dva sousední státy nemají stejnou barvu.



- Návrh integrovaných obvodů a desek s plošnými spoji.

  Př.: Schéma obvodu pro dvoužárovkový blikač a odpovídající graf.

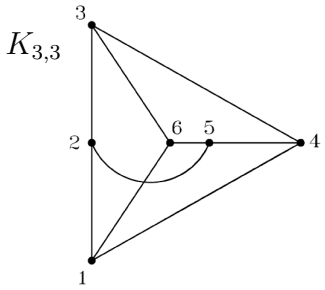


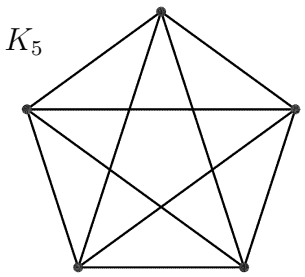


Otázka: Existuje takové nakreslení grafu, že se žádné 2 vodiče nekříží?

Řešení vychází z následující věty.

Věta: Obyčejný graf je rovinný právě tehdy, když neobsahuje část izomorfní s dělením grafu $K_{3,3}$ nebo K_5. (1930 Kuratowsky, složitý důkaz)





Snadno ověříme, že pokud budeme uvažovat pouze očíslované uzly a vynecháme tenké hrany, dostaneme přesně graf $K_{3,3}$.

$\Rightarrow$ Graf není rovinný.

Motto: "Matematika je schovaná všude, ovšem ne každý ji vidí"

Cílem přednášky je ukázat několik abstraktních matematických výsledků, které se používají při řešení praktických problémů. Některé matematické věty ovšem našly uplatnění až mnoho desítek či stovek let po svém vzniku.

Teorie čísel - šifrování zpráv

V dřívějších dobách se k šifrování používal *tajný klíč*. Nevýhodou bylo, že jej musely znát všechny komunikující strany a snadno mohlo dojít k jeho prozrazení. Ukážeme si metodu **RSA**, která slouží pro bezpečné šifrování pomocí veřejného klíče. Název metody je odvozen od iniciálů autorů: R. I. **R**ivest, A. **S**hamir, L.M. **A**dleman (1978).

Princip metody:

Alice a Bob si potřebují posílat zprávy. Nejprve textové zprávě přiřadí přirozené číslo (např. pomocí ASCII kódu). Pro vlastní šifrování si Alice a Bob vyberou každý 2 velká prvočísla (alespoň 200 cifer). Označíme je p_A, q_A pro Alici a p_B, q_B pro Boba. Každý si svá prvočísla vynásobí a získá číslo $n_A = p_A \cdot q_A$ (Alice) a $n_B = p_B \cdot q_B$ (Bob). Dále je ještě nutné zvolit tzv. **šifrovací exponenty** e_A (Alice) a e_B (Bob) a vypočítat **dešifrovací exponenty** d_A a d_B (podrobnosti viz. dále).

Každý zveřejní dvojici (n_A, e_A) a (n_B, e_B). A pokud chce jeden druhému poslat zprávu, použije pro šifrování klíč toho druhého. Řekneme, že Bob chce poslat Alici zprávu. Po převedení do ASCII kódu ji označíme X a nechť platí, že $X < n_A$ (pokud by bylo větší, rozdělila by se zpráva na více menších). šifrovanou zprávu označíme X^* a získáme jí z vlastnosti (kongruence):

$$X^* \equiv X^{e_A} \mod n_A,$$

jiným způsobem řečeno: dělíme-li X^{e_A} číslem n_A, vyjde X^* jako celočíselný zbytek ($X^* < n_A$). Alice zprávu dešifruje na číslo $(X^*)^*$ pomocí vlastnosti:

$$(X^*)^* \equiv (X^*)^{d_A} \mod n_A.$$

Otázka: Jak se určí e_A, d_A, aby platilo, že se dešifrovaná zpráva $(X^*)^*$ rovná původní zprávě X^*?

- Důležitou úlohu hraje tzv. Eulerova funkce $\varphi(n)$, která je definována jako počet přirozených čísel nepřevyšujících n, jež jsou s n nesoudělná.
- Je-li n dáno součinem 2 prvočísel ($p \neq q$), potom platí:

$$\varphi(n) = \underbrace{(p-1)}_{\varphi(p)} \cdot \underbrace{(q-1)}_{\varphi(q)} = \underbrace{p \cdot q}_{n} - (\underbrace{p}_{(1)} + \underbrace{q-1}_{(2)}),$$

(1) musím odečíst p násobků čísla q,

(2) musím odečíst q násobků čísla p, ale pq jsem už odečetl.

Příklady:

a) $125 \equiv 6 \mod 7$

$125 : 7 = 17$
55
-49
6

b) $100 \equiv 0 \mod 20$

$100 = 5 \cdot 20$

c) $1000 \equiv 12 \mod 13$

$1000 : 13 = 76$
-91
90
-78
12

- Šifrovací a dešifrovací exponent musí splňovat podmínku:

$$e \cdot d \equiv 1 \mod \varphi(n)$$

- Šifrovací exponent e musí být zvolen tak, aby byl s $\varphi(n)$ nesoudělný.

Odpověď na otázku:

Jsou-li e_A a $\varphi(n_A)$ nesoudělná a $e_A \cdot d_A \equiv 1 \mod \varphi(n_A) \quad \Rightarrow \quad (X^*)^* = X$

Důkaz je založen na **Euler-Fermatově větě** (18. století)

Pro nesoudělná x a n platí: $x^{\varphi(n)} \equiv 1 \mod n$.

Hlavní trik metody RSA

Vynásobit 2 velká prvočísla je velmi snadné, zatímco zpětně rozložit tento součin na prvočinitele není v současnosti v rozumném čase možné.

Pokud neznám rozklad na prvočinitele, nemůžu určit hodnotu Eulerovy funkce a tudíž nemohu určit dešifrovací exponent (ten je tajný a bez něj zprávu nelze dešifrovat).

Náznak důkazu:

Š: $X^* \equiv X^{e_A} \mod n_A \quad \Rightarrow \quad (X^*)^{d_A} \equiv (X^{e_A})^{d_A} = X^{e_A \cdot d_A} \mod n_A \qquad (A)$

D: $(X^*)^* \equiv (X^*)^{d_A} \mod n_A \qquad (B)$

Dále platilo:

$$e_A \cdot d_A \equiv 1 \mod (p_A - 1) \text{ a}$$
$$e_A \cdot d_A \equiv 1 \mod (q_A - 1).$$

Malá Fermatova věta: $\forall p$ prvočíslo a $\forall a \in Z$ nesoudělné: $a^{p-1} \equiv 1 \mod p$.

$\Rightarrow \qquad X^{e_A \cdot d_A} \equiv X \mod p_A$

$\Rightarrow \qquad X^{e_A \cdot d_A} \equiv X \mod q_A$

Čínská věta o zbytcích $\Rightarrow \quad X^{e_A \cdot d_A} \equiv X \mod \underbrace{p_A \cdot q_A}_{n_A} \qquad (C)$

$(A) + (C) \Rightarrow \qquad (X^*)^{d_A} \equiv X \mod n_A \qquad (D)$

$(B) + (D) \Rightarrow \qquad (X^*)^* = X$

□

Ukázkový příklad:

Uvažujeme velmi malá prvočísla - v praxi se používají o mnoho řádů větší.

$p_A = 61$ a $q_A = 53$ jsou dvě zvolená prvočísla

$n_A = p_A \cdot q_A = 3\ 233 \quad \ldots$ **modul** (veřejný)

$e_A = 17 \ldots$ zvolený **šifrovací exponent** tak, aby byl nesoudělný s

$$\varphi(n_A) = (p_A - 1)(q_A - 1) = 60 \cdot 52 = 3120$$

$d_A = 2\ 753 \ldots$ vypočtený soukromý **dešifrovací exponent** tak, aby platilo:

$d_A \cdot e_A \equiv 1 \mod \varphi(n_A)$

čili $d_A \cdot 17 \equiv 1 \mod 3120 \quad (d_A < n_A$ takové, že toto splňuje je jediné)

Veřejný klíč = modul + šifrovací exponent: $n_A = 3233$ a $e_A = 17$

```
   Generovani verejneho a tajneho klice pro sifrovani
-----------------------------------------------------------

 Prvni zadane prvocislo     p = 61 .
 Druhe zadane prvocislo     q = 53 .
 Zadany sifrovaci exponent  e = 17 .

 Verejny modul    n = p * q  = 3233
 Eulerova funkce  phi = (p-1) * (q-1) = 3120

 Spravne zadani pro sifrovani:
 -----------------------------

  1. prvocislo je   p = 61
  2. prvocislo je   q = 53
  Eulerova funkce phi = 3120
  verejny klic je   n = 3233, e = 17
  tajny klic   je   d = 2753
```

Chceme zašifrovat třeba zprávu $X = 123$.

$$X^* = 123^{17} \mod 3233 = 855$$

Dešifrujeme

$$(X^*)^* = 855^{2753} \mod 3233 = 123 = X.$$

```
                  Sifrovaci metoda RSA
-----------------------------------------------------------

 Prvni zadane prvocislo     p = 61 .
 Druhe zadane prvocislo     q = 53 .
 Zadany sifrovaci exponent  e = 17 .

 Verejny modul    n = p * q  = 3233
 Eulerova funkce  phi = (p-1) * (q-1) = 3120


  Verejny klic je   n = 3233, e = 17
  Tajny klic   je   d = 2753

  Zadej zpravu  ( 0<x<3233 )       x=123
  Sifrovana zprava je     855
  Desifrovana zprava je   123
```

Digitální podpis:

Pokud chce mít Alice jistotu, že zprávu X^* skutečně poslal Bob, pak Bob musí za X^* připojit číslo Y^*, které získá takto:

Elektronický podpis

$$Y^* \equiv X^{d_B} \mod n_B.$$

Alice po obdržení spočítá

$$\underbrace{(Y^*)^*}_{=X^*} = (Y^*)^{e_B} \mod n_B$$

Pokud při použití veřejného klíče (e_B, n_B) na dešifrování Y^* dostanu stejnou zprávu jako při dešifrování X^* pomocí mého tajného klíče (d_A, n_A), pak vím, že zprávu odeslal majitel klíče (e_B, n_B).

Ukázkový příklad (viz předchozí):

veřejný klíč příjemce $n_A = 3233$, $e_A = 17$

tajný klíč příjemce $d_A = 2753$

veřejný klíč odesílatele $n_B = 6319$, $e_B = 29$

tajný klíč odesílatele $d_B = 2549$

zpráva $X = 123$

šifrovaná zpráva $X^* = 855$

zprávu $X = 123$ zašifruji znovu, tentokrát s použitím svého tajného klíče:

$$123^{2549} \mod 6319 = 4662$$

Zašifrovaný podpis připojím za šifrovanou zprávu.

Dále příjemci pošlu můj veřejný klíč, aby ho použil pro identifikaci podpisu

$$4662^{29} \mod 6319 = 123.$$

Příjemce po dešifrování obdržel stejnou zprávu 2x.

Podpis mohl poslat pouze držitel tajného klíče k zaslanému veřejnému klíči.

Poznámka: Prakticky se jako digitální podpis neposílá celá zašifrovaná zpráva, ale pouze její tzv. *otisk*. Otisk je zhuštění původní zprávy (takové, že při změně původní zprávy se mění i její otisk) https://cs.wikipedia.org/wiki/Hašovací_funkce).

Příjemce tedy kontroluje shodnost dešifrovaného otisku a otisku vytvořeného z původní nešifrované zprávy.

Poznámka: Pokud by se posílal podpis pro celou zprávu i s veřejným klíče odesílatele, mohl by si každý zprávu dešifrovat a tím by se celá zpráva prozradila.

```
                    Digitalni podpis
-------------------------------------------------------

 Zadana data pro prijemce:
---------------------------
    p_prijemce = 61
    q_prijemce = 53
    e_prijemce = 17

 Zadana data pro odesilatele:
------------------------------
    p_odesilatele = 71
    q_odesilatele = 89
    e_odesilatele = 29


 Spravne zadani pro sifrovani pro prijemce:

  verejny klic prijemce je     n = 3233, e = 17
  tajny klic prijemce je       d = 2753


 Spravne zadani pro sifrovani pro odesilatele:

  verejny klic odesilatele je  n = 6319, e = 29
  tajny klic odesilatele je    d = 2549

  Zadej zpravu  ( 0<x<min(3233,6319) )       x=123
  Sifrovana zprava je     855
  Sifrovany podpis je     4662

  Desifrovana zprava je   123
  Pro desifrovani podpisu pouzij verejny klic odesilatele.
  Desifrovany podpis je   123
```

Teorie matic - řešení SLAR

Celá řada aplikací vede ve své podstatě na problém řešit soustavu lineárních algebraických rovnic (SLAR)

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

$\mathbf{A}$ ... regulární matice typu N x N (det $\mathbf{A} \neq 0$, $\forall$ vl. č. $\neq 0$, hod$(\mathbf{A}) = N$)

$\mathbf{b}$ ... sloupcový vektor o N složkách

$\mathbf{x}$ ... hledané řešení (sloupcový vektor o N složkách)

1. způsob řešení: **Cramerovo pravidlo**

i-tou složku vektoru řešení vypočteme ze vztahu

$$x_i = \frac{\det \mathbf{A}_i}{\det \mathbf{A}}.$$

$\mathbf{A}_i$ ... vznikne z původní matice tak, že $i-$tý sloupec nahradíme pravou stranou $\mathbf{b}$.

Počet operací:

- musíme vypočítat $N + 1$ determinantů
- při výpočtu determinantu je třeba $N!$ sčítání a v každém sčítanci je N - 1 násobení

Celkem operací:

$$(N + 1) \cdot [(N - 1)N! + N!] = N(N + 1)!$$

2. způsob řešení: **Gaussova eliminační metoda**

- nejprve převedeme na $\triangle$ tvar
- zpětným chodem dosazujeme a počítáme složky řešení

Počet operací:

- v přímém chodu postupně bereme každý řádek o N složkách a jeho násobek přičítáme ke zbývajícím ... N^2
- to opakujeme, abychom vynulovali všechny sloupce pod hlavní diagonálou ... N^3

Celkem operací přesněji

$$\frac{2}{3}N^3$$

3. způsob řešení: **Metoda sdružených gradientů** (1952 M. R. Hesteners, E. Stiefel)

- metoda pro symetrické, pozitivně definitní matice

$$\mathbf{A} = \mathbf{A}^T, \qquad \mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \ \text{ pro } \ \mathbf{x} \neq 0$$

- konverguje k řešení (nalezne řešení) po N krocích

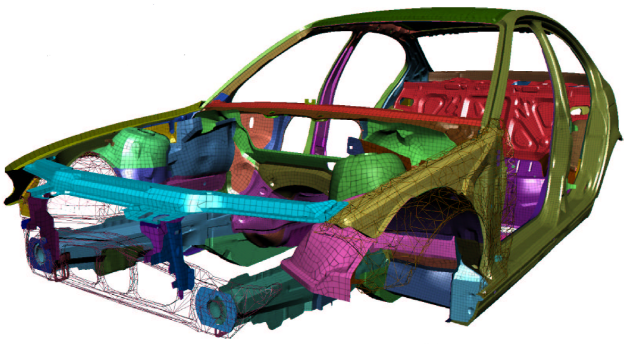
- počet iterací je řádově $2N^3$ (pro plné matice)

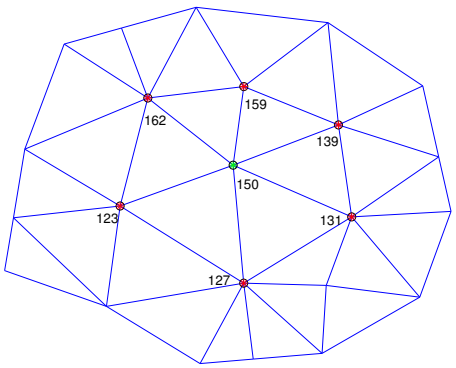
- metoda je odvozena pro řešení ekvivalentního problému:

Věta: Nechť $\mathbf{A}$ je symetrická a pozitivně definitní matice. Pak $\widehat{\mathbf{x}}$ je řešením soustavy $\mathbf{A}\mathbf{x} = \mathbf{b} \ \Leftrightarrow$ minimalizuje funkcionál $J(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$ na prostoru $\mathbb{R}^{\mathbb{N}}$.

- při jednotlivých iteracích metody sdružených gradientů se funkcionál J minimalizuje na podprostorech, jejichž dimenze postupně vzrůstá

V úlohách, kde matice $\mathbf{A}$ je plná vychází nejlépe použití Gaussovy eliminační metody. Situace se výrazně změní, pokud bude matice $\mathbf{A}$ řídká. To nastane velmi často, např. při použití metody konečných prvků dostáváme matice např. pro $N = 1000\,000$ neznámých, které jsou ovšem velmi řídké a mají řádově N prvků. Očíslujeme-li vrcholy pak bude mít matice nenulové prvky pouze v pozicích $[i, j]$ takových, že i a j jsou sousední vrcholy (spojeny hranou).





Pokud je matice $\mathbf{A}$ řídká, stačí pro její uložení použít řádově N buněk. V průběhu metody sdružených gradientů se matice $\mathbf{A}$ nemění, zatímco pokud bychom k řešení použili GEM, ztratili bychom vlastnost řídkosti, matice se postupně začne zaplňovat a pro její uložení budeme potřebovat opět řádově N^2 buněk.

V důsledku velké rychlosti konvergence metody sdružených lze ukončit proces dříve než po N krocích. Pro trojrozměrné úlohy vyžaduje GEM řádově $N^{\frac{7}{3}}$ operací, zatímco metoda sdružených gradientů řádově $N^{\frac{4}{3}}$ operací a tzv. metoda sdružených gradientů s předpodmíněním jen $N^{\frac{7}{6}}$ operací.

V tabulce uvedeme příslušné časy pro řešení $\mathbf{Ax} = \mathbf{b}$ různými metodami pro $N = 10^3$ a $N = 10^6$ a rychlosti 10^6 operací za sekundu.

Metoda	Plná matice		Řídká matice	
	$N = 10^3$	$N = 10^6$	$N = 10^3$	$N = 10^6$
Cramerovo pravidlo	$4 \cdot 10^{2567}$s	$(\star)$		
GEM	667 s	21125 let	10 s	3,17 roku
Sdružené gradienty			0,01 s	100 s
Předpodmíněné sdružené gradienty			0,0032 s	10 s

Pozn.: Rok má 60 x 60 x 24 x 365,25 = 3,15576 x 10^7 sekund.

$(\star)$ $1000000! \approx 8{,}2639 \cdot 10^{5565708}$ $\qquad \dfrac{N(N+1)!}{10^6} = (N+1)! \ldots$ pro $N = 10^6$

$$\frac{1000001 \cdot 1000000!}{3,15576 \cdot 10^7} = \frac{8,2639 \cdot 10^{5565708}}{31,5576} = 2,619 \cdot 10^{5565707} \text{ let.}$$

Motto: "Je příroda kolem nás spojitá nebo diskrétní ?"

Uvažujme např. stůl. Každý, kdo se o něj opře asi prohlásí, že je spojitý (nejsou v něm žádné díry). Pokud půjdeme k větším detailům a použijeme mikroskop, zjistíme, že spojitý není. Pokud půjdeme na úroveň atomů je situace ještě zajímavější.

Při modelování různých problémů závisí na měřítku, které uvažujeme.

- Pokud budu řešit reálný problém (velikost např. v cm), bude zcela vyhovující použít *spojitý model*.
- Pokud půjdu na úroveň nižší (na úroveň atomů), bude lepší použít *diskrétní model*.

  Poznámka:

  Musíme být opatrní, neboť podle *kvantové fyziky* je poloha částice rozprostřena v celém objemu vlnové funkce. Klasická představa, že polohu a rychlost lze určit s libovolnou přesností je nahrazena pravděpodobností, že tyto veličiny v jistém objemu nabývají určité hodnoty.

Odpověď na úvodní otázku nedám. Pokud se budu rozhodovat o tom, zda použiji diskrétní nebo spojitý model, musím přihlédnout k měřítku. V každém případě potom pracuji s *modelem*.

Co to vlastně znamená, když řeknu, že je něco spojité? Když tu věc rozpůlím, jsou obě části stále spojité a nemění své vlastnosti. To mohu opakovat do **nekonečna** (tj. jedná se o abstrakci).

Příklady:

Spojitost funkce v bodě	...	limita
Součet nekonečné řady	...	limita
Derivace	...	limita
Integrál určitý	...	limita

Problém chápání nekonečna a limity:

Zenon z Eleje (řecký matematik 400 l. př. n. l.)

- známý svou snahou zpochybnit učení pythagorovců, tj. zničit pojem čísla a pojem mnohosti
- chtěl dokázat neexistenci pohybu (mechanického)
- známé logické paradoxy, které vyvrátili až Pascal (1623 - 1662) a Leibniz (1646 - 1716)

(Špatné chápání nekonečného dělení a limity)

1. Letící šíp

   Pozorován v kterémkoli jednotlivém okamžiku letu, se nalézá *na určitém* místě v prostoru, na němž je v tom okamžiku v klidu. Když je ale v klidu de facto v každém libovolném okamžiku svého letu, pak je v klidu i v čase. To znamená, že letící šíp se nepohybuje.

2. Achilles a želva

   Závodí, a protože je Achilles, řekněme 10x rychlejší, dá želvě náskok, řekněme 100m. Achilles želvu nikdy nedohoní, protože když doběhne na místo, odkud želva startovala, ujde želva určitý kus cesty, když Achilles doběhne na toto nové místo, želva zase mezitím ujde další kus cesty a tak ji Achilles nemůže dohonit, i když se vzdálenost mezi nimi zmenšuje, nikdy se nemůže proměnit v nic.

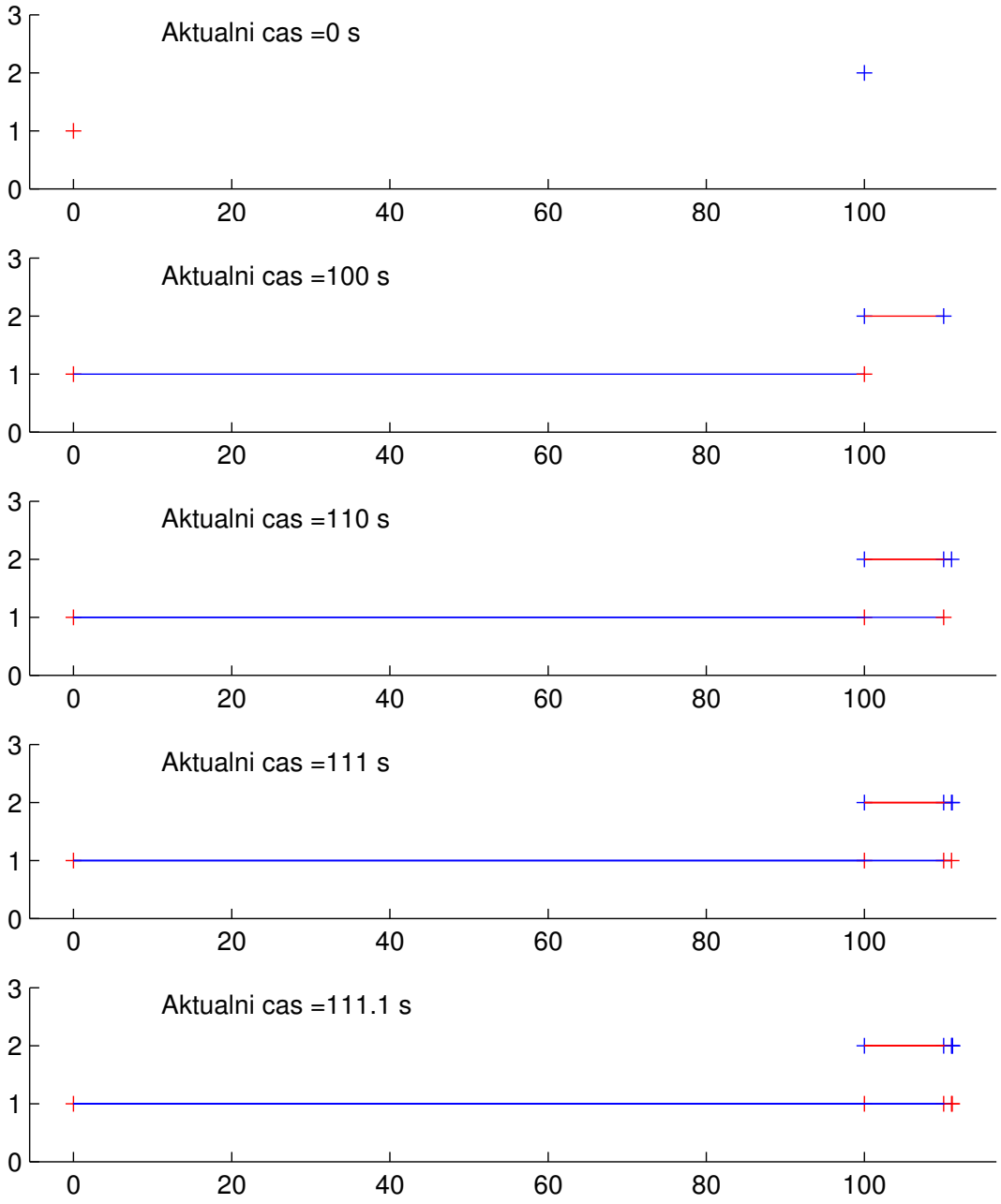
3. Dichotomie (půlení)

   Pokud chci dojít z místa A do místa B, musím nejprve dojít do středu úsečky AB (= bod C), ovšem do místa C se mohu dostat jen, když se dostanu do středu úsečky AC ... Důsledkem je, že nejen že nedosáhnu bodu B, ale při nekonečném dělení ani nevyrazím z místa A.

Kde byl problém?

ad 1. Pro definování pohybu potřebujeme čas, pokud odstraníme čas a mluvíme o okamžicích, odstraňujeme také pohyb. Ačkoliv se šíp nemusí hýbat v daném okamžiku, může se hýbat tehdy, pokud definujeme pohyb jako výskyt věci na jiném místě v pozdějším čase.

ad 2. Vzdálenost složená z nekonečného počtu konečných částí nemusí být nekonečná (nekonečná řada konverguje). Dosažení této vzdálenosti nastane v konečném čase.



krok	časový krok	čas	délkový krok	délka
0	$\triangle t_0 = 0\text{s}$	$t_0 = 0\text{s}$	$\triangle l_0 = 0$	$l_0 = 0$
1	$\triangle t_1 = 10\text{s}$	$t_1 = t_0 + \triangle t_1 = 10\text{s}$	$\triangle l_1 = 100\text{m}$	$l_1 = l_0 + \triangle l_1 = 100\text{m}$
2	$\triangle t_2 = 1\text{s}$	$t_2 = t_1 + \triangle t_2 = 11\text{s}$	$\triangle l_2 = 10\text{m}$	$l_2 = l_1 + \triangle l_2 = 110\text{m}$
3	$\triangle t_3 = 0{,}1\text{s}$	$t_3 = t_2 + \triangle t_3 = 11{,}1\text{s}$	$\triangle l_3 = 1\text{m}$	$l_3 = l_2 + \triangle l_3 = 111\text{m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i > 0$	$\triangle t_i = 10^{2-i}$		$\triangle l_i = 10^{3-i}$	

Celková vzdálenost:

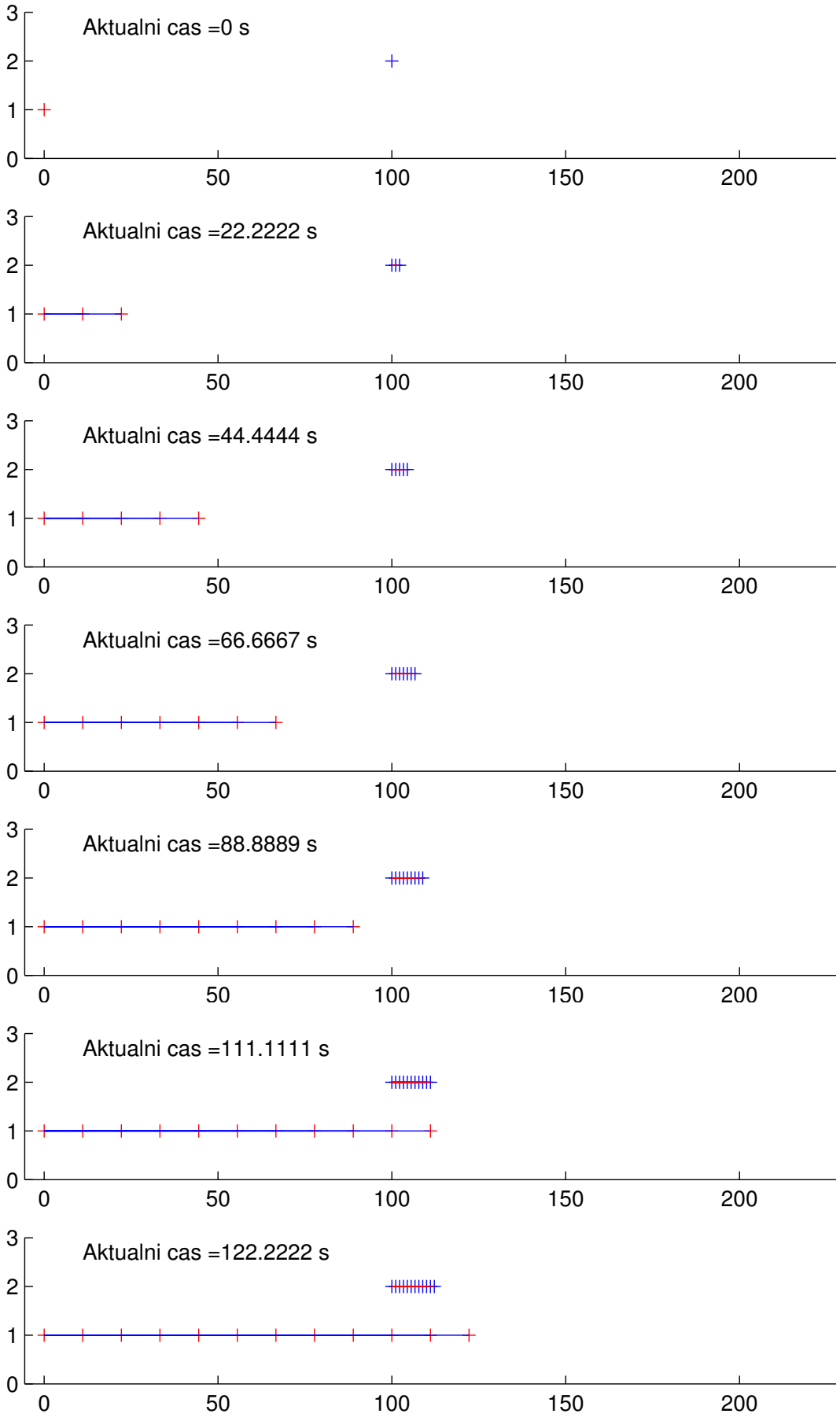
$$l = \sum_{i=1}^{\infty} \triangle l_i = \sum_{i=1}^{\infty} 10^{3-i} = 1000 \cdot \sum_{i=1}^{\infty} \left(\frac{1}{10}\right)^i = 1000 \cdot \frac{\frac{1}{10}}{1 - \frac{1}{10}} = \frac{1000}{9} = 111,\bar{1}\ \text{m}$$

Celkový čas:

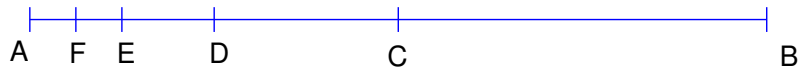
$$t = \sum_{i=1}^{\infty} \triangle t_i = \sum_{i=1}^{\infty} 10^{2-i} = 100 \cdot \sum_{i=1}^{\infty} \left(\frac{1}{10}\right)^i = 100 \cdot \frac{\frac{1}{10}}{1 - \frac{1}{10}} = \frac{100}{9} = 11,\bar{1}\ \text{s}$$

$$\Rightarrow\ \frac{l}{t} = 10\ \text{m/s} \ \ldots \text{ rychlost Achilla}$$

```
                Dohoni Achilles zelvu ?
-------------------------------------------------------
 Achilles je  10.000000  krat rychlejsi nez zelva.
 Zelva ma naskok  100.000000 metru.
 Rychlost zelvy je  0.100000 m/s.

 Achilles dohoni zelvu v case T = 111.111111.
```


Aktualni cas =0 s
Aktualni cas =22.2222 s
Aktualni cas =44.4444 s
Aktualni cas =66.6667 s
Aktualni cas =88.8889 s
Aktualni cas =111.1111 s
Aktualni cas =122.2222 s
0
1
2
3
0
50
100
150
200

ad 3. Podobná myšlenka jako v 2 (součet nekonečně mnoha konečných vzdáleností nebo časových kroků může být konečný.) Řekněme, že mám dojít z místa A do B za čas x (nebo také, že vzdálenost míst A a B je x)



Má platit: $$x = x\underbrace{(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots)}$$

$$\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = 1$$

Připomeňme si pojem spojitost funkce z *Matematické analýzy*

- funkce f je spojitá na (a, b), je-li spojitá v každém vnitřním bodě $x_0 \in (a, b)$
- funkce f je spojitá v bodě $x_0 \in D(f)$, když existuje $\lim\limits_{x \to x_0} f(x)$ a platí

$$\lim_{x \to x_0} f(x) = f(x_0)$$

- definice limity funkce f v bodě x_0 (Heine, Cauchy, topologická)

  Máme funkci $f : D(f) \to \mathbb{R}$ a $x_0 \in D(f)$ je hromadný bod. Když existuje $b \in \mathbb{R}$ tak, že $\forall$ posloupnosti $\{x_n\} \subset D(f)$, $x_n \neq x_0$, konvergující k číslu x_0, posloupnost $\{f(x_n)\}$ konverguje k číslu b, říkáme je funkce f má v bodě x_0 limitu b ($\lim\limits_{x \to x_0} f(x) = b$)

- definice limity posloupnosti

  Posloupnost $\{a_n\}_{n=1}^{\infty}$, je konvergentní v $\mathbb{R}$ platí-li:

$$\exists a \in \mathbb{R} \quad \forall \varepsilon > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n \in \mathbb{N} : n > n_0 \Rightarrow |a_n - a| < \varepsilon$$

$$(\lim_{n \to \infty} a_n = a \quad \text{nebo} \quad a_n \to a)$$

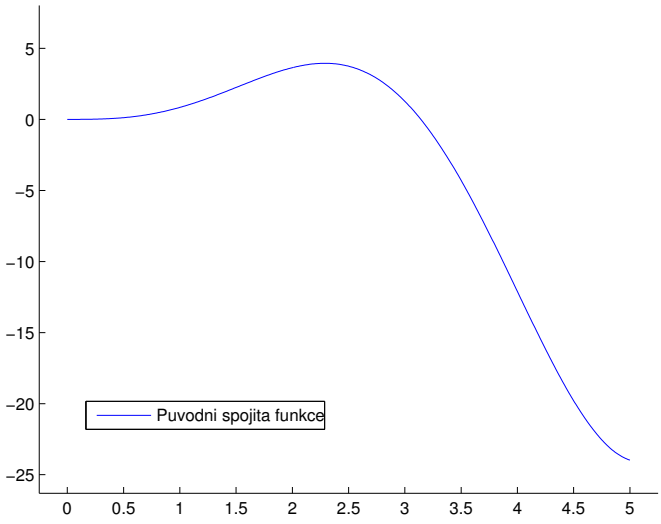
Poznámka: Ve všech definicích se objevuje pojem **nekonečna**.

- nekonečně mnoho vnitřních bodů x_0 z intervalu (a, b)
- $\forall$ posloupnosti $\{x_n\}$ ... těch je nekonečně mnoho
- posloupnost má nekonečně mnoho členů

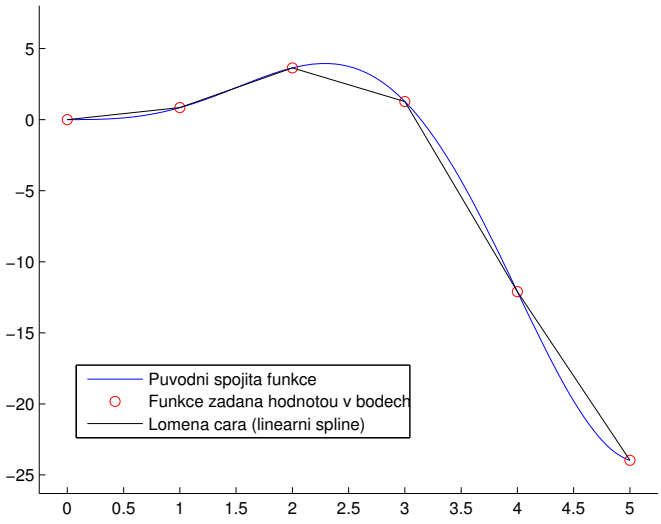
Otázka (filosofická): Skládá se vesmír z nekonečného počtu částic?

Příklad: Vykreslete například pomocí MATLABu graf funkce f na intervalu $< a, b >$.

Např. $f(x) = x^2 \sin x$ na $\langle 0, 5 \rangle$



Postupujeme tak, že zvolíme konečný počet bodů z intervalu a zakreslíme jejich funkční hodnoty. Ačkoliv víme, že funkce je spojitá a můžeme v každém bodě intervalu vykreslit funkční hodnotu, nemůžeme to udělat ve všech bodech (bodů je nekonečně mnoho). Mezery mezi body vyplníme čarou - nejjednodušší je použít úsečku.



Grafem je pro nás potom lomená čára spojující funkční hodnoty ve zvolených bodech.

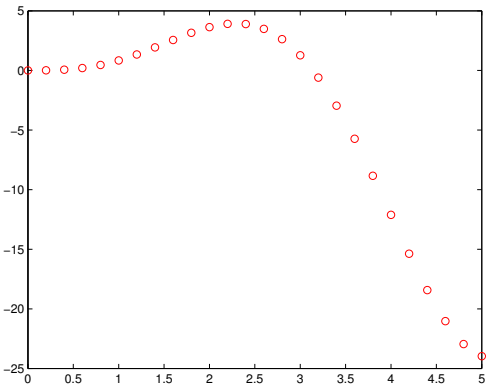
Pozn.: Pokud budu chtít vykreslit podrobněji na menším intervalu, musím na tomto intervalu zvolit více bodů.

Přesný graf je potom vlastně limitním případech, pro krok dělení (max. rozdíl 2 sousedních uzlů) jdoucí k nule.

Protože nelze graf do nekonečna zpřesňovat, jsme nuceni použít aproximaci funkce zadané tabulkou.

Zadání: Funkci f máme danou tabulkou bodů

x_i	.....
$f(x_i)$	.....



Naším cílem je pospojovat body pomocí nějakého jednoduchého pravidla. Toto pravidlo musíme specifikovat. Podle toho, jaké pravidlo použiji, dostanu různý přístup k aproximaci funkce.

1. způsob:

Nejvíce používanými funkcemi jsou asi polynomy. Řekněme, že máme tabulku s $(n+1)$ body pro navzájem různé argumenty. Těmito $(n+1)$ body pochází právě 1 polynom nejvýše $n-$tého stupně.

Interpolační úloha

Body prokládáme polynomem nejnižšího možného stupně tak, aby graf těmito body přesně procházel.

Možností jak určit interpolační polynom existuje více. Nejjednodušší je asi si napsat předpis pro polynom a požadovat splnění interpolačních podmínek (potom je třeba řešit soustavu lineárních algebraických rovnic - SLAR).

Např. pro 6 zadaných bodů $[x_0, f(x_0)], [x_1, f(x_1)], [x_2, f(x_2)], [x_3, f(x_3)], [x_4, f(x_4)], [x_5, f(x_5)]$ hledáme takový polynom $P_5(x)$ stupně nejvýše 5 tak, aby jeho graf procházel zadanými body.

$$P_5(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$$

Interpolační podmínky:

$$P_5(x_0) = ax_0^5 + bx_0^4 + cx_0^3 + dx_0^2 + ex_0 + f = f(x_0)$$

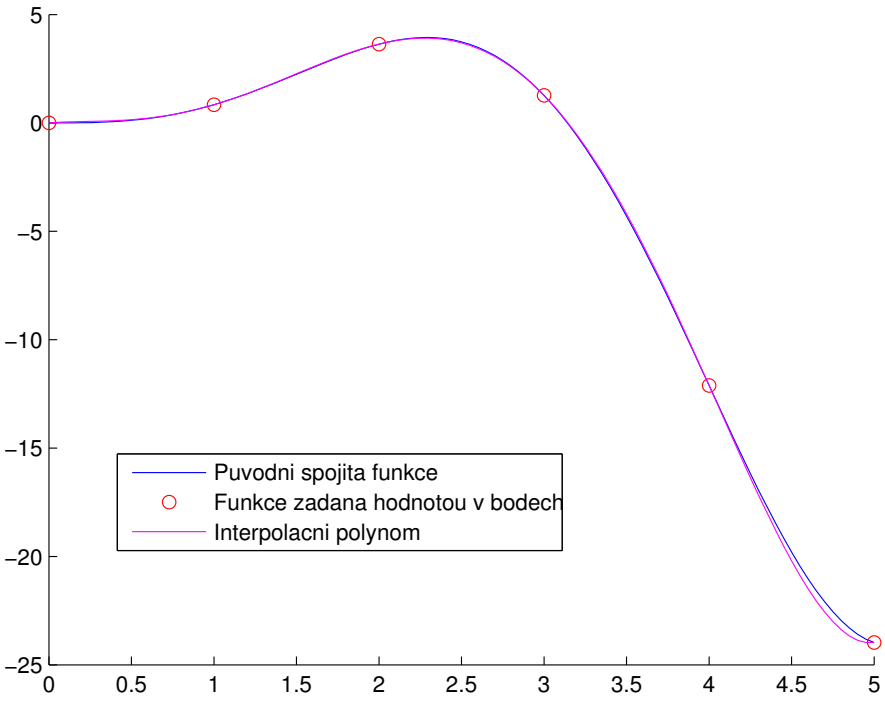
$$P_5(x_1) = ax_1^5 + bx_1^4 + cx_1^3 + dx_1^2 + ex_1 + f = f(x_1)$$

$$\vdots$$

$$P_5(x_5) = ax_5^5 + bx_5^4 + cx_5^3 + dx_5^2 + ex_5 + f = f(x_5)$$

Podmínky můžeme přehledně zapsat pomocí matice a vektorů:

$$\begin{bmatrix} x_0^5 & x_0^4 & x_0^3 & x_0^2 & x_0^1 & x_0^0 \\ x_1^5 & x_1^4 & x_1^3 & x_1^2 & x_1^1 & x_1^0 \\ & & \vdots & & & \\ x_5^5 & x_5^4 & x_5^3 & x_5^2 & x_5^1 & x_5^0 \end{bmatrix} \begin{bmatrix} a \\ b \\ \vdots \\ f \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_5) \end{bmatrix}$$



2. způsob:

Obdoba interpolační úlohy, ovšem nehledáme 1 polynom na celém intervalu $< a, b >$, ale výslednou funkci složíme z polynomů na dílčích intervalech $< x_i, x_{i+1} >$ a požadujeme splnění dalších podmínek, (např. spojitosti v uzlech) ... aproximace **spline funkcí**.

Lineární spline funkce ... lomená čára spojující funkční hodnoty v zadaných bodech.

Kubický spline ... funkce, která je na dílčím intervalu kubická (polynom 3 stupně), splňuje interpolační podmínky, podmínky spojitosti, spojitosti 1. derivace a spojitosti 2. derivace + další 2 podmínky.

Na vstupu: $n+1$ podmínek (bodů) $\Rightarrow n$ intervalů. Polynom 3 stupně je dán 4 koeficienty.

Na výstupu: Potřebujeme $4n$ podmínek, které určí kubický spline.

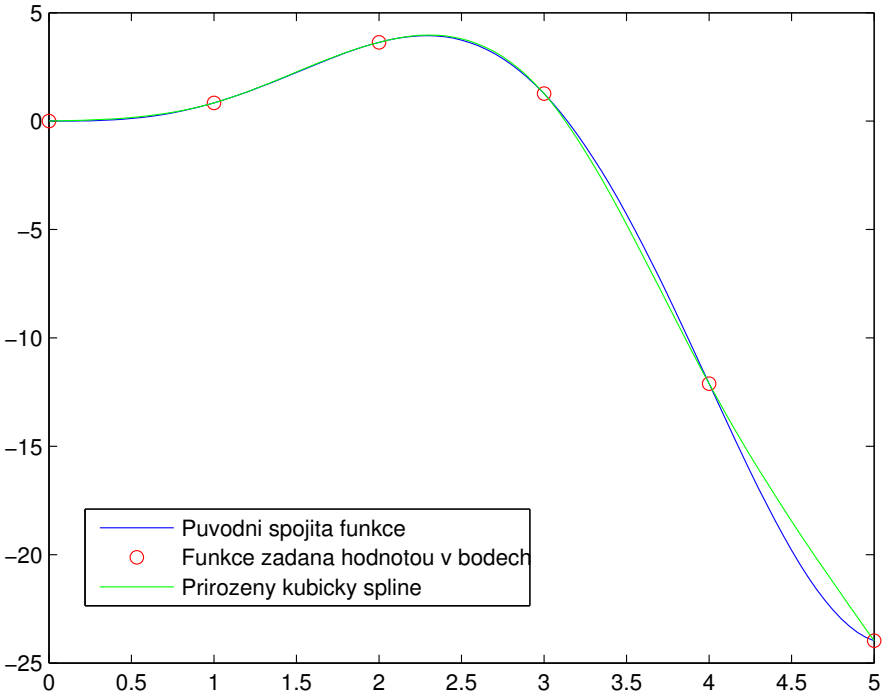
Interpolační podmínky	...	$n+1$
Podmínky spojitosti v zadaných bodech	...	$n-1$
Podmínky spojitosti 1. derivace	...	$n-1$
Podmínky spojitosti 2. derivace	...	$n-1$
dohromady máme	...	$4n$ - 2

Zbylé 2 podmínky můžeme doplnit například takto:

$f'(a) = \ldots; f'(b) = \ldots$

$f''(a) = 0; f''(b) = 0$

$f'(a) = f'(b); f''(a) = f''(b)$



Poznámka: V 1. i 2. způsobu předepisujeme splnění interpolačních podmínek. Ve 3. způsobu nebudeme vynucovat splnění interpolačních podmínek. Vždyť jsme se při vyčíslování hodnot $f(x_i)$ mohli dopustit chyby.

3. způsob:

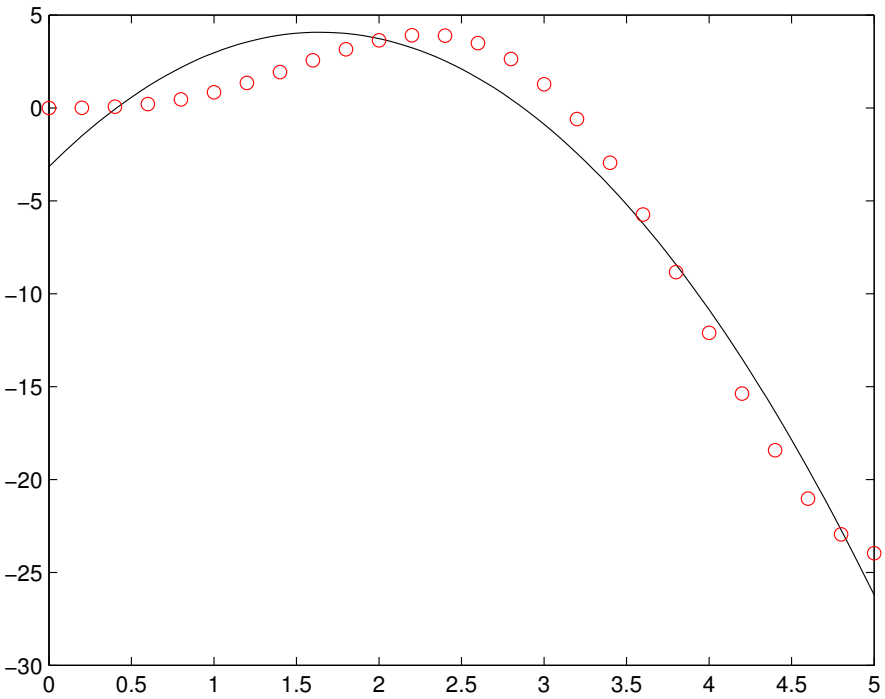
Hledáme funkci φ (aproximaci funkce f) a nepožadujeme aby procházela přesně zadanými body. Musíme odpovědět na 2 otázky:

a) Odkud funkci φ vybíráme?

b) Podle jakého kritéria ji určujeme?

ad a) Můžeme zůstat u polynomů a hledat φ mezi např. polynomy nejvýše 2 stupně.

ad b) Samozřejmě hledáme nějakou nejlepší aproximaci, naším cílem je minimalizovat chybu, otázka zůstává jak tu chybu měříme.



Co by šlo měřit ?

→ Vzdálenost zadaných bodů od výsledného grafu φ

φ by musela být hladká, abychom byli schopni v každém bodě určit normálu.

- dost složité !

→ Rozdíl funkčních hodnot (v absolutní hodnotě).

Chybová funkce r by potom byla dána:

$$r(f,\varphi) = \sum_{i=0}^{n} |f(x_i) - \varphi(x_i)|.$$

Naším cílem je tuto funkci minimalizovat a proto se nám nehodí, že r není hladká. Protože, kdyby byla hladká, věděli bychom že v bodě minima je nulová derivace a snadno bychom minimum našli. Pro nehladkou funkci je hledání minima složitější.

→ Pokud budeme měřit kvadrát rozdílu funkčních hodnot, bude chybová funkce hladká a minimum můžeme snadno určit z podmínky nulové derivace.

$$r(f,\varphi) = \sum_{i=0}^{n} (f(x_i) - \varphi(x_i))^2$$

Mluvíme o **metodě nejmenších čtverců** (**diskrétní** L_2 **aproximaci**).

Čím je tedy určena funkce φ?

Pokud hledám φ např. mezi polynomy stupně nejvýše 2, pak hledám 3 koeficienty a, b, c v předpisu:

$$\varphi(x) = ax^2 + bx + c.$$

Poznámka: Pro aproximaci můžeme použít i jiné funkce, např.: $\varphi(x) = ae^x + bx + c$.

Nejlepší aproximaci vybíráme z prostoru těchto funkcí. Výsledná aproximace φ je potom jednoznačně určena koeficienty lineární kombinace.

→ Obecně tedy nejprve zvolíme systém bázových funkcí $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$.

Hledáme koeficienty c_j, $j \in \{1, 2, \ldots, m\}$ tak, aby:

$$\varphi(x) = c_1 \cdot \varphi_1(x) + c_2 \cdot \varphi_2(x) + \ldots c_m \cdot \varphi_m(x)$$

byla nejlepší aproximací.

→ Chybová funkce potom vypadá takto:

$$r(c_j) = \sum_{i=0}^{n} \left(f(x_i) - \sum_{j=1}^{m} c_j \varphi_j(x_i) \right)^2 \quad \ldots \text{ funkce parametrů } c_j$$

→ Počet parametrů c_j, tzn. počet zvolených bázových funkcí $\varphi_j(x)$, musí být nejvýše roven počtu zadaných bodů. Pokud bychom hledali φ jako lineární kombinaci více než $n+1$ bázových funkcí, řešení by bylo nejednoznačné. Pokud je počet zadaných bodů stejný jako počet zvolených bázových funkcí (a bázové funkce jsou zvoleny rozumně), pak je L_2 aproximace interpolací a výsledný graf φ prochází zadanými body.

Co tedy dostaneme jako nutné podmínky minima chybové funkce?

$$\frac{\partial r}{\partial c_k} = -2\sum_{i=0}^{n}\left(f(x_i) - \sum_{j=1}^{m} c_j\varphi_j(x_i)\right)\cdot\varphi_k(x_i) = 0 \quad \text{pro} \quad k = 1, 2, \ldots, m.$$

$$\sum_{i=0}^{n} f(x_i)\cdot\varphi_k(x_i) = \sum_{i=0}^{n}\sum_{j=1}^{m}\varphi_j(x_i)\,\varphi_k(x_i)\,c_j$$

$$\sum_{j=1}^{m}\left(\sum_{i=0}^{n}\varphi_j(x_i)\cdot\varphi_k(x_i)\right)\,c_j = \sum_{i=0}^{n} f(x_i)\cdot\varphi_k(x_i)$$

Těchto m ($k = 1, 2, \ldots, m$) rovnic můžeme zapsat pomocí matice a vektorů:

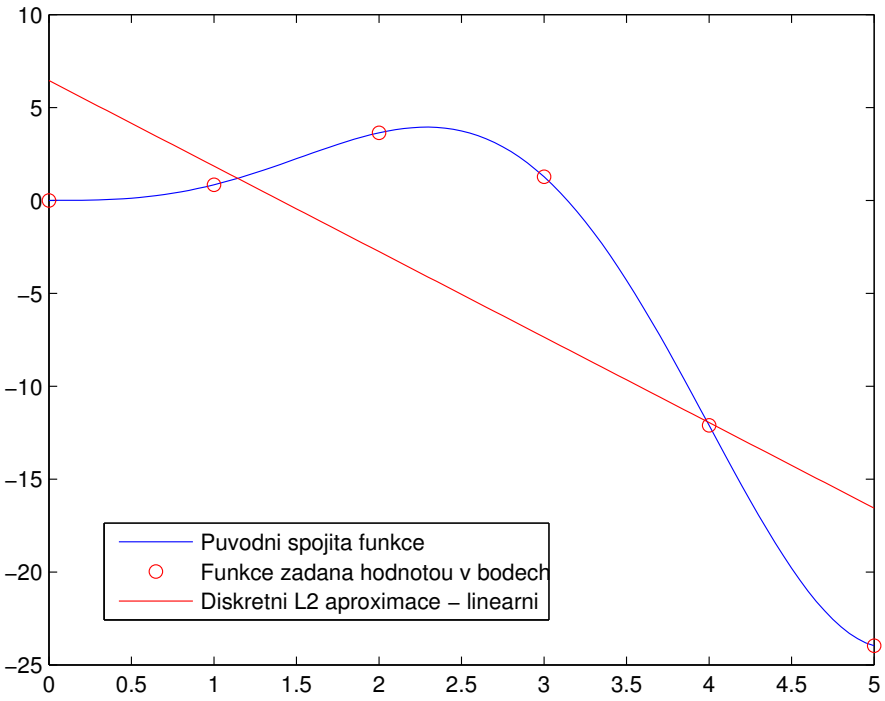
$$\begin{bmatrix} & \vdots & \\ \cdots\cdots & \sum_{i=0}^{n}\varphi_j(x_i)\cdot\varphi_k(x_i) & \cdots\cdots \\ & \vdots & \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \vdots \\ \sum_{i=0}^{n} f(x_i)\cdot\varphi_k(x_i) \\ \vdots \end{bmatrix}$$

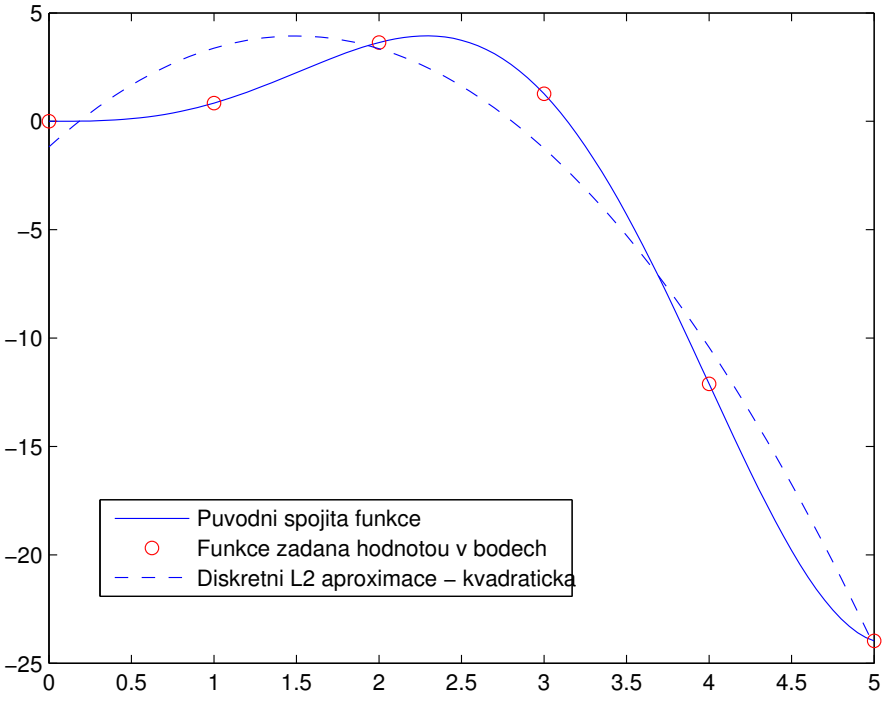
Řešíme tedy soustavu lineárních algebraických rovnic, kde

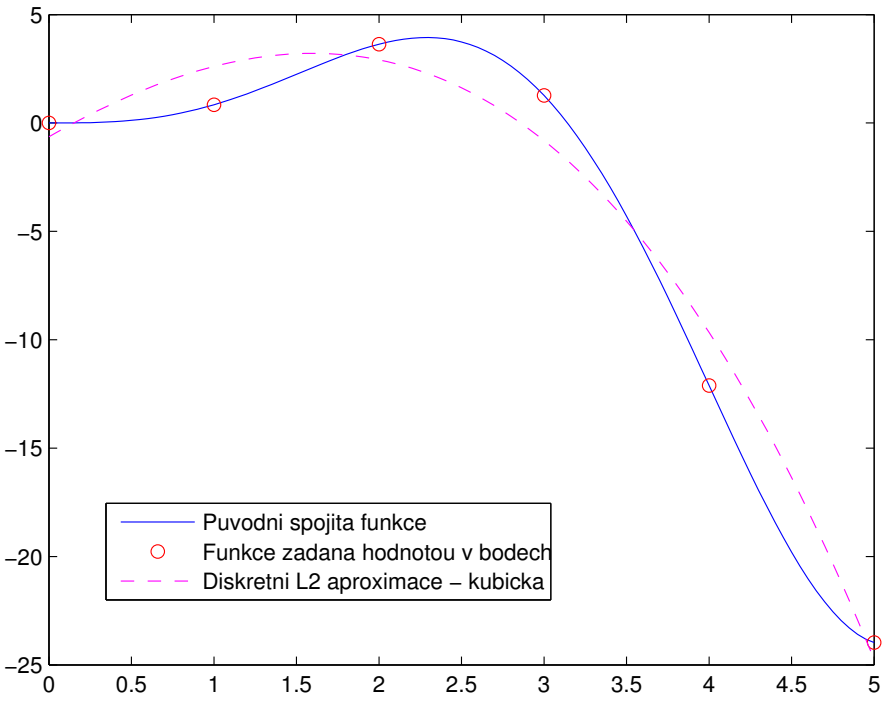
matice soustavy má v pozici $[j,k]$ prvek $\sum_{i=0}^{n}\varphi_j(x_i)\cdot\varphi_k(x_i)$,

na pravé straně je sloupcový vektor s k-tou složkou $\sum_{i=0}^{n} f(x_i)\cdot\varphi_k(x_i)$ a

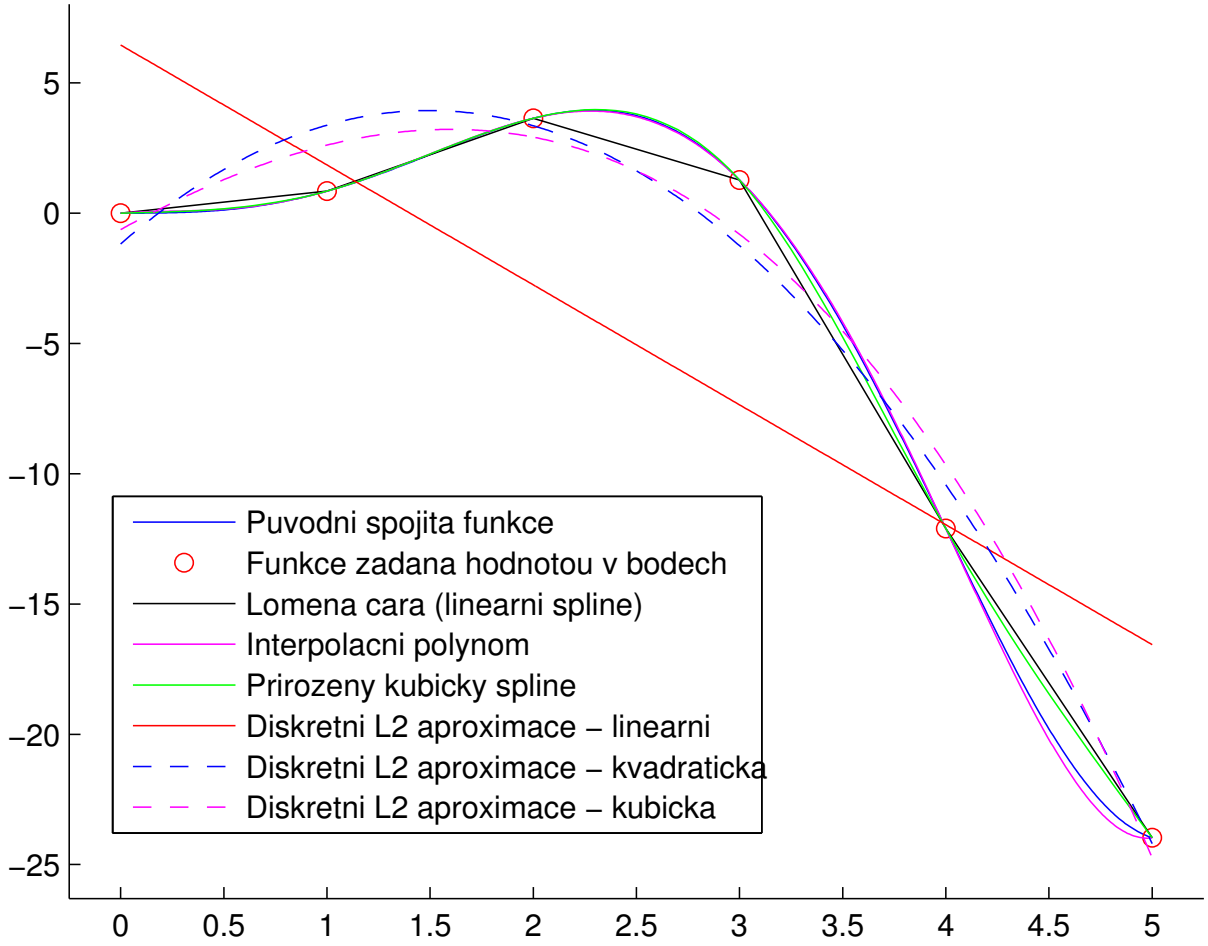
hledáme sloupcový vektor neznámých koeficientů $[c_1, c_2, \ldots, c_m]^T$.


10
5
0
−5
−10
−15
−20
−25
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Puvodni spojita funkce
Funkce zadana hodnotou v bodech
Diskretni L2 aproximace – linearni


5
0
−5
−10
−15
−20
−25
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Puvodni spojita funkce
Funkce zadana hodnotou v bodech
Diskretni L2 aproximace – kvadraticka


5
0
−5
−10
−15
−20
−25
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Puvodni spojita funkce
Funkce zadana hodnotou v bodech
Diskretni L2 aproximace – kubicka

Abychom mohli porovnat všechny použité aproximace, vykreslíme je do jednoho obrázku.



Spojitá x diskrétní matematika

Spojitý případ

Podívejme se na pojem **derivace funkce v bodě**. Derivace funkce vyjadřuje rychlost změny (růstu) této funkce vzhledem k nezávisle proměnné. V historii nejjednodušší představa o derivaci je: "derivace je mírou změny funkce v daném bodě". Pro změnu hodnoty používáme symbol $\triangle$.

Poměr lze potom symbolicky zapsat takto: $\dfrac{\triangle f}{\triangle x}$.

Derivace je hodnota tohoto podílu pro $\triangle x \to 0$ ($\triangle x$ ... konečné číslo).

Pokud $\triangle x$ nahradíme nekonečně malou změnou $\mathrm{d}x$, získáme definici:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \quad \text{(říkáme } \mathrm{d}f \text{ podle } \mathrm{d}x\text{, podíl 2 diferenciálů).}$$

Nejběžnější definice:

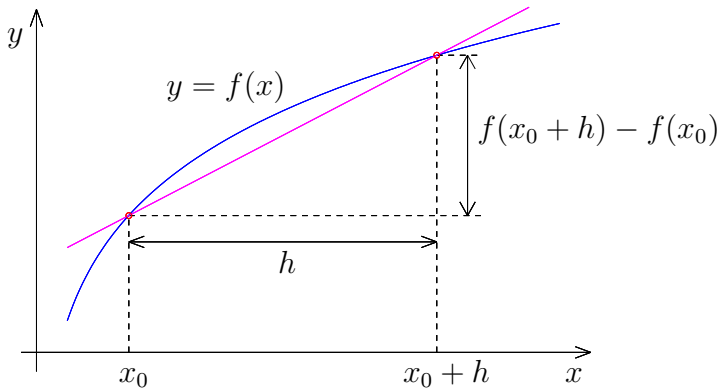
Řekneme, že funkce definovaná na okolí $U(x_0)$ má v bodě x_0 derivaci, existuje-li konečná limita:

$$\lim_{x\to x_0} \frac{f(x)-f(x_0)}{x-x_0} = \lim_{h\to 0} \frac{f(x_0+h)-f(x_0)}{h} = f'(x_0) = f'(x)|_{x=x_0}.$$

Této limitě říkáme **derivace funkce f v bodě x_0**.

Geometrický význam

... směrnice tečny ke grafu funkce f v bodě x_0



Způsoby výpočtu derivace:

1. Analyticky - Podle definice a nebo pomocí známých pravidel pro derivování s využitím tabulky s derivacemi "základních" funkcí.
2. Numericky - Aproximujeme pomocí diferenčního podílu pro zvolené malé $h > 0$.

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h} \qquad (\bullet)$$

Výhody, nevýhody:

ad 1. + dostaneme přesné vyjádření derivace

– pro obecné funkce může být velmi pracné

ad 2. + snadný výpočet

– jde o aproximaci, tj. nesmíme zapomenout na chybu (chybu aproximace)

Pokud budeme chtít určit chybu aproximace, je vhodné použít Taylorovu formuli. V té se vyskytují vyšší derivace. Abychom mohli definovat 2. derivaci v bodě, musíme definovat **derivaci funkce na intervalu** (a,b):

- Má-li funkce f derivaci v každém bodě $x \in (a,b)$, potom funkci $g : (a,b) \to \mathbb{R}$, která každému $x \in (a,b)$ přiřazuje číslo $\lim\limits_{h\to 0} \dfrac{f(x+h)-f(x)}{h} = f'(x)$, nazýváme derivace a opět značíme $f'(x)$.
- Nechť je funkce f diferencovatelná v bodě x_0 a na jeho okolí. Je-li navíc v bodě x_0 diferencovatelná funkce f', tj. existuje limita:

$$\lim_{h\to 0} \frac{f'(x_0+h) - f'(x_0)}{h} = f''(x_0),$$

pak se tato limita nazývá **druhá derivace funkce f v bodě x_0**.

Poznámka: Když existuje $f''(x_0)$, potom platí:

$$f''(x_0) = \lim_{h\to 0} \frac{f(x_0+2h) - 2f(x_0+h) + f(x_0)}{h^2}.$$

Důkaz:

$$f''(x_0) = \lim_{h\to 0} \frac{f'(x_0+h) - f'(x_0)}{h} = \lim_{h\to 0} \frac{\lim\limits_{h\to 0} \frac{f(x_0+2h)-f(x_0+h)}{h} - \lim\limits_{h\to 0} \frac{f(x_0+h)-f(x_0)}{h}}{h} =$$

$$= \lim_{h\to 0} \frac{f(x_0+2h) - 2f(x_0+h) + f(x_0)}{h^2}.$$

Věta (Taylorova): Nechť je funkce f diferencovatelná do 2 řádu v bodě x_0 a jeho okolí $U(x_0)$. Potom $\forall x \in U(x_0)\ \exists \xi$ mezi body x a x_0 tak, že platí:

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x - x_0)}_{\text{Taylorův polynom 1. stupně v bodě } x_0} + \underbrace{\frac{f''(\xi)}{2}(x - x_0)^2}_{\text{chyba aproximace}} \qquad (\star)$$

Důkaz: viz matematická analýza (věta o střední hodnotě).

Vztah $(\star)$ můžeme (pro $x = x_0 + h$) přepsat takto:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(\xi)}{2}h^2.$$

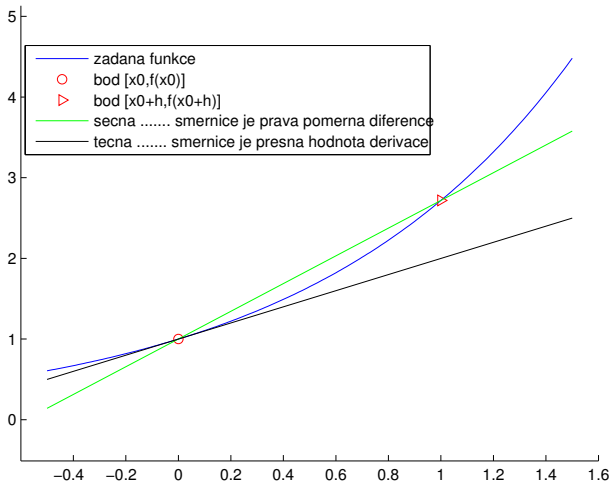
Odtud jsme schopni vyjádřit chybu našeho vzorce $(\bullet)$

$$f'(x_0)h = f(x_0 + h) - f(x_0) - \frac{f''(\xi)}{2}h^2,$$

$$f'(x_0) = \underbrace{\frac{f(x_0 + h) - f(x_0)}{h}}_{\text{náš vzorec } Df(x_0,h)} \quad \underbrace{- \frac{f''(\xi)}{2}h}_{\text{chyba vzorce}}.$$

Příklad: Použijte vzorec $Df(x_0, h)$ pro výpočet derivace funkce $f(x) = \mathrm{e}^x$ v bodě $x_0 = 0$ s krokem $h = 1$ a určete chybu aproximace.

```
 Geometricky vyznam pojmu derivace funkce v bode
-------------------------------------------------
 Derivace f(x)=exp(x) v bode x0=0.000000  (krok h=1)

 Priblizna hodnota derivace
 pomoci prave pomerne diference   D_P(f,x0) = 1.718282

 Presna hodnota derivace je         f'(x_0) = 1.000000

 Chyba aproximace je    D_P(f,x0) - f'(x_0) = 0.718282
```



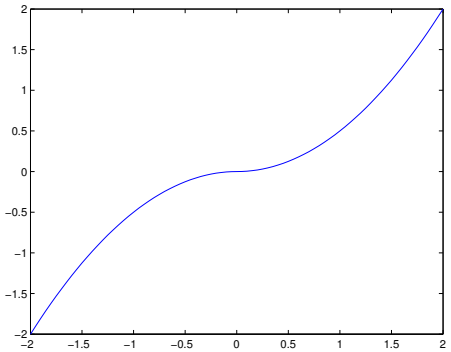
Pokud chceme odhadnout chybu vzorce pro výpočet $f'(x_0)$ se zvoleným krokem h, musíme umět odhadnout $f''(x_0)$ na intervalu $(x_0, x_0 + h)$ (přesnou hodnotu ξ neznáme).

Poznámka:

Pokud neexistuje 2. derivace funkce f na okolí x_0, pak nelze tento postup použít.

Příkladem funkce, která je diferencovatelná na okolí bodu $x_0 = 0$, ale neexistuje 2. derivace v x_0 je funkce

$$f(x) = \frac{x^2}{2} \operatorname{sign} x.$$



Jinak zapsáno

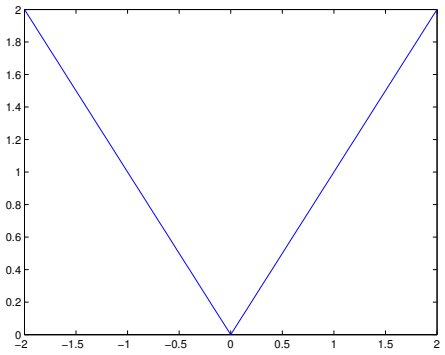
$$f(x) = \begin{cases} -\dfrac{x^2}{2}, & x < 0, \\ \dfrac{x^2}{2}, & x \geq 0. \end{cases}$$

Funkce f je spojitá na okolí bodu 0.

Pro derivaci f' platí:

$$f'(x) = \begin{cases} -x, & x < 0, \\ \\ x, & x \geq 0, \end{cases}$$

tj. $f'(x) = |x|$.



Platí:

$$f'(x)|_{x=x_0+} = f'(x)|_{x=x_0-} = 0 \quad \ldots \quad \text{limity zprava a zleva se rovnají}$$

$$\Rightarrow \quad \text{v bodě } x_0 = 0 \text{ existuje limita (tj. derivace)} \quad f'(x_0) = 0.$$

2. derivace f'' v bodě $x_0 = 0$ neexistuje, protože limita zprava a limita zleva nabývají různých hodnot:

$$f''(x)|_{x=x_0+} = 1, \qquad f''(x)|_{x=x_0-} = -1.$$

Podívejme se na <u>podmíněnost úlohy</u> hledat derivaci spojité funkce v bodě x_0 pomocí vzorce $Df(x_0, h)$.

$$f'(x_0) = \underbrace{\frac{f(x_0+h) - f(x_0)}{h}}_{Df(x_0,h)} \qquad \underbrace{- \quad f''(\xi)\,\frac{h}{2}}_{\text{chyba vzorce } \; r_1(h)=h\cdot c_1}$$

<u>Idea</u>:

Pokud budeme chtít získat přesnější výsledek, musíme zvolit menší krok h (chyba aproximace je potom menší). Pokud budeme ovšem zmenšovat krok h, budou potom při vyčíslování vzorce nastávat problémy, protože budeme dělit dvě nekonečně malá čísla (v limitě).

Uvažujme jak vypadají <u>zaokrouhlovací chyby</u>:

- místo přesné hodnoty $f(x_0)$ budeme mít k dispozici přibližnou hodnotu $\widehat{f}(x_0)$
- místo přesné hodnoty $f(x_0 + h)$ budeme mít k dispozici přibližnou hodnotu $\widehat{f}(x_0 + h)$

Dosazením do vzorce zjistíme, že jsme vypočetli:

$$\frac{\widehat{f}(x_0+h)-\widehat{f}(x_0)}{h} \quad \text{místo} \quad \frac{f(x_0+h)-f(x_0)}{h}.$$

Chyba která vznikla důsledkem nepřesného vyjádření $f(x_0)$ a $f(x_0+h)$ je potom rozdíl

$$r_2(h) = \frac{\widehat{f}(x_0+h)-\widehat{f}(x_0)}{h} - \frac{f(x_0+h)-f(x_0)}{h} =$$

$$= \frac{1}{h}\left[(\widehat{f}(x_0+h)-f(x_0+h)) + (f(x_0)-\widehat{f}(x_0))\right].$$

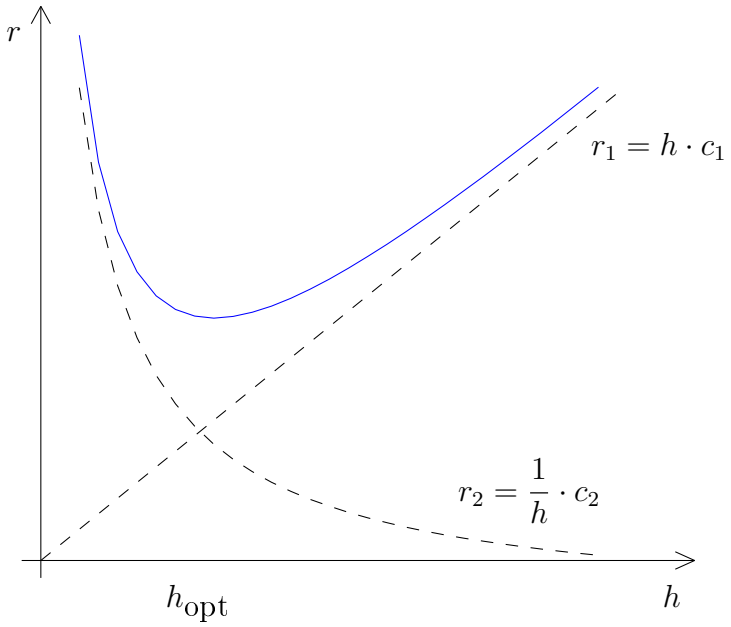
$$|r_2(h)| \le \frac{1}{h}\cdot\left[\underbrace{|\widehat{f}(x_0+h)-f(x_0+h)|}_{\sim \text{ strojová přesnost } \varepsilon} + \underbrace{|f(x_0)-\widehat{f}(x_0)|}_{\sim \text{ strojová přesnost } \varepsilon}\right]$$

$$|r_2(h)| \le \frac{1}{h}\cdot c_2 \qquad (c_2 = 2\varepsilon > 0).$$

Celková chyba je potom

$$r(h) = r_1(h) + r_2(h) = h\cdot c_1 + \frac{1}{h}\cdot c_2.$$

Pokud budeme chtít získat výsledek s menší chybou vzorce, musíme vzít $h \to 0$.

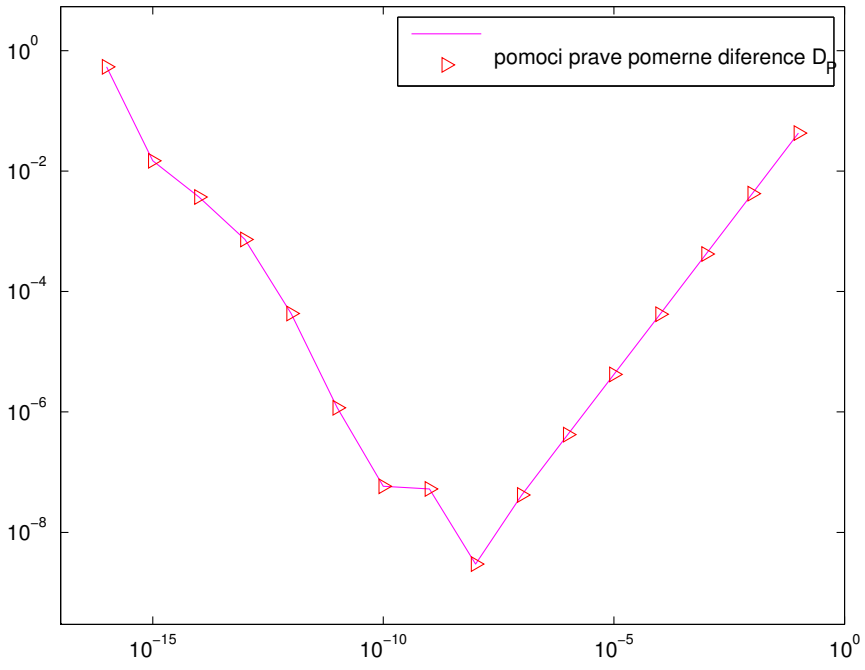


Chyba $r(h)$ pro $h \to 0+$ jde ale k $+\infty$!

Poznámka: Úloha hledat derivaci spojité funkce v bodě x_0 pomocí vzorce $Df(x_0, h)$ je **špatně podmíněná**, protože malá rel. chyba vstupních dat vyvolá velkou rel. chybu výstupních dat.

Příklad: Vykreslete závislost chyby při použití vzorce $Df(x_0, h)$ pro numerický výpočet derivace funkce $f(x) = \sin x$ v bodě $x_0 = 1$ pro kroky $h = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$. Výpočet proveďte v prostředí MATLAB.

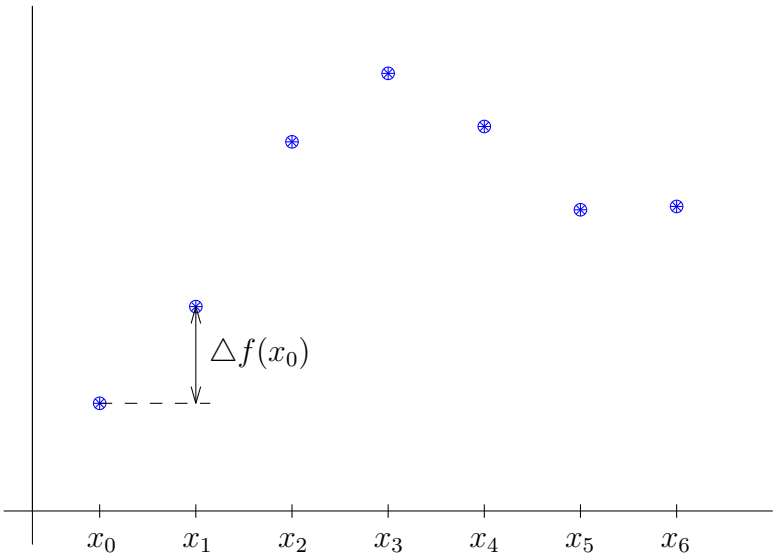
```
------------------------------------------------------------
 Vykresli chybu pri numerickem vypoctu derivace funkce
  f(x)=sin(x) v bode x0=1.000000 s kroky
  h=[1e-16,1e-15,1e-14,1e-13,1e-12,1e-11,1e-10,1e-09,
     1e-08,1e-07,1e-06,1e-05,0.0001,0.001,0.01,0.1]
------------------------------------------------------------

 chvilku strpeni (pocitam)
```



Diskrétní případ

Analogie pro funkci zadanou tabulkou:

Pro jednoduchost uvažujeme případ, kdy je dána funkce svými hodnotami v bodech: $x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, x_3 = x_0 + 3h, \ldots, x_n = x_0 + nh$ ($h \ldots$ krok dělení).



Pokud nás zajímá změna chování funkce při předmětu z bodu x_0 do x_1, pak je vhodné definovat **první diferenci** v bodě x_0:

$$\triangle f(x_0) = f(x_0 + h) - f(x_0).$$

Analogií 2.derivace je **druhá diference** funkce f v x_0:

$$\triangle^2 f(x_0) = \triangle(\triangle f(x_0)) = \triangle(f(x_0 + h) - f(x_0)) =$$

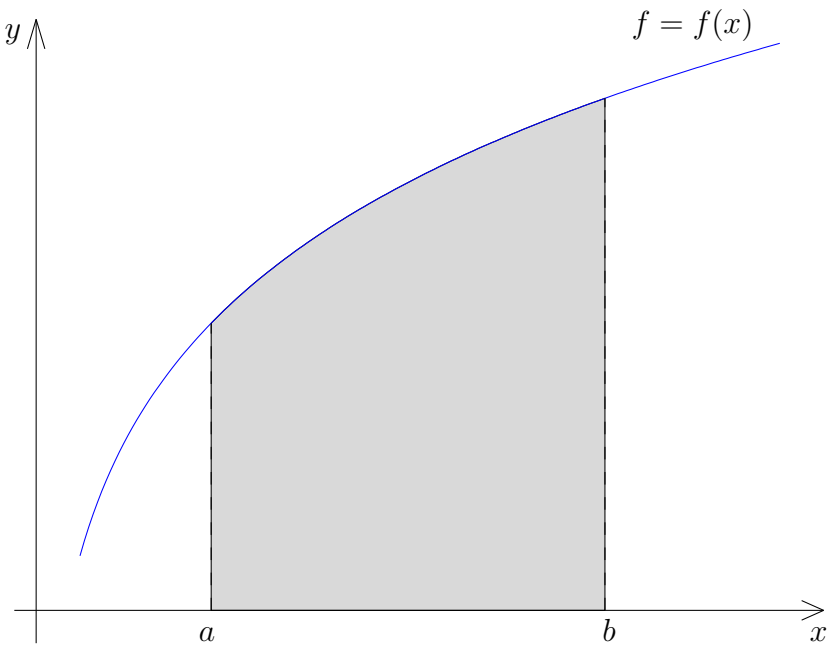
$$= (f(x_0 + 2h) - f(x_0 + h)) - (f(x_0 + h) - f(x_0)) = f(x_0 + 2h) - 2f(x_0 + h) + f(x_0).$$

Poznámka: U numerického výpočtu derivace narazíme na problém špatné podmíněnosti. To je dáno tím, že derivace je definována jako limina podílu, kde jmenovatel jde k 0! V diskrétním problému lze mluvit pouze o diferencích a ty jsou definovány jako rozdíl. Tudíž žádné problémy zde nevznikají.

Dosud jsme se věnovali pojmu derivace funkce v bodě pro spojitý případ a diference funkce pro diskrétní případ. Nyní se zaměříme na pojem určitého integrálu pro spojitý případ a sumace posloupnosti pro diskrétní případ.

Spojitý případ

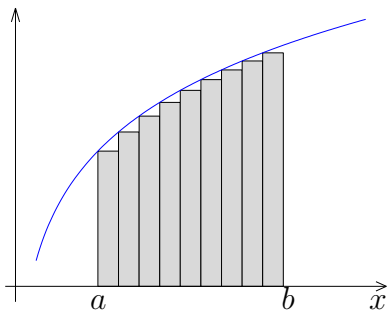
Podívejme se na pojem **určitého integrálu** z funkce f na intervalu $< a, b >$. Budeme mluvit o tzv. **Riemannově integrálu**. Definice vychází z intuitivní představy měření obsahu plochy pod grafem funkce.



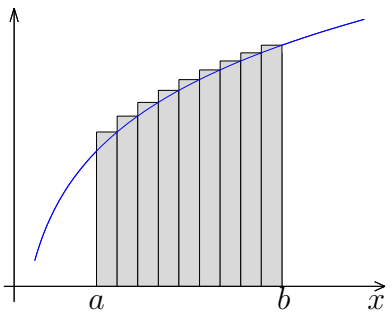
Při odvození opět postupujeme tak, že vycházíme z konečného počtu dílků, které neustále zmenšujeme a určitý integrál je pak limitním případem nějakého součtu.

Zavádíme dolní odhad a horní odhad, které získáme tak, že vyplňujeme vnitřek plochy a obrazec tak, aby obsahoval danou plochu.

dolní odhad



horní odhad



Pro vyplňování používáme obdélníky se stranami rovnoběžnými se souřadnými osami.

Definice:

- D ... dělení intervalu (a, b)
  $n-$tice $x_0, x_1, \ldots, x_n$ tak, že $a = x_0 < x_1 < \ldots < x_n = b$

- Horní součet pro f a D

$$S(f, D) = \sum_{i=1}^{n} \sup_{x \in (x_{i-1}, x_i)} f(x) \cdot (x_i - x_{i-1})$$

- Horní Riemannův integrál funkce f od a do b

$$\text{(HR)} \quad \overline{\int_a^b} f(x)\,\mathrm{d}x = \inf\{S(f, D), D \ldots \text{dělení } (a, b)\}$$

- Dolní součet pro f od D

$$s(f, D) = \sum_{i=1}^{n} \inf_{x \in (x_{i-1}, x_i)} f(x) \cdot (x_i - x_{i-1})$$

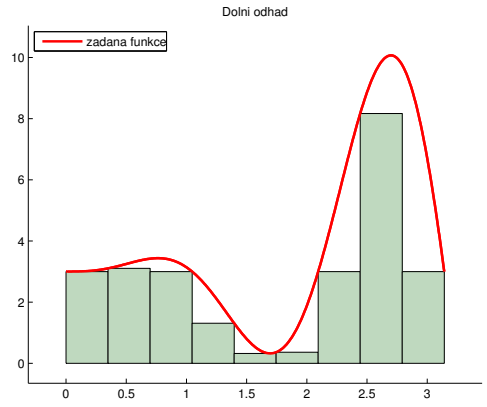
- Dolní Riemannův integrál funkce f od a do b

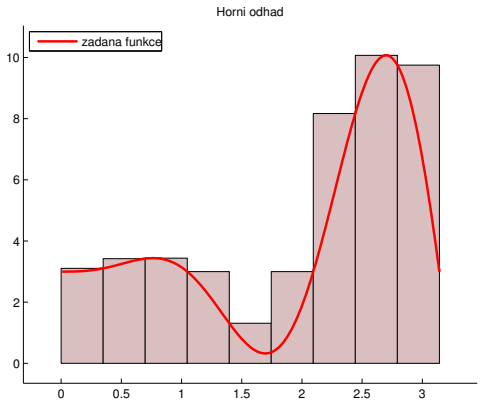
$$\text{(DR)} \quad \underline{\int_a^b} f(x)\,\mathrm{d}x = \sup\{s(f, D), D \ldots \text{dělení } (a, b)\}$$

Pokud se Dolní a Horní Riemannův integrál rovnají, nazýváme jej **Riemannovým integrálem** funkce f od a do b.

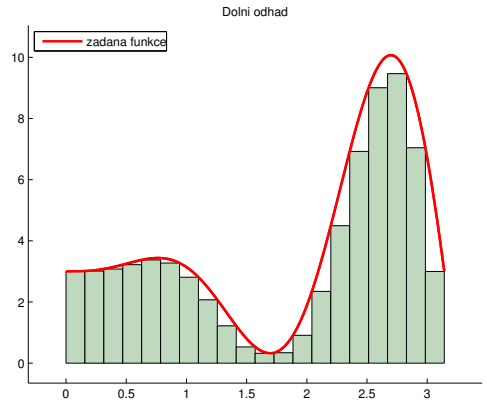
Příklad: Vypočtěte dolní a horní součty pro funkci $f(x) = x^2 \sin 3x + 3$ na intervalu $(0, \pi)$ pro počet rovnoměrného dělení intervalu 9, 20 a 50.

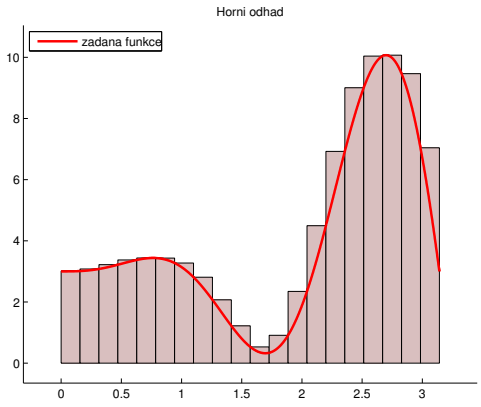
```
 Geometricky vyznam urciteho integralu
----------------------------------------
Integrujeme funkci f(x) = x^2*sin(3*x)+3
na intervalu < 0.000000 , 3.141593 >.
Interval delime na 9 casti.

Presna hodnota integralu je ...  12.566498
Dolni odhad integralu je ...  8.822230
Horni odhad integralu je ...  15.802981
```



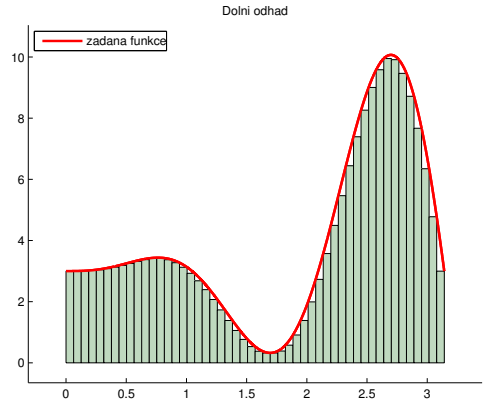


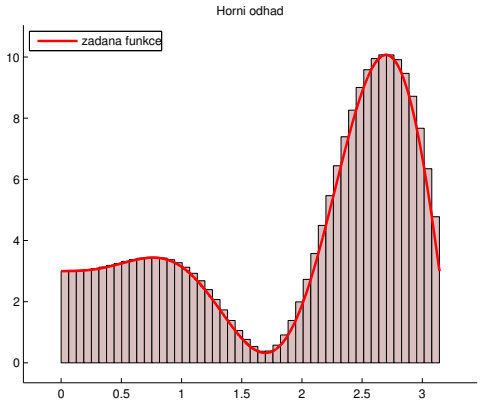

```
 Geometricky vyznam urciteho integralu
----------------------------------------
Integrujeme funkci f(x) = x^2*sin(3*x)+3
na intervalu < 0.000000 , 3.141593 >.
Interval delime na 20 casti.

Presna hodnota integralu je ...  12.566498
Dolni odhad integralu je ...  10.910270
Horni odhad integralu je ...  14.102775
```





```
  Geometricky vyznam urciteho integralu
-----------------------------------------
 Integrujeme funkci f(x) = x^2*sin(3*x)+3
 na intervalu < 0.000000 , 3.141593 >.
 Interval delime na 50 casti.

 Presna hodnota integralu je ...  12.566498

 Dolni odhad integralu je ...  11.916863
 Horni odhad integralu je ...  13.196682
```

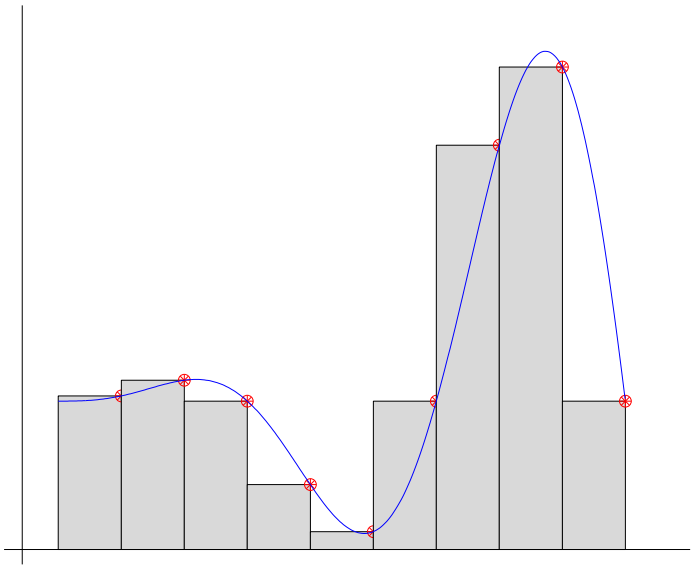




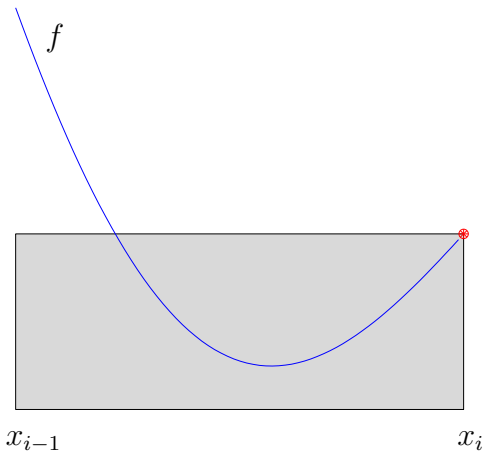
Podívejme se opět na způsob numerického výpočtu hodnoty určitého integrálu (stejná myšlenka jako v definici).

Uvažujme například vzorec:

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} f(x_i)(x_i - x_{i-1}).$$



Pokud nás zajímá chyba takového vzorce, musíme uvažovat funkce f, které jsou diferencovatelné. Na dílčím intervalu (x_{i-1}, x_i) potom dostáváme



Pro odvození chyby aproximace opět použijeme Taylorův rozvoj:

$$f(x) = f(x_i) + (x - x_i) \cdot f'(\xi_i), \qquad \xi_i \in (x_{i-1}, x_i).$$

Po integraci odvodíme chybu našeho vzorce:

$$\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x = f(x_i)h_i + \frac{1}{2}[\,\underbrace{(x_i - x_i)^2}_{=0} - \underbrace{(x_{i-1} - x_i)^2}_{h_i^2}\,]f'(\xi_i) = f(x_i)\,h_i - \frac{1}{2}\,h_i^2\,f'(\xi_i).$$

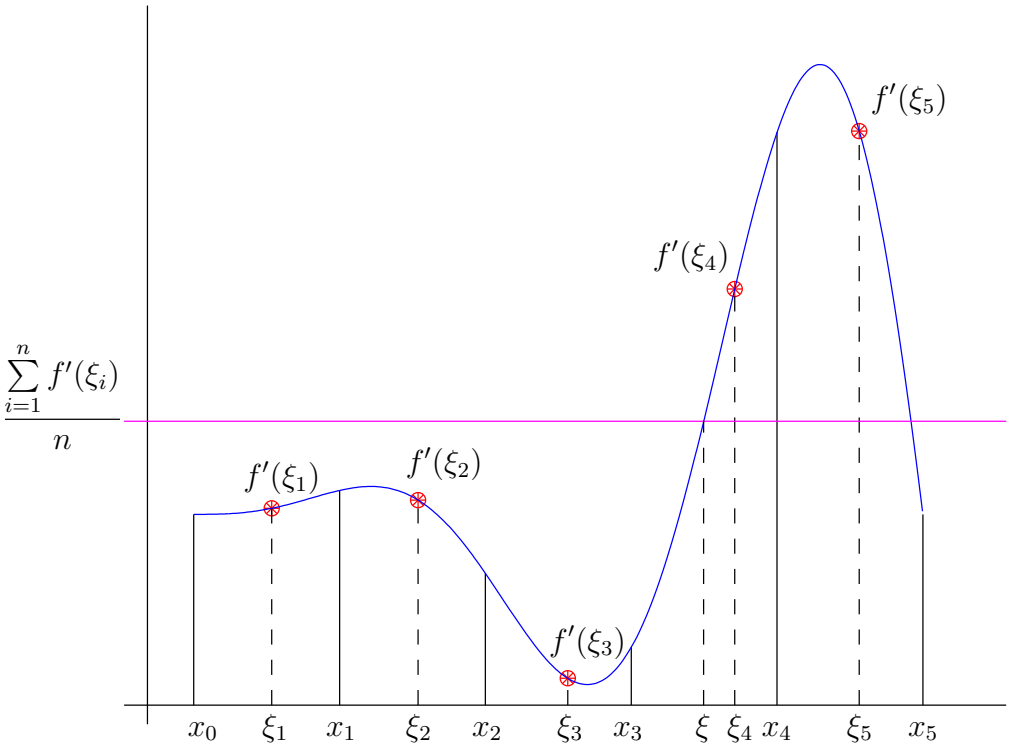
Po sečtení přes všechny dílčí intervaly:

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=1}^{n}\left[f(x_i)\,h_i - \frac{1}{2}\,h_i^2\,f'(\xi_i)\right] = \underbrace{\sum_{i=1}^{n} f(x_i)\,h_i}_{\text{náš vzorec}}\ \underbrace{-\frac{1}{2}\sum_{i=1}^{n} h_i^2\,f'(\xi_i)}_{\text{chyba vzorce}}.$$

Výraz pro chybu ještě zjednodušíme.

Předpokládejme, že je $f'(x)$ spojitá a krok h je ekvidistantní, tj. $\forall i:\ h_i = h$.

$$\sum_{i=1}^{n} -\frac{1}{2}\,h_i^2\,f'(\xi_i) = -\frac{1}{2}\,h^2\sum_{i=1}^{n} f'(\xi_i).$$



Průměr hodnot leží mezi minimální a maximální hodnotou

$$\min_i f'(\xi_i) \le \frac{\sum_{i=1}^{n} f'(\xi_i)}{n} \le \max_i f'(\xi_i)$$

Ze spojitosti $f'(x) \Rightarrow$

$$\exists \xi \in (a,b): \; f'(\xi) = \frac{\sum_{i=1}^{n} f'(\xi_i)}{n}$$

Pro chybu potom dostaneme

$$-\frac{1}{2}\,h^2 \sum_{i=1}^{n} f'(\xi_i) = -\frac{1}{2}\,h^2\,n\,f'(\xi) = -\frac{1}{2}h^2\,\frac{b-a}{h}\,f'(\xi) = \underbrace{-\frac{1}{2}\,(b-a)\,f'(\xi)}_{c}\,h.$$

Pro ekvidistantní krok h dostaneme:

$$\int_a^b f(x)\,\mathrm{d}x = h\sum_{i=1}^{n} f(x_i) - c\,h.$$

Poznámka: Úloha numerického výpočtu určitého integrálu je dobře podmíněná (narozdíl od numerického derivování).

Diskrétní případ

Tak jako derivaci funkce odpovídá pojem diference posloupnosti, tak určitému integrálu odpovídá **sumace posloupnosti**.

Definice:

Říkáme, že posloupnost $\{b_n\}$ je sumací posloupnosti $\{a_n\}$, platí-li

$$a_n = \triangle b_n \qquad \forall n = 0, 1, \ldots$$

Píšeme:

$$b_n = \triangle^{-1} a_n \quad \Leftrightarrow \quad a_n = \triangle b_n.$$

Příklad: Najděte sumaci posloupnosti $f(x_n) = n$.

Zadaná posloupnost je polynom prvního stupně, sumací musí být polynom 2. stupně, tj.:

$$\triangle^{-1} n = an^2 + bn + c.$$

Koeficienty a, b, c stanovíme z podmínky:

$$\triangle(an^2 + bn + c) = n.$$

$$\triangle(an^2 + bn + c) = a\triangle n^2 + b\triangle n + c\triangle 1 =$$
$$= a\left((n+1)^2 - n^2\right) + b\left((n+1) - n\right) + c \cdot 0 = 2an + a + b = n + 0.$$

Platí, pokud

$$2a = 1 \quad \text{a} \quad a + b = 0.$$

$$\Rightarrow \quad a = \frac{1}{2} \quad \text{a} \quad b = -\frac{1}{2}.$$

Závěr: $\triangle^{-1} n = \frac{1}{2}n^2 - \frac{1}{2}n + c.$

Poznámka:

Pojem sumace má význam při stanovení součtu s_n prvních $n+1$ členů posloupnosti $\{a_n\}$.

Posloupnost částečných součtů: $s_n = a_0 + a_1 + \ldots + a_n$

$$\Rightarrow s_{n+1} = a_0 + a_1 + \ldots + a_n + a_{n+1}$$

$$\triangle s_n = s_{n+1} - s_n = a_{n+1}$$

$$\Rightarrow \underline{s_n = \triangle^{-1} a_{n+1}}$$

Písemná práce z předmětu
Úvod do matematických výpočtů

Jméno a příjmení: .....

Přezdívka: ..... **Osobní číslo:** .....

Příklad 1.

Převeďte číslo 9,9 z desítkové soustavy do dvojkové soustavy. [5 b.]

Příklad 2.

Převeďte číslo $(0{,}1\overline{10})_2$ z dvojkové soustavy do desítkové soustavy. [5 b.]

Příklad 3.

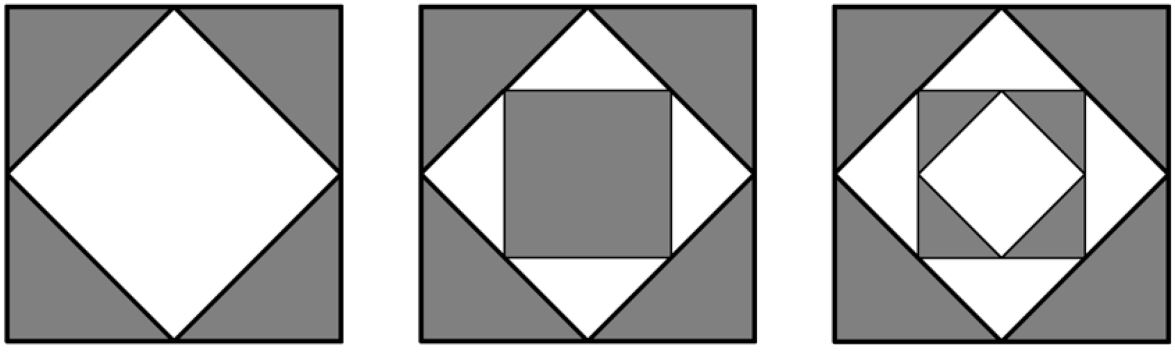
Vysvětlete pojem *špatně podmíněná úloha*. Posuďte podmíněnost úlohy vyčíslit hodnotu funkce $f(x) = \dfrac{10}{5\,x^2 - 1}$ v bodě $x = 1$. Hodnotu Δx volte 0,01. [5 b.]

Příklad 4.

Proveďte 4 kroky *metody půlení intervalu* pro řešení rovnice $\dfrac{x^2 - 7x - 18}{x + 1} = 0$ na intervalu $\langle 0; 16\rangle$. [5 b.]

Příklad 5.

Uvažujme čtverec o straně délky 1. Ze čtverce vyřízneme čtverec, jehož vrcholy jsou středy stran původního čtverce. Do vzniklé díry vložíme další čtverec tak, aby jeho vrcholy byly středy stran vyříznutého čtverce (viz obrázek). Proces opakujeme do nekonečna. Jak velký obsah bude mít výsledný obrazec?



[5 b.]

Příklad 6.

Vypočtěte přibližnou hodnotu derivace funkce $f(x) = \mathrm{e}^{\sqrt{x^2+1}}$ v bodě $x = 1$. Krok Δx volte 0,01. Uveďte vzorec, který jste použili. [5 b.]