Co je skryto za výpočty na PC ?

Můžeme jim věřit?

Josef Daněk

Katedra matematiky
Fakulta aplikovaných věd
Západočeská univerzita

Obsah

- motivační příklady
- počítání v konečné aritmetice
- podmíněnost úlohy a stabilita algoritmu
- iterační princip
- metody řešení nelineárních rovnic
- příklady závažných počítačových chyb

Motivační příklady

Motto: “Hodíme to do počítače a dostaneme výsledek”

Otázky:

- Jaký je výsledek?
- Dobrý?
- Špatný?
- Proč?

Příklad 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
```

Motivační příklady

Aritmetika s *n*-platnými ciframi s užitím zaokrouhlování nebo řezání

např. $\pi = 3{,}1415926535897932\ldots$

v aritmetice na 5 platných cifer :

se zaokrouhlováním $\pi \approx 3{,}1416$

s řezáním $\pi \approx 3{,}1415$

Motivační příklady

Příklad 2

Určete $f(300)$ a $g(300)$ pomocí aritmetiky se 6-ti platnými ciframi se zaokrouhlováním: $f(x) = x(\sqrt{x+1} - \sqrt{x})$ a $g(x) = \frac{x}{\sqrt{x+1}+\sqrt{x}}$

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

$$
\begin{aligned}
f(300) &\approx 300(\sqrt{301} - \sqrt{300}) \\
&= 300(17{,}3494 - 17{,}3205) = 300 \cdot 0{,}0289 \\
&= \mathbf{8{,}67000}
\end{aligned}
$$

$$
\begin{aligned}
g(300) &\approx \frac{300}{\sqrt{301}+\sqrt{300}} \\
&= \frac{300}{17{,}3494+17{,}3205} = \frac{300}{34{,}6699} \\
&= \mathbf{8{,}65304}
\end{aligned}
$$

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

Motivační příklady

Příklad 3

Určete $P(2{,}19)$ a $Q(2{,}19)$ v aritmetice na 3 platné cifry s zaokrouhlováním: $P(x) = x^3 - 3x^2 + 3x - 1$ a $Q(x) = ((x-3)x+3)x - 1$

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

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

$$
\begin{aligned}
Q(2{,}19) &\approx ((2{,}19 - 3)2{,}19 + 3)2{,}19 - 1 \\
&= (-0{,}81 \cdot 2{,}19 + 3)2{,}19 - 1 = (-1{,}77 + 3)2{,}19 - 1 \\
&= 1{,}23 \cdot 2{,}19 - 1 = 2{,}69 - 1 = \mathbf{1{,}69}
\end{aligned}
$$

Přesný výsledek je **1,685159 . . .**

(pro $P(2{,}19)$ je chyba 0,015159, pro $Q(2{,}19)$ je chyba -0,004841)

Motivační příklady

Příklad 4 Řešení kvadratické rovnice $ax^2 + bx + c = 0,\ a \neq 0$

Pro nezáporný diskriminant má rovnice řešení $x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$.
Lze použít alternativní formuli (pro $c \neq 0$) $x_{1,2} = \frac{2c}{-b \mp \sqrt{b^2-4ac}}$.

$$x_{1,2} = \frac{2c}{-b \mp \sqrt{b^2-4ac}} \cdot \frac{-b \pm \sqrt{b^2-4ac}}{-b \pm \sqrt{b^2-4ac}} \cdot \frac{\frac{1}{2c}}{\frac{1}{2c}} = \frac{-b \pm \sqrt{b^2-4ac}}{[b^2-(b^2-4ac)]\cdot\frac{1}{2c}} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

Uvažujeme rovnici s koeficienty a = 0,05010, b = -98,78, c = 5,015.
Kořeny rovnice pak jsou $x_1 \approx$ **1971,605916** a $x_2 \approx$ **0,05077069387**.

V aritmetice se 4 ciframi dostaneme pro diskriminant $D = b^2 - 4ac$
$D = 98,78^2 - 4 \cdot 0,05010 \cdot 5,015 = 9757 - 0,2004 \cdot 5,015 = 9756$
$\sqrt{D} = \sqrt{b^2 - 4ac} = 98,77$

Standardní formule:

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

Alternativní formule:

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

Motivační příklady

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

Počítání v konečné aritmetice

Číselné soustavy

Pro zobrazování čísel v počítači není vhodná desítková soustava. Používá se dvojková soustava. Jak převádíme čísla mezi 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:

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

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

Počítání v konečné aritmetice

Příklad 5 Převeďte číslo 139 z desítkové do dvojkové soustavy.

```
-------------------------------------------------------------------
Prevod cisla 139.00 z 10-soustavy do 2-soustavy na 8 desetinnych mist
Znamenko ............ +
Cela cast ........... 139
Desetinna cast ...... 0.00
-------------------------------------------------------------------
  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: $(139)_{10} = (10001011)_2$.

Počítání v konečné aritmetice

Příklad 6 Převeďte číslo $\frac{1}{10}$ z desítkové do dvojkové soustavy.

```
------------------------------------------------------------------
Prevod cisla 0.10 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 0.1 v 10-soustave prevedeno do 2-soustavy je 0.000110.
```

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

Ve dvojkové soustavě jde o periodické číslo (má nekonečný rozvoj).

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

Počítání v konečné aritmetice

Zobrazení čísel, strojová čísla

- 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 (počítač)
- výsledek v 10-soustavě

Binární zlomky lze vyjádřit jako sumu se zápornými mocninami dvou

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

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

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

Počítání v konečné aritmetice

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)

  $$\mathbf{x = \pm q \cdot 2^{n}}$$

  $\frac{1}{2} \leq q < 1 \ldots$ mantisa

  $n \ldots$ exponent

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

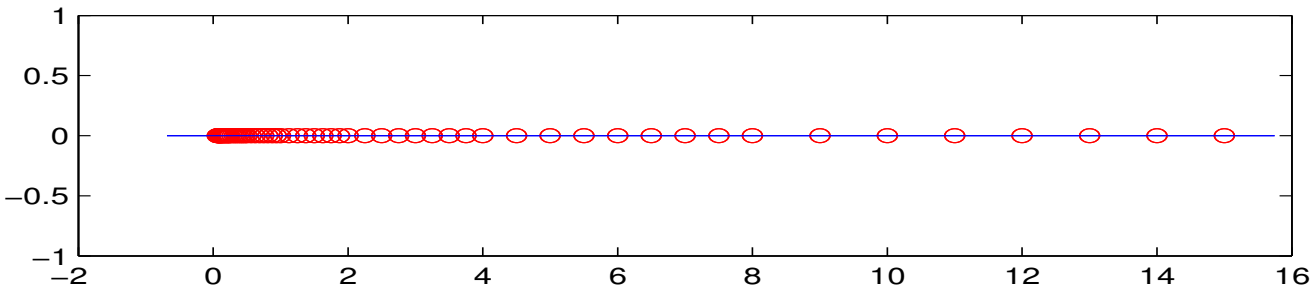
Počítání v konečné aritmetice

Příklad 7

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

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

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



Počítání v konečné aritmetice

Příklad 8

Uvažujme množinu strojových čísel z příkladu 7.
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
```

Počítání v konečné aritmetice

Příklad 9

Uvažujme množinu strojových čísel z příkladu 7.
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
```

Počítání v konečné aritmetice

Poznámka

V předposledním příkladu 8 se shodoval obraz přesného výsledku s obrazem součtu obrazů jednotlivých sčítanců.

V posledním příkladu 9 se obraz přesného výsledku s obrazem součtu obrazů jednotlivých sčítanců neshodoval !

Chyba výpočtu v posledním příkladu 8:

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

Relativně:

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

Počítání v konečné aritmetice

Přesnost počítače

- Vymezíme-li pro mantisu 24 bitů, získáme 7 desetinných míst ($2^{24} = 16\,777\,216$).
- Vymezíme-li pro mantisu 53 bitů, získáme 16 desetinných míst ($2^{53} = 9\,007\,199\,254\,740\,992$).

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

Počítání v konečné aritmetice

Příklad 10 S jakou chybou zobrazíme ve formátu SINGLE číslo $\frac{1}{10}$?

Pro mantisu je vyhrazeno 24 bitů, proto

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

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

Pokud počítáme $\sum_{k=1}^{100000} \frac{1}{10} = 9\,998,55664$, musí být chyba větší než

$$100000 \cdot 5,96 \cdot 10^{-9} = \mathbf{5,96} \cdot \mathbf{10^{-4}}.$$

Ve skutečnosti je **chyba ještě větší**, neboť v průběhu výpočtu suma roste a později přičítaná čísla $\frac{1}{10}$ jsou oproti sumě menší a jsou tedy počítány s menší přesností (viz následující příklady).

Počítání v konečné aritmetice

Příklad 11 Zobrazte ve formátu SINGLE číslo 10 000.

```
-----------------------------------------------------------------
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.
```

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

$$(10000)_{10} \approx 0,1001\,1100\,0100\,0000\,0000\,0000_2 \cdot 2^{14}$$

Počítání v konečné aritmetice

Příklad 12 **Sečtěte ve formátu SINGLE čísla 10 000 a $\frac{1}{10}$.**

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

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

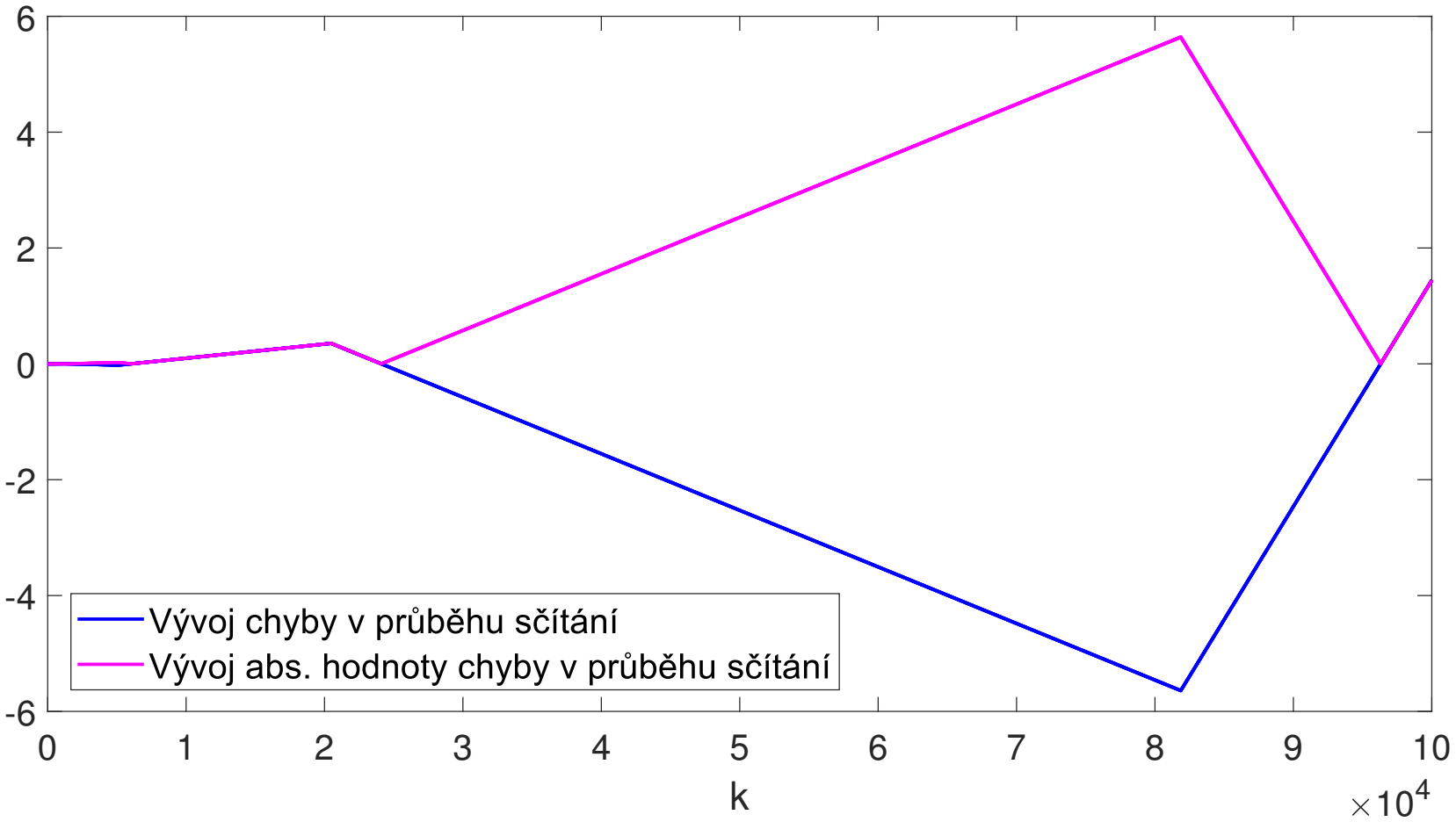
Číslo 10000 je zobrazeno přesně.

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

Shrnutí: 10000 + 0,1 $\rightarrow$ výsledek s chybou $3,90625 \cdot 10^{-4}$
(v sumě z motivačního příkladu jde o jeden krok)

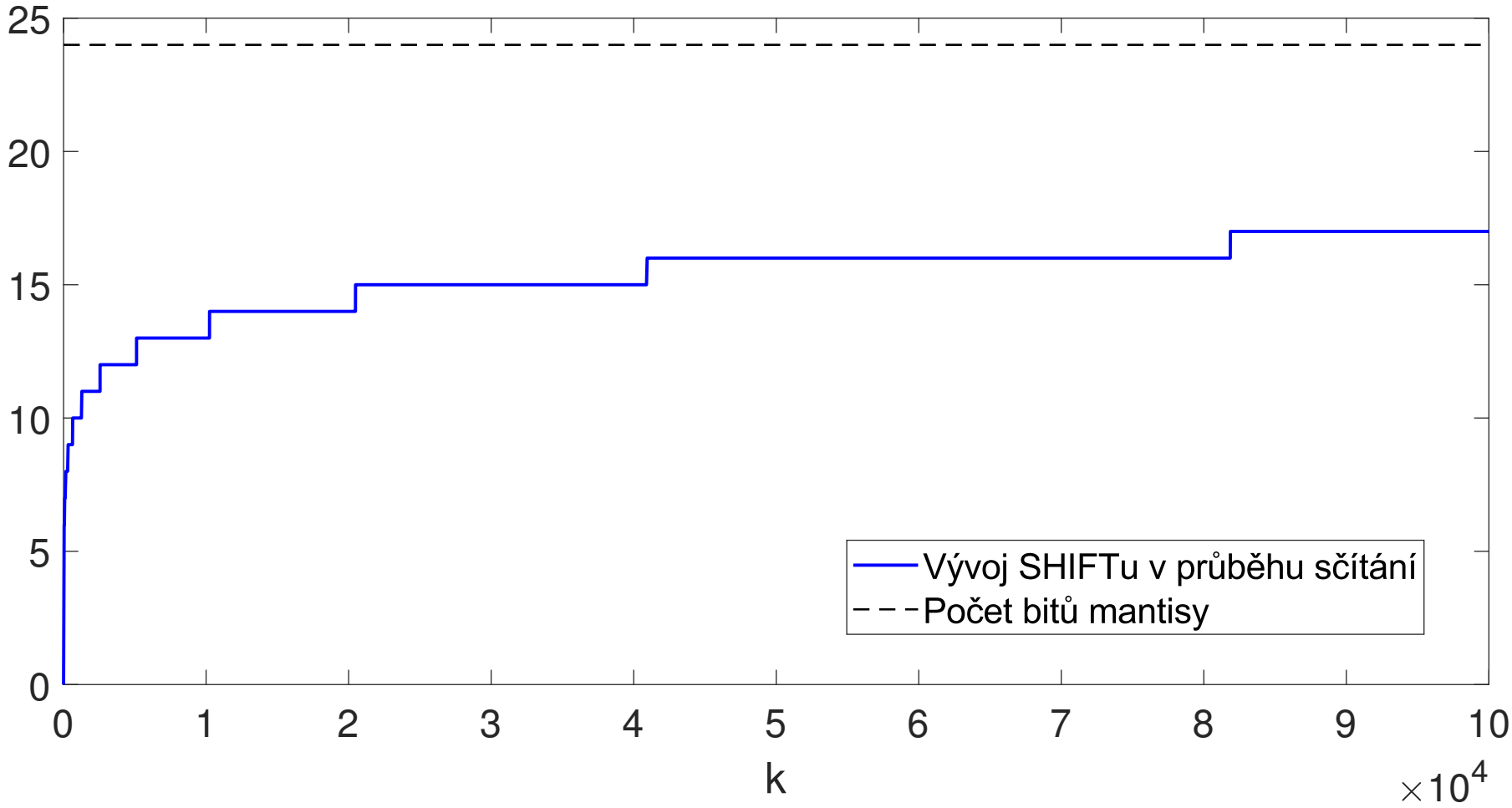
Počítání v konečné aritmetice

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



Počítání v konečné aritmetice

Následující obrázek ukazuje SHIFT exponentu menšího sčítance v průběhu sumace.



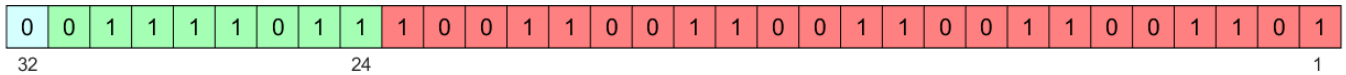
Počítání v konečné aritmetice

Poznámky

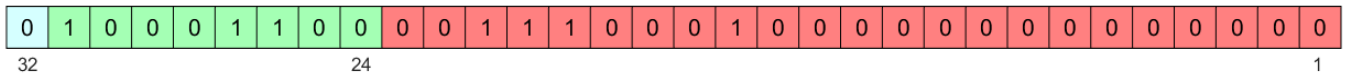
- Pokud použijeme formát DOUBLE, dostaneme přesnější výsledky, ovšem v principu bude situace obdobná, pouze se nepřesnosti projeví později.
- Existují postupy, jak eliminovat chybu při sumaci, např. https://en.wikipedia.org/wiki/Kahan_summation_algorithm

Znázornění čísel ve formátu SINGLE

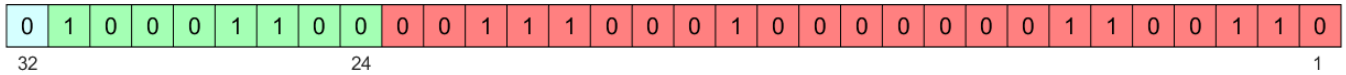
0,1



10 000



10 000,1



Počítání v konečné aritmetice

Příklad 13 Určete $S = 10\,000 + \pi + e$ v aritmetice na 6 platných číslic.

v dané aritmetice se zaokrouhlováním platí: $\pi \approx 3{,}14159$, $e \approx 2{,}71828$

$10\,000 + \pi \approx 10\,003{,}1 \quad 10\,003{,}1 + e \approx 10\,005{,}8 \quad$ tj. $S \approx \mathbf{10\,005{,}8}$

Přesně: $S \approx 10\,000 + 3{,}14159 + 2{,}71828 = 10\,005{,}85987 \approx \mathbf{10\,005{,}9}$

Kahanův algoritmus

```
function out = kahanova_sumace(x);
a = x(1);              % průběžný součet
e = 0;                 % chyba výpočtu
for j = 2:length(x)
  b = x(j) + e;        % korigovaný sčítanec
  s = a + b;           % odhad součtu
  e = b - (s - a); % chyba součtu
  a = s;
end;
out=a;
```

Počítání v konečné aritmetice

Kahanův algoritmus

```
function out = kahan(x);
a = x(1);
e = 0;
for j = 2:length(x)
  b = x(j) + e;
  s = a + b;
  e = b - (s - a);
  a = s;
end;
out=a;
```

x = [10000, π, e]
kahan(x)

---

a = 10000
e = 0

---

b = 3,14159 + 0
s = 10000 + 3,14159 = 10003,1
e = 3,14159 - (10003,1 - 10000) =
= 0,04159
a = 10003,1

---

b = 2,71828 + 0,04159 = 2,75987
s = 10003,1 + 2,75987 = 10005,9
e = 2,75987 - (10005,9 - 10003,1) =
= -0,04013
a = 10005,9

---

out = **10 005,9**

Podmíněnost úlohy a stabilita algoritmu

Absolutní a relativní chyba

- x ... teoretická (přesná) hodnota
- $\widehat{x}$ ... aproximace

Absolutní chyba:

$$\varepsilon = \widehat{x} - x \quad \Rightarrow \quad \boxed{\widehat{x} = x + \varepsilon}$$

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

Relativní chyba $(x \neq 0)$:

$$\delta = \frac{|(\widehat{x} - x)|}{|x|} = \frac{\text{absolutní chyba}}{\text{přesná hodnota}}$$

např.:

$$|3,1418 - 3,1415| = 0,0003 \qquad \frac{0,0003}{3,1415} \doteq 0,0001$$

$$|31418 - 31415| = 3 \qquad \frac{3}{31415} \doteq 0,0001$$

(Absolutní chyby) (Relativní chyby)

Platí:

$$\delta = \frac{\widehat{x} - x}{x} \quad \Rightarrow \quad \boxed{\widehat{x} = x(1 + \delta)}$$

Podmíněnost úlohy a stabilita algoritmu

Příklad 14 Výpočet hodnoty funkce kosinus pro argument blízký $\frac{\pi}{2}$.

```
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 a na vstupu  je 1.00000000

Pomer Relativni chyby na vystupu a na vstupu je 96208.717628
```

Podmíněnost úlohy a stabilita algoritmu

Předchozí problém byl **špatně podmíněný**, protože malá relativní změna vstupních dat způsobila velkou relativní změnu výstupních dat.

Ukažme teoreticky: $x \approx \frac{\pi}{2}$, $h > 0$ ... malá perturbace (chyba)

absolutní chyba:

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

relativní chyba:

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

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

$$\textbf{číslo podmíněnosti} = \frac{|\text{relativní chyba na výstupu}|}{|\text{relativní chyba na vstupu}|}$$

Pro použitá data vyšlo číslo podmíněnosti $\frac{0{,}61249}{0{,}637 \cdot 10^{-5}} \doteq \mathbf{0,96 \cdot 10^5}$

Podmíněnost úlohy a stabilita algoritmu

Podmíněnost úlohy řešit SLAR

Uvažujme soustavu $\mathbf{Ax} = \mathbf{b}$, kde $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

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 $\mathbf{Ax} = \mathbf{b}$

Pro pozměněná data platí: $(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x}^* + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$

Potom číslo podmíněnosti:

$$C_p = \frac{\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}^*\|}}{\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}} \quad (\Delta\mathbf{b} = \mathbf{0}), \qquad C_p = \frac{\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}^*\|}}{\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}} \quad (\Delta\mathbf{A} = \mathbf{0})$$

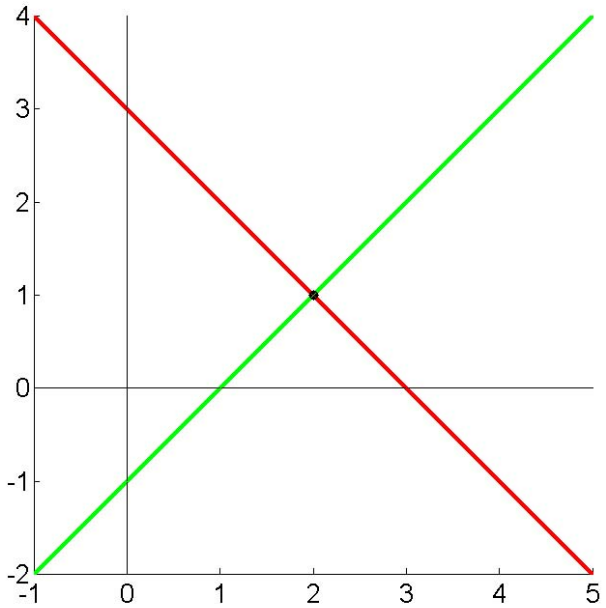
Podmíněnost úlohy a stabilita algoritmu

Příklad 15 Ukázka dobře podmíněné úlohy

(malá relativní změna vstupních dat vyvolá malou relativní změnu výstupních dat)

$$\begin{aligned} x + y &= 3 \\ x - y &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= 3 - x \\ y &= x - 1 \end{aligned}$$

řešení: $x^* = 2,\ y^* = 1$



Podmíněnost úlohy a stabilita algoritmu

Příklad 15 Ukázka dobře podmíněné úlohy

(malá relativní změna vstupních dat vyvolá malou relativní změnu výstupních dat)

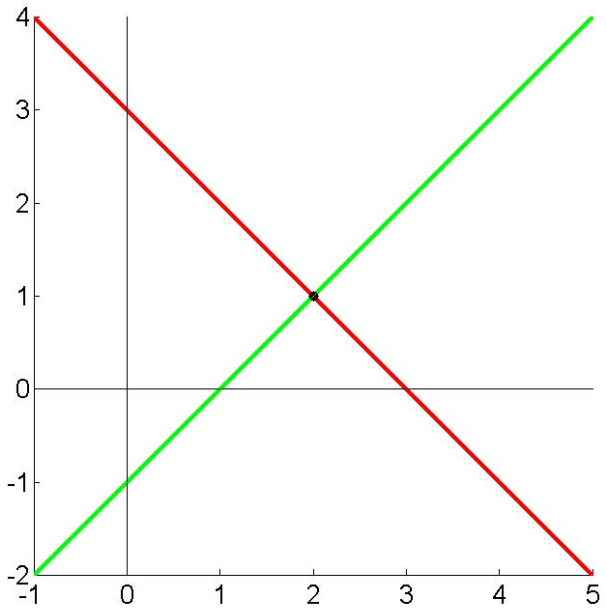
$$\begin{aligned} x + y &= 3 \\ x - y &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= 3 - x \\ y &= x - 1 \end{aligned}$$

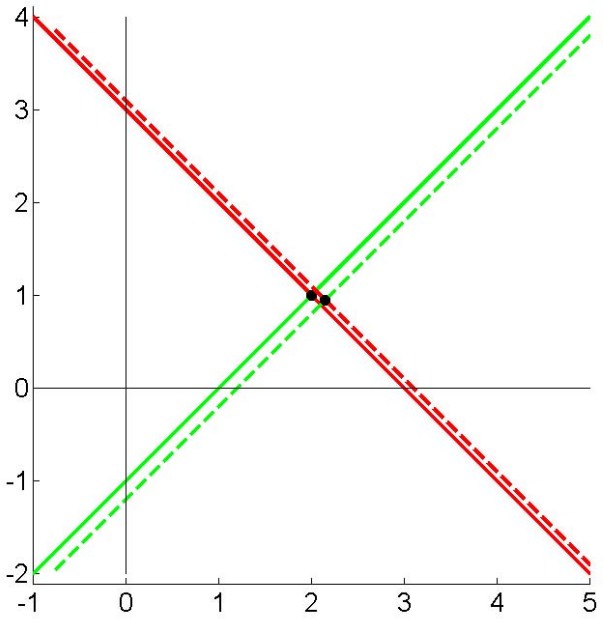
řešení: $x^* = 2,\ y^* = 1$

změna pravé strany

$$\begin{aligned} x + y &= 3{,}1 \\ x - y &= 1{,}2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= 3{,}1 - x \\ y &= x - 1{,}2 \end{aligned}$$

řešení: $x^* = 2{,}15,\ y^* = 0{,}95$





Podmíněnost úlohy a stabilita algoritmu

Příklad 15 Ukázka dobře podmíněné úlohy

(malá relativní změna vstupních dat vyvolá malou relativní změnu výstupních dat)

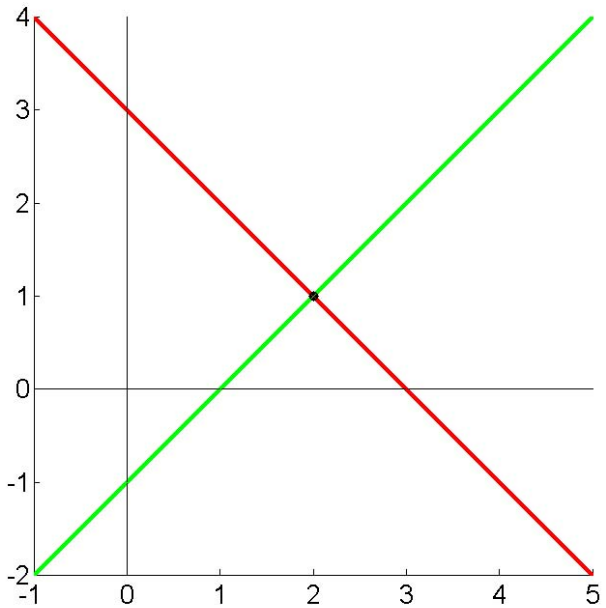
$$\begin{aligned} x + y &= 3 \\ x - y &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= 3 - x \\ y &= x - 1 \end{aligned}$$

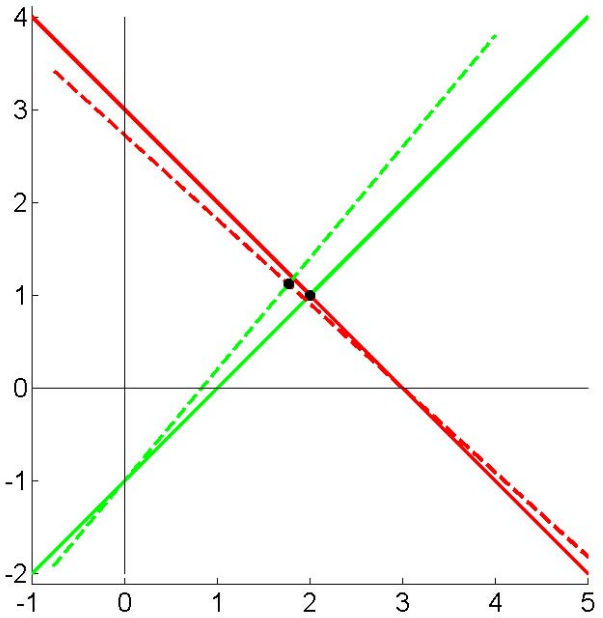
změna matice

$$\begin{aligned} x + 1{,}1\,y &= 3 \\ 1{,}2\,x - y &= 1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= \tfrac{1}{1{,}1}(3 - x) \\ y &= 1{,}2\,x - 1 \end{aligned}$$

řešení: $x^* = 2,\ y^* = 1$

řešení: $x^* \doteq 1{,}77,\ y^* \doteq 1{,}12$





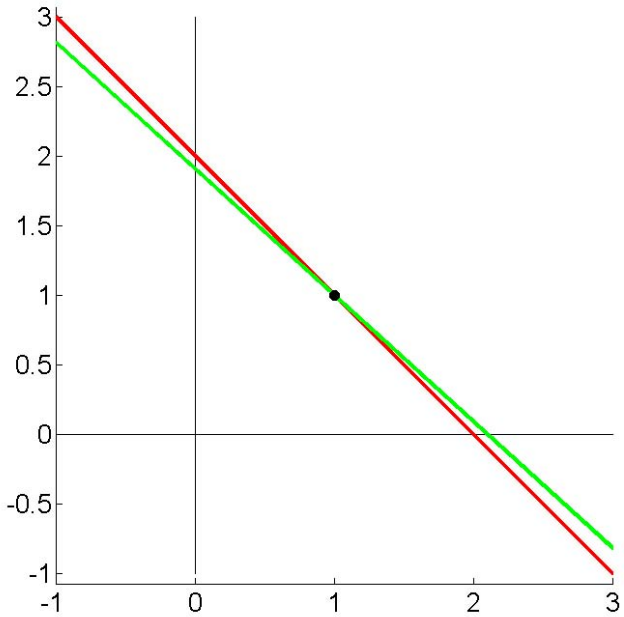
Podmíněnost úlohy a stabilita algoritmu

Příklad 16 Ukázka špatně podmíněné úlohy

(malá relativní změna vstupních dat vyvolá velkou relativní změnu výstupních dat)

$$\left.\begin{array}{c} x + y = 2 \\ x + 1{,}1\, y = 2{,}1 \end{array}\right| \Rightarrow \begin{array}{c} y = 2 - x \\ y = \frac{1}{1{,}1}(2{,}1 - x) \end{array}$$

řešení: $x^* = 1,\ y^* = 1$



Podmíněnost úlohy a stabilita algoritmu

Příklad 16 Ukázka špatně podmíněné úlohy

(malá relativní změna vstupních dat vyvolá velkou relativní změnu výstupních dat)

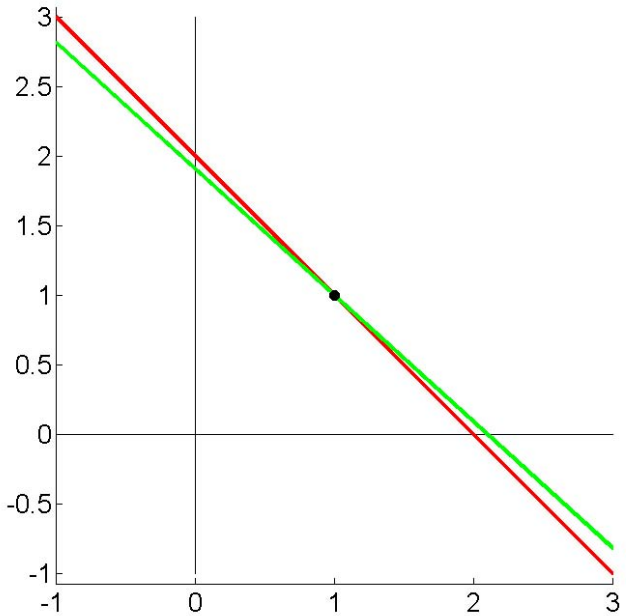
$$\begin{array}{c} x + y = 2 \\ x + 1{,}1\, y = 2{,}1 \end{array} \Rightarrow \begin{array}{c} y = 2 - x \\ y = \frac{1}{1{,}1}(2{,}1 - x) \end{array}$$

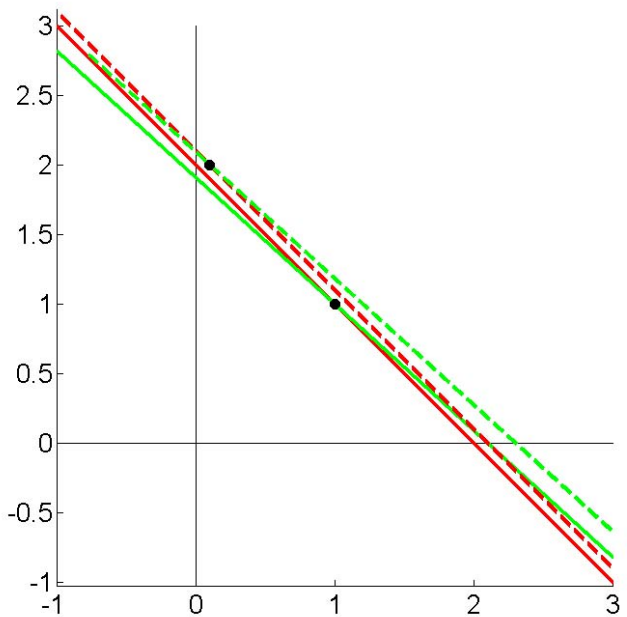
změna pravé strany

$$\begin{array}{c} x + y = 2{,}1 \\ x + 1{,}1\, y = 2{,}3 \end{array} \Rightarrow \begin{array}{c} y = 2{,}1 - x \\ y = \frac{1}{1{,}1}(2{,}3 - x) \end{array}$$

řešení: $x^* = 1,\ y^* = 1$

řešení: $x^* = 0{,}1,\ y^* = 2$





Podmíněnost úlohy a stabilita algoritmu

Příklad 16 Ukázka špatně podmíněné úlohy

(malá relativní změna vstupních dat vyvolá velkou relativní změnu výstupních dat)

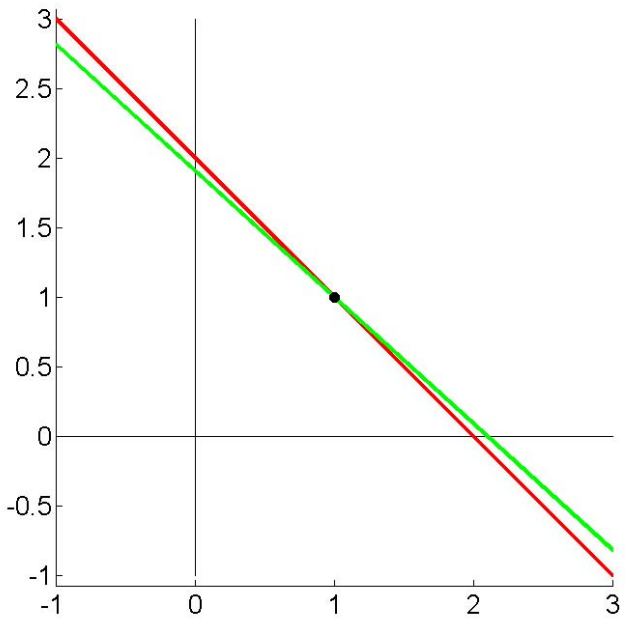
$$\begin{aligned} x + y &= 2 \\ x + 1{,}1\, y &= 2{,}1 \end{aligned} \Rightarrow \begin{aligned} y &= 2 - x \\ y &= \tfrac{1}{1{,}1}(2{,}1 - x) \end{aligned}$$

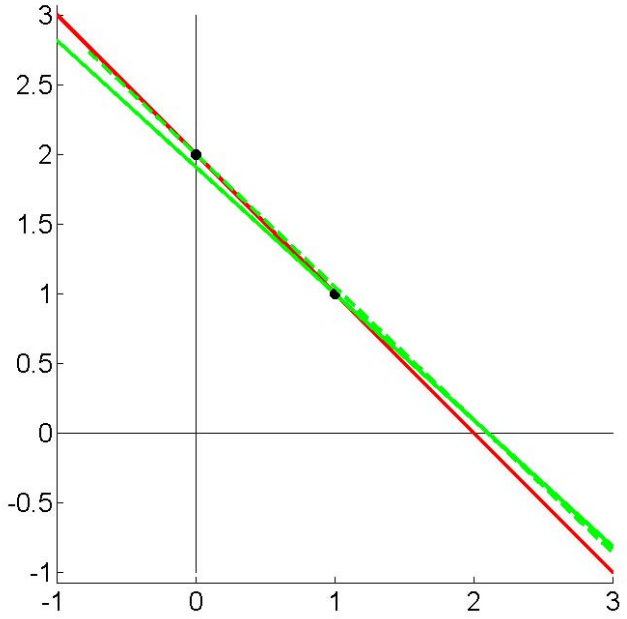
řešení: $x^* = 1,\ y^* = 1$

změna matice

$$\begin{aligned} x + y &= 2 \\ x + 1{,}05\, y &= 2{,}1 \end{aligned} \Rightarrow \begin{aligned} y &= 2 - x \\ y &= \tfrac{1}{1{,}05}(2{,}1 - x) \end{aligned}$$

řešení: $x^* = 0,\ y^* = 2$





Podmíněnost úlohy a stabilita algoritmu

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

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

stabilita algoritmu + **dobrá podmíněnost úlohy** $\Rightarrow$

$\Rightarrow$ **přesnost výsledku**

Podmíněnost úlohy a stabilita algoritmu

Vraťme se k příkladu 4, ve kterém jsme řešili kvadratickou rovnici $ax^2 + bx + c = 0$ s koeficienty a = 0,05010, b = -98,78, c = 5,015.

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 (číslo podmíněnosti ≈ 1).

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

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

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

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

Podmíněnost úlohy a stabilita algoritmu

```
-----------------------------------------
 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
-----------------------------------------
```

Podmíněnost úlohy a stabilita algoritmu

Příklad 17

Určete prvních 15 členů posloupnosti, která je zadána rekurentní formulí

$$\begin{cases} x_1 = 5 \\ x_2 = 1 \\ x_n = 20{,}2\, x_{n-1} - 4\, x_{n-2}, \quad n \geq 3. \end{cases}$$

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

Ověření:

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

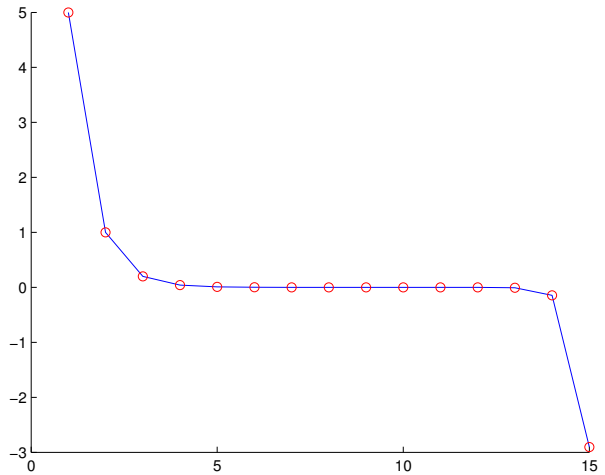
$$x_2 = \left(\frac{1}{5}\right)^{2-2} = 1$$

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

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

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

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



Podmíněnost úlohy a stabilita algoritmu

Obecné řešení diferenční rovnice $x_n = 20,2\, x_{n-1} - 4x_{n-2}$

Řešení hledáme ve tvaru $x_n = A^n$

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

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

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

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

V našem případě je $c_1 = 0$.

Vlivem zaokrouhlovacích chyb se stane koeficient c_1 nenulový

$\Rightarrow$ **nestabilní rekurze**

Iterační princip

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

Vezměme si například polynom n-tého stupně:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0,\ a_k \in \mathbb{C},\ k = 0, 1 \ldots, n,\ a_n \neq 0.$$

Polynom má právě n kořenů v $\mathbb{C}$ počítáme-li je i s jejich násobností.

Důkaz podal Jean-Robert Argand (1806).

Uvažujme nejjednodušší případy:

- lineární rovnice $ax + b = 0$ má řešení $x_1 = -\frac{b}{a}$ ✓
- kvadratická rovnice $ax^2 + bx + c = 0$

  má řešení $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ (Babyloňané 2000 let př.n.l.) ✓

Iterační princip

- kubická rovnice $ax^3 + bx^2 + cx + d = 0$ má řešení, kde

$$x_1 = \sqrt[3]{\left(\frac{b\,c}{6\,a^2} - \frac{b^3}{27\,a^3} - \frac{d}{2\,a}\right) + \sqrt{\left(\frac{d}{2\,a} + \frac{b^3}{27\,a^3} - \frac{b\,c}{6\,a^2}\right)^2 + \left(\frac{c}{3\,a} - \frac{b^2}{9\,a^2}\right)^3}} - \ldots$$

$x_{2,3} = \ldots$ Niccolo Tartaglia, Scipione del Ferro (okolo 1500) ✓

- kvartická rovnice $ax^4 + bx^3 + cx^2 + dx + e = 0$ má řešení

$x_{1,2,3,4} = \ldots$ Gerolamo Cardano, Lodovico Ferrari (1540) ✓

- pro polynom stupně $n \geq 5$ $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$

bylo dokázáno, že obecně nelze řešení spočítat pomocí konečného počtu kroků pomocí operací $+, -, \cdot, /$ a libovolné odmocniny $\sqrt[r]{\ }$, $r \in \mathbb{N}$

Paolo Ruffini (nekompletní 1813), Niels Henrik Abel (1824) ✗

Iterační princip

Příklad 18 Hledání kořenů polynomu $P(x) = (x-1)^n$.

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

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

  polynom =   1    -4     6    -4     1

  koreny =

  1.00022371630863e+00
  9.99999706778071e-01 + 2.23342301514788 9e-04i
  9.99999706778071e-01 - 2.23342301514788 9e-04i
  9.99776687013525 2e-01
```

Příklad 19 Najděte kladné řešení rovnice $x^2 + x - 2 = 0$.

- 1. způsob:

  úlohu lze řešit použitím vzorce pro řešení kvadratické rovnice

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

  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 nalezneme přesné řešení. Je třeba si uvědomit, že jen pro některé problémy máme k dispozici vzorec a lze je řešit přímo.

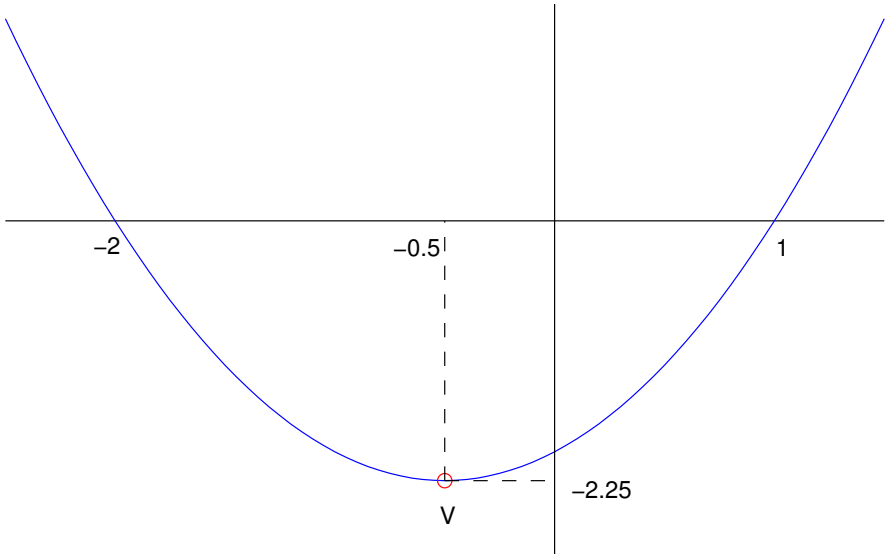
Metody řešení nelineárních rovnic

- 2. způsob:

  pro řešení můžeme použít nějakou **iterační metodu**

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

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



vrchol odpovídá bodu, ve kterém je funkční hodnota minimální, tj. pro $x = -\frac{1}{2}$ je $f\left(-\frac{1}{2}\right) = -2,25$

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

Metoda půlení intervalu (bisekce)

Algoritnus:

1. Zadáme $\varepsilon > 0$, a, b
2. $s = (a + b)/2$
3. Je-li $f(s) = 0$, pak $x = s$, KONEC
4. Je-li $f(s) \neq 0$, pak

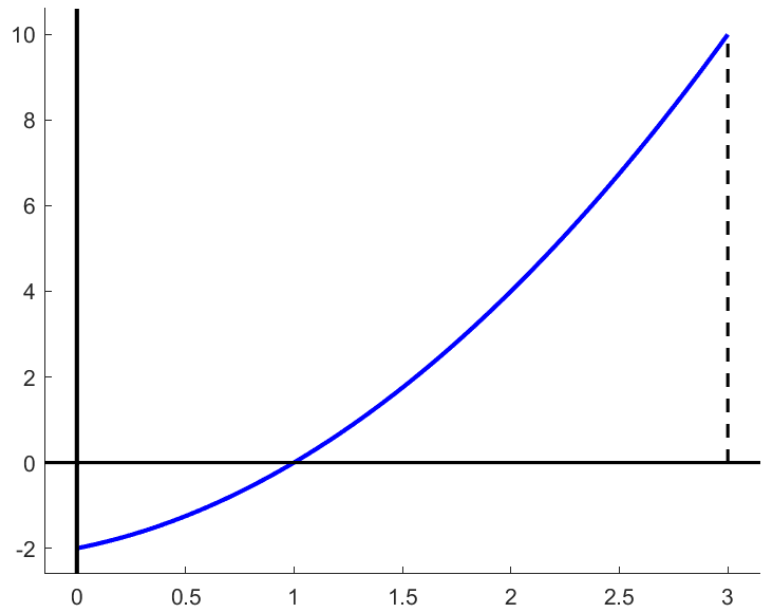
   je-li $f(a) \cdot f(s) < 0$, pak $b = s$

   jinak $a = s$
5. Je-li $b - a \geq \varepsilon$, pak jdi na 2)

   jinak $x = (a + b)/2$, KONEC

Metody řešení nelineárních rovnic

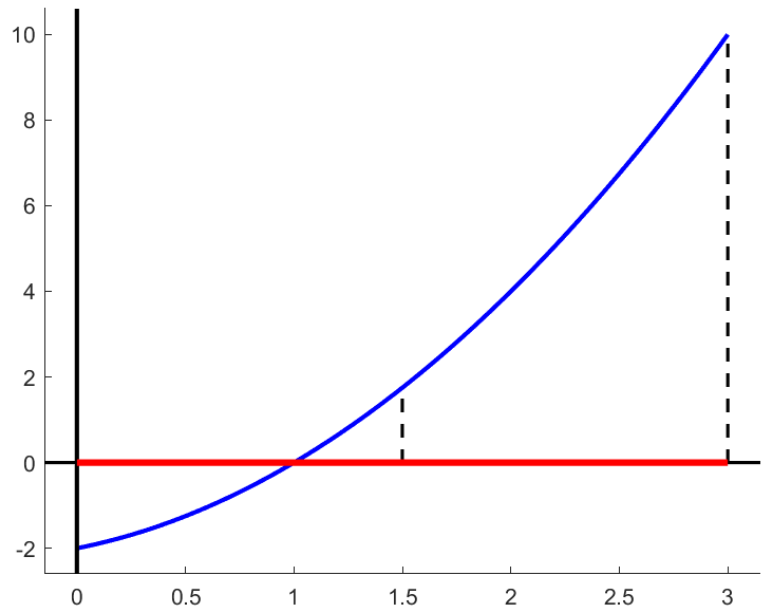
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```




```
|krok|     a     |     b     |     s     |    f(a)   |    f(b)   |    f(s)   |
-------------------------------------------------------------------------------
```

Metody řešení nelineárních rovnic

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```

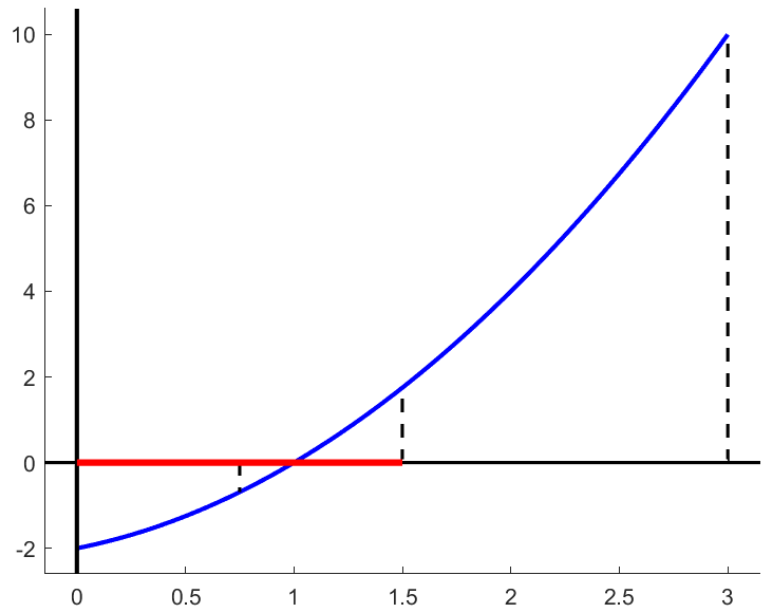


krok	a	b	s	f(a)	f(b)	f(s)
0	0.0000	3.0000	1.5000	-2.0000	10.0000	1.7500

Metody řešení nelineárních rovnic

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1
```

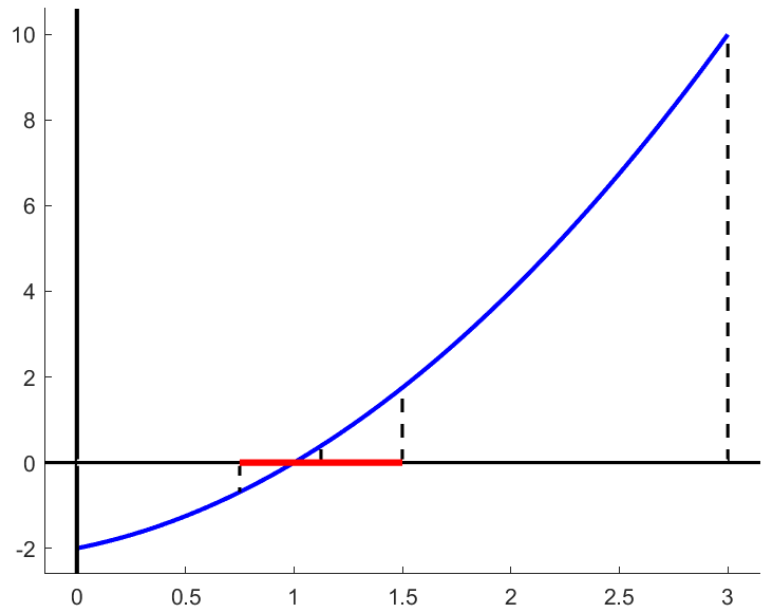
$f(x)=x^2+x-2$



krok	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

Metody řešení nelineárních rovnic

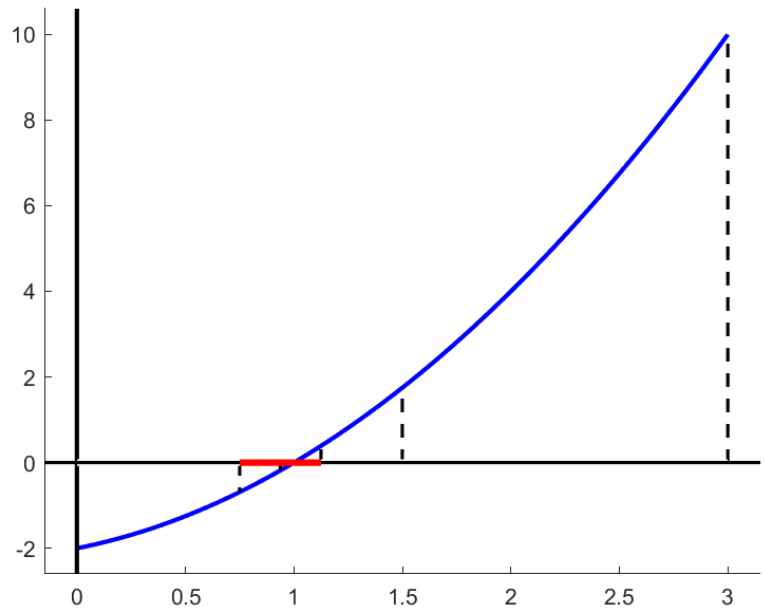
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```



krok	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

Metody řešení nelineárních rovnic

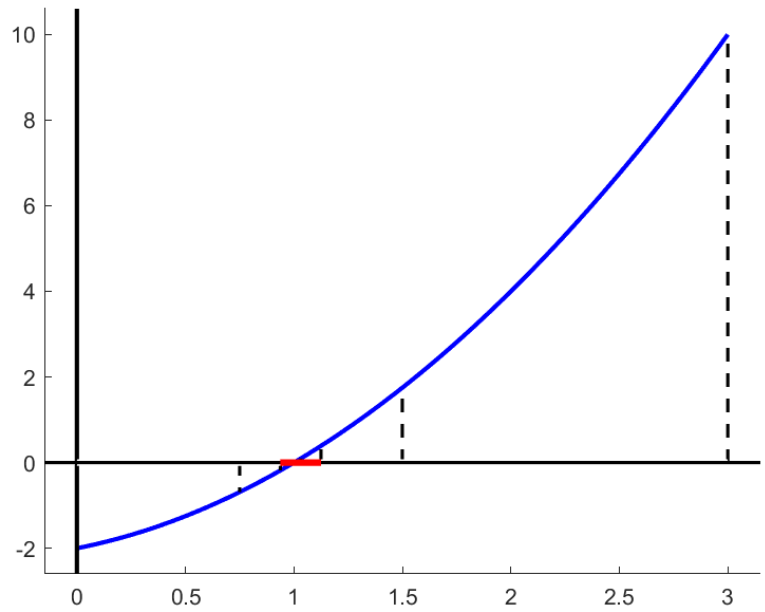
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```



krok	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

Metody řešení nelineárních rovnic

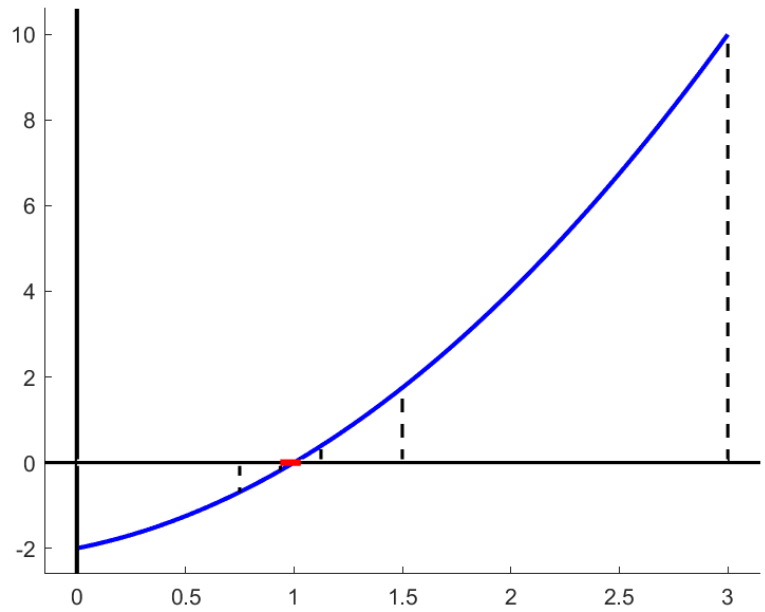
```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```



krok	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	1.0313	-0.1836	0.3906	0.0947

Metody řešení nelineárních rovnic

```
Metoda puleni intervalu pro reseni nelinearni rovnice f(x)=0
na intervalu <0,3> a zastavovaci podminku b-a<0.1

f(x)=x^2+x-2
```



krok	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	1.0313	-0.1836	0.3906	0.0947
5	0.9375	1.0313	0.9844	-0.1836	0.0947	-0.0466

Metoda prosté iterace

Algoritnus:

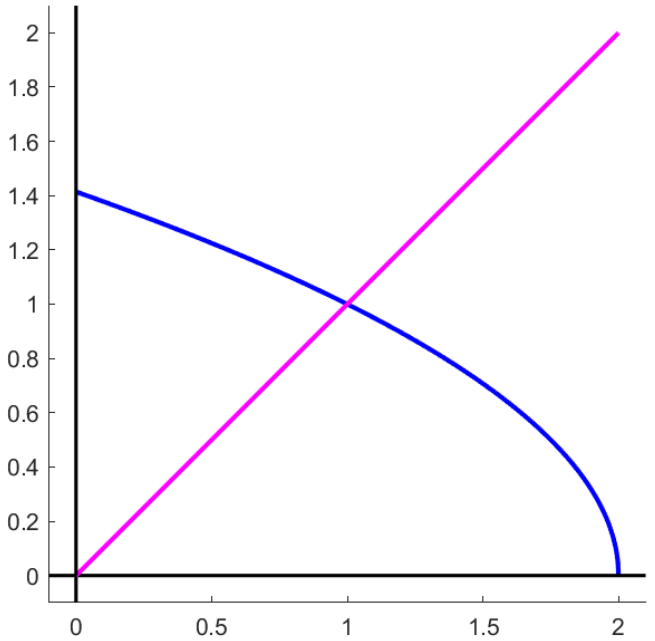
1. Zadáme $x_0 \in \langle a; b \rangle$, $\varepsilon > 0$
2. $x_{k+1} = \varphi(x_k)$
3. Je-li $|x_{k+1} - x_k| < \varepsilon$,

   pak $x = x_{k+1}$, KONEC

   jinak jdi na 2)

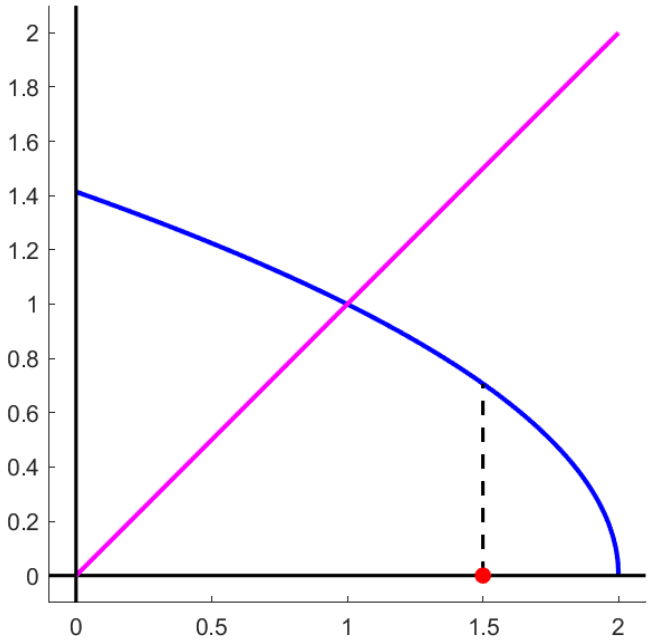
Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)


| krok |    x(k)    | x(k)-x(k-1) |
-------------------------------
```



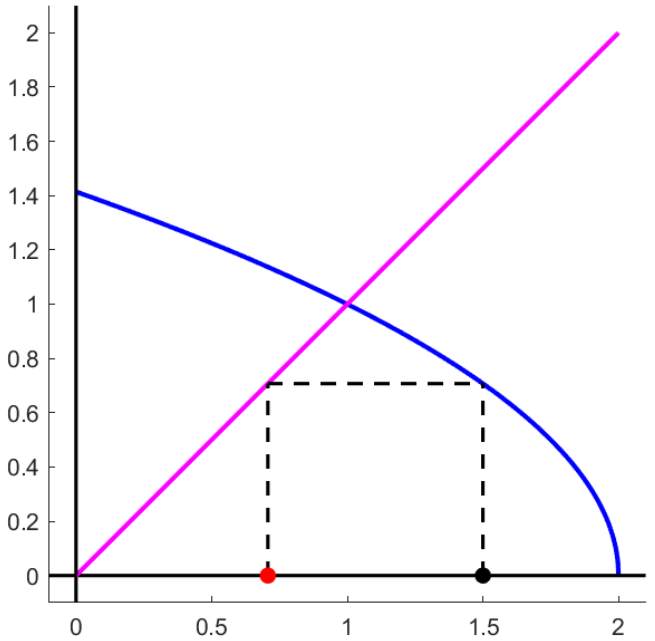
Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)


| krok |    x(k)    | x(k)-x(k-1)  |
-------------------------------------
|   0  | 1.500000 |                |
```



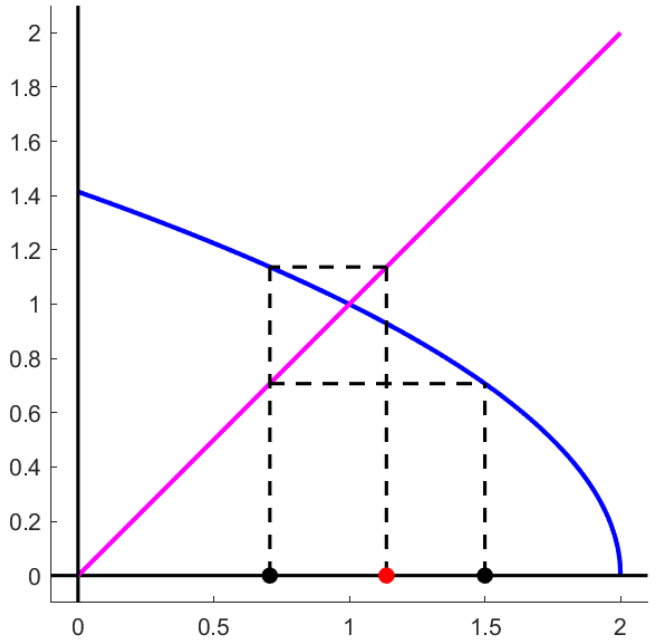
Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)


| krok |    x(k)    | x(k)-x(k-1)  |
-------------------------------------
|    0   | 1.500000 |                |
|    1   | 0.707107 |   -0.792893    |
```



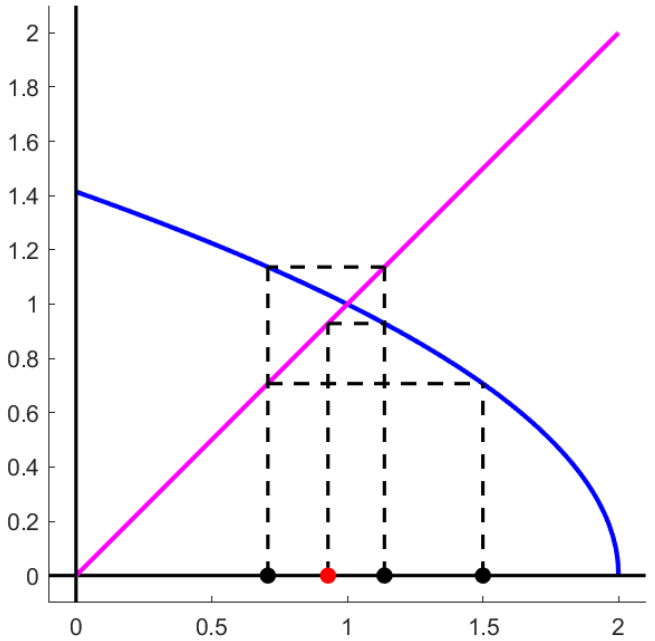
Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)


| krok |    x(k)    | x(k)-x(k-1)  |
-------------------------------------
|    0  | 1.500000 |                |
|    1  | 0.707107 |  -0.792893   |
|    2  | 1.137055 |   0.429948   |
```



Metody řešení nelineárních rovnic

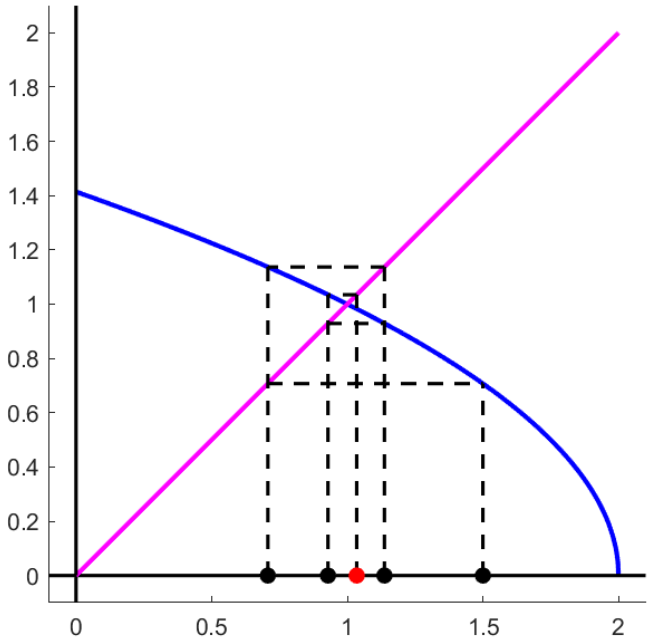
```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)


| krok |    x(k)    | x(k)-x(k-1)  |
-------------------------------------
|    0 | 1.500000 |                |
|    1 | 0.707107 |   -0.792893    |
|    2 | 1.137055 |    0.429948    |
|    3 | 0.928949 |   -0.208106    |
```



Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)
```

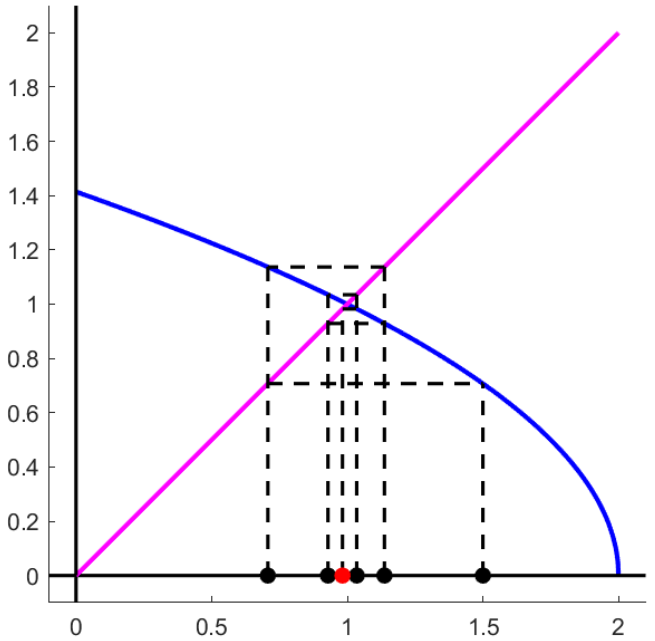
krok	x(k)	x(k)-x(k-1)
0	1.500000	
1	0.707107	-0.792893
2	1.137055	0.429948
3	0.928949	-0.208106
4	1.034916	0.105968



Metody řešení nelineárních rovnic

```
Metoda proste iterace pro reseni nelinearni rovnice x=phi(x)
pro pocatecni aproximaci x0=1.5 a zast. podminku |x(k)-x(k-1)|<0.1

  x^2+x-2=0

  x^2=2-x

  x=sqrt(2-x)

  tj.   phi(x)=sqrt(2-x)
```

krok	x(k)	x(k)-x(k-1)
0	1.500000	
1	0.707107	-0.792893
2	1.137055	0.429948
3	0.928949	-0.208106
4	1.034916	0.105968
5	0.982387	-0.052529



Newtonova metoda (metoda tečen)

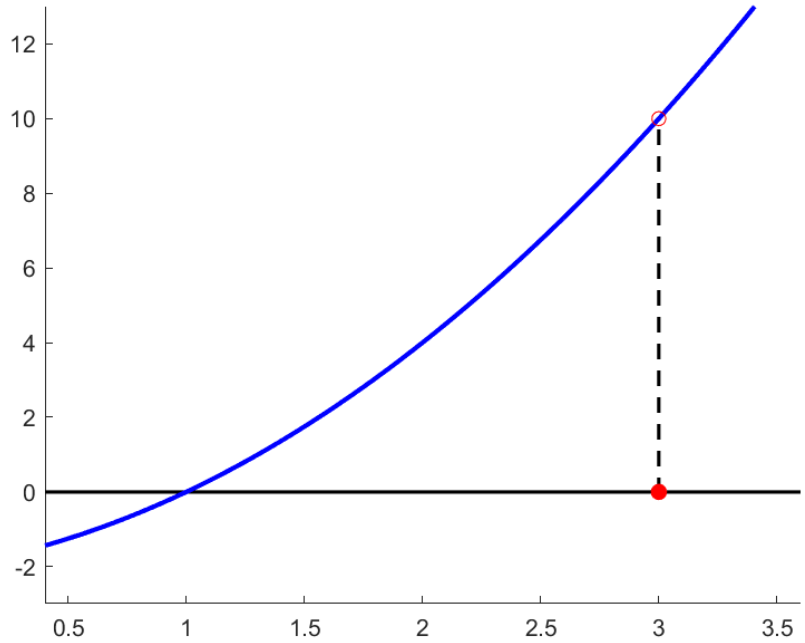
Algoritnus:

1. Zadáme $x_0 \in \langle a; b \rangle$, $\varepsilon > 0$
2. $x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)}$
3. Je-li $|x_{k+1} - x_k| < \varepsilon$,
   pak $x = x_{k+1}$, KONEC
   jinak jdi na 2)

Metody řešení nelineárních rovnic

```
Newtonova metoda pro reseni nelinearni rovnice f(x)=0
pro pocatecni aproximaci x0=3 a zastavovaci podminku |x(k)-x(k-1)|<0.1
```

```
f(x)=x^2+x-2

f'(x)=2*x+1;
```



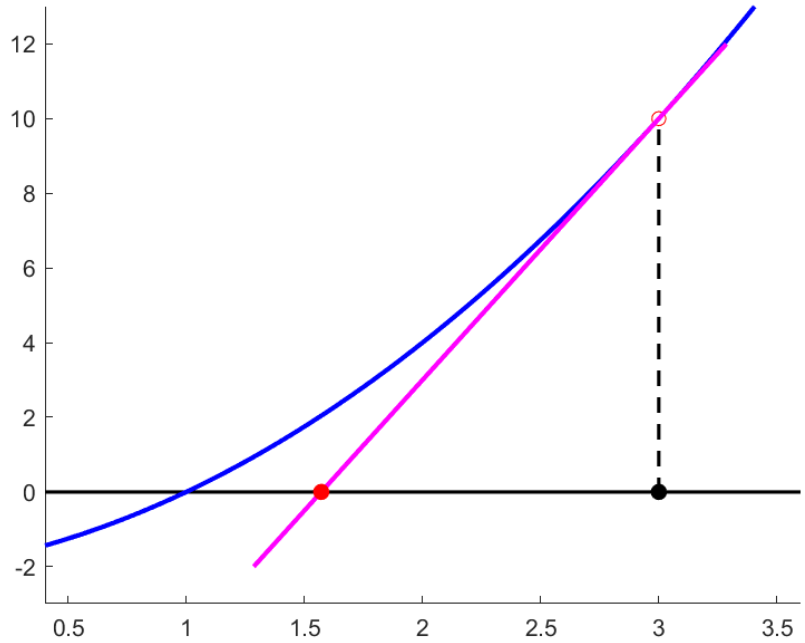

```
| krok |     x(k)     | x(k)-x(k-1) |    f(x(k))   |
-----------------------------------------------------
|    0   | 3.00000000 |             |              |
```

Metody řešení nelineárních rovnic

```
Newtonova metoda pro reseni nelinearni rovnice f(x)=0
pro pocatecni aproximaci x0=3 a zastavovaci podminku |x(k)-x(k-1)|<0.1
```

$f(x)=x^2+x-2$

$f'(x)=2*x+1;$



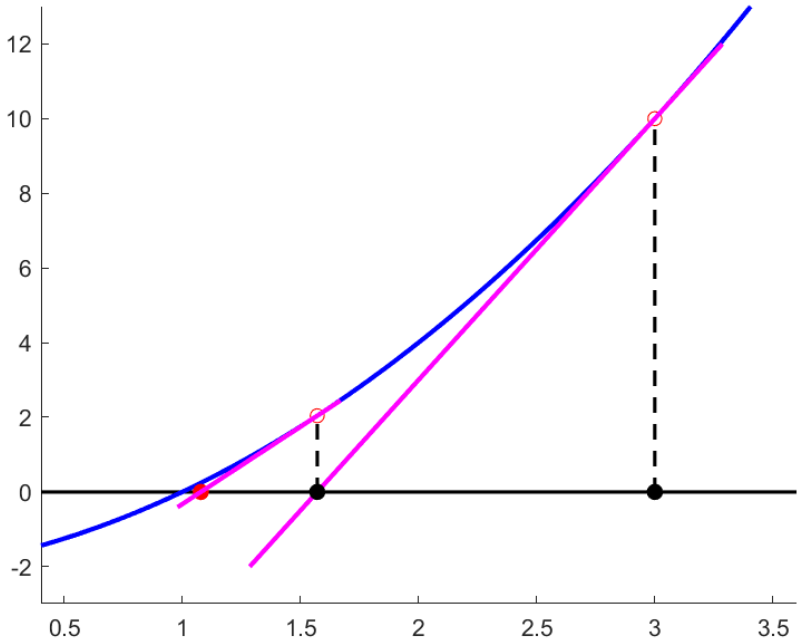
krok	x(k)	x(k)-x(k-1)	f(x(k))
0	3.00000000		
1	1.57142857	1.42857143	2.04081633

Metody řešení nelineárních rovnic

```
Newtonova metoda pro reseni nelinearni rovnice f(x)=0
pro pocatecni aproximaci x0=3 a zastavovaci podminku |x(k)-x(k-1)|<0.1
```

$f(x)=x^2+x-2$

$f'(x)=2*x+1;$



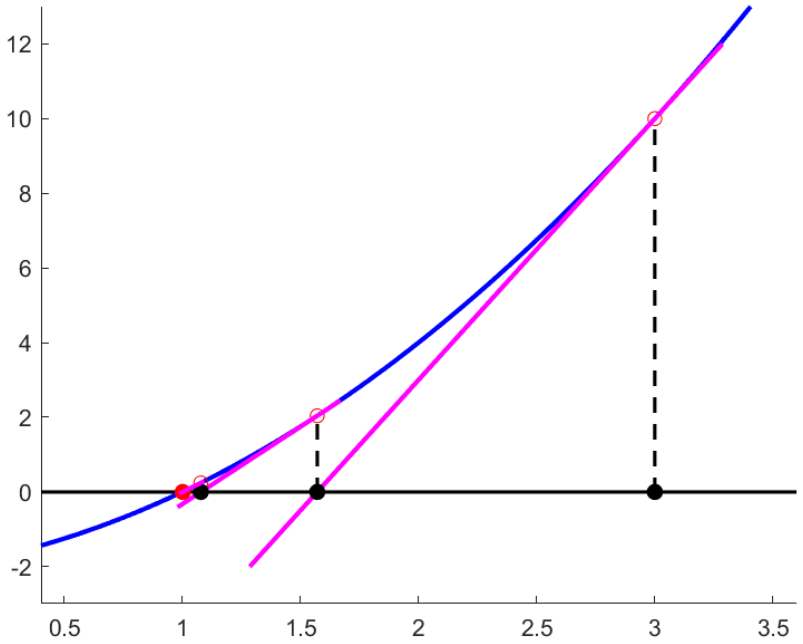
krok	x(k)	x(k)-x(k-1)	f(x(k))
0	3.00000000		
1	1.57142857	1.42857143	2.04081633
2	1.07881773	0.49261084	0.24266544

Metody řešení nelineárních rovnic

```
Newtonova metoda pro reseni nelinearni rovnice f(x)=0
pro pocatecni aproximaci x0=3 a zastavovaci podminku |x(k)-x(k-1)|<0.1
```

$f(x) = x^2 + x - 2$

$f'(x) = 2 * x + 1;$

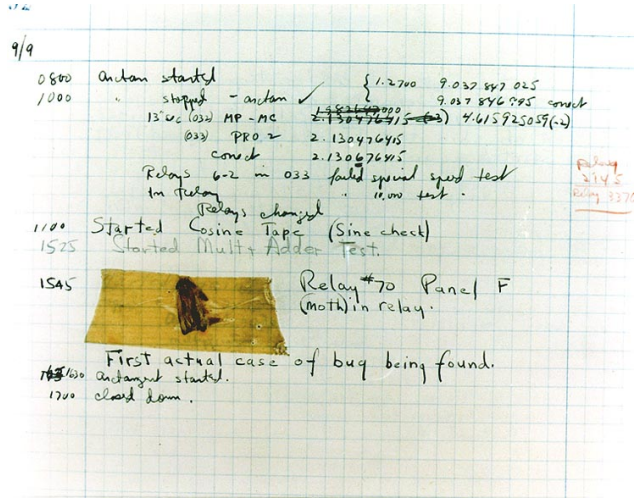


krok	x(k)	x(k)-x(k-1)	f(x(k))
0	3.00000000		
1	1.57142857	1.42857143	2.04081633
2	1.07881773	0.49261084	0.24266544
3	1.00196737	0.07685036	0.00590598

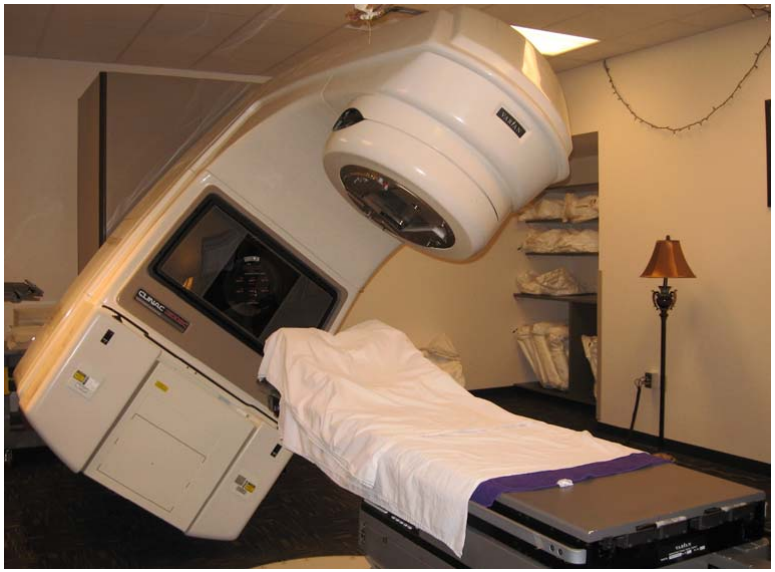
Příklady závažných počítačových chyb

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. Označení použil T. A. Edison již v roce 1878, když mluvil o svých vynálezech. S počítači pak pronikl do mnoha dalších jazyků.
- Známá je 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



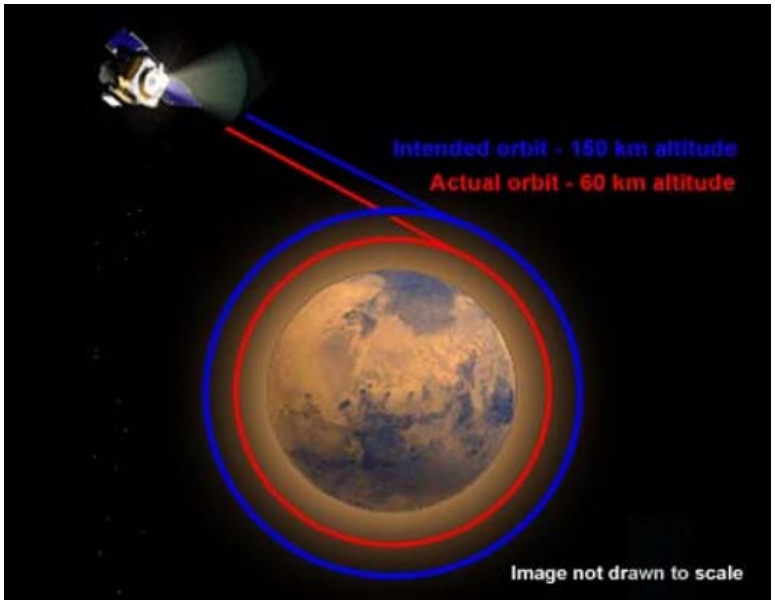
- radiologický přístroj pro léčbu nádorů ozařováním
- červen 1985 - leden 1987 prokázáno **6 případů úmrtí**
- ozařoval ve dvou režimech (x-paprsky, elektrony)

chyby v programu (nedostatečně ošetřené situace):

- **race condition** - souběh událostí
  chyba při nastavení parametrů operátorem a jejich rychlé opravě
- 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 = 8 bitů)

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)

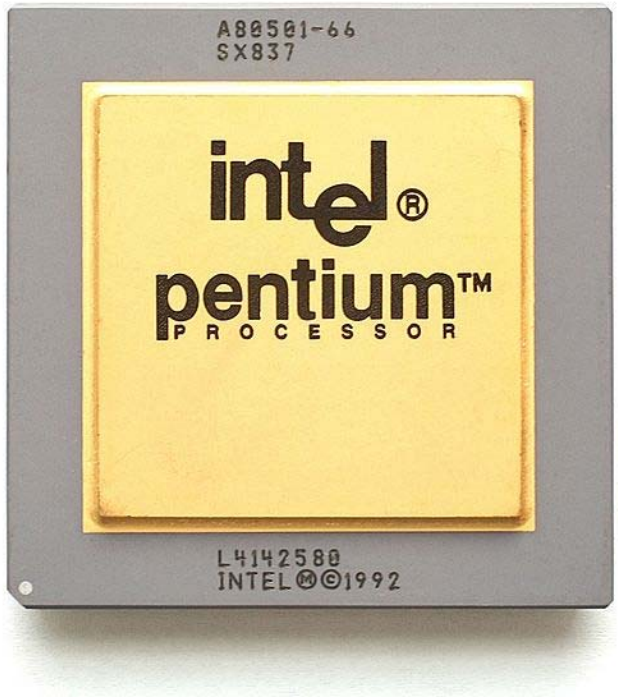


- ve skutečnosti se dostal na jinou oběžnou dráhu (cca 60 km)
- příčina: - tým v řídícím středisku v Coloradu používal anglické měrné jednotky
  - navigační tým v Kalifornii používali metrické
- satelit shoří v atmosféře **cca 300 000 000 USD**
- https://cs.wikipedia.org/wiki/Mars_Climate_Orbiter

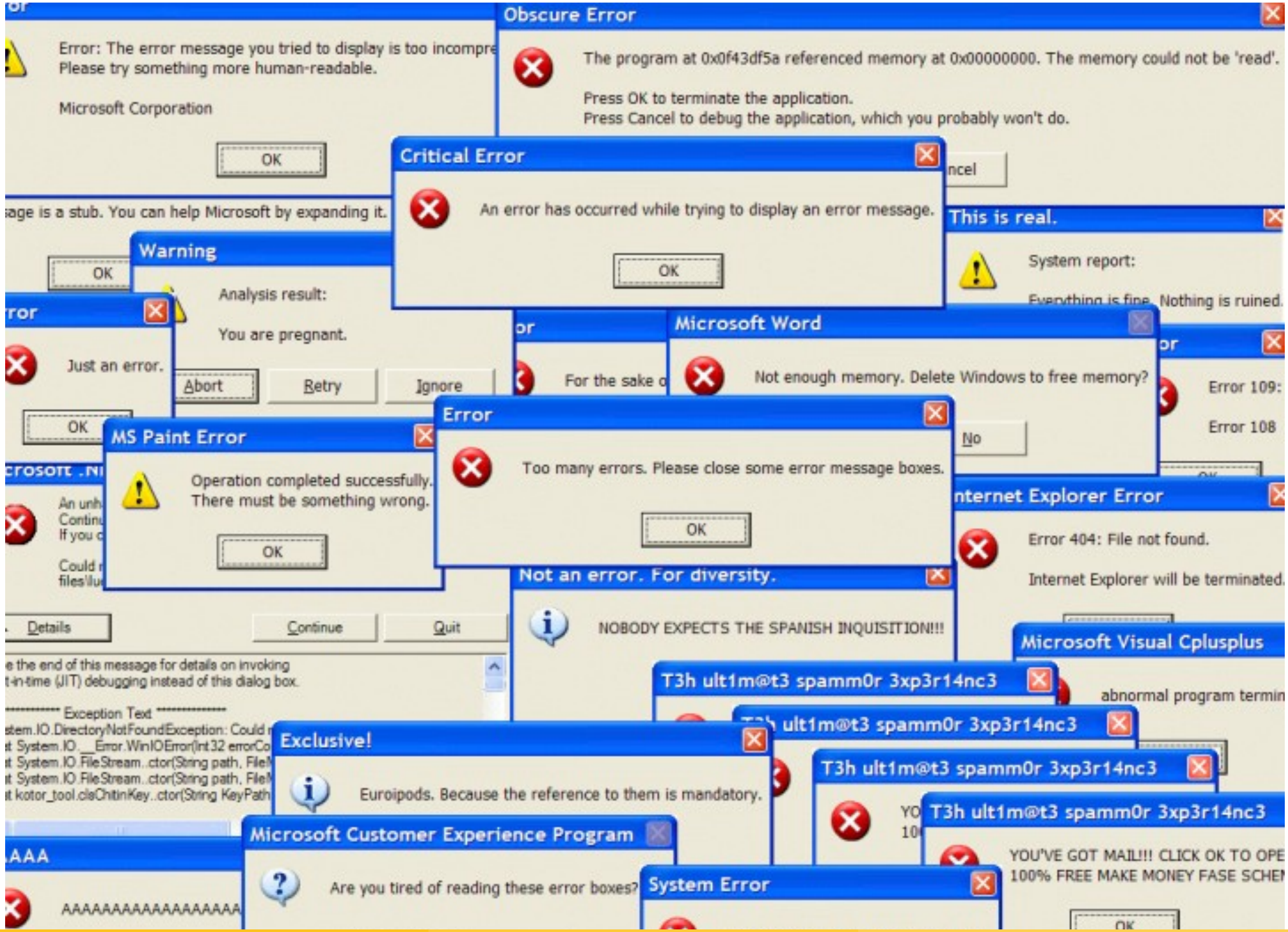
Ariane 5

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

Pentium FDIV bug



- 1994, Professor Thomas Nicely (Lynchburg College, Virginia, USA)
- chyba procesoru Intel P5 Pentium (in floating point unit)
- nesprávné výsledky při dělení desetinných čísel
- způsobeno nevyplněnou tabulkou v matematickém koprocesoru
- ruční ověření například pomocí zadání výpočtu
  1. správná hodnota $\frac{4195835}{3145727} = 1.3338204491362410 02$
     vypočtená hodnota $\frac{4195835}{3145727} = 1.3337390689020375 89$
  2. správná hodnota $\frac{3145727 \times 4195835}{3145727} = 4195835$
     vypočtená hodnota $\frac{3145727 \times 4195835}{3145727} = 4195579$
- celková škoda **cca 500 000 000 USD**
- https://faculty.lynchburg.edu/∼nicely/pentbug/pentbug.html


Obscure Error
The program at 0x0f43df5a referenced memory at 0x00000000. The memory could not be 'read'.
Press OK to terminate the application.
Press Cancel to debug the application, which you probably won't do.
Error: The error message you tried to display is too incompre
Please try something more human-readable.
Microsoft Corporation
OK
Critical Error
An error has occurred while trying to display an error message.
OK
Warning
Analysis result:
You are pregnant.
Abort
Retry
Ignore
This is real.
System report:
Microsoft Word
Not enough memory. Delete Windows to free memory?
No
Error
Too many errors. Please close some error message boxes.
OK
MS Paint Error
Operation completed successfully.
There must be something wrong.
OK
Just an error.
Internet Explorer Error
Error 404: File not found.
Internet Explorer will be terminated.
Not an error. For diversity.
NOBODY EXPECTS THE SPANISH INQUISITION!!!
Details
Continue
Quit
Microsoft Visual Cplusplus
abnormal program termin
T3h ult1m@t3 spamm0r 3xp3r14nc3
Exclusive!
Euroipods. Because the reference to them is mandatory.
Microsoft Customer Experience Program
Are you tired of reading these error boxes?
System Error
YOU'VE GOT MAIL!!! CLICK OK TO OPE
100% FREE MAKE MONEY FASE SCHE
AAAAAAAAAAAAAAAAAA

Děkuji za pozornost.