Chyby, podmíněnost a stabilita

Numerické metody

4. března 2018

FJFI ČVUT v Praze

Úvod

Čísla v počítači

Chyby

Citlivost

Stabilita

Čísla v počítači

Čísla v počítači - Celá čísla

jméno	bity	rozsah	typy
byte	8	-128 až 127	int8_t (C/C++), shortint (Pascal), int8 (Matlab/Python)
short	16	$-32\,768$ až $32\,767$	int (C/C++), Smallint (Pascal), int16 (Matlab/Python)
long	32	$-(2^{31})$ až $2^{31}-1$	long (C/C++), integer (Pascal), int32 (Matlab/Python)

Tabulka 1: Přehled vybraných typů celých čísel

- výhody: přesné výpočty, rychlé
- nevýhody: omezený rozsah, omezené operace

Čísla v počítači - Čísla s plovoucí čárkou

číslo s plovoucí čárkou (*floating-point*), jinak tedy reálné číslo (s konečný počtem platných cifer):

$$\underbrace{\pm}_{\text{znaménko}} \underbrace{1.2345678}_{\text{platné cifry}} \times \underbrace{10^2}_{\text{exponent}}$$

je nutné definovat speciální aritmetiku

Čísla v počítači - Standard IEEE

standard **IEEE** (*Institute of Electrical and Electronics Engineers*) 754 definuje

1. aritmetický formát
2. výměné formáty
3. zaokrouhlovací pravidla
4. operace
5. výjimky

Čísla v počítači - IEEE formát

Každé konečné reálné číslo (tedy číslo s plovoucí čárkou) se zkládá ze tří celých čísel

1. s znaménko (*a sign*) - jednička nebo nula
2. c mantisa (*a significand/a coefficient*)
3. q exponent (*an exponent*)

společně s daným základem b (*a radix*) (2 nebo 10) lze takové číslo zapsat jako

$$(-1)^s \times c \times b^q.$$

Speciálně jsou vyhrazené kombinace pro obě nekonečna ($\pm\infty$) a pro nedefinované číslo (*not a number* - NaN).

Často se provadí tzv. normalizace, kde řekneme, že každá mantisa má na začátku 1. Tedy se musí definovat i speciálně nula (ta není normalizovaná).

Čísla v počítači - Typy

jméno	bity	základ b	mantisa c	exponent q	platné číslice
single	32	2	24	8	$7-8$
double	64	2	53	11	$15-16$

Tabulka 2: Přehled vybraných typů celých čísel

Čísla v počítači - Operace

Operace, které platí i v aritmetice s plovoucí čárkou (pro x, y a z jako reálné čísla s plovoucí čárkou)

- $1 \cdot x = x$,
- $0 \cdot x = 0$,
- $x \cdot y = y \cdot x$,
- $x + x = 2x$,

a které nemusí vždy platit

- $x \cdot \frac{1}{x} = 1$,
- $(x + y) + z = x + (y + z)$.

Čísla v počítači - Strojové epsilon

Strojové epsilon ε je definované jako

- počet platných cifer ale v dvojkové soustavě (těžko představitelné) (2^{-c}, kde c je mantisa)
- rozdíl čísla 1 a nejbližšího většího čísla než 1.

Udává relativní změnu posobě jdoucích čísel.

Rozdělení čísel při základu 2 (při změně exponentu se mění násobek ε)

$$\ldots, 1, 1+\varepsilon, 1+2\varepsilon, \ldots, 2-\varepsilon, 2, 2+2\varepsilon, 2+4\varepsilon, \ldots, 4-2\varepsilon, 4, 4+4\varepsilon, \ldots$$

Strojové epsilon pro

- *single precision* $\varepsilon_{\mathrm{sp}} = 2^{-24} \approx 5.96 \times 10^{-8}$,
- *double precision* $\varepsilon_{\mathrm{dp}} = 2^{-53} \approx 1.11 \times 10^{-16}$.

Čísla v počítači - Odhad strojového epsilonu - algoritmus

1. definujeme `jednicka := 1.0` a `epsilon := 0.1`
2. dokud `jednicka + epsilon > jednicka`
   2.1 zmenšíme `epsilon` dvakrát,
   `epsilon := epsilon/2`
3. výsledný odhad strojového epsilon `2*epsilon` (jelikož poslední `epsilon` už nezměnilo jedničku, tak se musíme vrátit o jedno zpět)

Naprogramujte tento algoritmus a porovnejte výsledek se správným strojovým epsilonem.

Čísla v počítači - Odhad strojového epsilonu - výsledek

Výsledný program je ke stažený zde.

Výstup je následný

```
>> strojove_epsilon
Odhadnute strojove epsilon: 1.7764e-16
Spravne strojove epsilon: 2.2204e-16
```

Vidíme, že odhad je to relativně solidní.

Čísla v počítači - Nejmenší kladné číslo

Druhým zajímavým údajem může být nejmenší kladné číslo.

Každé menší číslo bude zaokrouhleno na nulu.

Kvůli normalizaci bude začínat jedničkou a bude mít největší možný záporný exponent.

Nejmenší číslo pro

- *single precision* $\min_{\mathrm{sp}} = 2^{-150} \approx 1.1 \times 10^{-38}$,
- *double precision* $\min_{\mathrm{dp}} = 2^{-1074} \approx 5.0 \times 10^{-324}$.

Čísla v počítači - Odhad nejmenšího kladného čísla - algoritmus

1. definujeme `mozne_nejmensi := 1.0`
2. dokud `mozne_nejmensi > 0.0`
   - 2.1 zapamatujeme si `nejmensi` jako poslední možné,
     `nejmensi := mozne_nejmensi`
   - 2.2 zmenšíme `mozne_nejmensi` dvakrát,
     `mozne_nejmensi := mozne_nejmensi/2`
3. výsledný odhad nejmenšího kladného čísla `nejmensi`

Naprogramujte tento algoritmus a porovnejte výsledek se správným nejmenším číslem.

Čísla v počítači - Odhad nejmenšího kladného čísla - výsledek

Výsledný program je ke stažený zde.

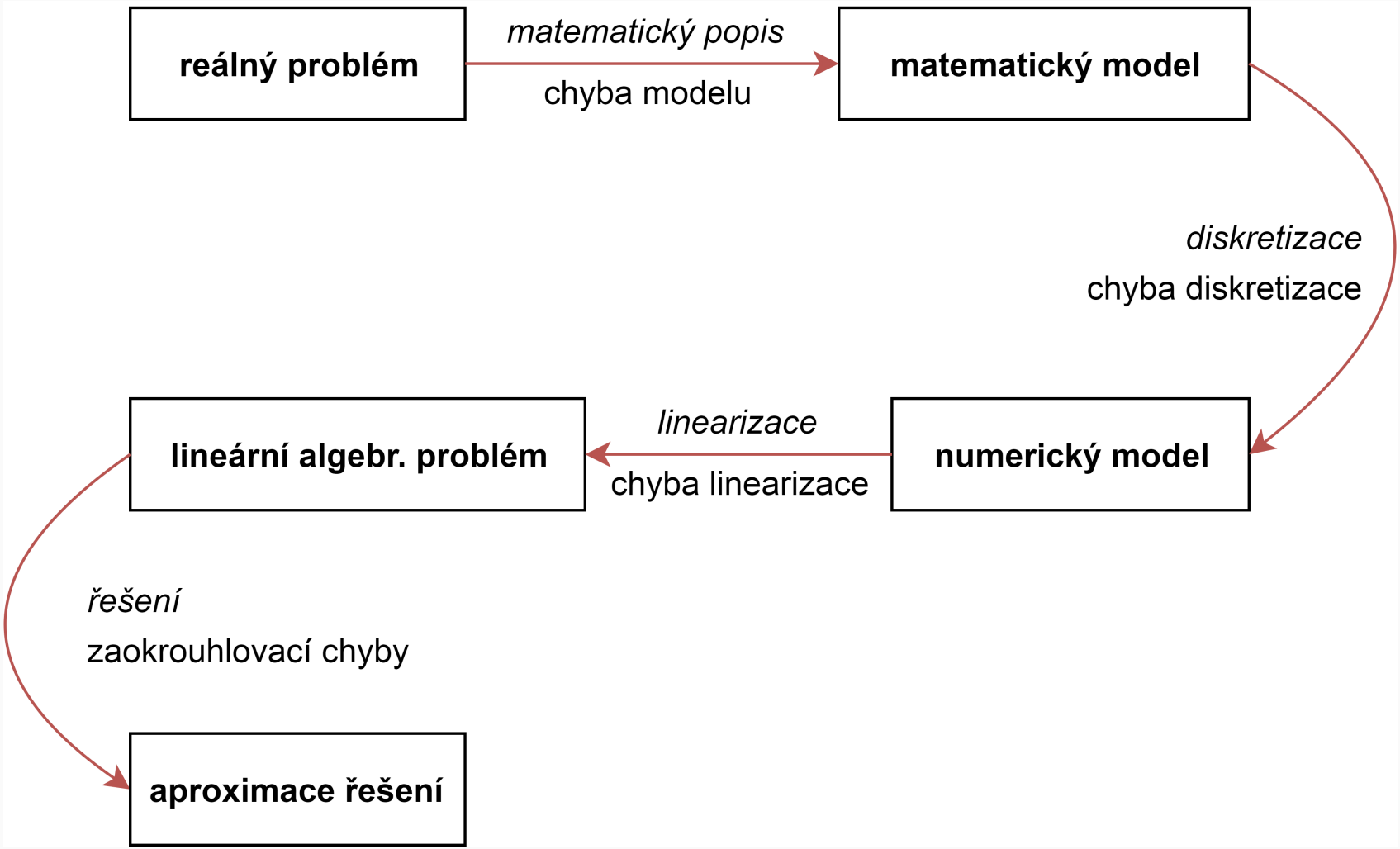
Výstup je následný

```
>> nejmensi_kladne_cislo
Nejmensi zjistene: 4.9407e-324
Nejmensi spravne: 2.2251e-308
```

Námi zjištěné nejmenší kladné číslo a udávané Matlabem se liší. Je ale schopen rozeznat to naše (které nezaokrouhlí na nulu).

Chyby

Chyby - Typy chyb



Chybu při převodu z matematického modelu na numerický označujeme také jako **chybu metody** (*truncation error*).

Chyby - Zaokrouhlovací

Při řešení úlohy na počítači (tedy s čísly s pohyblivou čárkou/s konečnou přesností) dochází k zaokrouhlovacím chybám (*roundoff errors*).

Důležité je, aby se tato chyba nestala v metodě osudnou.

Definujeme u jako polovinu strojové přesnosti, tedy $u := \varepsilon/2$.

Chyby - Operace

Operace $+$, $-$, $*$ a $/$ se provedou přesně a výsledek je zaokrouhlen, tak aby byl opět v daném typu.

Tedy platí

1. $\mathrm{fl}(x \pm y) = (x \pm y)(1 + \varepsilon_1)$, $|\varepsilon_1| < u$,
2. $\mathrm{fl}(x * y) = (x * y)(1 + \varepsilon_2)$, $|\varepsilon_2| < u$,
3. $\mathrm{fl}(x/y) = (x/y)(1 + \varepsilon_3)$, $|\varepsilon_3| < u$,

kde fl je zaokrouhlení do daného typu.

Chyby - Absolutní a relativní

Mějme x přesnou hodnotu a $\tilde{x}$ přibližnou, pak značíme

- **absolutní chybu** $A(x) := |\tilde{x} - x|$ a její horní dohad $a(x) \geq A(x)$,
- **relativní chybu** $R(x) := \frac{|\tilde{x}-x|}{x}$ a její horní odhad $r(x) \geq R(x)$.

Chyby - Chyby operací

Odhad absolutní a relativní chyby sčítání, resp. odčítání je

$$a(x \pm y) = a(x) + a(y) \qquad \Rightarrow \qquad r(x \pm y) = \frac{a(x) + a(y)}{|x \pm y|},$$

pro násobení

$$a(x * y) = |x|a(y) + |y|a(x) \qquad \Rightarrow \qquad r(x * y) = r(x) + r(y)$$

a dělení

$$a(x/y) = \frac{|x|a(y) + |y|a(x)a(x) + a(y)}{y^2} \quad \Rightarrow \quad r(x/y) = r(x)+r(y).$$

Chyby - Kritické operace

Je vidět, že sčítání, násobení a dělení nemůže výrazně změnit relativní chybu.

Oproti tomu, pokud při odčítání budeme pracovat s podobnými čísly, může chyba růst hodně.

Tedy pokud $x \to y$, pak $r(x - y) \to \infty$.

Chyby - Rekurentní vzorec

Definujeme následnou rekurentní posloupnost

$$a_{n+1} := \frac{34}{11}a_n - \frac{3}{11}a_{n-1}$$

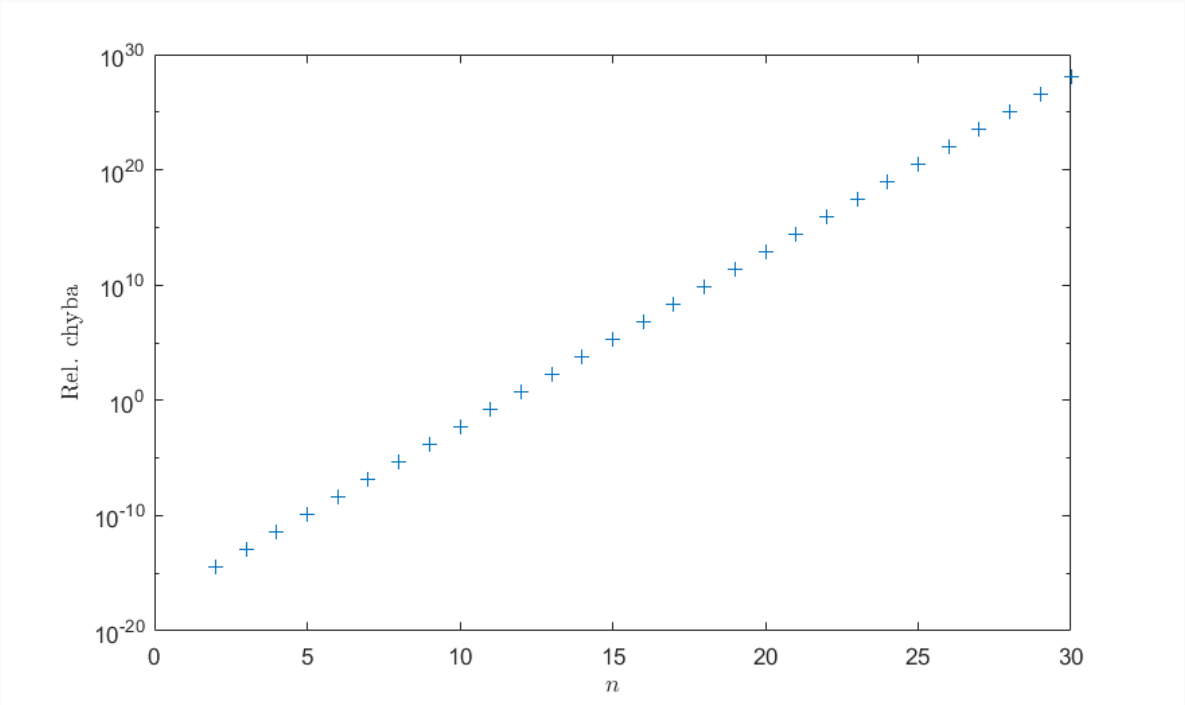
s počátečními hodnotami $a_0 = 1$ a $a_1 = \frac{1}{11}$.

Najděte předpis pro n-tý člen. Porovnejte výsledky rekurentního vzorce a vzorce pro n-tý člen pro $n \in \{5, 10, 20, 30\}$.

Chyby - Rekurentní vzorec - výsledek

Program naleznete zde.

Závislost relativního rozdílu rekurentního vzorce a vzorce pro n-tý člen na n



Vidíme, že chyba roste exponenciálně (jelikož osa y je logaritmická). To je dáno právě odčítáním podobných čísel.

Chyby - Závislost chyby na délce kroku

Nyní zkusíme spočítat derivaci funkce

$$f(x) := \sin(x).$$

Tedy analyticky

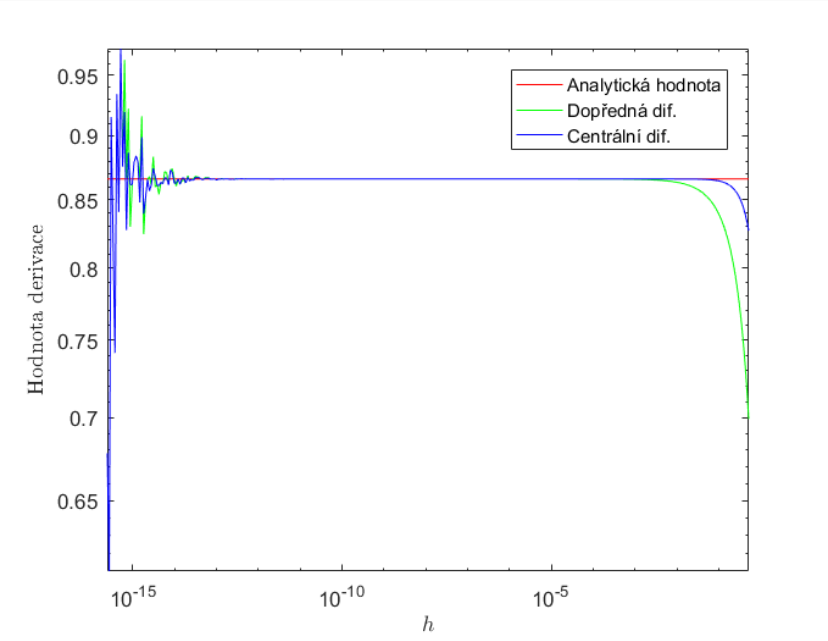
$$f'(x) := \cos(x).$$

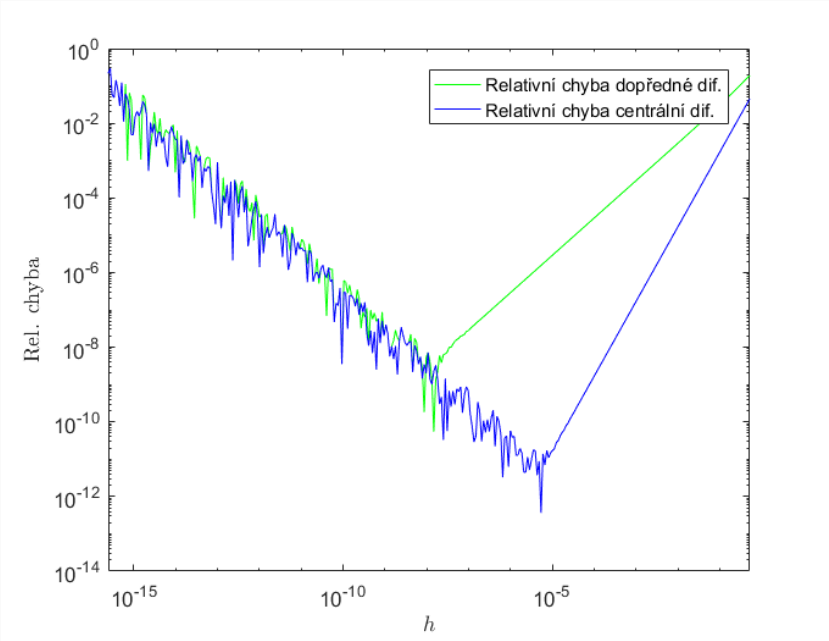
Použijeme dopřednou a centrální diferenci a vykreslíme relativní chybu v závislosti na délce kroku h.

Chyby - Závislost chyby na délce kroku - výsledek

Program naleznete zde.

Chování dopředné a centrální diference v závislosti na délce kroku h





Vidíme, že do určité hodnoty h je centrální diference (tedy metoda vyššího řádu) lepší, ale jakmile se dostane do kriticky krátkého h, chyba se začne zvětšovat.

Vlevo se jedná o chybu zaokrouhlovací, napravo o chybu metody.

Citlivost

Citlivost - Definice

Citlivost (jinak také podmíněnost) je vlastnost dané **úlohy** popisující vliv malé změny vstupních dat na změnu řešení.

Matematicky: Mějme dvě vstupní data x a $\tilde{x}$ a k nim příslušné řešení y, resp. $\tilde{y}$, pak pokud je číslo podmíněnosti úlohy dané

$$C_p = \frac{\frac{|y-\tilde{y}|}{y}}{\frac{|x-\tilde{x}|}{x}}$$

musí malé, úloha je dobře podmíněná, v opačném případě je špatně podmíněná.

Citlivost - Dobře a špatně podmíněná úloha

Tedy pokud $C_p \sim 1$ je úloha dobře podmíněná, pokud $C_p > 100$ úloha je špatně podmíněná.

Často i řešíme špatně podmíněné úlohy, je ale potřeba volit speciální metody, které jsou odolné.

Pro $C_p > \varepsilon^{-1}$ není úloha řešitelná v dané přesnosti.

Stabilita

Stabilita - Definice

Stabilní metoda je taková, kde chyba roste s počtem kroků maximálně lineárně.

U nestabilní metody dojde k akumulaci chyb, až dojde ke katastrofální ztrátě přesnosti.

Nestabilita může být dána jak zaokrouhlovací chybou tak i chybou metody. Často se vyskytuje při řešení diferenciálních rovnic.

Stabilita - Pád meteoritu

Meteorit, který padá skrze atmosféru, je brzděn exponenciálně. Jeho rychlost je dána diferenciální rovnicí

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -v(t).$$

Její analytické řešení je

$$v(t) = C\exp(-t),$$

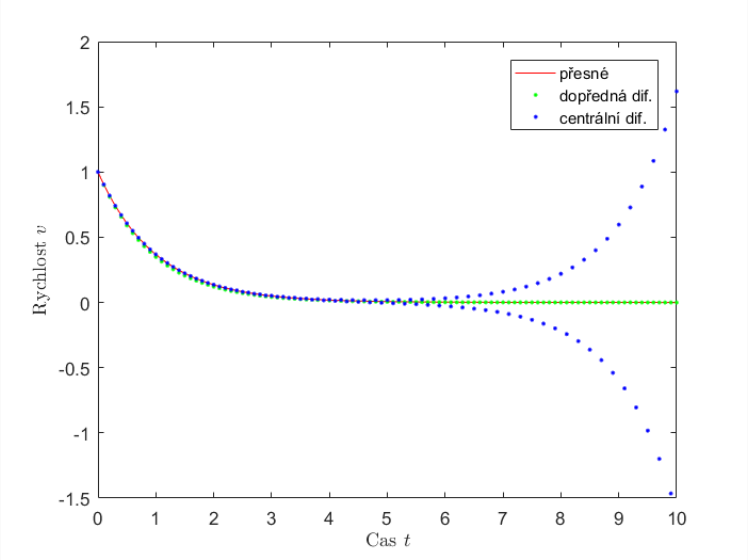
kde C je konstanta závislá pouze od počáteční podmínky.

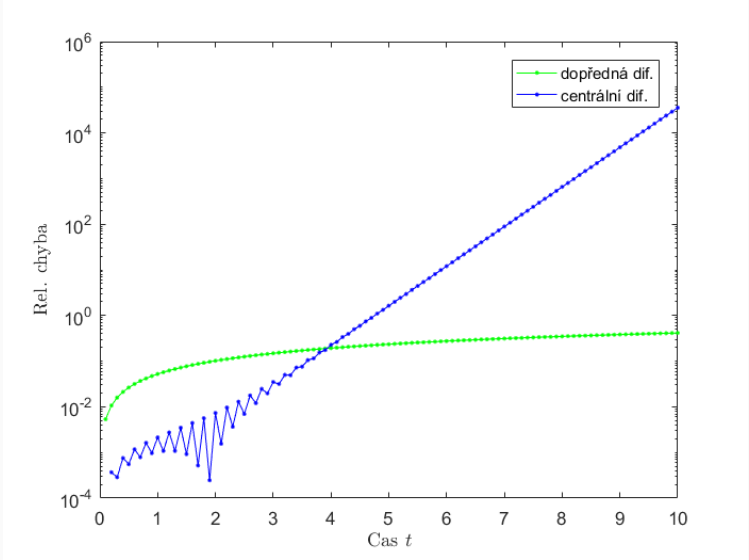
Podívejte se na řešení této rovnice numericky pomocí dopředné a centrální diference s počátečními podmínkou $v(t = 0) = 1$.

Stabilita - Pád meteoritu - výsledek

Program naleznete zde.

Porovnání chování centrální diference oproti dopředné v průběhu času (řešení rovnice)





Vidíme, že metoda centrální diference je nestabilní a její chyba časem roste exponenciálně (opět je osa y v pravém grafu logaritmická).

Oproti tomu dopředná diference je stabilní, i když je na začátku horší než metoda první.

Navíc

Navíc - Důkaz neplatnosti asociativity

Mějme $n \in \mathbb{N}$ a řekněme, že chceme sečíst následující řádu

$$S_n = \sum_{i=0}^{n} \left(\frac{10}{11}\right)^i.$$

Pokud by tato aritmetika byla asociativní, nezáleželo by, jestli sčítáme sestupně nebo vzestupně.

Naprogramujte vzestupný i sestupný součet a výsledek porovnejte pro $n \in \{10, 100, 200\}$. (Zkuste to pro *single* i *double* přesnost).

Navíc - Důkaz neplatnosti asociativity - výsledek

Program v *double precision* naleznete zde. Výstup pro $n = 200$

```
>> neplatnost_asociativity_double
Vzestupny soucet (n=200): 10.99999994734217
Sestupny soucet (n=200):  10.99999994734216
```

Vidíme, že rozdíl je až na 16. pozici (tedy, která už nemusí být směrodatná). Tedy tento algoritmus je v *double precision* asociativní.

Program v *single precision* naleznete zde. Výstup pro $n = 200$

```
>> neplatnost_asociativity
Vzestupny soucet (n=200): 10.99999
Sestupny soucet (n=200): 11
```

Oproti tomu v *single precision* je rozdíl už na druhé platné cifře, tedy tady asociativita neplatí a záleží jak sčítáme.

Navíc - Výpočet funkční hodnoty

Máme funkci

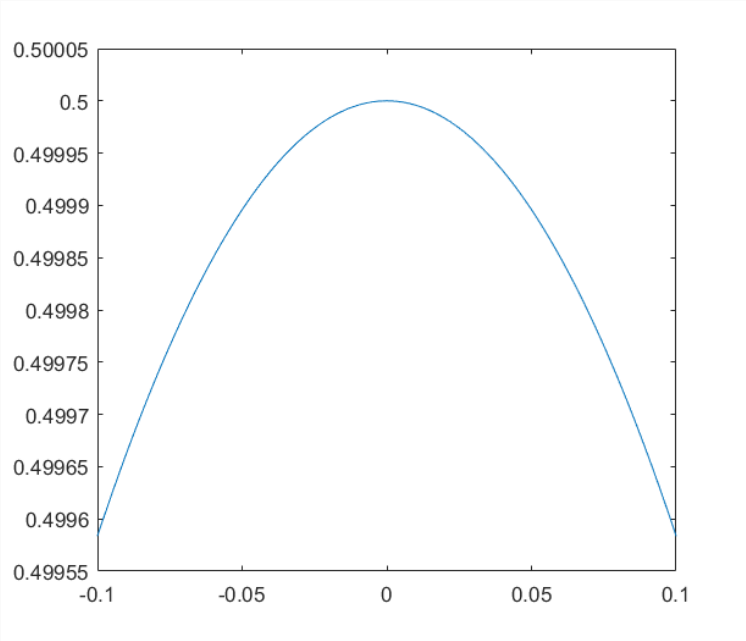
$$f(x) := \frac{1 - \cos(x)}{x^2}.$$

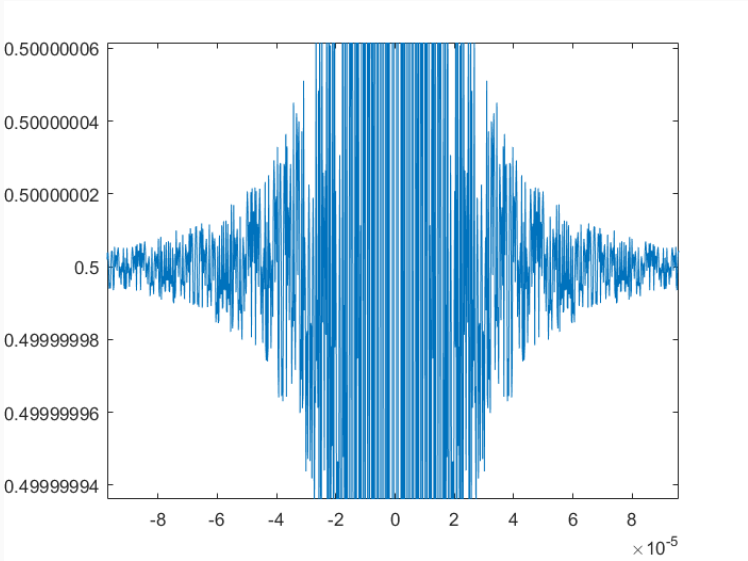
Vykreslíme chování takovéto funkce v okolí 0.

Navíc - Výpočet funkční hodnoty - výsledek

Program naleznete zde.

Vykreslení funkce na intervalu $\langle -0.1; 0.1 \rangle$ pro tisíc bodů (levý obrázek) a pro milión bodů (pravý obrázek) (resp. jeho detail)





Vidíme, že okolo nuly, kde je funkční hodnota $\cos(x)$ blízká jedničce, dochází k nestabilitě, jelikož odčítáme dvě podobná čísla.