

PA081: Programování numerických výpočtů

Aleš Křenek

jaro 2014

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

O čem to bude

O čem to bude

Čísla v
plovoucí
řádové čáře

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ zajímá nás chování skutečného světa
 - ▶ problémy přírodovědné, technické, humanitní ...
- ▶ popisujeme matematickými prostředky
 - ▶ zejména pomocí reálných čísel
 - ▶ umělý aparát, leckdy zcela neodpovídá skutečnosti
 - ▶ za staletí celkem zvládnutý, vyučovaný od základní školy, obecně přijímaný

O čem to bude

- ▶ zákonitosti zkoumaných systémů typicky vyjádřeny rovnicemi
- ▶ zajímavé systémy → složité rovnice
 - ▶ reprezentativní příklad – Schrödingerova rovnice

$$i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{x}, t) = \hat{H} \Psi(\mathbf{x}, t)$$

- ▶ analyticky řešitelná jen pro triviální případy
 - ▶ izolovaná částice, částice v potenciálové jámě, ..., atom vodíku
 - ▶ i tak je řešení dost komplikované
 - ▶ numerické řešení je jediný realisticky možný přístup

O čem to bude

- ▶ numerická matematika
 - ▶ řešení matematicky formulovaného problému aritmetickými prostředky
 - ▶ zpravidla algoritmický postup - numerická metoda
- ▶ disciplína podstatně starší než počítače

O čem to bude

- ▶ numerická matematika
 - ▶ řešení matematicky formulovaného problému aritmetickými prostředky
 - ▶ zpravidla algoritmický postup - numerická metoda
- ▶ disciplína podstatně starší než počítače
- ▶ metody jsou známé, naprogramujeme je a je hotovo
- ▶ ne tak docela

(De)motivační příklad

- ▶ uměle zkonstruovaný, ilustruje podstatu problému
- ▶ pro dané n vypočítejte integrál

$$E_n = \int_0^1 x^n e^{x-1} dx$$

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

(De)motivační příklad

- ▶ uměle zkonstruovaný, ilustruje podstatu problému
- ▶ pro dané n vypočítejte integrál

$$E_n = \int_0^1 x^n e^{x-1} dx$$

- ▶ očekávané vlastnosti

- ▶ $E_0 = [e^{x-1}]_0^1 = 1 - \frac{1}{e}$
- ▶ pro $0 \leq x \leq 1$ platí $x^n \geq 0$ a $0 < e^{x-1} \leq 1$,
tedy $E_n > 0$
- ▶ podobně

$$E_n \leq \int_0^1 x^n dx = \frac{1}{n+1} \quad \text{a tedy} \quad \lim_{n \rightarrow \infty} E_n = 0$$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

(De)motivační příklad

- ▶ pro numerický výpočet lze transformovat na rekurentní posloupnost
- ▶ integrací per-partes $u = x^n$, $dv = e^{x-1} dx$

$$E_n = \left[x^n e^{x-1} \right]_0^1 - \int_0^1 n x^{n-1} e^{x-1} dx = 1 - n E_{n-1}$$

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

(De)motivační příklad

- ▶ pro numerický výpočet lze transformovat na rekurentní posloupnost
- ▶ integrací per-partes $u = x^n$, $dv = e^{x-1} dx$

$$E_n = \left[x^n e^{x-1} \right]_0^1 - \int_0^1 n x^{n-1} e^{x-1} dx = 1 - n E_{n-1}$$

```
float    e = 2.7182818;  
float    E[20];  
int      n;  
  
E[0] = 1.0-1.0/e;  
for (n=1; n<20; n++) {  
    E[n] = 1.0 - n * E[n-1];  
    printf("%d: %f\n",n,E[n]);  
}
```

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

(De)motivační příklad

1: 0.367879
2: 0.264241
3: 0.207277
4: 0.170893
5: 0.145534
6: 0.126796
7: 0.112430
8: 0.100563
9: 0.094933
10: 0.050674
11: 0.442581
12: -4.310974
13: 57.042664
14: -797.597290
15: 11964.958984
16: -191438.343750
17: 3254452.750000
18: -58580148.000000
19: 1113022848.000000

(De)motivační příklad

- ▶ co je špatně?
- ▶ pracujeme s konečnou reprezentací reálného čísla
 - ▶ společný problém ručního i strojového zpracování
 - ▶ v počítači daleko plíživější podoba (nevidíme mezivýsledky)

(De)motivační příklad

- ▶ co je špatně?
- ▶ pracujeme s konečnou reprezentací reálného čísla
 - ▶ společný problém ručního i strojového zpracování
 - ▶ v počítači daleko plíživější podoba (nevidíme mezivýsledky)
- ▶ v proměnné $E[n]$ není uloženo přesně E_n
už na začátku zatíženo chybou, $E[0] = E_0 + \epsilon$

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

(De)motivační příklad

- ▶ co je špatně?
- ▶ pracujeme s konečnou reprezentací reálného čísla
 - ▶ společný problém ručního i strojového zpracování
 - ▶ v počítači daleko plíživější podoba (nevidíme mezivýsledky)
- ▶ v proměnné $E[n]$ není uloženo přesně E_n
už na začátku zatíženo chybou, $E[0] = E_0 + \epsilon$
- ▶ i při zcela přesném výpočtu:

$$E[1] = 1 - E[0] = 1 - E_0 - \epsilon = E_1 - \epsilon$$

$$E[2] = 1 - 2E[1] = 1 - 2E_1 + 2\epsilon = E_2 + 2\epsilon$$

$$E[3] = 1 - 3E[2] = 1 - 3E_2 - 3(2\epsilon) = E_3 - 3(2\epsilon)$$

$$\vdots$$

$$E[n] = \dots = E_n + (-1)^n n! \epsilon$$

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

O čem to tedy bude

- ▶ představení numerických metod pro řešení vybraných problémů
 - ▶ pragmaticky, bez rozsáhlé teoretické analýzy, okrajových podmínek atd.
- ▶ formulace přiměřeně přesného a efektivního algoritmu
 - ▶ matematicky korektní metoda nestačí
 - ▶ pro různá použití jsou vhodné různé alternativy
- ▶ použití na smysluplném příkladu

Čísla v plovoucí řádové čárce

- ▶ standard IEEE 754
 - ▶ vychází návrhu reprezentace čísel v koprocesoru Intel 8087
- ▶ základní formát $\pm 1.\text{mmmmmmmmmm} \dots \times 2^{\pm ee\dots}$
 - ▶ znaménko mantisy (1 bit)
 - ▶ mantisa $1.\text{mmmmmmmmmm}$
 - ▶ binárně, absolutní hodnota
 - ▶ číslo v intervalu $[1, 2)$ v dané přesnosti
 - ▶ nekonečná množina pokryta konečným počtem hodnot
 - ▶ základní zdroj nepřesnosti
 - ▶ exponent
 - ▶ binární číslo v rozsahu 1 až např. 254
 - ▶ znamená hodnoty exponentu -126 až +127
 - ▶ speciální význam hodnot 00...00 a 11...11 - kódování $\pm 0, \pm \infty, \pm \text{NaN}$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Čísla v plovoucí řádové čárce

- ▶ nejběžnější typy - velikosti v bitech a rozsah exponentu

typ	exp.	mantisa	celkem	rozsah exp.
Single (float)	8	23	32	127
Double	11	52	64	1023
Quad (long double)	15	112	128	16383

- ▶ tj. např. největší číslo typu `float` je $2^{127} \doteq 10^{38}$
- ▶ nejmenší nenulové $2^{-126} \doteq 10^{-37}$
- ▶ relativní chyba je omezena $2^{-25} \doteq 10^{-8}$
 - ▶ tedy počítáme na cca. 8 platných cifer

Problémy konečné reprezentace

Ztráta přesnosti sčítání

- ▶ pro zjednodušení v desítkové soustavě a na 3 platné cifry mantisy
- ▶ sečtěme $1.23e3$ a $1.00e0$
 - ▶ nutné sjednocení exponentů:
 $1.23e3 + 0.001e3 = 1.231e3 \doteq 1.23e3$
 - ▶ v rámci dané přesnosti korektní výsledek

Problémy konečné reprezentace

Ztráta přesnosti sčítání

- ▶ pro zjednodušení v desítkové soustavě a na 3 platné cifry mantisy
- ▶ sečtěme $1.23e3$ a $1.00e0$
 - ▶ nutné sjednocení exponentů:
 $1.23e3 + 0.001e3 = 1.231e3 \doteq 1.23e3$
 - ▶ v rámci dané přesnosti korektní výsledek
- ▶ k $1.23e3$ desetkrát přičtěme $1.00e0$
 - ▶ opakované aplikování předchozího postupu dává opět $1.23e3$
 - ▶ nemusí to být to, co jsme chtěli
 - ▶ $1.23e3 + (1.00e0 + \dots + 1.00e0) = 1.24e3$

Problémy konečné reprezentace

Ztráta přesnosti sčítání

- ▶ pro zjednodušení v desítkové soustavě a na 3 platné cifry mantisy
- ▶ sečteme $1.23e3$ a $1.00e0$
 - ▶ nutné sjednocení exponentů:
 $1.23e3 + 0.001e3 = 1.231e3 \doteq 1.23e3$
 - ▶ v rámci dané přesnosti korektní výsledek
- ▶ k $1.23e3$ desetkrát přičteme $1.00e0$
 - ▶ opakované aplikování předchozího postupu dává opět $1.23e3$
 - ▶ nemusí to být to, co jsme chtěli
 - ▶ $1.23e3 + (1.00e0 + \dots + 1.00e0) = 1.24e3$
- ▶ operace v plovoucí řádové čárce nejsou asociativní
 - ▶ ani tam, kde jsme na to z algebry zvyklí

Problémy konečné reprezentace

Ztráta přesnosti sčítání

Čekáme-li kumulativní efekt, sčítání a odčítání je třeba provést nejprve na číslech se srovnatelným exponentem.

Problémy konečné reprezentace

Katastrofální zrušení

- ▶ odečtení dvou vzájemně blízkých čísel vede k výrazné ztrátě přesnosti

Problémy konečné reprezentace

Katastrofální zrušení

- ▶ odečtení dvou vzájemně blízkých čísel vede k výrazné ztrátě přesnosti
- ▶ rozdíl dvou měření s přesností na 3 platné cifry:

$$1.23 - 1.22$$

- ▶ binární reprezentace: 1.0011101 a 1.0011100
 - ▶ zatížena chybami 0.3% a 0.1%

Problémy konečné reprezentace

Katastrofální zrušení

- ▶ odečtení dvou vzájemně blízkých čísel vede k výrazné ztrátě přesnosti
- ▶ rozdíl dvou měření s přesností na 3 platné cifry:

$$1.23 - 1.22$$

- ▶ binární reprezentace: 1.0011101 a 1.0011100
 - ▶ zatížena chybami 0.3% a 0.1%
- ▶ odečtením dostaneme binárně 0.0000001, tj. 0.0078125
 - ▶ správný výsledek byl 0.01, relativní chyba **22%**
 - ▶ navíc působí dojmem **falešné přesnosti**

Problémy konečné reprezentace

Katastrofální zrušení

- ▶ odečtení dvou vzájemně blízkých čísel vede k výrazné ztrátě přesnosti
- ▶ rozdíl dvou měření s přesností na 3 platné cifry:

$$1.23 - 1.22$$

- ▶ binární reprezentace: 1.0011101 a 1.0011100
 - ▶ zatížena chybami 0.3% a 0.1%
- ▶ odečtením dostaneme binárně 0.0000001, tj. 0.0078125
 - ▶ správný výsledek byl 0.01, relativní chyba **22%**
 - ▶ navíc působí dojmem **falešné přesnosti**

Vyhňeme se odčítání dvou blízkých čísel. Je-li to i tak nezbytné, počítejme s výsledkem zatíženým potenciálně velkou chybou.

Problémy konečné reprezentace

Násobení a dělení

- ▶ samo o sobě nezanáší významnou chybu
- ▶ zachovává existující chybu

Problémy konečné reprezentace

Násobení a dělení

- ▶ samo o sobě nezanáší významnou chybu
- ▶ zachovává existující chybu
- ▶ příklad: $(a + b)^2$ pro $a = 1.23$, $b = 0.0155$, na 3 cifry
- ▶ přesný výsledek je 1.55127025

Problémy konečné reprezentace

Násobení a dělení

- ▶ samo o sobě nezanáší významnou chybu
- ▶ zachovává existující chybu
- ▶ příklad: $(a + b)^2$ pro $a = 1.23$, $b = 0.0155$, na 3 cifry
- ▶ přesný výsledek je 1.55127025
- ▶ přímý výpočet na 3 platné cifry:

$$a + b \doteq 1.23 + 0.02 = 1.25$$

- ▶ tedy $(a + b)^2 = 1.56$, chyba 0.56%
 - ▶ není to tak zlé, ale může být lepší

Problémy konečné reprezentace

Násobení a dělení

- ▶ po transformaci $(a + b)^2 = a^2 + 2ab + b^2$:

$$\begin{aligned} 1.51 + 2 \times 0.0191 + 0.000240 &= \\ 1.51 + 0.0382 + 0.000240 &\doteq 1.55 \end{aligned}$$

- ▶ chyba 0.082 %, tedy téměř o řád menší
- ▶ za cenu 6 aritmetických operací místo 2
- ▶ bude 3× pomalejší

Problémy konečné reprezentace

Násobení a dělení

- ▶ po transformaci $(a + b)^2 = a^2 + 2ab + b^2$:

$$\begin{aligned} 1.51 + 2 \times 0.0191 + 0.000240 &= \\ 1.51 + 0.0382 + 0.000240 &\doteq 1.55 \end{aligned}$$

- ▶ chyba 0.082 %, tedy téměř o řád menší
- ▶ za cenu 6 aritmetických operací místo 2
- ▶ bude 3× pomalejší
... uvidíme

Měření rychlosti výpočtu

Naivní přístup

- ▶ POSIX volání `gettimeofday()`

```
gettimeofday(&start, NULL);  
c = a + b;  
c *= c;  
gettimeofday(&stop, NULL);
```

- ▶ současné CPU 2-3 GHz
- ▶ 1 aritmetická operace \sim 1-10 ns
- ▶ nemáme tak přesné hodiny

Měření rychlosti výpočtu

Zopakujeme v cyklu

- ▶ opakování 10^9 × navýší čas na měřitelnou ~ 1 s

```
gettimeofday(&start, NULL);  
for (i=0; i<1000000000; i++) {  
    c = a + b;  
    c *= c;  
}  
gettimeofday(&stop, NULL);
```

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Měření rychlosti výpočtu

Zopakujeme v cyklu

- ▶ opakování 10^9 × navýší čas na měřitelnou ~ 1 s

```
gettimeofday(&start, NULL);  
for (i=0; i<1000000000; i++) {  
    c = a + b;  
    c *= c;  
}  
gettimeofday(&stop, NULL);
```

- ▶ počítá podstatně rychleji
 - ▶ trochu podezřelé

Měření rychlosti výpočtu

Zopakujeme v cyklu

- ▶ optimalizující kompilátor `gcc -O3 -funroll-loops`

```
movss    44(%rsp), %xmm0
xorl     %eax, %eax
addss   40(%rsp), %xmm0
mulss   %xmm0, %xmm0

.L2:
addl     $8, %eax
cmpl    $1000000000, %eax
jne     .L2
movss   %xmm0, 36(%rsp)
```

- ▶ v těle cyklu se nic nepočítá
- ▶ optimalizaci obecně chceme
 - ▶ použití registrů pro proměnné, max. využití FPU, ...
- ▶ eliminace zbytečného opakování je příliš

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Měření rychlosti výpočtu

Zopakujeme v cyklu

- ▶ brzdící kód
 - ▶ uměle vložený do těla cyklu
 - ▶ za běhu programu se nevyvolá - nebrzdí doopravdy
 - ▶ zabrání příliš agresivní optimalizaci cyklu

```
nikdy = (strlen(argv[0]) == 0);
for (i=0; i<1000000000; i++) {
    c = a + b;
    c *= c;
    if (nikdy) brzda(&nikdy,&a,&b,&c);
}
```

- ▶ kompilátor musí předpokládat:
 - ▶ proměnná `nikdy` nemusí být 0
 - ▶ funkce `brzda()` má vliv na odkazované proměnné

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Měření rychlosti výpočtu

Jak to dopadlo

- ▶ Intel i7-2600 3.4 GHz

vzorec	čas	Mflop/s	Mcyklů/s
$(a + b)^2$	0.61	3278	1639
$a^2 + 2ab + b^2$	0.99	6060	1010

- ▶ rozvinutý výpočet jen $1.6\times$ pomalejší
 - ▶ vnitřní paralelismus procesoru
 - ▶ více FPU jednotek
 - ▶ pipelining
 - ▶ spekulativní výpočet větví programu
 - ▶ stále dosahujeme jen 25% FPU výkonu jednoho jádra CPU

Problémy konečné reprezentace

Násobení a dělení

Snažme se vzorce transformovat tak, aby se nejdříve násobilo/dělilo, pak teprve sčítalo/odčítalo. Je šance na přesnější výsledek, dopad na výkon nemusí být významný.

Problémy konečné reprezentace

Přetečení a podtečení

- ▶ násobení dvou velkých čísel může vést k nerepresentovatelnému exponentu

$$1.2e30 \times 1.2e30 = 1.44e61$$

- ▶ typ `float` umí exponent $[-37, 38]$
- ▶ výsledek operace už je ∞

Problémy konečné reprezentace

Přetečení a podtečení

- ▶ násobení dvou velkých čísel může vést k nerepresentovatelnému exponentu

$$1.2e30 \times 1.2e30 = 1.44e61$$

- ▶ typ `float` umí exponent $[-37, 38]$
- ▶ výsledek operace už je ∞
- ▶ podobně násobení dvou malých čísel vede k podtečení
 - ▶ podle nastavení aritmetiky je výsledek ± 0 nebo denormalizované číslo

$$0.000000\text{mmm} \times 10^{-37}$$

- ▶ další výpočty nepřesné a navíc citelně pomalejší

Problémy konečné reprezentace

Přetečení a podtečení

Při násobení a dělení více čísel uspořádejme posloupnost operací, abychom nedělili velmi malé číslo velkým, velké malým, a nenásobili vzájemně velká nebo malá čísla.

Problémy konečné reprezentace

Porovnávání

- ▶ nevinný výpočet

```
float a=1.505, b1=1.315, b2=1.695,  
      c=a+a, d=b1+b2;
```

```
printf("%f %f %s\n",c,d,c == d ?  
      "jsou stejna" : "jsou ruzna");
```


Problémy konečné reprezentace

Porovnávání

- ▶ nevinný výpočet

```
float a=1.505, b1=1.315, b2=1.695,  
      c=a+a, d=b1+b2;
```

```
printf("%f %f %s\n",c,d,c == d ?  
      "jsou stejna" : "jsou ruzna");
```

- ▶ má překvapivý výstup

3.010000 3.010000 jsou ruzna

Problémy konečné reprezentace

Porovnávání

- ▶ nevinný výpočet

```
float a=1.505, b1=1.315, b2=1.695,  
      c=a+a, d=b1+b2;
```

```
printf("%f %f %s\n",c,d,c == d ?  
      "jsou stejna" : "jsou rozna");
```

- ▶ má překvapivý výstup

3.010000 3.010000 jsou rozna

- ▶ přesnější formát výpisu "%15.12f":

3.009999990463 3.010000228882 jsou rozna

Problémy konečné reprezentace

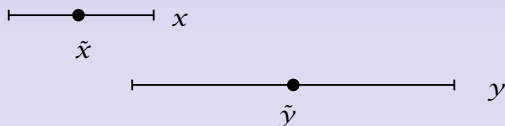
Porovnávání

- ▶ operátor `==` na reálných číslech nemá valný smysl
 - ▶ téměř vždy je zasažen chybou předchozího výpočtu
- ▶ místo něj test na dostatečnou blízkost
$$\text{fabs}(c-d) < \text{EPSILON}$$
- ▶ stanovení toleranční konstanty zpravidla empiricky
 - ▶ různá pro hodnoty $1e-15$ a $1e30$
 - ▶ silně záleží na dané úloze
 - ▶ ovlivněna i každým konkrétním výpočtem

Problémy konečné reprezentace

Porovnávání

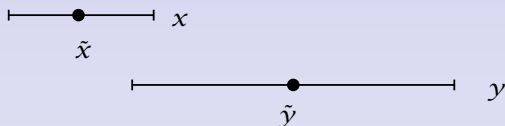
- ▶ porovnávání $< a >$ trpí stejným problémem
 - ▶ záludnější podoba a komplikovanější řešení
- ▶ chci porovnat přesné hodnoty x a y
 - ▶ znám jen přibližná (spočtená) $\tilde{x} = x \pm \epsilon_x$, $\tilde{y} = y \pm \epsilon_y$:



Problémy konečné reprezentace

Porovnávání

- ▶ porovnávání $<$ a $>$ trpí stejným problémem
 - ▶ záludnější podoba a komplikovanější řešení
- ▶ chci porovnat přesné hodnoty x a y
 - ▶ znám jen přibližná (spočtená) $\tilde{x} = x \pm \epsilon_x$, $\tilde{y} = y \pm \epsilon_y$:

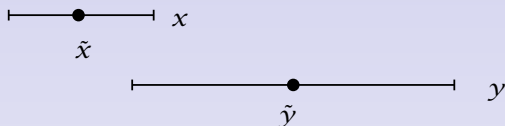


- ▶ opatrný („miserly“) přístup
 - ▶ intervaly se i částečně překrývají \Rightarrow tvrdíme $x = y$
 - ▶ jinak porovnáme $\tilde{x} \leq \tilde{y}$

Problémy konečné reprezentace

Porovnávání

- ▶ porovnávání $<$ a $>$ trpí stejným problémem
 - ▶ záludnější podoba a komplikovanější řešení
- ▶ chci porovnat přesné hodnoty x a y
 - ▶ znám jen přibližná (spočtená) $\tilde{x} = x \pm \epsilon_x$, $\tilde{y} = y \pm \epsilon_y$:



- ▶ **opatrný** („miserly“) přístup
 - ▶ intervaly se i částečně překrývají \Rightarrow tvrdíme $x = y$
 - ▶ jinak porovnáme $\tilde{x} \leq \tilde{y}$
- ▶ **hladový** („eager“) přístup
 - ▶ chceme vědět, že mohlo nastat $x > y$
 - ▶ porovnáváme $\tilde{x} + \epsilon_x > \tilde{y} - \epsilon_y$.

Problémy konečné reprezentace

Porovnávání

Na rovnost porovnávejme vždy s tolerancí.

U porovnání na nerovnost si uvědomme, co chceme vědět.

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Příklad

Kvadratická rovnice

- ▶ rovnice tvaru
- ▶ školní vzorec

$$ax^2 + bx + c = 0$$

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

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Příklad

Kvadratická rovnice

- ▶ rovnice tvaru

$$ax^2 + bx + c = 0$$

- ▶ školní vzorec

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

- ▶ je-li $b^2 \gg ac$, počítáme x_+ z rozdílu dvou velmi blízkých čísel
 - ▶ problém katastrofálního zrušení
- ▶ „ \gg “ neznamená až příliš velký rozdíl koeficientů
 - ▶ typ `float` - přesnost na 7-8 dekadických číslic
 - ▶ při $a = 1$ tedy stačí $b \sim 10^3 c$

Příklad

Kvadratické rovnice

- ▶ konkrétně pro $b = 2000$, $c = 1$
- ▶ „přesné“ hodnoty
 - ▶ $\sqrt{D} = 1999.9989999997499998749999218749453 \dots$
 - ▶ $x_+ = -.00050000012500006250003906252734377 \dots$
 - ▶ $x_- = -1999.9994999998749999374999609374 \dots$
- ▶ pro `float`
 - ▶ $\sqrt{D} = 1999.999$
 - ▶ $x_+ = -0.00048828125$, chyba 2.5%
 - ▶ $x_- = -1999.9995$, chyba odpovídá přesnosti typu

Příklad

Kvadratické rovnice

- ▶ bezproblémové řešení pro x_+
- ▶ vlastnosti kořenů rovnice:

$$ax^2 + bx + c = (x - x_+)(x - x_-) \quad \text{tedy} \quad c = x_+x_-$$

a lze počítat $x_+ = c/x_-$;

- ▶ pro předchozí příklad dostáváme
 - ▶ $x_+ = -0.00050000014$, chyba odpovídá přesnosti typu
- ▶ analogicky pro $b < 0$ je nepřesné x_- , vypočte se symetricky

Příklad

Kvadratické rovnice

- ▶ bezproblémové řešení pro x_+
- ▶ vlastnosti kořenů rovnice:

$$ax^2 + bx + c = (x - x_+)(x - x_-) \quad \text{tedy} \quad c = x_+x_-$$

a lze počítat $x_+ = c/x_-$;

- ▶ pro předchozí příklad dostáváme
 - ▶ $x_+ = -0.00050000014$, chyba odpovídá přesnosti typu
- ▶ analogicky pro $b < 0$ je nepřesné x_- , vypočte se symetricky
- ▶ i v triviálním výpočtu může vzniknout problém
- ▶ řešení může být docela jednoduché

Numerická stabilita

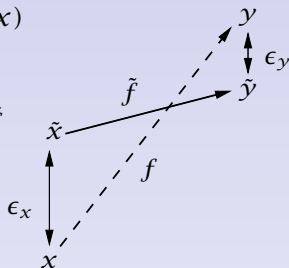
(Pseudo)definice

- ▶ „metoda počítá sice špatně, ale jenom trochu špatně“
 - ▶ žádoucí a přitom realistická vlastnost všech numerických algoritmů

Numerická stabilita

(Pseudo)definice

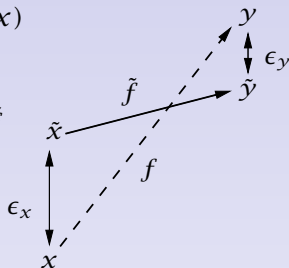
- ▶ „metoda počítá sice špatně, ale jenom trochu špatně“
 - ▶ žádoucí a přitom realistická vlastnost všech numerických algoritmů
- ▶ pro vstup x hledáme řešení $y = f(x)$
- ▶ ve skutečnosti
 - ▶ na aproximaci vstupu $\tilde{x} = x + \epsilon_x$
 - ▶ nepřesnou numerickou metodou \tilde{f}
 - ▶ dostaneme výsledek $\tilde{y} = y + \epsilon_y$



Numerická stabilita

(Pseudo)definice

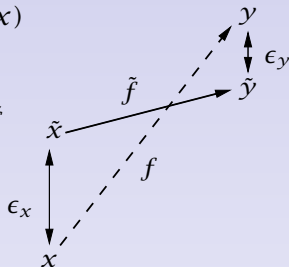
- ▶ „metoda počítá sice špatně, ale jenom trochu špatně“
 - ▶ žádoucí a přitom realistická vlastnost všech numerických algoritmů
- ▶ pro vstup x hledáme řešení $y = f(x)$
- ▶ ve skutečnosti
 - ▶ na aproximaci vstupu $\tilde{x} = x + \epsilon_x$
 - ▶ nepřesnou numerickou metodou \tilde{f}
 - ▶ dostaneme výsledek $\tilde{y} = y + \epsilon_y$
- ▶ algoritmus je **stabilní**, existuje-li pro každé x malé ϵ_x tak, že ϵ_y je malé
 - ▶ „malé ϵ “ znamená zpravidla $|\epsilon| \leq |\delta x|$



Numerická stabilita

(Pseudo)definice

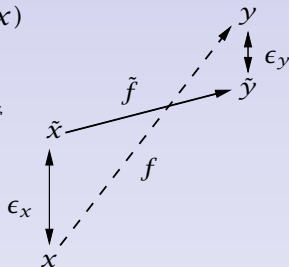
- ▶ „metoda počítá sice špatně, ale jenom trochu špatně“
 - ▶ žádoucí a přitom realistická vlastnost všech numerických algoritmů
- ▶ pro vstup x hledáme řešení $y = f(x)$
- ▶ ve skutečnosti
 - ▶ na aproximaci vstupu $\tilde{x} = x + \epsilon_x$
 - ▶ nepřesnou numerickou metodou \tilde{f}
 - ▶ dostaneme výsledek $\tilde{y} = y + \epsilon_y$
- ▶ algoritmus je **stabilní**, existuje-li pro každé x malé ϵ_x tak, že ϵ_y je malé
 - ▶ „malé ϵ “ znamená zpravidla $|\epsilon| \leq |\delta x|$
- ▶ silnější definice **zpětné stability** - $\epsilon_y = 0$



Numerická stabilita

(Pseudo)definice

- ▶ „metoda počítá sice špatně, ale jenom trochu špatně“
 - ▶ žádoucí a přitom realistická vlastnost všech numerických algoritmů
- ▶ pro vstup x hledáme řešení $y = f(x)$
- ▶ ve skutečnosti
 - ▶ na aproximaci vstupu $\tilde{x} = x + \epsilon_x$
 - ▶ nepřesnou numerickou metodou \tilde{f}
 - ▶ dostaneme výsledek $\tilde{y} = y + \epsilon_y$
- ▶ algoritmus je **stabilní**, existuje-li pro každé x malé ϵ_x tak, že ϵ_y je malé
 - ▶ „malé ϵ “ znamená zpravidla $|\epsilon| \leq |\delta x|$
- ▶ silnější definice **zpětné stability** - $\epsilon_y = 0$
- ▶ pro speciální druhy problémů jiné definice stability
 - ▶ např. numerické integrování - „systém nevybuchne“



O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Numerická stabilita

Příklady

- ▶ `adds` je numericky stabilní implementace sčítání

$$y = x_1 + x_2$$

- ▶ zvolíme $\delta = 2^{-22}$
- ▶ pro daná x_1, y_1 označme chybu jejich reprezentace $\delta_{1,2}$
- ▶ ve `float` platí $|\delta_{1,2}| \leq 2^{-25}$
- ▶ relativní chyba tohoto konkrétního sčítání je $|\delta_a| \leq 2^{-25}$

$$\begin{aligned}\tilde{y} &= \text{adds}(\tilde{x}_1, \tilde{x}_2) = (\tilde{x}_1 + \tilde{x}_2)(1 + \delta_a) = \\ & x_1(1 + \delta_1)(1 + \delta_a) + x_2(1 + \delta_2)(1 + \delta_a) = \\ & (x_1 + x_2) + x_1(\delta_1 + \delta_a + \delta_1\delta_a) + x_2(\delta_2 + \delta_a + \delta_2\delta_a)\end{aligned}$$

- ▶ i v nejhorším případě $|\delta_{1,2} + \delta_a + \delta_{1,2}\delta_a| \leq 3 \times 2^{-25}$
- ▶ pro nejhorší případ $x_1 = x_2$ nastane

$$|\tilde{y} - y| = |2x_1(\delta_1 + \delta_a + \delta_1\delta_a)| \leq |\delta y|$$

O čem to bude

Čísla v
plovoucí
řádové čářeNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ sčítání řady čísel $1230 + 1 + 1 + 1 + \dots$ je nestabilní
 - ▶ stabilní alternativa – seřadit a začít od nejmenšího

- ▶ sčítání řady čísel $1230 + 1 + 1 + 1 + \dots$ je nestabilní
 - ▶ stabilní alternativa - seřadit a začít od nejmenšího

- ▶ úvodní příklad $E_n = 1 - nE_{n-1}$
 - ▶ pro dostatečně velké N položíme $E_N = 0$
 - ▶ počítáme

$$E_{n-1} = \frac{1 - E_n}{n}$$

- ▶ v každém kroku je chyba redukována faktorem $1/n$

Zaokrouhlování

- ▶ žádoucí chování implementace aritmetických operací
 $+, -, *, /, \sqrt{}$:

$$\text{op}(x, y) = (x \text{ op } y)(1 + \delta) \quad \text{pro } |\delta| \leq u(\text{strojová přesnost})$$

- ▶ výsledek je stejně dobrý jako zaokrouhlený výsledek přesného výpočtu

Zaokrouhlování

- ▶ žádoucí chování implementace aritmetických operací
 $+, -, *, /, \sqrt{\cdot}$:

$$\text{op}(x, y) = (x \text{ op } y)(1 + \delta) \quad \text{pro } |\delta| \leq u(\text{strojová přesnost})$$

- ▶ výsledek je stejně dobrý jako zaokrouhlený výsledek přesného výpočtu
- ▶ stačí výpočet v dané přesnosti?

Zaokrouhlování

- ▶ žádoucí chování implementace aritmetických operací
 $+, -, *, /, \sqrt{}$:

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

$$\text{op}(x, y) = (x \text{ op } y)(1 + \delta) \quad \text{pro } |\delta| \leq u (\text{strojová přesnost})$$

- ▶ výsledek je stejně dobrý jako zaokrouhlený výsledek přesného výpočtu
- ▶ stačí výpočet v dané přesnosti?
 - ▶ např. $0.100 \times 2^1 - 0.111 \times 2^0$ (dvojkově, 3 cifry přesnosti)
 - ▶ po zarovnání řádu: $(0.100 - 0.011) \times 2^1 = 0.001 \times 2^1$
 - ▶ správný výsledek byl 0.0001×2^1 , chyba je
 $\delta = 1 > 0.001 = u$
- ▶ potřebujeme počítat s vyšší přesností
 - ▶ $(0.1000 - 0.0111) \times 2^1 = 0.0001 \times 2^1$
 - ▶ tzv. **guard digit**, pro $+$, $-$ stačí jedna

- ▶ základní zaokrouhlení – k nejbližšímu reprezentovatelnému číslu

$$1.24 \doteq 1.2 \quad 1.26 \doteq 1.3$$

- ▶ při nejednoznačnosti:
 - ▶ vždy nahoru
 - ▶ vždy dolů
 - ▶ **k sudému**: $1.25 \doteq 1.2$, ale $1.35 \doteq 1.4$
- ▶ asymetrické zaokrouhlení vnáší nežádoucí systematickou chybu
 - ▶ např. průměr velkého souboru čísel vyšší/nížší

Elementární funkce

- ▶ goniometrické, hyperbolické, exponenciální, logaritmické, odmocniny, ...

- ▶ goniometrické, hyperbolické, exponenciální, logaritmické, odmocniny, ...
- ▶ pohled matematika
 - ▶ zaběhaný intuitivní aparát
 - ▶ příjemné analytické vlastnosti (spojitost, derivace, ...)
 - ▶ většina matematických modelů se bez nich neobejde

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ goniometrické, hyperbolické, exponenciální, logaritmické, odmocniny, ...
- ▶ pohled matematika
 - ▶ zaběhaný intuitivní aparát
 - ▶ příjemné analytické vlastnosti (spojitost, derivace, ...)
 - ▶ většina matematických modelů se bez nich neobejde
- ▶ pohled programátora
 - ▶ samy o sobě numericky stabilní
 - ▶ optimalizované implementace v knihovnách
 - ▶ problematické chování v okrajových případech vede na numerickou nestabilitu
 - ▶ netriviální výpočetní náročnost

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Elementární funkce

Co je špatně?

- ▶ celý aparát vede na iracionální čísla ($\pi, e, \sqrt{2}, \dots$)
 - ▶ ve většině případů zdroj nepřesnosti
- ▶ e^x
 - ▶ tendence k přetečení - $e^{88} = 1.65 \times 10^{38}$
- ▶ $\tan x$
 - ▶ přetečení pro $x \rightarrow \frac{\pi}{2}$ (a perioda) roste nade všechny meze
- ▶ $\sin x$ a $\cos x$
 - ▶ pro $x \rightarrow \frac{\pi}{2}$ resp. $x \rightarrow 0$ vedou na *špatně podmíněné* rovnice
 - ▶ malá změna vstupu vede k velké změně výstupu
 - ▶ např. $\sin x = t$ pro $t \rightarrow 1$

Dilema tvůrce tabulek

- ▶ počítáme např. e^x na 3 místa, vychází (po n iteracích) 0.23450000 ...
 - ▶ je to zaokrouhlením přesného výsledku 0.2344999999, správné zaokrouhlení 0.234
 - ▶ nebo jen nepočítám dostatečně přesně 0.23450001, správné zaokrouhlení 0.235

- ▶ počítáme např. e^x na 3 místa, vychází (po n iteracích) 0.23450000 ...
 - ▶ je to zaokrouhlením přesného výsledku 0.2344999999, správné zaokrouhlení 0.234
 - ▶ nebo jen nepočítám dostatečně přesně 0.23450001, správné zaokrouhlení 0.235
- ▶ lze ukázat, že e^x nemůže být strojově reprezentovatelné pro strojově reprezentovatelné x
 - ▶ tedy k rozhodnutí stačí konečný počet kroků
 - ▶ pro různé funkce různě, zpravidla cca. dvojnásobná přesnost

O čem to bude

Čísla v
plovoucí
řádové čáře

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Co s tím?

- ▶ uvědomit si možné problematické chování
- ▶ potřebuji skutečně tyto funkce pro svůj výpočet?
 - ▶ mnoho problémů má i jiné řešení
- ▶ provést transformace, kterými se vyhneme numerické nestabilitě
 - ▶ čistě algebraické úpravy
 - ▶ podobné $(a + b)^2 = a^2 + 2ab + b^2$
- ▶ podle kontextu zvolit vhodnou implementaci
 - ▶ knihovní funkce
 - ▶ vlastní (Taylorův rozvoj) uzpůsobené specifickým podmínkám

- ▶ výraz $\sqrt{1+x} - \sqrt{1-x}$
- ▶ pro malá x odčítání velmi blízkých čísel
- ▶ transformace

$$\begin{aligned}\sqrt{1+x} - \sqrt{1-x} &= \\ &= \frac{(\sqrt{1+x} - \sqrt{1-x})(\sqrt{1+x} + \sqrt{1-x})}{\sqrt{1+x} + \sqrt{1-x}} \\ &= \frac{(1+x) - (1-x)}{\sqrt{1+x} + \sqrt{1-x}} \\ &= \frac{2x}{\sqrt{1+x} + \sqrt{1-x}}\end{aligned}$$

- ▶ sčítání velmi blízkých čísel - dostatečně přesné

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ výraz $\ln \sqrt{x+1} - \ln \sqrt{x}$
- ▶ pro velká x rozdíl blízkých čísel
- ▶ transformace

$$\begin{aligned}\ln \sqrt{x+1} - \ln \sqrt{x} &= \\ &= \ln(x+1)^{\frac{1}{2}} - \ln x^{\frac{1}{2}} = \frac{1}{2}(\ln(x+1) - \ln x) \\ &= \frac{1}{2} \ln \frac{x+1}{x} = \frac{1}{2} \ln\left(1 + \frac{1}{x}\right)\end{aligned}$$

- ▶ možná ztráta přesnosti, ale i tak lepší

Mocninné řady

Taylorův rozvoj

- ▶ jedna ze základních vět matematické analýzy

$$\begin{aligned} f(b) &= f(a) + f'(a) \frac{b-a}{1!} + \\ &+ f''(a) \frac{(b-a)^2}{2!} + \dots + f^{(n)}(a) \frac{(b-a)^n}{n!} + \\ &+ \xi \end{aligned}$$

kde ξ je dostatečně malé

Mocninné řady

Taylorův rozvoj

- ▶ jedna ze základních vět matematické analýzy

$$\begin{aligned} f(b) = & f(a) + f'(a) \frac{b-a}{1!} + \\ & + f''(a) \frac{(b-a)^2}{2!} + \dots + f^{(n)}(a) \frac{(b-a)^n}{n!} + \\ & + \xi \end{aligned}$$

kde ξ je dostatečně malé

- ▶ základ implementací většiny knihovních funkcí
- ▶ v extrémních případech potřebujeme „rozepsat“ a provést algebraické úpravy

Mocninné řady

Taylorův rozvoj základních funkcí

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{2 \cdot 4} + \frac{3x^3}{2 \cdot 4 \cdot 6} - \dots$$

$$\frac{1}{\sqrt{1+x}} = 1 - \frac{x}{2} + \frac{3x^2}{2 \cdot 4} - \frac{3 \cdot 5x^3}{2 \cdot 4 \cdot 6} + \dots$$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Mocninné řady

Demonstrační příklad

- ▶ výraz

$$\frac{1 - \cos x}{x^2}$$

se pro $x \rightarrow 0$ blíží $\frac{1}{2}$

- ▶ pro malá x odečtení dvou blízkých čísel
- ▶ konkrétně pro $x = 0.0005$ ve `float`:

$$\cos x = 0.99999988$$

$$1 - \cos x = 1.1920928 \times 10^{-7} \quad (\text{falešná přesnost})$$

$$\frac{1 - \cos x}{x^2} = 0.47683709$$

- ▶ správný výsledek je 0.49999999 (na 8 míst)

Mocninné řady

Demonstrační příklad

- ▶ náhrada $\cos x$ Taylorovým rozvojem

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

- ▶ pro výpočet na 8 cifer stačí do řádu x^4

$$\frac{1 - \cos x}{x^2} = \frac{1 - 1 + \frac{x^2}{2} - \frac{x^4}{24}}{x^2} = \frac{1}{2} - \frac{x^2}{24}$$

- ▶ výsledek výpočtu pro $x = 0.0005$ je 0.49999997
- ▶ podstatně lepší přesnost
- ▶ řádově méně aritmetických operací

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ rychlá konvergence pro malá x , pomalá pro větší
 - ▶ přímý důsledek Taylorovy věty
- ▶ vyplatí se použít vlastní vzorec pro malé x
 - ▶ resp. blízko problematického bodu numerické stability
- ▶ v ostatních případech zůstat u knihovní funkce
- ▶ jak poznáme, co je „malé x “?

Mocninné řady

Kdy použít

- ▶ známe požadovanou přesnost
 - ▶ bezpečné je použít přesnost daného typu
 - ▶ pro `float` a výsledky ~ 1 je to 10^{-7}
 - ▶ při méně přesných vstupních datech méně striktní
- ▶ hledáme c aby pro $x < c$ byl největší zanedbaný člen řady menší než požadovaná přesnost

Mocninné řady

Kdy použít

- ▶ známe požadovanou přesnost
 - ▶ bezpečné je použít přesnost daného typu
 - ▶ pro `float` a výsledky ~ 1 je to 10^{-7}
 - ▶ při méně přesných vstupních datech méně striktní
- ▶ hledáme c aby pro $x < c$ byl největší zanedbaný člen řady menší než požadovaná přesnost
- ▶ konkrétně v předchozím příkladu

$$\frac{x^4}{6!} < 10^{-7} \quad \text{tedy} \quad x < 0.09211$$

- ▶ zkontrolujeme chování výpočtu v c (v `double`):

$$\begin{aligned}\cos c &\doteq 0.995760\ 87237414645243 \\ (1 - \cos c)/c^2 &\doteq 0.499646\ 58945642923198 \\ 1/2 - c^2/24 &\doteq 0.499646\ 48949583333334 \\ c^4/6! &\doteq 0.000000\ 09997574124493\end{aligned}$$

O čem to bude

Čísla v
plovoucí
řádové čářeNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Mocninné řady

Kdy použít

- ▶ výsledný kód pro

$$\frac{1 - \cos x}{x^2}$$

```
float x,y;  
...  
if (x < 0.09211)  
    y = 0.5 + x*x/24.0;  
else  
    y = (1-cos(x))/(x*x);
```

- ▶ ve většině používána k normalizaci vektorů
- ▶ silové působení, např. dva bodové náboje **a** a **b**
 - ▶ velikost síly

$$F = k_c \frac{q_a q_b}{r^2}$$

kde $r = \|\mathbf{a} - \mathbf{b}\|$

- ▶ silový vektor

$$\mathbf{F} = F \cdot \frac{\mathbf{a} - \mathbf{b}}{r}$$

- ▶ snadný výpočet

$$r^2 = \sum (a_i - b_i)^2$$

- ▶ r už vyžaduje odmocninu

- ▶ počítačová grafika – osvětlení plochy
 - ▶ zpravidla interpolací počítáme normály k zakřivené ploše
 - ▶ vektory mají správný směr ale nejsou jednotkové
 - ▶ pro správný výpočet osvětlení potřebná normalizace
- ▶ ke všem těmto výpočtům nepotřebujeme striktně \sqrt{x}
- ▶ $\frac{1}{\sqrt{x}}$ se hodí mnohem více
 - ▶ násobení je lepší než dělení
- ▶ navíc zpravidla stačí velmi hrubá aproximace
 - ▶ citlivý je hlavně směr vektoru, ten zůstává
- ▶ přímo Taylorův rozvoj $\frac{1}{\sqrt{1+x}}$
- ▶ iterační metody (Newtonova atd.)

Odmocnina

Fast inverse square root

- ▶ hra Quake III na konci 90. let
- ▶ převratně realistická grafika
- ▶ vděčila velmi rychlému výpočtu $\frac{1}{\sqrt{x}}$
- ▶ kód zřejmě pochází z grafických knihoven SGI

Odmocnina

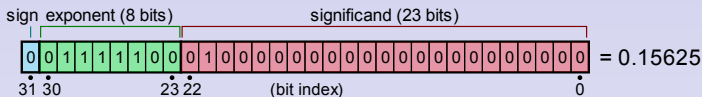
Fast inverse square root

- ▶ hra Quake III na konci 90. let
- ▶ převratně realistická grafika
- ▶ vděčila velmi rychlému výpočtu $\frac{1}{\sqrt{x}}$
- ▶ kód zřejmě pochází z grafických knihoven SGI

```
float invSqrt(float x) {  
    float xhalf = 0.5f*x,y;  
    union {  
        float x;  
        int i;  
    } u;  
    u.x = x;  
    u.i = 0x5f3759df - (u.i >> 1);  
    y = u.x * (1.5f - xhalf * u.x * u.x);  
    return y;  
}
```

- ▶ funguje na základě IEEE reprezentace čísla

$$x = (1 + m_x) \times 2^{e_x}$$



- ▶ přitom $m_x = M_x / 2^{23}$ a $e_x = E_x - 127$
- ▶ základ výpočtu

$$y = \frac{1}{\sqrt{x}}$$

$$\log_2 y = -\frac{1}{2} \log_2 x$$

$$\log_2(1 + m_y) + e_y = -\frac{1}{2} \log_2(1 + m_x) - \frac{1}{2} e_x$$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ pro $m \in [0, 1)$ si lze dovolit aproximaci

$$\log_2(1 + m) = m + \sigma$$

- ▶ dosazením a úpravami dostaneme

$$E_y \times 2^{23} + M_y = R - \frac{1}{2}(E_x \times 2^{23} + M_x)$$

tj. červený řádek v kódu s empiricky stanovenou konstantou R

- ▶ více viz http://en.wikipedia.org/wiki/Fast_inverse_square_root

- ▶ poslední krok je jedna iterace Newtonovy metody
- ▶ pro vstup x hledáme takové y aby $y = \frac{1}{\sqrt{x}}$
- ▶ tj. hledáme kořen funkce $f(y) = \frac{1}{y^2} - x$
- ▶ Newtonovou metodou

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} = y_n - \frac{\frac{1}{y_n^2} - x}{-\frac{2}{y_n^3}} = y_n + \frac{y_n(1 - xy_n^2)}{2}$$

což už odpovídá poslednímu výrazu v kódu

Odmocnina

Fast inverse square root

- ▶ aproximace je překvapivě přesná
 - ▶ v Quake III je zakomentována další Newtonova iterace
 - ▶ nebyla potřeba
- ▶ cca. $4\times$ rychlejší než dělení
- ▶ dnes překonána instrukcí `rsqrtss`

Odmocnina

Fast inverse square root

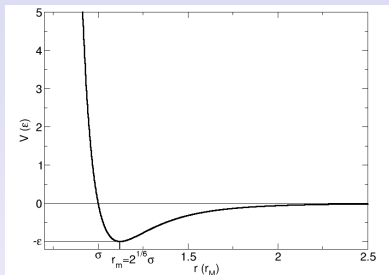
- ▶ aproximace je překvapivě přesná
 - ▶ v Quake III je zakomentována další Newtonova iterace
 - ▶ nebyla potřeba
- ▶ cca. $4\times$ rychlejší než dělení
- ▶ dnes překonána instrukcí `rsqrtss`
- ▶ přesné pochopení konkrétního účelu výpočtu
 - ▶ včetně zhodnocení „má smysl tvrdě optimalizovat“
- ▶ volba adekvátní metody
 - ▶ přispěla k velkému komerčnímu úspěchu
 - ▶ pro jiné účely by byla zcela nevhodná

Odmocnina

Lze se jí i vyhnout

- ▶ Lenard-Jonesuv potenciál
 - ▶ nevazebná interakce dvou atomů
 - ▶ významná pro modelování biochemických dějů
- ▶ vzorec pro *energii* (potenciál)

$$E_{LJ} = \frac{A}{r^{12}} - \frac{B}{r^6} \quad \text{kde } r \text{ je vzdálenost atomů}$$



O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Odmocnina

Lze se jí i vyhnout

- ▶ výpočet velikosti síly

$$F = \frac{dE}{dr} = -\frac{12A}{r^{13}} + \frac{6B}{r^7}$$

- ▶ liché exponenty \Rightarrow bude potřeba odmocnina

O čem to bude

Číslo v
plovoucí
řádové čáře

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Odmocnina

Lze se jí i vyhnout

- ▶ výpočet velikosti síly

$$F = \frac{dE}{dr} = -\frac{12A}{r^{13}} + \frac{6B}{r^7}$$

- ▶ liché exponenty \Rightarrow bude potřeba odmocnina
- ▶ zajímá mě většinou silový vektor

$$\mathbf{F} = F \frac{\mathbf{a} - \mathbf{b}}{r} = (\mathbf{a} - \mathbf{b}) \left(\frac{6B}{r^8} - \frac{12A}{r^{14}} \right)$$

- ▶ tedy vystačím s r^2

O čem to bude

Číslo v
plovoucí
řádové čáře

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Odmocnina

Lze se jí i vyhnout

- ▶ implementace

```
float n,r2,r4,r8,r14,F[3];
int i;
for (n=0,i=0; i<3; i++) {
    F[i] = a[i] - b[i];
    n += F[i]*F[i];
}
r2 = 1.0/n;
r4 = r2*r2; r8 = r4*r4;
r14 = r8*r4*r2;

n = 12*A*r14 - 6*B*r8;
for (i=0; i<3; i++) F[i] *= n;
```

O čem to bude

Číslo v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Odmocnina

Lze se jí i vyhnout

- ▶ implementace

```
float n,r2,r4,r8,r14,F[4];
int i;
for (n=0,i=0; i<4; i++) {
    F[i] = a[i] - b[i];
    n += F[i]*F[i];
}
r2 = 1.0/n;
r4 = r2*r2; r8 = r4*r4;
r14 = r8*r4*r2;

n = 12*A*r14 - 6*B*r8;
for (i=0; i<4; i++) F[i] *= n;
```

- ▶ kvůli vektorizaci může mít smysl $F[4]$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Aneb jak se vyhnout goniometrickým funkcím

- ▶ ve 2D jeden úhel ϕ
- ▶ složení součtem, inverze $-\phi$
- ▶ aplikace násobením maticí

$$R = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix}$$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Aneb jak se vyhnout goniometrickým funkcím

- ▶ ve 2D jeden úhel ϕ
- ▶ složení součtem, inverze $-\phi$
- ▶ aplikace násobením maticí

$$R = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix}$$

- ▶ goniometrickým funkcím se můžeme vyhnout

$$\sin \phi = \frac{2t}{1+t^2} \quad \cos \phi = \frac{1-t^2}{1+t^2}$$

- ▶ numericky příjemné, nevyjádříme $\phi = \pm \frac{\pi}{2}$
- ▶ nelineární vztah ϕ a t

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Eulerovy úhly

- ▶ 2 stupně volnosti
 - ▶ teodolit, sledování bodu ve 3D
 - ▶ vykazuje singularity

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

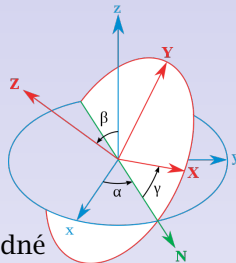
Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Eulerovy úhly

- ▶ 2 stupně volnosti
 - ▶ teodolit, sledování bodu ve 3D
 - ▶ vykazuje singularity
- ▶ 3 stupně volnosti – Eulerovy úhly
 - ▶ na první pohled intuitivní
 - ▶ nekonzistentní konvence
 - ▶ častější singularity – *gimbal lock*
- ▶ numericky a programátorsky nevýhodné
 - ▶ značně nepřehledné
 - ▶ záleží na pořadí
 - ▶ singularity – numerické problémy



Parametrizace rotace

Malice

- ▶ maticové vyjádření
 - ▶ složení 3 rotací podél osy

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & -\sin \beta \\ 0 & \sin \beta & \cos \beta \end{pmatrix}$$

- ▶ aplikovatelná stejná substitute
 - ▶ vede na komplikované polynomy

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Matice

- ▶ přímo maticí 3×3
- ▶ numericky samo o sobě stabilní, pokrývá všechny případy
- ▶ příliš mnoho „parametrů navíc“
- ▶ musí být ortogonální
- ▶ snadno degeneruje
 - ▶ nepřesností výpočtu ztrácí ortogonalitu
 - ▶ korekce není jednoduchá
- ▶ zabere více paměti

Parametrizace rotace

Představa prostoru rotací

- ▶ rotace v rovině
 - ▶ skalár $x \in [-1, 1]$, kde $t = \sin \frac{\phi}{2}$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

Parametrizace rotace

Představa prostoru rotací

- ▶ rotace v rovině
 - ▶ skalár $x \in [-1, 1]$, kde $t = \sin \frac{\phi}{2}$
- ▶ rotace s dvěma stupni volnosti (teodolit)
 - ▶ 2D vektor \mathbf{v} , $|\mathbf{v}| \in [0, 1]$
 - ▶ směr určuje osu, velikost úhel $|\mathbf{v}| = \sin \frac{\phi}{2}$
 - ▶ všechny rotace reprezentovány kruhem

Parametrizace rotace

Představa prostoru rotací

- ▶ rotace v rovině
 - ▶ skalár $x \in [-1, 1]$, kde $t = \sin \frac{\phi}{2}$
- ▶ rotace s dvěma stupni volnosti (teodolit)
 - ▶ 2D vektor \mathbf{v} , $|\mathbf{v}| \in [0, 1]$
 - ▶ směr určuje osu, velikost úhel $|\mathbf{v}| = \sin \frac{\phi}{2}$
 - ▶ všechny rotace reprezentovány kruhem
- ▶ pokrytí (2:1) jednotkovou koulí
 - ▶ projekce \mathbf{v} do roviny určuje osu, $z = \cos \frac{\phi}{2}$
 - ▶ \mathbf{v} a $-\mathbf{v}$ představují tutéž rotaci

Parametrizace rotace

Představa prostoru rotací

- ▶ rotace v rovině
 - ▶ skalár $x \in [-1, 1]$, kde $t = \sin \frac{\phi}{2}$
- ▶ rotace s dvěma stupni volnosti (teodolit)
 - ▶ 2D vektor \mathbf{v} , $|\mathbf{v}| \in [0, 1]$
 - ▶ směr určuje osu, velikost úhel $|\mathbf{v}| = \sin \frac{\phi}{2}$
 - ▶ všechny rotace reprezentovány kruhem
- ▶ pokrytí (2:1) jednotkovou koulí
 - ▶ projekce \mathbf{v} do roviny určuje osu, $z = \cos \frac{\phi}{2}$
 - ▶ \mathbf{v} a $-\mathbf{v}$ představují tutéž rotaci
- ▶ parametrizace x, y, z s omezením $x^2 + y^2 + z^2 = 1$
 - ▶ netrpí singularitami
 - ▶ lze snadno interpolovat (včetně normalizace) apod.

Parametrizace rotace

Představa prostoru rotací

- ▶ analogická konstrukce pro obecné rotace ve 3D
- ▶ 3D vektor velikosti $[0, 1]$
- ▶ všechny rotace jsou reprezentovány plnou koulí
- ▶ pokrytí (2:1) jednotkovou hyperkoulí ve 4D
- ▶ parametrizace x, y, z, w s omezením
$$x^2 + y^2 + z^2 + w^2 = 1$$
- ▶ stejné numericky výhodné vlastnosti

- ▶ algebraický aparát na vektory ve 4D
- ▶ více ekvivalentních definic
- ▶ nejjednodušší $\mathbb{H} = \mathbb{R}^4$ s kanonickou bází $(1, i, j, k)$ a násobením

$$i^2 = j^2 = k^2 = ijk = -1$$

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ algebraický aparát na vektory ve 4D
- ▶ více ekvivalentních definic
- ▶ nejjednodušší $\mathbb{H} = \mathbb{R}^4$ s kanonickou bází $(1, i, j, k)$ a násobením

$$i^2 = j^2 = k^2 = ijk = -1$$

- ▶ chápeme $\mathbb{R}^3 \subset \mathbb{H}$ jako ryze imaginární kvaterniony
- ▶ potom pro $|q| = 1$ je zobrazení $x \mapsto qx\bar{q}$ rotace ve 3D

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ algebraický aparát na vektory ve 4D
- ▶ více ekvivalentních definic
- ▶ nejjednodušší $\mathbb{H} = \mathbb{R}^4$ s kanonickou bází $(1, i, j, k)$ a násobením

$$i^2 = j^2 = k^2 = ijk = -1$$

- ▶ chápeme $\mathbb{R}^3 \subset \mathbb{H}$ jako ryze imaginární kvaterniony
- ▶ potom pro $|q| = 1$ je zobrazení $x \mapsto qx\bar{q}$ rotace ve 3D
- ▶ nakrytí 2:1 (q a $-q$ reprezentují stejnou transformaci)
- ▶ skládání rotací je násobení
- ▶ inverze je \bar{q}
- ▶ korekce degenerace normalizací

O čem to bude

Čísla v
plovoucí
řádové čárceNumerická
stabilita

Zaokrouhlování

Elementární
funkceAlgebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ součást mnoha knihoven
 - ▶ libquat, Boost, Ogre, DirectX, ...
 - ▶ optimalizované implementace
- ▶ typické funkce
 - ▶ norma/normalizace
 - ▶ komplement
 - ▶ násobení
 - ▶ transformace vektoru/vektorů
 - ▶ převod z/na Eulerovy úhly
 - ▶ převod z/na matice 3×3
- ▶ v praxi zpravidla použijeme příhodnou knihovnu
 - ▶ tyto operace už před námi někdo naprogramoval lépe
 - ▶ musíme chápat principy, na kterých stojí

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ použití pro animace ve 3D
- ▶ lineární interpolace

$$\text{lerp}(q_0, q_1, t) = \frac{(1-t)q_0 + tq_1}{\|(1-t)q_0 + tq_1\|}$$

- ▶ nevhodné vlastnosti
 - ▶ nelineární vztah t a úhlu rotace
 - ▶ špatná kontrola úhlové rychlosti

- ▶ sférická lineární interpolace (SLERP)

$$\text{slerp}(q_0, q_1, t) = \frac{\sin(1-t)\Omega}{\sin \Omega} q_0 + \frac{\sin t\Omega}{\sin \Omega} q_1$$

kde $\cos \Omega = q_0 \cdot q_1$

O čem to bude

Číslo v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ sférická lineární interpolace (SLERP)

$$\text{slerp}(q_0, q_1, t) = \frac{\sin(1-t)\Omega}{\sin \Omega} q_0 + \frac{\sin t\Omega}{\sin \Omega} q_1$$

kde $\cos \Omega = q_0 \cdot q_1$

- ▶ v kvaternionech lze vyjádřit

$$\text{slerp}(q_0, q_1, t) = q_0 (q_0^{-1} q_1)^t$$

- ▶ velmi efektivní implementace s použitím SSE, AVX

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ sférická lineární interpolace (SLERP)

$$\text{slerp}(q_0, q_1, t) = \frac{\sin(1-t)\Omega}{\sin \Omega} q_0 + \frac{\sin t\Omega}{\sin \Omega} q_1$$

kde $\cos \Omega = q_0 \cdot q_1$

- ▶ v kvaternionech lze vyjádřit

$$\text{slerp}(q_0, q_1, t) = q_0 (q_0^{-1} q_1)^t$$

- ▶ velmi efektivní implementace s použitím SSE, AVX
- ▶ pro většinu operací s rotacemi jsou kvaterniony nejlepší volba
 - ▶ přítel Google ochotně poradí

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí

- ▶ matematické modely reality
- ▶ použití idealizovaného aparátu reálných čísel
- ▶ konečná reprezentace čísel je zdrojem problémů
 - ▶ různé projevy pro různé operace
 - ▶ i v triviálním výpočtu může dojít k numerickému problému
 - ▶ míra závadnosti formalizována pojmem numerické stability
- ▶ elementární funkce
 - ▶ samotná implementace numericky stabilní
 - ▶ použití ve výrazech může vést na numerické problémy
 - ▶ explicitní Taylorův rozvoj pro okrajové případy
- ▶ rychlost kritické části výpočtu
 - ▶ kompromis rychlost vs. přesnost
 - ▶ změření nemusí být triviální

O čem to bude

Čísla v
plovoucí
řádové čárce

Numerická
stabilita

Zaokrouhlování

Elementární
funkce

Algebraické
úpravy

Mocninné řady

Odmocnina

Algebra
kvaternionů

Shrnutí