
Dokumentace k projektu pro předměty IZP a IUS

# Iterační výpočty

projekt č. 2

19. listopadu 2008

Autor: Vojtěch Kalčík, xkalci01@fit.stud.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

# Obsah

# 1 Úvod

Tento dokument se váže k projektu, který je zaměřený na iterační výpočty. Popisuje návrh a implementaci programu pro výpočet funkce tangens, přirozeného logaritmu a zrychlení nebo polohy objektu. Tento objekt se pohybuje v prostoru. Jeho zrychlení nebo polohu zjišťuje radar a posílá je programu ke zpracování. Jde o konzolovou aplikaci, u které lze pomocí parametrů nastavit, co má počítat ze vstupních hodnot.

K samotnému programu bylo nutné najít rozvoj, který by tyto funkce počítal. Přesně vyčíslit funkce není možné, takže se podle zadané přesnosti počítá přibližná hodnota, která se více nebo méně ke skutečné hodnotě blíží. S každým $x$ funkce není vhodné počítat. Je nutné počítat s takovými $x$, aby byl výsledek správný. Této úpravě na lepší definiční obor funkce se říká heuristika. Funkce tangens i přirozený logaritmus jsou různé. Každá má jiný definiční obor a jiné chování. Proto i heuristika a postup výpočtu je pro každou jiný. Pro část s počítáním s radary byly potřeba fyzikální vzorečky z mechaniky.

Všechny tři části programu jsou na sobě nezávislé. Dá se o nich uvažovat jako o třech samostatných podprogramech. Z toho důvodu se každou částí zabývám vždy v samostatné podkapitole.

# 2 Analýza problému a princip jeho řešení

V této kapitole popíši zadání a také se zaměřím podrobně na jednotlivé funkce. Proberu fyzikální závislosti pro výpočet zrychlení a polohy, když není konstantní rychlost.

## 2.1 Zadání pro celý program

Program je psaný v jazyce C. To, co se bude počítat, je dáno parametry programu. Vstupem jsou čísla typu double. Pokud vstupní hodnota není číslo, je vstupní hodnotou konstanta *NAN*, což je zkratka not a number. Na výstupu je stejný počet čísel jako na vstupu. Každé toto číslo je na novém řádku. Výsledkem může být také *NAN* nebo ±*INFINITY* (nekonečno). To lze získat například při dělení nulou. Pokud je program spuštěn s parametrem *-h*, tak se zobrazí nápověda.

## 2.2 Zadání pro tangens

Pokud je program spuštěn s parametry *-tan epsilon*, kde epsilon je číslo typu double větší než 0, tak počítá ze vstupu hodnotu blízkou hodnotě tangens. Odchylka je daná parametrem epsilon.

## 2.3 Zadání pro přirozený logaritmus

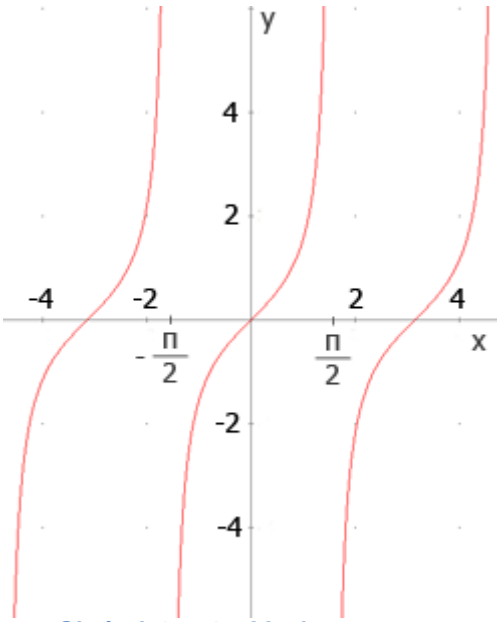
Pokud je program spuštěn s parametry *-ln epsilon*, kde epsilon je číslo typu double větší než 0, tak počítá ze vstupu hodnotu blízkou hodnotě přirozeného logaritmu. Odchylka je daná parametrem epsilon.

## 2.4 Zadání pro radar

Pokud je program spuštěn s parametrem *-radar1* nebo *-radar2*, tak zpracovává informace z radaru o objektu, který se pohybuje v prostoru. Tyto informace posílá radar programu v intervalu 0,5 *s*. Je-li parametrem *-radar1*, tak těmito informacemi je zrychlení v $m.s^{-2}$ po příslušné ose. První vstupní hodnota je zrychlení objektu na ose x, druhá na y, třetí na z, čtvrtá znovu na x, ale až po 0,5 *s* atd. Výstupem jsou naopak postupně souřadnice objektu v *m*. Nejdříve x, pak y, pak z a potom zase x po 0,5 s. Pokud je parametr *-radar2*, tak naopak vstupem jsou souřadnice objektu a výstupem je zrychlení. Z toho vyplývá, že pokud radar1 zpracuje hodnoty a jeho výstup zpracuje radar2, tak jeho výstupem budou původní vstupní hodnoty pro radar1.

## 2.5 Funkce Tangens

Funkce tangens má definiční obor $\mathbb{R} - \{\frac{\pi}{2} + k\pi\}$. Body $\{\frac{\pi}{2} + k\pi\}$ do ní nepatří, protože v těchto bodech diverguje k ±∞. Funkce je neomezená, to znamená, že obor hodnot je v intervalu (-∞, ∞). Je lichá, takže je středově souměrná podle počátku. Jde o periodickou funkci a v celém definičním oboru je rostoucí.




Obrázek 2.1: Graf funkce tangens

.

Hodnota funkce se dá počítat pomocí Taylorovy řady. Je to mocninná řada, která se obecně matematicky vyjádří takto:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f(3)}{3!}(x-a)^3 + \cdots = \sum_{k=0}^{\infty} \frac{f(k)}{k!}(x-a)^k$$

Tato řada je pro funkce, které má program počítat, nekonečná a pro vyjádření přesné hodnoty by bylo nutné sčítat donekonečna členy, což je technicky a časově neproveditelné. Z toho důvodu je nutné si zvolit epsilon, které udává odchylku. Jakmile dosáhne program menší odchylky než je tato, je výpočet ukončen.

Možné postupy jsem našel dva. První způsob je počítání řady přímo pro tangens. U této varianty je nutné počítat s Bernulliho čísly, která sama o sobě mají složitý vzoreček pro výpočet. Druhý způsob je vypočítat pomocí Taylorovy řady funkce sin a cos. Potom pomocí vztahu $\tan x = \frac{\sin x}{\cos x}$ spočítat výsledek. Zvolil jsem si druhou možnost kvůli jednoduššímu algoritmu.

Taylorova řada pro sinus:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

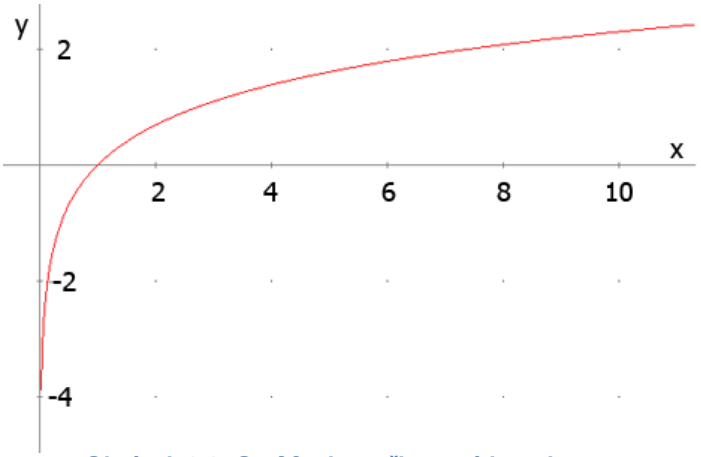
Taylorova řada pro kosinus:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Funkce tangens je periodická a pro výpočet pomocí Taylorova rozvoje je nejlepší využít interval $(-\frac{\pi}{2}, \frac{\pi}{2})$, ještě lépe interval $(-\frac{\pi}{4}, \frac{\pi}{4})$. Přepočítat matematicky *x* do intervalu $(-\frac{\pi}{2}, \frac{\pi}{2})$ není problém. Ovšem když to přepočítává program, tak tu již zádrhel je, ale o tom až dál. Pokud chci ještě o něco zvýšit přesnost, tak mohu převést *x* do intervalu $(-\frac{\pi}{4}, \frac{\pi}{4})$, a to pomocí vzorečku pro $\tan(2\alpha)$. Jakmile udělám tuto heuristiku, mohu již spočítat hodnotu pro sin a cos.

## 2.6 Funkce přirozený logaritmus

Funkce přirozený logaritmus je logaritmus o základu $e$, kde $e$ je Eulerovo číslo, které má hodnotu 2,7182818284… Má definiční obor v intervalu (0, ∞). Je to inverzní funkce k funkci $e^x$. V celém definičním oboru je rostoucí.




Obrázek 2.2: Graf funkce přirozený logaritmus

Tato funkce se dá počítat také pomocí Taylorovy řady. U funkce je nutné provést heuristiku do intervalu <1, 2>. Heuristika se dá udělat pomocí vztahů:

$$\ln(x \times y) = \ln x + \ln y$$

$$\ln \frac{x}{y} = \ln x - \ln y$$

Taylorova řada pro přirozený logaritmus:

$$\ln x = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \cdots = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^n}{n}$$

## 2.7 Radar

Zrychlení je fyzikální veličina, která udává změnu rychlosti za změnu času. Značí se *a*. Jednotka je $m.s^{-2}$. Například gravitační zrychlení na zemi je přibližně 10 $m.s^{-2}$. To znamená, že pokud pustím kámen, který bude padat volným pádem, tak v čase $t$ = 0*s* je rychlost $v$ = $0m.s^{-1}$. V čase $t$ = 1*s* => $v$ = $10m.s^{-1}$; $t$ = 2 *s* => $v$ = $20m.s^{-1}$…

Poloha objektu se spočítá ze vzorečku pro dráhu $s = s_0 + v_0 t + \frac{1}{2} a t^2$, kde $s_0$ je počáteční dráha a $v_0$ je počáteční rychlost. Pokud bude počáteční dráha a rychlost 0, tak vzoreček je pouze $s = \frac{1}{2} a t^2$. Jde o upravený vzoreček $s = v \times t$, kde se za $v$ dosadilo $v = a \times t$. Ovšem je tam ještě $\frac{1}{2}$. To je proto, že rychlost se během času $t$ mění, takže je nutné vzít rychlost průměrnou, a ta je přesně v čase $\frac{t}{2}$.

Pro výpočet zrychlení se použije vzoreček $a = \frac{2 \times (s - s_0 - v_0 t)}{t^2}$, což je pouze upravený vzoreček pro dráhu.

# 3 Návrh řešení problému

## 3.1 Čtení vstupních dat

V zadání stojí, že hodnoty vstupu jsou oddělené libovolným počtem bílých znaků, což jsou mezery, prázdné řádky, tabulátory atd. V programovacím jazyku C je na toto funkce *scanf*, která dokáže sama tyto hodnoty načíst. Hodnoty se nejen dají načíst, ale i přímo kontrolovat, jestli jsou korektní. Nebylo tedy nutné pro získání vstupních hodnot psát vlastní funkci. Je nutné zkontrolovat, jestli epsilon je větší než nula.

## 3.2 Taylorova řada

Existuje několik způsobů jak počítat Taylorovu řadu. Některé jsou lepší a některé horší. Pokud jsou v této řadě faktoriály nebo mocniny, tak čísla velice strmě rostou, takže se brzy může stát, že se nevejdou do datových typů, které jsou k dispozici. Proto nelze počítat vždy znovu a znovu faktoriál nebo mocninu, protože by bylo moc prováděných příkazů a brzy by čísla byla větší, než bychom mohli použít. Členy Taylorovy řady pro funkce, které počítá program, vypadají tak, že pokud je velké číslo nad zlomkovou čárou, je i pod ní. To znamená, že ve výsledku je číslo relativně malé. Faktoriál *n* je vynásobení řady čísel $1 \times 2 \times 3 \times \ldots n$. Mocnina z *n* je $n \times n \times n \times \ldots$ Díky tomuto můžeme následující člen spočítat vynásobením a vydělením vhodnými čísly člen předešlý. Uvedu příklad pro funkci sinus:

2. člen: $c_2 = -\frac{x^3}{3!} = -\frac{\mathrm{x} \times \mathrm{x} \times \mathrm{x}}{1 \times 2 \times 3}$

3.člen: $c_3 = c_2 \times (-1) \times \frac{\mathrm{x}^2}{4 \times 5} = \frac{\mathrm{x}^5}{5!}$

## 3.3 Návrh řešení funkce tangens

Nejdříve je nutné provést heuristiku. Prvním krokem je, že pokud je *x* mimo interval $< -\frac{\pi}{2}, \frac{\pi}{2} >$ , je nutné ho do tohoto intervalu převést. Převod provádím tak, že spočítám $k = \frac{x-(-\frac{\pi}{2})}{\pi}$. *k* je desetinné číslo, takže musím oříznout vše za desetinnou čárkou. Do správného intervalu se již dostanu $x = x - k \times \pi$. Problémem je, že hodnotu π používám jen přibližnou. A pokud je *x* hodně vzdálené od tohoto intervalu, tak se rozdíl od přesného π nasčítá a *x* se může dokonce dostat mimo tento interval. V tom případě posun provádím do doby, dokud se do daného intervalu nedostanu. Je ale jasné, že výsledná hodnota nebude správná. Z toho důvodu mi program pro velká nebo hodně malá čísla nepočítá správně. Konkrétně do 1e10 jsou výsledky v pořádku, do 1e15 jsou ještě ucházející a co je nad, je nepoužitelné.

Další zpřesnění dělám přes vzoreček pro $\tan(2\alpha) = \frac{2\tan\alpha}{1-\tan^2\alpha}$. Potom mohu *x* vydělit dvěma, a tím pádem počítat s *x* z intervalu $< -\frac{\pi}{4}, \frac{\pi}{4} >$. Ale ani toto nevyřeší problém, když se *x* blíží k $\pm\frac{\pi}{2}$. Zde dochází k velké odchylce od skutečné hodnoty. Konkrétně použitelné hodnoty jsou do 1,5707. Řešil jsem to tak, že jsem vzorečkem zmenšoval odchylku podle vzdálenosti od $\pm\frac{\pi}{2}$, ale pak již byla tak malá, že nezáleželo na hodnotě epsilon. Ať jsem nastavil jakékoli, tak výsledek byl stejný. Z toho důvodu jsem zmenšování odchylky nakonec nepoužil. Ovšem přece jen je nutné hodnotu odchylky změnit, protože počítám sinus i kosinus s touto odchylkou, takže je chyba dvojnásobná. Pro normální hodnoty stačí epsilon vynásobit 0,1 a potom je odchylka dostatečná. Neplatí pro hodnoty blízké $\pm\frac{\pi}{2}$, kde kosinus konverguje k nule. Když se tak malým číslem dělí, tak stačí malý rozdíl a vychází hodně jiné výsledky. Pokud je použit vzoreček pro tan(2α), tak znovu násobím epsilon s 0,1.

## 3.4 Návrh řešení funkce přirozený logaritmus

Heuristika se dá dělat pomocí vztahů, které jsem zmiňoval již výše. Jakékoli číslo z intervalu (2, ∞) mohu vyjádřit jako určité číslo z intervalu <1,2> krát $2^n$. A jakékoli číslo z intervalu (0, 1) jako číslo z intervalu <1,2> krát $2^{-n}$. Uložil jsem si konstantu přirozeného logaritmu 2. Pak už mi stačí spočítat akorát přirozený logaritmus z určitého čísla z intervalu <1,2> a buď přičíst, nebo odečíst $n \times \ln 2$. Podle toho, jestli původní číslo bylo větší než 2, nebo menší než 1.

## 3.5 Návrh řešení úlohy s radary

Radar vrací samostatné nezávislé hodnoty pro každou osu. Proto lze úlohu rozdělit do třech samostatných na sobě nezávislých výpočtů. Je nutné si pamatovat pro každou osu aktuální pozici a aktuální rychlost. Čas plyne ze zadání a je 0,5 sekund. A tím znám všechny potřebné údaje pro výpočet.

# 4 Specifikace testů

Zadání je takové, že by neměl existovat vstup, pro který by program zahlásil chybu. Pokud je na vstupu něco, co není číslem, tak se programu předá *nan*.

## 4.1 Testy pro tangens

Test 1: Správnost výpočtu -> Předpokládaná správná hodnota.

```
0 -> 0
-0.785398163 -> 1(přibližně)
1.5 -> 14.101419
100 -> -0.587213
1000 -> 1.470324
1e10 -> -0.558349
Abc -> nan
```

## 4.2 Testy pro přirozený logaritmus

Test 2: Správnost výpočtu -> Předpokládaná správná hodnota.

```
0 -> -infinity
-5 -> nan
1 -> 0
2.718281828 -> 1(přibližně)
1000 -> 6,907755
1e10 -> 23,025850
Bca -> nan
```

## 4.3 Testy pro radar1

Test 3: Správnost výpočtu -> Předpokládaná správná hodnota. (hodnoty na sebe navazují, jde pouze o jednu osu.)

```
0 -> 0
1 -> 0.125
2 -> 0.625
3 -> 1.75
-2.22 -> 2.9725
1e10 -> 1.25e9
nan -> nan
2 -> nan
```

## 4.4 Testy pro radar2

Test 4: Správnost výpočtu -> Předpokládaná správná hodnota. (hodnoty na sebe navazují, jde pouze o jednu osu.)

```
0 -> 0
1 -> 8
2 -> -8
3 -> 8
-2.22 -> -57.76
1e10 -> 8e10
nan -> nan
2 -> nan
```

# 5 Popis řešení

## 5.1 Ovládání programu

Program je konsolová aplikace, u které se nastavuje její chování pouze pomocí parametrů. Jaké parametry se mohou použít se dá najít v nápovědě, která se vyvolá parametrem *–h.* Na výstupu je stejný počet čísel jako na vstupu.

## 5.2 Datové typy

V programu se pracuje až na výjimky s datovým typem double, jenž reprezentuje reálná čísla. Epsilon, vstup i výstup jsou typu double. Pro radary jsem vytvořil pole datových struktur *Tpoloha*, kde počet prvků pole je počet os. Konkrétně pro toto zadání to jsou 3. Datová struktura obsahuje aktuální polohu a rychlost objektu v proměnných *s* a *v0*. Obě proměnné jsou typu double.

## 5.3 Vlastní implementace

Ve funkci *main* se zavolá funkce *zpracuj_parametry*, které se jí předávají. Funkce vrací datovou strukturu *Tparametry*. V datové struktuře je proměnná *stav* typu int, kde je vrácen stav pro případ, kdy je vše v pořádku, pro nápovědu a pro jednotlivé chyby, které mohou nastat. Dále obsahuje proměnnou *co_delat*, podle které program pozná, jestli má počítat tangens, logaritmus nebo radary. Poslední proměnná je *epsilon*, do které se uloží epsilon pro funkce tangens a logaritmus. Další funkce, která se zavolá z funkce *main,* je funkce *cti*. Funkce čte znaky ze vstupu pomocí *scanf* a volá buď funkci *tangens ,logaritmus,* nebo *radar*. Pokud je stav nastaven tak, že je nutné něco vypisovat, zavolá funkce *vypis_hlasku*, která rozpozná, kterou hlášku má vypsat, a vypíše ji. Také přizpůsobí výstup. Pokud vypisuje chybu, vypíše ji na chybový výstup, pokud nápovědu, tak na standardní výstup.

Ve funkci *tangens* se provede heuristika pomocí funkce *najdi_x* a zavolá se dvakrát funkce *rozvoj_tan*. Jednou pro sinus a jednou pro kosinus. Nakonec se navrácené hodnoty těchto dvou funkcí podělí.

Ve funkci *logaritmus* se provede heuristika a zavolá se funkce *rozvoj_ln*.

Funkce *radar* neobsahuje žádné další volání funkce. Zde se pouze vybere, jestli se má počítat dráha nebo zrychlení, a spočítá se. Je nutné také spočítat a uložit aktuální rychlost.

# 6 Závěr

Program funguje správně. Akorát s výpočtem funkce tangens nejsem příliš spokojen a kvůli speciálním případům, kdy nevychází příliš přesně, bych ji nemohl doporučit pro použití při dalších výpočtech. Funkce přirozený logaritmus počítá korektně a mohu ji doporučit pro použití při dalších výpočtech. U radarů se neprovádí počítání žádných dlouhých řad s využitím odchylky, takže tyto výsledky také považuji za korektní.

Radary by se jistě daly rozšířit o další výpočty. Potom by byly schopné lepšího uplatnění v praxi. Pravděpodobně by vracely trochu jiné údaje, než je v zadání.

Program je přenositelný mezi většinou používaných operačních systémů. Byly na něm provedeny testy zmíněné výše a program je spočítal správně.

# Literatura

[1] BARTSCH, Hans-Jochen. *Matematické vzorce*. Přeložil Mg. Mat. Vladimír Malý. Praha : Státní nakladatelství technické literatury, 1963. 580 s.

# A Metriky kódu

**Počet souborů:** 1 soubor

**Počet řádků zdrojového textu:** 434 řádků

**Velikost statických dat:** 376 B

**Velikost spustitelného souboru:** 16463B (systém Linux, 32 bitová architektura)