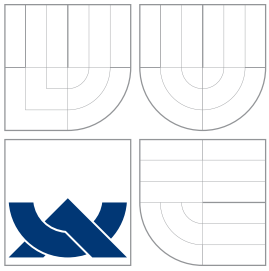
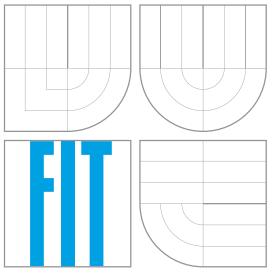


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VLIV PŘESNOSTI ARITMETICKÝCH OPERACÍ NA PŘESNOST NUMERICKÝCH METOD

NUMERICAL METHODS ACCURACY VS PRECISION OF ARITHMETIC

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. FRANTIŠEK KLUKNAVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Dr. Ing. PETR PERINGER
SUPERVISOR

BRNO 2012

Abstrakt

Práca je zameraná na hodnotenie vplyvu zaokrúhľovacích chýb na presnosť a efektivitu numerických integračných metód. Obsahuje teoretické predpoklady prebrané z existujúcej literatúry, implementáciu knižnice zvolených metód, experimentálne zisťovanie dosiahnutej presnosti za rôznych podmienok a ich porovnanie vzhľadom k časovej náročnosti. Knižnica obsahuje metódy Runge-Kutta do rádu 7 a Adams-Bashforth do rádu 20 implementované pomocou C++ šablón, ktoré dovoľujú použiť voliteľnú aritmetiku s viacnásobnou presnosťou. Na experimenty boli použité jednoduché modely so známym analytickým riešením.

Abstract

Thesis is dedicated to evaluation of roundoff impact on numerical integration methods accuracy and effectivity. Contains theoretical expectations taken from existing literature, implementation of chosen methods, experimental measurement of attained accuracy under different circumstances and their comparison with regard to time complexity. Library contains Runge-Kutta methods to order 7 and Adams-Bashforth methods to order 20 implemented using C++ templates which allow optional arbitrary-precision arithmetic. Small models with known analytic solution were used for experiments.

Klíčová slova

obyčajné diferenciálne rovnice, ODE, spojitá simulácia, presnosť simulácie, efektivita simulácie, aritmetika s viacnásobnou presnosťou, Runge-Kutta, Adams-Bashforth

Keywords

ordinary differential equations, ODE, continuous simulation, simulation accuracy, simulation effectivity, arbitrary precision arithmetic, Runge-Kutta, Adams-Bashforth

Citace

František Kluknavský: Vliv přesnosti aritmetických operací
na přesnost numerických metod, diplomová práce, Brno, FIT VUT v Brně, 2012

Vliv přesnosti aritmetických operací na přesnost numerických metod

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Petra Peringera. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
František Kluknavský
21. mája 2012

Poděkování

Ďakujem svojmu školiteľovi, Dr. Ing. Petrovi Peringerovi, za odborné vedenie, ochotu, cenné rady a usmernenia pri vypracovaní diplomovej práce.

Obsah

Kapitola 1

Úvod

Numerické metódy sú postupy riešenia matematických problémov, ktoré nedávajú presnú všeobecne platnú odpoveď, ale približnú číselnú hodnotu za konkrétnych podmienok. Používajú sa na rôzne účely tam, kde je analytické riešenie neexistujúce alebo nepraktické, napríklad na optimalizáciu, hľadanie koreňov rovníc, interpoláciu, kvadratúru alebo simuláciu. V tejto práci sa zameriame na simulačné riešenie obyčajných diferenciálnych rovníc.

Presnosť každej numerickej metódy je obmedzená jednak samotným princípom metódy, jednak presnosťou použitej aritmetiky. Keďže kritickou vlastnosťou simulácií býva ich rýchlosť, používa sa aritmetika zabudovaná v procesoroch počítačov. Na niektoré úlohy však nestačí. Ak je to nutné, presnejšia operácia sa dá získať zložením výsledkov niekoľkých menej presných. V súčasnosti sa simulácie s použitím takejto viacnásobnej presnosti aritmetiky v praxi prirodzene používajú v oblastiach, ktoré vyžadujú extrémnu presnosť, napríklad dlhodobé simulácie, veľkoplošné simulácie, sledovanie niektorého mikroskopického detailu v simulácii alebo simulácie chaotických systémov.

Neexistuje teória, ktorá by všeobecne dokázala predpovedať najvhodnejšiu simulačnú metódu a najvhodnejšiu presnosť aritmetiky. Súvisí to s tým, že nie je známa univerzálna automatizovaná metóda zisťovania dosiahnutej presnosti ľubovoľného výpočtu s pohyblivou desatinnou čiarkou [1]. Na základe skúseností s konkrétnymi simuláciami existujú odporúčania, ale tie platia iba v konkrétnych špecializovaných oblastiach a niekedy sú nekonzistentné. Táto práca sa venuje niekoľkým jednoduchým modelom so známym analytickým riešením. S týmito sa najčastejšie stretávajú študenti pri výuke modelovania dynamických systémov a spojitej simulácie. Dôraz je na sledovaní zovšeobecniteľných závislostí.

Úvodné kapitoly sa budú venovať teoretickým predpokladom, vo svetle ktorých potom budeme interpretovať výsledky experimentov. Ako to vyjadril Richard Hamming v knihe venovanej numerickým metódam [1] : „Zmysel počítania je pochopenie, nie číslo."

Ďalej bude nasledovať časť venovaná implementácii knižnice numerických integračných metód. Existujúcich nástrojov je mnoho, potreba vyvinúť nový pochádza zo špecifických požiadaviek. Okrem nastaviteľnej presnosti aritmetiky ide hlavne o vhodnú sadu metód. V súčasnosti býva zvykom používať metódy s adaptívnym krokom, preto prevažujú v existujúcich nástrojoch. Ich nevýhodou je, že v extrémnych prípadoch, keď sa prejaví nedostatočná presnosť základných aritmetických operácií, mechanizmus regulácie dĺžky kroku zlyhá. Práve vplyv zaokrúhlenia je predmetom záujmu tejto práce, preto potrebujeme metódy s pevným krokom. Ďalšia požiadavka je mať metódy rôznych, hlavne vysokých, rádov, lebo práve metódy vysokých rádov na svoje správne fungovanie vyžadujú vysokú presnosť aritmetiky. Z existujúcich simulačných knižníc stojí za zmienku Odeint (*www.odeint.com*), ktorý dovoľuje použiť viacnásobnú presnosť, lenže neobsahuje vhodnú sadu metód.

Posledná kapitola sa bude venovať experimentom a interpretácii nameraných výsledkov.

Kapitola 2

Základné pojmy z oblasti modelovania a simulácie

Cieľom tejto kapitoly je vytvorenie stručného prehľadu najdôležitejších základných pojmov a definícií z oblasti modelovania a simulácie spojitých dynamických systémov, ktoré boli použité v tejto dipomovej práci. Špecializovaná literatúra na túto tému je napríklad [2].

Systém „je z filozofického hľadiska relativne stabilná, usporiadaná množina elementov a vzťahov, ktorá je charakterizovaná pomocou existencie určitých zákonov t.j. pomocou všeobecne potrebných a podstatných súvislostí" [3]. Z pohľadu tejto práce sa pod pojmom *dynamický systém* rozumie „matematický opis časovo-závislého procesu, ktorý je vzhľadom k času homogénny, teda jeho časový priebeh je síce závislý od počiatočných podmienok avšak je nezávisý od počiatočného času" [4]. Autorstvo tejto modernej definície dynamického systému patrí matematikovi G. D. Birkhoffovi. Iné definície nevyžadujú homogenitu voči času a vymedzujú osobitnú kategóriu *autonómnych systémov*.

Autonómny dynamický systém Ak budúci stav systému závisí iba od súčasného stavu, ale nezávisí od aktuálneho času, potom sa systém označuje ako autonómny (v anglickej literatúre sa používa pojem *time-invariant* alebo *autonomous*). Pre každý dynamický systém existuje autonómny model. Stačí pridať jeden rozmer stavového vektora, ktorý bude predstavovať aktuálny čas. Preto v ďalšom texte môžeme bez straty všeobecnosti predpokladať autonómne systémy.

Experiment je metóda analýzy systému, pri ktorej sa cielene overuje určitý predpoklad alebo domienka o vlastnosti systému, pričom sa na základe získaných údajov predpoklad potvrdí alebo vyvráti. Cieľom experimentu je teda zisťovanie a analýza vlastností systému.

Model M je taký systém, ktorým môžeme nahradiť v experimente iný systém S a získanými údajmi zodpovedať otázku o S. Pojem model sa vždy vzťahuje k nejakému systému a experimentu. Napríklad drevený model človeka môžeme použiť na skúšanie ľudských šiat, ale nemôžeme ho použiť na zisťovanie horľavosti ľudí. Táto práca sa venuje výhradne experimentom s modelmi. Preto budeme pod pojmom systém vždy rozumieť model (každý model bude zároveň systémom diferenciálnych rovníc).

Simulácia je experiment s použitím modelu namiesto originálneho systému.

Vlastnosti dynamického systému sa menia pôsobením času. Vlastnosti v jednom konkrétnom čase sú charakterizované *stavom systému*. Aktuálny stav systému sa uchováva pomocou *stavových premenných* pričom množinu všetkých stavových premenných označujeme *stavový vektor systému* alebo skrátene *stav*.

Dynamický systém môžeme modelovať pomocou množiny možných stavov a funkcie, ktorá z aktuálneho času, stavu a vstupu určí budúci stav a výstup systému. Množinu možných stavov voláme aj *stavový priestor*.

Dynamický systém a jeho model majú niektoré spoločné vlastnosti, ktoré sa v čase vyvíjajú podobným spôsobom. Príklad: Skákajúca pružná loptička má v každom momente nejakú výšku nad zemou. Matematický alebo počítačový model skákajúcej loptičky v doslovnom zmysle výšku nad zemou nemá, ale má nejakú vnútornú číselnú hodnotu, ktorá sa časom mení rovnako, ako sa mení výsledok merania výšky skutočnej loptičky. Proces, ktorým sa overuje takáto analógia medzi systémom a jeho modelom, alebo sa zisťuje rozsah podmienok, za ktorých analógia platí, sa nazýva *validácia modelu*. V tejto práci nebude dôležitá podobnosť modelu so skutočnosťou, iba podobnosť rôznych simulácií modelu, preto budeme predpokladať validitu použitých modelov a ďalej sa ňou nebudeme zaoberať.

Spojitý dynamický systém Každá zložka stavového vektora môže byť spojitá alebo diskrétna. Ak uvažujeme, že je čas spojitý (reálne číslo) a každá zložka stavového vektora sa v čase mení spojito, potom sa jedná o *spojitý dynamický systém*.

2.1 Modelovanie dynamických systémov

Spojitý dynamický systém môžeme zapísať formou sústavy diferenciálnych rovníc. Pre rôzne typy systémov sa používajú typicky tieto základné typy rovníc:

- *Obyčajné diferenciálne rovnice prvého rádu.* Všeobecný tvar sústavy obyčajných diferenciálnych rovníc prvého rádu je možné vyjadriť vzťahom

$$\overrightarrow{y'(t)} = f(\overrightarrow{y(t)}) \tag{2.1}$$

  pričom

$$\overrightarrow{y(t)} = \{y_k(t);\ k = 1, 2, ..., K\}$$

  je vektor stavových premenných,

$$\overrightarrow{y'(t)} = \{y'_k(t);\ k = 1, 2, ..., K\}$$

  je vektor derivácií stavových premmenných podľa času s $y'_k(t) = dy_k(t)/dt$ a $f(\overrightarrow{y(t)})$ je ľubovoľná funkcia nad vektorom stavových premenných. V týchto rovniciach vystupujú iba prvé derivácie zložiek stavového vektora podľa času. Z typov diferenciálnych rovníc sú najjednoduchšie, zato veľmi univerzálne. Modelujú sa nimi systémy konečných rozmerov. Ostatné typy diferenciálnych rovníc sa dajú vyjadriť alebo aspoň aproximovať obyčajnými diferenciálnymi rovnicami prvého rádu [5]. V tejto práci sa pri modelovaní systémov bude vychádzať práve z tohto druhu zápisu.

- *Obyčajné diferenciálne rovnice vyššieho rádu.* Sústava differenciálnych rovníc N-tého rádu ($N > 1$) obsahuje vyššie derivácie stavových premenných podľa času až do rádu

N t.j. $y_k^{(i)}(t)$, $i = 1, 2, ..., N$. Dá sa ukázať, že takáto sústava rovníc sa dá plnohodnotne previesť na rovnice prvého rádu (2.1), napríklad metódou znižovania rádu derivácie alebo metódou postupnej integrácie [6].

- *Diferenciálno-algebraické rovnice.* Sústavy diferenciálnych a algebraických rovníc sa všeobecne nedajú jednoduchými úpravami prepísať do tvaru (2.1). Obvykle sa dajú do tohto tvaru dostať preformulovaním problému, alebo sa dajú použiť simulačné metódy, ktoré dokážu riešiť diferenciálno-algebraické rovnice priamo.

- *Parciálne diferenciálne rovnice.* Sústavy parciálnych diferenciálnych rovníc obsahujú nie len derivácie podľa času, ale aj derivácie podľa niektorej zložky stavového vektora. Stavový vektor má preto nekonečný počet zložiek a úloha sa nedá priamo preformulovať do tvaru (2.1). Dá sa obyčajnými diferenciálnymi rovnicami aproximovať, ak spojitý priestor stavových premenných diskretizujeme a derivácie podľa stavu nahradíme diferenciami. Tento spôsob sa volá metóda priamok (*method of lines* – MOL [5]). Parciálnymi diferenciálnymi rovnicami sa najčastejšie popisujú priestorové javy ako napríklad prúdenie tekutín alebo šírenie mechanického napätia.

Kapitola 3

Aritmetika reálnych čísel v počítači

Pri riešení numerických metód sa na dnešných počítačoch používa spravidla aritmetika s *pohyblivou desatinnou čiarkou* (iným slovom *plávajúcou*) [7, 1]. Pevná desatinná čiarka je nevhodná, pretože medzivýsledky výpočtov často prekročia dynamický rozsah použitej reprezentácie čísel a preto je nutné pracne implementovať prispôsobovanie rozsahu čísel. Použitie racionálnych čísel vo forme zlomku dvoch celých čísel je tiež nevhodné pre neudržateľný rast pamäťovej náročnosti v prípade nárastu zložitosti problému. Čísla s pohyblivou desatinnou čiarkou budeme skrátene volať *reálne čísla*, a to aj napriek tomu, že sa v princípe jedná o čísla racionálne.

Hlavnou nevýhodou reálnych čísel je *chyba zaokrúhlením* (*roundoff error*). Tento efekt nastane vtedy, keď v zvolenej aritmetike v dôsledku obmedzeného rozlíšenia neexistuje číslo, ktoré presne reprezentuje výsledok operácie.

Reprodukovateľnosť výpočtu

Dôležitou vlastnosťou aritmetiky je reprodukovateľnosť každého výpočtu t.j. výsledok operácie je deterministický a nezávislý na použitom počítači. To umožňuje dobrú prenositeľnosť programov. Tiež umožňuje, aby sa zaujímavý výsledok nejakého výpočtu vždy dal detailnejšie preskúmať, a to aj s odstupom času, keď pôvodný počítač už nie je k dispozícii.

Absolútna reprodukovateľnosť nie je pre praktické používanie reálnych čísel vždy nevyhnutná. Dôležité je, aby bola presnosť výpočtu v určitom limite, predpovedateľná a kontrolovateľná [17]. Spravidla je výhodné, ak je skutočná presnosť výpočtu lepšia, ako požadovaná. Ak počítač dokáže produkovať presnejšie výsledky v rovnakom čase, nebýva dôvod to nevyužiť. V tejto práci však budeme vyžadovať prísnejšiu podmienku absolútnej reprodukovateľnosti. Aby sme mohli experimentálne overovať vplyv presnosti aritmetiky je dôležité, aby bola požadovaná presnosť aritmetických operácií úplne pod kontrolou.

3.1 IEEE 754

IEEE 754 [14] je norma pre implementáciu aritmetiky reálnych čísel. Túto normu spĺňajú takmer všetky dnešné procesory [17]. Norma definuje:

- internú reprezentáciu reálnych čísel a špeciálnych hodnôt,
- kódovanie reálnych čísel do reťazcov pri komunikácii,
- pravidlá zaokrúhľovania,

- aritmetické operácie,
- hlásenie chybových stavov.

Interná reprezentácia reálnych čísel

Reálne číslo sa dá vyjadriť pomocou *znamienka*, *mantisy* a *exponentu* vo forme

$$x = s.m.b^e$$

kde:

- s je znamienko čísla s hodnotou $s = (-1)^S$, pričom $S = 0$ pre pozitívne čísla a $S = 1$ pre negatívne čísla.
- m je mantisa s hodnotou $1 \leq m < 2$,
- b je základ exponentu s hodnotou 2,
- e je exponent.

Znamienko reprezentuje znamienko mantisy a tým aj polaritu celého čísla. Nula tiež môže mať kladné alebo záporné znamienko, ale kým nedôjde k chybe, tieto dve nuly sú ekvivalentné. Mantisa obsahuje platné číslice reálneho čísla. *Šírka mantisy* (počet bitov) udáva, na koľko platných číslic sa zaokrúhli výsledok výpočtu. Presnosť pri reprezentácii čísla preto závisí hlavne od šírky mantisy. Najbežnejšie používané šírky mantisy sú:

- 24 bitov – jednoduchá presnosť, *float, single, binary32*,
- 53 bitov – dvojnásobná presnosť, *double, binary64*.

Exponent udáva, kde sa nachádza rádová čiarka čísla[1].

Špeciálne nečíselné hodnoty

Okrem formátov číselných hodnôt definuje norma nasledujúce nečíselné hodnoty:

- *Inf* – nekonečno (infinity). Číslo príliš veľké, pre ktoré nestačí rozsah exponentu.
- *-Inf* – záporné nekonečno.
- *Nan – not a number* – žiadne číslo. Výsledok nedefinovanej operácie, napríklad delenia nulou alebo odmocniny záporného čísla.

Raz vzniknutá nečíselná hodnota sa šíri ďalšími aritmetickými operáciami. Ak je vstupom operácie, výsledkom by tiež mala byť niektorá nečíselná hodnota, nie obyčajné číslo. To uľahčuje odhalenie a diagnostikovanie chybného výpočtu. Nebezpečné je, že nečíslo sa nemusí šíriť, ak máme vo výpočte použité relačné a logické operácie, ako porovnanie na rovnosť. Napríklad vyhľadanie minima môže vrátiť najmenšie normálne číslo a skryť Nan, čím zamaskuje vzniknutú chybu. Preto IEEE 754 umožňuje signalizovať vznik nečíselných hodnôt, túto vlastnosť ale podporuje iba málo programovacích jazykov. Jednou z výnimiek je C99 s funkciami z `fenv.h`.

[1] Ekvivalent desatinnej čiarky, lenže nemusí oddeľovať jednotky a desatiny. Podľa použitej číselnej sústavy môže oddeľovať napríklad jednotky a polovice alebo tisícky a stovky.

Epsilon

Jeden *ulp* (anglicky *unit in the last place*) vyjadruje hodnotu najmenej významného bitu mantisy. Hodnota ulp závisí od šírky mantisy, ale aj od aktuálnej hodnoty exponentu. IEEE 754 vyžaduje, aby výsledok každej operácie bol presný na $\frac{1}{2}$ ulp, teda s najlepšou možnou presnosťou.

Epsilon ε je najväčšia relatívna chyba operácie s reálnymi číslami, čiže $\frac{1}{2}$ ulp v pomere k veľkosti čísla. Epsilon je rovné $\frac{1}{2}$ ulp, ak má reálne číslo absolútnu hodnotu 1. Pri experimentálnom určovaní sa dá využiť vlastnosť, že ε je najväčšie možné číslo, pre ktoré po zaokrúhlení na príslušnú šírku mantisy platí $\varepsilon + 1 = 1$. Táto skutočnosť bude využitá v experimentálnej časti práce pre označenie hranice dosiahnuteľnej presnosti výpočtov.

3.2 Zdroje výpočtových chýb

Chyba zaokrúhlením môže vzniknúť v rôznych situáciách. Niektoré sa ale často opakujú a často spôsobujú nezanedbateľnú stratu presnosti [1].

Šírenie chyby

Epsilon je najväčšia chyba jednej elementárnej operácie za predpokladu, že operandy sú presné. Pri zložitejších výpočtoch však dochádza k tzv. *šíreniu chyby* (*error propagation*), keď niektoré operácie pracujú s operandami zaťaženými chybou predošlých operácií. V takom prípade môže chyba konečného výsledku rásť v závislosti od počtu, postupnosti a druhu operácií logaritmicky, lineárne alebo dokonca exponenciálne. Výpočty s exponenciálnym rastom chyby sú numericky nestabilné.

Numerická nestabilita

Výpočet, v ktorom sa aj malá neistota niektorého čísla (spôsobená napríklad zaokrúhlením) prejaví veľkou neistotou výsledku je numericky nestabilný[10]. Chyba je pri nestabilnom výpočte mnohonásobne zväčšená. Napríklad vo výraze

$$y = \left(\sqrt[n]{x}\right)^n, n = 128 \tag{3.1}$$

ktorý vyčíslime opakovaním druhej odmocniny a druhej mocniny bude výsledok pre akékoľvek $x > 1$ bude výsledok s použitím aritmetiky reálnych čísel s bežnou presnosťou (pravdepodobne) 1, čiže nesprávny. Stane sa tak aj napriek tomu, že výpočet sa skladá z pomerne malého počtu umocňovaní a odmocňovaní, každé s relatívnou chybou maximálne ε. Výsledok $\sqrt[128]{x}$ je tak blízky jednotke, že všetky ďalšie platné číslice sa zaokrúhlením stratia a vznikne rovná 1. Je to akceptovateľne presný výsledok s prihliadnutím na šírku mantisy. $1^{128} = 1$, to je tiež primerane presný výsledok. Ak však tieto dve operácie spojíme, dostaneme nezmyselné $\left(\sqrt[128]{10}\right)^{128} \approx 1$. Chyba, ktorá vznikla odmocnením a zaokrúhlením sa pri neskoršom umocnení mnohonásobne posilní.

Sčítanie malého a veľkého čísla

Ak sčítame veľké číslo v s malým číslom m, pričom pre exponenty čísel platí $e_v \gg e_m$, dochádza v dôsledku zarovnania desatinnej čiarky pred samotným sčítaním k "*vysunutiu*" menej významových bitov menšieho čísla zo zobraziteľnej časti mantisy a teda aj k orezaniu

ich príspevku k výsledku po zaokrúhlení. Tento efekt sa anglicky nazýva *shiftout* [1]. V extrémnom prípade, keď $m < \frac{ulp(v)}{2}$, bude z dôvodu zaokrúhľovania platiť $m + v = v$, čím sa vplyv malého čísla m na výsledku vôbec neprejaví. Samo o sebe to nie je veľký problém, výsledok má stále relatívnu chybu najviac ε.

Uvažujme teraz o probléme *shiftout* v inom kontexte. Ak sčítame veľký počet malých čísel m, pričom ich je tak veľa, že súčet presiahne v, potom pripočítaním ľubovoľného počtu ďalších m sa už súčet nezmení. Sčítanie množiny čísel je preto možné iba do určitého počtu sčítancov závislého od šírky mantisy a od štatistického rozloženia sčítancov. Prekročením tejto hranice sa výsledok stane nezmyselným. Obmedziť tento efekt sa dá okrem rozšírenia mantisy aj vylepšeným algoritmom sčítania, napríklad sčítaním po dvojiciach (*pairwise summation*) alebo Kahanovým sčítaním (*Kahan summation* [8]). Kahanove sčítanie sa snaží určiť veľkosť zaokrúhlenia a pripočítať ho ako korekciu k nasledujúcemu sčítancu. Pri sčítaní po dvojiciach (princíp binárneho stromu) rozdelíme rad sčítancov na dvojice a tieto dvojice sčítame nezávisle na ostatných dvojiciach. Postup opakujeme na výsledkoch predchádzajúceho sčítania, až kým nezískame jediné výsledné číslo. Pre ďalšie zlepšenie presnosti výsledku je môžné rad sčítancov usporiadať podľa veľkosti. Tak zabezpečíme, aby sa sčítavali vždy "najbližšie" čísla, teda čísla s čo najmenším rozdielom hodnôt.

S efektom *shiftout* sa stretávame napr. pri simulácii, kde sa problém sčítania rôzne veľkých čísel často objavuje vo výpočte aktuálneho času. Výpočet typu $t_n = t_{n-1} + \Delta t$ vedie k veľkému akumulovaniu chyby v dôsledku narastania rozdielu medzi t_{n-1} a Δt a preto je výhodnejší algoritmus $t_n = t_0 + n\Delta t$.

Iný prípad výskytu *shiftout* v súvislosti so simuláciou spojitých systémov nastane, keď sa pripočítava zmena stavu systému počas Δt k aktuálnemu stavu, čo pri malom Δt môže viesť k silnému *shiftout*. Stretneme sa s tým v experimentálnej časti tejto práce.

Odčítanie dvoch podobných čísel

Pri násobení a delení platí, že ε sa vzťahuje k veľkosti výsledku. Pri sčítaní a odčítaní je to chyba relatívna voči väčšiemu zo sčítancov. Ak odčítame dve podobné čísla, absolútna hodnota rozdielu je veľmi malá a chyba, ktorá vznikne, je relatívne malá voči vstupným číslam, ale veľká voči výsledku. Najvýznamnejšie bity sa vo výsledku vzájomne vynulujú, čím zostane v reprezentácii čísla iba málo platných číslic. Tento efekt sa anglicky nazýva *cancellation*.

Pretečenie a podtečenie

Pretečenie (*overflow*) a podtečenie (*underflow*) nastáva, keď hodnota čísla presiahne rozsah exponentu. Ak je výsledok operácie príliš veľký, nastane pretečenie. Výraz nadobudne špeciálnu hodnotu "nekonečno". V prípade, že sa výsledok blíži k hodnote nula, nastane podtečenie. Vznikne *denormalizované číslo* (prvá číslica mantisy nie je 1) s menšou presnosťou, než je plná šírka mantisy, lebo najvýznamnejší bit (alebo niekoľko najvýznamnejších bitov) je rovný nule. Pri ďalšom zmenšovaní hodnoty sa číslo zaokrúhli na nulu.

Ak dôjde k pretečeniu alebo podtečeniu, obvykle nepomáha zväčšiť rozsah exponentu aritmetiky, pretože často je chyba v nevhodnej štruktúre výpočtov.

Význam denormalizovaných čísel je sporný. Namiesto prudkej straty presnosti zaokrúhlením na nulu sa presnosť stráca postupne, lebo mantisa denormalizovaného čísla stále obsahuje niekoľko platných číslic. To zlepší presnosť výpočtu. Na druhej strane denormalizované číslo môže sťažiť detekciu chyby, ktorú by zaokrúhlenie na nulu zviditeľnilo.

Deliteľ blízky nule

Predchádzajúce efekty nastávajú najčastejšie vtedy, keď je jeden z medzivýsledkov rádovo oveľa väčší ako ostatné. Obrovský medzivýsledok je zas najčastejšie spôsobený deliteľom blízkym nule.

Singularita

Singularitou je bod, v ktorom sa funkcia chová inak ako v bežne používanej oblasti. Presná definícia závisí od kontextu, napríklad:

- funkcia je v tomto bode nedefinovaná
- je v tomto bode nespojitá
- je v tomto bode nediferencovateľná
- funkčná hodnota sa blíži nekonečnu

Problémom výpočtov v oblasti singularity nebýva iba presný bod singularity, ale aj jeho okolie. Všeobecne platí, že čím presnejšiu aritmetiku používame, tým bližšie sa môže nachádzať hodnota vstupných dát k singularite. Napríklad funkcia (3.1) má singularitu v $n \to \infty$, pričom 128 je už príliš blízke nekonečnu.

Zisťovanie presnosti výpočtov

Univerzálny spôsob, ako zistiť veľkosť chyby akéhokoľvek výpočtu s reálnymi číslami nie je známy. Príkladom často používaných techník, ktoré spravidla, ale nie vždy, umožňujú kvantifikovať chybu výpočtu, patrí:

- Porovanie s výpočtom pomocou iného algoritmu.
- Porovanie s výpočtom so zašumenými vstupmi. Tak sa niekedy ukáže neprimerane veľká citlivosť na malú zmenu čísel. Vyžaduje to mnohonásobné opakovanie výpočtu a štatistické spracovanie výsledkov.
- Porovanie s výpočtom s iným spôsobom zaokrúhľovania. Zašumenie vstupu sa môže v zložitejšom výpočte z dôvodu zaokrúhľovania stratiť a neovplyvniť neskoršie fázy výpočtu. Naproti tomu zmena zaokrúhľovania vplýva na všetky medzivýpočty. Problémom tejto metódy býva slabá podpora rôznych spôsobov zaokrúhľovania v programovacích jazykoch.
- Porovanie s výpočtom s inou presnosťou aritmetiky.
- Použitie *intervalovej aritmetiky* [9].

V intervalovej aritmetike je každé číslo reprezentované dvojicou hodnôt – vrchnou a spodnou hranicou. Každá aritmetická operácia sa vykoná dvakrát, pričom výsledok pre vrchnú hranicu sa zaokrúhľuje nahor, výsledok pre spodnú hranicu nadol. Tým sa zabezpečí, aby skutočný nezaokrúhlený výsledok ležal vo vnútri intervalu. V integračných metódach sa intervalová aritmetika väčšinou nepoužíva z niekoľkých dôvodov:

- Zohľadňuje iba chybu zaokrúhlením, nezohľadňuje chybu orezaním, pričom chyba orezaním býva nezanedbateľná.

- Nezohľadňuje stabilitu metódy, ktorá zaručí, že vplyv každej jednotlivej chyby sa časom zmenšuje. Výsledný interval je preto mnohonásobne širší, než je nutné.
- Presná hodnota nebýva rozložená rovnomerne okolo stredu intervalu. Napríklad keď umocníme neznáme číslo 0,5 o ktorom vieme iba interval (0, 1), dostaneme interval (0, 1), ale správna nezaokrúhlená hodnota 0,25 je systematicky mimo stredu intervalu.

Pre zmenšovanie zaokrúhľovacích chýb sa z pohľadu tejto práce najviac hodí použitie rôznych algoritmov a presnejšej aritmetiky.

3.3 Knižnice pre výpočty s ľubovoľnou presnosťou

Ľubovoľná presnosť (v anglickej literatúre *arbitrary precision*, alebo aj *multiprecision, multiple precision, bignum, infinite precision*) znamená, že počet bitov aritmetiky nie je obmedzený aritmetickou jednotkou použitého počítača. Jedna operácia s číslom viacnásobnej presnosti sa interne skladá z viacerých operácií obyčajnej presnosti, alebo z viacerých celočíselných operácií.

Problémom reálnych čísel s ľubovoľnou presnosťou je, že neexistuje všeobecne uznávaný štandard podobný IEEE-754. Užívateľ si musí byť vedomý dôsledkov špecifikácie svojej konkrétnej knižnice.

V tejto časti budeme uvažovať iba o knižniciach s rozhraním pre jazyky C/C++, ktoré priamo poskytujú výpočty základných aritmetických operácií aj bežnejších transcendentálnych funkcií (sínus, kosínus, exponenciála, logaritmus) pre reálne čísla. Vzhľadom na požiadavky, ktoré sú kladené na knižnicu numerických metód nie je nutné, aby bola presnosť aritmetiky nastaviteľná dynamicky počas behu programu. Postačujúca je aj staticky nastaviteľná presnost operandov v dobe kompilácie.

Pri experimentoch, kde nie je potrebné nastavovať presnosť aritmetiky, budeme z dôvodu šetrenia výpočtovým časom používať jednoduchú presnosť IEEE 754 na simuláciu, dvojnásobnú presnosť na výpočet referenčného riešenia a chyby. Matematické knižnice pri rôznych operačných systémoch a kompilátoroch nie sú rovnakej kvality a nedávajú rovnaké výsledky trigonometrických funkcií. Je preto dôležité poznať vlastnosti a obmedzenia konkrétnej matematickej knižnice pri jej aplikácii.

Príkladom knižníc pre výpočty s ľubovoľnou presnosťou sú:

- GMP *(gmplib.org)*
- MPFR (*www.mpfr.org*, [13])
- apfloat (*www.apfloat.org*)
- hfloat (*www.jjj.de/hfloat/*)
- BSDMP (*bsdmp.org*)
- arprec (*crd-legacy.lbl.gov/~dhbailey/mpdist/*)
- CLN (*www.ginac.de/CLN/*).

GMP

Názov GMP je skratkou anglického názvu "*The GNU Multiple Precision Arithmetic Library*". Je navrhnutá s dôrazom na rýchlosť. Používa rôzne algoritmy optimalizované pre konkrétne veľkosti operandov a konkrétnu architektúru počítača.

Medzi jej ďalšie výhody patrí:

- Škálovateľnosť. Nemá žiadne obmedzenia na presnosť operandov okrem veľkosti pamäte počítača.
- Otvorenosť. Je voľne dostupná pod slobodnou licenciou LGPL.
- Prenositeľnosť. Funguje pod viacerými operačnými systémami Unix-ového typu vrátane Linuxu, aj pod Windows v 32- aj 64- bitovom móde.
- Spoľahlivosť. Má pomerne veľkú komunitu používateľov a vývojárov, čo naznačuje dobré testovanie [13].

Medzi nevýhody patrí:

- Reprodukovateľnosť výpočtov nie je zaručená. GMP garantuje, že *minimálna* požadovaná presnosť bude dodržaná, skutočná presnosť môže byť iná.
- Nízkoúrovňové rozhranie. Je potrebných veľa úprav hotového programu, kým sa môže GMP používať pri výpočtoch namiesto štandardných dátových typov jazyka C. GMP obsahuje aj objektovo orientované C++ rozhranie, ale to je zatiaľ v experimentálnom štádiu. Samotní autori odporúčajú pri vývoji nových programov nepoužiť ho a radšej použiť MPFR.

Pre tieto nevýhody nie je GMP priamo vhodná na experimenty s voliteľným počtom bitov, ale jej rýchle a spoľahlivé algoritmy sú základom viacerých ďalších knižníc.

MPFR

Knižnica MPFR ("Multiple-precision Binary Floating Point Library with Correct Rounding") sa snaží aplikovať myšlienky štandardu IEEE-754 na čísla viacnásobnej presnosti. Je vytvorená ako nadstavba knižnice GMP. Výhody spomínané pri GMP platia aj pre MPFR. Okrem toho zaručuje MPFR aj korektné zaokrúhľovanie a reprodukovateľnosť bez ohľadu na použitý procesor a operačný systém. Podporuje viac transcendentálnych funkcií, rôzne módy zaokrúhľovania a špeciálne hodnoty reálnych čísel: NaN, kladné a záporné nekonečno.

Hlavnou nevýhodou je opäť nízkoúrovňové užívateľské rozhranie. Nasleduje príklad použitia knižnice MPFR prebraný z [15]:

```
void schwefel(mpfr_t y, mpfr_t x)
{
  mpfr_t t; mpfr_init(t);
  mpfr_abs(t,x,GMP_RNDN);
  mpfr_sqrt(t,t,GMP_RNDN);
  mpfr_sin(t,t,GMP_RNDN);
  mpfr_mul(t,t,x,GMP_RNDN);
  mpfr_set_str(y,"418.9829",10,GMP_RNDN);
  mpfr_sub(y,y,t,GMP_RNDN);
  mpfr_clear(t);
}
```

Preto vzniklo viacero alternatívnych rozhraní:

- mpfr C++ [15]
- mpfr::real (*chschneider.eu/programming/mpfr_real/*)
- mpfrcpp (*beshenov.ru/mpfrcpp/*)
- gmpfrxx (*math.berkeley.edu/~wilken/code/gmpfrxx/*)

Tieto sa odlišujú okrem architektúry aj rôznou filozofiou pri nastavovaní presnosti medzivýpočtov, takže výsledky pri použití rôznych rozhraní nemusia byť rovnaké.

Mpfr C++

Rozhranie, ktoré zabaľuje dátové typy MPFR do objektov. Aritmetické operácie sú dostupné pomocou preťažovania operátorov, takže umožňuje pohodlnú konverziu hotového algoritmu so štandardnými typmi jazyka C na algoritmus s použitím viacnásobnej presnosti. Obsahuje vlastný systém správy pamäte, ktorý má podľa autora zrýchliť prácu s medzivýsledkami v zložitejších výrazoch. Presnosť výpočtov je nastaviteľná dynamicky, presnosť medzivýsledkov je daná vždy maximom presnosti operandov.

Problémy pri použití knižnice:

- Automatická konverzia na typ double, ktorá môže spôsobiť vo výpočtoch neočakávanú stratu presnosti núti k veľkej opatrnosti pri zapisovaní výrazov. Automatickej konverzii sa dá zabrániť zakázaním príslušných funkcií v zdrojovom kóde knižnice.
- Slabá dokumentácia.

Licencia dovoľuje použiť Mpfr C++ v nekomerčných projektoch. Príklad použitia rozhrania mpfr C++ [15], rovnaký výpočet ako v predchádzajúcom algoritme:

```
mpreal schwefel(mpreal& x)
{
  return 418.9829-x*sin(sqrt(abs(x)));
}
```

Mpfr::real

je rozhranie podobné mpfr C++. Odlišuje sa svojou vnútornou architektúrou využívajúcou šablóny. Mpfr::real nastavuje presnosť dátových typov staticky už v dobe kompilácie. Pre použitie čísel s rôznou presnosťou je preto potrebná definícia rôznych dátových typov. Ich spoločné použitie v jednom výraze zabezpečujú konverzie. Táto filozofia presne zodpovedá návrhu knižnice, ktorá je predmetom tejto práce.

Vážnou nevýhodou mpfr::real je skoré štádium vývoja, čiže vyššie riziko chýb. Pri experimentoch v tejto práci boli dosiahnuté zhodné výsledky výpočtov v porovnaní s knižnicou mpfr C++. Textový vstup bol však v niektorých situáciách nefunkčný. Autor prisľúbil skorú nápravu a zdokumentovanie.

Príklad použitia rozhrania mpfr::real, rovnaký výpočet ako v predchádzajúcom algoritme:

```
mpfr::real<128> schwefel(mpfr::real<128>& x)
{
  return "418.9829"-x*sin(sqrt(abs(x)));
}
```

Apfloat a hfloat

Knižnice sú optimalizované pre použitie čísel s mimoriadne veľkou presnosťou (väčšie ako 100000 bitov). Na násobenie používajú iba algoritmy založené na FFT. Experimenty v tejto práci sa nezameriavajú na až tak presné čísla.

BSDMP

Názov knižnice vznikol zlúčením skratiek "BSD" a "GMP". Je to nový projekt, odštartoval v roku 2010. Snaží sa poskytnúť funkcionalitu GMP dostupnú pod licenciou BSD. Zatiaľ nebola vydaná žiadna oficiálna verzia vhodná na použitie. V budúcnosti by sa mohla stať dobrou náhradou za GMP.

CLN

je distribuovaná pod licenciou GPL, čo znemožňuje jej použitie v projektoch s nekompatibilnou licenciou.

Po zvážení všetkých vlastností knižníc ako aj v súlade s požiadavkami diplomovej práce boli pre účely implementácie knižnice numerických metód a experimentov zvolené knižnice GMP a MPFR s rozhraním mpfr::real.

Kapitola 4

Princípy numerických integračných metód

Presné analytické riešenia sústav diferenciálnych rovníc často neexistujú, alebo je príliš náročné ich zistiť. Preto sa v praxi používajú približné numerické riešenia. Ich princípom je z informácie o stave a o derivácii v tomto stave a/alebo v jeho blízkom okolí odhadnúť stav v blízkej budúcnosti. Z tohto odhadu sa potom vychádza pri odhade vzdialenejšej budúcnosti. Predpokladom je, že derivácia je vždy konečné reálne číslo a v čase sa mení spojito. Preto sa dá v malom časovom intervale považovať za približne konštantnú.

Chyba, ktorá vznikne pri jednom kroku takéhoto opakovaného výpočtu, sa nazýva *lokálna chyba*. *Globálnou chybou* sa označuje chyba, ktorá vznikne akumuláciou lokálnych chýb počas simulácie daného časového intervalu.

Podkapitoly sú zoradené tak, aby sa každá mohla oprieť o konkrétne príklady z predchádzajúcich podkapitol. Preto sú časti venované konkrétnym metódam a časti venované ich všeobecným vlastnostiam vzájomne premiešané. Ani táto kapitola si nekladie za cieľ kompletnosť ani formálnu korektnosť. Ide iba o načrtnutie hlavných myšlienok využitých neskôr v tejto práci. Prípadný záujemca má na výber mnoho kvalitnej špecializovanej literatúry, napríklad [5, 1].

V ďalšom texte budeme pod skrátenými pojmami "numerická metóda", "integračná metóda", "simulačná metóda" a "metóda" vždy rozumieť numerickú integračnú metódu na simuláciu modelov popísaných obyčajnými diferenciálnymi rovnicami, pretože tejto podmnožine numerických metód je venovaná celá práca.

Numerické metódy sa zvyknú deliť na skupiny podľa spoločných vlastností: explicitné a implicitné, čisto numerické a kombinované numericko-analytické, jednokrokové a viackrokové, metódy rôznym spôsobom zložené z iných metód.

4.1 Explicitné a implicitné metódy

Rozdiel medzi implicitnými a explicitnými metódami si vysvetlíme na konkrétnych príkladoch najstarších a najjednoduchších metód.

Eulerova metóda

Eulerova metóda je historickým pionierom v numerickom prístupe k riešeniu diferenciálnych rovníc. Výpočet je daný opakovaným použitím rovnice

$$y(t+h) \approx y(t) + hf(y(t)) \tag{4.1}$$

pričom $h > 0$ nazývame *dĺžka kroku*. Dĺžka kroku reprezentuje časový interval, v ktorom predpokladáme, že $f(y,t)$ je približne konštanta. Ide teda o po častiach lineárnu aproximáciu – krivka z bodu $y(t)$ do bodu $y(t+h)$ sa nahradí priamkou cez bod $y(t)$ so smerom $f(y,t)$.

Simulácia Eulerovou metódou je priamočiary proces. Výsledok každého kroku je závislý len na predchádzajúcom kroku a na derivácii. Derivácia je tiež závislá len na predchádzajúcom kroku. Vždy teda máme k dispozícii všetky potrebné údaje na ďalší výpočet. Metódy s touto vlastnosťou sa nazývajú *explicitné*.

Implicitná Eulerova metóda

Na rozdiel od obyčajnej (explicitnej) Eulerovej metódy, pri implicitnej metóde sa derivácia nevyhodnocuje v bode $y(t)$, ale v bode $y(t+h)$

$$y(t+h) \approx y(t) + hf(y(t+h)) \tag{4.2}$$

Výraz $y(t) + hf(y(t+h))$ sa nedá priamo vyčísliť. Na získanie nového stavu potrebujeme deriváciu. Na získanie derivácie ale potrebujeme nový stav. Narazili sme na algebraickú slučku. Metódy s touto vlastnosťou sa nazývajú *implicitné*. Na ich riešenie sa dá použiť numerické hľadanie koreňov nelineárnych rovníc, najčastejšie Newtonova metóda. Pre nás je teraz dôležité riešenie pomocou prediktora: spojenie implicitnej a explicitnej metódy. Výsledná metóda je explicitná, ale zachováva si niektoré dobré vlastnosti implicitných metód. Zovšeobecnením tohto princípu sú metódy Runge-Kutta.

Ešte predtým potrebujeme zaviesť niekoľko dôležitých pojmov, na čo využijeme Taylorov rad.

4.2 Numerické a poloanalytické metódy

Predchádzajúce metódy boli čisto numerické. Výpočty sa týkali iba numerickej hodnoty stavu a derivácie. Poloanalytické metódy využívajú aj symbolické operácie. Príkladom takejto metódy je Taylorov polynóm.

Taylorov rad a Taylorov polynóm

Taylorov rad nekonečne diferencovateľnej funkcie $g(x)$ v okolí bodu a je:

$$g(x) = \frac{g(a)}{0!}(x-a)^0 + \frac{g'(a)}{1!}(x-a)^1 + \frac{g^{(2)}(a)}{2!}(x-a)^2 + \frac{g^{(3)}(a)}{3!}(x-a)^3 + \dots \tag{4.3}$$

V kontexte symboliky diferenciálnych rovníc označme $a = t$, $g = y$, $x - a = h$, $y'(t) = f(y(t))$. Potom z rovnice (4.3) dostaneme:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{h^2}{2!}y^{(2)}(t) + \frac{h^3}{3!}y^{(3)}(t) + \dots \\ y(t+h) &= y(t) + hf(y(t)) + \frac{h}{2!}^2 f'(y(t)) + \frac{h^3}{3!}f^{(2)}(y(t)) + \dots \end{aligned}$$

Prvých n+1 členov Taylorovho radu sa nazýva *Taylorov polynóm* n-tého stupňa. Symbolickou operáciou je získanie vyšších derivácií funkcie $f(y(t))$. V tejto práci nebudeme skúmať presné podmienky, za ktorých rad konverguje a uvedená rovnosť platí. Bližšie informácie opäť napríklad v [1].

Za predpokladu približne rovnakých absolútnych hodnôt $f, f', f^{(2)} \ldots$ a $h < 1$ sa absolútna hodnota členov radu zmenšuje. Preto pri praktickom výpočte môžeme nahradiť Taylorov rad polynómom n-tého stupňa. Lokálna chyba, ktorej sa pri tom dopustíme, je úmerná h^{n+1}. Nazýva sa chyba orezaním alebo *truncation error*. V anglickej literatúre sa ale niekedy tým istým pojmom označuje aj to, čo tu nazývame *roundoff error* – chyba vzniknutá zaokrúhlením (orezaním) reálneho čísla. To spôsobuje nejasnosť vo významoch pojmov.

Čím menšia je dĺžka kroku h, tým rýchlejšie klesá význam členov vyšších rádov. Rozvoj môžeme obmedziť na menej členov a výpočet jedného kroku je rýchlejší. Zároveň však potrebujeme pre tú istú simuláciu viac krokov. Ak je lokálna chyba úmerná h^{n+1}pre $h \to 0$, integračnú metódu nazveme *metódou n-tého rádu*. Čisto numerické integračné metódy využívajú na dosiahnutie vyššieho rádu namiesto symbolického vyjadrenia vyšších derivácií ich nepriamu numerickú aproximáciu. Všimnime si, že rovnica 4.1 Eulerovej metódy sa zhoduje s Taylorovym polynómom prvého stupňa.

Dôležitý je aj vplyv presnosti aritmetiky. Čím vyšší stupeň Taylorovho polynómu, tým sú väčšie rozdiely v hodnote jednotlivých členov a silnejšie sa prejavuje zaokrúhľovanie. Od určitého stupňa už pridanie ďalších členov na numerickom výsledku vôbec neprejaví. Použitie konkrétnej presnosti aritmetiky, stupňa a dĺžky kroku teda musí byť zladené v závislosti od požadovanej presnosti riešenia [5]. Tento poznatok je hlavnou motiváciou celej diplomovej práce [1].

Taylorov polynóm sa používa pri analýze a porovnávaní integračných metód nie len preto, že jeho rád sa dá neobmedzene zvyšovať. Všetko, čo sa dá vypočítať kombináciou sčítania, odčítania, násobenia a delenia, je ekvivalent podielu dvoch polynómov. Existujú aj iné integračné metódy, ktoré využívajú napríklad trigonometrické alebo exponenciálne funkcie a na ich analýzu je Taylorov rad nevhodný. Napríklad pre spektrálne algoritmy založené na trigonometrických funkciách sa používa Fourierov rad.

4.3 Zložené metódy

Integračné metódy sa dajú skladať rôznymi spôsobmi, čím vznikajú nové metódy s novými vlastnosťami. Priblížime si metódy založené na princípe prediktor-korektor, pretože tieto vedú na metódy, ktorými sa budeme zaoberať v experimentálnej časti.

Metódy typu prediktor-korektor

Uvažujme explicitnú a implicitnú Eulerovu metódu. Ak do rovnice 4.2 dosadíme $y(t+h)$ z rovnice 4.1, dostaneme *modifikovanú Eulerovu metódu*:

$$y(t+h) \approx y(t) + hf(y(t) + hf(y(t)))$$

Celkovo sme získali explicitnú metódu druhého rádu. Explicitná Eulerova metóda bola použitá na odhad – predikciu hodnoty $y(t+h)$, implicitná Eulerova metóda potom tento

[1] Experimenty ukázali, že pri iných integračných metódach sa zvyšovanie stupňa prejavuje trochu inak než v [5]

odhad spresnila – korigovala. Možnosťou na ďalšie zlepšenie presnosti je opakovať korektor – využiť výsledok korektora ako prediktor pre ďalšiu iteráciu korektora. Výpočet sa ukončí, eď je rozdiel medzi dvoma iteráciami dostatočne malý. V praxi sa na zlepšenie chyby častejšie používa iný postup. Namiesto opakovania korektora sa celý výpočet zopakuje so skrátenou dĺžkou kroku. Ak je naopak rozdiel zbytočne malý, a teda presnosť pravdepodobne veľmi dobrá, krok sa predĺži.

Metódy Runge-Kutta

Explicitné metódy Runge-Kutta (RK) sú zovšeobecnením princípu prediktor-korektor [1]. Výpočet jedného kroku sa skladá z niekoľkých stupňov. Prvý stupeň je krok Eulerovou metódou. Každý ďalší stupeň využíva vážený priemer predchádzajúcich stupňov ako prediktor. Celkový výsledok je potom vážený priemer jednotlivých stupňov.

$$\begin{aligned} k_1 &= hf(y(t)) \\ k_2 &= hf(y(t) + a_{21}k_1) \\ k_3 &= hf(y(t) + a_{31}k_1 + a_{32}k_2) \\ k_4 &= hf(y(t) + a_{41}k_1 + a_{42}k_2 + a_{43}k_3) \end{aligned}$$

$$\vdots$$

$$y(t+h) \approx y(t) + b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4 + \ldots \tag{4.4}$$

Matice konštánt a a b identifikujú konkrétnu metódu. Zvyknú byť usporiadané do mnemotechnickej pomôcky zvanej Butcherova tabuľka (Butcher tableau, Butcher table)[12]:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_n & a_{n1} & a_{n2} & \ldots & a_{n,n-1} & \\ \hline & b_1 & b_2 & \ldots & b_{n-1} & b_n \end{array}$$

V tabuľke je ešte vektor c, ktorý vznikol súčtom riadkov matice a, pričom $c_i = a_{i1} + a_{i2} + \ldots$ Tento parameter je dôležitý pri neautonómnych (time-variant) systémoch. Udáva čas relatívne voči začiatku kroku, v ktorom sa vyhodnocuje derivácia ($k_i = hf(t + c_i, y(t) + a_{i1}k_1 + \ldots)$).

Maximálny rád, ktorý môže RK metóda dosiahnuť, je závislý od jej stupňa. Jednostupňová Runge-Kutta s Butcherovou tabuľkou

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

je vlastne explicitná Eulerova metóda, čiže prvého rádu. Dvojstupňová metóda

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 0 & 1 \end{array}$$

je modifikovaná Eulerova metóda druhého rádu. Podobne metóda tretieho rádu musí mať najmenej tri stupne, pre štvrtý rád štyri stupne. Potom lineárna závislosť končí. Metóda piateho rádu potrebuje šesť stupňov a minimálny počet stupňov ďalej rastie nelineárne. Nie je známy vzťah pre minimálny počet stupňov metódy ľubovoľného rádu, ale nie je horší ako kvadratický [11]. Závislosť rádu od stupňa RK metódy pre rády 1 až 7 je znázornený v tabuľke 4.1.

rád	1	2	3	4	5	6	7
stupeň	1	2	3	4	6	7	9

Tabuľka 4.1: Závislosť maximálneho dosiahnuteľného rádu metódy Runge-Kutta od jej stupňa.

Všeobecne pre daný rád a stupeň nie je len jedna metóda Runge-Kutta, môže ich existovať niekoľko. Používať sa zvyknú tie, ktoré majú absolútne hodnoty čísel v matici a Butcherovej tabuľky približne rovnaké, aby sa predišlo silnému zaokrúhľovaniu pri sčítaní nerovnakých čísel. Ďalším kritériom je počet núl a jednotiek v matici a. Jednotka ušetrí pri výpočte jednu operáciu násobenia, nula ušetrí jedno násobenie a jedno sčítanie.

Najznámejšia metóda typu Runge-Kutta je štvrtého rádu, nazývaná "klasická Runge-Kutta" alebo iba "Runge-Kutta" s Butcherovou tabuľkou

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

4.4 Jednokrokové a viackrokové metódy

Všetky doterajšie metódy vychádzali na začiatku každého kroku z jedinej informácie – okamžitého stavu. Po vypočítaní nového stavu všetky medzivýsledky zahodili a začali odznova. Naproti tomu viackrokové metódy na dosiahnutie vyššieho rádu využívajú extrapoláciu z niekoľkých minulých hodnôt.

Extrapolácia polynómom

Ak máme daných $n+1$ bodov v tvare $[t, y(t)]$ pričom $t \in \{t_0, t_1, ...t_n\}$, existuje práve 1 polynóm n-tého stupňa, ktorý prechádza týmito bodmi. Ak do všeobecného vzorca polynómu

$$y(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots + a_1 t + a_0 \tag{4.5}$$

dosadíme tieto body, dostaneme $n+1$ rovníc o $n+1$ neznámych $a_0, a_1, \ldots a_n$. Ak žiadne dva body nemali rovnaký čas t, sústava rovníc má práve 1 riešenie – koeficienty hľadaného polynómu.

Modifikovanú metódu určovania koeficientov aproximačného polynómu získame, ak pre zostavenie sústavy rovníc použijem okrem rovnice (4.5) aj derivovanú rovnicu polynómu

$$y'(t) = n a_n t^{n-1} + (n-1) a_{n-1} t^{n-1} + ... + a_1 \tag{4.6}$$

V tomto prípade dosadzujeme do (4.6) body $[t, y'(t)] = [t, f(t)]$ reprezentujúce derivácie stavových premenných. Podmienkou pre existenciu jednoznačného riešenia je, že pri

zostavovaní sústavy rovníc musíme použiť aspoň jeden známy bod s funkčnou hodnotou stavovej premennej$[t, y(t)]$. Tento postup sa dá zovšeobecniť aj na použitie vyšších derivácií. Špeciálne, ak sa rozhodneme pre jediný čas t použiť všetky derivácie od nultej až po n-tú, dostaneme Taylorov polynóm n-tého rádu.

Ak sme pri odvodení polynómu použili aspoň jednu hodnotu $y(t_1)$ a zároveň aspoň jednu hodnotu $f(t_2)$ (nemusí nutne platiť $t_1 = t_2$), môžeme ho využiť na odhad hodnoty v čase $t_1 + h$. Získali sme integračnú metódu rovnakého rádu, ako je rád použitého polynómu.

Všeobecným problémom viackrokových metód je ich inicializácia. Vyžadujú niekoľko minulých hodnôt stavu, ktoré v priebehu simulácie máme k dispozícii. Lenže na začiatku simulácie máme spravidla iba jednu aktuálnu hodnotu. V niektorých prípadoch minulé hodnoty vyplývajú zo zadania úlohy. Ak nie, musíme na niekoľko prvých krokov simulácie použiť jednokrokovú metódu a potom môžeme pokračovať viackrokovou metódou.

Metódy Adams-Bashforth

Ak na odvodenie polynómu použijeme aktuálnu hodnotu $y(t)$, aktuálnu hodnotu $f(t)$, niekoľko historických hodnôt $f(t-h), f(t-2h), f(t-3h)\ldots$ v rovnakých intervaloch od seba a tento polynóm vyhodnotíme v čase $t+h$, dostaneme metódu Adams-Bashforth (AB) [5] :

$$\begin{aligned} y(t+h) &= y(t) + \frac{h}{\alpha_n}(\beta_{n,1} f(t) + \beta_{n,2} f(t-h) + \ldots + \beta_{n,n} f(t-(n-1)h)) \\ y(t+h) &= y(t) + (\frac{h\beta_{n1}}{\alpha_n} f(t) + \frac{h\beta_{n2}}{\alpha_n} f(t-h) + \ldots + \frac{h\beta_{n,n}}{\alpha_n} f(t-(n-1)h)) \quad (4.7) \end{aligned}$$

$$\alpha_n = \sum_{i=1}^{n} \beta_{ni}$$

pričom n udáva rád AB metódy, α a β sú konštanty. Pre prvé štyri rády sú hodnotyβ_{ni} dané nasledovnou tabuľkou

$i:$	1	2	3	4
n=1:	1			
$n=2:$	3	-1		
$n=3:$	23	-16	5	
$n=4:$	55	-59	37	-9

Metódy Adams-Bashforth rádu vyššieho ako šesť sú nestabilné – neexistuje taká dĺžka kroku, pri ktorej by riešenie lineárnej diferenciálnej rovnice bolo stabilné. Napriek tomu sú za určitých podmienok použiteľné. Globálna chyba s dĺžkou simulácie exponenciálne rastie, ale pri krátkej simulácii to nemusí vadiť a vyšší rád metódy môže dať presnejší výsledok aj napriek nestabilite.

Systémy, pri simulácii ktorých predlžovaniu kroku nebráni presnosť, ale stabilita, nazývame *tuhé systémy*. Tuhosť je relatívny pojem. Či sa bude systém správať ako tuhý, závisí na konkrétnej požadovanej presnosti a na použitej simulačnej metóde. Rozlišovať tuhosť má zmysel iba pri simulácii analyticky stabilných systémov. Ak dostaneme numericky stabilné riešenie systému, ktorý je analyticky nestabilný, je to skôr na škodu, pretože vzniknutá chyba bude veľká podobne ako pri nestabilnom riešení stabilného systému. Modely použité v tejto práci sú ne-tuhé.

Kapitola 5

Efektivita integračných metód

Efektivita je vzťah medzi vynaloženou námahou a vykonanou užitočnou prácou. V kontexte numerických metód budeme za užitočnú prácu pokladať presnosť simulácie, za námahu strojový čas procesora alebo počet elementárnych aritmetických operácií.

5.1 Chyby integračných metód

Pri použití numerických metód sa typicky objavujú niektoré zdroje chýb súvisiace s aritmetikou reálnych čísel, ako aj niektoré ďalšie:

- zaokrúhľovanie
- akumulácia chyby a nestabilita spôsobená spätnou väzbou
- orezanie
- chyby modelu

Zaokrúhľovanie

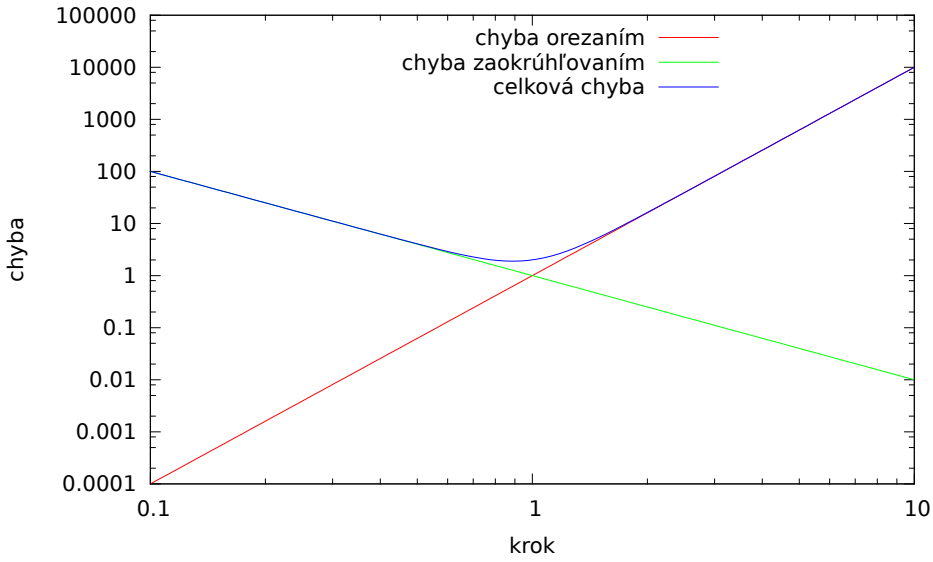
Problém, ktorý sa typicky objavuje pri použití reálnych čísel v numerických integračných metódach, je shiftout. Vyplýva z toho, že v praxi je veľkosť hodnôt stavových premenných násobne väčšia oproti dĺžke kroku, a teda aj oproti rýchlosti ich zmeny za jeden krok, preto dochádza k sčítaniu malých čísel s veľkými. Čím menšia dĺžka kroku, tým je vplyv výraznejší.

O tejto aj ďalších chybách súvisiacich s pohyblivou desatinnou čiarkou bližšie hovorí kapitola 3.2.

Spätná väzba

V každej integračnej metóde je výsledok jedného kroku vstupom do ďalšieho kroku. Ako v každom systéme so spätnou väzbou je preto dôležitá otázka stability. Metóda musí byť tak robustná, aby pri použití približných vstupov dávala približne správne výsledky. Význam vzniknutých chýb sa musí s postupom času strácať a nie narastať.

Stabilita metódy je závislá od konkrétneho riešeného problému aj od dĺžky kroku. Pri riešení lineárnych diferenciálnych rovníc explicitnou metódou vždy existuje dĺžka kroku, pri ktorej sa riešenie stane nestabilným. Je vhodné použiť kratší krok, ako táto hraničná dĺžka. Implicitné metódy bývajú všeobecne stabilnejšie ako explicitné metódy.



Obr. 5.1: Náčrt očakávanej závislosti chyby simulácie od dĺžky kroku.

Orezanie

Neexistuje simulačná metóda nekonečného rádu, pomocou ktorej by sa dalo prakticky počítať v konečnom čase. Musíme použiť metódu obmedzeného rádu, a preto aj krok obmedzenej dĺžky. Rád metódy sa vzťahuje k lokálnej chybe, ktorá vznikne z (teoreticky) presných vstupných údajov po jednom kroku. Prakticky nás zaujíma globálna chyba – presnosť celého iterovaného výpočtu. Dôležitý je nasledujúci poznatok: ak je riešenie stabilné a metóda je rádu n, čiže lokálna chyba úmerná h^{n+1}, potom globálna chyba je úmerná h^n [5]. Ak je riešenie nestabilné, globálna chyba môže s dĺžkou simulácie rásť exponenciálne a je ťažšie ju udržať v tolerovanej veľkosti.

Pri teoreticky nulovej dĺžke kroku je chyba orezaním každej metódy teoreticky nulová – konvergujú k správnemu riešeniu.

Chyby modelu

Podobne, ako je dôležitá stabilita integračnej metódy, je dôležitá aj citlivosť modelu na nepresnosť parametrov. Pri experimentoch v tejto práci boli použité jednoduché modely, aby chyba modelu neznehodnotila porovnávanie metód.

Celková chyba

Celková chyba je netriviálnou kombináciou všetkých chýb, ktoré sa počas simulácie vyskytli. Väčšinou sa ale stáva, že jeden druh chyby je o niekoľko rádov väčší ako ostatné a robí ich zanedbateľnými. Pri veľmi veľkej dĺžke kroku je to nestabilita, pri strednej orezanie, pri veľmi malej zaokrúhľovanie. Pri určitej pevnej presnosti výpočtov sa najmenšia relatívna chyba (t.j. optimum) dosahuje pri takej hodnote kroku, kedy sú chyby orezaním a zaokrúhľovaním približne rovnaké. Tento bod nazvyme *optimálny bod*. Ilustrácia tohto princípu je na obrázku 5.1.

5.2 Určovanie chyby simulácie v experimentoch

Pri experimentoch v tejto práci bude pri analýze chyby vždy dostupné referenčné analytické riešenie. Toto sa vypočíta s použitím presnejšej aritmetiky, než s akou sa bude počítať samotný experiment. Označme $y_r(t)$ referenčnú hodnotu stavovej premennej a $y(t)$ hodnotu stavovej premennej získanú simuláciou, pričom obe sú vztiahnuté k rovnakému času t.

Pri analýze chýb signálov sa používajú rôzne definície chýb. Medzi najčastejšie patria absolútna chyba (norma l^1), Euklidovská vzdialenosť (norma l^2) a maximálna chyba (norma l^∞). Pre účely tejto práce budeme používať *okamžitú chybu* stavovej premennej, a *maximálnu chybu* stavového vektora na simulovanom časovom intervale [5]. Tým zaručíme, že bez ohľadu na to, ktorá zložka je pre užívateľa dôležitá, všetky zložky stavového vektora budú v uvedenej tolerancii. Tiež je takto porovnateľná chyba medzi vektormi o rôznych počtoch prvkov. Ak je pridaná do vektora ďalšia zložka a je aspoň tak presná ako ostatné, meranie to neovplyvní.

Okamžitá chyba

Okamžitá chyba vyjadruje odchýlku simulovanej $y(t)$ od referenčnej hodnoty $y_r(t)$ stavovej premennej v čase t t.j. je to rozdiel vypočítanej a referenčnej hodnoty stavovej premennej v ľubovoľnom čase t

$$d(t) = y(t) - y_r(t) \tag{5.1}$$

Túto veličinu použijeme pri zobrazovaní priebehu okamžitej chyby v čase, aby bolo vidieť veľkosť aj polaritu chyby, t.j. kde má kladnú a kde zápornú hodnotu.

Absolútna chyba simulácie

Nech $y_k(t)$ je simulovaná hodnota k-tej stavovej premennej v čase t a $y_{rk}(t)$ je hodnota k-tej referenčnej stavovej premennej v čase t. Potom maximálna absolútna chyba stavového vektora na simulovanom časovom intervale je

$$E_a = \max_{\forall k,t}\{|y_k(t) - y_{rk}(t)|\} \tag{5.2}$$

Ak je stav systému v čase t vektor reálnych čísel, potom celý priebeh simulácie je reprezentovaný vektorom vektorov hodnôt stavových premenných t.j. maticou. Na meranie chyby celej matice budeme používať rovnakú metriku ako na meranie chyby jedného vektora – maximum z chýb jednotlivých zložiek. Je dôležité poznamenať, že nestačí zisťovať chybu na konci simulácie, pretože pri riešení stabilných systémov pravdepodobne vznikne najväčšia chyba niekde v priebehu simulácie a nevieme, v ktorom konkrétnom čase je stav systému pre užívateľa zaujímavý.

Relatívna chyba simulácie

Nevýhodou použitia absolútnej chyby je jej závislosť na amplitúde stavových premenných. Tento problém rieši použitie relatívnej chyby definovanej vzťahom

$$E = \frac{E_a}{\max_{\forall k,t}\{|y_k(t)|\}} = \frac{\max_{\forall k,t}\{|y_k(t) - y_{rk}(t)|\}}{\max_{\forall k,t}\{|y_k(t)|\}} \tag{5.3}$$

V tomto prípade budeme absolútnu chybu simulácie normovať pomocou maximálnej referenčnej hodnoty zo všetkých stavových premenných z celej dĺžky simulácie. Tento prístup má viacero výhod:

- Nedôjde k deleniu nulou, ani keď správna hodnota v čase vzniku chyby bola nulová.
- Maximálna absolútna chyba simulácie nastane v rovnakom čase ako maximálna relatívna chyba simulácie.
- Výsledok je správne škálovateľný, nezávislý od použitých fyzikálnych jednotiek.

Vzťah medzi d a E maximálnou absolútnou chybou je na grafe (7.7).

5.3 Časová náročnosť integračných metód

Podobne, ako pri Taylorovom polynóme (4.2), aj pri iných integračných metódach máme pri simulácii na výber medzi:

- metódou s vysokým rádom s dlhým krokom simulácie a s tým spojenou malou lokálnou chybou orezaním, malým počtom krokov, veľkou časovou náročnosťou na jeden krok, malým rizikom akumulácie zaokrúhľovacích chýb
- metódou s nízkym rádom a krátkym krokom simulácie a spolu s tým veľkou lokálnou chybou orezaním, veľkým počtom krokov, malou časovou náročnosťou na jeden krok, veľkým rizikom akumulácie zaokrúhľovacích chýb.

Nie je jednoduchá odpoveď na to, ktorý prístup je pri danej tolerancii globálnej chyby výhodnejší.

Meranie času

Ak chceme porovnávať časovú zložitosť, musíme si najprv vyjasniť pojem merania času.

Modelový čas sa vzťahuje k simulovanému dynamickému systému. Ak simulujeme jazdu auta od štartu v čase nula po príchod do cieľa v čase jedna hodina, znamená to simuláciu jednej hodiny modelového času.

Čas, ktorý uplynie počas práce počítača na simulácii, budeme volať *reálny čas*. Jedna hodina modelového času jazdy auta môže na pomalom počítači znamenať minútu reálneho času čakania na výsledok simulácie alebo na rýchlom počítači sekundu reálneho času.

Pri praktickej simulácii nás zaujíma, koľko reálneho času musíme čakať na vypočítanie výsledku. Samotný fakt, že počítače majú rôznu rýchlosť nie je pre porovnávanie problém. Relatívna rýchlosť rôznych metód by zostala aj pri inom počte operácií za sekundu zachovaná. Problém je, že pomer rýchlosti rôznych operácií na dvoch počítačoch môže byť rôzny. Ak je sčítanie dvakrát rýchlejšie, neznamená to, že musí byť dvakrát rýchlejší aj výpočet kosínusu. Jedna integračná metóda sa môže po prechode na iný počítač zrýchliť, ďalšia spomaliť.

Ak chceme porovnávať časovú zložitosť numerických metód a výsledky zovšeobecniť, okrem reálneho času je potrebná aj nejaká jednotka nezávislá od konkrétneho počítača. Jedna možnosť je zrátať *počet potrebných operácií* (napr. FLOP – Floating Point Operation). Metódy, na ktoré sa sústredíme v tejto práci, využívajú sčítanie a násobenie približne v pomere 1:1, nijaké odčítanie ani delenie. Tým, že je pomer konštantný, vyhneme sa problému rôznych relatívnych rýchlostí rôznych operácií.

Ďalšia možnosť je vychádzať z predpokladov, že:

- jeden výpočet derivácií zo stavových rovníc modelu $f(y(t))$ prebehne v konštantnom čase,
- model je zložitý, napríklad ak obsahuje transcendentálne funkcie náročné na výpočet,
- počet operácií integračnej metódy je v porovnaní so stavovými rovnicami modelu zanedbateľný.

V takom prípade môžeme ako univerzálnu metriku času použiť *počet vyhodnotení derivácie*, ktorý daná metóda vyžaduje.

Časová zložitosť aritmetických operácií

V tejto práci budeme považovať presnosť aritmetiky pre všetky medzivýpočty jednej simulácie za konštantnú, ak nebude v popise experimentu uvedená výnimka.

Pre sčítanie a odčítanie je časová zložitosť v závislosti od počtu bitov lineárna.

Pri násobení a delení je situácia zložitejšia. Používajú sa desiatky algoritmov s rôznou zložitosťou od najjednoduchších kvadratických až po *Schönhage-Strassenov* algoritmus násobenia zložitosti $O(n \log n \log\log n)$ založený na FFT [18]. Algoritmy s dobrou časovou zložitosťou sú nie len implementačne náročnejšie, ale aj pomalšie pre relatívne malé počty bitov v porovnaní s asymptoticky horšími, zato jednoduchšími algoritmami.

Náročnosť jedného kroku simulácie

Eulerova metóda (rovnica 4.1), ktorá je spoločnou podmnožinou všetkých skupín explicitných numerických integračných metód, vyžaduje v každom kroku jednu hodnotu $f(y(t))$, jedno násobenie a jedno sčítanie. Ak označíme d počet operácií potrebných na vyčíslenie $f(y(t))$, potom celkový počet operácií na jeden krok

$$o_E = d + 2 \tag{5.4}$$

Vo všetkých uvažovaných metódach je pomer sčítaní a násobení približne 1:1, preto ich nebudeme rozlišovať.

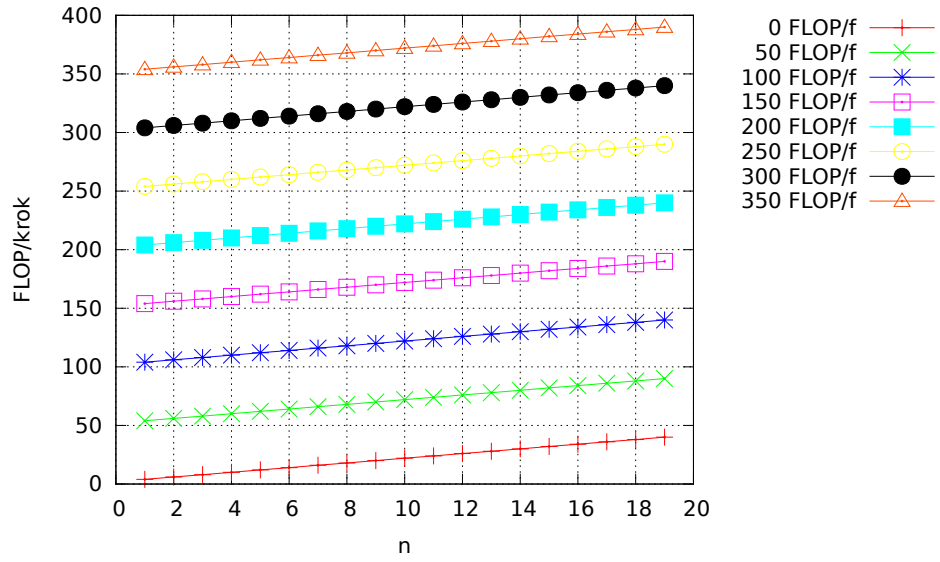
Adams-Bashforth (rovnica 4.7) bez ohľadu na rád vyžaduje na jeden krok jednu hodnotu derivácie $f(y(t))$. Počet ostatných aritmetických operácií v závislosti od rádu n je lineárny $2n$. Každé zvýšenie rádu pridá jedno násobenie a jedno sčítanie. Celkový počet operácií na jeden krok je tak

$$o_{AB} = d + 2n \tag{5.5}$$

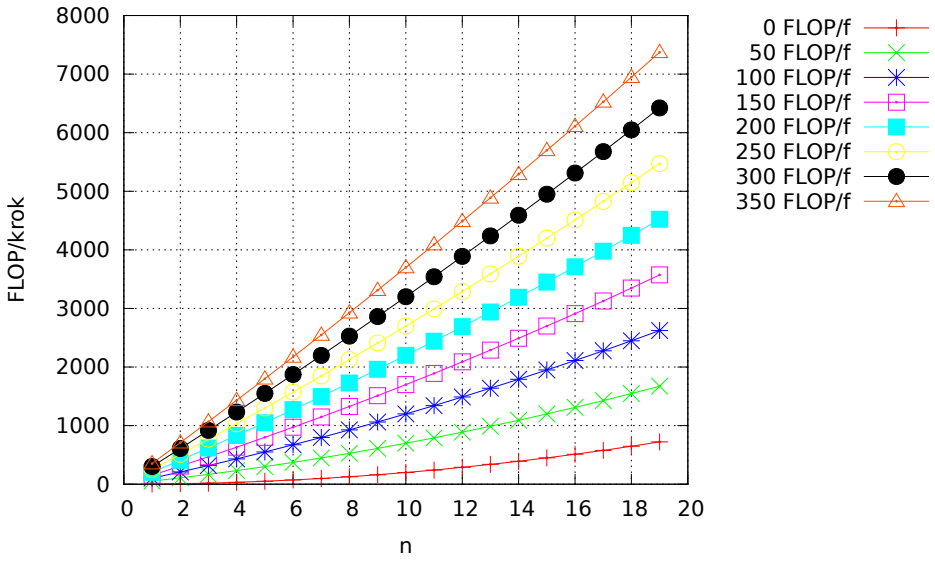
Runge-Kutta (rovnica 4.4) vyžaduje v jednom kroku toľko výpočtov derivácií, aký je stupeň metódy s. Počet ostatných aritmetických operácií rastie v závislosti od stupňa kvadraticky $2s^2$. Potom celková zložitosť na jeden krok je

$$o_{RK} = sd + 2s^2 \tag{5.6}$$

Viac nás zaujíma náročnosť v závislosti od rádu a nie od stupňa, lenže všeobecný vzťah medzi rádom a stupňom nie je známy. Známe hodnoty pre niektoré metódy sú v tabuľke 4.1. Graf závislosti výpočtovej náročností metód RK a AB vo FLOP pre niektoré konkrétne hodnoty n, s a d je na obrázku 5.2, pričom parameter f reprezentuje výpočtovú náročnosť jednej derivácie. Experimentálne namerané trvanie jedného kroku integračnej metódy v závislosti od počtu bitov aritmetiky je na grafe 7.2.



(a) Adams-Bashforth



(b) Runge-Kutta

Obr. 5.2: Počet aritmetických operácií na jeden krok metód Adams-Bashforth (5.2a) a Runge-Kutta (5.2b) v závislosti od stupňa a od počtu operácií na jedno vyhodnotenie derivácie.

Časová náročnosť simulácie

Časová náročnosť jedného kroku vyznieva jednoznačne v prospech metód Adams-Bashforth pred Runge-Kutta. Lenže to nie je jednoznačná odpoveď na otázku, ktorá metóda je rýchlejšia. Runge-Kutta má všeobecne lepšiu presnosť ako Adams-Bashforth rovnakého rádu pri rovnakej dĺžke kroku. Aj oblasť stability sa u Runge-Kutta s rastúcim rádom zväčšuje, u Adams-Bashforth zmenšuje. To môže pri metóde Adams-Basforth nútiť k veľmi malým dĺžkam kroku, a preto k výraznému zhoršeniu časovej náročnosti. Konkrétne experimentálne výsledky sú v časti 7.

Kapitola 6

Návrh knižnice numerických metód

Táto kapitola popisuje implementovanú knižnicu numerických integračných metód, ktorú nazveme DESSL (Differential Equation System Simulation Library). Pri návrhu knižnice boli zohľadnené nasledujúce ciele a požiadavky:

- Podpora jednokrokových aj viackrokových metód vysokého rádu.
- Implementácia v programovacom jazyku C++.
- Podpora výpočtov s viacnásobnou presnosťou ako aj so štandardnými číselnými typmi pomocou šablón.
- Implementácia nástrojov na analýzu chyby simulácie.
- Implementácia nástrojov na vyhodnotenie efektivity simulácie.
- Použitie iba takých existujúcich komponentov, ktoré svojou licenciou neobmedzia použiteľnosť, šírenie a ďalší vývoj.
- Prenositeľnosť.
- Čitateľnosť kódu.

V ďalšej časti sú priblížené hlavné časti knižnice spolu s niekoľkými príkladmi jej použitia.

6.1 Štruktúra knižnice

Triedy a funkcie knižnice DESSL sú logicky rozdelené do štyroch častí (Obrázok 6.1):

- Systém,
- Simulácia,
- Spracovanie,
- Automatizácia.

Časť *Systém* obsahuje triedy a funkcie určné pre definovanie simulovaných systémov a ich počiatočných podmienok. Jadro tvorí materská trieda `equations`, od ktorej sa odvádzajú triedy konkrétnych systémov charakterizovaných pomocou sústavy stavových rovníc. Súčasťou knižnice je niekoľko tried preddefinovaných systémov, ktoré boli použité v experimetoch. Ako príklad uvádzame časť kódu s definíciou stavových rovníc systému netlmeného oscilátora s parametrom dátového typu `number_t`:

```
template<class number_t>
class sinus_cosinus_eq: public equations<number_t>
{
  public:
  ...
  virtual void derivative(
  const vector<number_t> &state,
  vector<number_t> &result)
  {
    result[0] = state[1];
    result[1] = state[0] * -1;
  }
...
};
```

Na popis linearnych systémov je určená trieda `configurable_linear_equation`, ktorá umožňuje popis sústavy lineárnych diferenciálnych rovníc maticou A s konštantnými koeficientami získanej z externého súboru. Matica pritom nemusí byť známa v čase prekladu programu. Trieda `ivp` (Initial Value Problem) spája rovnice a počiatočné podmienky do kompletného popisu úlohy určenej na simuláciu.

Časť *Simulácia* obsahuje triedy a funkcie pre definovanie simulátorov spojitých dynamických systémov. Zo základnej triedy `simulator` sú tu odvodené špecifické simulátory pre metódy Euler (`euler_simulator`), Adams-Bashforth (`ABsimulator`), Runge-Kutta (`RKsimulator`) a simulátor analytického riešenia systému (`analytic_simulator`). Princíp vytvorenia systému a simulátora naznačuje nasledovný kód:

```
void simuluj()
{
  // definovanie pociatocných hodnôt stav. vektora
  vector<number_t> init_value = ...

  // vytvorenie systému
  system_type<number_t> system;

  // vytvorenie ivp
  ivp<number_t> problem(init_value, system);

  // vytvorenie simulátora a prepojenie so systémom
  simulator_type<number_t> sim(problem);
}
```

Časť knižnice *Spracovanie* poskytuje triedy a funkcie pre záznam a spracovanie priebehu a výsledkov simulácie. Trieda `history_t` poskytuje dátovú štrultúru pre záznam priebehu stavových premmenných pomocou triedy `history_recorder`. Trieda `result_processor_t` je rozhraním pre spracovanie výsledkov simulácie. Metodika spustenia simulácie s určením spôsobu spracovania výsledkov napr. so záznamom hodnôt stavového vektora a výpisom na štandardný výstup je naznačená v nasledovnom kóde

```
void simuluj()
{
  // definícia typu spracovania výsledkov
  // v tomto prípade záznam stavu po každom kroku simulácie
  history_recorder<number_t> observer;

  //spustenie simulácie a prepojenie s rekordérom
  sim.run(start_time, stop_time, step_length, observer);

  // príkaz na vypísanie zaznamenaného priebehu simulácie
  observer.history.print();
}
```

Keďže dôležitou súčasťou tejto práce bola experimentálna analýza implementovaných metód s použitím množstva rôznych parametrov a nastavení, bolo pri implementácii knižnice nutné automatizovať riadenie experimentov pre definované rozsahy parametrov. Na tieto účely boli vyvinuté triedy a funkcie v časti *Automatizácia*. Základom je tu materská trieda `simulator_factory`, ktorá slúži na odvodenie tried generujúcich simulátory rôznych metód pre definované rozsahy parametrov napr. pre dávkovú analýzu metód Runge-Kutta (`rk_factory`) a Adams-Beshforth (`ab_factory`).

Nasledovný zoznam sumarizuje najdôležitejšie elementy knižnice DESSL:

`number_t` Typový parameter šablóny, ktorý určuje použitú aritmetiku.

`number_ref_t` Aritmetika na výpočet referenčného riešenia a chyby, obyčajne presnejšia ako number_t.

Triedy časti: Systém

`equations` Bázová trieda pre zadávanie diferenciálnych rovníc. Užívateľ môže zadať vlastné zdedením a predefinovaním metód.

- `exp_eq` trieda pre popis systému s exponenciálnym priebehom stavovej premennej
- `sinus_cosinus_eq` trieda pre popis netlmeného kyvadla (netlmený oscilátor)
- `sinus_rounded_eq` trieda pre popis netlmeného kyvadla so zaokrúhleným vypočtom derivácií, použité pri experimente o vplyve zaokrúhlenia na rôzne časti výpočtu
- `parabola_eq` trieda pre popis systému s polynomiálnym rastom hodôt stavovej premennej
- `pendulum_eq` trieda pre popis systému tlmeného kyvadla (tlmený oscilátor)
- `configurable_linear_eq` trieda pre popis systému určeného maticou A s konštantnými koeficientami uloženou v externom súbore

`ivp` Initial Value Problem, spája rovnice a počiatočné podmienky do kompletného popisu úlohy určenej na simuláciu.

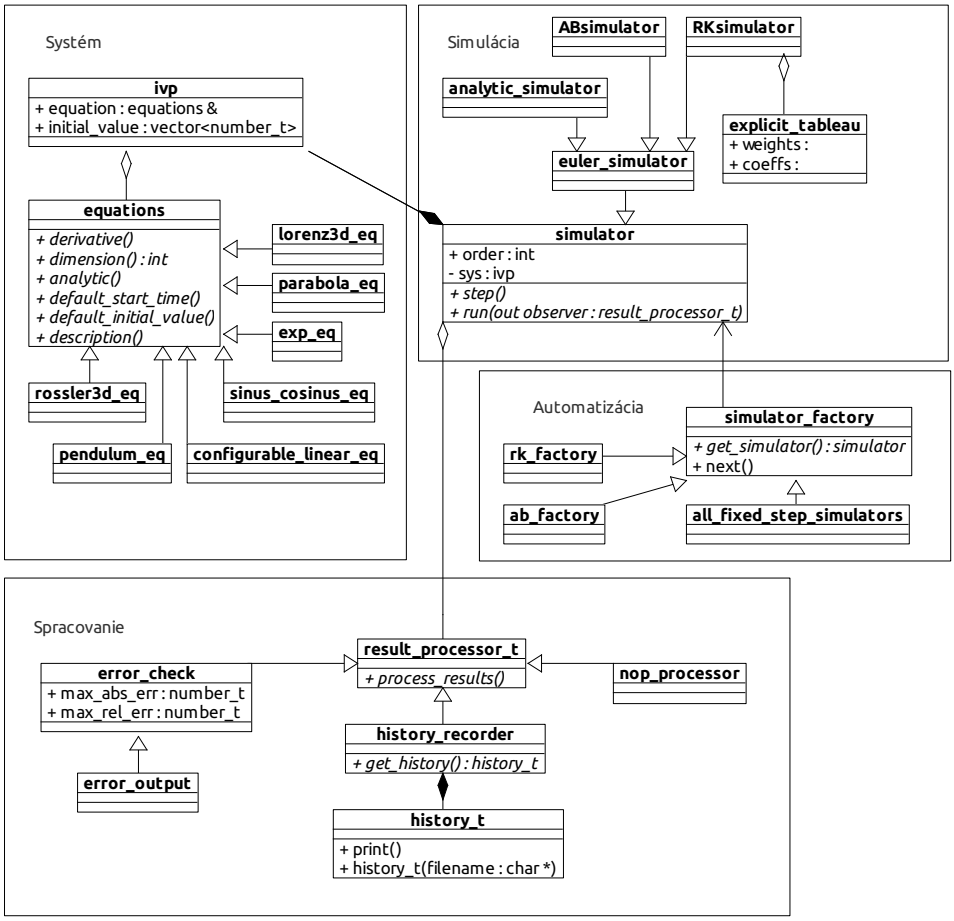


Diagram: class diagram Page 1

Obr. 6.1: Principiálne zobrazenie štruktúry knižnice

Triedy časti: Simulácia

`simulator` Spoločná bázová trieda rôznych integračných metód.

- `euler_simulator` Simulátor Eulerovou metódou.
- `analytic_simulator` Pseudosimulátor, ktorý namiesto simulácie počíta analytické riešenie. Použiteľný napríklad pri experimentoch s presnosťou.
- `RKsimulator` Simulátor explicitnou metódou Runge-Kutta. Konkrétny rád je určený použitou Butcherovou tabuľkou.
- `ABsimulator` Simulátor metódou Adams-Bashforth.

`explicit_tableau` Butcherova tabuľka, obsahuje konštanty pre explicitné metódy Runge-Kutta.

Triedy časti: Spracovanie

`history_t` Záznam vývoja stavových premenných v čase.

`analyzer` Porovnanie záznamov simulácie (off-line po skončení simulovania) a ich jednoduché štatistické spracovanie.

`result_processor_t` Materská trieda pre objekty, ktoré dostávajú informáciu o priebehu simulácie po každom simulačnom kroku (on-line spracovanie za behu simulácie). Užívateľ môže definovať vlastné priebežné spracovanie výsledkov zdedením a predefinovaním metód.

- `history_recorder_t` Špecializácia `result_processor_t` zaznamenáva celý priebeh simulácie.
- `error_check` Špecializácia `result_processor_t` pomocou známeho analytického riešenia vyhodnocuje chybu simulácie.

`epsilon` Zistí ε použitej aritmetiky.

Triedy časti: Automatizácia

`simulator_factory` Bázová trieda pre objekty, ktoré vyrábajú simulátory rôznych metód. Použiteľné napríklad pri automatizácii experimentov.

- `all_fixed_step_simulators` Špecializácia `simulator_factory`, ktorá vyrobí všetky simulátory s pevným krokom.
- `high_order_factory` Špecializácia `simulator_factory`, ktorá vyrobí simulátory vysokého rádu. Vhodná na experimenty s veľkou presnosťou aritmetiky.
- `ab_factory` Trieda pre objekty, ktoré vyrábajú simulátory s metódou Adams-Bashforth.
- `rk_factory` Trieda pre objekty, ktoré vyrábajú simulátory s metódou Runge-Kutta.

Ďalšie možnosti a elementy knižnice je možné vyčítať priamo zo zdrojového kódu knižnice, ktorý je priložený v elektronickej podobe k tejto diplomovej práci.

6.2 Príklady použitia knižnice

V tejto časti poskytneme pre názornosť niekoľko konkrétnych príkladov použitia knižnice DESSL. Ďalšie príklady použitia knižnice je možné nájst v priloženom zdrojovom kóde, ktorý vo fukcii `choose_experiment()` v subore `settings.h` obsahuje kód všetkých experimentov vykonaných v tejto práci. Výber a spustenie zvoleného experimentu je možné pomocou priloženého `makefile` s použitím zodpovedajúceho parametra.

Príklad 1: Simulácia a zobrazenie výsledkov

Ako prvý príklad uvedieme úryvok z C++ funkcie, ktorá odsimuluje predpripravený model predpripravenou simulačnou metódou:

```
//simulovaný systém
lorenz_eq<double> eq;

//systém s počiatočnými podmienkami
ivp<double> problem(some_initial_value_of_type_vector_double, eq);

//simulačná metóda
euler_simulator<double> sim(problem);

//spracovanie stavu po každom kroku simulácie
history_recorder<double> observer;

//spustenie simulácie
sim.run(start_time, stop_time, step_length, observer);

//výpis na štandardný výstup
observer.history.print();
```

Podrobnejší popis:

`lorenz_eq<double> eq;` Tento riadok vyrobí model, ktorý chceme simulovať, podľa predpripravenej triedy. Parameter šablóny je typ aritmetiky, ktorú chceme použiť.

`ivp<double> problem(..initial_value..., eq);` Spojí model s počiatočným stavom. Počiatočný stav je vektor čísel rovnakého typu, aký sme zvolili pri modeli.

`euler_simulator<double> sim(problem);` Vyrobí pre náš model simulátor s použitím Eulerovej metódy.

`history_recorder<double> observer;` Vyrobí objekt, ktorý bude sledovať a zaznamenávať priebeh simulácie pre neskoršie spracovanie.

`sim.run(start_time, stop_time, step_length, observer);` Spustí simuláciu zvoleného časového intervalu so zvolenou dĺžkou kroku a zvoleným spôsobom spracovania výsledkov.

`observer.history.print();` Vypíše výsledky. Okrem metódy print() možno s objektom `observer.history` manipulovať ako so štandardným vektorom a je teda prístupný ľubovoľnému ďalšiemu spracovaniu.

Príklad 2: Chyba simulácie voči analytickému riešeniu

Tenko príklad popisuje jeden z najčastejšie používaných experimentov v tejto práci. Zisťuje maximálnu relatívnu chybu simulácie pomocou presnejšieho referenčného riešenia.

```
//simulovaný model
sinus_cosinus_eq<float> eq;

//referenčný model s presnejšou aritmetikou
sinus_cosinus_eq<double> refeq;

//systém s počiatočnými podmienkami
ivp<float> problem(eq.default_initial_value(), eq);

// spôsob analýzy chyby
error_check<float, double> observer(refeq);

// spustenie simulácie
sim.run(start_time, stop_time, step_length, observer);

// vypíše E - najväčšiu nameranú relatívnu chybu
std::cout << observer.max_rel_err;
```

Podrobnejší popis:

`eq.default_initial_value()` Získa počiatočný stav systému, ku ktorému sa vzťahuje analytické riešenie. Je to dôležité v prípade, že implementované analytické riešenie sa vzťahuje iba na nejakú podmnožinu počiatočných stavov, ako napríklad v prípade kyvadla.

`error_check<float, double> observer(refeq);` Vyrobí objekt, ktorý bude porovnávať referenčné riešenie so simulačným a zapamätá si výskyt najväčšej chyby.

Kapitola 7

Experimentálne porovnanie presnosti a efektivity metód

V tejto kapitole sa budeme venovať experimentom s implementovanou knižnicou. Najdôležitejšie experimenty sa budú venovať závislosti chyby na rôznych parametroch simulácie, hľadaniu optimálnych parametrov presnosti a kroku simulácie pre dosiahnutie minimálnej chyby a nakoniec spotrebovanému strojovému času v týchto optimách.

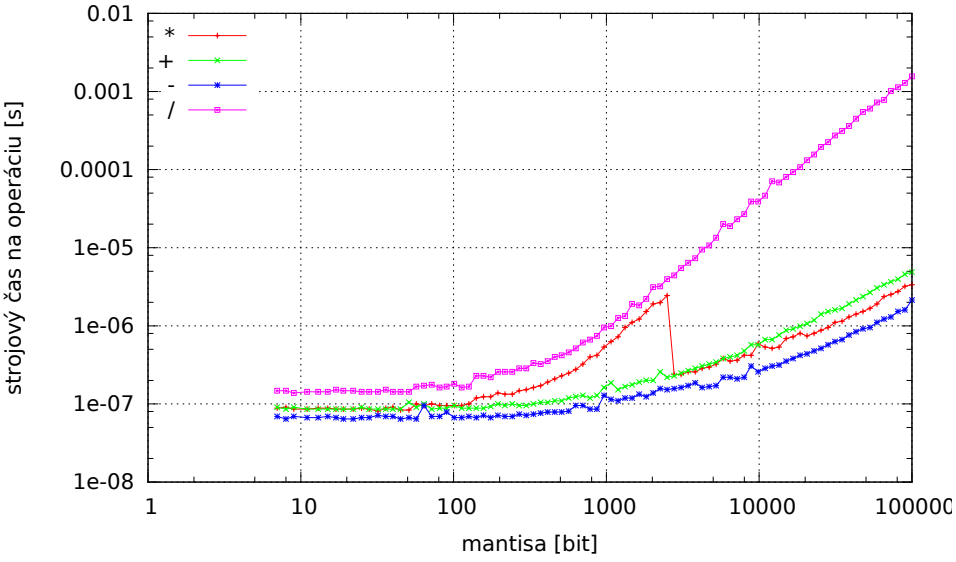
7.1 Časová náročnosť základných aritmetických operacií

Jedným z cieľov tejto práce je meranie reálneho času potrebného na simuláciu integračných metód s viacnásobnou presnosťou pomocou implementovanej knižnice. Keďže knižnica využíva pre implementáciu typov a operácií vo viacnásobnej presnosti knižnice MPFR a GMP, vyvstáva otázka, aká je časová náročnosť základných aritmetických operácií v týchto knižniciach v závislosti od presnosti. Experiment prebieha tak, že pre zvolenú presnosť mantisy s počtom bitov n sa zvolená aritmetická operácia opakuje toľkokrát, aby sa dal pomocou funkcií operačného systému zmerať reálny čas. Nameraný čas sa potom vydelí počtom operácií. Jednotlivé operácie sú na sebe nezávislé. Preto, ak to použitá knižnica dovolí, môže byť využité aj paralelné a prúdové spracovanie. Je zrejmé, že hodnoty časovej náročností operácií ako aj celých algoritmov závisia od použitého hardwaru a softvérového prostredia a je nemožné namerané údaje zovšeobcniť. Pri aplikáciách, kde záleží na výkone, je dôležité doladiť nastavenia knižnice pre konkrétny počítač. Napriek tomu sú tieto údaje cenné pre úlohy predikcie a porovnania s inými systémami. Kvôli objektívnosti, boli všetky úlohy určovania časovej náročnosti operácií a algoritmov v tejto práci vykonané na jednom výpočtovom systéme a pri rovnakých podmienkach. Parametre tohto systému sú zhrnuté v tabuľke 7.1.

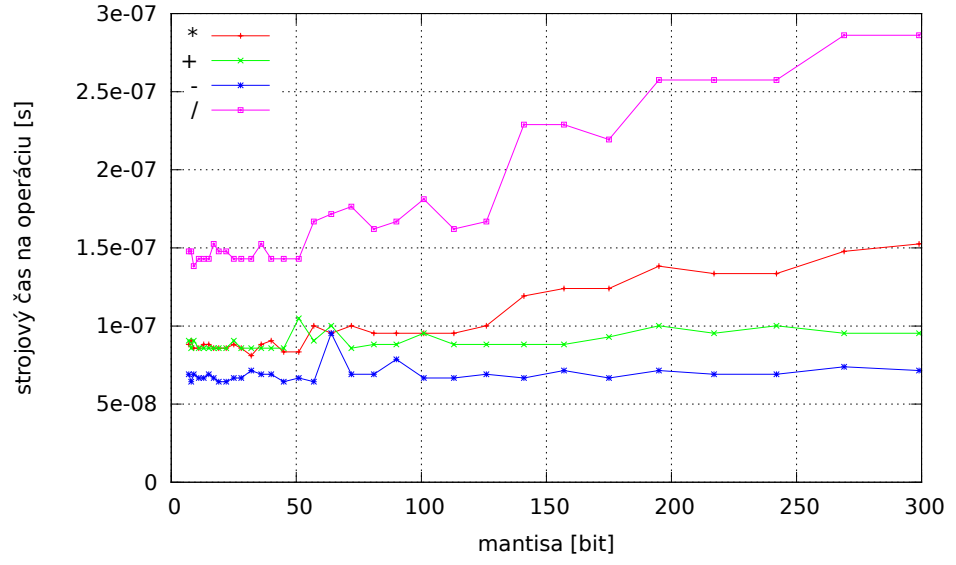
Získané namerané výsledky časovej náročnosti základných aritmetických operácií v závislosti od presnosti výpočtov sú na obrázku 7.1. Výrazný skok v rýchlosti násobenia je spôsobený neoptimálne nastaveným prahom, pri ktorom sa menia použité algoritmy.

Pre porovnanie uvedieme výkon daného systému s ohľadom na základné aritmetické operácie pri použití natívnych a MPFR dátových typov s rovnakou presnosťou a rôznych stratégiách optimalizácie::

- Ak sú operandy v registroch procesora, testovací počítač dosahuje s 53-bitovou presnosťou rýchlosť približne 0,4 až 1 gigaflop za sekundu.



(a)



(b)

Obr. 7.1: Nameraná dĺžka trvania základných aritmetických operácií v sekundách (7.1a) a výrez najčastejšie používanej oblasti z tohto grafu v lineárnej mierke (7.1b).

Parameter	Hodnota
PC Systém	HP EliteBook 8540w
Procesor	Intel i7 720 QM 64-bit
Taktovacia frekvencia	1,6 MHz
Max. turbo taktovacia frekvencia	2,4 MHz
Počet jadier	4
Smart Cache	6 MB
Veľkosť RAM	DDR3 8GB
Typ RAM	DDR-1066/1333
OS	Linux 3.0.0 x86-64
Knižnica	MPFR 64-bit ver. 3.0.1
Knižnica	GMP 64-bit ver. 5.0.1
Prekladač	GCC ver. 4.6.1
Parameter optimalizácie prekladu	-O3

Tabuľka 7.1: Prehľad najdôležitejších parametrov použitého výpočtového systému

- Pri použití SIMD operácií je to 6 až 15 gigaflop.
- Ak sú operandy v RAM a výsledok sa ukladá znova do RAM, rýchlosť je len 15 megaflop za sekundu.
- Pri použití knižnice MPFR s rovnakou presnosťou je rýchlosť približne 25 megaflop za sekundu.

Tieto údaje sú iba orientačné, závisia od konkrétnej štruktúry výpočtu. Graf závislosti trvania jedného kroku od presnosti aritmetiky pre niektoré konkrétne integračné metódy je na obrázku (7.2). V tomto pokuse bol výpočet derivácií časovo zanedbateľný.

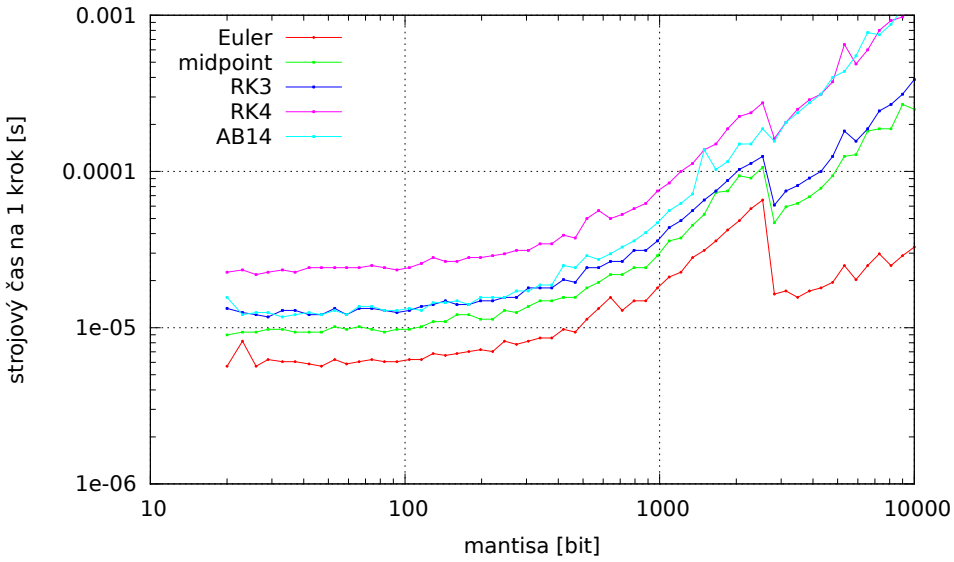
7.2 Modely systémov použité na testovanie

Z dôvodu nutnosti analýzy chyby, je dôležité na testovanie zvoliť systémy, ktoré majú známe analytické riešenie. V takom prípade môže nameraná chyba zodpovedať skutočnosti a nieje potrebné porovnávanie s nepresným referenčným riešením. V nasledujúcej časti sú opísané modely systémov, ktoré boli použité pri testovaní implementovanej knižnice.

Netlmené kyvadlo, kruhový test

Model netlmeného harmonického oscilátora možno opísať diferenciálnymi rovnicami a počiatočnými podmienkami:

$$\begin{aligned} p'(t) &= v(t) \\ v'(t) &= -p(t) \\ v(0) &= 1 \\ p(0) &= 0 \end{aligned} \tag{7.1}$$



Obr. 7.2: Reálny čas trvania jedného kroku vybraných integračných metód.

pričom p označuje polohu kyvadla a v jeho rýchlosť. Tento systém má analytické riešenie vo forme

$$\begin{aligned} v(t) &= \cos(t) \\ p(t) &= \sin(t) \end{aligned}$$

Časový priebeh stavových premenných spolu so zobrazením stavového priestoru pre referenčné riešenieje na obrázku 7.3. Taylorov rad analytického riešenia má všetky členy nenulové, čo znamená, že je možné neobmedzene zvyšovať rád integračnej metódy a tým zvyšovať presnosť simulácie s ohľadom na chybu orezania.

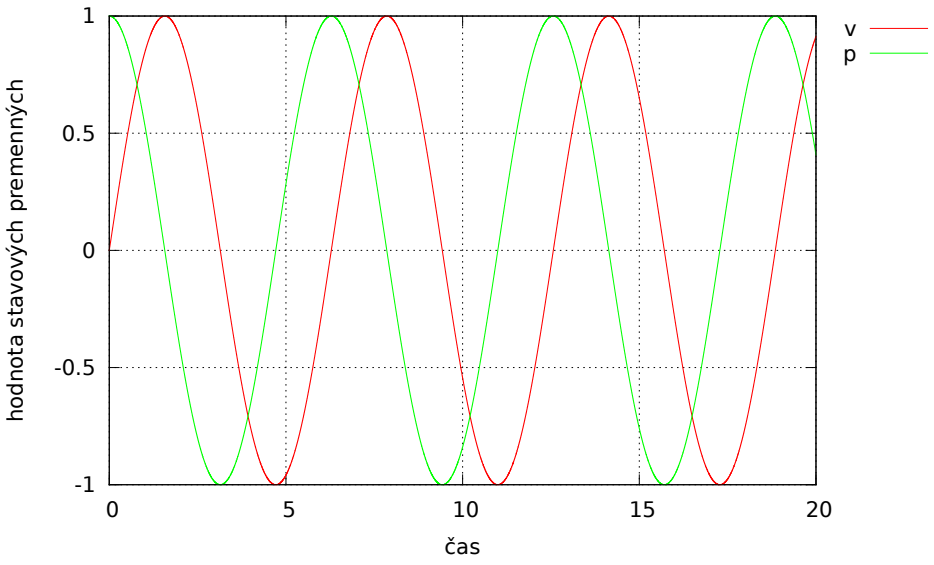
Tlmené kyvadlo

Model tlmeného kyvadla (oscilátora) získame zovšeobecnením modelu netlmeného kyvadla, keď predpokladáme lineárnu závislosť odporu prostredia σ od okamžitej rýchlosti (čiastočne prevzaté z [16]):

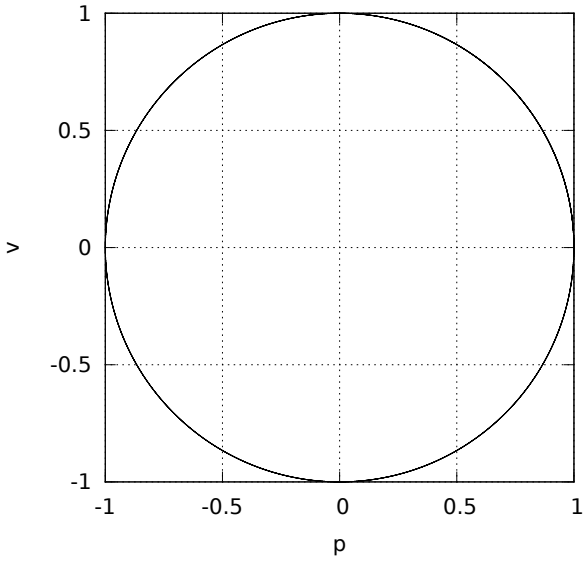
$$\begin{aligned} p\text{'(t)} &= v(t) \\ v'(t) &= -2\sigma v(t) - (\sigma^2 + \omega^2)p(t) \\ \omega &= \frac{2\pi}{T} \\ p(0) &= 1 \\ v(0) &= 0 \end{aligned} \tag{7.2}$$

pričom T je perióda kyvadla, ω uhlová rýchlosť kyvadla a $\sigma > 1$ je koeficient tlmenia. V experimentoch budeme používať $T = 1$ a $\sigma = 2$. Analytické riešenie je v tvare

$$\begin{aligned} p(t) &= e^{-dt}(\cos(\omega t) + \frac{\sigma \sin(\omega t)}{\omega}) \\ v(t) &= e^{-\sigma t}((\sigma \cos(\omega t) - \omega \sin(\omega t)) - \sigma(\frac{\sigma \sin(\omega t)}{\omega} + \cos(\omega t))) \end{aligned} \tag{7.3}$$



(a)



(b)

Obr. 7.3: Referenčné riešenie modelu netlmeného kyvadla. Časový priebeh a) a stavový priestor b)

Na rozdiel od netlmeného kyvadla je tento systém stabilný. Ak je stabilné aj jeho numerické riešenie, vtedy s rastúcim časom konverguje k správnemu riešeniu $p = v = 0$. Je to príklad, kde najväčšia okamžitá chyba nemusí byť blízko konca simulácie.

Časový priebeh stavových premenných spolu so zobrazením stavového priestoru pre referenčné riešenie je na obrázku (7.4).

Polynomiálny rast

Model polynomiálneho rastu je v tvare

$$\begin{aligned} y'(t) &= n y(t)^{\frac{n-1}{n}} \\ y(1) &= 1 \end{aligned} \tag{7.4}$$

kde n je konštanta – kladné celé číslo. Táto forma zápisu je autonómna a má jednozložkový stavový vektor. Iný ekvivalentný spôsob zápisu je napríklad $y'(t) = nt(t)^{n-1}$. Analytické riešenie polynomiálneho rastu je

$$y(t) = t^n \tag{7.5}$$

Pre $n = 2$ je grafom parabola, pre $n = 3$ kubika. Riešenie je polynóm, preto jeho Taylorov rad má konečný počet členov závislý od n.

Pri simulácii je dôležité zvoliť počiatočné podmienky mimo bodu $t = 0, y(0) = 0$, pretože vtedy je systém vo vratkej rovnováhe a analytické aj simulačné riešenie je $y(t) = 0$. Časový priebeh stavovej premennej referenčného riešenia pre rôzne n je na obrázku 7.5.

7.3 Časový priebeh chyby

Graf získaný simuláciou s vhodnými parametrami býva voľným okom nerozoznateľný od grafu získaného analytickým riešením. Ak chceme vidieť priebeh chyby, musíme zobraziť priamo rozdiel medzi analytickým a simulačným riešením.

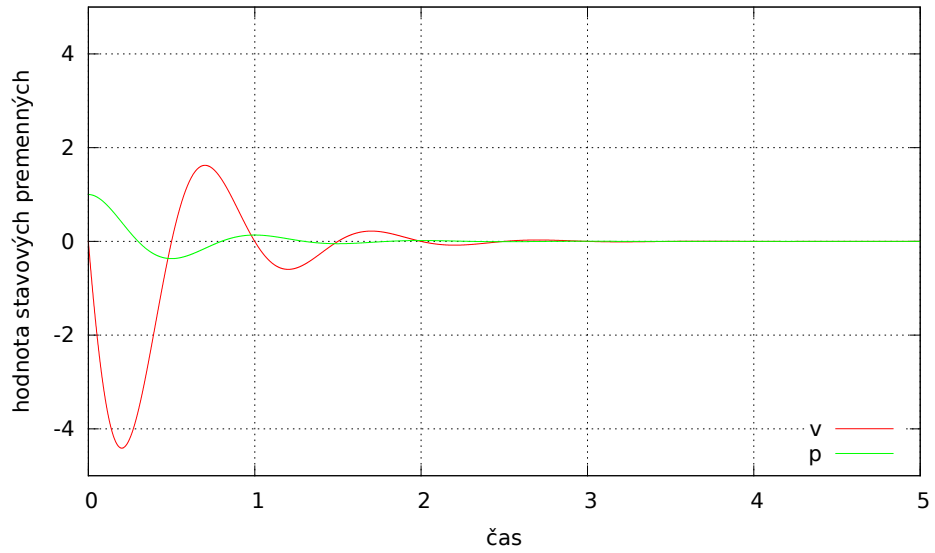
Netlmené kyvadlo

Pre demonštráciu si zvolíme metódu Runge-Kutta tretieho rádu (RK3). Pri tejto metóde a 24-bitovej presnosti sa všetky želané efekty prejavia v pohodlnom intervale krokov. Pretože sa k tejto úlohe budeme viackrát vracať, nazvyme ju *Problém číslo 1* s parametrami:

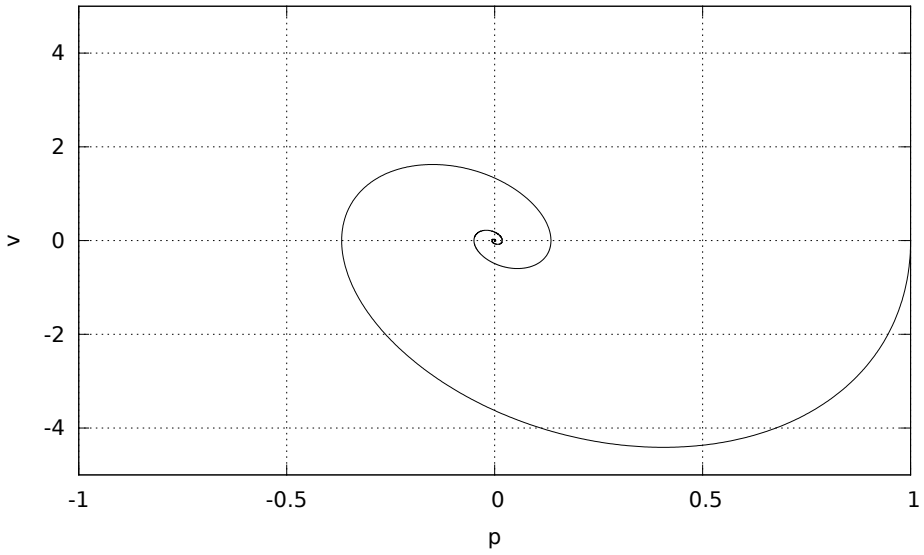
- model netlmeného kyvadla podľa rovnice (7.1),
- modelový čas 0 - 20,
- metóda RK3,
- mantisa 24 bitov.

Ďalej si zvolíme dĺžku kroku $h = 10^{-1}$. Pri tomto kroku je hlavným zdrojom chyby orezanie (použili sme metódu tretieho rádu). Zanedbaná štvrtá derivácia má tvar

$$\begin{aligned} p^{(4)}(t) &= \cos(t) \\ v^{(4)}(t) &= \sin(t) \end{aligned}$$

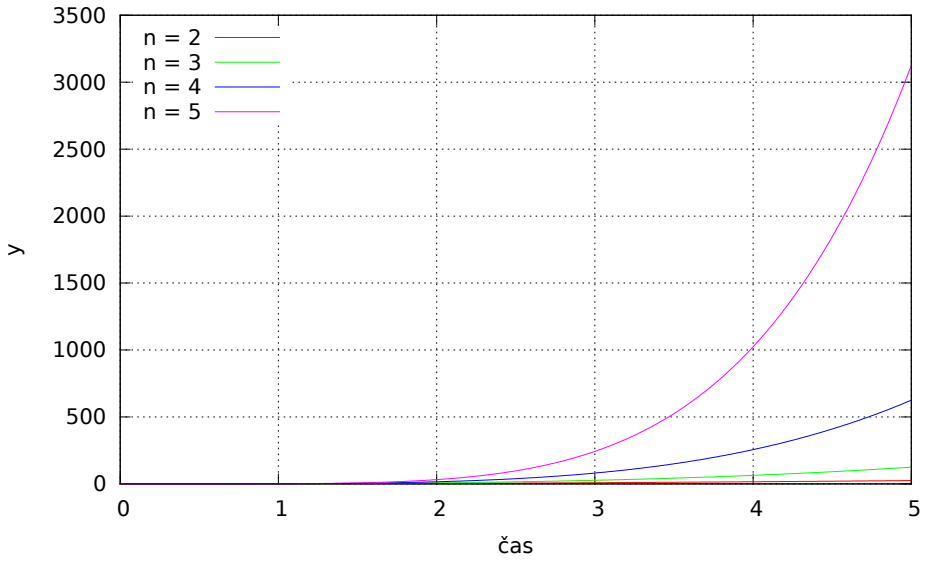


(a)



(b)

Obr. 7.4: Referenčné riešenie modelu tlmeného kyvadla. Časový priebeh a) a stavový priestor b)



Obr. 7.5: Polynomiálny rast

Rozdiel medzi analytickým a simulačným riešením je úmerný zanedbanému najvyššiemu členu Taylorovho radu. Tiež je úmerný dĺžke simulácie, pretože chyba sa akumuluje.

$$\begin{aligned} d_p(t) &\approx t\cos(t) \\ d_v(t) &\approx t\sin(t) \end{aligned}$$

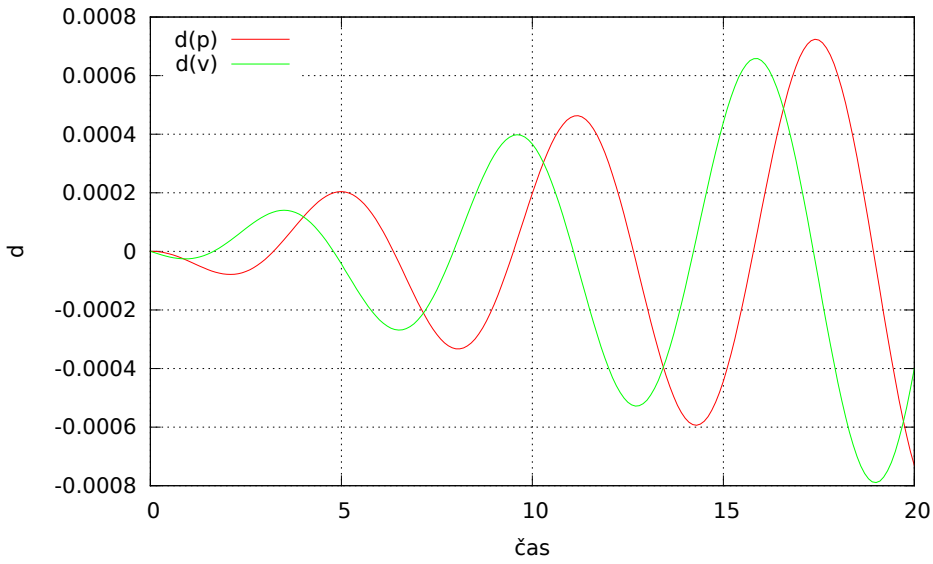
Graf časového priebehu stavových premenných je na obrázku (7.6).

Aby sme ozrejmili vzťah medzi okamžitým rozdielom simulačného a analytického riešenia d a relatívnou chybou simulácie E, zobrazíme tento graf ešte raz, aj s vyznačenou relatívnou chybou – obrázok 7.7. Pretože maximálna hodnota referenčného riešenia je 1 a nastane hneď na začiatku simulácie v čase 0, absolútna a relatívna chyba sú pre tento systém vždy rovnaké. Chyba je maximum absolútnej hodnoty odchýlky, preto narastie vždy, keď okamžitá odchýlka dosiahne nový extrém (kladný alebo záporný).

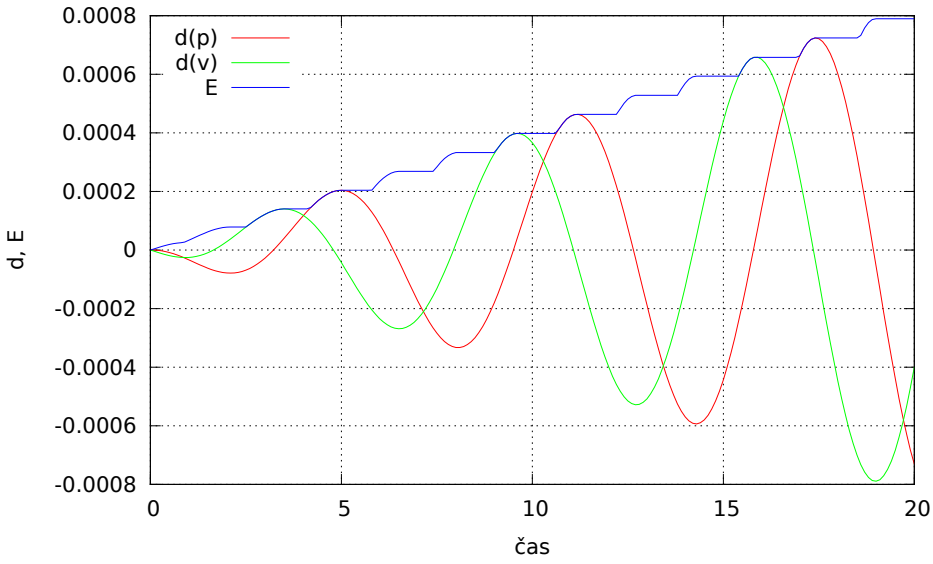
Aby sme zobrazili priebeh okamžitej chyby pri iných dĺžkach kroku, kde sa prejavujú iné efekty, musíme tieto dĺžky kroku najprv identifikovať. Zistíme závislosť relatívnej chyby E od kroku h (obrázok 7.8 log-log zobrazenie).

Na grafe môžeme rozoznať niekoľko úsekov zodpovedajúcich efektom nestability, orezania a zaokrúhľovania. Pri dĺžke kroku väčšej ako 2 sa najviac prejavuje nestabilita (viac o nestabilite v podkapitole 7.3). Keď sa dĺžka kroku ešte viac predlžuje, začne sa blížiť dĺžke simulácie a nameraná chyba stráca výpovednú hodnotu. Pri krokoch dĺžky 2 až 10^{-2} je hlavným zdrojom chyby orezanie. Sklon tejto časti krivky je približne rovný rádu použitej metódy. Pri dĺžke kroku 10^{-2} sa prejavuje približne rovnakým dielom orezanie aj zaokrúhľovanie, preto je celková chyba najmenšia. Ďalším skracovaním kroku sa vplyv zaokrúhľovania stupňuje, presahuje vplyv orezania, celková chyba sa zväčšuje. Priebehy chyby pre niektoré významné body z tohto grafu sú na obrázkoch 7.9 až 7.12.

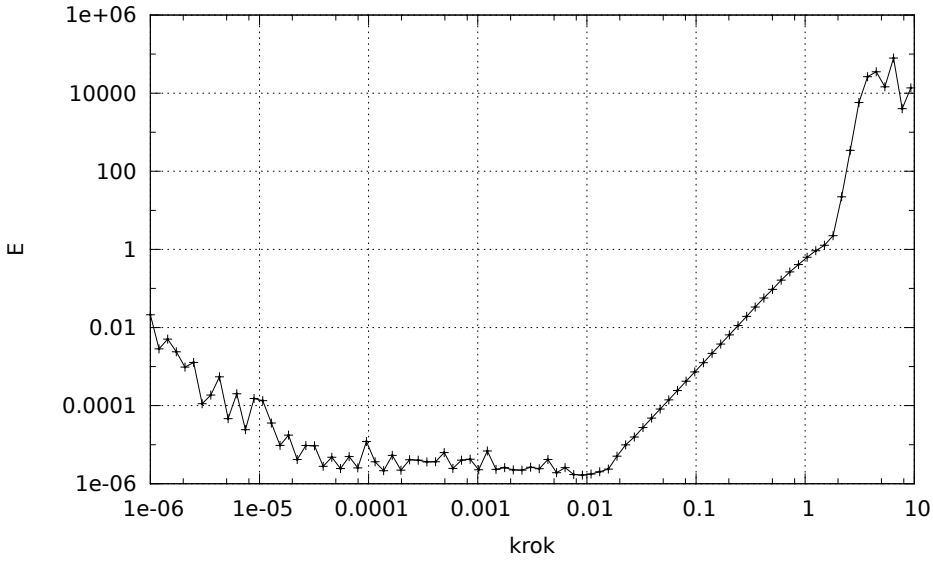
Skrátením kroku z 10^{-3} na 10^{-5} sa priebeh chyby výrazne zmenil. Priebeh chyby stratil svoj charakter a nadobudol viditeľné pravidelnosti. Chyba orezaním už nepripomína štvrtú deriváciu. Oproti priebehom s dlhšími krokmi je graf fázovo posunutý. Zaokrúhľovanie je



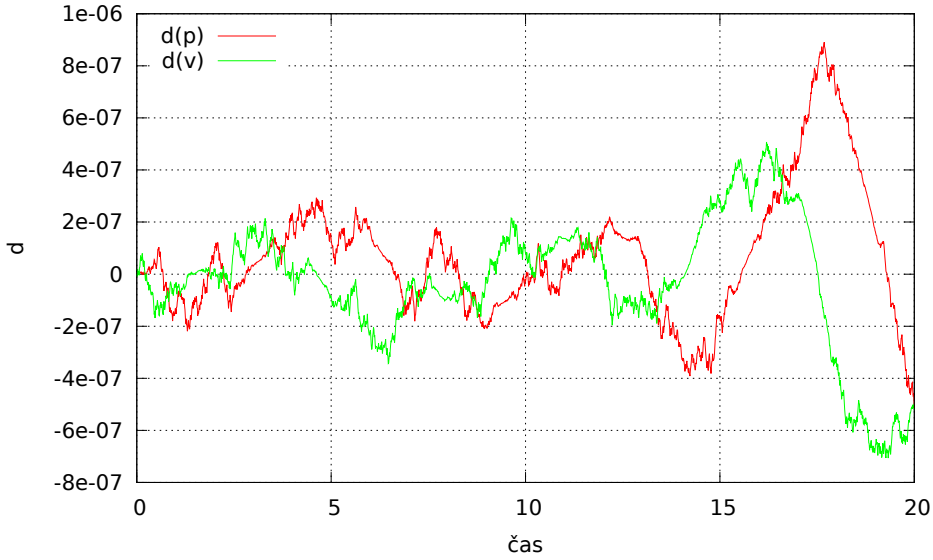
Obr. 7.6: Problém č. 1, dĺžka kroku 10^{-1}, rozdiel medzi analytickým a simulačným riešením.



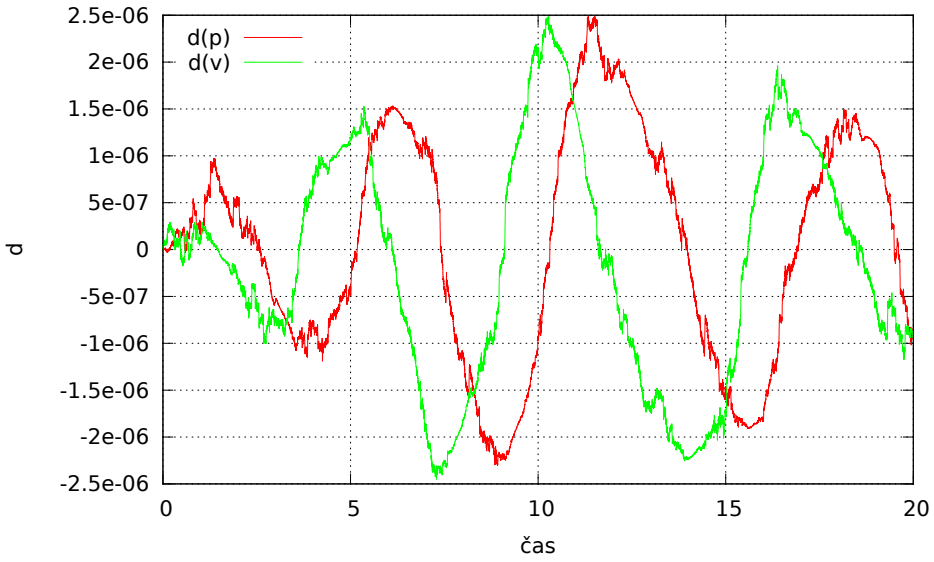
Obr. 7.7: Problém č. 1, dĺžka kroku 10^{-1}, vzťah medzi okamžitou odchýlkou d a maximálnou relatívnou chybou simulácie E.



Obr. 7.8: Problém č. 1, závislosť maximálnej chyby simulácie od dĺžky kroku.



Obr. 7.9: Priebeh okamžitej chyby d pre problém č. 1, dĺžka kroku 10^{-2}.



Obr. 7.10: Priebeh okamžitej chyby d pre problém č. 1, dĺžka kroku 10^{-3}.

také silné, že niektoré časti vzorca integračnej metódy vôbec neovplyvňujú výsledok, preto sa znížil rád metódy. Celková chyba prudko stúpla. Priblížili sme sa k hranici rozlišovacej schopnosti aritmetiky.

Tlmené kyvadlo

Simuláciu tlmeného kyvadla označíme ako *Problém číslo 2* s parametrami:

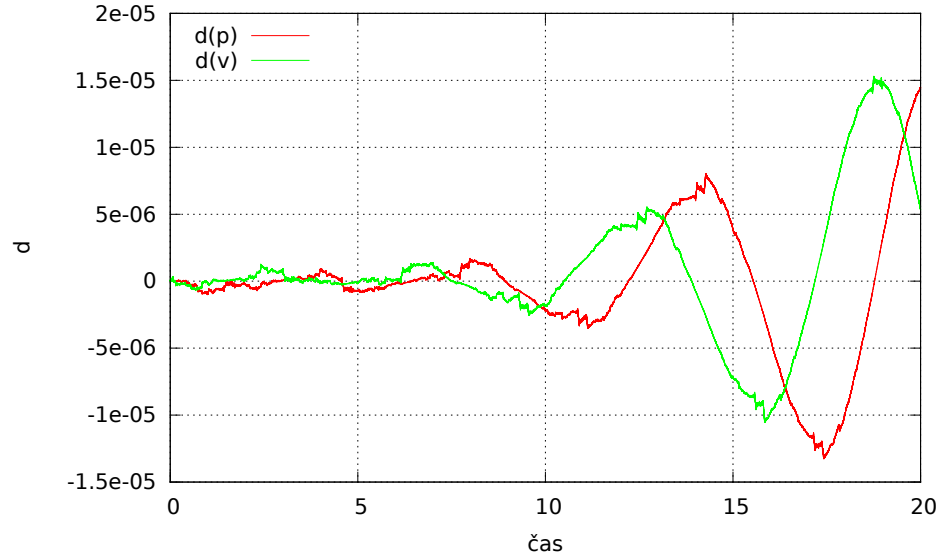
- model tlmeného kyvadla podľa rovnice (7.2),
- $T = 1$,
- $\sigma = 2$,
- modelový čas 0 - 5,
- mantisa 24 bitov,
- metóda RK3.

Časový priebeh okamžitej chyby d medzi analytickým a simulačným riešením je na obrázku 7.13. Pre dĺžku kroku 10^{-1}je riešenie stabilné. Chyba najprv s rastúcim časom rastie, ale od okamihu, kedy začne tlmenie systému prevažovať nad akumulovanou chybou začne chyba klesať. Simulačné riešenie sa časom začne približovať analytickému. Pre dĺžku kroku 4.10^{-1} je riešenie nestabilné. Chyba sa zväčšuje do nekonečna.

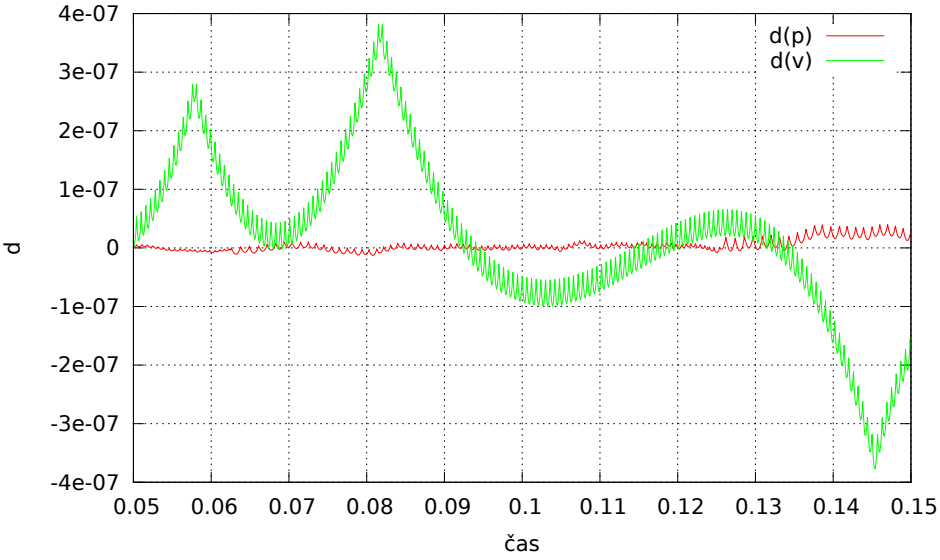
Parabola

Simuláciu tlmeného kyvadla označíme ako *Problém číslo 3* s parametrami:

- model polynomiálneho rastu podľa rovnice (7.4),
- $n = 2$,

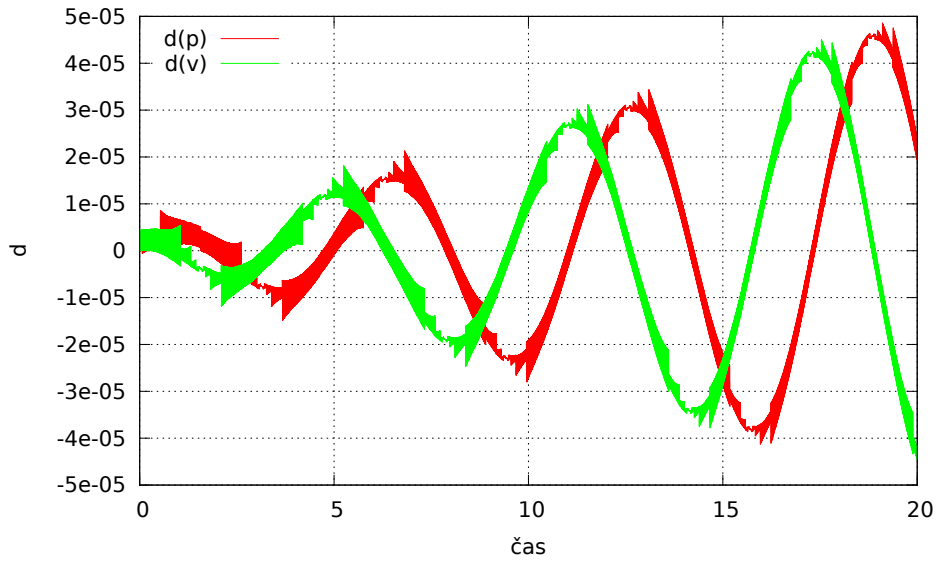


(a)

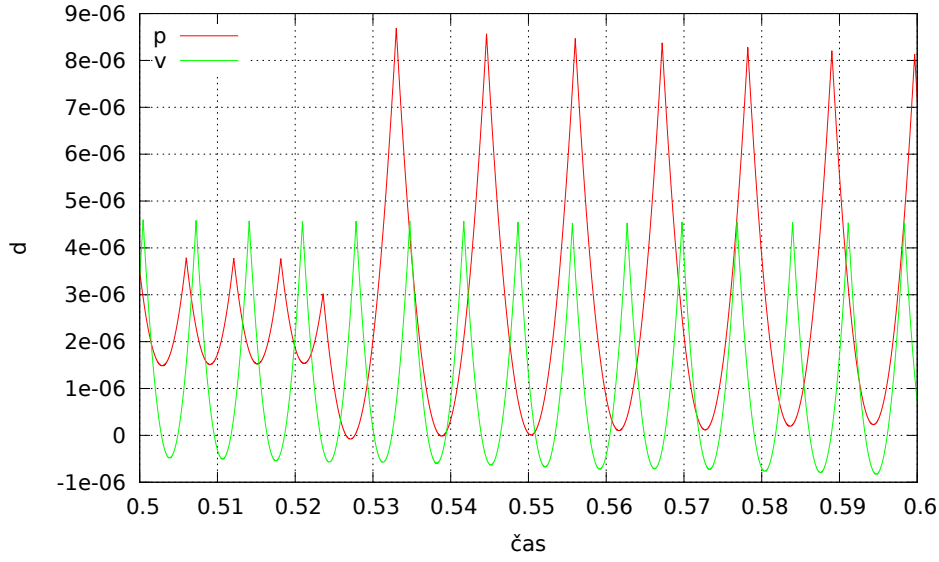


(b)

Obr. 7.11: Priebeh okamžitej chyby d pre problém č. 1, dĺžka kroku 10^{-4} (7.11a). Výrez z toho istého grafu (7.11b).

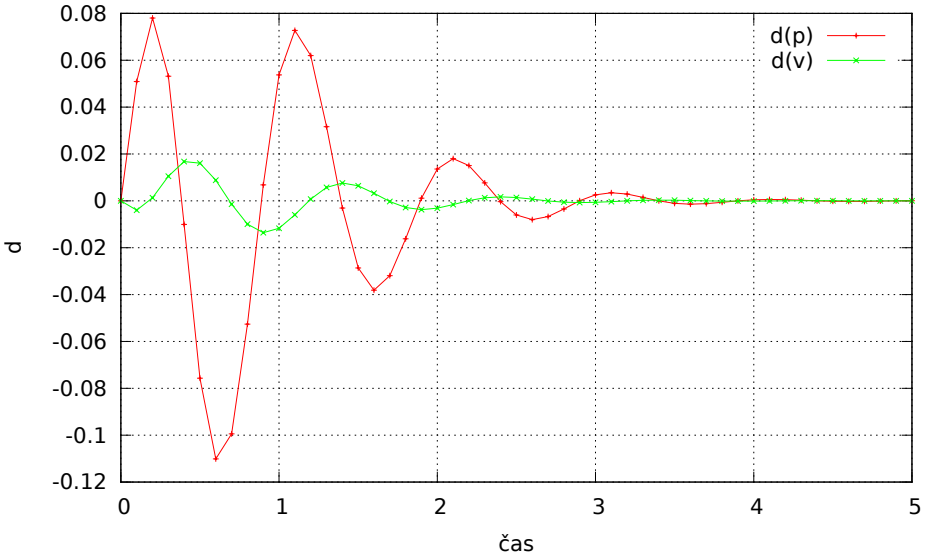


(a)

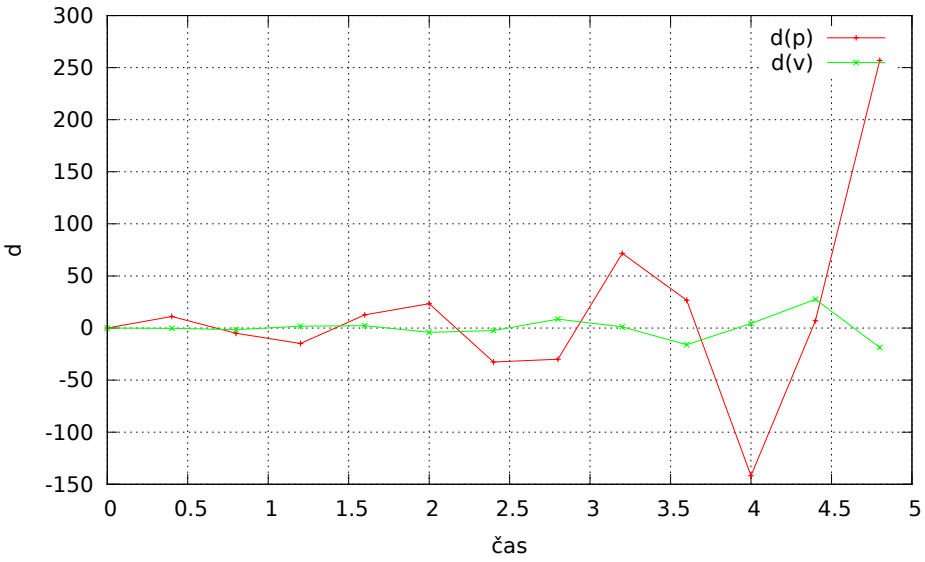


(b)

Obr. 7.12: Priebeh okamžitej chyby d pre problém č. 1, dĺžka kroku 10^{-5}. Výrez z toho istého grafu (7.12b).

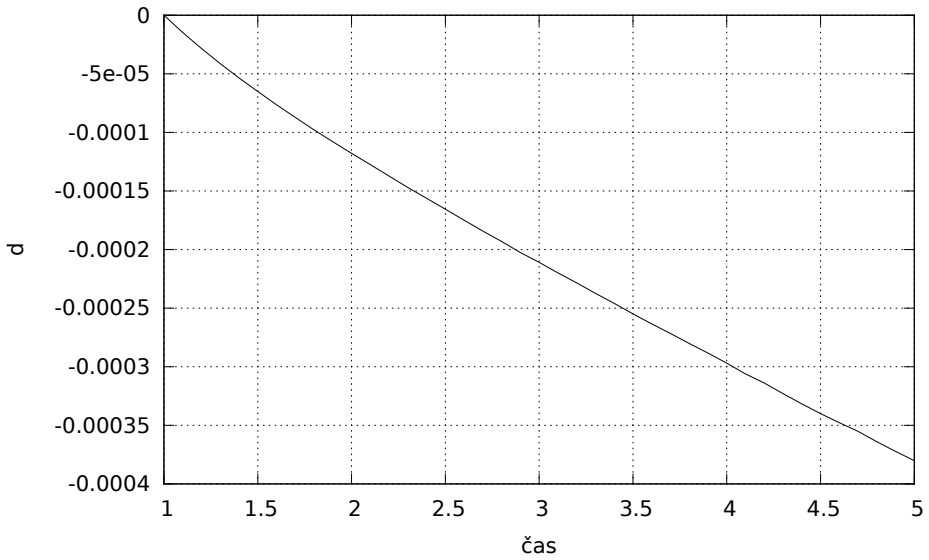


(a)



(b)

Obr. 7.13: Problém č. 2, rozdiel medzi analytickým a simulačným riešením pre dĺžku kroku 10^{-1} (7.13a) a $4 \cdot 10^{-1}$ (7.13b).



Obr. 7.14: Problém číslo 3, dĺžka kroku 10^{-1}, rozdiel medzi analytickým a simulačným riešením.

- modelový čas 0 - 5,
- mantisa 24 bitov,
- metóda RK3.

Parabola $y = t^2$ je monotónna funkcia, aj okamžitá chyba (hlavný zdroj je chyba orezaním) sa na rozdiel od netlmeného kyvadla mení monotónne (obrázok 7.14). Pri skrátení kroku na 10^{-2} priebeh okamžitej chyby prestane byť monotónny, začne sa prejavovať zaokrúhľovanie (obrázok 7.15).

7.4 Vplyv metódy, rádu a kroku na presnosť

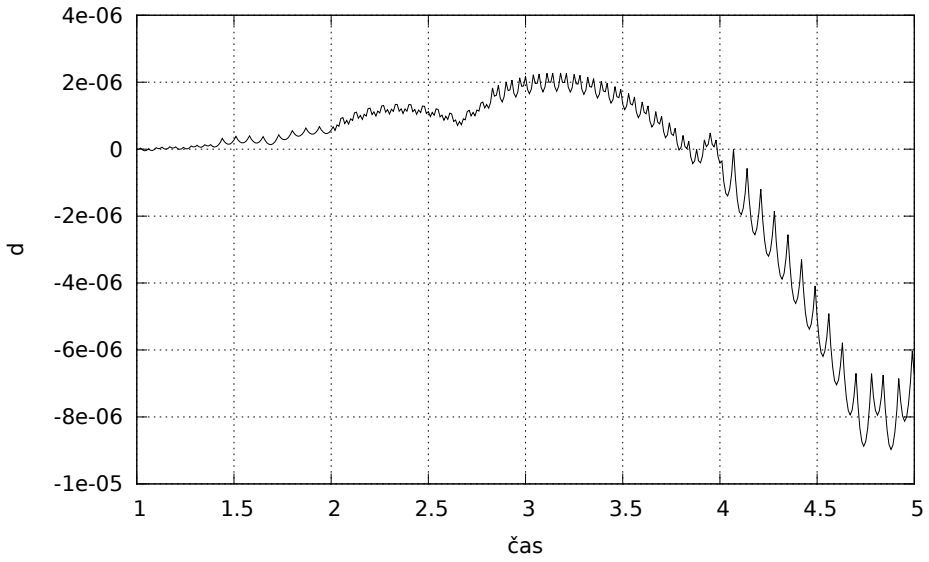
Graf závislosti relatívnej chyby simulácie E netlmeného kyvadla od kroku h z obrázku 7.8 teraz rozšírime o rôzne metódy. Pre orientáciu bude v každom grafe znázornená hladina označujúca epsilon práve použitej aritmetiky. Zo vzdialenosti relatívnej chyby od epsilonu na log-log grafe sa dá približne určiť, koľko najnižších bitov riešenia je nesprávnych.

Problém číslo 4 (je to problém č 1, ale s použitím iných integračných metód):

- model netlmeného kyvadla podľa rovnice (7.1),
- modelový čas 0 - 20,
- mantisa 24 bitov.

Metódy Runge-Kutta

Pre metódy Runge-Kutta rádu 1 až 7 dostaneme grafy na obrázku 7.16. Každá krivka vo svojom optimálnom bode (bod minimálnej chyby) dosahuje približne takú relatívnu chybu, ako všetky metódy vyššieho rádu pri rovnakej dĺžke kroku. Oblasť, v ktorej dominuje chyba



Obr. 7.15: Problém číslo 3, dĺžka kroku 10^{-2}, rozdiel medzi analytickým a simulačným riešením.

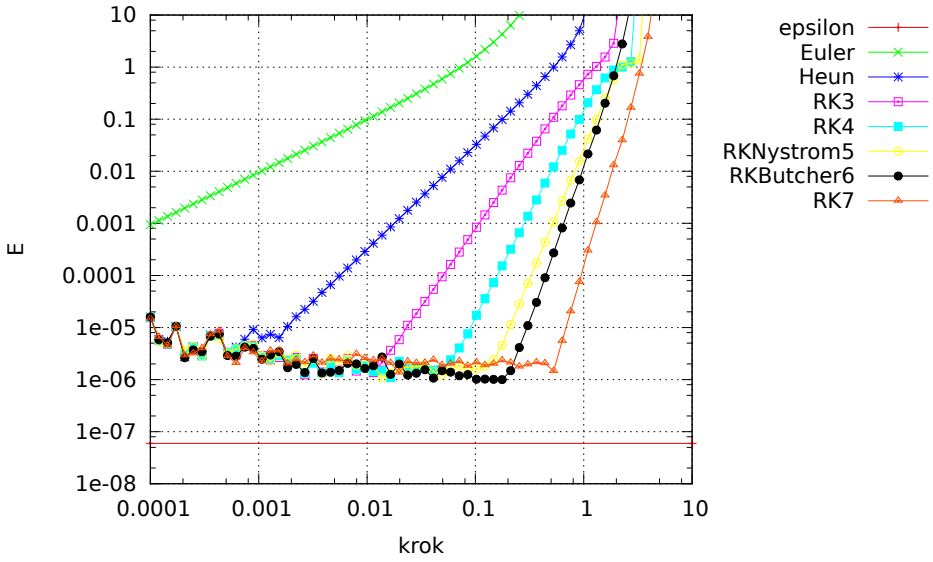
zaokrúhľovania nemá monotónny priebeh, prejavuje sa tam náhodný charakter zaokrúhľovania avšak trend chyby je monotónny. Za povšimnutie stojí, že krivky blízko optimálnych bodov sú zašumené, ale ďalej smerom k veľmi krátkym krokom majú všetky krivky takmer presne rovnaký, aj keď zašumený trend priebehu. Je to spôsobené tým, že zaokrúhľovanie postupne redukuje vplyv konštánt, ktorými sa jednotlivé metódy odlišujú. Ich vplyv vynásobený dĺžkou kroku a pripočítaný k okažitej hodnote stavových premenných je po zaokrúhlení nulový a každá metóda s takto krátkym krokom vypočíta približne to isté (v extrémnom prípade nie približne, ale presne to isté).

Metódy Adams-Bashforth

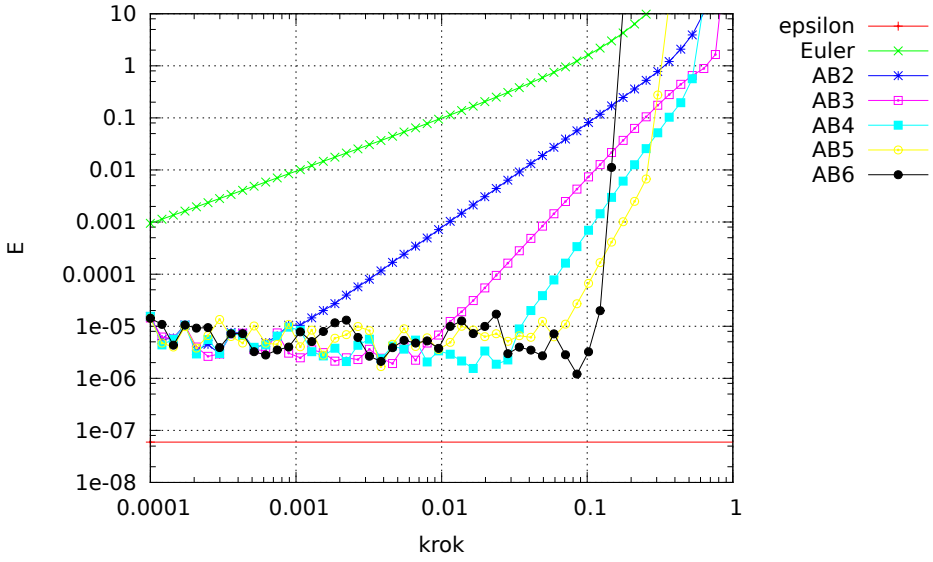
Ak namiesto metód Runge-Kutta použijeme Adams-Bashforth, dostaneme grafy na obrázku 7.17. Priebehy chyby su podobné ako v prípade metód RK. Rozdiel je, že so stúpajúcim rádom sa oblasť stability zmenšuje, preto sú metódy vyššieho rádu presnejšie ako nižšie rády iba pri malých dĺžkach kroku, pri dlhších krokoch to nemusí platiť.

Strmo klesajúce krivky v oblasti krokov 1 až 10 sú nevýznamné artefakty. Vznikli, pretože viackroková metóda Adams-Bashforth bola naštartovaná analytickým riešením. Keď sa dĺžka tohto štartovacieho úseku približuje dĺžke celej simulácie, aj presnosť sa blíži presnosti analytického riešenia.

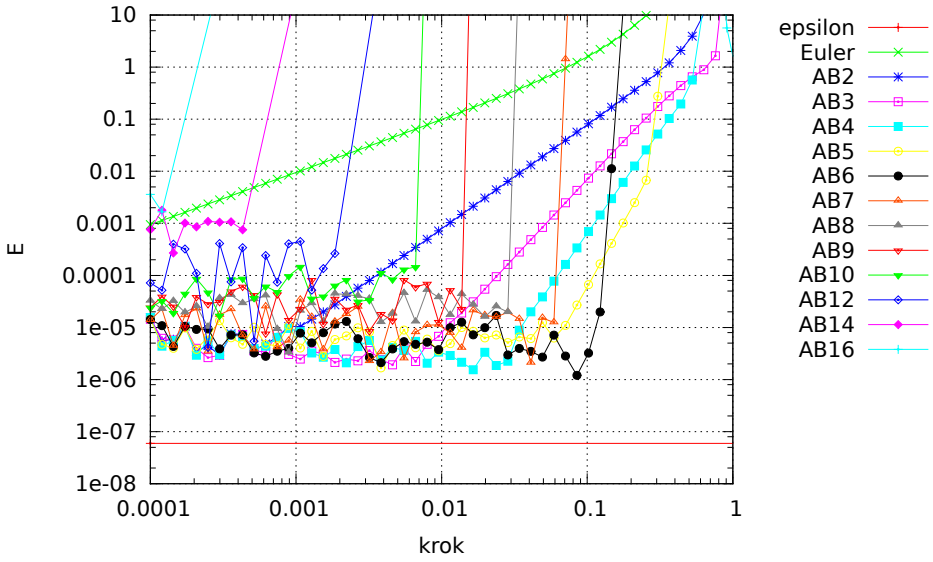
Pridním metód Adams-Bashforth vyšších rádov dostaneme graf na obrázku 7.18. Obsahuje v sebe celý graf (Obr.7.17), preto sa osobitný graf (Obr.7.17) môže zdať zbytočný, ale bol vložený pre prehľadnosť a analógiu s grafom na obrázku 7.16. Križujúce sa čiary v grafoch na obrázku 7.18 pôsobia neprehľadne. Na tomto grafe sa však ukázal nový jav. Metódy veľmi vysokých rádov dosahujú úplne inú chybu nezávislú od kroku. Priebeh chyby v čase pre jeden bod z tejto oblasti (10. rád, krok 10^{-3}, 24 bitov presnosť aritmetiky) je na obrázku 7.20. Môžeme sa presvedčiť, že to tiež súvisí so zaokrúhľovaním, keď spočítame rovnaký graf s použitím vyššej presnosti aritmetiky, napríklad 96 bitov (obrázok 7.19). Viac o vplyve presnosti aritmetiky je v kapitole 7.5.



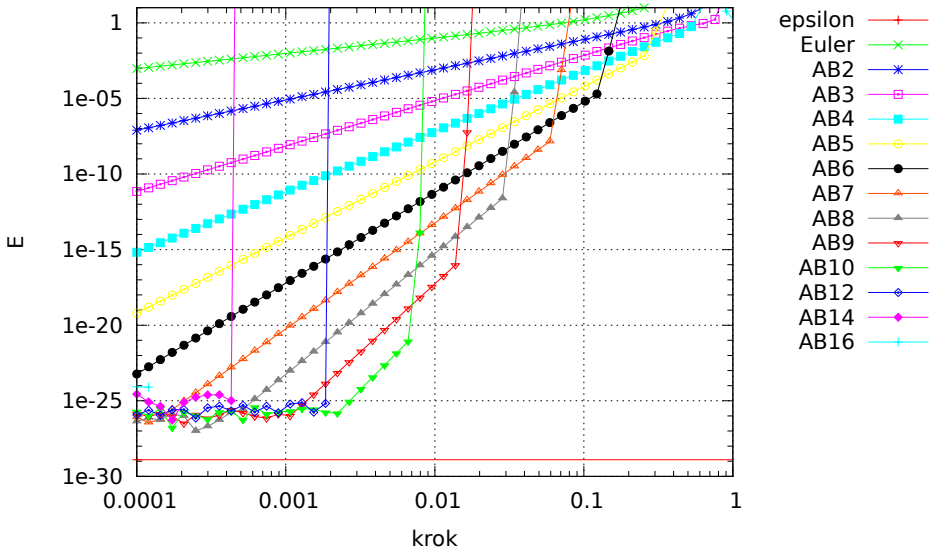
Obr. 7.16: Závislosť relatívnej chyby od kroku metód Runge-Kutta, problém č. 4.



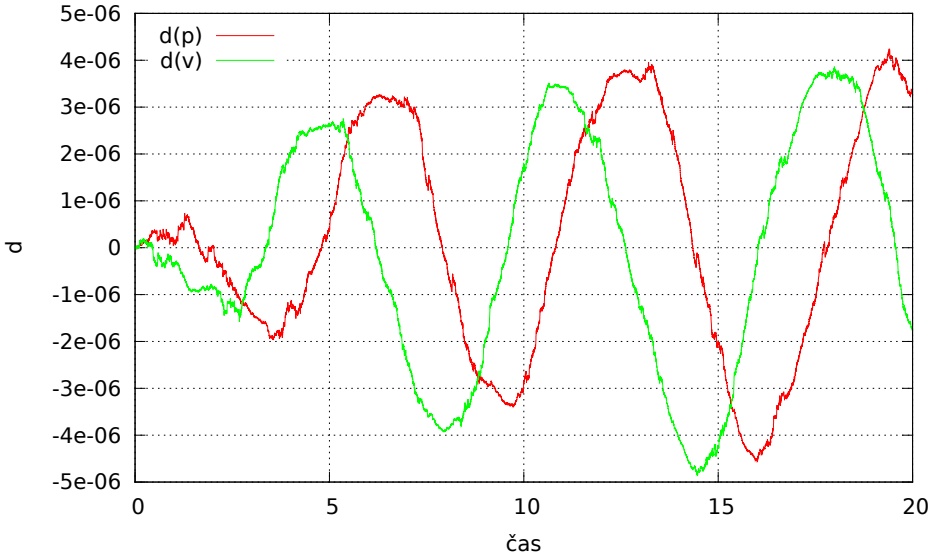
Obr. 7.17: Závislosť relatívnej chyby od kroku metód Adams-Bashforth, problém č. 4.



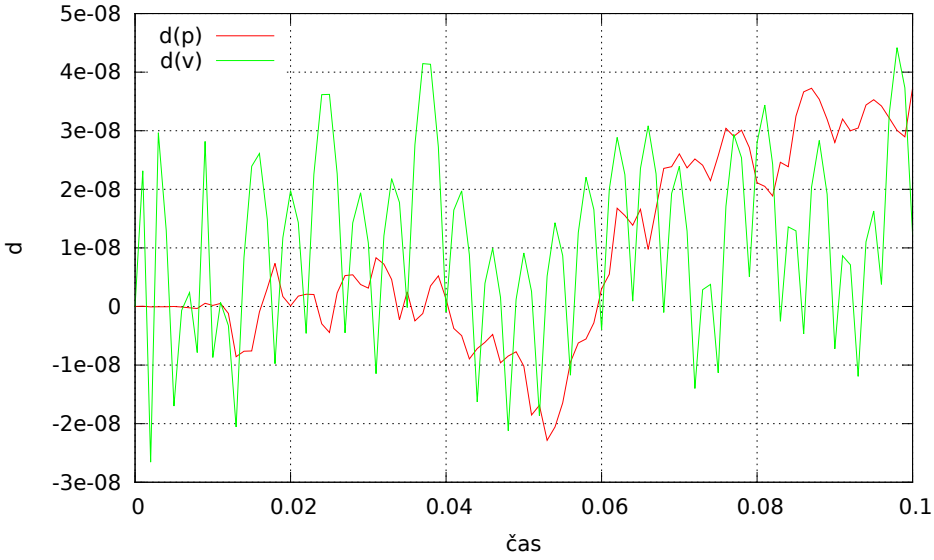
Obr. 7.18: Závislosť maximálnej chyby od kroku metód Adams-Bashforth vysokých rádov.



Obr. 7.19: Závislosť maximálnej chyby od kroku metód Adams-Bashforth s použitím viacnásobnej presnosti 96 bitov.

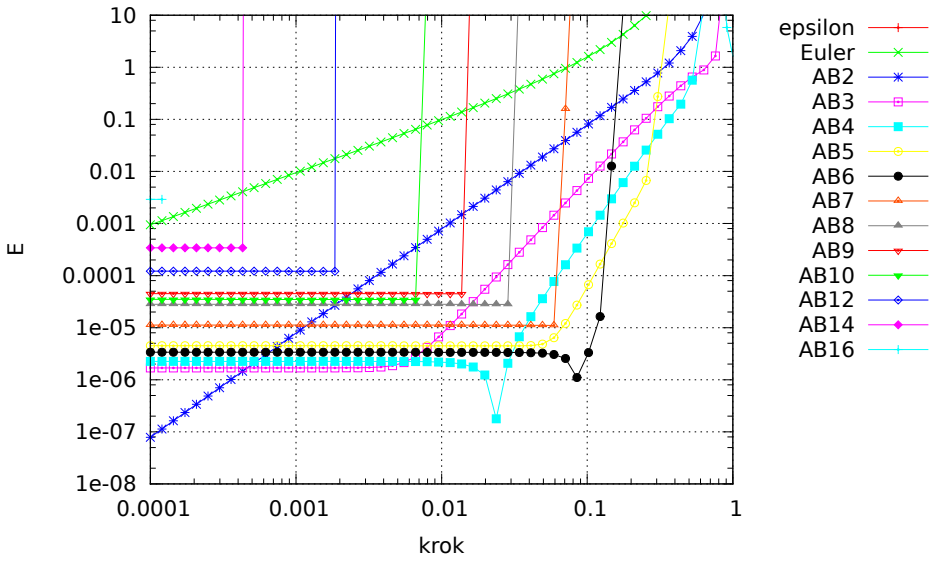


(a)



(b)

Obr. 7.20: Priebeh chyby v čase metódy AB10, problém č.4, krok 10^{-3} (7.20a)a výrez z tohto grafu (7.20b).



Obr. 7.21: Závislosť chyby od kroku pri presných výpočtoch, ale zaokrúhlených konštantách.

Vysvetlenie tejto nezávislosti chyby na kroku je v presnosti konštánt. Presvedčíme sa o tom, keď v simulácii použijeme vysokú presnosť aritmetiky (napr. 96 bitov), ale ešte pred začiatkom simulácie zaokrúhlime všetky konštanty na nižšiu presnosť, aká bola použitá v pôvodnom experimente 7.18 (24 bitov). Dostaneme graf na obrázku 7.21. Je veľmi podobný grafu (7.18). Vyplýva z toho záver, že nie je dobré používať metódy Adams-Bashforth vysokých rádov bez predchádzajúcej analýzy. Okrem rizika nestability je aj riziko veľkých zaokrúhľovacích chýb, ktorým sa vyhneme metódami nižšieho rádu. Použitie metód Adams-Bashforth vysokých rádov je výhodné v prípade, že vyžadujeme veľmi presné riešenie, máme k dispozícii veľmi presnú aritmetiku a tolerujeme krátke kroky simulácie.

Vplyv rádu metódy

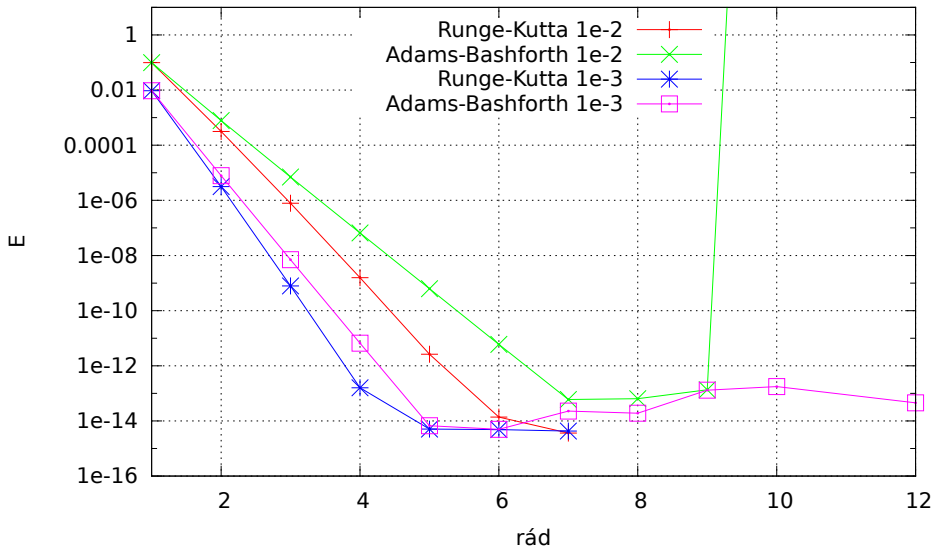
Vplyv rádu metódy na presnosť simulácie už bol obsiahnutý v predchádzajúcej časti o vplyve kroku na presnosť. Na sprehľadnenie zobrazíme rovnaké údaje do grafu iným spôsobom (obrázok 7.22a). Ide o netlmené kyvadlo, pre konštantný krok 10^{-2} a 10^{-3}, dĺžka simulácie 20, dvojnásobná presnosť (53 bitov). Ako je vydieť z grafu, so zvyšujúcim rádom metódy dochádza k lineárnemu poklesu relatívnej chyby (na log-log škále) až po dosiahnutie oblasti saturácie, ktorá je charakteristická dominanciou chyby zaokrúhľovaním.

Vplyv testovacieho systému

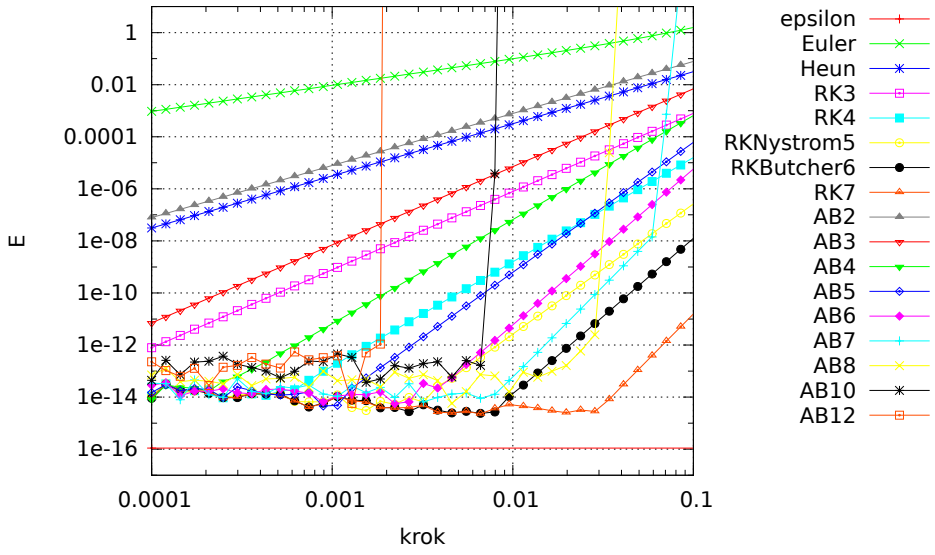
Doteraz sme používali ako testovací systém netlmené kyvadlo, teraz ho vymeníme za parabolu a budeme sledovať, či sa zmení dosiahnutá relatívna chyba. Zvolíme kratší čas simulácie a dvojnásobnú presnosť, aby sme nenarazili na pretečenie referenčného riešenia.

Problém číslo 5:

- model polynomiálneho rastu podľa rovnice (7.4),
- modelový čas 0 - 20,
- mantisa 23 bitov.



(a)



(b)

Obr. 7.22: Vplyv rádu na presnosť simulácie netlmeného kyvadla pre kroky 10^{-2} a 10^{-3} (7.22a). Chyba pre rôzne kroky pri ostatných nastaveniach rovnakých (7.22b).

Z výsledkov pre $n = 2$ na obrázku 7.23 vidieť, že:

- Runge-Kutta šiesteho rádu sa ukazuje vyššieho rádu ako Runge-Kutta siedmeho rádu.
- Všetky metódy Adams-Bashforth okrem Eulerovej dosahujú vynikajúcu presnosť a žiadnu chybu orezaním.

Ak použijeme ako testovací systém namiesto druhej mocniny piatu ($n = 5$), dostaneme graf na obrázku 7.24. Metódy Adams-Bashforth rádu 4 a nižšieho vykazujú bežnú chybu orezaním, naproti tomu metódy rádu 5 a vyššieho žiadnu. Vyplýva to z princípu metódy (extrapolácia polynómom). Ak riešením je polynóm piateho stupňa a extrapolujeme polynómom piateho (a vyššieho) stupňa, máme v rámci zaokrúhľovania presné riešenie. Ak extrapolujeme polynómom nižšieho stupňa, zanedbané vyššie derivácie sa prejavia ako chyba orezaním.

7.5 Vplyv zaokrúhľovania aritmetických operacií na presnosť metód

Graf z obrázku 7.8 teraz rozšírime o rôzne presnosti aritmetiky. Pretože tento experiment je náročnejší na procesorový čas, použijeme metódu vyššieho rádu (Runge-Kutta 6. rádu), aby sa želané efekty prejavili pri dlhších krokoch. Tiež skrátime dĺžku simulácie.

Problém číslo 6:

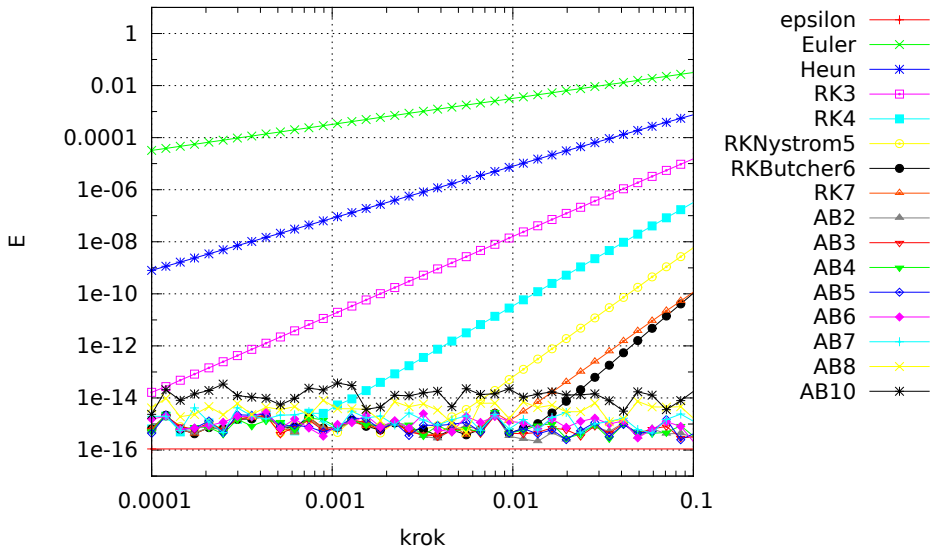
- model netlmeného kyvadla podľa rovnice (7.1),
- metóda RK6
- modelový čas 0 - 2.

Výsledné grafy sú na obrázku 7.25a. V grafe 7.25b boli odstránené časti, kde prevláda chyba orezaním, a ponechané sú len tie časti, kde dominuje chyba zaokrúhľovaním. Tiež tam boli doplnené priamky získané lineárnou regresiou. Lineárne sa javia iba na log-log grafe,na lineárnej škále sú to polynómy, preto bola lineárna regresia použitá na zlogaritmované hodnoty chyby aj kroku. Tieto priamky je možné popísať parametrami p (strmosť) a q (vertikálne posunutie) na osiach x, y pričom Δ_q je rozdiel q parametrov dvoch priamok. Pre log-log zobrazenie ďalej platí:

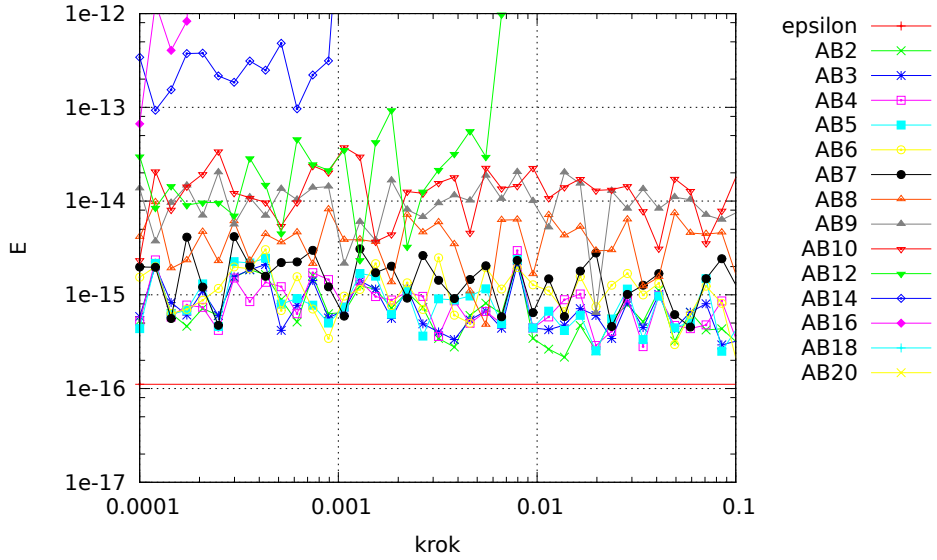
$$\begin{aligned} x &= \log_{10}(h) \\ y &= px + q \\ E &= 10^y \end{aligned}$$

Hodnoty regresných parametrov p a q pre danú simuláciu su zoradené v nasledovnej tabuľke.

bity	p	q	Δq
24	$-0,323$	$-7,14$	–
32	$-0,421$	$-9,72$	$2,58$
40	$-0,462$	$-12,2$	$2,46$
48	$-0,515$	$-14,8$	$2,64$
56	$-0,497$	$-17,1$	$2,31$
64	$-0,468$	$-19,5$	2.34
priemer	$-0,448$	–	2.45

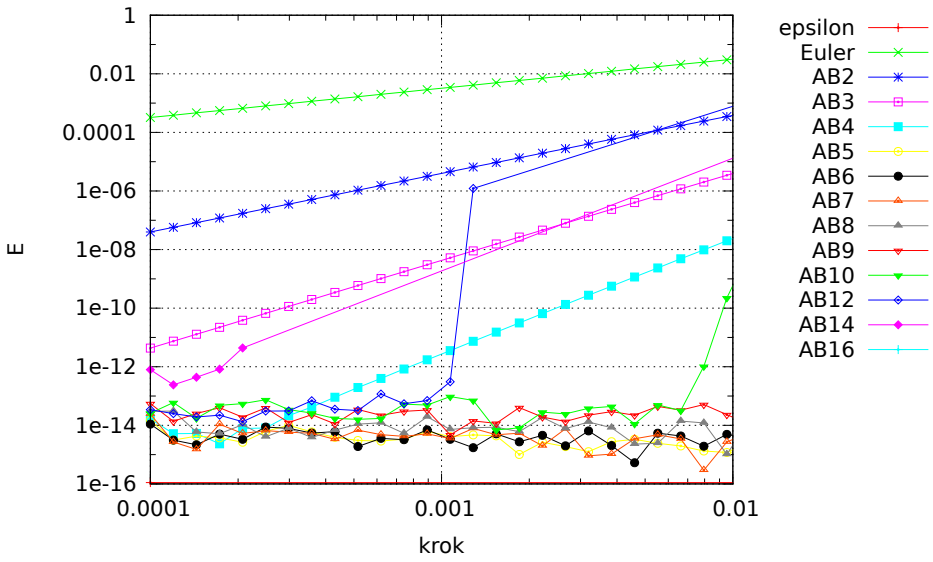


(a)



(b) Výrez, ktorý obsahuje iba metódy Adams-Bashforth.

Obr. 7.23: Problém č.5, $n = 2$, závislosť maximálnej relatívnej chyby na dĺžke kroku.



Obr. 7.24: Problém č.5, $n = 5$, závislosť chyby od dĺžky kroku.

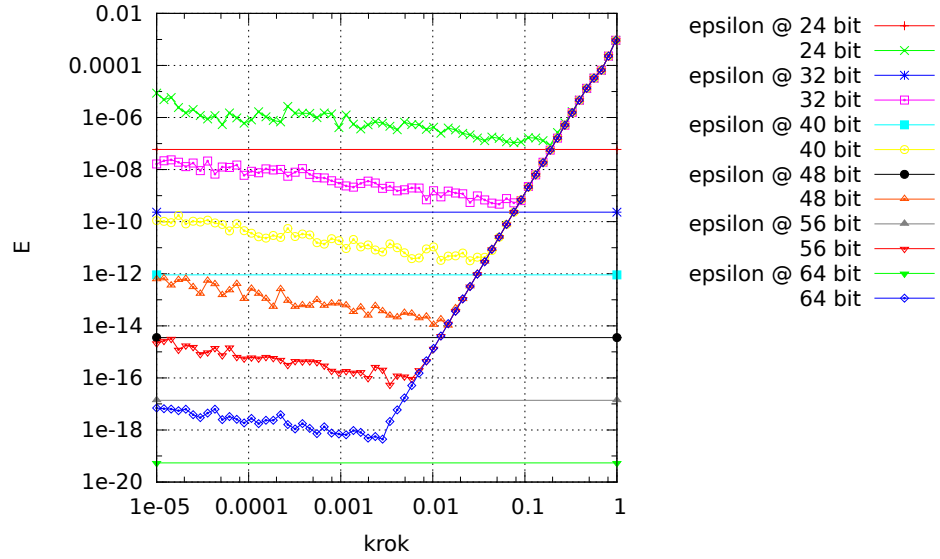
Z tabuľky je zrejmé, sklon priamok $y = px + q$ je približne rovnaký (podobné p) a že konštantnému prírastku presnosti (v tomto prípade 8 bitov) zodpovedá približne konštantný prírastok $\Delta_q$na log-log mierke. Ako kontrolu správnosti a presnosti môžeme overiť, že ak zachováme rovnakú dĺžku kroku a pridáme aritmetike 8 bitov šírky, chyba sa zlepší priemerne $10^{\Delta b}$-násobne. $10^{\Delta b} \doteq 10^{2.45} \doteq 281 \doteq 2^{8,13} \doteq 2^8$, takže chyba sa naozaj zlepší približne o 8 bitov. Uvedené vzťahy platia iba približne, pretože namerané údaje majú veľký rozptyl a výpočty v logaritmickej mierke sú veľmi citlivé na presnosť.

Minimá kriviek pre jednotlivé počty bitov nie sú pri rovnakej dĺžke kroku. Pridaním bitov sa optimálny bod posúva smerom ku kratším krokom. Preto, ak máme simuláciu s optimálnou dĺžkou kroku a pridáme 8 bitov, chyba sa nezlepší o celých 8 bitov. Musíme skrátiť dĺžku kroku a chyba zaokrúhlením sa pri skracovaní kroku zhoršuje. O koľko musíme skrátiť krok, to záleží od sklonu časti krivky dominovanej orezaním, a ten závisí od rádu metódy.

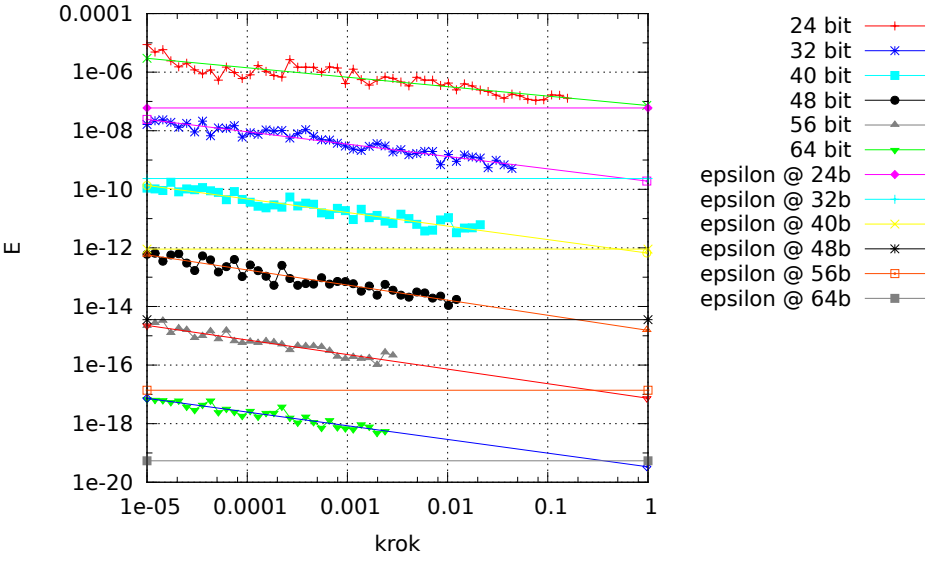
7.6 Vzťah medzi optimálnymi bodmi

Bod, ležiaci na grafe závislosti relatívnej chyby E od kroku h, v ktorom majú rovnaký vplyv chyby orezaním a zaokrúhlením, a v ktorom je relatívna chyba najmenšia, budeme volať *optimálny bod*.

V praxi môže nastať situácia, keď máme experimentálne zistený optimálny bod simulácie pre určitý počet bitov. Potom sa sprísnia požiadavky na presnosť a je potrebné určiť nový vyhovujúci optimálny bod. Úloha je teda nájsť novú dĺžku kroku a novú presnosť aritmetiky. Teraz sa pokúsime odvodiť približný vzťah pre odhad týchto hodnôt. Celý výpočet bude prebiehať na logaritmickej mierke. Na začiatku zlogaritmujeme vstupné čísla, výsledok prevedieme z logaritmickej mierky opäť do lineárnej. Aby sme zachovali ekvivalenciu so všetkými zobrazenými grafmi, budeme používať dekadický logaritmus. Počet pridaných bitov je dvojkový logaritmus, preto ho nakoniec prepočítame. Geometrický význam nasledujúcich vzťahov je znázornený na obrázku 7.26. Nech je:



(a)



(b)

Obr. 7.25: Problém č. 6, závislosť chyby od kroku pre rôzne počty bitov (7.25a). Separované klesajúce časti kriviek (7.25b).

- E – chyba v známom optimálnom bode,
- E_x– želaná chyba,
- Δ_E – rozdiel chyby na logaritmickej mierke.

Potom pre rozdiel chyby v logaritmickej mierke platí

$$\Delta_E = log_{10}E - \log_{10} E_x \tag{7.6}$$

Zavedením parametrov:

- h – dĺžka kroku v známom optimálnom bode,
- h_x – hľadaná dĺžka kroku,
- Δ_h – zmena kroku v logaritmickej mierke

platí pre zmenu kroku Δ_h na logaritmickej mierke

$$\Delta_h = log_{10}h - \log_{10} h_x \tag{7.7}$$

Ak teraz definujeme:

- α – uhol medzi vodorovnou osou a časťou krivky, kde prevláda chyba orezaním (krivka zobrazená na log-log grafe). $r = \tan(\alpha)$, pričom$r$ je rád integračnej metódy
- β – uhol medzi vodorovnou osou a regresnou priamkou: $v = -\tan(\beta)$ z grafu 7.25b (použijeme zaokrúhlenú hodnotu -0,5)
- Δ_d – prírastok presnosti aritmetiky vyjadrený v počte dekadických číslic, je to ekvivalent posunutia regresnej priamky z grafu (7.25b) a tiež ekvivalent posunutia strojového ε,

potom približne platí:

$$\begin{aligned} \Delta_h &= \frac{\Delta_E}{\tan(\alpha)} = \frac{\Delta_E}{r} \\ \Delta_d &= \Delta_E + \Delta_h \tan(\beta) = \Delta_E - \Delta_h v \end{aligned}$$

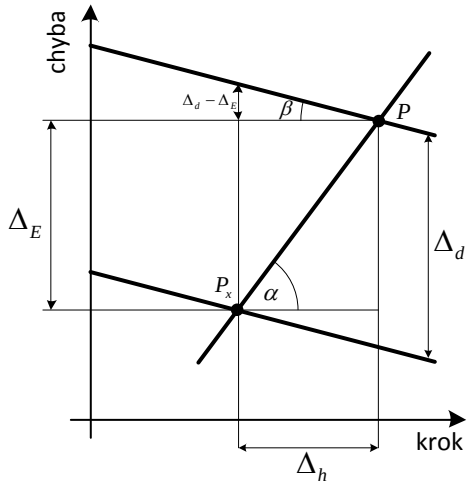
$$\Delta_d = \Delta_h(\frac{1}{r} - v) = \Delta_E(1 - \frac{v}{r}) \tag{7.8}$$

Z výsledku vo vzťahu (7.8) vyplýva, že (na log-log mierke) zmena presnosti aritmetiky Δ_d je priamo úmerná zmene kroku Δ_h a zmene relatívnej chyby Δ_E. Konštanty úmernosti sú pritom určené len sklonmi priamok r a v. Pomocou $\Delta_h$môžme teraz prepočítať potrebnú dĺžku kroku $h_x$na lineárnej mierke

$$h_x = 10^{log_{10}h - \Delta_h} \tag{7.9}$$

Počet bitov, o ktorý je potrebné rozšíriť mantisu použitých čísel Δ_b je daný vzťahom

$$\Delta_b = \log_2(10^{\Delta_d}) = \frac{\Delta_d}{\log_{10} 2} \tag{7.10}$$



Obr. 7.26: Znázornenie závislosti medzi optimálnymi bodmi na log-log škále. P je referenčný optimálny bod, P_x hľadaný.

Použitie týchto vzťahov demonštruje nasledujúci príklad. Pre našu simuláciu máme zistený optimálny bod P s dĺžkou kroku $h = 10^{-1}$ a chybou $E = 10^{-7}$ za použitia mantisy 24 bitov a metódy šiesteho rádu t.j. $r = 6$. Teraz potrebujeme presnejšiu simuláciu toho istého systému s chybou $E_x = 10^{-19}$. Riešenia je uvedené v nasledujúcom výpočte

$\Delta_E = (\log_{10} 10^{-7}) - (\log_{10} 10^{-19}) = 12$
$\Delta_h = \frac{12}{6} = 2$
$h_x = 10^{log_{10}10^{-1}-2} = 10^{-3}$
$\Delta_d = 12 - 2.(-0.5) = 13$
$\Delta_b = \frac{13}{log_{10}2} \doteq 43$

Čiže pre novú simuláciu musíme použiť dĺžku kroku $h_x = 10^{-3}$ a šírku mantisy zvýšiť o $\Delta_b = 43$ bitov (na 24+43 = 67 bitov). Pohľadom na graf na obrázku 7.25a sa presvedčíme, že tieto výsledky sú približne správne.

7.7 Vplyv presnosti aritmetických operacií na efektivitu metód

V predchádzajúcej časti sme sa venovali závislosti chyby od dĺžky kroku, pretože dĺžka kroku:

- Je nastaviteľná.
- Podstatne ovplyvňuje chybu.
- Podstatne ovplyvňuje strojový čas potrebný na simuláciu. Problém je, že čas potrebný na jeden krok rôznych metód nie je rovnaký.

Ak chceme porovnávať efektivitu metód, musíme vyhodnotiť, koľko strojového času ktorá metóda vyžaduje na dosiahnutie danej presnosti.

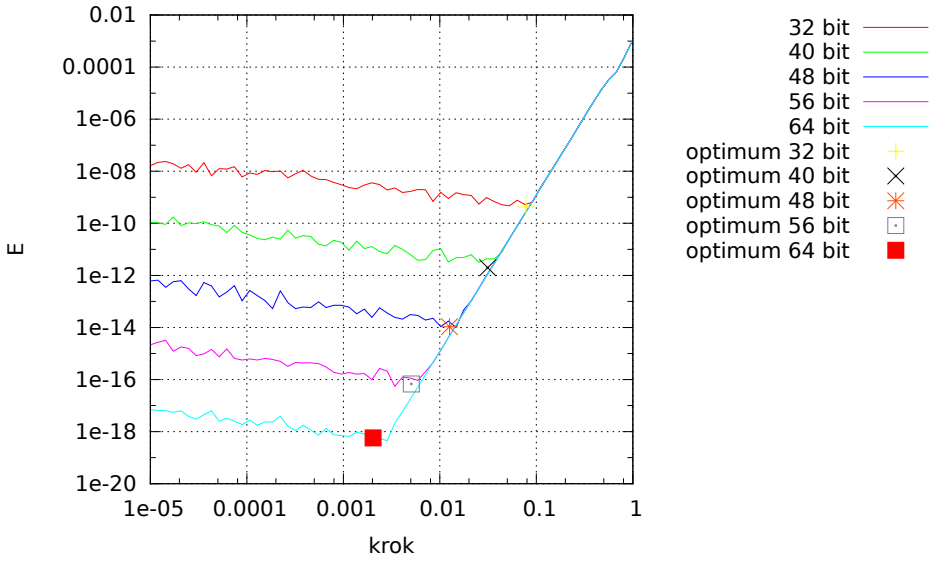
Teraz sa zameriame na optimálny bod definovaný v podkapitole 7.6 a bude nás zaujímať reálny strojový čas simulácie v tomto bode. Pokúsime sa nájsť trendy pre rôzne metódy a vzájomne ich porovnať.

Hľadanie optimálneho bodu zautomatizujeme tak, aby sme sa pokiaľ možno vyhli simuláciám s krátkym krokom a tak ušetrili procesorový čas. Využijeme vzťahy z predchádzajúcej podkapitoly. Tiež budeme predpokladať, že v nejakom úseku grafu závislosť chyby na kroku približne zodpovedá rádu metódy (oblasť chyby orezaním). Algoritmus určovania optimálnych bodov možno zosumarizovať do nasledovných bodov:

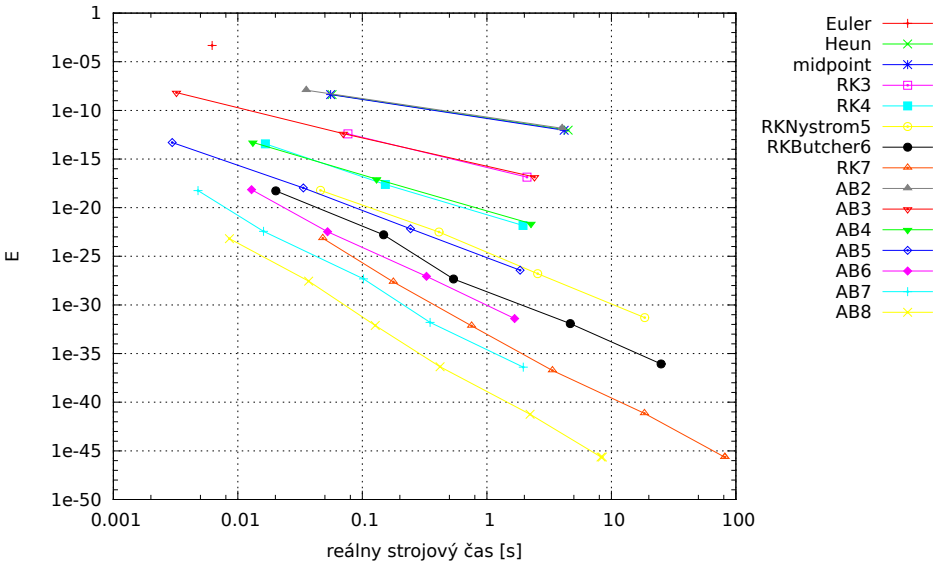
1. Vyberieme si testovací systém a dĺžku simulácie.
2. Vyberieme si integračnú metódu a aritmetiku.
3. Nastavíme počiatočný odhad dĺžky kroku, ktorý je určite dlhší, ako krok v optimálnom bode.
4. Simulujeme a meriame dosiahnutú chybu.
5. Zmenšíme dĺžku kroku, znova simulujeme a meriame chybu.
6. Pokračujeme v skracovaní kroku, až kým pokles chyby nezodpovedá rádu metódy. Tým sme prekonali oblasť nestability.
7. Pokračujeme v skracovaní kroku, až kým sa chyba nezačne znova zvyšovať.
8. Nájdené minimum vyhlásime za približný optimálny bod, zaznamenáme pre neho dĺžku kroku, chybu, presnosť použitej aritmetiky a trvanie simulácie.
9. Opakujeme pre rôzne presnosti aritmetiky a rôzne integračné metódy.

Hľadanie optimálneho bodu je vždy nepresné. Okrem obmedzenej hustoty vzorkovania dĺžky krokov je zneistené meranie chyby tým, že zaokrúhľovacie chyby majú vždy náhodný charakter. Príklad nájdenia optimálnych bodov pre problém č. 6 je na obrázku 7.27. Spresniť by sa dalo napríklad opakovaním pokusu s rôznymi počiatočnými podmienkami a spriemerovaním.

Príklad výsledkov experimentu pre netlmené kyvadlo simulované do modelového času 2 sú na grafe (7.28). Vidíme, že čím je metóda vyššieho rádu, tým menej času potrebuje na dosiahnutie danej chyby. Metódy Runge-Kutta a Adams-Bashforth rovnakého rádu sú približne rovnako efektívne. Pri najvyšších rádoch je efektívnejší Adams-Bashforth.



Obr. 7.27: Problém č. 6, závislosť chyby od kroku a nájdené optimálne body.



Obr. 7.28: Závislosť chyby simulácie od spotrebovaného strojového času v optimálnych bodoch. Každá čiara predstavuje jednu metódu, každý bod na tejto čiare jednu optimálnu kombináciu bitov a kroku. (netlmené kyvadlo, modelový čas 0-2)

Kapitola 8

Záver

V knihe venovanej spojitej simulácii [5] odporúča profesor Cellier prispôsobiť rád integračnej metódy požiadavke na presnosť simulácie s odôvodnením, že nemá zmysel mrhať strojovým časom pri výpočte presnej metódy, keď máme nepresný model. Ďalej odporúča pri použití aritmetiky jednoduchej presnosti iba Eulerovu metódu, pri dvojnásobnej presnosti metódu štvrtého rádu. Neuvádza reprodukovateľné experimenty, ktoré by to potvrdzovali. Experimenty v tejto práci naproti tomu naznačujú, že aj metódy vyšších rádov (6 - 10) sú použiteľné s obyčajnou presnosťou aritmetiky. Metódy Runge-Kutta vyšších rádov umožňujú predĺžiť krok simulácie, a teda celý proces zrýchliť a zefektívniť. Pri metódach Adams-Bashforth je obmedzením nestabilita, ktorá nedovolí predlžovať krok. Na základe toho by sa dalo odporúčať použitie metód Runge-Kutta čo najvyššieho rádu a dvojnásobnú presnosť (double). Iba ak sú požiadavky na presnosť simulácie také prísne, že je nutná viacnásobná presnosť, nastupujú metódy Adams-Bashfort dvojciferných rádov.

Implementovaná knižnica obsahuje metódy Runge-Kutta do rádu 7 a Adams-Bashforth do rádu 20. Námetom na ďalšiu prácu je rozšírenie o metódy Runke-Kutta vyšších rádov, ktorých koeficienty nie sú racionálne čísla, ale riešenia nelineárnych rovníc. Tieto sa musia vypočítať podľa aktuálnej požadovanej presnosti. Ďalej by sa dali experimenty rozšíriť o iné skupiny metód, napríklad implicitné. V experimentoch boli použité jednoduché modely so známym analytickým riešením. Vhodným rozšírením do budúcna je možnosť odhadu chyby iba na základe viacerých simulácií s rôznymi nastaveniami, čo by umožnilo doplniť experimenty o rôzne zaujímavé a prakticky významné modely.

Skratky a symboly

t	čas
f	funkcia na výpočet derivácie
y	stav systému
y'	derivácia stavu podľa času
$y^{(n)}$	n-tá derivácia stavu podľa času
y_r	referenčná hodnota stavu vypočítaná analyticky
ulp	*unit in the last place*, jednotka na poslednom mieste
ε	epsilon, maximálna relatívna chyba zaokrúhlením na daný počet bitov
n	prirodzené číslo
h	dĺžka kroku integračnej metódy, $h = \Delta t = t_n - t_{n-1}$
d	rozdiel medzi skutočnou a vypočítanou hodnotou
E	celková chyba simulácie
RK3	Runge-Kutta tretieho rádu
AB6	Adams-Bashforth šiesteho rádu

Literatúra

[1] HAMMING, Richard. *Numerical methods for scientists and engineers.* New York: McGrav-Hill, 1962.

[2] CELLIER, Francois. *Continuous System Modeling.* New York: Springer-Verlag, 1991.

[3] HÖRZ, H., WESSEL, K. *Philosophische Entwicklungstheorie*, Berlin, 1983.

[4] BIRKHOFF, George. *Dynamical Systems.* Rev. Ed., Providence: AMS, 1966.

[5] CELLIER, F., KOFMAN, E. *Continuous System Simulation.* New York: Springer-Verlag, 2006.

[6] RÁBOVÁ, Z. et al. *Modelování a simulace.* Brno: VUT v Brně, 1992. ISBN 80-214-0480-9

[7] GOLDBERG, David. *What every computer scientist should know about floating-point arithmetic.* ACM Computing Surveys (CSUR), v.23 n.1, Mar. 1991.

[8] KAHAN, Wiliam. *Pracniques: further remarks on reducing truncation errors.* Communications of the ACM, v.8 n.1, Jan 1965.

[9] MOORE, Ramon. *Interval arithmetic and automatic error analysis in digital computing.* Stanford: Stanford University, 1963.

[10] CHAPRA, S., CANALE, R. *Numerical Methods for Engineers: With Programming and Software Applications.* New York: McGraw-Hill, 1997. ISBN:0070109389

[11] WERSCHULZ, Arthur. *Computational Complexity of One-Step Methods for Systems of Differential Equations.* Mathematics of Computation Vol. 34, No. 149, Jan 1980

[12] BUTCHER, John. *Numerical Methods for Ordinary Differential Equations.* Chichester: J. Wiley, 2003.

[13] FOUSSE, L. et al. *MPFR: A multiple-precision binary floating-point library with correct rounding.* ACM Transactions on Mathematical Software (TOMS), v.33 n.2, Jun 2007

[14] IEEE Computer Society (August 29, 2008). *IEEE Standard for Floating-Point Arithmetic.* IEEE. doi:10.1109/IEEESTD.2008.4610935, IEEE Std 754-2008

[15] HOLOBORODKO, Pavel. *MPFR C++* [online]. [cit. 2012-05-19]. URL: <www.holoborodko.com/pavel/mpfr/>

[16] BERG, Ingo. *Vergleich numerischer Integrationsschemen am Beispiel des Magnetpendels* [online]. [cit. 2012-05-19]. URL: <*http*://beltoforion.de/pendulum_revisited/pendulum_revisited_de.html>

[17] KAHAN, Wiliam. *How Java's Floating-Point Hurts Everyone Everywhere* [online]. [cit. 2012-05-19]. URL: <http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf>

[18] SCHÖNHAGE, A., STRASSEN, V. *Schnelle Multiplikation großer Zahlen.* Computing 7, 1971.