VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

VÍCENÁSOBNÉ INTEGRÁLY

MULTIPLE INTEGRALS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **NIKOLA VALEŠOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÁCLAV ŠÁTEK, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20093/2016/xvales02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Valešová Nikola**
Obor: Informační technologie
Téma: **Vícenásobné integrály**
Multiple Integrals
Kategorie: Modelování a simulace

Pokyny:

1. Seznamte se s paralelním numerickým řešením obyčejných diferenciálních rovnic.
2. Seznamte se s problematikou analytického řešení integrálního počtu funkcí více proměnných a odvoďte metodiku numerického výpočtu určitých jednorozměrných a vícerozměrných integrálů ve víceslovní aritmetice.
3. Navrhněte a implementujte program, jehož vstupem bude zadaný integrál. Výstupem programu bude číselný výsledek společně s postupem výpočtu pro různé diferenční výrazy prostorové proměnné.
4. Zobrazte vliv řádu numerické integrační metody na přesnost a rychlost výpočtu.
5. Pomocí testovací sady určitých integrálů ověřte správnost numerického výpočtu a proveďte zhodnocení časové náročnosti.

Literatura:

- Kunovský, J.: Modern Taylor series method. Habilitation work 1994 VUT Brno
- M. Kubíček, M. Dubcová, D. Janovská: Numerické metody a algoritmy. VŠCHT Praha, 2005. ISBN
- Kalas, J., Kuben, J.: Integrální počet funkcí více proměnných, Brno 2006, Masarykova univerzita, Přírodovědecká fakulta

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šátek Václav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Problematika výpočtu určitých integrálů a diferenciálních rovnic stále tvoří významnou část několika vědeckých disciplín a řešení úloh integrálního počtu se vyskytuje také ve spoustě průmyslových odvětví. Při řešení těchto úloh se často setkáváme s požadavky na přesnost a rychlost výpočtu, které určují výběr metody vhodné pro výpočet. Cílem této práce je návrh, popis, implementace a testování nové numerické metody, jenž kombinuje řešení určitých integrálů převodem na diferenciální rovnice řešené Taylorovou řadou s tradičními numerickými metodami využívajícími Newton-Cotesovy vzorce. Výsledkem je aplikace umožňující rychlé řešení určitých dvojných integrálů, která poskytuje alespoň tak přesné výsledky jako MATLAB. Hlavním přínosem této práce je vznik nové numerické metody a její srovnání s existujícími způsoby výpočtu.

Abstract

The problem of definite integral and differential equation computation is still a significant part of many scientific branches and the solution of integral calculus tasks can be found in many industrial fields too. During the computation of such tasks, the accuracy and high-speed requirements are often confronted. These requirements are crucial during the process of the suitable method choice. The aim of this thesis is to propose, describe, implement and test a new numerical method, which combines the solution of definite integrals by transforming them into differential equations solved by the Taylor series with the traditional methods, which use the Newton-Cotes formulas. As a result, a new application has been developed, that provides fast results of definite two-dimensional integrals and reaches at least the precision of MATLAB. The major accomplishment of this thesis is the development of a new numerical method and its comparison to other established ways of computation.

Klíčová slova

určitý integrál, numerické metody, numerická integrace, Newton-Cotesovy vzorce, Taylorův polynom, víceslovní aritmetika, TKSL/C, vysoce přesné výpočty

Keywords

finite integral, numerical methods, numerical integration, Newton-Cotes formulas, Taylor polynomial, multiple-precision arithmetic, TKSL/C, high-precision computation

Citace

VALEŠOVÁ, Nikola. *Vícenásobné integrály.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Šátek Václav.

Vícenásobné integrály

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Václava Šátka, Ph.D. Další informace mi poskytl pan doc. Ing. Jiří Kunovský, CSc. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....

Nikola Valešová

16. května 2017

Poděkování

Mé poděkování patří panu Ing. Václavu Šátkovi, Ph.D., za vedení a cenné rady poskytnuté při zpracování této bakalářské práce. Dále bych na tomto místě ráda poděkovala panu doc. Ing. Jiřímu Kunovskému, CSc., a Ing. Janu Chaloupkovi za podnětné připomínky.

Obsah

Kapitola 1

Úvod

Integrální počet je významnou součástí oboru matematická analýza. Jak je uvedeno v článku [12], první doložené zmínky o zkoumání a řešení těchto problémů sahají před počátek našeho letopočtu. V průběhu minulého století se však integrální počet stal velmi významnou oblastí a i dnes je středem zájmu nejen matematiků, ale i odborníků ze všech odvětví, která mají určitý matematický základ. Numerická i analytická řešení těchto úloh tedy nejsou zkoumána pouze v rámci věd matematických, nýbrž i dalších, převážně přírodních, věd – fyziky, informatiky, chemie, statiky a dalších technických disciplín. V okamžiku, kdy je nějaký zkoumaný problém, či systém popsán s využitím diferenciálních nebo integrálních rovnic, vyvstává otázka, jak tyto řešit s dosažením co největší přesnosti výsledku. Důležité faktory ovšem představují i doba potřebná k získání řešení a složitost implementace. Kombinace důležitosti a míry splnění jednotlivých požadavků poté určuje, které ze známých metod je vhodné na tento příklad použít.

Cílem této práce je prozkoumat a představit netradiční metody numerického výpočtu určitých integrálů využívající převod integrálu na diferenciální rovnici v kombinaci se známými numerickými metodami řešení určitých integrálů využívajícími Newton-Cotesovy vzorce.

Tento dokument lze pomyslně rozdělit na dvě části – teoretickou a praktickou. Do první z nich bychom zařadili kapitoly 2 až 5, jejichž cílem je seznámit čtenáře s často používanými pojmy a uvést jej do problematiky integrálního počtu. Druhá kapitola je věnována základům dané matematické disciplíny, jsou v ní připomenuty pojmy neurčitého a určitého integrálu, jenž jsou vzápětí rozšířeny na multidimenzionální oblasti. Obsahem třetí kapitoly je poté popis metod, kterými lze určité a neurčité integrály řešit, jejich výhody a zápory. Diferenciálním počtem se zabývá kapitola čtvrtá. I ten je z hlediska této práce důležitý, jelikož je představena metoda využívající převod integrálů na diferenciální rovnice. Účelem páté kapitoly je objasnění pojmu víceslovní aritmetika, důvody pro její zavedení a s tím související také způsob uložení desetinných čísel na číslicových počítačích.

Praktická část zahrnuje kapitoly 6 až 8 a je věnována popisu implementované aplikace. Nejprve je v kapitole šest objasněn návrh aplikace a specifikace požadavků na ni. V této kapitole je čtenář také seznámen s navrhovanou numerickou metodou výpočtu určitých integrálů. Sedmá kapitola je věnována implementaci samotné, jsou popsány jednotlivé funkce, ze kterých aplikace sestává, a je detailně vysvětlen význam jednotlivých parametrů řídících výpočet. Poslední, osmá kapitola se zabývá testováním aplikace, porovnáním jejích výsledků s jinými matematickými programy, zkoumáním vlivu hodnot parametrů na velikost chyby výsledku a v neposlední řadě srovnáním přesnosti jednotlivých implementovaných metod výpočtu.

Kapitola 2

Definice základních pojmů

Než přejdeme ke složitějším numerickým metodám pro řešení integrálů funkcí více proměnných, měli bychom nejprve připomenout definice základních pojmů z oblasti integrálního počtu, které budou v této práci často používány. Nejdříve budou uvedeny definice pojmu *neurčitý integrál* a termínů s ním souvisejících, poté přejdeme k vysvětlení *určitého integrálu* a jeho významu.

Definice a věty uvedené v této kapitole jsou převzaty z literatury [4], [6] a [20].

2.1 Neurčitý integrál

Základním kamenem v teorii integrálního počtu je pojem *integrál*, který se dále dělí na určitý a neurčitý. Abychom je mohli definovat, potřebujeme nejprve zavést pojem *primitivní funkce*, která s nimi velmi úzce souvisí.

2.1.1 Primitivní funkce

Definice 2.1.1. Nechť $\mathcal{I}$ je interval v $\mathbb{R}$ a $f : \mathcal{I} \to \mathbb{R}$ funkce. Funkci F nazveme ***primitivní*** k funkci f v intervalu $\mathcal{I}$, platí-li pro každé $x \in \mathcal{I}$ vztah

$$F'(x) = f(x). \tag{2.1}$$

Věta 2.1.1. *Je-li funkce F primitivní funkcí k nějaké funkci f v intervalu $\mathcal{I}$, pak je funkce F v $\mathcal{I}$ spojitá.*

Věta 2.1.2. *Je-li funkce F primitivní k funkci f v intervalu $\mathcal{I}$, pak $\{F + c \,|\, c \in \mathbb{R}\}$ je množinou všech primitivních funkcí k funkci f.*

Věta 2.1.3. *Nechť f je spojitá funkce na intervalu $\mathcal{I}$. Potom k ní na tomto intervalu existuje primitivní funkce.*

Z výše zmíněné definice vyplývá, že primitivní funkce F k funkci f existuje pouze tehdy, je-li funkce f na intervalu $\mathcal{I}$ spojitá. A naopak, pokud existuje k funkci f na intervalu $\mathcal{I}$ primitivní funkce F, víme, že je funkce f na daném intervalu spojitá. Dále je také patrné, že ke každé spojité funkci neexistuje pouze jedna primitivní funkce, ale nekonečně mnoho takových primitivních funkcí, které se vzájemně liší o konstantu. Pokud je tedy F primitivní funkcí k funkci f na intervalu $\mathcal{I}$ a funkce P je definována jako $P = F + c, c \in \mathbb{R}$, i P je jistě primitivní funkcí k funkci f na intervalu $\mathcal{I}$.

2.1.2 Neurčitý integrál

Definice 2.1.2. Symbolem $\int f(x)\,\mathrm{d}x$ označujeme systém všech primitivních funkcí k funkci f a nazýváme jej ***neurčitý integrál*** funkce f. Potom píšeme

$$\int f(x)\,\mathrm{d}x = F(x) + c, \quad \text{případně jen} \quad \int f(x)\,\mathrm{d}x = F(x), \tag{2.2}$$

kde F je některá primitivní funkce funkce f.

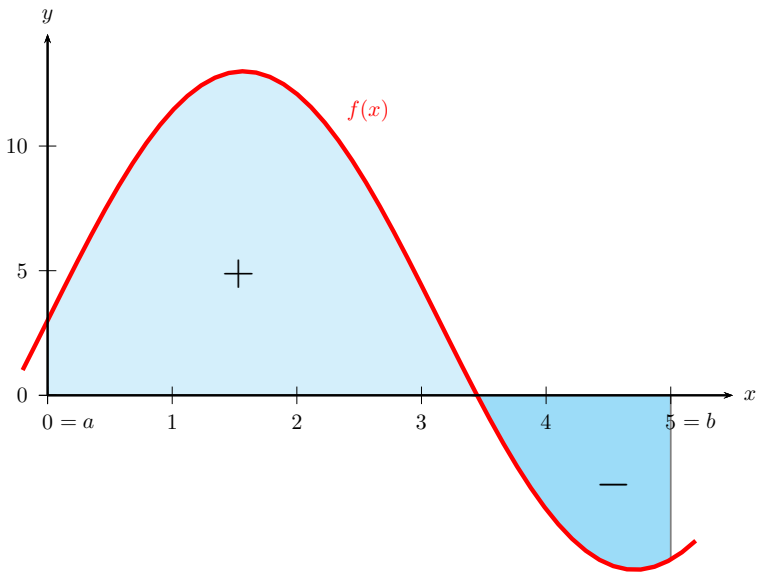
Funkce f je nazývá ***integrand*** nebo též ***integrovaná funkce***, argument x je ***integrační proměnná***.

Proces nalezení primitivní funkce k dané funkci nazýváme ***integrováním*** nebo též ***integrací***.

Neurčitý integrál funkce f je tedy množina všech k ní příslušejících primitivních funkcí F lišících se o přičtenou konstantu c.

2.2 Určitý integrál

Pokud neurčitý integrál omezíme na konkrétní interval, získáváme určitý integrál. Výsledkem určitého integrálu není funkce, jako tomu bylo u integrálu neurčitého, ale číslo, jenž udává rozdíl obsahů ploch pod křivkou funkce $f(x)$ nad a pod osou x (viz obrázek 2.1). Potřeba výpočtu určitého jednorozměrného integrálu má hlavní význam při určení obsahu plochy pod grafem nezáporné funkce na daném intervalu. Tento interval je možné rozdělit na několik podintervalů a v každém z nich původní funkci aproximovat konstantní hodnotou získanou z funkčních hodnot aproximované funkce. Obsah pod grafem tedy nahradíme součtem obsahů několika takto definovaných obdélníků. Tato představa určitého integrálu byla vytvořena Bernhardem Riemannem a nese název *Riemannův integrál*.



Obrázek 2.1: Význam jednorozměrného určitého integrálu

Definice 2.2.1. ***Dělením*** intervalu $\langle a, b\rangle$ nazýváme množinu intervalů

$$D = \{\langle x_0, x_1\rangle, \langle x_1, x_2\rangle, \ldots, \langle x_{n-1}, x_n\rangle\}, \tag{2.3}$$

kde $a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$. Body $x_0, \ldots, x_n$ poté nazýváme ***dělícími body***. Číslo $\nu(D) = \max(x_1 - x_0, x_2 - x_1, \ldots, x_n - x_{n-1})$ nazveme ***normou*** dělení D.

Čím jemnější dělení intervalu zvolíme, a tedy čím více dělících bodů použijeme, tím víc se bude vypočtený obsah blížit skutečné hodnotě integrálu. Pokud bychom zvolili $\nu(D) \to 0$, vypočtená suma se bude limitně blížit hodnotě integrálu.

2.2.1 Riemannův součet

Riemannův součet využívá Riemannovu definici integrálu. Jedná se o jeden ze základních způsobů výpočtu ohraničené plochy pod křivkou. Po rozdělení intervalu integrace na jednotlivé podintervaly určíme v každém z nich hodnotu, kterou budeme funkci aproximovat. Podle metody použité k získání této hodnoty rozlišujeme různé typy Riemannova součtu – dolní, horní, levý a pravý. Pro celkovou plochu, přes kterou integrujeme, potom platí, že je přibližně rovna součtu obsahů všech vzniklých obdélníků. Získáme tím tedy odhad hodnoty určitého integrálu.

Definice 2.2.2. Nechť je funkce $f : \langle a, b\rangle \to \mathbb{R}, a, b \in \mathbb{R}, a < b$, na intervalu $\langle a, b\rangle$ omezená shora i zdola, tzn. existují konstanty m, M takové, že pro $\forall x \in \langle a, b\rangle$ platí $m \leq f(x) \leq M$. Nechť $D = \{x_0, x_1, \ldots, x_n\}, n \in \mathbb{N}$, je libovolné dělení intervalu $\langle a, b\rangle, a, b \in \mathbb{R}, a < b$, označme pro $i = 1, 2, \ldots, n$:

$$m_i = \inf\{f(x) | x \in \langle x_{i-1}, x_i\rangle\}, \tag{2.4}$$

$$M_i = \sup\{f(x) | x \in \langle x_{i-1}, x_i\rangle\}. \tag{2.5}$$

Hodnota

$$s(f, D) = \sum_{i=1}^{n} m_i(x_i - x_{i-1}) \tag{2.6}$$

se nazývá ***dolní Riemannův součet*** funkce f při dělení D a hodnota

$$\mathcal{S}(f, D) = \sum_{i=1}^{n} M_i(x_i - x_{i-1}) \tag{2.7}$$

se nazývá ***horní Riemannův součet*** funkce f při dělení D.

Pro hodnotu určitého integrálu vždy platí, že je větší, nebo rovna hodnotě dolního Riemannova součtu a zároveň menší, nebo rovna hornímu Riemannovu součtu. Při využití výše zavedené notace tedy lze zapsat:

$$s(f, D) \leq \int_a^b f(x)\,\mathrm{d}x \leq \mathcal{S}(f, D). \tag{2.8}$$

Dalšími typy Riemannova součtu jsou tzv. levý a pravý Riemannův součet. Jak již název napovídá, výška obdélníku v každém podintervalu je dána funkční hodnotou aproximované funkce v levém, respektive pravém krajním bodě intervalu, tedy v bodě x_{i-1}, respektive x_i.

Definice 2.2.3. Nechť je funkce $f : \langle a, b\rangle \to \mathbb{R}, a, b \in \mathbb{R}, a < b$, na intervalu $\langle a, b\rangle$ omezená shora i zdola. Nechť $D = \{x_0, x_1, \ldots, x_n\}, n \in \mathbb{N}$, je libovolné dělení intervalu $\langle a, b\rangle, a, b \in \mathbb{R}, a < b$, označme pro $i = 1, 2, \ldots, n$. Hodnota

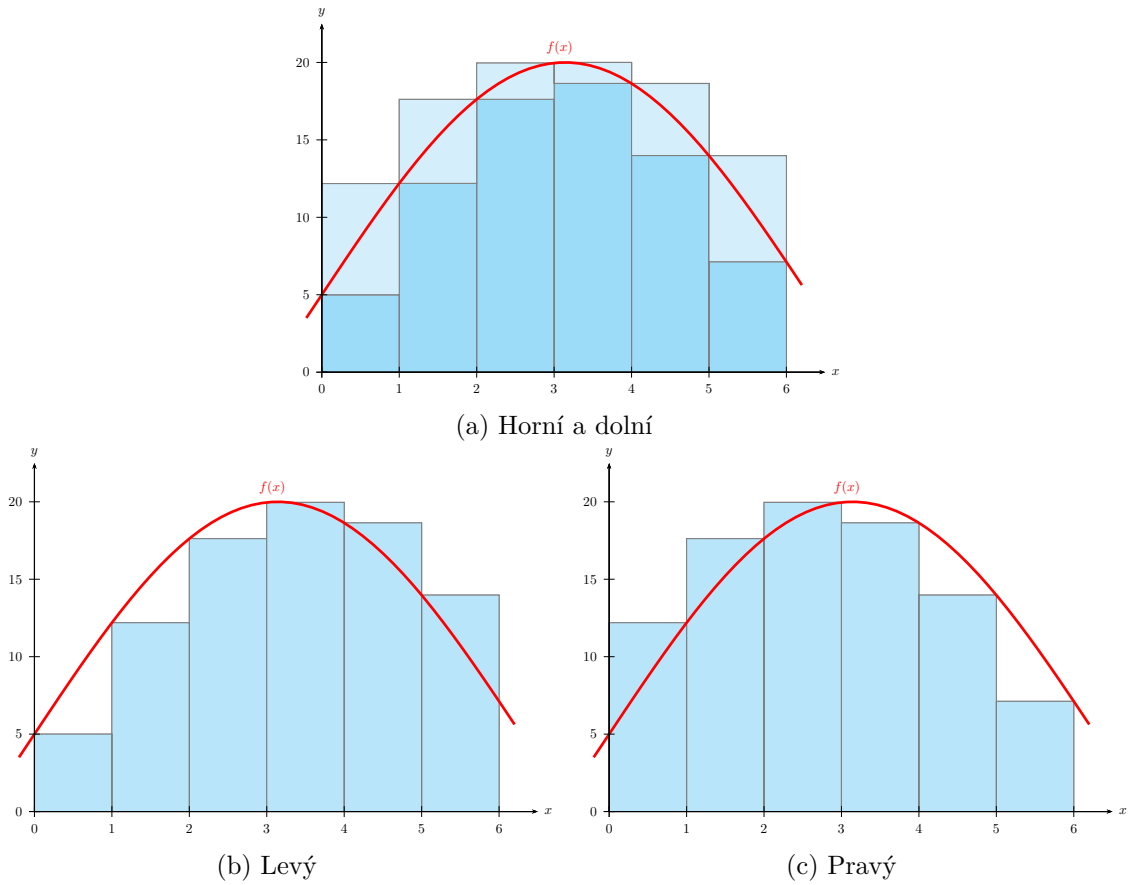
$$L(f, D) = \sum_{i=1}^{n} f(x_{i-1}) \cdot (x_i - x_{i-1}) \tag{2.9}$$

se nazývá ***levý Riemannův součet*** funkce f při dělení D a hodnota

$$R(f, D) = \sum_{i=1}^{n} f(x_i) \cdot (x_i - x_{i-1}) \tag{2.10}$$

se nazývá ***pravý Riemannův součet*** funkce f při dělení D.

Princip všech čtyř zmíněných typů Riemannových součtů je znázorněn na obrázku 2.2.



(a) Horní a dolní

(b) Levý

(c) Pravý

Obrázek 2.2: Typy Riemannových součtů

2.2.2 Riemannův integrál

Definice 2.2.4. Nechť $f : \langle a, b\rangle \to \mathbb{R}$ je ohraničená funkce. Řekneme, že f je ***integrovatelná*** na intervalu $\langle a, b\rangle$, existuje-li číslo $I \in \mathbb{R}$ takové, že ke každému $\varepsilon > 0$ existuje $\delta > 0$ tak, že pro každé dělení D intervalu $\langle a, b\rangle$, jehož norma $\nu(D) < \delta$, platí

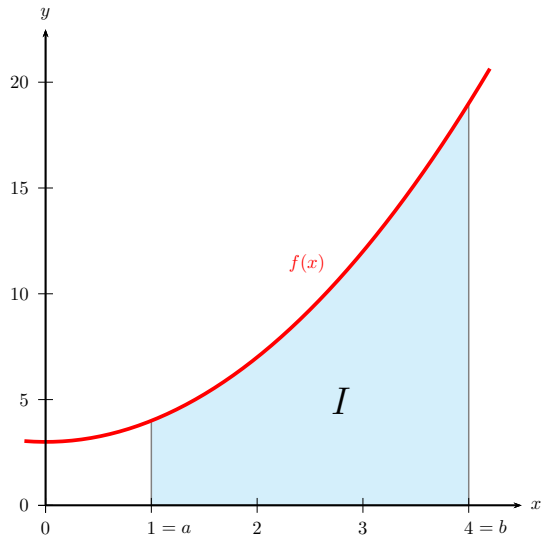
$$|\mathcal{S}(f, D) - I| < \varepsilon. \tag{2.11}$$

Číslo I nazýváme ***určitým (Riemannovým) integrálem*** funkce f od a do b a píšeme

$$I = \int_a^b f(x)\,\mathrm{d}x. \tag{2.12}$$

Krajní hodnoty intervalu, na kterém integrujeme, se nazývají ***meze***. Pro integrál 2.12 nazýváme a ***dolní*** a b ***horní*** mez.

Určitý integrál jako obsah plochy pod křivkou funkce $f(x)$ na intervalu $\langle a, b\rangle$ je znázorněn na obrázku 2.3.



Obrázek 2.3: Určitý (Riemannův) integrál

2.2.3 Newton-Leibnizova věta

Další metoda výpočtu hodnoty určitého integrálu dané spojité funkce na intervalu je pojmenovaná po Isaacu Newtonovi a Gottfriedu Leibnizovi *Newton-Leibnizova věta*. Ta definuje určitý integrál jako rozdíl hodnot primitivní funkce v mezních bodech integrace. Tímto způsobem nezískáváme pouze přibližnou hodnotu výsledku, nedochází k žádné aproximaci a přesnost řešení tedy závisí pouze na zaokrouhlování.

Věta 2.2.1. *Nechť f je funkce spojitá v $\langle a, b\rangle$. Jestliže v $\langle a, b\rangle$ platí $F'(x) = f(x)$, tj. $\int f(x)\,\mathrm{d}x = F(x) + c$, potom*

$$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a) = \Big[F(x)\Big]_a^b. \tag{2.13}$$

2.3 Vícenásobné integrály

V následujících odstavcích budou znalosti o jednorozměrném integrálu rozšířeny na vícerozměrné oblasti, budou tedy vysvětleny pojmy vedoucí k zavedení definice vícenásobných integrálů a jejich vlastností.

Při tvorbě definic v tomto oddílu byly využity zdroje [1], [5] a [11].

Definice 2.3.1. Nechť $M \subset \mathbb{R}^n$ je uzavřená ohraničená množina. Číslo

$$d(M) = \max\{|X - Y| \mid X, Y \in M\} \tag{2.14}$$

se nazývá ***průměr množiny M***.

U jednorozměrných integrálů se vyskytuje pojem norma dělení, který udává maximální délku dělícího intervalu. U vícenásobných integrálů hovoříme nikoliv o normě, ale o průměru množiny. Ten vyjadřuje největší možnou vzdálenost mezi libovolnými dvěma body množiny.

Definice 2.3.2. Nechť $\Omega \in \mathbb{R}^k$ je k-rozměrný interval; tj. pro $k = 2$ obdélník, pro $k = 3$ kvádr, $f : \Omega \to \mathbb{R}$ je ohraničená funkce.

- Systém intervalů

$$\{\Omega_1, \Omega_2, \ldots, \Omega_n\} \tag{2.15}$$

se nazývá ***dělení intervalu*** Ω, jestliže platí

$$\Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n = \Omega \quad \text{a současně} \quad |\Omega_i \cap \Omega_j| = 0 \quad \forall i, j = 1, \ldots, n. \tag{2.16}$$

- ***Normou dělení*** $D(I) = \{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ rozumíme číslo

$$\nu(D) = \max\{d(\Omega_i), i = 1, \ldots, n\}. \tag{2.17}$$

- ***Integrální součet*** příslušný funkci f a dělení $D(\Omega)$ s vybranými body je číslo

$$\mathcal{S}(f, D) = \sum_{i=1}^{n} f(\xi_i) m(\Omega_i), \quad \text{kde} \quad \xi_i \in \Omega_i. \tag{2.18}$$

- Řekneme, že číslo $\mathcal{J}$ je ***dvojným (trojným) integrálem*** funkce $f : \Omega \to \mathbb{R}$ na intervalu Ω a píšeme

$$\mathcal{J} = \int_\Omega f(x, y)\,\mathrm{d}x\,\mathrm{d}y \qquad \left(\mathcal{J} = \int_\Omega f(x, y, z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z\right), \tag{2.19}$$

jestliže pro každé $\epsilon > 0$ lze najít takové $\delta > 0$ a $n \in \mathbb{N}$, že pro všechna dělení $D(\Omega) = \{\Omega_1, \ldots, \Omega_n\}$, pro které je $\nu(D) < \delta$ nezávisle na volbě bodů $\xi_i \in \Omega_i$ $(i = 1, \ldots, n)$ platí

$$|\mathcal{J} - \mathcal{S}(f, D)| < \epsilon. \tag{2.20}$$

Vyhovuje-li některá funkce f a číslo $\mathcal{J}$ předchozí definici, říkáme, že integrál

$$\mathcal{J} = \int_\Omega f(x, y)\,\mathrm{d}x\,\mathrm{d}y \qquad \left(\mathcal{J} = \int_\Omega f(x, y, z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z\right) \tag{2.21}$$

existuje a že funkce f je na Ω ***integrovatelná***.

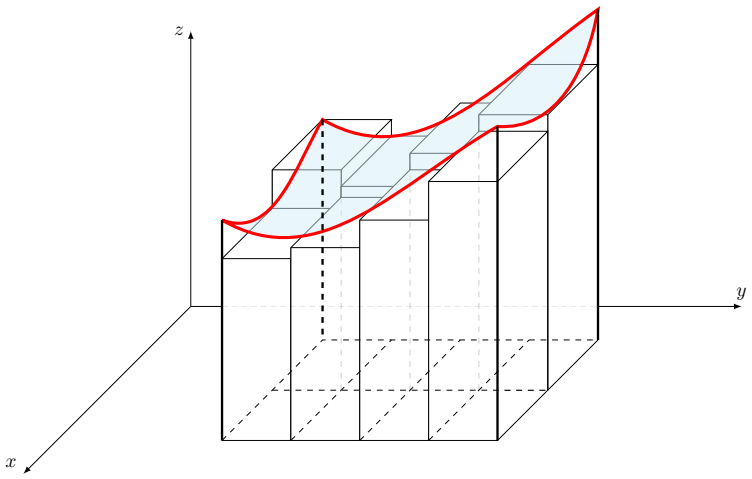
Věta 2.3.1 (Fubiniova věta)**.** *Nechť* $\Omega = \langle a, b\rangle \times \langle c, d\rangle \times \ldots \times \langle r, s\rangle$*, kde* $x_1 \in \langle a, b\rangle, x_2 \in \langle c, d\rangle, \ldots, x_n \in \langle r, s\rangle$*. Je-li* $f : \Omega \to \mathbb{R}$ *integrovatelná na* Ω*, pak existují integrály*

$$\begin{aligned}\int_\Omega & f(x_1, \ldots, x_n)\,\mathrm{d}x_1 \ldots \mathrm{d}x_n = \\ &= \int_{a_1}^{b_1} \left(\int_{a_2}^{b_2} \left(\cdots \left(\int_{a_n}^{b_n} f(x_1, \ldots, x_n)\,\mathrm{d}x_n \right) \cdots \right) \mathrm{d}x_2 \right) \mathrm{d}x_1 = \\ &= \int_{a_{i_1}}^{b_{i_1}} \left(\int_{a_{i_2}}^{b_{i_2}} \left(\cdots \left(\int_{a_{i_n}}^{b_{i_n}} f(x_1, \ldots, x_n)\,\mathrm{d}x_{i_n} \right) \cdots \right) \mathrm{d}x_{i_2} \right) \mathrm{d}x_{i_1}.\end{aligned} \tag{2.22}$$

pro každou permutaci $(i_1, i_2, \ldots, i_n)$ *množiny indexů* $\{1, 2, \ldots, n\}$*.*

Fubiniova věta říká, že obecně n-rozměrný určitý integrál lze vypočítat pomocí n určitých integrálů, a to tak, že postupně integrujeme vždy podle jedné proměnné, přičemž na pořadí proměnných volených pro integraci nezáleží.

Stejný postup, jakým Riemannův součet aproximuje určité jednorozměrné integrály, lze aplikovat i na integrály vícenásobné; například dvojný integrál můžeme nahradit součtem objemů kvádrů (viz obrázek 2.4). I zde platí, že čím jemnější dělení intervalů zvolíme, tím větší přesnosti jsme schopni dosáhnout.



Obrázek 2.4: Riemannův součet pro dvojný integrál

Definice 2.3.3. Řekneme, že ohraničená funkce f je Riemannovsky integrovatelná na A a číslo $a \in \mathbb{R}$ nazveme **n-rozměrný Riemannův integrál funkce f na množině** A, když pro každou konečnou posloupnost $D(k)$ dělení intervalu A a pro každou volbu reprezentantů v těchto děleních platí

$$\lim_{k\to\infty} S(f, D(k)) = a. \tag{2.23}$$

Kapitola 3

Metody výpočtu integrálů

V této kapitole budou představeny dvě hlavní třídy metod, které je možno využít při výpočtu integrálů, a to *analytickou* a *numerickou integraci*.

Nejprve si ukážeme analytické metody řešení integrálů. Jejich výhodou je vysoká přesnost a rychlý výpočet při dosazení jiných hodnot parametrů. Analyticky však nejsme schopni řešit rozsáhlé systémy a složité matematické vztahy, a proto se v praxi příliš nepoužívá, případně je třeba daný popis zjednodušit a některé aspekty abstrahovat.

Poté se přesuneme ke druhé možnosti, kterou je použití některé ze skupiny metod numerických. Tyto poskytují pouze přibližné řešení, jejich výhodou je však univerzální využití a relativně snadná implementace.

Při tvorbě definic a vět uvedených v této kapitole bylo čerpáno z literatury [3], [6], [7] a [20].

3.1 Analytická integrace

Analytické metody představují základní přístup, který je možno použít pro řešení integrálů. Umožňují nám dosáhnout velmi přesných výsledků, jelikož poskytují teoretický návod, soubor pravidel, která je možné při řešení integrálů využívat. Jejich výhodou je již zmíněná přesnost a u určitých integrálů také možnost velmi rychlého a snadného získání výsledku integrálu nad jiným intervalem, tedy při změně mezí. Tyto metody však není možné aplikovat na integrál libovolné funkce. Pro složitější výrazy nenalezneme žádné aplikovatelné pravidlo, nebo je třeba využít velmi složitých úprav a výpočet tak trvá poměrně dlouho.

Úpravy nejjednodušších výrazů můžeme nalézt v tabulce základních integrálů[1]. Pomocí tabulky jsme schopni nalézt primitivní funkci k jednoduchým výrazům, dále také pro součet a rozdíl takových výrazů, jelikož integrál součtu či rozdílu můžeme zapsat jako součet, respektive rozdíl integrálů. Pokud však máme za úkol zintegrovat součin nebo podíl, už si s tabulkou integrálů nevystačíme a musíme využít jiné dostupné metody. Proto zde nyní budou popsány dva nejznámější a v praxi nejčastěji používané analytické postupy – *metoda per partes* a *metoda substituce*.

3.1.1 Metoda per partes

Integrace metodou per partes (v překladu „po částech“) se používá v případě, že chceme integrovat součin dvou funkcí. Princip této metody je založen na vztahu pro derivaci součinu

[1] Tabulka integrálů je dostupná například na stránce: https://math.feld.cvut.cz/sedlacko/vz2u.pdf.

$(uv)' = u'v + uv'$, ze kterého můžeme vyjádřit člen uv' jako $uv' = (uv)' - u'v$. Pokud nyní zintegrujeme obě strany, získáme vztah pro řešení integrálu metodou per partes

$$\int u(x)v'(x)\,\mathrm{d}x = u(x)v(x) - \int u'(x)v(x)\,\mathrm{d}x. \tag{3.1}$$

Integrál součinu je v tomto vztahu vyjádřen opět pomocí integrálu součinu, za členy u a v je tedy třeba vhodně volit substituce tak, abychom si celý výraz po použití této metody zjednodušili. Tuto úpravu je možné používat opakovaně za předpokladu, že je metoda aplikovatelná a po jejím využití dojde ke zjednodušení výrazu.

Ze vztahu (3.1) pro řešení neurčitých integrálů lze přidáním mezí odvodit vztah metody per partes pro vyhodnocení určitých integrálů

$$\int_a^b u(x)v'(x)\,\mathrm{d}x = \Big[u(x)v(x)\Big]_a^b - \int_a^b u'(x)v(x)\,\mathrm{d}x. \tag{3.2}$$

3.1.2 Metoda substituce

Pokud k funkci f na intervalu $\mathcal{I}$ přísluší primitivní funkce F, můžeme integrál funkce f zapsat následovně:

$$\int f(t)\,\mathrm{d}t = \int F'(t)\,\mathrm{d}t = \int \mathrm{d}F(t), \tag{3.3}$$

kde $\mathrm{d}F(t)$ je diferenciál primitivní funkce F.

Pokud použijeme větu o derivaci složené funkce $(F(g(x)))' = F'(g(x))g'(x)$ a položíme $t = g(x)$, dostáváme pro diferenciál primitivní funkce $\mathrm{d}F(t)$

$$\mathrm{d}F(t) = \mathrm{d}F(g(x)) = F'(g(x))g'(x)\,\mathrm{d}x = f(g(x))g'(x)\,\mathrm{d}x, \tag{3.4}$$

z čehož můžeme odvodit vztah pro metodu substituce

$$\int f(t)\,\mathrm{d}t = \int f(g(x))g'(x)\,\mathrm{d}x, \quad \text{kde} \quad t = g(x). \tag{3.5}$$

Tato metoda patří mezi nejdůležitější metody pro integraci. Je užitečná například při výpočtu integrálu podílu se složitější funkcí ve jmenovateli.

Pro výpočet určitých integrálů je postup opět obdobný jako pro neurčité integrály. Oproti předchozímu postupu přibývá potřeba vypočítat nové meze, díky čemuž však odpadá nutnost zpětného dosazování substituční funkce, které se provádělo na závěr výpočtu.

Věta 3.1.1. *Jestliže funkce $f \circ g$, g' jsou spojité na intervalu $\langle a, b\rangle$, potom*

$$\int_a^b f[g(x)]g'(x)\,\mathrm{d}x = \int_{g(a)}^{g(b)} f(t)\,\mathrm{d}t. \tag{3.6}$$

Jestliže f je spojitá na $\langle a, b\rangle$ a $x = g(t)$ je monotónní funkce se spojitou derivací a oborem hodnot $\langle a, b\rangle$, potom

$$\int_a^b f(x)\,\mathrm{d}x = \int_{g^{-1}(a)}^{g^{-1}(b)} f[g(t)]g'(t)\,\mathrm{d}t. \tag{3.7}$$

3.2 Numerická integrace

Metody numerické integrace představují postupy, jakými lze získat přibližnou hodnotu určitého integrálu. Nevýhodu, jenž spočívá v nepřesnosti výsledku, vyvažují důvody pro využití numerické integrace, kterými jsou

- k integrované funkci neexistuje primitivní funkce,
- integrál není možné spočítat analytickými metodami,
- analytický výpočet je příliš složitý a časově náročný,
- integrovaná funkce je dána pouze tabulkou hodnot v konkrétních bodech.

Numerické metody vychází z myšlenky rozdělení plochy pod integrovanou funkcí na několik menších částí, jejichž plochu jsme schopni vypočítat. Po sečtení obsahů těchto dílčích ploch získáme přibližně plochu pod integrovanou funkcí, tedy hodnotu určitého integrálu na daném intervalu. Podle způsobu vytváření jednotlivých ploch rozlišujeme metody *obdélníkovou*, *lichoběžníkovou* a *Simpsonovu*.

3.2.1 Metody numerické integrace

Následující numerické metody patří mezi Newton-Cotesovy vzorce[2] a používají se především pro řešení jednorozměrných integrálů, jejich modifikace lze však použít i při výpočtu vícenásobných integrálů.

Obdélníková metoda

Asi nejjednodušší metodou je metoda obdélníková. Při aplikaci této metody je interval integrace $\langle a, b\rangle$ rozdělen na n stejně velkých podintervalů $\langle x_i, x_{i+1}\rangle$ a na každém z nich je funkce f aproximována konstantní funkcí, jejíž hodnota je rovna $f\left(\frac{x_i+x_{i+1}}{2}\right)$, tedy funkční hodnotě aproximované funkce v bodě uprostřed intervalu. Plochu pod křivkou se tedy v tomto případě snažíme nahradit obsahy obdélníků, podobně jako u Riemannova součtu. Pro každý podinterval tedy platí vztah pro jednoduchou obdélníkovou metodu

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \approx f\left(\frac{x_i + x_{i+1}}{2}\right) \cdot (x_{i+1} - x_i), \tag{3.8}$$

potom pro výpočet určitého integrálu na celém intervalu $\langle a, b\rangle$ slouží složená obdélníková metoda, kterou můžeme zapsat pomocí následujícího vztahu:

$$\int_a^b f(x)\,\mathrm{d}x \approx h \cdot \sum_{i=0}^{n-1} f\left(\frac{x_i + x_{i+1}}{2}\right), \tag{3.9}$$

kde $h = (x_{i+1} - x_i) = \frac{b-a}{n}$.

Přesnost této metody je závislá na šířce podintervalů a velikosti intervalu integrace – čím jemnější dělení zvolíme, tím více se aproximace přibližuje skutečné hodnotě integrálu, a čím

[2] Více o Newton-Cotesových vzorcích lze nalézt například na adrese https://archive.lib.msu.edu/crcmath/math/math/n/n080.htm.

užší interval pro integraci zvolíme, tím menší bude globální akumulovaná chyba. Pro určení velikost chyby platí nerovnost

$$E \leq \frac{(b-a)h^2}{24} f''(\xi), \tag{3.10}$$

kde $\xi \in (a, b)$.

Lichoběžníková metoda

Druhou metodou je lichoběžníková metoda. Její princip je podobný předchozí metodě, avšak aproximovaná funkce je na každém z podintervalů $\langle x_i, x_{i+1} \rangle$ nahrazena lineární funkcí a určitý integrál je tak aproximován obsahy lichoběžníků. Pro hodnotu určitého integrálu přes jednotlivé podintervaly platí vztah pro jednoduchou lichoběžníkovou metodu

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \approx \Big(f\left(x_i\right) + f\left(x_{i+1}\right) \Big) \cdot \frac{x_{i+1} - x_i}{2}. \tag{3.11}$$

Pro výpočet obsahu plochy pod křivkou na celém intervalu $\langle a, b \rangle$ slouží složená lichoběžníková metoda, jenž lze popsat vztahem:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{2} \sum_{i=0}^{n-1} \Big(f\left(x_i\right) + f\left(x_{i+1}\right) \Big), \tag{3.12}$$

kde $h = (x_{i+1} - x_i) = \frac{b-a}{n}$.

Pro přesnost této metody platí stejná pravidla jako u obdélníkové metody. Horní hranice odhadované chyby je u této metody větší než u předchozí, její teoretická přesnost je tedy menší, ale není to pravidlem. Velikost chyby je daná nerovností

$$E \leq \frac{(b-a)h^2}{12} f''(\xi), \tag{3.13}$$

kde $\xi \in (a, b)$.

Simpsonova metoda

Simpsonova metoda se od předchozí metody liší typem polynomu, kterým aproximuje integrovanou funkci $f(x)$. Tato metoda využívá polynomy druhého stupně, aproximovanou funkci tedy na jednotlivých podintervalech nahrazujeme částí paraboly. Další odlišností Simpsonovy metody je počet uzlů, které vystupují ve funkčním předpisu metody. Kromě funkčních hodnot v krajních bodech intervalu $\langle a, b \rangle$ využívá tato metoda i střed tohoto intervalu $\frac{a+b}{2}$ a funkční hodnotu v tomto bodě $f\left(\frac{a+b}{2}\right)$. Pro každý dílčí interval lze využít vztah pro jednoduchou Simpsonovu metodu

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \approx \frac{x_{i+1} - x_i}{6} \left(f(x_i) + 4f\Big(\frac{x_i + x_{i+1}}{2}\Big) + f(x_{i+1}) \right). \tag{3.14}$$

Pro aplikaci složené Simpsonovy metody musíme interval $\langle a, b \rangle$ rozdělit na sudý počet n dílků. Poté můžeme integrál spočítat s využitím vzorce

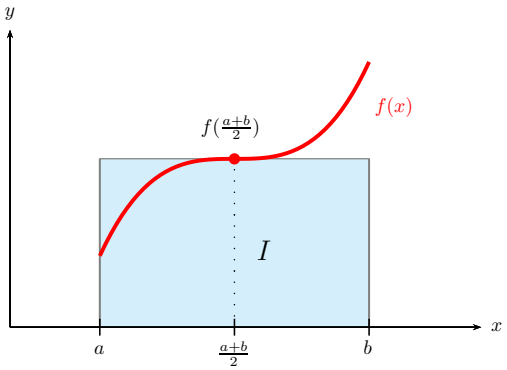
$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3} \Big(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{n-1}) + f(x_n) \Big), \tag{3.15}$$

kde $h = \frac{b-a}{n}$ a $x_i = a + h \cdot i$ pro $\forall i \ \in \{0, 1, \ldots, n-1, n\}$.

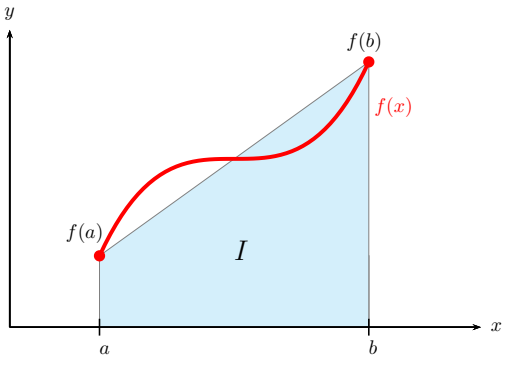
Chyba Simpsonovy metody je přímo úměrná velikosti intervalu $\langle a, b\rangle$ a nepřímo úměrná počtu podintervalů n. Ze všech uvedených metod má nejvyšší přesnost, nejlépe aproximuje integrovanou funkci. Chyba metody je dána nerovností

$$E \leq \frac{h^4}{180}(b-a)f^{(4)}(\xi), \quad a \ \leq \xi \leq b. \tag{3.16}$$

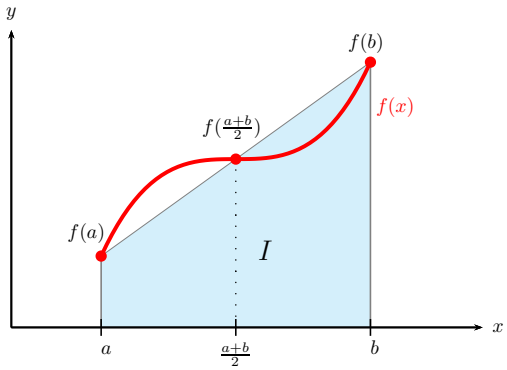
Způsoby konstrukce a rozdíly všech výše zmíněných numerických metod jsou znázorněny na obrázku 3.1. Výsledek Simpsonovy metody se v tomto případě neliší od řešení metodou lichoběžníkovou, jelikož aplikací Simpsonovy metody na funkci $f(x)$ vznikla lineární funkce, tedy polynom 1. stupně.



(a) Obdélníková metoda



(b) Lichoběžníková metoda



(c) Simpsonova metoda

Obrázek 3.1: Numerické integrační metody

Monte Carlo integrování

Metoda *Monte Carlo* nepopisuje jeden konkrétní postup, ale jedná se o celou třídu algoritmů, které využívají generátory náhodných čísel. Zatímco jiné metody obvykle vyhodnocují integrand v ekvidistantně rozložených bodech, metoda Monte Carlo tyto body určuje náhodně. Využití nachází především při řešení N-rozměrných integrálů pro velká N, kdy oproti jiným metodám vyniká v rychlém výpočtu a snadné implementaci. Pro vypočtení jednoduchých integrálů není její použití příliš vhodné, jelikož jiné metody dosahují vyšší přesnosti.

Základní myšlenka metody Monte Carlo spočívá v provádění experimentů a určení střední hodnoty veličiny, která je jejich výsledkem. Pro vypočtení integrálu funkce $f(x)$ na intervalu $\langle a, b\rangle$

$$I = \int_a^b f(x)\,\mathrm{d}x \tag{3.17}$$

vygenerujeme N náhodných bodů x_i s rovnoměrným rozložením v rozsahu $\langle a, b)$ a vypočteme průměr funkčních hodnot v získaných bodech

$$C = \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{3.18}$$

Výsledek určíme jako násobek průměrné funkční hodnoty a šířky intervalu, přes který integrujeme

$$I = \int_a^b f(x)\,\mathrm{d}x \approx C \cdot (b - a). \tag{3.19}$$

Získáme tak určitý integrál, obsah plochy pod křivkou $f(x)$ na intervalu $\langle a, b\rangle$, který aproximujeme obdélníkem o délce základny stejné jako šířka intervalu integrace a výškou danou průměrnou funkční hodnotou původní funkce na daném intervalu. Tento postup lze modifikovat pro výpočet multidimenzionálních integrálů.

Velikost chyby při použití metody Monte Carlo je závislá na počtu provedených experimentů. Pro odhad relativní chyby platí:

$$E = \frac{1}{\sqrt{N}}. \tag{3.20}$$

Ze vztahu je patrné, že pro zdvojnásobení přesnosti výpočtu je potřeba použít čtyřikrát více náhodně vygenerovaných bodů. Pro výpočty náročné na přesnost tato metoda tedy není ideální. Chyba metody však nezávisí na počtu dimenzí integrálu, což potvrzuje její vhodné použití pro vícerozměrné integrály.

Kapitola 4

Teorie diferenciálního počtu

K integrálnímu počtu neodmyslitelně patří počet diferenciální. Společně tyto dvě disciplíny tvoří tzv. infinitezimální počet, jehož předmětem je počítání mezních hodnot, kterým se blíží funkce různých proměnných. V této kapitole budou popsány základní pojmy oboru diferenciálního počtu, z nichž nejdůležitějším je *diferenciální rovnice*. Dále budou představeny její typy a na závěr budou vysvětleny možnosti jejího řešení.

Definice a vztahy použité v této kapitole jsou převzaty ze zdrojů [3], [9], [10], [11] a [14].

4.1 Základní pojmy diferenciálního počtu

Diferenciální rovnice jsou rovnice, ve kterých jako proměnné vystupují funkce a ve kterých se vyskytují i derivace těchto funkcí. *Obyčejné diferenciální rovnice* obsahují derivace hledané funkce podle jediné proměnné. *Parciální diferenciální rovnice* obsahují derivace hledané funkce podle dvou nebo více proměnných.

Definice 4.1.1. ***Obyčejnou diferenciální rovnicí*** n***-tého řádu*** nazýváme rovnici, v níž se vyskytuje neznámá funkce jedné proměnné a její derivace až do řádu n. Zapisujeme ji obecně ve tvaru

$$F(x, y, y', \ldots, y^{(n)}) = 0, \tag{4.1}$$

kde F je funkce $n+2$ proměnných. ***Řád*** diferenciální rovnice je řád nejvyšší derivace, která se v dané diferenciální rovnici vyskytuje.

Definice 4.1.2. ***Řešením*** obyčejné diferenciální rovnice řádu n na intervalu $\mathcal{I}$ nazýváme každou n-krát diferencovatelnou funkci na intervalu $\mathcal{I}$, která vyhovuje dané rovnici. ***Obecným řešením*** obyčejné diferenciální rovnice rozumíme obecný předpis závisející na n různých parametrech $[c_1, c_2, \ldots, c_n] \in M \subseteq \mathbb{R}^n$, kde libovolnou volbou těchto parametrů dostaneme konkrétní řešení, které nazýváme ***partikulárním (částečným) řešením***.

Definice 4.1.3. Úlohu najít řešení $y(x)$ diferenciální rovnice (4.1) definované na intervalu $\mathcal{I}$ a splňující podmínky

$$y(x_0) = y_0, \quad y'(x_0) = y_1, \quad \ldots, \quad y^{n-1}(x_0) = y_{n-1}, \quad x_0 \in \mathcal{I} \tag{4.2}$$

nazýváme ***počáteční (Cauchyho) úlohou*** nebo ***počátečním (Cauchyho) problémem***. Podmínky (4.2) se nazývají ***počáteční podmínky***.

Definice 4.1.4. Nechť $\Omega \subseteq \mathbb{R}^2$ a $f : \Omega \to \mathbb{R}^2$ je funkce dvou proměnných. Potom ***diferenciální rovnicí prvního řádu*** nazýváme rovnici

$$y' = f(x, y). \tag{4.3}$$

Věta 4.1.1 (Existenční)**.** *Nechť $f(x, y)$ je spojitá na otevřené množině $M \subseteq \mathbb{R}^2$. Pak pro každé $[x_0, y_0] \in M$ má úloha*

$$y' = f(x, y), \quad y(x_0) = y_0 \tag{4.4}$$

alespoň jedno řešení definované na nějakém otevřeném intervalu $J \subseteq (a, b), x_0 \in J$.

Věta 4.1.2 (O existenci a jednoznačnosti řešení)**.** *Nechť $f(x, y)$ je spojitá na otevřené množině $M \subseteq \mathbb{R}^2$ a v každém bodě množiny M je splněna Lipschitzova podmínka, tj. existuje $L > 0$ tak, že platí*

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2| \tag{4.5}$$

pro každé dva body $[x, y_1]$, $[x, y_2]$ z nějakého okolí bodu $[x_0, y_0]$. Pak pro libovolný bod $[x_0, y_0] \in M$ má úloha (4.4) právě jedno řešení.

Převod integrálu na diferenciální rovnici

Výpočet určitého integrálu

$$F(x) = \int_a^b f(x)\,\mathrm{d}x \tag{4.6}$$

je možné převést na řešení obyčejné diferenciální rovnice

$$F'(x) = f(x) \tag{4.7}$$

zderivováním obou stran rovnice, čímž odstraníme integrál, a přidáním počáteční podmínky

$$F(a) = 0. \tag{4.8}$$

Výše zmíněný princip předpokládá dolní mez integrálu a rovnou 0 a zároveň $a < b$. Pokud platí $a > b$, aplikací vztahu

$$\int_n^m f(x)\,\mathrm{d}x = -\int_m^n f(x)\,\mathrm{d}x \tag{4.9}$$

jsme schopni původní integrál převést na takový, pro který platí, že dolní mez integrace je menší než mez horní. Pokud je nyní dolní mez integrace různá od 0, je třeba navíc provést v integrované funkci $f(x)$ substituci x za $x-k$ a přepočet mezí přičtením čísla k k hodnotám a i b. Pro $a < b$ vypadá vztah pro posun integrálu v čase následovně:

$$\int_a^b f(x)\,\mathrm{d}x = \int_{a+k}^{b+k} f(x-k)\,\mathrm{d}x, \tag{4.10}$$

přičemž musí platit, že $a + k = 0$.

4.2 Metody řešení diferenciálních rovnic

Metody používané k řešení diferenciálních rovnic se dělí na dvě skupiny – metody analytické a numerické. Stejně jako u analytických metod pro výpočet integrálů, jsou i analytické metody pro řešení obyčejných diferenciálních rovnic časově náročné a nelze je aplikovat na systémy popsané složitými rovnicemi. Metody numerické nám opět poskytují pouze přibližné výsledky, avšak tyto postupy umožňují rychlejší řešení, a to i pro komplexnější systémy. Dále se budeme zabývat jenom tímto typem metod, jelikož analytické metody nejsou z hlediska dalšího výkladu v této práci významné.

4.2.1 Numerické metody

Numerické metody lze rozdělit na

- jednokrokové,
- vícekrokové,

podle toho, z kolika předchozích bodů využívají informací k výpočtu hodnoty v aktuálním bodě. Jednokrokové metody používají informace z jediného předcházejícího bodu a patří mezi ně například Eulerova metoda a metody Runge-Kutta. Metody, které pro výpočet aktuální hodnoty potřebují informace z více předcházejících bodů, se nazývají vícekrokové a jedná se například o Adams-Bashforthovu metodu nebo metodu prediktor-korektor. U jednokrokových metod nám pro zahájení výpočtu stačí znát hodnotu počáteční podmínky, avšak u startu metod vícekrokových musíme řešit problém, kdy nemáme informace o minulých stavech.

Jednokrokové metody

Eulerova metoda Eulerova metoda je nejjednodušší jednokrokovou metodou. Pro získání výsledku používá pouze derivaci v předcházejícím bodě. Při využití této metody lze hodnotu v aktuálním bodě spočítat pomocí vztahu

$$y_{i+1} = y_i + hf(x_i, y_i) \quad \forall i \ \in \{0, 1, \ldots, n-1\}, \tag{4.11}$$

kde $f(x_i, y_i)$ je hodnota derivace v předchozím bodě (viz rovnice (4.3)). Pro první krok metody použijeme informace z počáteční podmínky, hodnota řešení v bodě x_0 je rovna y_0.

Existují různé modifikace Eulerovy metody, které pracují na stejném principu, avšak během výpočtu používají další body v intervalu $\langle x_i, x_{i+1}\rangle$. Mezi nejznámější modifikace Eulerovy metody patří první a druhá modifikace, které jsou zároveň jednoduchými příklady Runge-Kutta metod.

Horní hranici celkové chyby při použití této metody lze podle zdroje [19] určit s využitím nerovnosti

$$|E_n| \leq \frac{Mh}{2K}(e^{Kt_n} - 1), \tag{4.12}$$

kde

$$K = \max_{(t,y)\in\mathbb{R}} |f_y(t,y)| < \infty, \quad M = \max_{(t,y)\in\mathbb{R}} |(f_t + f \cdot f_y)(t,y)| < \infty, \tag{4.13}$$

kde $f_t = \frac{\mathrm{d}f}{\mathrm{d}t}$ a $f_y = \frac{\mathrm{d}f}{\mathrm{d}y}$.

Runge-Kutta metody Runge-Kutta metody tvoří nejdůležitější skupinu jednokrokových metod. Jejich obecný tvar je

$$y_{i+1} = y_i + h\left(w_1 k_1 + \cdots + w_s k_s\right), \tag{4.14}$$

pro $i = 0, 1, \ldots, n-1$, kde

$$k_1 = f(x_i, y_i) \tag{4.15}$$

$$k_r = f(x_i + \alpha_r h, y_i + h\sum_{j=1}^{r-1} \beta_{rj} k_j), \quad r = 2, 3, \ldots, s \tag{4.16}$$

a w_r, α_r a β_{rj} jsou konstanty volené tak, aby metoda měla maximální řád.

Nejznámější a nejčastěji používanou je metoda Runge-Kutta 4. řádu, která díky vyššímu počtu pomocných koeficientů poskytuje přesnější hodnotu následujícího bodu. Její výpočet je dán následující soustavou rovnic:

$$\begin{aligned} y_{i+1} &= y_i + \tfrac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_1) \\ k_3 &= f(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}hk_2) \\ k_4 &= f(x_i + h, y_i + hk_3). \end{aligned} \tag{4.17}$$

Vícekrokové metody

Vícekrokové metody využívají k vypočtení hodnoty v dalším bodě znalosti několika bodů předcházejících. Obecně nelze říci, že jsou vícekrokové metody efektivnější než jednokrokové, avšak především u ne příliš dynamických systémů bývá odhad nové hodnoty přesnější. Obecný postup získání řešení v následujícím bodě popisuje tzv. *lineární k-kroková metoda*

$$y_{i+1} = a_1 y_i + a_2 y_{i-1} + \cdots + a_k y_{i-k+1} + h(b_0 f_{i+1} + b_1 f_i + \cdots + b_k f_{i-k+1}), \tag{4.18}$$

kde $k \in \mathbb{N}$ a a_j a b_j jsou vhodné konstanty, z nichž alespoň jedna je nenulová.

Je-li $b_0 \neq 0$, je nutno vztah (4.18) řešit iteračně, jelikož je řešení v dalším bodě závislé na neznámé hodnotě funkce $f_{n+1} = f(x_{n+1}, y_{n+1})$. Metoda se v takovém případě nazývá *implicitní*. Pokud je $b_0 = 0$, metodu označíme jako *explicitní*. Výpočet je v tomto případě jednodušší, provádí se pouze dosazením do vzorce (4.18), explicitní metody jsou však méně přesné a ne tak stabilní jako implicitní metody.

Pro výpočet prvních k kroků, kdy nemáme k dispozici historii informací o dostatečném počtu bodů, se používají vhodně zvolené metody jednokrokové, které mají podobnou přesnost jako vícekroková metoda, jejíž výpočet následuje.

Adams-Bashforthova metoda Příkladem explicitní metody je Adams-Bashforthova metoda. Hodnota v novém bodě je obecně vyjádřena vztahem

$$y_{n+1} = y_n + h(b_1 f_n + b_2 f_{n-1} + \cdots + b_k f_{n-k+1}), \tag{4.19}$$

kde koeficienty b_i závisí na počtu kroků k a pro $k < 6$ jsou dostupné v článku [10].

Adams-Moultonova metoda Jednou z implicitních metod je Adams-Moultonova metoda, pro jejíž výpočet slouží obecný vztah

$$y_{n+1} = y_n + h(b'_0 f_{n+1} + b'_1 f_n + b'_2 f_{n-1} + \cdots + b'_k f_{n-k+1}), \tag{4.20}$$

kde koeficienty b'_i také závisí na počtu kroků a pro $k < 6$ je lze najít v příspěvku [10]. Oproti předchozí metodě vyniká vyšší stabilitou, jedinou komplikací je nalezení neznámé hodnoty f_{n+1}.

4.2.2 Řešení obyčejných diferenciálních rovnic pomocí Taylorovy řady

Taylorův rozvoj je postup, kterým nahradíme funkci y nekonečnou mocninnou řadou složenou ze členů, k jejichž vypočtení se využívají derivace funkce y v daném bodě. Takto vzniklá mocninná řada nese název *Taylorova řada*. Pokud chceme vyjádřit hodnotu funkce pouze přibližně, můžeme zanedbat členy Taylorovy řady s vyššími derivacemi. Tím získáme *Taylorův polynom*, který aproximuje hodnotu funkce v daném bodě pomocí polynomu s konečným počtem členů.

Definice 4.2.1. Nechť funkce y má v bodě x_0 derivace všech řádů. Potom ***Taylorovou řadou*** funkce y v bodě x_0 nazýváme výraz

$$T_y^{x_0}(x) = y_0 + \sum_{k=1}^{\infty} \frac{y^{(k-1)}(x_0)}{k!}(x - x_0)^k. \tag{4.21}$$

Taylorovu řadu jsme schopni nalézt vždy, když má funkce y v bodě x_0 derivaci všech řádů. V opačném případě by rozvoj nekonvergoval k reprezentované funkci.

Definice 4.2.2. ***Taylorovým polynomem řádu*** n ***v bodě*** x_0 nazýváme polynom

$$\begin{aligned} T_n(x) &= y(x_0) + \frac{y'(x_0)}{1!}(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n = \\ &= \sum_{k=0}^{n} \frac{y^{(k)}(x_0)}{k!}(x - x_0)^k. \end{aligned} \tag{4.22}$$

Jedním z možných využití Taylorovy řady je numerické řešení obyčejných diferenciálních rovnic s počáteční podmínkou

$$y' = f(x, y), \quad y(x_0) = y_0, \tag{4.23}$$

kterým rozumíme nalezení posloupnosti

$$[y(x_0) = y_0],\ [y(x_1) = y_1],\ [y(x_2) = y_2],\ \ldots \tag{4.24}$$

V praxi je využívána aproximace funkce Taylorovým polynomem. Nejpřesnější a nejznámější metodou výpočtu nové hodnoty numerického řešení diferenciální rovnice (4.23) je sestavení Taylorova polynomu ve tvaru:

$$y_{i+1} = y_i + h \cdot y' + \frac{h^2}{2!} \cdot y'' + \cdots + \frac{h^n}{n!} \cdot y^{(n)}, \tag{4.25}$$

kde $h = x_{i+1} - x_i$ je integrační krok.

Při použití Taylorova polynomu vyvstává otázka na požadovanou přesnost výsledku. Pro tu platí, že narůstá se vzrůstajícím řádem Taylorova polynomu. Nejčastěji je řešena nadefinováním konstanty ε, která udává minimální požadovanou přesnost. Výpočet poté probíhá tak dlouho, dokud není absolutní hodnota dalšího přičítaného členu menší než ε. V případě, že posloupnost členů konverguje příliš pomalu, nebo nekonverguje vůbec, je třeba použít jiné techniky.

Kapitola 5

Přesnost numerických výpočtů

Při provádění numerických výpočtů s desetinnými čísly na počítačích je důležité porozumět způsobu uložení těchto čísel v paměti a principu vzniku zaokrouhlovacích chyb, který s tím velmi úzce souvisí. Jelikož jsou numerické výpočty určitých integrálů citlivé na přesnost, je tato kapitola věnována typům chyb, které při výpočtech vznikají, a způsobům, jakými lze vliv těchto nepříznivých jevů zmírnit.

5.1 Standard IEEE 754

V počítačích jsou všechna čísla převedena na binární reprezentaci a uložena na určitém, konečném počtu bitů. Tento počet a způsob převodu do binární podoby je dán datovým typem proměnné, do které číselnou hodnotu ukládáme. Jelikož i paměť počítače je konečná, nelze bez zaokrouhlení zaznamenat iracionální čísla, která mají nekonečný desetinný rozvoj, např. Ludolfovo číslo π, Eulerovo číslo e nebo $\sqrt{2}$. Kvůli nedostatečnému počtu bitů pro uložení hodnoty a způsobu binární reprezentace desetinných čísel však není možné zcela přesně reprezentovat ani nespočet jiných čísel s nenulovou desetinnou částí. Dochází tedy k nahrazení přesného čísla nejbližší možnou reprezentovatelnou hodnotou, čímž vzniká určitá chyba již v okamžiku uložení desetinného čísla.

Asociace IEEE vytvořila standard IEEE 754 dostupný na adrese [13], který specifikuje formát binárních čísel v plovoucí řádové čárce a způsob provádění aritmetických a jiných operací nad nimi. Jeho hlavním cílem je dosáhnout nezávislosti způsobu provádění operací na cílové architektuře – aby dva výsledky stejné operace se shodnými operandy spočítané na jiných zařízeních dávaly totožné výsledky. Dále tento standard definuje zaokrouhlovací pravidla, tedy jaké požadavky musí být splněny při zaokrouhlování čísel v průběhu konverzí a aritmetických operací, ošetření podmínek, jenž slouží k upozornění na nějakou neočekávanou událost (např. dělení nulou nebo přetečení), a formáty pro výměnu dat, které mají zajistit efektivní a kompaktní přenos. V současnosti je tento standard dodržován ve většině moderních zařízení.

Lze v něm najít specifikace hned několika formátů dat, které se od sebe vzájemně liší především počtem bitů, a tedy poskytovaným rozsahem a přesností. Každý formát obsahuje dvě reprezentace pro nekonečno, ∞ a $-\infty$, dva nečíselné typy `NaN`, a to tzv. tiché `NaN` a signalizační `NaN`, a konečná čísla. Každé konečné číslo je popsáno s využitím tří celých čísel reprezentujících znaménko, exponent a mantisu. Celkový počet bitů, na které se číslo ukládá, je dán formátem použitého datového typu. U čísel s jednoduchou přesností je využito 32 bitů, pro dvojitou přesnost je rezervováno 64 bitů a u čtyřnásobné přesnosti

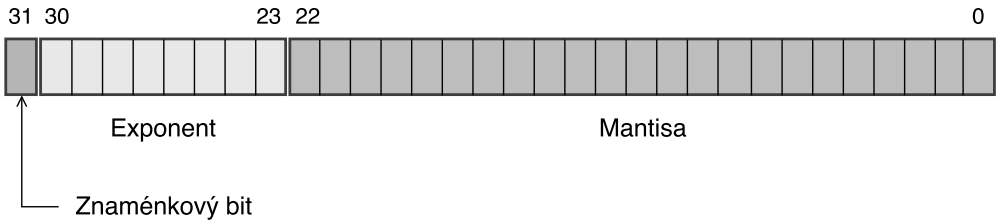
pracujeme se 128 bity. Detailnější přehled počtu bitů pro jednotlivé části u datových typů s dvojkovým základem je uveden v tabulce 5.1.

Název	Přesnost	Počet bitů exponentu	Počet bitů mantisy
binary16	Poloviční	5	10
binary32	Jednoduchá	8	23
binary64	Dvojitá	11	52
binary128	Čtyřnásobná	15	112

Tabulka 5.1: Počet bitů v jednotlivých částech základních binárních datových typů definovaných standardem IEEE 754 dostupným na stránkách [13]

5.1.1 Vnitřní reprezentace

Jak již bylo zmíněno, každé konečné číslo je v paměti uloženo s využitím rozložení na tři celá čísla – znaménko, exponent a mantisu. V poli hodnot reprezentujících daný datový typ je na pozici nejvíce významného bitu vždy tzv. znaménkový bit, který nabývá hodnot 0 pro kladné číslo a 1 pro číslo záporné. Za ním následuje několik bitů reprezentujících hodnotu exponentu a zbývající nejméně významné bity představují mantisu. Ta je uložena v normalizované formě, což znamená, že je mantisa implicitně zvětšena o 1, která se ale nikam neukládá. Díky tomu získáme u 32bitových čísel v plovoucí řádové čárce celkem 24 bitů mantisy – 23 explicitně uložených bitů a 1 implicitní. Rozložení jednotlivých částí pro konkrétní příklad, 32bitový binární datový typ, je vyobrazeno na obrázku 5.1.



Obrázek 5.1: Rozložení bitů v datovém typu binary32 (s jednoduchou přesností) definovaném standardem IEEE 754 (vlastní tvorba podle schématu v dokumentaci [13])

Převod čísla v plovoucí řádové čárce na dekadickou hodnotu lze provést s využitím vztahu

$$(-1)^S \cdot M \cdot B^E, \tag{5.1}$$

kde S je hodnota znaménkového bitu, M je mantisa, B představuje základ soustavy (nejčastěji 2 nebo 10) a E je hodnota exponentu.

5.2 Zdroje chyb numerických výpočtů

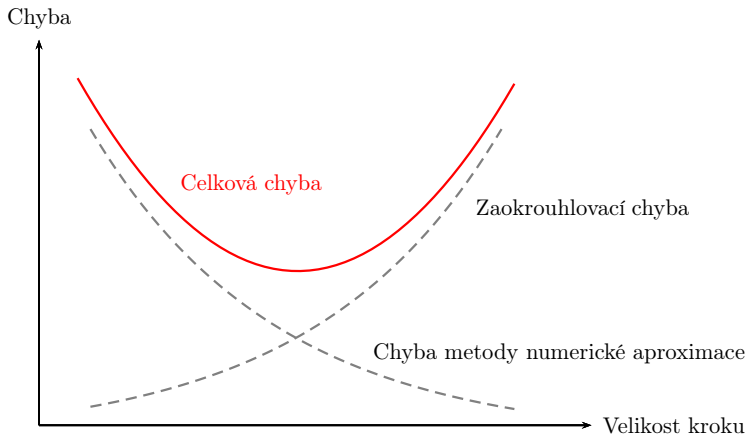
Při numerickém řešení integrálů hraje významnou roli požadavek na přesnost výpočtu. Velikost chyby výsledku (tzv. ***globální***, nebo-li ***akumulovaná chyba***) je rovna součtu všech lokálních chyb, kterých se dopouštíme v jednotlivých krocích výpočtu. Akumulovaná

chyba tedy v průběhu výpočtu průběžně narůstá o jednotlivé velikosti lokální chyby. Tu můžeme rozdělit na součet dvou hodnot od jednotlivých zdrojů této nepřesnosti, a to na chybu ***zaokrouhlovací*** a ***chybu metody numerické aproximace***.

Zaokrouhlovací chyba (*round-off error*) je dána neschopností číslicových počítačů přesně ukládat a zobrazovat všechna reálná čísla. Velikost této chyby závisí na přesnosti použité aritmetiky a lze ji ovlivnit datovým typem, který použijeme pro ukládání hodnot v průběhu výpočtu. Pokud bychom využili datový typ *double*, jenž ukládá desetinnou část hodnoty na 52 bitech a má přesnost $1 \cdot 10^{-15}$, byl by výpočet postižen nižší chybou, než kdybychom počítali s využitím datového typu *float*, který pro uložení hodnot za desetinnou čárkou využívá pouze 23 bitů a poskytuje přesnost $1 \cdot 10^{-7}$.

Chyba metody numerické aproximace (*truncation error*) je určena řádem metody, tedy počtem použitých členů polynomiálního rozvoje. Přesnost výpočtu narůstá se vzrůstajícím řádem metody. Tento druh chyby lze ovlivnit změnou hodnoty ε, která udává maximální přípustnou chybu způsobenou zanedbáním zbývajících členů rozvoje.

Nepřesnost výpočtu je také ovlivněna velikostí kroku. Zatímco velikost chyby numerické aproximace se zvyšuje s narůstající délkou kroku (jelikož je třeba využít více členů polynomiálního rozvoje pro dosažení stejné přesnosti, jaké jsme dosáhli při použití menšího kroku), vliv chyby zaokrouhlovací klesá. Není tedy pravidlem, že pro zvýšení přesnosti výpočtu je vhodné použít menší délku kroku. Závislost velikosti obou typů chyby i celkové chyby na délce kroku je znázorněna v grafu 5.2.



Obrázek 5.2: Závislost chyby numerických metod na velikosti kroku h (vlastní tvorba podle grafu dostupného ze zdroje [2])

5.3 Víceslovní aritmetika

Aritmetiku můžeme obecně definovat jako odvětví matematiky zabývající se zkoumáním čísel a operací, které jsou nad nimi definovány – sčítáním, odčítáním, násobením a dělením. Doteď jsme byli omezeni počtem bitů, který nám poskytly datové typy ze standardu IEEE 754. V nich bylo možno pro reprezentaci čísla využít nejvýše 128 bitů a docházelo tedy často k zaokrouhlování, jenž je pro výpočty citlivé na přesnost kritické. Za účelem zprostředkování reprezentace většího počtu desetinných čísel, a tedy zmenšení rozdílu mezi dvěma následujícími reprezentovatelnými hodnotami, vznikla ***víceslovní aritmetika***. Jedná se o aritmetiku, která pracuje s datovými typy o libovolném počtu bitů. Ten je teoreticky

neomezený, ve skutečnosti je jeho jediným limitem množství dostupné paměti. Výhodou použití víceslovní aritmetiky je její nezávislost na konstrukci cílové architektury. Knihovny, které ji implementují, poskytují řadu funkcí, jenž provádí všechny potřebné aritmetické operace a pro programátora zapouzdřují rozsáhlost jejich realizace. Datové typy, které tyto knihovny zprostředkovávají, ve skutečnosti sestávají z pole určitého počtu standardních datových typů. Nad nimi vytváří abstraktní vrstvu, díky níž umožňují programátorovi k celému úseku paměti přistupovat jako k jediné proměnné i s ní tak pracovat.

Práce s víceslovní aritmetikou je podstatně pomalejší než s klasickými čísly v plovoucí řádové čárce, jelikož se hodnoty takových proměnných nevejdou celé do registrů procesoru a provádění veškerých operací je řízeno především na softwarové úrovni. Víceslovní aritmetika však necílí na rychlost výpočtů, nýbrž na vyšší přesnost, a proto je její použití pro numerické výpočty (nejen) určitých integrálů velmi vhodné.

Způsob implementace aritmetických operací nad datovými typy knihoven víceslovní aritmetiky bude nyní ukázán na operacích pro celá čísla. Implementačně nejjednodušší je vykonávání operací sčítání a odčítání. Ty jsou prováděny postupně po jednotlivých slovech a mezi nimi je předáván příznak přenosu. Porovnání dvou čísel je realizováno pomocí komparace jednotlivých slov od číslic nejvyššího řádu, až dokud není nalezena rozdílná hodnota. Principy vyčíslování násobení a dělení jsou již značně složitější, detailnější informace o těchto i dalších operacích lze nalézt například na manuálových stránkách knihovny GMP[1]. Pro čísla v plovoucí řádové čárce se používají modifikace výše zmíněných algoritmů.

5.3.1 Knihovny implementující víceslovní aritmetiku

Existuje celá řada knihoven, které poskytují datové typy pro víceslovní aritmetiku. Dalo by se říci, že ke každému rozšířenému modernímu programovacímu jazyku lze nalézt alespoň jednu takovou knihovnu. Pro jazyk C je mimo jiné možné využít knihovnu GMP, na jejím základě vybudovaná knihovna GMPY je určena pro Python. Mezi jejich ekvivalenty pro jazyk Java patří například knihovny JAS a JScience. Při implementaci této práce byla využita již zmíněná knihovna GMPY, která bude blíže představena v následující sekci.

GMPY

GMPY je rozšiřující modul pro skriptovací jazyk Python, který poskytuje přístup ke knihovně GMP a zprostředkovává tak funkce rychlé víceslovní aritmetiky. GMPY jako takové je napsáno v jazyce C. Mezi dostupné datové typy patří `gmpy.mpz` pro víceslovní celočíselné hodnoty, `gmpy.mpq` sloužící k uložení racionálních čísel a `gmpy.mpf` umožňující neomezenou přesnost desetinným číslům v plovoucí řádové čárce. V současnosti je dostupná i novější verze knihovny, GMPY2. Ta již obsahuje i datový typ pro komplexní čísla `gmpy2.mpc`, je lépe kompatibilní s Pythonem verze 3 a číselné typy v plovoucí řádové čárce nyní poskytuje knihovna MPFR, je tedy nutné používat direktivu `gmpy2.mpfr`. Bližší informace o zmíněných datových typech a seznam operací, jejichž použití knihovna GMPY2 umožňuje, lze nalézt na manuálových stránkách knihovny [18].

Příklad použití knihovny GMPY2 – nastavení přesnosti na 100 bitů a vypočtení druhé odmocniny z 5:

```
import gmpy2
gmpy2.get_context().precision = 100
gmpy2.sqrt(5)
```

[1] Manuálové stránky knihovny GMP jsou dostupné na adrese https://gmplib.org/manual.

5.3.2 Prostředí poskytující víceslovní aritmetiku

Většina matematických a simulačních prostředí využívá některou z knihoven implementujících víceslovní aritmetiku pro poskytnutí co největší přesnosti výpočtů. Můžeme sem zařadit programy jako Maple, MATLAB nebo Mathematica. V aplikaci implementované jako součást této práce byl využit simulační nástroj TKSL/C. Následně byla pro porovnání přesnosti výsledků použita prostředí MATLAB a Maple. Všechny využité programy budou nyní popsány blíže.

TKSL/C

Nástupcem původního prostředí TKSL/386 se stal simulační nástroj TKSL/C. Oba programy jsou blíže popsány na stránkách [15] a v práci [8]. TKSL/C má podobnou syntaxi a k řešení diferenciálních rovnic také využívá Taylorovu řadu. Na rozdíl od svého předchůdce je však naprogramován v jazyce C/C++, lze jej tedy bez podpůrných programů spouštět na současných počítačích, a to na několika operačních systémech. Mezi další výhody TKSL/C patří také možnost využití víceslovní aritmetiky za použití funkcí z knihovny GMP, odstraněný horní limit počtu řešených rovnic a způsob spouštění z příkazové řádky, čímž je umožněno provádění řady výpočtů pomocí skriptu bez potřeby manuálního zásahu uživatele.

Syntaxe vstupních dat je oproti původní verzi také zjednodušena. Nyní je možno zapisovat přímo jednotlivé diferenciální rovnice ve formátu `y' = f(t,y) & 0;`. Za znakem `&` se zapisuje počáteční podmínka a jednotlivé rovnice jsou ukončeny středníkem.

Nastavení parametrů výpočtu se provádí pomocí přepínačů, s nimiž je program spuštěn. Diferenciální rovnice je poté možno vepsat do konzole, nebo zadáním souboru s rovnicemi na pozici posledního parametru. Popis přepínačů, které byly využity při implementaci této práce, je uveden v tabulce 5.2.

Parametr	Význam
`A`	nastavení přesnosti výpočtu
`W`	nastavení délky víceslovní aritmetiky v bitech
`t`	nastavení maximálního času, pro který se má úloha řešit
`p`	nastavení přesnosti pro výpis výsledku v počtu znaků
`w`	nastavení šířky tisku v počtu znaků

Tabulka 5.2: Parametry TKSL/C, které byly využity při tvorbě této práce (více informací o těchto i dalších přepínačích lze najít na stránkách [15])

Příklad zápisu diferenciálních rovnic pro výpočet integrálu funkcí $sin(x)$ a e^x při provedení substituce t za x:

```
I1' = sin(t) & 0;
I2' = exp(t) & 0;
```

MATLAB

MATLAB je matematické prostředí vyvíjené firmou MathWorks. Název získal zkratkou ze slov *matrix laboratory*, což v překladu znamená „maticová laboratoř“. Jedním z mnoha

cílů MATLABu je tedy rychlé a přesné provádění maticových operací. Jeho využití je však velmi široké, od řešení simulací, přes oblast zpracování signálů až po analýzu dat. MATLAB také podporuje víceslovní aritmetiku, k tomuto účelu je v něm implementována funkce `vpa` (zkratka *variable-precision arithmetic*). Na rozdíl od TKSL/C je MATLAB vybaven grafickým uživatelským rozhraním a nástroji pro vykreslování grafů. Programy se v MATLABu zapisují formou skriptů. Jejich syntaxe je podobná Pythonu a jiným skriptovacím jazykům, podrobnosti o způsobu zápisů příkazů a struktuře programů jsou umístěny na oficiálních stránkách MATLABu [16].

Pro výpočet (nejen) dvojnásobných integrálů nabízí MATLAB hned několik příkazů. Prvním z nich je funkce `integral2(fun, xmin, xmax, ymin, ymax)`, která provádí numerické vyčíslení funkce `fun` přes oblast $\langle$`xmin`, `xmax`$\rangle \times \langle$`ymin`, `ymax`$\rangle$. Využívá principu „rozděl a panuj“ (*divide and conquer*) a postupuje iterativně, postupně tedy integruje přes jednotlivé dimenze. Její druhou variantou je funkce `quad2d(fun, a, b, c, d)`, jenž provádí také numerickou integraci funkce `fun` na oblasti $\langle$`a`, `b`$\rangle \times \langle$`c`, `d`$\rangle$. Tato funkce také pracuje na výše zmíněném principu. Rozděluje integrovanou oblast na kvadranty a poté aproximuje integrál na každém kvadrantu pomocí dvojrozměrného kvadraturního pravidla. Pokud chyba není dostatečně malá, provede se znovu rozdělení kvadrantu. Další možností integrace je funkce `polyint(p, k)`. Ta pracuje na principu polynomiální integrace a jejími parametry jsou vektor koeficientů polynomu `p` a nepovinná integrační konstanta `k` s implicitní hodnotou 0.

Příklad obsahu skriptu používajícího výše představené funkce:

```
fun = @(x,y) y.*sin(x)+x.*cos(y);
integral2(fun,0,2,0,1)
quad2d(fun,0,2,0,1)
p = [3 0 -4 10 -25];
polyint(p)
```

Maple

Maple je matematický program určený k analýze dat, vizualizaci, zkoumání a řešení matematických úloh. Stejně jako MATLAB disponuje grafickým uživatelským rozhraním a vlastní syntaxí příkazů, na rozdíl od předcházejícího jej vyvíjí firma Maplesoft. Více informací o prostředí Maple lze nalézt na oficiálních stránkách produktu [17].

Co se týče řešení určitých integrálů, nabízí Maple mimo jiné funkci pro výpočet lichoběžníkové metody `trapezoid(f(x), x = a..b, n)` a funkci implementující metodu Simpsonovu `simpson(f(x), x = a..b, n)`. Jejich parametr `f(x)` obsahuje integrovanou funkci, `a` a `b` jsou integrační meze a `n` představuje počet dílčích intervalů použitých při výpočtu. Kombinace těchto funkcí s direktivou `evalf` zajišťuje čistě numerickou integraci.

Příklad výpočtu určitého dvojného integrálu funkce $y \cdot \sin(x) + x \cdot \cos(y)$ v prostředí Maple s využitím výše zmíněných funkcí:

```
with(student):
evalf(trapezoid(trapezoid(y · sin(x) + x · cos(y), x = 0..1,1), y = 0..1, 1), 100);
evalf(simpson(simpson(y · sin(x) + x · cos(y), x = 0..1,1), y = 0..1, 1), 100);
```

Kapitola 6

Specifikace a návrh aplikace

Tato kapitola pojednává o popisu a návrhu aplikace ***mis***[1], jenž byla vytvořena v rámci této práce a jejímž hlavním účelem je výpočet dvojrozměrných integrálů s využitím nástroje TKSL/C. V následujícím textu budou nejdříve představeny hlavní cíle a požadavky na aplikaci, které byly stanoveny před samotným začátkem realizace. Zmíníme se i o způsobu, jakým bude aplikace testována. Druhou významnou část této kapitoly tvoří sekce věnovaná popisu metody, kterou tato aplikace implementuje.

6.1 Cíle aplikace

Hlavním úkolem aplikace je výpočet dvojrozměrného integrálu za využití numerické metody popsané v sekci 6.4.

U skriptu jsou požadovány následující vstupy:

- rovnice obsahující až dvě proměnné,
- krajní body oblasti, přes kterou bude integrál počítán,
- počet podintervalů, na něž bude integrovaná oblast v každé dimenzi rozdělena,
- metoda zvolená pro výpočet integrálu ve druhé integrované dimenzi (z hodnot, které budou získány pomocí TKSL/C), přičemž na výběr budou tyto tři možnosti: obdélníková, lichoběžníková a Simpsonova.

Výstupem aplikace bude pro zachování přehlednosti pouze výsledek určitého dvojrozměrného integrálu, uživatel by neměl být zatěžován množstvím pomocných výstupů. V případě zájmu o podrobnější informace ohledně průběhu výpočtu a pomocné hodnoty by však měl mít možnost si je zobrazit.

6.2 Požadavky

Aplikace `mis` byla implementována tak, aby splňovala několik na ni kladených požadavků. Prvním z nich je přenositelnost, aplikace by měla být spustitelná jak na operačních systémech řady Windows, tak na linuxových distribucích. Mezi další nároky patří jednoduché uživatelské rozhraní, které postihuje nastavení všech důležitých aspektů běhu aplikace,

[1] Název aplikace `mis` vznikl zkratkou z anglických slov *multiple integral solver*.

avšak nenutí uživatele všechny parametry aktivně využívat a orientovat se v nich. Na implementaci samotnou byl stanoven požadavek, aby skript pracoval s víceslovní aritmetikou pro vyšší přesnost výsledků. Aplikace by také měla být schopna zkontrolovat hodnoty vstupních argumentů a případně ošetřit neočekávaná a nevalidní data, vypsat příslušné chybové hlášení a skončit.

6.3 Návrh aplikace

Při návrhu aplikace byl zvolen způsob realizace ve formě konzolové aplikace. Možnost řízení parametrů běhu programu pomocí nastavení hodnot přepínačů zadaných při spuštění se zdál být ideální volbou. Uživatel tak bude moci využívat jenom ty parametry, které jsou pro něj důležité, a ostatní mohou být nastaveny na implicitní hodnoty. Další nespornou výhodou tohoto přístupu je možnost začlenění implementovaného skriptu do jiného nebo rychlé spouštění jednoho výpočtu za druhým.

Ve fázi návrhu byla také vytvořena specifikace finálního testování aplikace, tedy způsobu, kterým bude ověřeno, do jaké míry aplikace splňuje požadavky a jak přesné výsledky poskytuje. Byla sestavena sada funkcí různých typů, které mají být řešeny jak ve vznikající aplikaci, tak i v dalších prostředích, a poté porovnány s analytickým řešením. Výstupem testování by mělo být také zjištění, jaká z implementovaných metod dosahuje nejlepší a která naopak nejhorší přesnosti. Výsledky provedených testů jsou uvedeny v kapitole 8.

6.4 Implementovaná metoda řešení integrálů

V této sekci bude představena metoda, která byla navržena pro realizaci této práce. Nejprve bude uvedena motivace k jejímu zavedení a poté přejdeme k popisu metody samotné.

6.4.1 Motivace

Nástroj TKSL/C umožňuje velmi přesné výpočty integrálů pomocí převodu na diferenciální rovnice, které řeší s využitím Taylorovy řady. Výhodou TKSL/C není pouze nastavitelná přesnost výpočtu a s tím související nízká chyba výsledku, ale také rychlost, s jakou je schopno tato řešení poskytnout. Jedno omezení však TKSL/C má, a tím je počet dimenzí řešeného integrálu – lze v něm řešit pouze integrály jednorozměrné. Nabízí se tedy snaha o navržení metody, která by využívala výpočet integrálů z TKSL/C a získané výsledky nějakým způsobem transformovala na výsledek integrálů ve dvou a více dimenzích. Jakým způsobem toho lze docílit bude předmětem následujících odstavců.

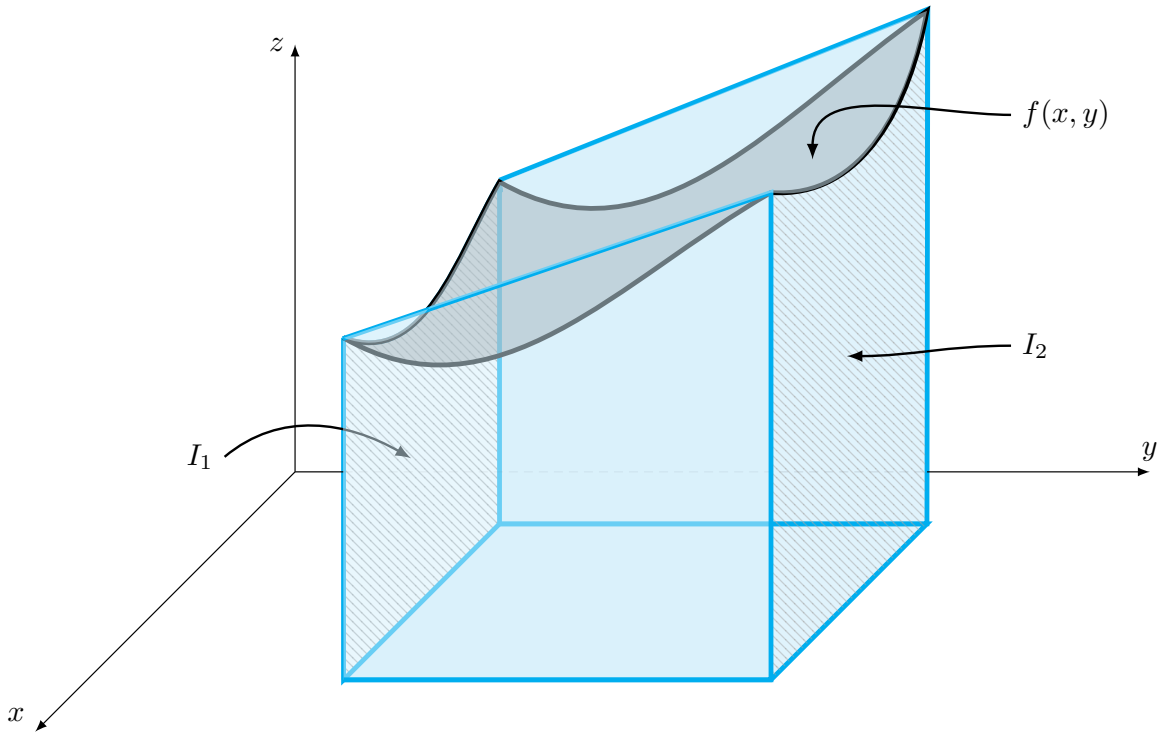
6.4.2 Popis metody

Navrženou metodu lze zařadit mezi numerické metody výpočtu určitých integrálů. Implementovaná verze slouží k řešení dvojrozměrných integrálů, lze ji však zobecnit na libovolný počet dimenzí.

Základní myšlenkou výpočtu je představa podobná Riemannově sumě, avšak transformovaná do více dimenzí. Jednorozměrný integrál udává obsah plochy pod funkcí, zatímco dvojný integrál znázorňuje objem vymezený grafem funkce. Na obdélníkové oblasti můžeme tedy dvojný integrál aproximovat obsahem plochy řezu ve směru osy y vynásobeným „šířkou“ – rozdílem mezí na téže ose. V takovém případě by byl integrál aproximován objemem jednoho tělesa. Pokud bychom rozdíl mezí na ose y rozdělili na několik stejně dlouhých

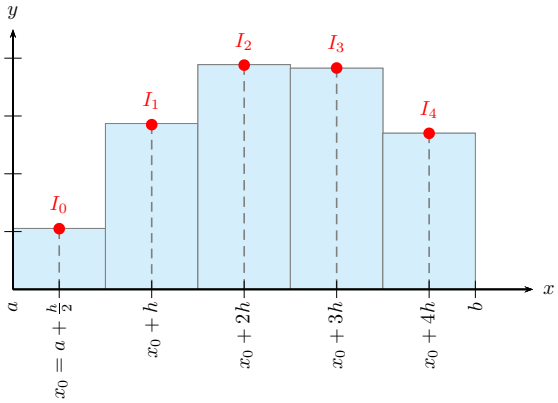
úseků a vypočítali hodnotu jednorozměrného integrálu pro každý z dílčích intervalů, po jejich sečtení a vynásobení délkou podintervalu bychom získali přesnější aproximaci dvojného integrálu. Pro počet dílčích intervalů jdoucí do nekonečna, a tedy jejich šířku limitně se blížící nule, bychom získali velmi přesnou aproximaci původního dvojného integrálu blížící se jeho skutečné hodnotě.

Výše popsaný princip je znázorněn na obrázku 6.1. Šedou barvou je v něm vyznačena integrovaná funkce, modře šrafovaně potom plochy I_1 a I_2, které představují jednorozměrné integrály ve dvou řezech vedených kolmo k ose y. Ty jsou ve vytvářené aplikaci počítány pomocí prostředí TKSL/C. Dále je v něm modře vyznačeno těleso, kterým je integrál aproximován. To vzniklo aplikováním jednoduchého lichoběžníkového pravidla na výsledky jednorozměrných integrálů a byl použit jeden dílčí interval.

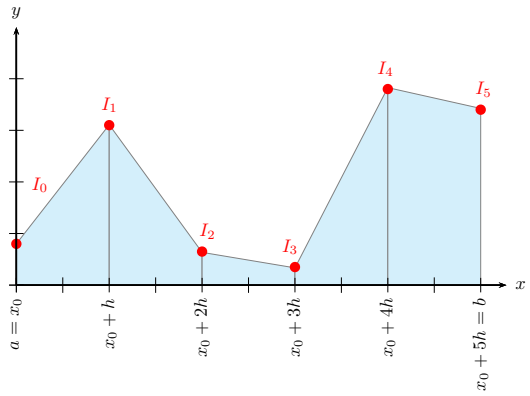


Obrázek 6.1: Aproximace dvojného integrálu objemem dvou těles

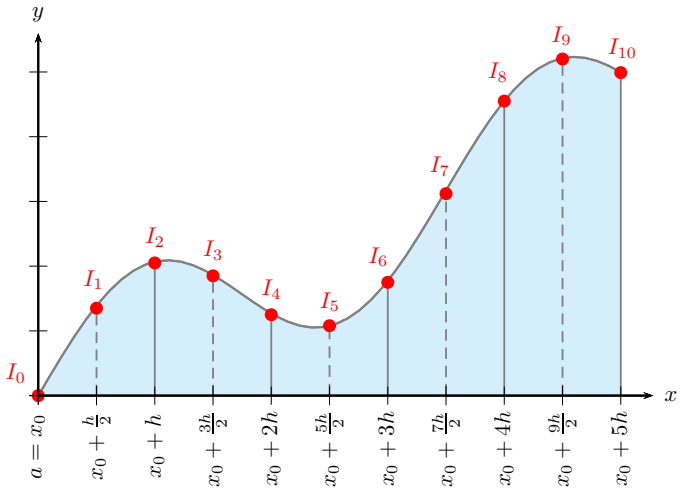
Doposud nebylo specifikováno, v jakých bodech dílčích intervalů budou řezy prováděny, protože to pro porozumění jádru metody není nezbytné a také se to liší podle použité metody numerické integrace ve druhé integrované dimenzi. U obdélníkové metody se vždy využívají hodnoty v polovině daného podintervalu, lichoběžníková metoda zase využívá oba krajní body vymezující dílčí interval. Pro aplikaci Simpsonovy metody potřebujeme znát hodnoty jak bodů krajních, tak i v bodě uprostřed intervalu. Zvolená metoda výpočtu nám tedy určuje nejen vztah, jakým ze získaných hodnot určíme výsledek integrálu ve druhé integrované dimenzi, ale také umístění bodů, kterými povedou jednotlivé řezy funkce, a zároveň i jejich počet. Grafické znázornění aplikace výše zmíněných numerických metod na výpočet dvojného integrálu z hodnot integrálů jednorozměrných vedených řezy původní funkce je uvedeno v grafech 6.2.



(a) Obdélníková metoda



(b) Lichoběžníková metoda



(c) Simpsonova metoda

Obrázek 6.2: Výpočet dvojného integrálu aplikací numerických metod na výsledky jednorozměrných integrálů

Ukázka použití metody

Pro lepší pochopení implementované metody bude nyní uveden jednoduchý příklad řešený všemi třemi numerickými metodami.

Příklad Mějme daný dvojný integrál

$$\int_0^1 \int_0^4 x + 10 \cdot y \, \mathrm{d}x \, \mathrm{d}y \tag{6.1}$$

a požadovaný počet podintervalů nechť je 2. Pořadí diferenciálů nám říká, že budeme nejprve integrovat podle proměnné x. V prvním kroku tedy vytvoříme soubor integrálů jedné proměnné, ve kterých bude x jako proměnná a souřadnici y nahradíme v každém řezu její fixní hodnotou. Vzniknou nám tedy tyto integrály:

1. Obdélníková metoda vytváří tolik řezů, kolik je požadováno podintervalů, a to v bodech v polovině každého dílčího intervalu. Vznikne nám tedy následující množina

integrálů:

$$\int_0^4 x + 10 \cdot 0,25 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 0,75 \,\mathrm{d}x.$$

2. Lichoběžníková metoda využívá krajní body podintervalů, počet provedených řezů bude tedy o jedna vyšší než počet dílčích intervalů. Po nahrazení souřadnice y fixními hodnotami získáme následující soubor integrálů:

$$\int_0^4 x + 10 \cdot 0 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 0,5 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 1 \,\mathrm{d}x.$$

3. Simpsonova metoda kombinuje oba předchozí přístupy a v každém podintervalu používá jak body krajní, tak i bod uprostřed intervalu. Počet vzniklých řezů můžeme tedy vyjádřit vztahem $2 \cdot p + 1$, kde p představuje počet dílčích intervalů. Pro Simpsonovu metodu dostaneme tyto integrály:

$$\int_0^4 x + 10 \cdot 0 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 0,25 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 0,5 \,\mathrm{d}x \qquad \int_0^4 x + 10 \cdot 0,75 \,\mathrm{d}x$$

$$\int_0^4 x + 10 \cdot 1 \,\mathrm{d}x.$$

V dalším kroku řešení získáme hodnoty všech jednorozměrných integrálů. V implementaci je k tomuto účelu využito prostředí TKSL/C, u tohoto jednoduchého příkladu můžeme použít analytické řešení. Získáme následující sadu dat:

y	0	0,25	0,5	0,75	1
Hodnota integrálu	8	18	28	38	48

V poslední fázi výpočtu využijeme získané hodnoty jednonásobných integrálů a dosazením do vztahu pro příslušnou metodu dostáváme řešení příkladu.

1. Obdélníková metoda:

$$(1 - 0) \cdot \frac{18 + 38}{2} = 28.$$

2. Lichoběžníková metoda:

$$\frac{1 - 0}{2 \cdot 2} \cdot (8 + 2 \cdot 28 + 48) = 28.$$

3. Simpsonova metoda:

$$\frac{1 - 0}{4 \cdot 3} \cdot (8 + 4 \cdot 18 + 2 \cdot 28 + 4 \cdot 38 + 48) = 28.$$

Řešení integrálu 6.1 vyšlo všemi třemi metodami 28, což je zároveň i analytické řešení. Tento výsledek není postižen žádnou chybou díky tomu, že všechny výše zmíněné numerické metody poskytují přesnou aproximaci lineárních funkcí.

Kapitola 7

Implementace aplikace `mis`

Tato kapitola bude detailně zaměřena na popis finální implementace aplikace `mis`. Na úvod budou zmíněny systémové požadavky, které zahrnují především potřebné nástroje a knihovny. Druhá sekce je věnována způsobu, jakým jsou řešeny vstupní parametry aplikace a bude uveden význam jednotlivých přepínačů. Poté bude provedena dekompozice implementace skriptu na jednotlivé funkce, jejichž význam a způsob realizace bude popsán, a na závěr budou nastíněny různé modifikace aplikace, o které by bylo možné ji do budoucna rozšířit a zdokonalit.

7.1 Systémové požadavky

Pro realizaci aplikace byl zvolen skriptovací jazyk Python, a to díky své relativně jednoduché syntaxi a možnosti importování funkcí, které poskytuje operační systém a které lze vykonávat přímo ve skriptu stejným způsobem, jako kdyby byly vepsány do konzole. Jelikož se jedná o interpretovaný jazyk, je nezbytné mít jej nainstalovaný. Další nutností je knihovna gmpy2 poskytující víceslovní aritmetiku, z čehož vyplývá požadavek na verzi Pythonu, která musí být 3.0 nebo vyšší a zároveň kompatibilní s nainstalovanou verzí knihovny gmpy2. Poslední potřebnou součástí je TKSL/C, jenž je využíváno pro přesný numerický výpočet jednorozměrných integrálů.

7.2 Implementace vstupů aplikace

Na základě návrhu aplikace bylo implementováno zadávání parametrů řídících výpočet dvojrozměrného integrálu prostřednictvím hodnot přepínačů, s nimiž je aplikace spuštěna. Navíc byla implementace rozšířena o možnost načtení vstupních hodnot ze souboru, který je aplikaci zadán jako argument `input`. Jeho formát syntaxe je uveden v souboru README, který je také součástí této práce. Způsob zadávání parametrů s využitím externího souboru může být užitečný především při opakovaném řešení integrálů různých funkcí, avšak se stejnými parametry. Poté stačí skript spouštět pouze se zadáním integrované funkce a vstupního souboru.

Přehled významu všech parametrů aplikace `mis` je uveden v tabulce 7.1. Valná většina parametrů je volitelných. V případě, že uživatel některý z nich nezadá, použije se implicitní hodnota (kromě přepínačů `help` a `input`, jenž nejsou pro jádro výpočtu nezbytné). Výjimku tvoří parametr `function`, ten je nutné zadat vždy. Implicitní hodnoty uvedených

přepínačů, možnosti validně zadaných dat a příklady použití lze také najít v přiloženém souboru README.

Parametr	Význam
`-h`, `--help`	vypsání nápovědy a ukončení aplikace
`-i`, `--input`	načtení parametrů výpočtu ze souboru zadaném jako argument
`--xmin`	dolní mez integrační oblasti na ose x
`--xmax`	horní mez integrační oblasti na ose x
`--ymin`	dolní mez integrační oblasti na ose y
`--ymax`	horní mez integrační oblasti na ose y
`-s`, `--stepcount`	počet podintervalů, na které bude rozdělena integrační oblast
`-m`, `--method`	metoda numerické aproximace integrálu
`-f`, `--function`	integrovaná funkce
`-e`, `--epsilon`	minimální přesnost výpočtu jednorozměrných integrálů
`-w`, `--width`	počtu bitů pro víceslovní aritmetiku GMP
`-p`, `--precision`	počet číslic zobrazeného výsledku
`-d`, `--differential`	pořadí integrovaných proměnných[1]
`-a`, `--average`	zapne/vypne výpočet jako průměr výsledků integrace $dxdy$ a $dydx$[1]
`-t`, `--taylor`	přepnutí mezi výpočtem Taylorovou řadou a numerickou metodou[1]

Tabulka 7.1: Parametry spouštění aplikace `mis`

7.3 Následnost a specifikace implementovaných funkcí

V této sekci bude podrobně popsána implementace aplikace. Všechny implementované funkce zde budou uvedeny v takovém sledu, v jakém jsou volány z funkce `main`, a jejich název bude doplněn účelem použití.

`parse_arguments`

Tato funkce slouží ke zpracování argumentů zadaných z příkazové řádky. Využívá třídu `ArgumentParser` z knihovny `argparse`, která implicitně zajišťuje také kontrolu datových typů zadaných argumentů a u přepínačů, které mají přesně danou množinu přípustných hodnot, vyhodnotí jiná data jako nevalidní vstup. Tato funkce vrací slovník všech parametrů, hodnoty nezadaných přepínačů jsou nastaveny na implicitní čísla, či řetězce.

[1] Detailní popis účelu tohoto přepínače je uveden v kapitole 7.4.

`get_parameters_from_file`

Jedná se o funkci volanou pouze v případě, že je zadán přepínač `input`. Jejím účelem je získání hodnot parametrů ze vstupního souboru. Nejprve volá funkci `get_file_content`, která se zadaný soubor pokusí otevřít a vrací jeho obsah, případně skončí s chybou. Poté skript přechází do funkce `parse_input_file`. Zde je s využitím několika regulárních výrazů zpracován text vstupního souboru a rozpoznané hodnoty parametrů jsou uloženy do slovníku místo implicitních, nebo zadaných z příkazové řádky. Vstupní soubor má tedy před hodnotami zadanými při spuštění přednost.

`check_parameters`

Význam funkce `check_parameters` spočívá v podrobné kontrole parametrů zadaných výše zmíněnými dvěma způsoby. Je zde provedeno ověření validity hodnot získaných ze souboru a navíc určitá kontrola logické správnosti dat. Testuje se například, zda není počet bitů pro víceslovní aritmetiku a požadovaný počet dílčích podintervalů menší, nebo roven nule, dále jestli je zadána integrovaná funkce a také zda se má výsledek vypsat zaokrouhlený na kladný počet číslic.

`create_subfolder`

Po ověření, zda jsou všechny zadané hodnoty validní, je zavolána funkce na vytvoření podsložky v aktuální složce, do které budou později ukládány všechny soubory související s aktuálním výpočtem. Nejprve je získán název podsložky jako návratové hodnoty z funkce `get_dirname`. Ta jej vytvoří na základě názvu souboru se vstupními argumenty, pokud je zadán, a to přidáním „`_dir`“ za jméno souboru. Jinak název sestává pouze ze slova „`dir`“ a první volné číslovky v aktuálním adresáři. Po návratu zpět do funkce `create_subfolder` je případně odstraněna již existující složka se stejným názvem a vytvořena nová pro aktuální běh aplikace.

`create_logfile`

Účel funkce `create_logfile` je vytvoření pomocného souboru obsahujícího informace o výpočtu. Jeho hlavním smyslem je uchování parametrů výpočtu v případě, že byly zadány do příkazové řádky a uživatel by tak po spuštění více běhů aplikace nevěděl, jaká funkce a na jakém intervalu byla řešena. Skript se nejprve pokusí v podadresáři přiřazeném tomuto výpočtu vytvořit soubor s názvem `logfile`. Po neúspěchu následuje výpis chybového hlášení a návrat z funkce, jelikož výpočet může stále korektně proběhnout. V případě úspěšného vytvoření souboru jsou do něj zapsány všechny informace o výpočtu, tedy obsah slovníku uchovávajícího hodnoty parametrů transformovaný do čitelnější formy.

`create_tkslc_input`

Tato funkce je stěžejní částí celé aplikace. Pokud by uvnitř ní došlo k chybě, celý výpočet by s největší pravděpodobností neproběhl v pořádku. Jejím smyslem je vytvoření souboru obsahujícího diferenciální rovnice, které budou následně řešeny v prostředí TKSL/C. Nejprve je vytvořen nový soubor s názvem `input.tksl` a ošetřen vznik případné chyby. V dalším kroku proběhne převod funkce zadané uživatelem do podoby, v jaké ji je schopno zpracovat TKSL/C. To se provádí voláním funkce `function_to_tksl_form`. Poté je na základě proměnné, podle které je integrováno, vypočtena délka kroku, určen název proměnné, která

bude mít v jednotlivých řezech fixní hodnotu, a počáteční souřadnici, v níž bude proveden první řez. Následně je pro každý řez dvojným integrálem vytvořena příslušná reprezentace integrované funkce náhradou fixní proměnné za její hodnotu v aktuálním řezu a poté zapsána do souboru ve tvaru:

```
I{number}' = {function} & 0;
```

kde `number` udává pořadové číslo řezu a `function` představuje aktuální reprezentaci funkce. Na závěr je do souboru připsána poslední funkce, která přímo aplikuje vybranou numerickou metodu na výsledky výše zapsaných diferenciálních rovnic.

`function_to_tksl_form`

Cílem této funkce je převod uživatelem zadané integrované funkce do takové podoby, kterou přijímá nástroj TKSL/C. Probíhá tedy záměna názvu integrované proměnné za t, to je v TKSL/C rezervováno pro čas. Navíc je umocnění Eulerova čísla e na určitou hodnotu zapsané ve formátu „$e\hat{}(x)$“ nahrazeno výrazem „$\exp(x)$“.

`run_computation`

Funkce `run_computation` zjistí, podle jaké proměnné bude integrováno dříve, a nastaví příslušné meze integrace. V dalším kroku vyhodnotí, o jaký operační systém se jedná, a na základě těchto informací a hodnot vstupních parametrů vytvoří příkaz na spuštění výpočtu v TKSL/C, který vzápětí také vykoná. Prováděný příkaz má tvar:

```
cltksl -A {args.epsilon} -W {args.width} -t {args.[xmax|ymax]}
-p {args.precision} -w {args.precision} {input_filepath} »
{output_filepath} 2» {log_filepath},
```

kde proměnné `args.*` obsahují hodnoty vstupních parametrů výpočtu a `input_filepath`, `output_filepath` a `log_filepath` značí po řadě cesty k souboru se vstupními rovnicemi pro TKSL/C, souboru, do kterého bude uložen výsledek výpočtu a souboru, kam bude přesměrován pomocný výstup z TKSL/C.

`get_results`

V okamžiku, kdy je výpočet v prostředí TKSL/C dokončen, volá se funkce `get_results`. V ní je nejprve zavolána funkce `open_outputfile`, ve které je proveden pokus otevřít soubor s výsledky. Opět je v případě neúspěchu vypsáno chybové hlášení a aplikace uzavřena, nebo je vrácen deskriptor souboru. Ve funkci `get_results` se potom přečte obsah souboru a pomocí operací nad řetězci je získána a vrácena hodnota výsledku.

`change_differential`

Tato funkce je volána po dokončení prvního výpočtu, a to pouze v případě, že byl nastaven přepínač `average`. Jejím jediným argumentem je pořadí diferenciálů použité u právě provedeného výpočtu, návratovou hodnotou je potom opačné pořadí diferenciálů.

`compute_2d_integral`

Jedná se o funkci volanou v případě, že bylo pomocí přepínače `taylor` vypnuto použití Taylorovy řady, integrace tedy proběhne v obou dimenzích aplikací numerických metod.

V prvním kroku je vytvořena stromová reprezentace integrované funkce. K tomuto účelu je volána funkce `parse_function`. Po návratu z ní se pro každý řez dvojným integrálem volá funkce `compute_1d_integral`, ve které probíhá integrace podle první integrované proměnné. Její návratové hodnoty jsou uloženy do pole, na které je vzápětí aplikována zvolená numerická metoda ve funkci `evaluate_numeric_method`. Vrácená hodnota představuje vypočtené řešení dvojného integrálu a je ihned vrácena i z funkce `compute_2d_integral`.

`parse_function`

Cílem této funkce je v první fázi převod na vstupu zadané integrované funkce z formátu, který využívá aplikace `mis`, do podoby používané knihovnou `parser`. Následně je proveden převod transformované funkce do stromové reprezentace s ošetřením vzniku chyby, která by znamenala nevalidně zadanou funkci.

`compute_1d_integral`

Hlavním účelem funkce `compute_1d_integral` je provést numerickou integraci jednorozměrného integrálu. Na základě toho, podle které proměnné bude integrace probíhat, je vypočtena délka integračního kroku a určena hodnota fixní proměnné pro aktuální řez. Poté je pro každý uzlový bod dosazena příslušná hodnota souřadnic x a y do stromové reprezentace funkce a voláním funkce `eval` je vyčíslena. Všechny tyto dílčí výsledky jsou uloženy do pole, na které je poté aplikován zadaný způsob numerické integrace ve funkci `evaluate_numeric_method`.

`evaluate_numeric_method`

Funkce `evaluate_numeric_method` je volána pro provedení numerické integrace nad hodnotami v obou dimenzích integrace. Podle zvolené metody zavolá příslušnou funkci, která danou metodu implementuje. Řízení je tedy předáno jedné z funkcí `compute_rect_method`, `compute_trap_method` a `compute_simp_method`. Výsledek určitého integrálu je návratovou hodnotou každé z nich.

`write_result_to_logfile`

Tato funkce je poslední volanou před skončením skriptu. Nejprve je zde proveden pokus otevřít soubor uchovávající informace o výpočtu i s ošetřením případného vzniku chyb. Poté je k jeho obsahu připsána hodnota výsledku výpočtu a soubor opět uzavřen.

7.4 Implementovaná rozšíření nad rámec specifikace

V průběhu implementace byla navržena další rozšíření, která jsou nyní součástí aplikace. Všechny tři jsou řešeny prostřednictvím přidaných přepínačů a aplikaci lze spouštět jak s jejich využitím, tak i standardním způsobem. V této sekci budou všechny přepínače představeny a bude detailně vysvětlen jejich význam i princip práce.

7.4.1 Přepínač `differential`

Prvním představeným rozšířením je parametr `differential`, který byl navržen společně s přepínačem `average` za účelem zlepšení přesnosti výpočtu. Slouží k zadání pořadí proměn-

ných, podle kterých bude integrace provedena. Má dvě přípustné hodnoty – „$dxdy$“ a „$dydx$“. V prvním případě bude provedena nejprve integrace ve směru osy x a poté ve směru osy y, jednorozměrné integrály budou tedy počítány s fixní hodnotou proměnné y a následně bude na výsledky aplikovaná numerická metoda podle mezí na ose y. Pokud bude hodnota tohoto přepínače „$dydx$“, bude postup probíhat přesně opačně. Vznikající řezy budou tedy rovnoběžné s osou y a konstantní hodnotu bude mít souřadnice na ose x. Pokud je tento parametr zadán sám, bez přepínače `average`, pak slouží k možnosti porovnat chyby výsledků získaných těmito dvěma způsoby.

7.4.2 Přepínač `average`

Druhým popisovaným doplněním aplikace je parametr `average`, jehož cílem je zlepšení přesnosti výpočtu. Pokud je zapnuta jeho funkce, provede se výpočet dvakrát za sebou, poprvé pro zadané pořadí diferenciálů, poté s opačným. Výsledná hodnota dvojného integrálu je nakonec získána jako aritmetický průměr výsledků těchto dvou výpočtů. K nejvyššímu zpřesnění řešení dochází pro funkce, které mají dynamičtější průběh ve směru jedné osy než ve směru osy druhé a byla by požadována integrace po řadě podle méně dynamické proměnné a poté podle dynamičtější proměnné.

7.4.3 Přepínač `taylor`

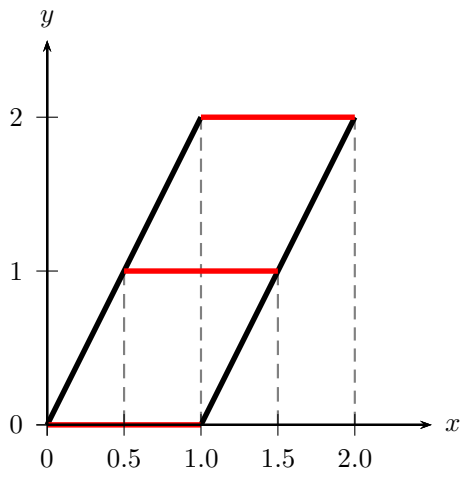
Posledním doimplementovaným parametrem je přepínač `taylor`. S jeho využitím lze zapnout, či vypnout počítání dílčích jednorozměrných integrálů s využitím Taylorovy řady, které je prováděno uvnitř prostředí TKSL/C. Implicitně je zapnuto, jelikož dosahuje vyšší přesnosti. Pokud je vypnuto, používá se na výpočet v obou dimenzích zvolená numerická metoda, tedy obdélníková, lichoběžníková, nebo Simpsonova, vztahy na jejichž výpočet jsou také součástí implementace. Tento přepínač byl navržen především za účelem porovnání přesnosti a rychlosti výpočtů výše zmíněných numerických metod a také pro porovnání jejich výsledků s hodnotami získanými s využitím TKSL/C.

7.5 Plánovaná rozšíření aplikace

Výše bylo popsáno, jak byla aplikace navržena a jakým způsobem je implementována. V průběhu realizace této aplikace však vyplynulo i pár jejích omezení, v následujícím textu bude tedy navrženo několik cest, kterými by se aplikace mohla ubírat do budoucna, a o jaké další možnosti výpočtu ji lze rozšířit. Ke každé z nich bude uveden i návrh, jak by bylo možné dané rozšíření řešit.

7.5.1 Oblast integrace

Prvním z možných vylepšení je odstranění omezení na obdélníkovou oblast integrace. Snadno lze implementovat například rovnoběžníkovou oblast, a to tak, že se v každém řezu numerické integrace provede substituce t za $t + p$, kde p značí souřadnici času, od které se bude v aktuálním řezu integrovat. Pokud bychom uvažovali rovnoběžníkovou oblast znázorněnou na obrázku 7.1 a chtěli vypočítat tři jednorozměrné integrály se zafixovanou souřadnicí y, které jsou na obrázku znázorněny červenou barvou, substituce za proměnnou t by byly následující: $t + 0$, $t + 0,5$ a $t + 1$.



Obrázek 7.1: Ukázka řešení rovnoběžníkové oblasti

7.5.2 Paralelizace výpočtu

Dalším navrhovaným zdokonalením je implementace paralelizace výpočtu, které by pro velký počet dílčích integrálů vedlo na značné zrychlení celého běhu programu. Jelikož lze integrál vypočítat jako obsah plochy pod grafem funkce, bylo by možné integrovanou oblast rozdělit na n částí (pro $n \in \mathbb{N}; n \geq 2$). Dalo by se tedy vytvořit n vstupních souborů s diferenciálními rovnicemi pro prostředí TKSL/C, které by se poté řešily paralelně na n procesorech. Po skončení výpočtu všech dílčích integrálů by byla získána jejich suma, jenž je rovna výsledku řešeného dvojrozměrného integrálu.

7.5.3 Počet dimenzí integrálu

Posledním navrhovaným rozšířením je zobecnění výpočtu pro k-dimenzionální integrál (kde $k \in \mathbb{N}; k \geq 1$). Zde by bylo potřeba vyřešit způsob zadávání hranice oblasti integrace, aby byl pokud možno jednoduchý a jednoznačný. Po odstranění omezení na dvojrozměrné integrační oblasti by mohlo být zajímavé mimo jiné srovnání přesnosti a rychlosti výpočtu s metodou Monte Carlo, která je vhodná právě pro integrály přes velký počet dimenzí.

Kapitola 8

Testování

Implementovaná aplikace byla v průběhu vývoje testována na sadě určitých integrálů, která vznikla již ve fázi návrhu a představena bude níže v této kapitole. Hlavním cílem provedeného testování aplikace bylo nejprve srovnat přesnosti jednotlivých implementovaných metod výpočtu, díky čemuž byla také zjištěna ta metoda, která poskytuje výsledky s nejnižší chybou a bude tedy použita pro další experimenty. Druhou fází testování poté bylo hodnocení implementace v porovnání s dalšími existujícími systémy – MATLABem a Maplem. Na závěr byl zkoumán vliv různých hodnot přepínačů na velikost chyby výpočtu. Zaměříme se na parametry `average`, `stepsize` a `width`, které mohou mít významný vliv na přesnost řešení.

8.1 Testovací sada funkcí

Nyní bude uvedena množina funkcí pro testování aplikace zmíněná v sekci 6.3. Byla sestavena tak, abychom byli schopni určit hodnotu přesného analytického řešení jejich určitých dvojných integrálů pro porovnání přesnosti výsledků a také aby v ní byli zástupci různých typů funkcí dvou proměnných běžně používaných a řešených v praktických úlohách z různých technických oblastí – lineární, goniometrické, exponenciální, racionální lomené atd. Všechny funkce i s hodnotou analytického řešení jejich určitého dvojného integrálu na oblasti $\langle 0, 1\rangle \times \langle 0, 1\rangle$ jsou uvedeny v tabulce 8.1.

Funkce	**Analytické řešení**
$x + 10 \cdot y$	5,5
$x^2 + y$	$\frac{5}{6}$
e^{x+y}	$(e-1)^2$
$\sin(x+y)$	$2 \cdot \sin(1) - \sin(2)$
$x^3 + x^2 + y^3 + x \cdot y^2$	1
$\frac{x}{y+1}$	$\frac{\ln(2)}{2}$

Tabulka 8.1: Funkce použité pro testování aplikace `mis` a analytické řešení jejich dvojných integrálů

První funkce je funkcí lineární. Všechny tři implementované numerické metody poskytují přesnou aproximaci lineárních funkcí, díky tomu by řešení této funkce nemělo být postiženo

žádnou chybou. Další funkce je kvadratická ve směru osy x. Takové funkce přesně aproximuje pouze metoda Simpsonova, ta by měla i v tomto případě poskytovat exaktní řešení. Pro výpočty s využitím TKSL/C se očekávají přesnější výsledky při pořadí diferenciálů $dxdy$, kdy ve směru osy x je integrál řešen v prostředí TKSL/C a numerické metody jsou aplikovány na ose y. Dále bude aplikace testována na funkci exponenciální, goniometrické, polynomiální a lomené. Ty jsou pro aproximaci složitější a jejich výsledky budou postiženy větší či menší odchylkou od analytického řešení.

8.2 Porovnání přesnosti výpočtu implementovaných metod

Obsahem první fáze testování je porovnání přesnosti metod, které jsou součástí implementace. Jedná se o metodu obdélníkovou, lichoběžníkovou a Simpsonovu, jenž jsou všechny implementovány ve dvou variantách – s využitím TKSL/C v jedné z dimenzí a s aplikací pouze výše zmíněných metod v obou dimenzích. Pro porovnání byly využity funkce z předchozí sekce a aplikace byla spouštěna s parametry `-p 100 -e 1e-70 -w 128 -s 1 -a 0 -d [dxdy|dydx] -t [1|0] -m [rectangle|trapezoidal|simpson]`, je tedy využit pouze jeden dílčí interval. Odchylky výsledků výpočtů integrálů daných funkcí řešených jednotlivými metodami na oblasti $\langle 0,1\rangle \times \langle 0,1\rangle$ jsou uvedeny v tabulkách 8.2a a 8.2b, v nichž každý řádek odpovídá jedné funkci v takovém pořadí, v jakém jsou uvedeny v tabulce 8.1. Zvýrazněny jsou hodnoty, které představují pro každou funkci nejmenší získanou chybu.

S využitím TKSL/C			**Bez použití TKSL/C**		
Obdélníková	**Lichoběžníková**	**Simpsonova**	**Obdélníková**	**Lichoběžníková**	**Simpsonova**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	$2,083 \cdot 10^{-2}$	$1,667 \cdot 10^{-3}$	**0**
$3,053 \cdot 10^{-2}$	$2,42 \cdot 10^{-1}$	$\mathbf{9,954 \cdot 10^{-4}}$	$6,075 \cdot 10^{-2}$	$5,039 \cdot 10^{-2}$	$1,991 \cdot 10^{-3}$
$8,118 \cdot 10^{-3}$	$6,557 \cdot 10^{-2}$	$\mathbf{2,768 \cdot 10^{-4}}$	$1,632 \cdot 10^{-2}$	$1,256 \cdot 10^{-3}$	$5,538 \cdot 10^{-4}$
$4,167 \cdot 10^{-2}$	$3,333 \cdot 10^{-1}$	**0**	$9,375 \cdot 10^{-2}$	$7,5 \cdot 10^{-3}$	**0**
$3,716 \cdot 10^{-3}$	$2,843 \cdot 10^{-2}$	$\mathbf{6,486 \cdot 10^{-4}}$	$3,716 \cdot 10^{-3}$	$2,843 \cdot 10^{-3}$	$\mathbf{6,486 \cdot 10^{-4}}$

(a) Pro parametr `differential` roven *dxdy*

S využitím TKSL/C			**Bez použití TKSL/C**		
Obdélníková	**Lichoběžníková**	**Simpsonova**	**Obdélníková**	**Lichoběžníková**	**Simpsonova**
0	**0**	**0**	**0**	**0**	**0**
$2,083 \cdot 10^{-2}$	$1,667 \cdot 10^{-1}$	$\approx \mathbf{0}$	$2,083 \cdot 10^{-2}$	$1,667 \cdot 10^{-3}$	**0**
$3,053 \cdot 10^{-2}$	$2,42 \cdot 10^{-1}$	$\mathbf{9,954 \cdot 10^{-4}}$	$6,075 \cdot 10^{-2}$	$5,039 \cdot 10^{-2}$	$1,991 \cdot 10^{-3}$
$8,118 \cdot 10^{-3}$	$6,557 \cdot 10^{-2}$	$\mathbf{2,768 \cdot 10^{-4}}$	$1,632 \cdot 10^{-2}$	$1,256 \cdot 10^{-3}$	$5,538 \cdot 10^{-4}$
$5,208 \cdot 10^{-2}$	$4,167 \cdot 10^{-1}$	$4,167 \cdot 10^{-6}$	$9,375 \cdot 10^{-2}$	$7,5 \cdot 10^{-3}$	**0**
$\approx \mathbf{0}$	$\approx \mathbf{0}$	$\approx \mathbf{0}$	$3,716 \cdot 10^{-3}$	$2,843 \cdot 10^{-3}$	$6,486 \cdot 10^{-4}$

(b) Pro parametr `differential` roven *dydx*

Tabulka 8.2: Chyba řešení určitého dvojného integrálu podle použité metody výpočtu

V tabulce se vyskytuje několik hodnot, které byly aproximovány na 0. Jsou to čísla řádu 10^{-17} a menšího. Za předpokladu, že délka slov víceslovní aritmetiky použitá pro předchozí výpočty poskytuje přesnost $1 \cdot 10^{-17}$, můžeme chyby menší než je tato hodnota považovat za nulovou odchylku výpočtu, jelikož se jedná pouze o chybu zaokrouhlovací.

Při porovnávání tří implementovaných vztahů pro numerickou integraci dosahuje při počtu jednoho řezu nejvyšší chyby metoda lichoběžníková. O něco přesnější výsledky poskytuje metoda obdélníková a nejlepší aproximace vytváří metoda Simpsonova. Po zkoumání, zda je řešení postiženo menší chybou při použití TKSL/C, nebo bez něj, bychom vyhodnotili jako přesnější výpočet s využitím TKSL/C. Výjimku tvoří pouze dva případy, kdy samotná Simpsonova metoda dosáhla chyby nižší než její kombinace s TKSL/C. Pro další experimenty s aplikací `mis` bude tedy použit výpočet spojením TKSL/C a Simpsonovy metody.

V tabulkách 8.2a a 8.2b stojí také za povšimnutí, že numerické metody bez využití TKSL/C poskytují výsledky nezávisle na hodnotě parametru `differential`, získali jsme tedy stejné velikosti chyb při spuštění s hodnotou *dxdy* jako při zadání *dydx*. To je dáno tím, že tyto numerické metody využívají k výpočtu pouze funkční hodnoty v ekvidistantně rozložených bodech. Pořadí integrovaných proměnných nemění polohu těchto uzlových bodů, zůstává tedy stejná i hodnota řešení.

8.3 Srovnání aplikace `mis` s jinými aplikacemi

Ve druhé části testování se zaměříme na porovnání přesnosti výsledků získaných z aplikace `mis` s hodnotami poskytnutými matematickými prostředími MATLAB a Maple. Konkrétně se bude jednat o metody kombinující TKSL/C s lichoběžníkovou a Simpsonovou metodou v aplikaci `mis` a funkcí implementujících lichoběžníkovou a Simpsonovu metodu v obou komerčních prostředích (funkce `trapezoid` a `simpson` v prostředí Maple a funkce `trapz` a `simpsons` v MATLABu). Pro všechny výpočty byl zvolen jeden dílčí interval a v aplikaci `mis` byly ostatní parametry nastaveny na stejné hodnoty jako v předchozí sekci. Přehled výsledků provedených experimentů je uveden v tabulce 8.3, kde je pro každou funkci a metodu výpočtu tučně vyznačena nejnižší odchylka. Příklady byly řešeny se zadáním obou možných pořadí diferenciálů. V tabulce je vždy uvedena nižší z těchto dvou chyb a také to pořadí diferenciálů, které pro danou funkci umožnilo získat výsledek s nižší chybou, případně pomlčka, pokud si byly tyto dva výsledky rovny.

mis			**Maple**		**MATLAB**
Pořadí integrace	**TKSL/C + lichoběžníková metoda**	**TKSL/C + Simpsonova metoda**	**Lichoběžníková**	**Simpsonova**	**Lichoběžníková/ Simpsonova**
dxdy	**0**	**0**	**0**	**0**	**0**
dxdy	**0**	**0**	$1,667 \cdot 10^{-1}$	**0**	$1,667 \cdot 10^{-1}$
–	$\mathbf{2,42 \cdot 10^{-1}}$	$\mathbf{9,954 \cdot 10^{-4}}$	$5,039 \cdot 10^{-1}$	$1,991 \cdot 10^{-3}$	$5,039 \cdot 10^{-1}$
–	$\mathbf{6,557 \cdot 10^{-2}}$	$\mathbf{2,768 \cdot 10^{-4}}$	$1,226 \cdot 10^{-1}$	$5,538 \cdot 10^{-4}$	$1,255 \cdot 10^{-1}$
dxdy	$\mathbf{3,333 \cdot 10^{-1}}$	**0**	$7,5 \cdot 10^{-1}$	**0**	$7,5 \cdot 10^{-1}$
dydx	$\approx \mathbf{0}$	$\approx \mathbf{0}$	$2,843 \cdot 10^{-2}$	$6,486 \cdot 10^{-4}$	$2,843 \cdot 10^{-2}$

Tabulka 8.3: Srovnání chyby výpočtu lichoběžníkové a Simpsonovy metody aplikace `mis` s prostředími MATLAB a Maple

8.4 Vliv hodnot přepínačů na velikost chyby výpočtu

Poslední část testování bude věnována prozkoumání vlivu hodnot parametrů `average`, `stepsize` a `width` na přesnost výpočtu. Právě tyto přepínače ovlivňují velikost chyby výsledků, jelikož zodpovídají za zaokrouhlovací odchylky, nebo za nepřesnosti způsobené nedokonalostmi numerických metod.

8.4.1 Parametr `average`

Přepínač `average` popsaný v sekci 7.4.2 byl navržen za účelem zvýšení přesnosti výpočtu. Bylo tedy provedeno několik experimentů spouštěných příkazy s parametry `-p 100 -e 1e-70 -s 1 -w 128 -a [0|1] -d [dxdy|dydx] -t 1 -m [rectangle|trapezoidal|simpson]`. Na základě získaných hodnot vznikla tabulka 8.4 zobrazující odchylky řešení určitého dvojného integrálu dané funkce pro obě možná pořadí diferenciálů (tedy $dxdy$ a $dydx$) a dále chyby výsledků získaných při použití parametru `average`. Všechny uvedené metody byly použity v kombinaci s prostředím TKSL/C.

Obdélníková			**Lichoběžníková**			**Simpsonova**		
dxdy	***dydx***	`average`	***dxdy***	***dydx***	`average`	***dxdy***	***dydx***	`average`
0	0	0	0	0	0	0	0	0
0	$2{,}1\cdot10^{-2}$	$1\cdot10^{-2}$	0	$1{,}7\cdot10^{-1}$	$8{,}3\cdot10^{-2}$	0	≈ 0	≈ 0
$3{,}1\cdot10^{-2}$	$3{,}1\cdot10^{-2}$	$3{,}1\cdot10^{-2}$	$2{,}4\cdot10^{-1}$	$2{,}4\cdot10^{-1}$	$2{,}4\cdot10^{-1}$	$1\cdot10^{-3}$	$1\cdot10^{-3}$	$1\cdot10^{-3}$
$8{,}1\cdot10^{-3}$	$8{,}1\cdot10^{-3}$	$8{,}1\cdot10^{-3}$	$6{,}6\cdot10^{-2}$	$6{,}6\cdot10^{-2}$	$6{,}6\cdot10^{-2}$	$2{,}8\cdot10^{-4}$	$2{,}8\cdot10^{-4}$	$2{,}8\cdot10^{-4}$
$4{,}2\cdot10^{-2}$	$5{,}2\cdot10^{-2}$	$4{,}7\cdot10^{-2}$	$3{,}3\cdot10^{-1}$	$4{,}2\cdot10^{-1}$	$3{,}7\cdot10^{-1}$	0	$4{,}2\cdot10^{-6}$	$2{,}1\cdot10^{-6}$
$3{,}7\cdot10^{-3}$	≈ 0	$1{,}9\cdot10^{-3}$	$2{,}8\cdot10^{-2}$	≈ 0	$1{,}4\cdot10^{-2}$	$6{,}5\cdot10^{-4}$	≈ 0	$3{,}2\cdot10^{-4}$

Tabulka 8.4: Chyby řešení určitého dvojného integrálu podle použité metody při pořadí integrace $dxdy$, $dydx$ a s využitím parametru `average`

Výsledky první funkce nejsou postiženy žádnou chybou, nastavení parametru `average` tedy v tomto případě nijak nezměnilo přesnost řešení. Žádný vliv na velikost chyby neměl přepínač ani na funkce e^{x+y} a $\sin(x+y)$, jelikož se jedná o funkce symetrické podle přímky $x = y$. U funkcí x^2+y a $x^3+x^2+y^3+x\cdot y^2$ byl nejpřesnější výpočet s pořadím diferenciálů $dxdy$, méně přesný s parametrem `average` a největší odchylku měl výsledek s diferenciály $dydx$. Přesně naopak tomu bylo u funkce $\frac{x}{y+1}$.

Pokud by byl řešen příklad, jehož analytické řešení nám není známo, nebyli bychom schopni určit, jaké pořadí diferenciálů je v takovém případě přesnější. Spuštění výpočtu se zapnutím parametru `average` by v takové situaci mohlo napomoci snížení chyby.

8.4.2 Parametr `stepsize`

Parametr `stepsize` udává počet dílčích intervalů, na které bude v každé z dimenzí rozdělena integrovaná oblast. Je tedy zřejmé, že hodnota tohoto přepínače má velmi významný vliv na přesnost výpočtu.

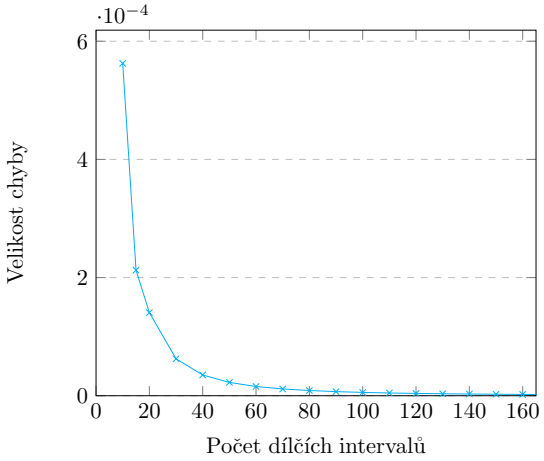
Nejprve byly provedeny experimenty za účelem zjištění, pro jaký počet dílčích intervalů lze u jednotlivých metod dosáhnout jednoduché přesnosti (na 6 desetinných míst). Výpočty

všech funkcí z testovací sady tedy byly spouštěny s parametry `-p 100 -e 1e-70 -w 128 -d dxdy -a 0 -t [1|0] -m [rectangle|trapezoidal|simpson]` a vzrůstající hodnotou přepínače `stepsize`, až dokud nebyla dosažena přesnost $1 \cdot 10^{-6}$. Počet dílčích intervalů potřebný k dosažení minimálně takto přesného výpočtu je pro jednotlivé metody a funkce uveden v tabulce 8.5. Každé funkci přísluší jeden řádek a jsou opět uvedeny v takovém pořadí, v jakém byly představeny v tabulce 8.1. Zvýrazněné hodnoty představují pro každou funkci tu metodu, která při vzrůstajícím počtu dílčích intervalů jako první dosáhla požadované přesnosti.

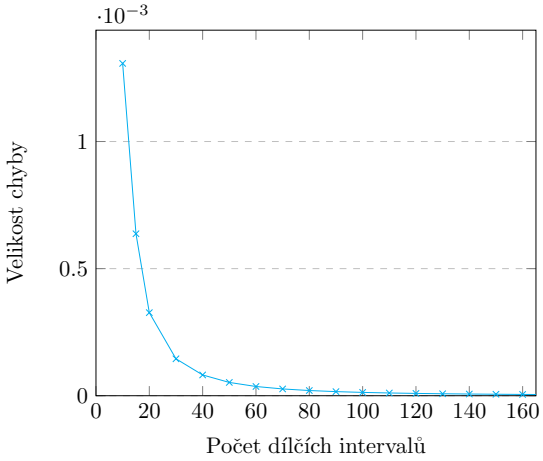
S využitím TKSL/C			**Bez použití TKSL/C**		
Obdélníková	**Lichoběžníková**	**Simpsonova**	**Obdélníková**	**Lichoběžníková**	**Simpsonova**
1	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	1582	1582	**1**
528	665	**7**	747	940	8
266	345	**5**	376	488	6
409	578	**1**	613	867	**1**
417	1794	**18**	417	1794	**18**

Tabulka 8.5: Počet dílčích intervalů potřebný k dosažení jednoduché přesnosti ($1 \cdot 10^{-6}$)

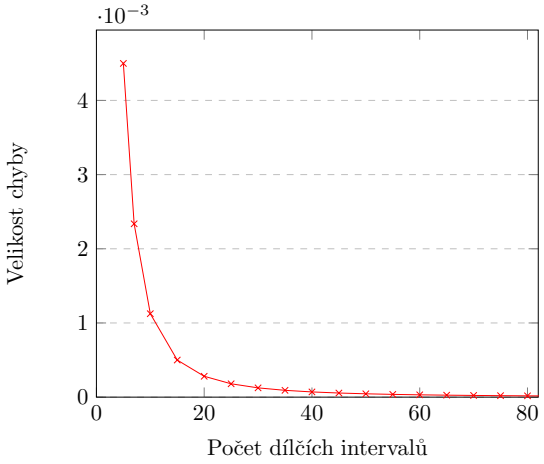
Závislost vstupní hodnoty parametru `stepsize` na velikosti chyby výsledku byla zkoumána pro výpočty se vstupními parametry `-p 100 -e 1e-70 -w 128 -d dxdy -a 0 -t [1|0] -m [rectangle|trapezoidal|simpson]`. Grafy těchto závislostí pro jednotlivé metody lze nalézt v obrázku 8.1. Každá jedna hodnota v grafu představuje aritmetický průměr chyb výsledků všech šesti funkcí z testované sady při daných parametrech výpočtu.



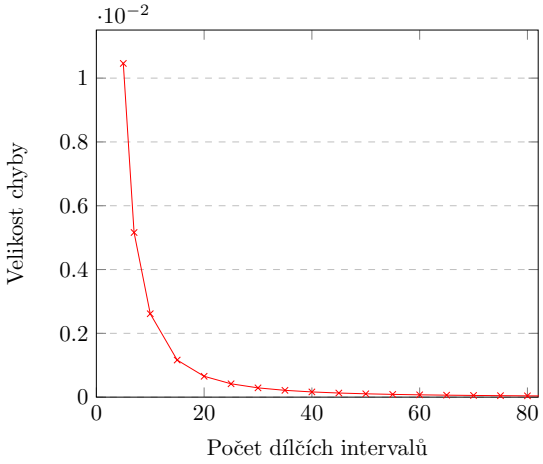
(a) Taylorova řada s obdélníkovou metodou



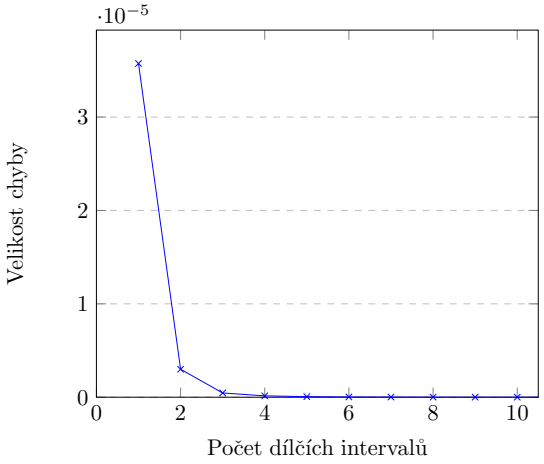
(b) Obdélníková metoda v obou dimenzích



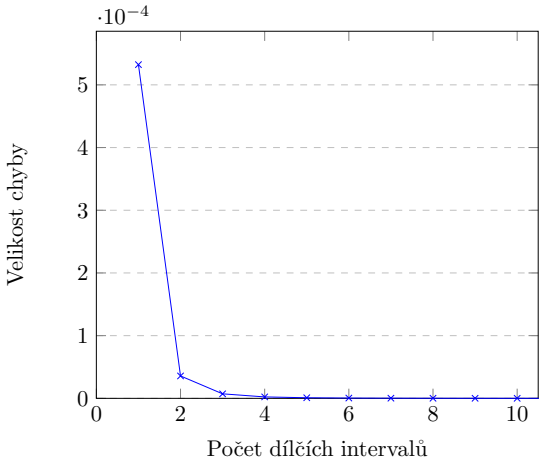
(c) Taylorova řada s lichoběžníkovou metodou



(d) Lichoběžníková metoda v obou dimenzích



(e) Taylorova řada se Simpsonovou metodou



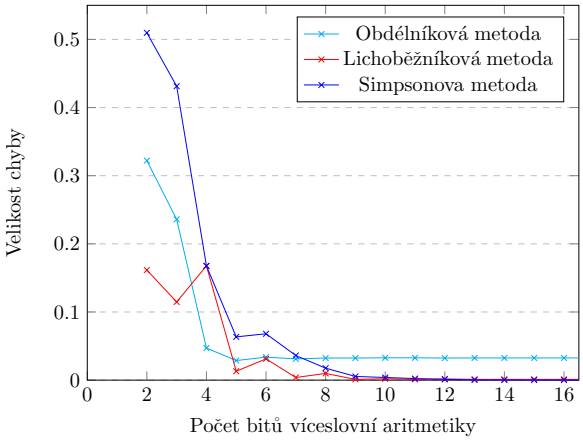
(f) Simpsonova metoda v obou dimenzích

Obrázek 8.1: Vliv počtu dílčích intervalů výpočtu na velikost chyby výsledku

Z grafů je patrné, že nejvíce citlivá na počet podintervalů je metoda obdélníková, naopak Simpsonova metoda poskytuje velmi přesné výsledky i pro jednociferný počet dílčích intervalů.

8.4.3 Parametr `width`

Počet bitů víceslovní aritmetiky je možné nastavit přepínačem `width`. Ten ovlivňuje přesnost výpočtu především velikostí zaokrouhlovacích chyb, pro velmi malé hodnoty bude tedy výsledek ovlivňovat značně, dále však jeho význam prudce klesá a od určité hodnoty je chyba téměř konstantní. Závislost velikosti odchylky řešení na počtu bitů víceslovní aritmetiky je pro jednotlivé metody uvedena v grafu 8.2. Zobrazené hodnoty byly získány spouštěním výpočtů šesti testovaných funkcí s parametry `-p 100 -e 1e-70 -w 128 -a 0 -d dxdy -t 1 -m [rectangle|trapezoidal|simpson]` a odchylky jednotlivých funkcí byly opět zprůměrovány.



Obrázek 8.2: Vliv délky slova víceslovní aritmetiky na velikost chyby výsledku výpočtu kombinací TKSL/C s numerickou metodou

Z grafu vyplývá, že nízkým počtem bitů je nejvíce ovlivněná Simpsonova metoda, nejméně postižené výsledky poskytuje metoda lichoběžníková. Od hodnoty 12 nedochází k výraznému ovlivnění přesnosti výsledků, pro ne vysoce přesné výpočty (požadující přesnost $1 \cdot 10^{-15}$ a menší) by tedy měl být dostačující rozsah standardních datových typů.

Kapitola 9

Závěr

V této bakalářské práci byly připomenuty definice z oblasti integrálního a diferenciálního počtu. Následně byly uvedeny dva přístupy k řešení integrálů – analytická a numerická integrace a jejich klady i zápory. Poslední kapitola věnovaná teorii se zabývala nepřesnostmi souvisejícími se způsobem uložení desetinných čísel v počítači. Praktická část této práce byla uvedena návrhem implementované aplikace `mis` a popisem naprogramované numerické metody. Poté byl proveden náhled do implementace vzniklé aplikace a na závěr byla aplikace několika způsoby testována.

Cílem práce bylo navrhnout numerickou metodu pracující na principu vytváření řezů skrze funkci dvou proměnných. Obsahy takto vzniklých řezů byly poté počítány pomocí převodu určitých integrálů na diferenciální rovnice řešené s využitím Taylorovy řady v prostředí TKSL/C. Na hodnoty těchto dílčích integrálů byla poté aplikována některá z představených metod numerické integrace – obdélníkového, lichoběžníkového, či Simpsonova pravidla.

Jako součást práce vznikl výsledný program `mis`, který je implementován jako konzolová aplikace v jazyce Python 3. Umožňuje použití víceslovní aritmetiky, pro jejíž realizaci využívá knihovnu GMPY2. Ovládání parametrů výpočtu probíhá pomocí udání hodnot přepínačů zadaných při spuštění aplikace. Výsledky poskytované tímto programem byly srovnány s hodnotami získanými ze dvou komerčních matematických prostředí MATLAB a Maple.

Výsledná aplikace poskytuje možnost řešení určitých dvojných integrálů až šesti různými metodami. Je také možné řídit přesnost výpočtu změnou hodnot parametrů ovlivňujících například počet dílčích integrálů, kolik bitů bude použito pro víceslovní aritmetiku, či měnit pořadí diferenciálů a dosáhnout tak výsledku s nižší chybou. Obsahem této práce je také pár návrhů, o jakou funkcionalitu by bylo možné aplikaci rozšířit a jakým směrem by se mohl její vývoj ubírat do budoucna.

Dále lze například doplnit implementaci o způsob řešení integrálů i nad jinými než obdélníkovými oblastmi, nebo zobecnit celý výpočet, aby jej bylo možné využít pro řešení integrálů více než dvou proměnných. Mezi další možnosti rozšíření můžeme zařadit doplnění implementace o další kvadraturní formule nebo zjišťování gradientu funkce a následného použití neuniformní diskretizace oblasti integrace.

Literatura

[1] Daněček, J.; Dlouhý, O.; Přibyl, O.: *Matematika II. Modul 1 [BA02-MO1] : Dvojný a trojný integrál*. Brno: Vysoké učení technické v Brně, Fakulta stavební, 2004.

[2] *Solution of the Diffusion Equation by Finite Differences*. Září 2003, [Online; navštíveno 10.03.2017].
URL https://me.ucsb.edu/~moehlis/APC591/tutorials/tutorial5/node3.html

[3] Fajmon, B.; Hlavičková, I.; Novák, M.; aj.: *Numerická matematika a pravděpodobnost*. [Online; navštíveno 07.02.2017].
URL http://matika.umat.feec.vutbr.cz/inovace/texty/INM/CZ/INM_plna_verze_CZ.pdf

[4] *Newton-Leibniz formula - Encyclopedia of Mathematics*. [Online; navštíveno 27.01.2017].
URL https://www.encyclopediaofmath.org/index.php/Newton-Leibniz_formula

[5] Kalas, J.; Kuben, J.: *Integrální počet funkcí více proměnných*. Brno: Masarykova univerzita, Přírodovědecká fakulta, 2009, ISBN 978-80-210-4975-8.

[6] Krupková, V.; Fuchs, P.: *Matematická analýza pro FIT*. [Online; navštíveno 25.01.2017].
URL http://www.umat.feec.vutbr.cz/~krupkova/IMA2014.pdf

[7] Kubíček, M.; Dubcová, M.; Janovská, D.: *Numerické metody a algoritmy*. Praha: VŠCHT Praha, druhé vydání, 2005, ISBN 80-7080-558-7.

[8] Kunovský, J.: *Modern Taylor Series Method*. FEI-VUT Brno, 1994, habilitační práce.

[9] Kunovský, J.: *Modern Taylor Series Method. Scientific Conference on Informatics, 2015 IEEE 13th*, 2015, doi:10.1109/Informatics.2015.7377798.

[10] Macur, J.: *Dynamické systémy*. Říjen 2000, [Online; navštíveno 10.02.2017].
URL http://www.fce.vutbr.cz/studium/materialy/Dynsys/

[11] *MATEMATIKA online*. [Online; navštíveno 09.02.2017].
URL http://mathonline.fme.vutbr.cz

[12] O'Connor, J. J.; Robertson, E. F.: *Eudoxus biography*. [Online; navštíveno 24.01.2017].
URL http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Eudoxus.html

[13] *754-2008 – IEEE Standard for Floating-Point Arithmetic*. Srpen 2008, [Online; navštíveno 14.03.2017].
URL http://ieeexplore.ieee.org/servlet/opac?punumber=4610933

[14] Peringer, P.: *Modelování a simulace*. Prosinec 2012, [Online; navštíveno 09.02.2017].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIMS-IT%2Ftexts%2Fopora-ims.pdf&cid=11453

[15] *High Performance Computing*. Březen 2005, [Online; navštíveno 20.03.2017].
URL http://www.fit.vutbr.cz/~kunovsky/TKSL/index.html.cs

[16] *MathWorks - Makers of MATLAB and Simulink*. Březen 2017, [Online; navštíveno 19.03.2017].
URL https://www.mathworks.com

[17] *Maple 2016 – Technical Computing Software for Engineers, Mathematicians, Scientists, Instructors and Students – Maplesoft*. Březen 2017, [Online; navštíveno 21.03.2017].
URL https://www.maplesoft.com/products/maple

[18] *gmpy2 2.0.8 : Python Package Index*. Březen 2017, [Online; navštíveno 19.03.2017].
URL https://pypi.python.org/pypi/gmpy2

[19] *Error in Euler's Method*. Leden 2004, [Online; navštíveno 19.02.2017].
URL http://www.math.unl.edu/~gledder1/Math447/EulerError

[20] Čermák, L.; Hlavička, R.: *Numerické metody*. Brno: Akademické nakladatelství CERM, druhé vydání, 2008, ISBN 978-80-214-3752-4.