# Statistická metoda Monte Carlo

Monte Carlo statistical method

Bc. Jakub Kupčík

Diplomová práce
2009



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Jakub KUPČÍK** |
| Studijní program: | **N 3902 Inženýrská informatika** |
| Studijní obor: | **Informační technologie** |
| Téma práce: | **Statistická simulační metoda Monte Carlo.** |

Zásady pro vypracování:

1. **Nastudujte možnosti využití statistické metody Monte Carlo při simulacích reálných procesů a zpracujte literární rešerši na dané téma.**
2. **Navrhněte a naprogramujte některé experimenty využívající metody Monte Carlo. Zejména se jedná o výpočet integrálů, určení hodnoty Ludolfova čísla, řešení systému lineárních algebraických rovnic metodou Monte Carlo. Při realizaci použijte prostředí WebMathematica.**
3. **Nastudujte a naprogramujte postupy vedoucí ke zpřesnění výpočtů.**
4. **Proveďte srovnání přesnosti dosažených výsledků.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:


1. **FABIÁN, F, KLUIBER, Z. Metoda Monte Carlo a možnosti jejího uplatnění. Praha : Prospektum s.r.o., 1998. 148 s. ISBN 80-7175-058-1.**
2. **HRABEC, Tomáš. Řešení systému lineárních algebraických rovnic metodou Monte Carlo. [s.l.], 2005. 40 s. Bakalářská práce.**
3. **DANÍČEK, Ladislav. Využití geometrické pravděpodobnosti při experimentálním ověření Ludolfova čísla. [s.l.], 2005. 37 s. Bakalářská práce.**
4. **Metoda Monte Carlo [online]. [2005] [cit. 2009-02-20]. Dostupný z WWW: <http://artemis.osu.cz/pmfch/lekce10.pdf>.**
5. **VIRIUS, M. Aplikace matematické statistiky: metoda Monte Carlo. [s.l.] : [s.n.], 1998. 210 s. ISBN 80-01-01779-6.**


Vedoucí diplomové práce: **Ing. Bronislav Chramcov, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
1

prof. Ing. Vladimír Vašek, CSc.
*děkan*

doc. Ing. Ivan Zelinka, Ph.D.
*ředitel ústavu*

## ABSTRAKT

Diplomová práce se zabývá statistickou simulační metodou Monte Carlo. Konkrétně se věnuje především jejím využitím v různých aplikacích. Teoretická část obsahuje obecný popis metody Monte Carlo v konkrétních aplikacích a dále pak využití internetových pomůcek při výuce technických předmětů, především prostředí webMathematica. Hlavní část práce je praktická. Ta se skládá jednak z tvorby programů v prostředí webMathematica, které slouží jako praktické ukázky využití metody Monte Carlo. Další částí je experimentální srovnání základní metody s různými úpravami algoritmu pro zpřesnění.



Klíčová slova: Monte Carlo, WebMathematica, pseudonáhodná, integrál, Ludolfovo číslo, rovnice



## ABSTRACT

The graduation theses is interested in the statistic simulation method Monte Carlo. Concretely is mainly interested in the utilization in the the various aplications. A theoretic part includes a general describe of the method Monte Carlo in the specific aplications and then the utilization of the internet aids during the education of the technical subjects, mainly the environment of webMathematica. The main part of the thesis is practical. This consists of the making of programs in the environment of webMathematica, which serves as a practical illustration of the utilization of the method Monte Carlo. The other part is experimental comparison of the base method with a various modifications of an algorithm for the specification.



Keywords: Monte Carlo, WebMathematica, Ludolf's number, integral, linear equation, pseudo-random

Děkuji panu Ing. Bc. Bronislavu Chramcovovi, Ph.D za pomoc při tvorbě této práce a za poskytnutí potřebných materiálů.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval.

V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

# OBSAH

**ÚVOD** ..... **9**

**I TEORETICKÁ ČÁST** ..... **10**

**1 METODA MONTE CARLO** ..... **11**

1.1 PRINCIP METODY ..... 11

1.2 URČENÍ HODNOTY LUDOLFOVA ČÍSLA ..... 13

1.3 VÝPOČET JEDNOROZMĚRNÝCH URČITÝCH INTEGRÁLŮ METODOU MONTE CARLO ..... 14

1.3.1 Geometrická metoda ..... 15
1.3.2 Metoda odhadu střední hodnoty funkce ..... 16
1.3.3 Zpřesňující metody ..... 16

1.4 VÝPOČET VÍCEROZMĚRNÉHO URČITÉHO INTEGRÁLU METODOU MONTE CARLO ..... 17

1.5 ŘEŠENÍ SOUSTAVY LINEÁRNÍCH ROVNIC METODOU MONTE CARLO ..... 19

1.6 VYUŽITÍ METODY MONTE CARLO V EKONOMICE ..... 21

**2 POČÍTAČOVÉ ALGEBRAICKÉ SYSTÉMY** ..... **22**

2.1 WEBMATHEMATICA ..... 23

2.1.1 Výroba vlastních MSP (Mathematica Server Pages) ..... 24

**II PRAKTICKÁ ČÁST** ..... **27**

**3 VYTVOŘENÍ PROGRAMŮ S ALGORITMY METODY MONTE CARLO** ..... **28**

3.1 PROGRAM NA URČENÍ HODNOTY LUDOLFOVA ČÍSLA ..... 28

3.1.1 Princip výpočtu ..... 28
3.1.2 Vzhled a ovládání programu ..... 29

3.2 PROGRAM NA VÝPOČET URČITÉHO INTEGRÁLU ..... 30

3.2.1 Princip výpočtu ..... 30
3.2.2 Vzhled a ovládání programu ..... 32

3.3 PROGRAM NA VÝPOČET DVOJNÉHO INTEGRÁLU ..... 34

3.3.1 Princip výpočtu geometrické metody ..... 34
3.3.2 Vzhled a ovládání geometrické metody ..... 35
3.3.3 Princip výpočtu metodou odhadu střední hodnoty ..... 36
3.3.4 Vzhled a ovládání odhadu střední hodnoty ..... 37

3.4 PROGRAM NA ŘEŠENÍ SOUSTAVY LINEÁRNÍCH ROVNIC METODOU MONTE CARLO ..... 39

3.4.1 Princip výpočtu ..... 39
3.4.2 Vzhled a ovládání programu ..... 40

**4 ZPŘESŇUJÍCÍ ALGORITMY METODY MONTE CARLO** ..... **42**

4.1 POPIS NAPROGRAMOVANÝCH ALGORITMŮ ..... 42

4.1.1 Klasická metoda odhadu střední hodnoty a geometrická metoda ..... 42
4.1.2 Metoda zpřesnění - vymezení hlavní části ..... 42

4.1.3 Metoda zpřesnění - výběr na podintervalech ..... 43
4.1.4 Metoda zpřesnění - symetrizace integrované funkce ..... 44
4.2 TESTOVÁNÍ METOD ..... 45
4.3 METODA ZPŘESNĚNÍ - HLAVNÍ VÝBĚR ..... 49
**ZÁVĚR ..... 52**
**ZÁVĚR V ANGLIČTINĚ ..... 53**
**SEZNAM POUŽITÉ LITERATURY ..... 54**
**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK ..... 55**
**SEZNAM OBRÁZKŮ ..... 56**
**SEZNAM TABULEK ..... 57**
**SEZNAM PŘÍLOH ..... 58**

# ÚVOD

Jak už bylo mnohokrát napsáno, s rozvojem výpočetní techniky se při řešení nejrůznějších problémů stále častěji používá numerických a jiných netradičních metod. Málokterá z nich má však tak bohatou historii jako právě Monte Carlo. Jedná se o statistickou stochastickou metodu využívající modelování náhodných veličin.

První náznak jejího použití pochází již z roku 1777 při formulaci takzvané Buffovy úlohy pro určení hodnoty Ludolfova čísla. Za samotný vznik metody Monte Carlo se považuje její formulace a využití J. von Neumannem a S. Ulamonem při simulaci chování neutronu za 2. světové války.

Se zvyšováním výpočetního výkonu počítačů nacházela metoda stále nová uplatnění. Je až překvapivé, co vše lze spočítat za použití pseudonáhodných čísel. Tato práce by měla posloužit studentům k seznámení se s některými z využití této metody. A jak jinak dnes umožnit přístup k algoritmům co největšímu množství zájemců, než přes webové rozhraní.

Praktická část práce se opírá o možnosti rozhraní webMathematica, které umožňuje snadný přístup k ukázkovým programům z pohodlí domova přes obyčejný prohlížeč. Je možné využívat všech možností prostředí Mathematica a umožnit tak studentům seznámit se s algoritmy, jako právě například Monte Carlo, pomocí interaktivních programů s názornými grafy. Proč předhazovat studentům několik stránek teorie, kterou nemusí pochopit ani po opakovaném přečtení, když názorná ukázka mnohdy zabere podstatně lépe.

A právě to je v podstatě hlavní smysl této práce. Naprogramování experimentů metody Monte Carlo řešila už řada předešlých prací (např. [2], [3], [4], [7]). Ovšem v žádné z nich nejsou programy dostupné právě přes webové rozhraní a většinou je využíváno prostředí Matlab. Navíc většina z nich se věnuje konkrétně některému z problémů, který metoda umožňuje řešit. V této práci je ukázán určitý průřez různými aplikacemi od určení hodnoty Ludolfova čísla až po řešení soustavy lineárních rovnic.

Pro ukázku různorodosti této metody jsou v druhé části praktické části srovnány různé algoritmy pro zpřesnění výpočtu. Při jejich využití je pak možné generovat menší množství pseudonáhodných čísel než u klasických metod při zachování požadované přesnosti.

# I. TEORETICKÁ ČÁST

# 1 METODA MONTE CARLO

Ne náhodou si čtenář vybaví pod názvem metody světoznámé kasino. Samotný název vznikl, když pánové John von Neumann a Stanislav Ulam použili k modelování předpovědi „historie života neutronu“ techniku kola rulety.

Pomocí této metody bylo možné předpovědět trajektorii každého neutronu daného svazku. Simulace pohybu neutronů „putujících“ ve vodě a náhodně se srážejících s atomy vodíku a kyslíku vypadala následovně: je například známo, že náhodný jev – neutron při srážce s atomem vodíku bude atomem pohlcen – nastane průměrně v jednom ze sta možných případů. Pro vytyčení trasy pohybu neutronu se roztočí kolo rulety rozdělené na sto dílků, z nichž pouze jeden je odlišně vyznačený a označuje „pohlcení neutronu atomem vodíku“. Jestliže se kolo rulety zastaví právě na tomto dílku, označuje se to „konec života“ neutronu. V opačné případě se zjistí pomocí jiného kola rulety směr a rychlost neutronu po srážce. Potom pomocí dalšího kola rulety se rovněž náhodně určí, jakou trajektorií proběhne neutron, než nastane další srážka buď s atomem vodíku nebo kyslíku, atd. Simulace „historie života“ neutronu se provádí tak dlouho, dokud buď není pohlcen, nebo dokud neztratí tolik energie, že jeho konečné vylétnutí z nádrže nás přestane zajímat. Přesto, že je takto celý problém formulován již ve značně zjednodušené formě, je celkem jasně zřetelný vysoký stupeň komplikovanosti celého problému. Když provedeme velký počet takových simulací, získáme dříve či později více méně přesnou informaci o procentu neutronů, kterým se podařilo uniknout z nádrže. [1]

Jelikož v celém algoritmu vždy hraje velkou roli náhoda, není přesnost její nejsilnější stránkou. Například u výpočtu integrálů je přesnějších výsledků dosaženo při použití klasických numerických metod. Pro požadovanou přesnost by metodou Monte Carlo byla potřeba většího počtu ohodnocení účelové funkce a tím i větší časové náročnosti. Její přednosti se však projeví právě při řešení složitějších problémů, kdy složitost samotného algoritmu nenarůstá. Proto se v praxi používá při řešení vícenásobných integrálů, kde již klasické metody selhávají.

## 1.1 Princip metody

Všechny algoritmy založené na Metodě Carlo mají jedno společné a to výpočet založený na mnohokrát opakovaných náhodných pokusech (odhadech náhodné veličiny). Takto

můžeme řešit úlohy jak deterministické tak stochastické. Sestrojíme pravděpodobnostní úlohu, která má stejné řešení s úlohou původní.

Odhady hledané veličiny se získávají statistickou cestou a mají tedy pravděpodobnostní charakter. Odhady

$$\theta_1, \theta_2, ...., \theta_n, .... \tag{1}$$

označíme za hledanou hodnotu veličiny $\theta$, které získáme statistickým zpracováním experimentů, které jsme obdrželi jako výsledek mnohokrát opakovaných náhodných pokusů. Požadujeme, aby v tomto případě veličina $\theta_n$, kde n značí počet pokusů, která je tzv. veličinou náhodnou, při $n \to \infty$ konvergovala k hledané hodnotě $\theta$ v pravděpodobnosti. Tím rozumíme splnění vztahu, aby pro libovolně malé $\varepsilon > 0$ platilo

$$\lim_{n \to \infty} P\left(| \theta_n - \theta | < \varepsilon\right) = 1 \tag{2}$$

Výběr odhadu $\theta_n$ je podmíněn typem a zvláštnostmi specifikujícími řešenou úlohu. Samotný výsledek má pak vždy povahu statistického charakteru. Z výše uvedeného vztahu tedy vyplývá, že požadované přesnosti můžeme dosáhnout vždy zvyšováním počtu náhodných pokusů. To však může vést k velké časové náročnosti, proto je vždy potřeba hledat optimalizace algoritmu pro dosažení dostatečné přesnosti při zachování rozumného počtu opakování.

O tom že některé náhodné procesy odpovídají řešením příslušným diferenciálním řešením se vědělo již na začátku minulého století. Například na vztah mezi procesem náhodného bloudění (např. částice v kapalině), upozornili ve svých pracích fyzikové Albert Einstein a Marian Smoluchovský. Ovšem právě hledání diferenciálních rovnic, které by náhodným procesům odpovídaly, se ukázalo být nepraktické. Postupem času se stále víc začal prosazovat model, kdy naopak hledáme pravděpodobnostní model, který by svým řešením odpovídal analytickému řešení. A právě s rozvojem výpočetní techniky se tato metoda ukazuje jako stále užitečnější. Je tedy v rámci tohoto přístupu nutné vždy sestrojit takový pravděpodobnostní model, jehož parametry reprezentují řešení formulované úlohy. Prakticky jsou zajímavé samozřejmě jen ty pravděpodobnostní modely, které připouštějí relativně jednoduchou realizaci a jsou proveditelné na soudobých počítačích, přičemž

dovolují získat odhady neznámých parametrů (hodnoty výsledků úloh) s pokud co možno nejmenší chybou (měřenou např. rozptylem). [1], [4]

Není také potřeba znát přesný matematický popis zkoumaného systému. Tím že používáme pravděpodobnostní model, považujeme systém za „černou skříňku“. Tato okolnost poskytuje možnost využít metodu Monte Carlo pro řešení úloh nejrozmanitějšího typu, včetně úloh logického charakteru.

Tato práce se konkrétně bude zaobírat těmito typy úloh, které jsou řešitelné metodou Monte Carlo:

- Určení hodnoty Ludolfova čísla
- Výpočet jednoduchých určitých integrálů
- Řešení dvojitých integrálů (násobných)
- Řešení systémů lineárních rovnic

K řešení úloh naprosto různé podstaty je možné užít jednoho pravděpodobnostního modelu a naopak pro řešení jedné konkrétní úlohy je možné v rámci metody Monte Carlo vytvořit řadu různých simulačních schémat. I proto je možné metodu použít na velké množství aplikací. Přesnost a efektivnost celého výpočtu metodou Monte Carlo na počítači je určen třemi základními faktory:

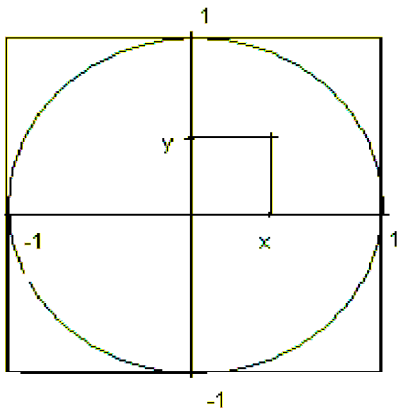
1. kvalitou generátoru náhodných čísel, resp. pseudonáhodných čísel
2. výběrem racionálního algoritmu výpočtu
3. kontrolou přesnosti získaného výsledku. [2]

## 1.2 Určení hodnoty Ludolfova čísla

Výše už bylo zmíněno, že jako první náznak využití podobné metody lze považovat Buffonovu úlohu pro hledání hodnoty Ludolfova čísla. Náhodnou veličinu zde reprezentovala jehla, kterou vrhal na desku, na které byly nakresleny rovnoběžné horizontální přímky. Ty jsou od sebe stejně vzdálené o vzdálenost $d$. Délku jehly označíme $l$ a musí platit $l<d$. Pokus je pak úspěšný, pokud vržená jehla protne jednu z rovnoběžek. Označme poměr úspěšných/počtu pokusů m/n. Hodnotu Ludolfova čísla pak můžeme určit ze vztahu

$$\pi \approx \frac{2dn}{lm} \tag{3}$$

Tento způsob výpočtu je však příliš nepřesný. Lepší je pro výpočet použít jednotkové kružnice. Budeme potřebovat čtverec 2 x 2 cm a v ní vepsanou jednotkovou kružnici.




*Obr. 1. Jednotková kružnice*

Všechny body ležící uvnitř čtverce pak můžeme popsat uspořádanou dvojicí *[x,y]*, kde *x* i *y* ∈ <-1,1>. Pro body ležící uvnitř jednotkové kružnice navíc platí $x^2 + y^2 \leq 1$. Generujeme-li pak náhodně uspořádané dvojice *[x,y]*, podíl těch které leží uvnitř kruhu (m) a podíl těch které leží uvnitř čtverce (všechny generované - n) můžeme vyjádřit jako podíl obsahu kruhu a čtverce.

$$\frac{m}{n} = \frac{S_k}{S_č} = \frac{\pi r^2}{(2r)^2} \tag{4}$$

Odtud již pak není problém vyjádřit $\pi$ jako

$$\pi = \frac{4m}{n} \tag{5}$$

[7]

## 1.3 Výpočet jednorozměrných určitých integrálů metodou Monte Carlo

Výpočet určitého integrálu je jednou z nejčastějších aplikací metody Monte Carlo. Používají se 2 základní algoritmy. Tzv. geometrická metoda a metoda odhadu střední hodnoty integrované funkci. Výhodou první jmenované je výborná názornost při vykreslení

grafu generovaných bodů. Metoda odhadu střední hodnoty však dosahuje pro většinu funkcí lepších výsledků.

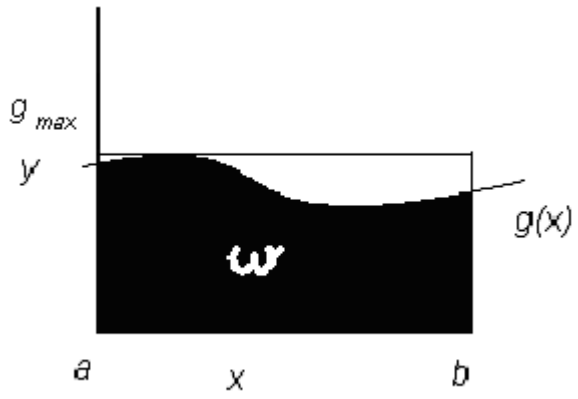
### 1.3.1 Geometrická metoda

Tato metoda vychází z geometrické interpretace integrálu jako plochy pod křivkou na zadaném intervalu.

Odhadujeme – li integrál

$$\int_a^b g(x)dx\,, \tag{6}$$

pokud integrand splňuje podmínku 0 <= *g(x)* a *x* je z intervalu <*a,b*>, jedná se o výpočet plochy, která je omezena křivkou $y = g(x)$ , osou *x* a přímkami $x = a$ a $x = b$. Oblast $\Omega$ je definována nerovnostmi $a <= x <= b$ , 0 <= y <=$g_{max}$ .




*Obr. 2. Geometrická metoda*

Nyní nám už tedy stačí pouze určit plochu $\omega$. Základem metody Monte Carlo je zde náhodné vybírání bodů se souřadnicemi ($x_i$, $y_i$) z plochy $\Omega$. Pravděpodobnost, že bod padne do oblasti $\omega$ se vzhledem na geometrickou interpretaci pravděpodobnosti rovná poměru $\frac{\omega}{\Omega}$.

Pomocí generátoru pseudonáhodných čísel tedy generujeme body se souřadnicemi ($x_i$, $y_i$). Jestliže se při realizaci pokusu objeví bod, patřící do oblasti $\omega$, pak byl proveden zdařilý pokus. Po provedení *n* pokusů se zjistí počet zdařilých pokusů *m* a vypočte se relativní

četnost jevu, že náhodný bod ($x_i$, $y_i$) padne do oblasti $\omega$. Přibližnou hodnotu integrálu pak určíme jako:

$$I' = \frac{m}{n}(b-a)g_{\max}. \tag{7}$$

Ze samotného principu metody vyplývá, že čím více budeme generovat bodů, tím kvalitnější informaci o hodnotě integrálu získáme.

### 1.3.2 Metoda odhadu střední hodnoty funkce

Pokud se nám podaří naleznout střední hodnotu $\overline{g}$ integrované funkce *g(x)* na intervalu <*a*,*b*>, výpočet integrálu je pak snadný. Přibližná hodnota $I'$ je pak

$$I' = (b-a)\overline{g} \tag{8}$$

Metoda Monte Carlo zde spočívá právě v pseudonáhodném hledání střední hodnoty. Generují se hodnoty $x$ z rovnoměrného rozdělení na intervalu <*a*, *b*> a zavede se náhodná veličina $Y = g(x)$, jejíž matematické očekávání $E(Y)$ je rovno průměrné hodnotě funkce $g(x)$ na intervalu <*a*, *b*>. Hodnota $E(Y)$ se odhadne pomocí aritmetického průměru $n$ nezávislých realizací veličiny *Y*. Hodnota integrálu $I'$ se pak vypočítá podle vztahu

$$I' = (b-a)\frac{1}{n}\sum_{i=1}^{n} g(x_i). \tag{9}$$

Tento algoritmus je velmi jednoduchý jak ve svém principu, tak pro naprogramování, proto je u většiny funkcí pro použití vhodnější než geometrická metoda. [1], [4]

### 1.3.3 Zpřesňující metody

Obě metody jsou zatíženy poměrně velkou chybou v porovnání s klasickými numerickými metodami. Obecně lze říci že vždy pomůže zvětšení počtu náhodných pokusů, to však vede ke zvyšování počtu ohodnocení účelové funkce a tím roste časová náročnost algoritmu. Proto se používá několik zpřesňujících postupů. Vždy ale závisí na konkrétní funkci a ne vždy musí nutně dojít ke zlepšení.

- vymezení hlavní části

Předpokladem této metody je, že známe funkci *h(x),* která je blízká hledané funkci *g(x).* Hledání se pak omezuje na zpřesnění rozdílu mezi těmito funkcemi a hledaný integrál můžeme zapsat jako:

$$I = \int_a^b f(x)dx = \int_a^b g(x)dx + \int_a^b h(x)dx \,. \tag{10}$$

- výběr na podintervalech

Funkci definovanou na intervalu <*a*, *b*> můžeme dále rozdělit na více podintervalů. Kvalita zpřesnění závisí právě na tomto rozdělení. Není totiž nutné generovat ve všech podintervalech stejné množství náhodných bodů. Větším podintervalům, na kterých více závisí konečná hodnota integrálu $I'$, je přiřazeno více náhodných bodů, menším méně.

Integrační interval <*a*,*b*> se tedy rozděluje na několik dílčích intervalů $\langle a, c_1\rangle, \langle c_1, c_2\rangle, \ldots, \langle c_k, b\rangle$ a v každém se generuje jiný počet náhodných čísel $x_t$. Pokud se zvolí počty náhodných čísel $n_i$ úměrné jak šířce dílčího intervalu tak i průměrné velikosti integrované funkce na tomto podintervalu, efektivita použitých algoritmů se zvýší. Pro integraci se pak používá vztah

$$I = \int_a^c g(x)dx + \int_{c_1}^{c_2} g(x)dx + \ldots + \int_{c_n}^b g(x)dx \,. \tag{11}$$

- metoda symetrizace integrované funkce

Pokud je integrovaná funkce monotónní na intervalu <*a*, *b*>, je vhodné ji symetrizovat podle vztahu

$$g(x) = 0.5[f(x) + f(a+b-x)] \tag{12}$$

Doba výpočtu se pak prodlouží přibližně dvakrát, rozptyl se však může zmenšit ještě víc. [1]

## 1.4 Výpočet vícerozměrného určitého integrálu metodou Monte Carlo

Řešení vícerozměrných integrálů je u metody Monte Carlo podobné jako řešení integrálů jednorozměrných. Opět máme k dispozici geometrickou metodu a metodu odhadu střední hodnoty. Rozdíl v algoritmu je pouze v generování pseudonáhodných hodnot pro další rozměr, což je velká výhoda oproti klasickým numerickým metodám.

Při výpočtu jednorozměrného integrálu se berou funkční hodnoty v *n* uzlových bodech, při přechodu k *d*-rozměrnému případu vzroste počet uzlových bodů na $n^d$. Použití klasických numerických metod by proto bylo časově velmi náročné.

Jelikož metoda Monte Carlo tuto vlastnost nemá, je její uplatnění v tomto druhu aplikací více než vhodné. Pokud by přesto pro požadovanou přesnost bylo potřeba příliš velkého poštu generovaných bodů, je možné použít i výše zmiňované zpřesňující algoritmy.

Chceme – li tedy vypočítat vícerozměrný integrál *I,* můžeme postupovat následovně:

$$I = \int_G g(P) f(P) dP, \tag{13}$$

kde *f(P)* zastupuje zadanou hustotu pravděpodobnosti v oblasti *G*, přes kterou se integruje, a $P = (x, y, \ldots)$ je bod z oblasti *G*.

Stejně jako u jednorozměrných integrálů generujeme náhodné body $P_1$, $P_2$, ... s hustou *f(P)* a zavedeme náhodnou veličinu *X=g(P),* jejíž matematické očekávání je rovno *I*

$$I = E(X) = \int_G g(P) f(P) dP. \tag{14}$$

Integrál pak vypočítáme jako

$$\overline{X} = I' = \frac{1}{n} \sum_{i=1}^{n} X_i = \frac{1}{n} \sum_{i=1}^{n} g(P_i). \tag{15}$$

Podobně lze použít na vícerozměrné integrály i geometrická metoda. Například u dvojrozměrných integrálů pak nebudeme samozřejmě hledat body pod křivkou, ale pod plochou. Pokud je funkce $g(P)$ je ve vyšetřované oblasti *G* omezená vztahem $0 \le g(P) \le 1$, vytvoří se nová *d* + 1 rozměrná oblast $G \times (0, c)$ a v ní se pak generují náhodné body *Q*. Zjišťujeme kolik z *n* vygenerovaných bodů leží pod povrchem plochy $z = g(P)$. Jejich počet označíme $n'$. Hodnotu hledaného integrálu hledáme jako

$$I' = cG \frac{n'}{n}. \tag{16}$$

[4]

## 1.5 Řešení soustavy lineárních rovnic metodou Monte Carlo

Pro snadnější algoritmizaci budeme pracovat se soustavou rovnic v maticovém tvaru

$$Ax = b, \tag{17}$$

Kde $A = [a_{ij}]$ pro i,j = 1, 2, …, N,

$b$ – sloupcový vektor pravých stran.

Metoda Monte Carlo bude dále potřebovat soustavu upravit na tvar

$$x = a + Hx. \tag{18}$$

Předpokládejme, že prvky matice $H$ splňují podmínky:

1. $h_{ij} \geq 0$ pro i = 1, 2, …, N, a j = 1, 2, …, N,

2. $\sum_{j=1}^{N} h_{ij} < 1$ pro $i$ = 1, 2, …, N.

Při splnění podmínek můžeme řešení zapsat ve tvaru

$$x = [J - H]^{-1} a, \tag{19}$$

matici $[J - H]^{-1}$ můžeme za uvedených předpokladů vyjádřit ve formě řady

$$[J - H]^{-1} = J + H + H^2 + ... = \sum_{k=0}^{\infty} H^k \tag{20}$$

takže $x = a + Ha + H^2 a + H^3 a + ...$ . Speciálně pro $i$-tou složku řešení platí

$$x_i = a_i + \sum_j h_{ij} a_j + \sum_k \sum_j h_{ik} h_{kj} a_j + ... \tag{21}$$

Doplňíme-li matici $H$ podmíněných pravděpodobnostní přechodu $P$ tak, že přidáme jeden absorpční stav, matice $P$ bude mít potom tvar

$$P = \left[ \frac{H \mid r}{0 \mid 1} \right], \tag{22}$$

Kde 0 = (0, 0, …, 0) a $r$ je sloupcový vector se složkami

$$r_i = 1 - \sum_{j=1}^{N} h_{ij} \quad \text{pro i = 1, 2, …, N.} \tag{23}$$

Vytvořili jsme tedy Markovův řetězec s N + 1 stavy, ve kterém poslední stav je absorpční, neboť pravděpodobnost setrvání v tomto stavu je jednotková. Generujme nyní náhodné posloupnosti stavů Markovova řetězce s maticí podmíněných pravděpodobností přechodu za předpokladu, že vychází z *i*-tého stavu pro i = 1, 2, …, N. Dostaneme posloupnosti stavů typu

$$\alpha_i = [i, j_1, j_2, ..., j_k, N+1], \tag{24}$$

kde $j_1, j_2, \ldots, j_k \in \{1, 2, \ldots, N\}$. Všechny posloupnosti $\alpha^i$ končí číslem absorpčního stavu. Posloupnostem $\alpha^i$ přiřadíme hodnoty náhodné veličiny $X(\alpha^i)$ předpisem

$$X(\alpha^i) = \frac{a_{jk}}{r_{jk}}. \tag{25}$$

Veličina *X* nabývá tudíž hodnot

$$\frac{a_1}{r_1}, \frac{a_2}{r_2}, ..., \frac{a_N}{r_N}, \tag{26}$$

které závisí na čísle posledního stavu posloupnosti $\alpha^i$ před absorbcí. Pro střední hodnotu diskrétní náhodné veličiny $X(\alpha^i)$ platí

$$E[X(\alpha^i)] = \sum_{j=1}^{N} \frac{a_j}{r_j} p_j, \tag{27}$$

kde $p_j$ jsou pravděpodobnosti, s nimiž velična *X* nabývá hodnot pravděpodobnosti, že v stavové posloupnosti v i-tém stavu, bude předposledním prvkem *j*-tý stav. Pravděpodobnosti $p_j$ můžeme snadno určit následující úvahou. Pravděpodobnost, že systém přejde v k krocích ze stavu i do stavu j a v bezprostředně následujícím kroku do absorpčního stavu, je dána součinem

$$h_{ij}^k r_j \quad \text{pro k = 1, 2, ….,} \tag{28}$$

kde $h_{ij}^k$ je prvek matice $H^k$. Pravděpodobnosti $p_j$ dostaneme sečtením součinů přes všechna *k*, neboli

$$p_j = \sum_{k=0}^{\infty} h_{ij}^k r_j, \tag{29}$$

Dosazením dostaneme:

$$E[X(\alpha^i)] = \sum_{j=1}^{N} \frac{a_j}{r_j} \sum_{k=0}^{\infty} h_{ij}^k r_j = \sum_{k=0}^{\infty} \sum_{j=1}^{N} h_{ij}^k a_j. \tag{30}$$

Generujeme náhodné posloupnosti stavů, které končí absorpčním stavem. Na základě dostatečně velkého počtu posloupností vycházejícího z *i*-tého stavu vypočítáme průměr hodnot veličiny *X*, přiřazených jednotlivým posloupnostem. Průměr je nestranným odhadem hodnoty $E[X(\alpha^i)]$, a tedy přibližnou hodnotou $x_i$. [2]

## 1.6 Využití metody Monte Carlo v ekonomice

Za zmínku jistě stojí také využití metody Monte Carlo v ekonomice. Poprvé byla využita při oceňování aktiv, častěji se ale používá pro určení míry rizika. Používájí se náhodné veličiny popsané určitým stochastickým rozložením ve tvaru *{X(t), t∈T}*, kde *T* je časový interval, v tzv. Markovských procesech, což jsou procesy, kde budoucí hodnota *X(s)* záleží jen na hodnotě současné *X(t)* kde *s>t*, ale ne na jejím předchozím vývoji (hodnotách *X(k)*, kde *k<t*).

Výhodou metody při využití v ekonomických procesech je především možnost prověřit si možná rizika při rozhodování. Naopak nevýhodou je především její pracnost a to, že ekonomické procesy jsou často nepředvídatelné a zasahují do nich nové nečekané faktory. Při simulacích jsou však zohledněny pouze faktory známé z minulosti. [7]

# 2 POČÍTAČOVÉ ALGEBRAICKÉ SYSTÉMY

Hlavním úkolem této práce je praktické zpřístupnění výše popsaných algoritmů většímu množství studentů. Ti samozřejmě nemusí mít žádné, nebo třeba jen základní znalosti o problému. Nabízí se použití počítačových algebraických systémů (PAS), které umožňují velké usnadnění práce v technických a matematických oborech. Příkladem jsou AXIOM, MAPLE, MATHEMATICA, MATLAB, MATHCAD.. Problematika počítačových algebraických systému je složitá v tom, že je mnoho systémů a dají se velmi špatně mezi sebou porovnávat. Pro zvládnutí každého systému je potřeba dost času a zjistíte, že se systémem umíte toho čím dál více jen pokud s ním pravidelně a soustavně pracujete. Pokud zvládáte dobře jeden systém, není problém se naučit základy nějakého jiného, ale opravdu umět plně využít možnosti systémů jako je Mathematica a Matlab není jednoduché a možná ani v možnostech jediného člověka. Prakticky většinou školy využívají toho softwaru, na který získají nejvýhodněji licence. Všechny tyto programy nacházejí využití především tam, kde je potřeba řešení složitějších matematických výpočtů a především jejich následné grafické zobrazení.

Obrovským přínosem těchto aplikací je především možnost propojit je právě s internetem. Student tím získá možnost vyzkoušet si na vlastní kůži příklady probírané v hodinách. Webmasteři ovládající technologie Java Script, PHP, Java a podobně by pravděpodobně dokázali většinu z výše jmenovaných problémů zvládnout i bez PAS. Ale třeba jen maticové násobení není v PHP nebo Java Scriptu příliš jednoduché. Pomocí dvourozměrných polí se sice k výsledku dostaneme, ale program bude mít hodně řádků a slabší programátor si s tím třeba neporadí vůbec. Při použití PAS není problém násobit matice na jednom řádku. Na internetové stránky se umístí formuláře, které odešlou data do speciálních forem výše zmíněných programů. Ty jsou zpracovány na serveru, kde je program nainstalován a výsledky jsou odeslány zpět na stránku, která se zobrazí studentovi. Příklady těchto forem PAS jsou:

- **MapleNet**, který umožňuje používat výpočetní možnosti Maple "na dálku" pouze pomocí webového rozhraní. **Maple T.A.** je webový nástroj pro vytváření testů a příkladů, které mohou sloužit nejen ke zkoušení a hodnocení, ale i domácí práci studentů.
- **MATLAB Web Server** je nástroj, který umožňuje vytváření aplikací prezentací

v Matlabu. Vstupní parametry simulace nebo regulační smyčky se zadávají prostřednictvím internetu pomocí HTML stránek, přičemž vlastní výpočet, resp. regulační proces, běží na vzdáleném počítači = serveru. [6]

## 2.1 WebMathematica

Další z forem PAS. Jelikož byla WebMathematica použita v praktické části diplomové práce, podívejme se na ni podrobněji. WebMathematica je jednoduchý způsob jak vytvářet interaktivní výpočty na webu. Tato technologie umožňuje vytvářet webové stránky, kde jsou využity výpočetní a vizualizační schopnosti Mathematicy v celé její šíři. Každému uživateli stačí standardní internetový prohlížeč (browser) pro zadávaní výpočtů i vizualizaci výsledků, tedy i přehledných grafů. WebMathematica a Mathematica jsou založené na stejné technologii, ale úplně jiném uživatelském rozhraní a jiné cílové skupině uživatelů. Webmathematica nabízí přístup do Mathematicy přes browser. Používá vše z Mathematicy.

WebMathematica dává známé webové prostředí, které potřebuje ale trochu tréninku, aby se dalo efektivně používat. Uživatelé nemusí znát prostředí Mathematici. Mathematica je jako vývojové prostředí pro webMathematicu. Například, analytici mohou v Mathematice vytvářet modely a inženýři si je mohou spouštět přes webMathematicu.

WebMathematica je založena na dvou standardních Java technologiích:

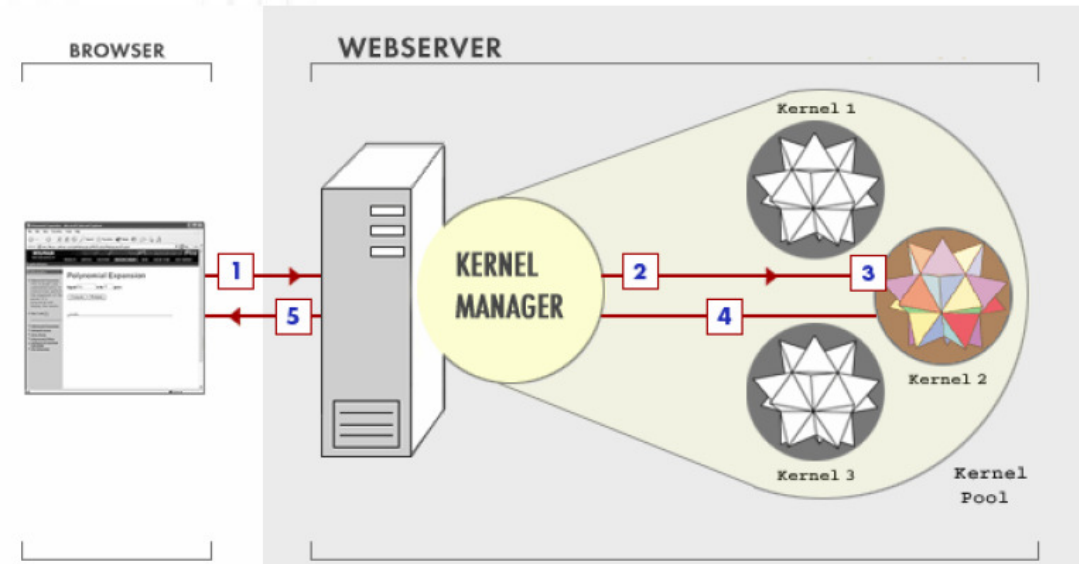
**- Java Servlet**

Servlety jsou speciální programy v Javě, které se spouštějí přes webový server, obvykle se nazývají"servlet container" někdy "servlet engine".

**- Java Server Pages (JSP)**

Při použití JSP se podobně jako v ASP nebo v PHP přímo do HTML kódu zapisují příkazy. Ty jsou nyní zapisovány v jazyce Java. Speciální servlet se stará o to, aby byla JSP stránka vždy po své modifikaci automaticky přeložena do byte-code.

**Zpracování požadavku Webmathematicou:**




*Obr. 3. Postup zpracování požadavku WebMathematicou*

1. webový prohlížeč posílá požadavek na webMathematica server
2. webMathematica server si rezervuje Mathematica kernel
3. Mathematica kernel je nastaven, provede kalkulace a vrací výsledky
4. webMathematica server vrací Mathematica kernel do „bazénu“
5. webMathematica server vrací výsledky webovému prohlížeči

### 2.1.1 Výroba vlastních MSP (Mathematica Server Pages)

- Statické stránky

**1. výroba notebooku v Mathematice**

a) nahrání balíčku

b) deklarace proměnných

c) funkce a hlavní kód

**2. vytvoreníHTML stránky, která bude obsahovat potřebný text (bez Mathematica tagů)**

**3. uložit soubor jako JSP do adresáře kde je nainstalovaná webMathematica**

**4. přidání webMathematica tagů**

**5. přidaní kódu v Mathematice mezi webMathematica tagy**

**6. přidání specifických příkazů do kódu v Mathematice**

**7. otestování stránky v prohlížeči**

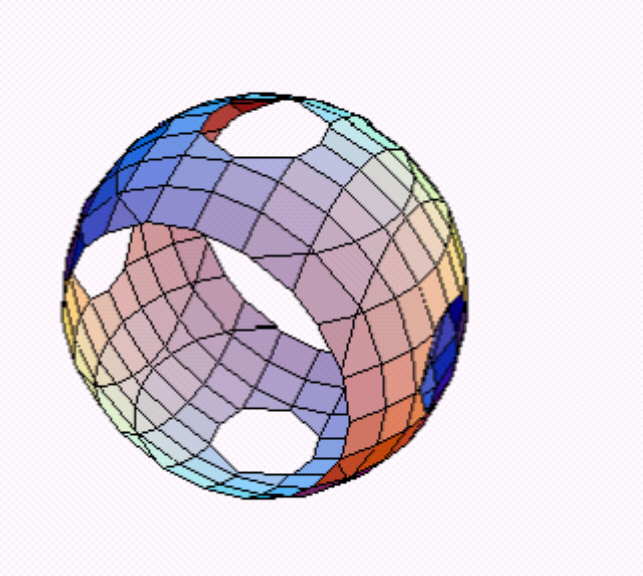
Příklad kódu:

```
<msp:evaluate>
    Get["Graphics`ContourPlot3D`"];
</msp:evaluate>

<msp:evaluate>
    xmin=-3; xmax=3;
    ymin=-2; ymax=2;
    zmin=-3; zmax=3;
</msp:evaluate>

<msp:evaluate>
    ContourPlot3D[Sin[Sqrt[x^2+y^2 + z^2]],
    {x, xmin, xmax}, {y, ymin, ymax}, {z, zmin, zmax},
    Boxed-> False]
</msp:evaluate>
```

**Výsledek:**



*Obr. 4. Výsledek statického kódu*

[9]

- Dynamické stránky

Tvorba dynamické stránky probíhá obdobně jako tvorba stránky dynamické. Rozdíl je v tom, že uživatel může měnit hodnoty proměnných pomocí webových formulářů. Ty pak například ovlivní vlastnosti zobrazovaných grafů, proto tedy dynamické stránky. Jediným problém je správné předávání hodnot z formulářů do proměnných kódu Mathematicy.

Například:

Formulář: <input type = “text” name = “x”>

V Mathematice bude hodnota proměnné: $$x [6]

# II. PRAKTICKÁ ČÁST

# 3 VYTVOŘENÍ PROGRAMŮ S ALGORITMY METODY MONTE CARLO

Všechny programy jsou vytvořeny pomocí prostředí webMathematica, které je popsáno v kapitole 2.1. Volba padla na toto prostředí především z důvodu přístupnosti programů přes webové rozhraní, což umožní jejich snadnou rozšiřitelnost. Další výhodou je snadné vykreslení přehledných grafů k jednotlivým algoritmům. Také je do budoucna počítáno se začleněním odkazů na tyto programy do webové pomůcky (www stránek) pro předmět Simulace systémů. Vzhledem k tomu, že tyto stránky nejsou ještě dokončeny, byly do programů přidány i stránky se základní teorií ke každému algoritmu. Po zapojení do webové pomůcky budou programy vždy u příslušné kapitoly s teorií.

Konkrétně programy ukazují algoritmy pro určení Ludolfova čísla, výpočet jednoduchých integrálů, dvojitých integrálů a výpočet soustavy lineárních rovnic, jejich princip je ukázán v teoretické části této práce.

## 3.1 Program na určení hodnoty Ludolfova čísla

Pro první program byl vybrán nejjednodušší algoritmus využívající metody Monte Carlo, který však umožní studentům snadno pochopit základní myšlenku metody. Jedná se o určení hodnoty Ludolfova čísla z jednotkové kružnice, kdy pomocí generování pseudonáhodných čísel a jednoduchým výpočtem můžeme dojít k poměrně přesným výsledkům.

### 3.1.1 Princip výpočtu

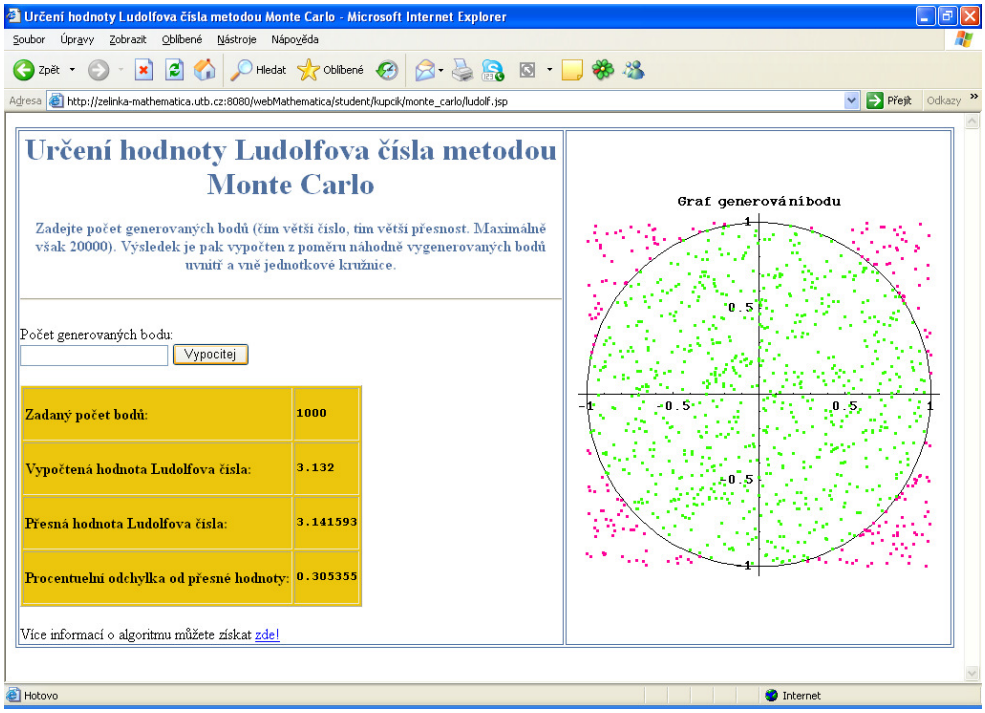
Vstupním parametrem algoritmu je počet generovaných bodů, který zadává uživatel. Dále probíhá výpočet následovně:

- Je vygenerována dvojice bodů $x$ a $y$ z rovnoměrného rozdělení z intervalu <-1,1>
- Pokud dvojice splňuje předpis $x^2 + y^2 \leq 1$, leží uvnitř jednotkové kružnice
- Podle počtu generovaných bodů je vygenerováno příslušné množství dvojic a u každé je rozhodnuto, jestli leží uvnitř jednotkové kružnice
- Nakonec je určena hodnota Ludolfova čísla pro daný počet generovaných bodů podle rovnice 5.

- Zdrojový kód je k nahlédnutí v příloze P I.

### 3.1.2 Vzhled a ovládání programu

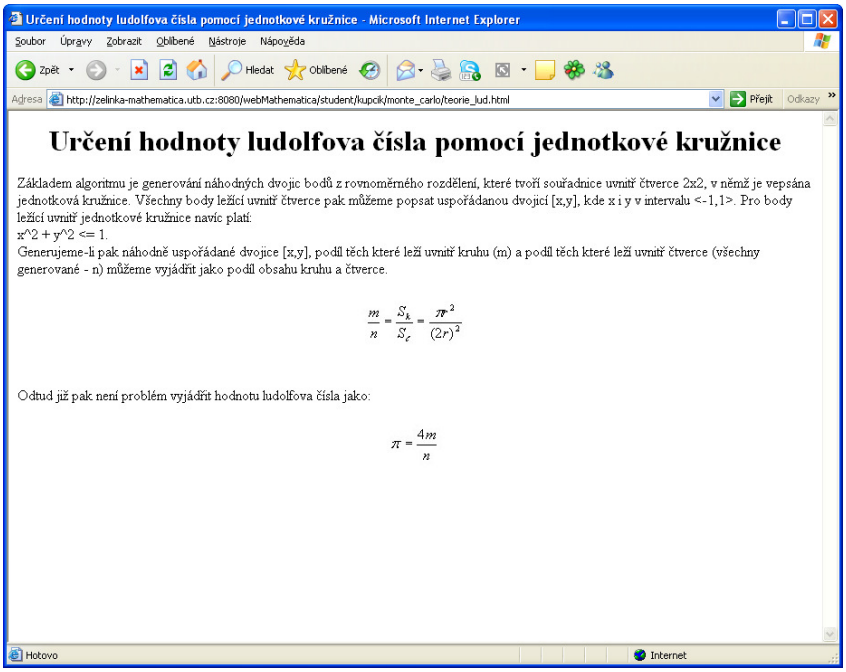
Jelikož tento algoritmus potřebuje jediný vstupní parametr, počet generovaných bodů, obsahuje program pouze jedno formulářová pole pro zadávání hodnoty uživatelem. Výsledky jsou pak vypisovány v tabulce. V pravé části obrazovky je vykreslen graf znázorňující generované body, které padly dovnitř jednotkové kružnice (zelené) a naopak které jsou mimo kružnici (červené).




*Obr. 5. Vzhled programu na určení Ludolfova čísla*

Počet generovaných bodů musí být z rozmezí <1,20000>. Toto rozmezí bylo zvoleno z důvodu timeoutu na serveru, kdy při dlouhé době výpočtu dochází k zastavení výpočtu webMathematiky. Pokud uživatel zadá hodnotu mimo tento interval nebo jinak neplatnou hodnotu, zobrazí se v tabulce místo hodnoty výsledku hláška: „Zadejte počet bodů: 0<pocet_bodu<=20000“. V tabulce je dále zobrazen počet generovaných bodů, přesná hodnota Ludolfova čísla na 6 desetinných míst a procentuelní odchylka od přesné hodnoty.

Program dále obsahuje odkaz na html stránku se základem teorie k algoritmu.




*Obr. 6. Teorie k programu na výpočet Ludolfova čísla*

Odkaz na program:

http://zelinka-mathematica.utb.cz:8080/webMathematica/student/kupcik/monte_carlo/ludolf.jsp

## 3.2 Program na výpočet určitého integrálu

Mezi aplikacemi využívajícími metodu Monte Carlo nemohl chybět výpočet určitého integrálu jako pravděpodobně nejznámější využití metody. V programu je použit výpočet pro zadanou odchylku od přesné hodnoty a pravděpodobnost, že odchylky bude dosaženo. Výpočet je prováděn geometrickou metodou popsanou v kapitole 1.3.1.

### 3.2.1 Princip výpočtu

Vstupními parametry, které zadává uživatel, je zadaná funkce (vzhledem k potřebě zadání funkce ve tvaru pro software Mathematica má uživatel možnost vybrat si funkci z předem nadefinovaných nebo pokusit se zadat vlastní podle přiloženého návodu), dále meze v kterých se má integrace provádět a také vybere povolenou odchylku v procentech a koeficient spolehlivosti v procentech (procentuelní pravděpodobnost dosažení stanovené odchylky). Celý algoritmus je proveden dvakrát, nejprve pro 500 bodů (500 souřadnic bodů

*x,y*) pro každý podinterval na zadané funkci, pro určení pravděpodobností padnutí bodů pod křivku funkce. Tyto pravděpodobnosti jsou pak použity k výpočtu potřebného množství generovaných bodů pro požadovanou přesnost. Algoritmus pak tedy proběhne podruhé pro tento počet bodů.

- Nejdříve je potřeba najít všechny kořeny funkce na zadaném intervalu. Integrace se pak rozdělí na podintervaly. Pokud se funkce nachází pod osou *x*, je znaménko hodnoty integrálu opačné než v případě nad osou *x*. Proto je potřeba provést výpočet pro každý interval zvlášť.
- Dále je potřeba nalézt maxima, případně minima, funkce na daných podintervalech, kvůli určení obdélníku pro generování náhodných bodů
- Na všech podintervalech je postupně generováno 500 hodnot *x* a *y* z rovnoměrného rozdělení na intervalu <*a,b*>, kde *a, b* jsou meze právě počítaného podintervalu. Pro vygenerované *x* je také spočítána funkční hodnota zadané funkce. Ta se porovná s generovanou hodnotou y a rozhodne se, jestli vygenerovaný bod leží pod křivkou nebo ne.
- Podle četnosti padnutí bodů pod křivku je určena hodnota pravděpodobnosti *P* a podle rovnice 7 přibližná hodnota integrálu.
- Ze získaných pravděpodobností je vypočten potřebný počet generovaných bodů *N* pro zadanou odchylku a koeficient spolehlivosti podle vztahu:

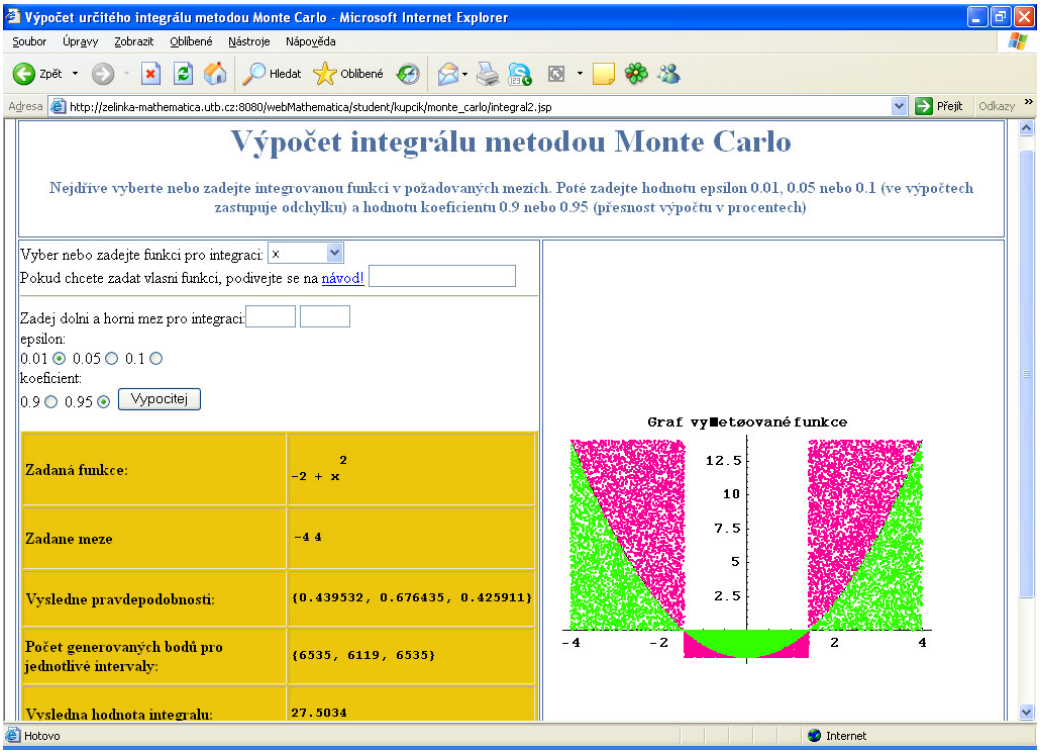
$$N \geq \frac{u_{1=\alpha/2}^2 P(1-P)}{\varepsilon^2} \tag{31}$$

  kde $u_{1=\alpha/2}$ je kvantil normálního rozdělení určený z tabulek podle zadaného koeficientu spolehlivosti, *P* je pravděpodobnost pro daný podinterval a $\varepsilon$ je požadovaná odchylka.

- Celý výpočet provedeme znovu pro zjištěný počet generovaných bodů pro příslušné podintervaly
- Konečnou hodnotu integrálu získáme jako součet výsledků pro podintervaly.
- Zdrojový kód je k nahlédnutí v příloze P II.

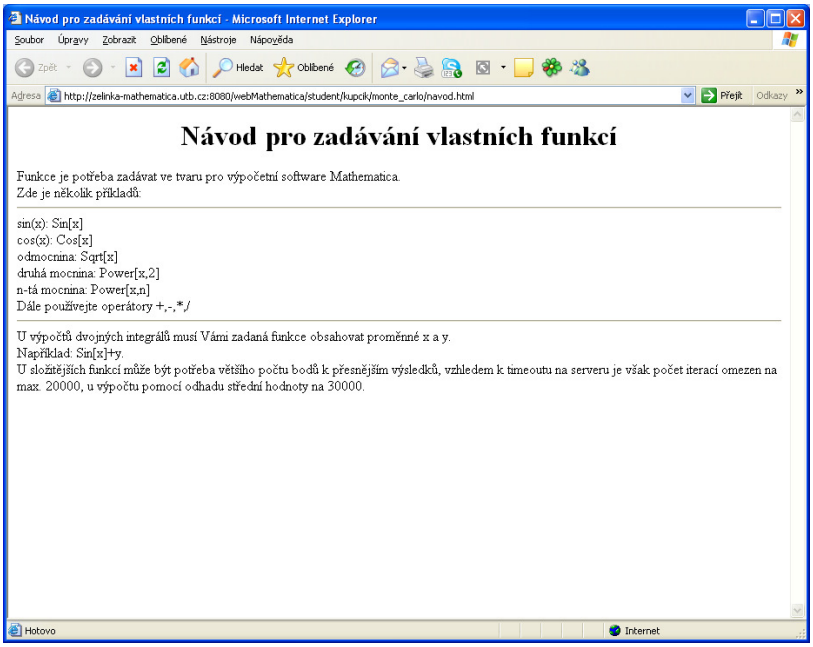
### 3.2.2 Vzhled a ovládání programu

Okno programu je opět rozděleno na část, kde se zadávají vstupní parametry a vypisuje tabulka s výsledky, a dále část, kde se zobrazuje výsledný graf generování bodů.




*Obr. 7. Vzhled programu na výpočet určitých integrálů*

Uživatel může integrovanou funkci vybrat z předem nadefinovaných nebo se pokusit podle návodu zadat svou vlastní.



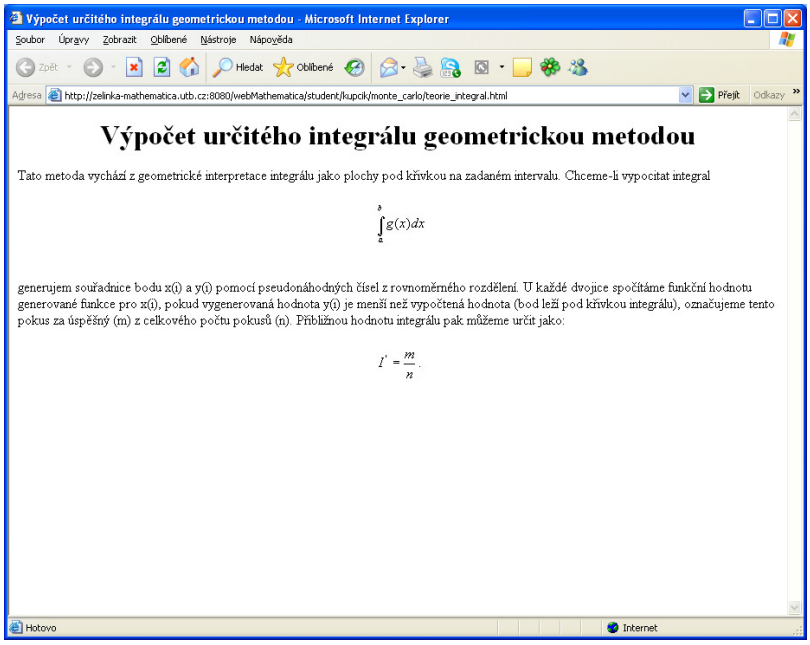

*Obr. 8. Návod pro zadávání vlastních funkcí*

Dále okno obsahuje textová pole pro zadání mezí integrálu a výběr povolené odchylky a koeficientu spolehlivosti.

Tabulka s výsledky pak obsahuje zadanou funkci, meze, výsledné pravděpodobnosti, počty bodů generovaných v jednotlivých intervalech, vypočítanou hodnotu integrálu, hodnotu integrálu spočtenou analyticky, relativní a absolutní odchylku od analytické hodnoty.

Program dále obsahuje chybové hlášky pro chybné zadání některého z parametrů. Konkrétně pak: "Byla zadána funkce ve špatném tvaru nebo funkci nelze spočítat v daných mezích." při problému se zadanou funkcí, nebo "Zadejte meze v rozmezí: -10 až 10" při špatně zadaných mezích.

Podobně jako u programu pro určení Ludolfova čísla je pod tabulkou odkaz na základní teorii.




*Obr. 9. Teorie k programu na výpočet integrálů*

Přímý odkaz na program:

http://zelinka-mathematica.utb.cz:8080/webMathematica/student/kupcik/monte_carlo/integral2.jsp

## 3.3 Program na výpočet dvojného integrálu

Program se skládá ze dvou částí. V první je výpočet prováděn geometrickou metodou pro předem nadefinované funkce. Pokud chce uživatel zadat vlastní funkci, může zvolit druhou část programu, kde je výpočet prováděn metodou odhadu střední hodnoty. Důvodem k tomuto rozdělení bylo umožnění výpočtu většího množství funkcí, se kterými by geometrická metoda měla problémy (především u funkcí zasahující pod rovinu $z$=0). Výpočet tentokrát není prováděn podle požadované přesnosti, ale uživatel zadává počet generovaných bodů.

### 3.3.1 Princip výpočtu geometrické metody

Vstupními parametry jsou. kromě funkce pro integraci, počet generovaných bodů (čím více bodů, tím větší přesnost výpočtu) a meze pro integraci pro x a pro y.

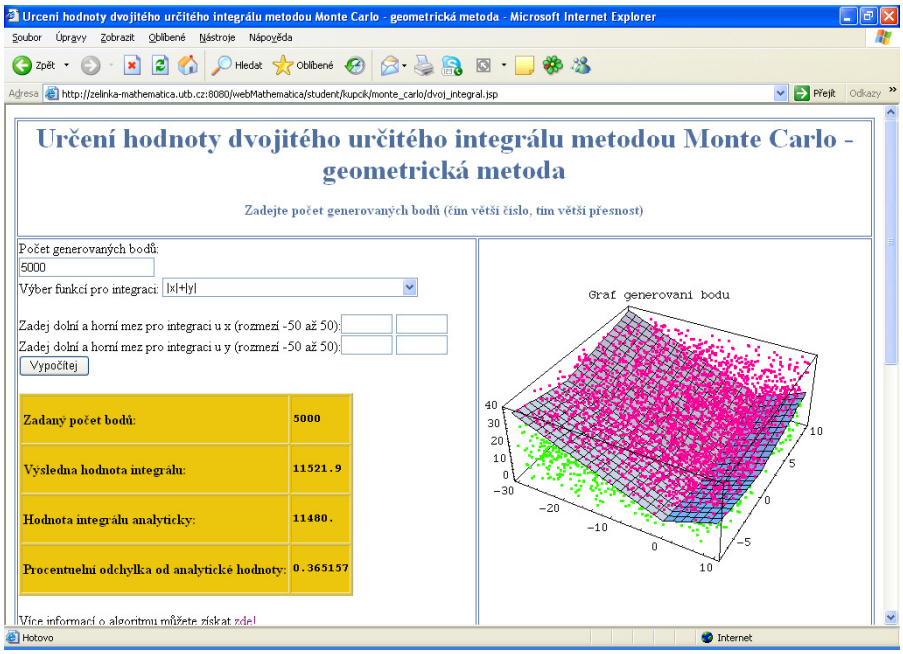
Postup výpočtu:

- Nejdříve je zjištěno minimum a maximum funkce v zadaných mezích

- Dále jsou generovány pseudonáhodné body z rovnoměrného rozdělení pro *x, y* a *z* v daných intervalech (*x* v mezích pro *x*, *y* v mezích *y* a *z* v mezích nalezeného maxima a minima pro meze *x* a *y*)
- Pro vygenerovanou dvojici *x* a *y* je vypočítána skutečná funkční hodnota vyšetřované funkce a porovnána s generovanou hodnotou *z*
- Tímto porovnáním se rozhodne, jestli vygenerovaný bod leží nebo neleží pod plochou funkce.
- Generování a porovnání se provede pro zadaný počet bodů
- Přibližná hodnota integrálu se na základě padlých bodů pod plochu a počtu celkově generovaných bodů spočítá podle rovnice 17.

### 3.3.2 Vzhled a ovládání geometrické metody

Opět je okno rozděleno na část textovou se zadáváním parametrů a výpisem výsledků a část pro vykreslení grafu generování bodu.

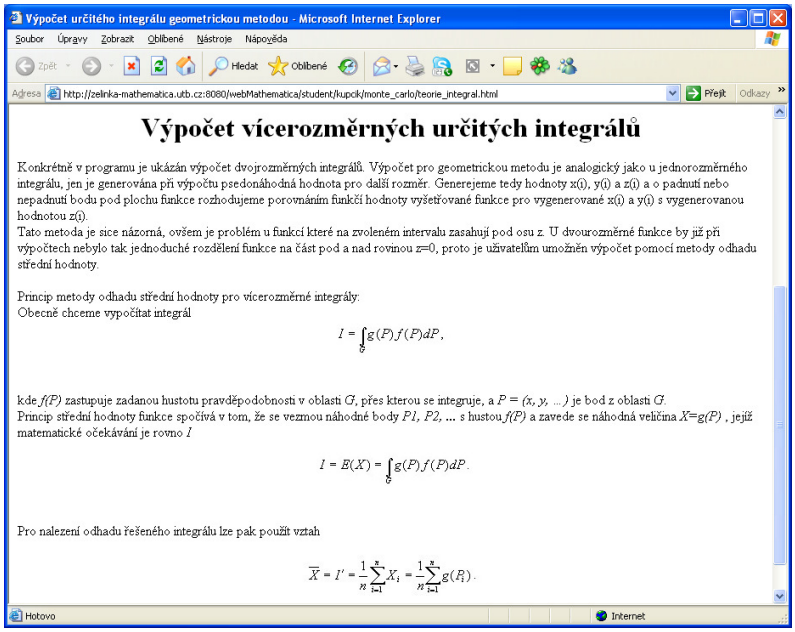



*Obr. 10. Vzhled programu na výpočet dvojných integrálů*

Na začátku uživatel zadává počet generovaných bodů a vybírá z předem nadefinovaných funkcí. Dále zadává integrační meze pro *x* a *y*. Tabulka s výpisem výsledků obsahuje počet

zadaných bodů, vypočítanou hodnotu integrálu a přesnou hodnotu spočítanou analyticky. Program může místo výsledku vypsat chybové hlášky. Konkrétně "Zadávejte meze v rozmezí -50 až 50, dolní mez musí byt menši než horní" při chybném zadání mezí nebo "Zadejte počet bodu: 0<pocet_bodu<=20000" při překročení maximálního povoleného počtu bodů nebo neplatné hodnotě.

Také tento program obsahuje odkaz na základní teorii o algoritmu.




*Obr. 11. Teorie k vícenásobným integrálům*

### 3.3.3 Princip výpočtu metodou odhadu střední hodnoty

Vstupní parametry jsou obdobné jako u geometrické metody, tedy integrovaná funkce, meze a počet generovaných bodů.

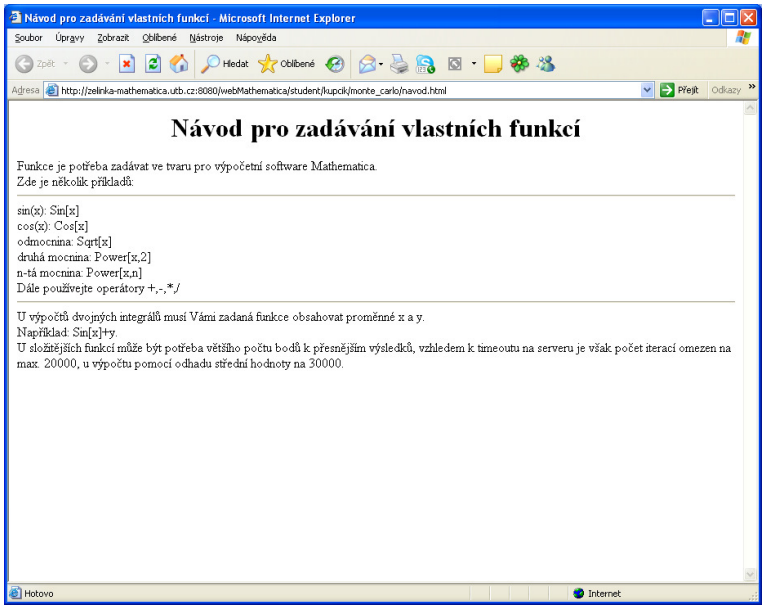
Postup výpočtu:

- Jsou generovány pseudonáhodné hodnoty pro $x$ a $y$ z rovnoměrného rozdělení pro zadané meze
- Podle rovnice 16 je postupně vypočítán odhad střední hodnoty, který odpovídá přibližné hodnotě integrálu

### 3.3.4 Vzhled a ovládání odhadu střední hodnoty

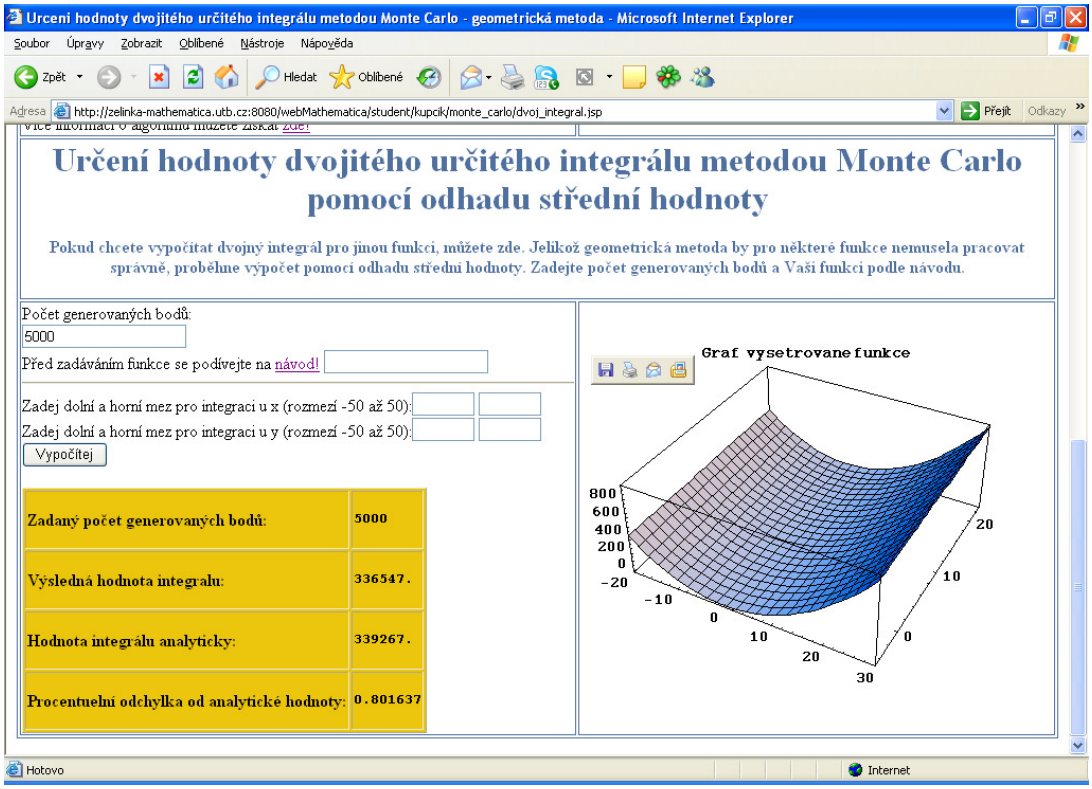
Také zde je okno rozděleno na část textovou a část pro vykreslení grafu.

Uživatel nejprve zadává počet generovaných bodů a podle návodu požadovanou funkci.




*Obr. 12. Návod pro zadávání funkcí u dvoj. integrálů*

Dále je opět potřeba zadat meze pro integraci. Po proběhnutí výpočtu je vykreslen graf funkce a výsledky jsou opět v tabulce.




*Obr. 13. Vzhled programu dvojitých integrálů druhá část*

Konkrétně je vypsán počet generovaných bodů, vypočtená hodnota integrálu, hodnota zjištěná analyticky a procentuelní odchylka od analytické hodnoty.

Program opět může vypsat místo výsledku následující chybové hlášky: "Zadejte počet bodu: 0<pocet_bodu<=30000" při špatné hodnotě počtu bodů, "Zadávejte meze v rozmezí -50 až 50, dolní mez musí být menší než horní" při špatně zadaných mezích a "Zadali jste funkci ve špatném tvaru nebo neobsahuje x i y" při špatně zadané funkci.

Odkaz na teorii je tentokrát společný pro geometrickou metodu i metodu odhadu střední hodnoty.

Přímý odkaz na program:

http://zelinka-mathematica.utb.cz:8080/webMathematica/student/kupcik/monte_carlo/dvoj_integral.jsp

## 3.4 Program na řešení soustavy lineárních rovnic metodou Monte Carlo

Využívat metodu Monte Carlo pro řešení jednoduchých soustav lineárních rovnic o dvou nebo třech neznámých nemá velký praktický význam. Avšak pro studenta, který se seznamuje s problematikou metody, může být tento program názornou ukázkou. V praxi se pak metoda Monte Carlo používá na řešení soustav s velkým počtem neznámých, kde by jiné způsoby byly příliš zdlouhavé nebo nenacházely řešení.

Omezení programu na soustavy rovnic o 2 nebo 3 neznámých je především z praktických důvodů, hlavně kvůli složitosti zadávání soustav. Je nutné zadat soustavu v maticovém tvaru $Ax=b$ a matice je potřeba zadat ve tvaru pro PAS Mathematica. Rozšíření programu na větší počet proměnných by sestávalo pouze z přidání dalších podmínek pro procházení nových stavů „bloudící částice“.

### 3.4.1 Princip výpočtu

Vstupy které zadává uživatel jsou: počet bodů = počet startů „bloudící částice“, matice $A$ a matice $b$.

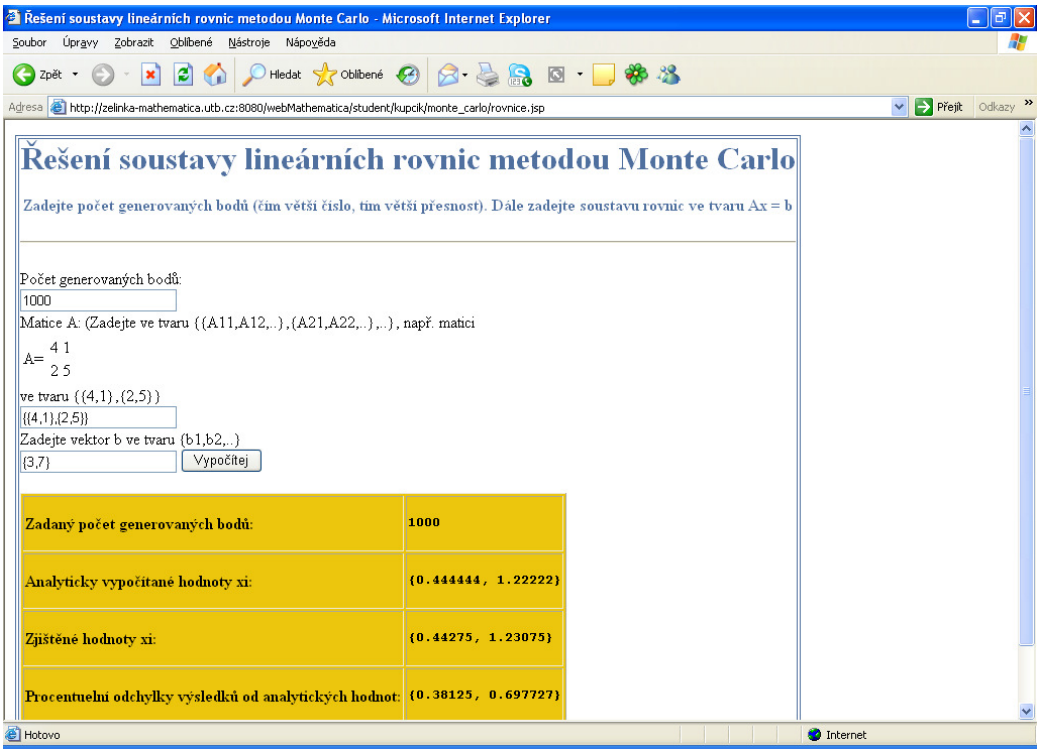
Postup výpočtu:

- Po kontrole vstupů jsou zjištěny rozměry matice $A$ a vektoru $b$. Podle toho jsou pak vygenerovány další proměnné v příslušných rozměrech.
- Soustavu je potřeba upravit na tvar $x=a+Hx$. Úprava je provedena tak, že od diagonálních prvků se odečte jednička a tento zbytek se převede na levou stranu. Poté se celá soustava podělí levou stranou. Důvodem úpravy je snaha o to, aby norma matice nebyla větší než jedna a aby sumace jednotlivých prvků nebyla větší než jedna. To jsou podmínky řešitelnosti soustavy metodou Monte Carlo. Pokud soustava nesplňuje podmínky ani po úpravě, je algoritmus zastaven.
- Dále je vygenerována matice vah a matice pravděpodobností. Váhy nabývají stavů +- 1. Zjištění vah se provádí procházením upravené matici $Hx$, pro prvky menší než nula se váha nastaví na -1, pro větší než nula na +1. Matice pravděpodobností se vytvoří podle postupů v kapitole 1.5.
- Samotný výpočet je pak získán po náhodném „bloudění částice“ podle kapitoly 1.5. Vždy je generována náhodná hodnota z rovnoměrné ho rozdělení <0,1>, po

porovnání této hodnoty s příslušným prvkem v matici pravděpodobností je rozhodnuto o přesunu částice do jiného stavu nebo o setrvání v současném. Bloudění končí přesunem do stavu koncového. Nakonec je z počtu průchodů mezi stavy statisticky vypočítána hodnota právě počítané neznámé. Konkrétní řešení je k nahlédnutí v příloze P IV.

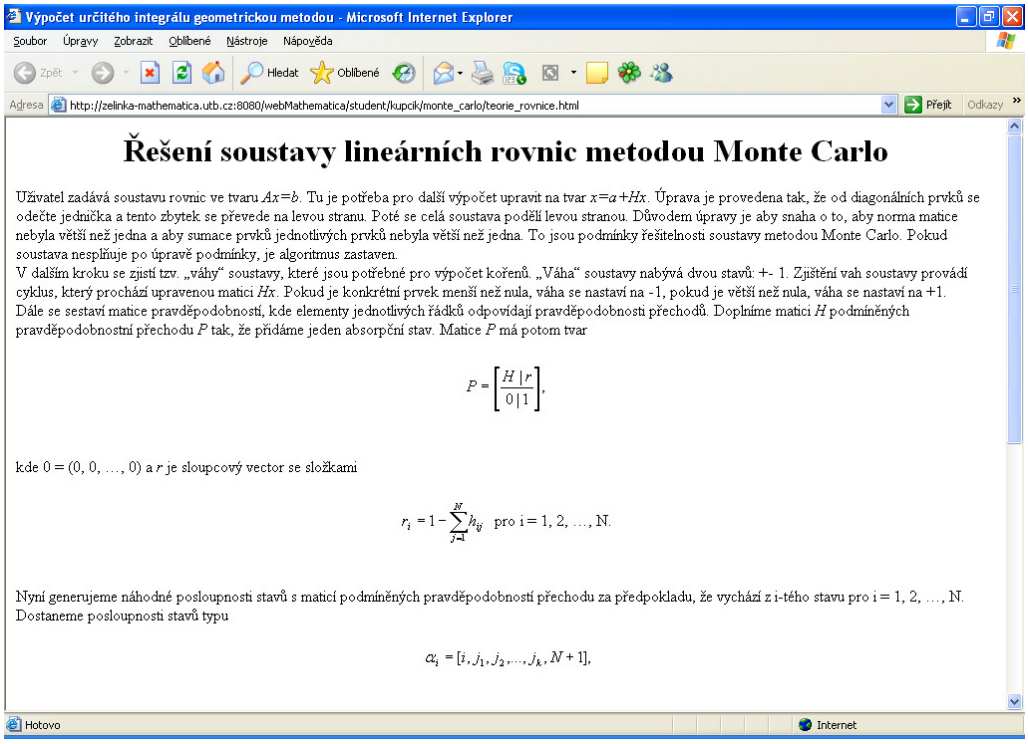
### 3.4.2 Vzhled a ovládání programu

Oproti předchozím programům není zde vymezena část na vykreslení grafů. Okno obsahuje pouze formuláře na zadání počtu bodů (počtu startů bloudící částice), dále ukázku jak zadávat matice soustavy, formuláře pro zadávání matic a tabulku s výsledky.




*Obr. 14. Vzhled programu na výpočet soustavy lineárních rovnic*

Také zde je odkaz na html stránku se základní teorií.




*Obr. 15. Teorie k řešení soustav rovnic metodou Monte Carlo*

Tabulka s výsledky obsahuje počet zadaných bodů, analytické řešení, řešení vypočítané metodou Monte Carlo a procentuelní odchylky od analytického řešení. Místo výsledku se mohou zobrazit následující chybové hlášky: "Nebyla splněna podmínka konvergence, výpočet nemohl proběhnout!!!" při nevyhovujícím tvaru upravené soustavy, "Vektor b musí obsahovat jen reálna čísla a musí mít stejnou délku jako rozměry matice A" při špatně zadaném vektoru *b*, "Matice A musí být čtvercová (max 3x3) a musí obsahovat jen reálna čísla" při špatně zadané matici *A* a "Zadejte pocet bodu: 0<pocet_bodu<=20000" při špatně zadaném počtu bodů.

Přímý odkaz na program:

http://zelinka-mathematica.utb.cz:8080/webMathematica/student/kupcik/monte_carlo/rovnice.jsp

# 4 ZPŘESŇUJÍCÍ ALGORITMY METODY MONTE CARLO

Smyslem této kapitoly je ukázat praktické využití některých zpřesňujících algoritmů metody Monte Carlo. Metody zpřesnění byly použity na algoritmus pro výpočet určitého integrálu. Každá metoda je navržena pro jiný typ funkcí, proto ne vždy musí nutně vést ke zlepšení. Algoritmy zpřesnění budou ukázány na 3 různých funkcích a výsledky porovnány s klasickou metodou pro odhad střední hodnoty, klasickou geometrickou metodou a analytickým řešení.

## 4.1 Popis naprogramovaných algoritmů

Pro každou metodu je výsledek vypočten jako střední hodnota ze 100 opakování výpočtu. Nejdříve si naznačme postup jednotlivých algoritmů. Všechny algoritmy jsou naprogramovány v prostředí Mathematica konkrétně v souboru metody_zpresneni.nb který je k dispozici jako příloha P V.

### 4.1.1 Klasická metoda odhadu střední hodnoty a geometrická metoda

Pro porovnání výsledků byla nejdříve naprogramována klasická metoda odhadu střední hodnoty a geometrická metoda, jejichž princip je popsán v kapitolách 1.3.1 a 1.3.2. Geometrická metoda je provedena pro generování 300 náhodných bodů, metoda odhadu střední hodnoty taktéž pro 300 náhodných ohodnocení účelové funkce. Také všechny zpřesňující metody jsou počítány pro 300 ohodnocení. Důvodem poměrně malého počtu opakování je potřeba vzniku větší chyby řešení, aby bylo možné názorně srovnat jednotlivé metody a jejich efektivnost.

### 4.1.2 Metoda zpřesnění - vymezení hlavní části

Podmínkou této medy je znalost funkce $h(x)$, která přibližně odpovídá řešení hledané funkce $g(x)$ (= hlavní část). Pro potřeby experimentu posloužila místo funkce $h(x)$ klasická metoda odhadu střední hodnoty, do které byla navíc uměle přidána chyba při každém ohodnocení účelové funkce.

Postup výpočtu:

- Nejdříve je na funkci použita klasická metoda odhadu střední hodnoty. Hodnoty účelových funkcí s uměle přidanou chybou pro jednotlivá ohodnocení jsou uloženy

do pole pro další výpočet jako hodnoty funkce *h(x)* pro náhodně generované hodnoty *x*

- Do pole jsou uloženy také hodnoty náhodně generované hodnoty *x* z rovnoměrného rozdělení na požadovaném intervalu funkce pro další výpočet
- Při samotném odhadu hodnoty integrálu je použit vztah

$$I = \frac{b-a}{n}\sum_{i=1}^{n}[g(x_i) - h_n(x_i)] + J \tag{32}$$

kde *a, b* jsou meze požadované pro integraci, *n* počet ohodnocení funkce, *g(x)* hodnoty hledané funkce a *h(x)* uložené hodnoty hlavní části. *J* je výsledný hodnota integrálu pro funkci *h(x)*

Ze vztahu 33 je jasně vidět, že ve druhé části aplikujeme metodu Monte Carlo na rozdíl mezi hodnotami hlavní části *h(x)* a funkce *g(x)*.

### 4.1.3 Metoda zpřesnění - výběr na podintervalech

Pokud integrovanou funkci rozdělíme na podintervaly, nemusíme použít na všechny části stejného počtu náhodných bodů, ale na větší podintervaly (více na nich bude záležet výsledná hodnota integrálu) použijeme více bodů, na menší méně.

Postup výpočtu:

- Na integrované funkci jsou nejdříve vybrány 3 podintervaly
- Pro každý podinterval je vypočítán potřebný počet generovaných bodů podle vzorce

$$\tilde{n} = \frac{nl}{b-a} \tag{33}$$

kde *n*=300, *l* je rozdíl mezí na podintervalu (velikost podintervalu) a *a,b* jsou meze pro celou integrovanou funkci.

- Výsledná hodnota integrálu je pak vypočítána jako

$$I = \sum_{j=1}^{s}\frac{l_j}{n_j}\sum_{i=1}^{n_j} g(x_i^{(j)}) \tag{34}$$

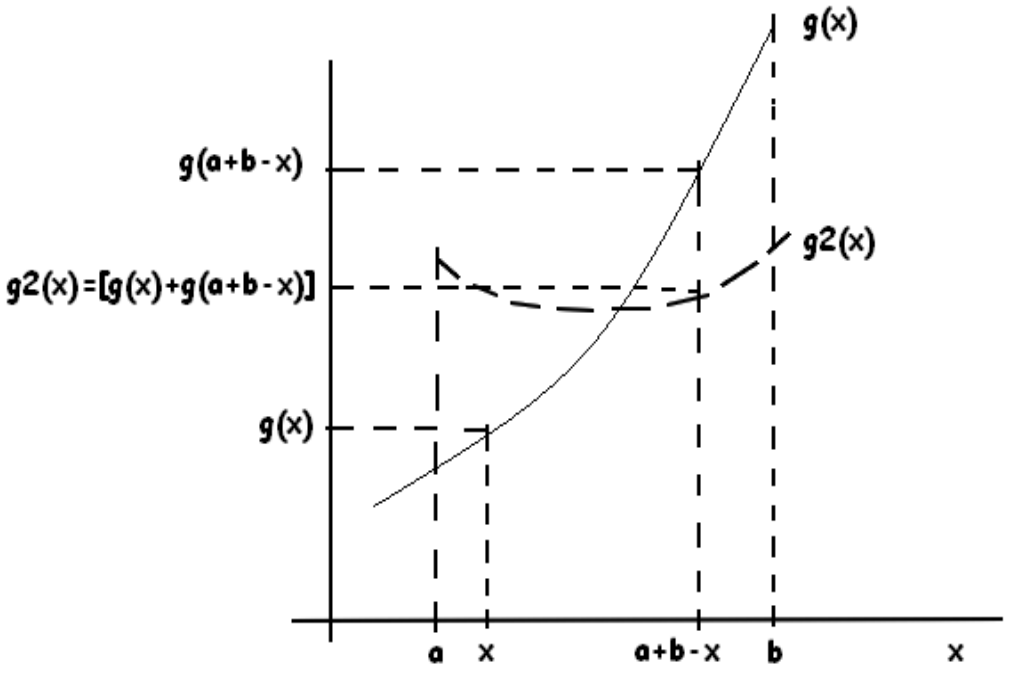
kde *s* je počet podintervalů.

Z celé metody je patrné, že výsledek a tím i úspěšnost zpřesnění bude vždy záležet na konkrétním rozdělení integrované funkce na podintervaly.

### 4.1.4 Metoda zpřesnění - symetrizace integrované funkce

Pokud je integrovaná funkce monotonní, je vhodné ji pro snížení rozptylu symetrizovat podle vztahu

$$g(x)' = \frac{1}{2}[g(x) + g(a+b-x)] \tag{35}$$




*Obr. 16. Symetrizace integrované funkce*

Postup výpočtu:

- Nejdříve jsou generovány náhodné body stejně jako u metody odhadu střední hodnoty
- Výsledný integrál je však odhadován podle vztahu

$$I = \frac{1}{2n}\sum_{i=1}^{n}[g(x_i) + g(a+b-x_i)](b-a) \tag{36}$$

Nevýhodou metody je prodloužení doby výpočtu, zmenšení rozptylu je ale výrazné. Také je potřeba znát informace o funkci kvůli provedení symetrizace. U složitějších funkcí by výše popsaný způsob nestačil a symetrizace by se prováděla složitěji. Podmínkou je také monotonnost funkce.

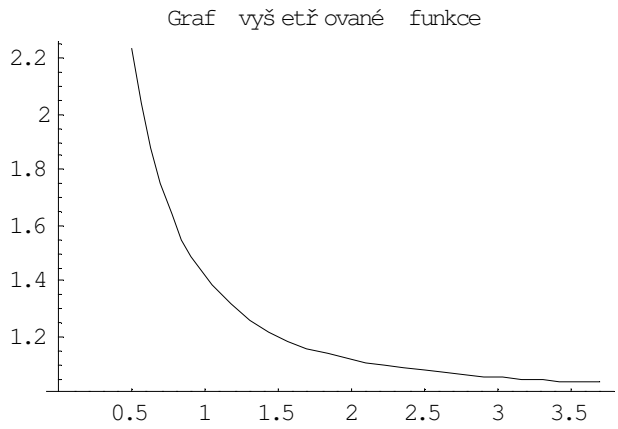
## 4.2 Testování metod

- Jako první byla vybrána funkce

$$g(x) = \frac{\sqrt{x^2 + 1}}{x} \tag{37}$$

a na ní byl všemi výše uvedenými metodami počítán integrál

$$\int_{0.5}^{3.7} g(x)\, dx \tag{38}$$




*Obr. 17. Graf první zkoumané funkce*

Výsledky experimentů jsou zobrazeny v následující tabulce. Obsahuje hodnoty pro neupravenou metodu odhadu střední hodnoty, geometrickou metodu, metodu vymezení hlavní části, výběru na podintervalech a symetrizaci integrované funkce. Výsledky jsou střední hodnoty ze sta realizací výpočtu, dále je v tabulce směrodatná odchylka a procentuelní odchylka od přesného analytického řešení. Jeho hodnota pro tento integrál je **3.89127**.

*Tab. 1. Srovnání zpřesňujících metod u první funkce*

| | **Vypočtená hodnota integrálu ze 100 pokusů** | **Směrodatná odchylka** | **Procentuelní odchylka od analytického řešení [%]** |
|---|---|---|---|
| **Odhad střední hodnoty** | 3.895 | 0.043 | 0.08 |
| **Geometrická metoda** | 3.929 | 0.186 | 0.98 |
| **Vymezení hlavní části** | 3.887 | 0.038 | 0.11 |
| **Výběr na podintervalech** | 3.903 | 0.027 | 0.30 |
| **Symetrizace funkce** | 3.889 | 0.022 | 0.06 |

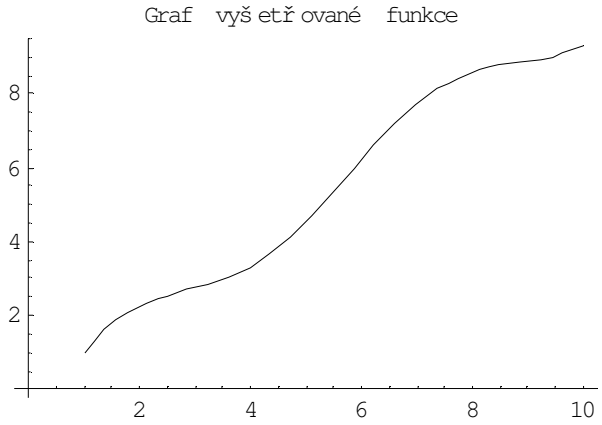
Z výsledků je patrné, že u funkce s takto jednoduchým průběhem na integrovaném intervalu dosahuje velmi dobrých výsledků samotná metoda odhadu střední hodnoty. Výraznějšího zlepšení bylo dosaženo pouze metodou symetrizace integrované funkce. Ta zde měla své opodstatnění, protože touto metodou došlo ke zmenšení rozdílů mezi funkčními hodnotami na integrovaném intervalu a tím i k přesnějšímu výpočtu střední hodnoty. Ostatní metody sice vedly ke zmenšení směrodatné odchylky, avšak od analytické hodnoty se lišily ještě nepatrně víc. Naopak nejhorších výsledků podle očekávání bylo dosaženo geometrickou metodou, která by pro větší přesnost potřebovala podstatně větší množství generovaných bodů.

- Druhá vyšetřovaná funkce má tvar

$$g(x) = Sin(\frac{x-1}{x})Cos(x-1) + x \tag{39}$$

a na ní byl všemi výše uvedenými metodami počítán integrál

$$\int_{1}^{10} g(x)\,dx \tag{40}$$




*Obr. 18. Graf druhé zkoumané funkce*

Výsledky experimentů jsou opět zobrazeny v následující tabulce. Přesná analytická hodnota je v tomto případě **49.511**.

*Tab. 2. Srovnání zpřesňujících metod u druhé funkce*

| | **Vypočtená hodnota integrálu ze 100 pokusů** | **Směrodatná odchylka** | **Procentuelní odchylka od analytického řešení [%]** |
|---|---|---|---|
| **Odhad střední hodnoty** | 49.38 | 1.41 | 0.27 |
| **Geometrická metoda** | 49.63 | 2.51 | 0.24 |
| **Vymezení hlavní části** | 49.55 | 1.38 | 0.08 |
| **Výběr na podintervalech** | 49.63 | 0.69 | 0.25 |
| **Symetrizace funkce** | 49.51 | 0.07 | 0.00 |

U této již o něco složitější funkce došlo ke zhoršení klasické metody odhadu střední hodnoty a její výsledek je srovnatelný s geometrickou metodou. Směrodatná odchylka geometrické metody je však pořád o hodně větší. K výraznějšímu zlepšení došlo u metody vymezení hlavní části, metoda výběru na podintervalech je příliš závislá na výběru podintervalů a při jiné volbě by se výsledky mohly výrazně lišit. Jelikož i tato funkce byla na integrovaném intervalu monotonní, dosáhla nejlepšího výsledku opět metoda symetrizace integrované funkce. Ta se lišila od analytického

řešení až na třetím desetinném místě, které je již však vzhledem ke směrodatné odchylce nepodstatné.

- Třetí integrovaná funkce má tvar

$$g(x) = Sin(2x) + x/2 \tag{41}$$

a na ní byl všemi výše uvedenými metodami počítán integrál

$$\int_1^{10} g(x)\,dx \tag{42}$$


*Obr. 19. Graf třetí zkoumané funkce*

Výsledky experimentů jsou opět zobrazeny v následující tabulce. Přesná analytická hodnota je v tomto případě **24.3379**.

*Tab. 3. Výsledky zpřesňujících metod u třetí funkce*

| | **Vypočtená hodnota integrálu ze 100 pokusů** | **Směrodatná odchylka** | **Procentuelní odchylka od analytického řešení [%]** |
|---|---|---|---|
| **Odhad střední hodnoty** | 24.26 | 0.70 | 0.32 |
| **Geometrická metoda** | 24.43 | 1.45 | 0.38 |
| **Vymezení hlavní části** | 24.43 | 0.66 | 0.36 |
| **Výběr na podintervalech** | 24.43 | 0.44 | 0.38 |
| **Symetrizace funkce** | 24.37 | 0.35 | 0.16 |

Opět se potvrdilo, že metoda odhadu střední hodnoty postupně ztrácí na efektivitě s rostoucí složitostí průběhu funkce. Většina metod zde skončila podobně, nejmenší procentuelní odchylka i nejmenší směrodatná odchylka byla opět u symetrizace integrované funkce, i tady už ale není výsledek tak přesný jako u předchozí funkce. Důvodem je, že tato funkce není monotonní na celém svém intervalu a pro přesnější symetrizaci by bylo potřeba složitějších postupů. U metody vymezení hlavní části a metody výběru na podintervalech je opět možné výrazně ovlivnit výsledky volbou funkce pro výpočet hlavní části, respektive výběrem podintervalů.

Výsledky experimentů potvrdily, že vhodnost volby jednotlivých metod zpřesnění metody Monte Carlo závisí na tvaru integrované funkce. Každý z postupů je navržen pro jiný druh funkcí. U jednoduchých monotonních funkcí se dá dosáhnout přesných výsledků i s neoptimalizovanou metodou odhadu střední hodnoty, u složitějších funkcí se pak nejlépe osvědčila metoda symetrizace integrované funkce. Ta dosahuje nejlepších výsledků u všech funkcí, které jsou monotonní. Metody vymezení hlavní části je velmi důležitá znalost funkce právě pro vymezení hlavní části a právě s její přesností roste nebo klesá přesnost celé metody. U výběru na podintervalech můžeme přesnost ovlivnit správnou volbou podintervalů.

## 4.3 Metoda zpřesnění - hlavní výběr

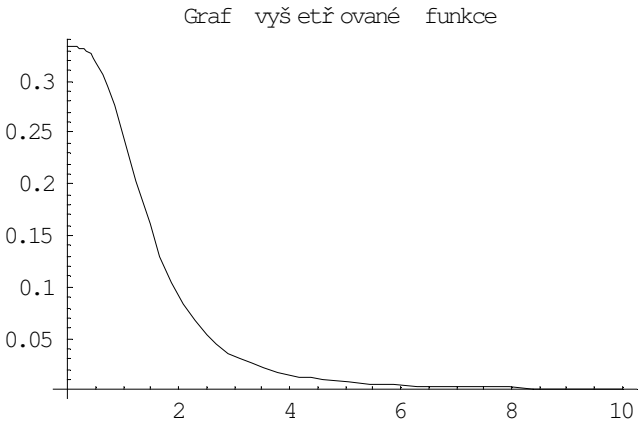
Dalším způsobem zpřesnění výpočtu metody Monte Carlo je použití jiného rozdělení než rovnoměrného. To volíme tak, aby většina náhodných čísel byla soustředěna do té části integrační oblasti, na které nejvíce závisí hodnota integrálu. Integrál pak hledáme jako

$$I = \int_a^b \frac{g(x)}{f(x)} f(x)dx \tag{43}$$

kde $f(x)$ je hustota pravděpodobnosti náhodné veličiny pro kterou platí $f(x)>0$ pro $a<x\leqq b$ a $\int_a^b f(x)dx = 1$.

Hlavní vyžití nachází tato metoda tam, kde generování bodů z rovnoměrného rozdělení není možné. Například chceme – li počítat integrál

$$\int_0^{\infty} \frac{1}{x^3+3}\,dx \tag{44}$$




*Obr. 20. Graf ukázkové funkce*

musíme zvolit takové rozdělení, které má rovněž definiční obor na intervalu $(0,\infty)$. Takovým je například exponenciální rozdělení s parametrem 1. Hustota pravděpodobnosti tohoto rozdělení $f(x) = \ell^{-x}$. Podle rovnice 44 pak musíme integrál upravit na tvar

$$\int_0^{\infty} \frac{1}{(x^3+3)\ell^{-x}}\,\ell^{-x}\,dx \tag{45}$$

V samotném algoritmu pak můžeme generovat náhodná čísla $x$ z exponenciálního rozdělení a při metodě odhadu pomocí střední hodnoty dosazovat do vztahu $\frac{1}{(x^3+3)\ell^{-x}}$.

Výpočet byl proveden stejně jako u předchozích metod pro 300 náhodných bodů a výsledek je střední hodnota ze 100 opakování experimentu. Výsledek je v následující tabulce, přesná hodnota tohoto integrálu je **0.5813**.

*Tab. 4. Výsledky při použití metody hlavního výběru*

| | **Vypočtená hodnota integrálu ze 100 pokusů** | **Směrodatná odchylka** | **Procentuelní odchylka od analytického řešení [%]** |
|---|---|---|---|
| **Odhad střední hodnoty hlavním výběrem** | 0.57 | 0.02 | 0.49 |

Díky volbě správného rozdělení bylo dosaženo dostatečné přesnosti a především malé směrodatné odchylky.

# ZÁVĚR

Pro zpracování této práce jsem využil prostředí webMathematica, které je pro studenty na Univerzitě Tomáše Bati ve Zlíně přístupné. Využil jsem tak zkušenosti získané z práce bakalářské. Naučil jsem se využívat snadné práce s grafy v tomto prostředí, které výrazně oživily a znázornily výsledky algoritmů. Jedinou nevýhodou používání webMathematicy vidím ve složitém ladění zdrojového kódu kvůli absenci chybových hlášek. Vzhled programů jsem volil s ohledem na předpokládané zařazení odkazů na programy do webové pomůcky pro předmět Simulace systémů. Teorii k naprogramovaným algoritmům obsahuje teoretická část práce.

Také jsem naprogramoval některé zpřesňující algoritmy metody Monte Carlo a provedl jejich srovnání s klasickými postupy. Pro samotnou realizaci algoritmů jsem použil prostředí Mathematica. Výsledky dopadly podle očekávání. Jednotlivé metody se ukázaly být vhodné pro různé druhy funkcí. Prokázalo se, že využití metody pro nevhodnou funkci nemusí nutně dospět ke zpřesnění výsledku.

Celou práci jsem vytvářel jako učební pomůcku pro studenty seznamující se statistickou stochastickou metodou Monte Carlo. Proto výsledné programy mají především názorně ukazovat její využití. Pro seznámení se základní teorií k řešenému problému slouží odkazy na html stránku u každé aplikace.

# ZÁVĚR V ANGLIČTINĚ

For the making of this thesis I used the environment of webMathematica, which is available for the students at the Thomas Bata University in Zlín. I used the experiences got from the bachelor thesis. I learned to use the easy work with the graphs in this environment, which markedly demonstrated the results of the algorithms. The only one disadvantage in the utilization of the webMathematica is in a complicated debugging of the source code in case of the missing of the error messages. A look of the programs is chosen considering the suggested ranking of the links of the programs to the web aid for the subject Simulation of the systems. A theory for the programmed algorithms is included in the theoretical part of the thesis.

I have also programmed some specifying algorithms of the method Monte Carlo and made their comparison with the classical processes. For the realization of the algorithms I used the environment Mathematica. The results were according to the expections. The individual methods seemed to be suitable for a various kind of the function. It was shown that the utilization of the method for an unsuitable function is not always the right way how to reach a specifying of the result.

I created this thesis as a learning aid for a students being introduced in the statistic stochastic method Monte Carlo. That is the reason why the resulting programs should mainly clearly explain its utilization. For a familiarization with a base theory of a solved problem serve the links to the html sites by each aplication.

# SEZNAM POUŽITÉ LITERATURY

[1] FABIÁN, F, KLUIBER, Z. *Metoda Monte Carlo a možnosti jejího uplatnění*. Praha : Prospektum s.r.o., 1998. 148 s. ISBN 80-7175-058-1.

[2] HRABEC, Tomáš. *Řešení systému lineárních algebraických rovnic metodou Monte Carlo*. [s.l.], 2005. 40 s. Bakalářská práce.

[3] DANÍČEK, Ladislav. *Využití geometrické pravděpodobnosti při experimentálním ověření Ludolfova čísla*. [s.l.], 2005. 37 s. Bakalářská práce.

[4] BÖNISCH, Michal. *Využití statistické metody Monte Carlo pro výpočet určitých integrálů*. [s.l.], 2008. 31 s. Bakalářská práce.

[5] VIRIUS, M. *Aplikace matematické statistiky: metoda Monte Carlo.*. [s.l.] : [s.n.], 1998. 210 s. ISBN ISBN 80-01-01779-6

[6] KUPČÍK, Jakub. *Lineární binární bezpečnostní kódy*. [s.l.], 2008. 54 s. Bakalářská práce.

[7] SLABEJOVÁ, Danica. *Metóda Monte Carlo*. [s.l.], 2007. 38 s. Bakalářská práce.

[8] *Metoda Monte Carlo* [online]. 2005 [cit. 2009-04-24]. Dostupný z WWW: <http://artemis.osu.cz/pmfch/lekce10.pdf>.

[9] PRSKAVEC, Ladislav. *WebMathematica 2* [online]. 19.1.2005 [cit. 2007-02-10]. Dostupný z WWW: < http://k315.feld.cvut.cz/vyuka/webmathematica/>.

# SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PAS Počítačové algebraické systémy.

JSP Java server pages.

# SEZNAM OBRÁZKŮ

*Obr. 1. Jednotková kružnice* ..... *14*
*Obr. 2. Geometrická metoda* ..... *15*
*Obr. 3. Postup zpracování požadavku WebMathematicou* ..... *24*
*Obr. 4. Výsledek statického kódu* ..... *26*
*Obr. 5. Vzhled programu na určení Ludolfova čísla* ..... *29*
*Obr. 6. Teorie k programu na výpočet Ludolfova čísla* ..... *30*
*Obr. 7. Vzhled programu na výpočet určitých integrálů* ..... *32*
*Obr. 8. Návod pro zadávání vlastních funkcí* ..... *33*
*Obr. 9. Teorie k programu na výpočet integrálů* ..... *34*
*Obr. 10. Vzhled programu na výpočet dvojných integrálů* ..... *35*
*Obr. 11. Teorie k vícenásobným integrálům* ..... *36*
*Obr. 12. Návod pro zadávání funkcí u dvoj. integrálů* ..... *37*
*Obr. 13. Vzhled programu dvojitých integrálů druhá část* ..... *38*
*Obr. 14. Vzhled programu na výpočet soustavy lineárních rovnic* ..... *40*
*Obr. 15. Teorie k řešení soustav rovnic metodou Monte Carlo* ..... *41*
*Obr. 16. Symetrizace integrované funkce* ..... *44*
*Obr. 17. Graf první zkoumané funkce* ..... *45*
*Obr. 18. Graf druhé zkoumané funkce* ..... *47*
*Obr. 19. Graf třetí zkoumané funkce* ..... *48*
*Obr. 20. Graf ukázkové funkce* ..... *50*

# SEZNAM TABULEK

*Tab. 1. Srovnání zpřesňujících metod u první funkce* ..... *46*

*Tab. 2. Srovnání zpřesňujících metod u druhé funkce* ..... *47*

*Tab. 3. Výsledky zpřesňujících metod u třetí funkce* ..... *48*

*Tab. 4. Výsledky při použití metody hlavního výběru* ..... *51*

# SEZNAM PŘÍLOH

P I Zdrojový kód 1

P II Zdrojový kód 2

P III Zdrojový kód 3

P IV Zdrojový kód 4

P V Soubor metody_zpresneni.nb

# PŘÍLOHA P I: ZDROJOVÝ KÓD 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta HTTP-EQUIV="Content-type" Content="text/html; charset=UTF-8">
        <title>Urcení hodnoty Ludolfova čísla metodou Monte Carlo</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td>
<font color="#5070a0">
<h1><center>Určeni hodnoty Ludolfova čísla metodou Monte Carlo</center></h1>
<center><h4> Zadejte počet generovaných bodů (čím větší číslo, tim větší přesnost. Maximálně však 20000). Výsledek je pak vypočten
z poměru náhodně vygenerovaných bodů uvnitř a vně jednotkové kružnice.</h4></center>
</font><hr></center>
<form method="post" action="ludolf.jsp">
Pocet bodu:<br>
<INPUT type="text" name="bodu">
<input type="submit" value="Vypocitej">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadaný počet bodů:</b>
 </td>
 <td>
<msp:evaluate>
iterace=ToExpression[$$bodu,TraditionalForm];
MSPFormat[iterace]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
klad = {};
zapor = {};
vevnitr = 0;
If[And[IntegerQ[iterace],0<iterace<=20000],
For[i = 1, i < iterace,
  y1 = Random[Real, {-1, 1}];
  y2 = Random[Real, {-1, 1}];
  If[
     Power[y1,2] + Power[y2,2] <= 1, vevnitr++; klad = Append[klad, {y1, y2}];,
          zapor = Append[zapor, {y1, y2}];];
     i++;
       ];
pi = N[(4*vevnitr)/iterace];
procenta = Abs[pi - 3.141593]/(3.141593/100);,
pi="Zadejte počet bodů: 0<pocet_bodu<=20000";
];
</msp:evaluate>
<tr>
  <td>
   <b>Vypočtená hodnota Ludolfova čísla:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[pi]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Přesná hodnota Ludolfova čísla:</b>
  </td>
  <td>
```

```
<msp:evaluate>
MSPFormat[N[\[Pi],7]]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Procentuelní odchylka od přesné hodnoty:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[procenta]
</msp:evaluate>
  </td></tr>
</table></b>
<br>
Více informací o algoritmu můžete získat <a href="teorie_lud.html" onclick="return !window.open(this.href)">zde!</a>
</td>
<td>
<msp:evaluate>
graf0=Plot[Sqrt[-Power[x,2]+1],{x, -1, 1}, PlotLabel -> StyleForm["Graf generování bodu"], AspectRatio -> 1, ImageSize -> 400,
TextStyle->{FontSize->14,FontWeight->"Bold"}];
graf01=Plot[-Sqrt[-Power[x,2]+1],{x, -1, 1}, PlotLabel -> StyleForm["Graf generování bodu"]];
graf1=ListPlot[zapor,PlotStyle -> Hue[.9],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf generování bodu"]];
graf2=ListPlot[klad,PlotStyle -> Hue[.3],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf generování bodu"]];
MSPShow[Show[graf0,graf01,graf1,graf2]]
</msp:evaluate>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

# PŘÍLOHA P II: ZDROJOVÝ KÓD 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta HTTP-EQUIV="Content-type" Content="text/html; charset=UTF-8">
        <title>Výpočet určitého integrálu metodou Monte Carlo</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td colspan="2">
<font color="#5070a0">
<h1><center>Výpočet integrálu metodou Monte Carlo</center></h1>
<center><h4>Nejdříve vyberte nebo zadejte integrovanou funkci v požadovaných mezích. Poté zadejte hodnotu epsilon 0.01, 0.05 nebo
0.1 (ve výpočtech zastupuje odchylku) a hodnotu koeficientu 0.9 nebo 0.95 (Jedná se o koeficient spolehlivosti, který v procentech
zastupuje pravděpodobnost dosažení požadované odchylky).</h4></center>
</font></center>
</td></tr>
<tr>
<td>
<form method="post" action="integral2.jsp">
Vyber nebo zadejte funkci pro integraci:
<SELECT size="1" name="funkce">
  <OPTION  selected  value=x>x</OPTION>
  <OPTION  value=x+5>x+5</OPTION>
  <OPTION  value=Power[x,2]-2>x^2-2</OPTION>
  <OPTION  value=Power[x,2]-x-6>x^2-x-6</OPTION>
  <OPTION  value=Sin[x]>Sin(x)</OPTION>
  <OPTION  value=Sin[x]/2>Sin(x)/2</OPTION>
  <OPTION  value=Cos[x]>Cos(x)</OPTION>
  <OPTION  value=Sqrt[x]>Sqrt(x)</OPTION>
  <OPTION  value=Sqrt[x]/2>Sqrt(x)/2</OPTION>
 </SELECT><br>
Pokud chcete zadat vlastní funkci, podívejte se na <a href="navod.html" onclick="return !window.open(this.href)">návod!</a>
<INPUT type="text" name="funkce2"><br> <hr>
Zadej dolní a horní mez pro integraci (v rozmezí -10 až 10):<INPUT type="text" name="min" size=4> <INPUT type="text"
name="max" size=4><br>
epsilon:<br>
0.01<INPUT type="radio" name="epsilon" value=0.01 checked="true">
0.05<INPUT type="radio" name="epsilon" value=0.05>
0.1<INPUT type="radio" name="epsilon" value=0.1><br>
koeficient:<br>
0.9<INPUT type="radio" name="koeficient" value=0.9>
0.95<INPUT type="radio" name="koeficient" value=0.95 checked="true">
<input type="submit" value="Vypočítej">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadaná funkce:</b>
 </td>
 <td>
<msp:evaluate>
minimum=ToExpression[$$min,TraditionalForm];
maximum=ToExpression[$$max,TraditionalForm];
f=ToExpression[$$funkce,TraditionalForm];
If[$$funkce2!="",f=ToExpression[$$funkce2,TraditionalForm]];
eps=ToExpression[$$epsilon,TraditionalForm];
koef=ToExpression[$$koeficient,TraditionalForm];
MSPFormat[f]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
If[koef == 0.95, kvant = 1.64,
```

```
                If[koef == 0.9, kvant = 1.28, kvant = "Byla zadana
  spatna hodnota koeficientu"];] ;
pomoc = {};
mez1 = {};
mez2 = {};
protinaosu = 0;
If[And[IntegerQ[minimum],minimum>=-10,minimum<maximum],
If[And[IntegerQ[maximum],maximum<=10],
intervaly = {minimum, maximum};
koreny = Reduce[f == 0 && x < maximum && x > minimum , x, Reals];
koreny = {ToRules[koreny]};
koreny = ReplaceAll[x, koreny] // N;
For[i = 1, i <= Length[koreny],
If[Im[koreny[[i]]] == 0, protinaosu = 1;];
i++;];
If[protinaosu == 1,
For[i = 1, i < Length[koreny] + 1,
If[And[koreny[[i]] > minimum , koreny[[i]] < maximum],
intervaly = Append[intervaly, koreny[[i]]];
];
i++;
];];
intervaly = Sort[intervaly];
pocet_intervalu = Length[intervaly] - 1;
If[And[NumberQ[f /. x -> minimum // N],NumberQ[f /. x -> maximum // N],NumberQ[f /. x -> 1 // N], Im[f /. x -> minimum //
N]==0,Im[f /. x -> maximum // N]==0 ],
p = {};
s = {};
For[i = 1, i < Length[intervaly],
a = intervaly[[i]];
b = intervaly[[i + 1]];
sez = 0;
j = 0;
stred = ((b - a)/2) + a;
extrem = f /. x -> stred;
If[extrem < 0, extrem = Minimize[f, a <= x <= b,
     x], extrem = Maximize[f, a <= x <= b, x];];
While[j < 500,
  r1 = Random[];
  r2 = Random[];
  sx = ((b - a)*r1) + a;
sy = extrem[[1]]*r2;
    vysled = f /. x -> sx;
    If[extrem[[1]] > 0,
       If[sy > vysled, z = 3, sez++;];,
       If[sy < vysled, z = 3, sez++;];
       ];
    j++;
    ];
p = Append[p, sez/500];
    i++;
    ];
n = {};
For[i = 1, i <= Length[p],
n = Append[n,(Power[kvant,2]*p[[i]]*(1 - p[[i]]))/Power[eps,2]];
i++;
    ];
klad = {};
zapor = {};
p = {};
s = {};
For[i = 1, i < Length[intervaly],
  a = intervaly[[i]];
  b = intervaly[[i + 1]];
  mez1 = Append[mez1, a];
  mez2 = Append[mez2, b];
  sez = 0;
  j = 0;
  stred = ((b - a)/2) + a;
  extrem = f /. x -> stred;
If[extrem < 0, extrem = Minimize[f, a <= x <= b,
     x], extrem = Maximize[f, a <= x <= b, x];];
pomoc = Append[pomoc, extrem[[1]]];
```

```
While[j < (n[[i]] + 1),
  r1 = Random[];
  r2 = Random[];
  sx = ((b - a)*r1) + a;
sy = extrem[[1]]*r2;
    vysled = f /. x -> sx;
    If[extrem[[1]] > 0,
       If[sy > vysled, z = 3;
    zapor = Append[zapor, {sx, sy}], sez++; klad = Append[klad, {sx, sy}];];,
       If[sy < vysled, z = 3;
          zapor = Append[zapor, {sx, sy}], sez++; klad = Append[klad, {sx,sy}];];
       ];
    j++;
    ];
p = Append[p, sez/n[[i]]];
    i++;
    ];
soucet = 0;
For[i = 1, i <= Length[p],
    s = Append[s, p[[i]]*(mez2[[i]] - mez1[[i]])*pomoc[[i]]];
    i++;];
For[i = 1, i <= Length[p],
    soucet = soucet + s[[i]];
    i++;];
analyt=Integrate[f, {x, minimum, maximum}]//N;
n=Ceiling[n];
procenta = Abs[soucet - analyt]/(Abs[analyt]/100);,
soucet="Byla zadana funkce ve spatnem tvaru nebo funkci nelze spocitat v danych mezich.";
];,
soucet="Zadejte meze v rozmezí: -10 až 10";
];,
soucet="Zadejte meze v rozmezí: -10 až 10";
];
</msp:evaluate>


<tr>
  <td>
   <b>Zadané meze</b>
  </td>
  <td>
                        <table border=0>
                        <tr>
                        <td>
                        <msp:evaluate>
                                MSPFormat[minimum]
                        </msp:evaluate>
                        </td>
                        <td>
                        <msp:evaluate>
                                MSPFormat[maximum]
                        </msp:evaluate>
                        </td>
                        </tr>
                        </table>
  </td></tr>
<tr>
  <td>
   <b>Výsledné pravděpodobnosti:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[p]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Počet generovaných bodů pro jednotlivé intervaly:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[n]
</msp:evaluate>
```

```
  </td></tr>
<tr>
  <td>
   <b>Výsledná hodnota integrálu:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[soucet]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Hodnota určená analyticky:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[analyt]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Procentuelní odchylka od analytické hodnoty:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[procenta]
</msp:evaluate>
  </td></tr>
</table></b>
<br>
Více informací o algoritmu můžete získat <a href="teorie_integral.html" onclick="return !window.open(this.href)">zde!</a>
</td>
<td>
<msp:evaluate>
graf0=Plot[f,{x, minimum, maximum}, PlotLabel -> StyleForm["Graf vysetrovane funkce"],ImageSize -> 400, TextStyle->{FontSize-
>14,FontWeight->"Bold"}];
graf1=ListPlot[zapor,PlotStyle -> Hue[.9],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf
vysetrovane funkce"]];
graf2=ListPlot[klad,PlotStyle -> Hue[.3],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf
vysetrovane funkce"]];
MSPShow[Show[graf0,graf1,graf2]]
</msp:evaluate>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

# PŘÍLOHA P III: ZDROJOVÝ KÓD 3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta HTTP-EQUIV="Content-type" Content="text/html; charset=UTF-8">
        <title>Urceni hodnoty dvojitého určitého integrálu metodou Monte Carlo - geometrická metoda</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr>
<td colspan="2">
<font color="#5070a0">
<h1><center>Určení hodnoty dvojitého určitého integrálu metodou Monte Carlo - geometrická metoda</center></h1>
<center><h4> Zadejte počet generovaných bodů (čím větší číslo, tím větší přesnost)</h4></center>
</font></center>
</td></tr>
<tr><td>
<form method="post" action="dvoj_integral.jsp">
Počet bodů:<br>
<INPUT type="text" name="pokusy" value=5000>
<br>
Výber funkcí pro integraci:
<SELECT size="1" name="funkce">
   <OPTION  value=Abs[x]+Abs[y]>|x|+|y|</OPTION>
  <OPTION  value=Power[x,2]+Power[y,2]>x^2 + y^2</OPTION>
  <OPTION  value=Power[x*y,2]>(x*y)^2</OPTION>
  <OPTION  value=(Power[x,2]-10*Cos[6.28*x])+(Power[y,2]-10*Cos[6.28*y])+100>(x^2-10*Cos(6.28*x)+(y^2-
10*Cos(6.28*y)+100</OPTION>
  <OPTION  value=(-x*Sin[Sqrt[Abs[x]]])+(-y*Sin[Sqrt[Abs[y]]])+100>(-x*Sin(Sqrt(|x|)))+(-y*Sin(Sqrt(|y|)))+100</OPTION>
 </SELECT><br><br>
Zadej dolní a horní mez pro integraci u x (rozmezí -50 až 50):<INPUT type="text" name="min1" size=4> <INPUT type="text"
name="max1" size=4><br>
Zadej dolní a horní mez pro integraci u y (rozmezí -50 až 50):<INPUT type="text" name="min2" size=4> <INPUT type="text"
name="max2" size=4><br>
<input type="submit" value="Vypočítej">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadaný počet bodů:</b>
 </td>
 <td>
<msp:evaluate>
bodu=ToExpression[$$pokusy,TraditionalForm];
a1=ToExpression[$$min1,TraditionalForm];
b1=ToExpression[$$max1,TraditionalForm];
a2=ToExpression[$$min2,TraditionalForm];
b2=ToExpression[$$max2,TraditionalForm];
f=ToExpression[$$funkce,TraditionalForm];
<< Graphics`Graphics3D`;
MSPFormat[bodu]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
vyhov = 0;
klad = {};
zapor = {};
If[And[IntegerQ[bodu],0<bodu<=20000],
If[And[IntegerQ[a1],a1>=-50,a1<b1,IntegerQ[b1],b1<=50,IntegerQ[a2],a2>=-50,a2<b2,IntegerQ[b2],b2<=50],
spodek = Minimize[f, {a1 <= x <= b1, a2 <= y <= b2}, {x, y}]//N;
vrsek = Maximize[f, {a1 <= x <= b1, a2 <= y <= b2}, {x, y}]//N;
For[i = 1, i < bodu,
   rx = (b1 - a1)Random[] + a1;
   ry = (b2 - a2)*Random[] + a2;
```

```
   rz = (vrsek[[1]] - spodek[[1]])*Random[] + spodek[[1]]//N;
   If[rz < (f /. {x -> rx, y -> ry}//N), vyhov++; klad = Append[klad, {rx, ry,
   rz}];, zapor = Append[zapor, {rx, ry, rz}]];
      i++;
   ];
S = (b1 - a1)(b2 - a2)*(vrsek[[1]] - spodek[[1]]);
vysledek = S*vyhov/bodu + (b1 - a1)*(b2 - a2)*spodek[[1]]//N;
analyt=Integrate[f, {x, a1, b1}, {y, a2, b2}]//N;
procenta = Abs[vysledek - analyt]/(Abs[analyt]/100);,
vysledek="Zadávejte meze v rozmezí -50 az 50, dolní mez musí být menší než horní";
];,
vysledek="Zadejte počet bodů: 0<pocet_bodu<=20000";
];
</msp:evaluate>
<tr>
  <td>
   <b>Výsledna hodnota integrálu:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[vysledek]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Hodnota integrálu analyticky:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[analyt]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Procentuelní odchylka od analytické hodnoty:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[procenta]
</msp:evaluate>
  </td></tr>
</table></b>
<br>
Více informací o algoritmu můžete získat <a href="teorie_integral.html" onclick="return !window.open(this.href)">zde!</a>
</td>
<td>
<msp:evaluate>
graf0 = Plot3D[f, {x, a1, b1}, {y, a2, b2}, PlotLabel -> StyleForm["Graf generovani bodu"],ImageSize -> 400, TextStyle->{FontSize-
>14,FontWeight->"Bold"}];
graf1=ScatterPlot3D[zapor,PlotStyle -> Hue[.9],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf generovani bodu"]];
graf2=ScatterPlot3D[klad,PlotStyle -> Hue[.3],PlotStyle -> PointSize[0.02], PlotLabel -> StyleForm["Graf generovani bodu"]];
MSPShow[Show[graf0,graf1,graf2]]
</msp:evaluate>
</td>
</tr>
<tr>
<td colspan="2">
<font color="#5070a0">
<h1><center>Určení hodnoty dvojitého určitého integrálu metodou Monte Carlo pomocí odhadu střední hodnoty</center></h1>
<center><h4> Pokud chcete vypočítat dvojný integrál pro jinou funkci, můžete zde.
        Jelikož geometrická metoda by pro některé funkce nemusela pracovat správně, proběhne výpočet pomocí odhadu střední
hodnoty.
Zadejte počet opakování a Vaši funkci podle návodu.
</h4></center>
</font></center>
</td>
</tr>
<tr><td>
<form method="post" action="dvoj_integral.jsp">
Pocet bodu:<br>
<INPUT type="text" name="pokusy2" value=5000>
<br>
Před zadáváním funkce se podívejte na <a href="navod.html" onclick="return !window.open(this.href)">návod!</a>
```

```
<INPUT type="text" name="funkce2"><br> <hr>
Zadej dolní a horní mez pro integraci u x (rozmezí -50 až 50):<INPUT type="text" name="min3" size=4> <INPUT type="text"
name="max3" size=4><br>
Zadej dolní a horní mez pro integraci u y (rozmezí -50 až 50):<INPUT type="text" name="min4" size=4> <INPUT type="text"
name="max4" size=4><br>
<input type="submit" value="Vypočítej">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr>
 <td>
  <b>Zadaný počet opakování:</b>
 </td>
 <td>
<msp:evaluate>
bodu2=ToExpression[$$pokusy2,TraditionalForm];
a3=ToExpression[$$min3,TraditionalForm];
b3=ToExpression[$$max3,TraditionalForm];
a4=ToExpression[$$min4,TraditionalForm];
b4=ToExpression[$$max4,TraditionalForm];

f2=ToExpression[$$funkce2,TraditionalForm];
<< Graphics`Graphics3D`;
MSPFormat[bodu2]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
soucet = 0;
If[And[IntegerQ[bodu2],0<bodu2<=30000],
If[And[IntegerQ[a3],a3>=-50,a3<b3,IntegerQ[b3],b3<=50,IntegerQ[a4],a4>=-50,a4<b4,IntegerQ[b4],b4<=50],
If[And[NumberQ[f2 /. {x -> 1,y->1} // N],Not[StringFreeQ[ToString[f2], "y"]],Not[StringFreeQ[ToString[f2],"x"]] ],
i = 0;
soucet = 0;
While[i < bodu2,
    rx2 = ((b3 - a3)*Random[]) + a3;
    ry2 = ((b4 - a4)*Random[]) + a4;
    vysled = (b3 - a3)*(b4 - a4)*(f2 /. {x -> rx2, y -> ry2}//N);
    soucet = soucet + vysled;
    i++;
    ];
vysledek2 = soucet/bodu2;
analyt2=Integrate[f2, {x, a3, b3}, {y, a4, b4}]//N;
procenta2 = Abs[vysledek2 - analyt2]/(Abs[analyt2]/100);,
vysledek2="Zadali jste funkci ve špatném tvaru nebo neobsahuje x i y";
];,
vysledek2="Zadávejte meze v rozmezi -50 az 50, dolní mez musi být menši než horní";
];,
vysledek2="Zadejte počet bodů: 0<pocet_bodu<=30000";
];
</msp:evaluate>
<tr>
  <td>
   <b>Výsledná hodnota integralu:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[vysledek2]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Hodnota integrálu analyticky:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[analyt2]
</msp:evaluate>
  </td></tr>
<tr>
  <td>
   <b>Procentuelní odchylka od analytické hodnoty:</b>
  </td>
```

```
  <td>
<msp:evaluate>
MSPFormat[procenta2]
</msp:evaluate>
  </td></tr>
</table></b>
</td>
<td>
<msp:evaluate>
graf3 = Plot3D[f2, {x, a3, b3}, {y, a4, b4}, PlotLabel -> StyleForm["Graf vysetrovane funkce"],ImageSize -> 400, TextStyle-
>{FontSize->14,FontWeight->"Bold"}];
MSPShow[Show[graf3]]
</msp:evaluate>
</td>
</tr>
</table>
</body>
</msp:allocateKernel>
</html>
```

# PŘÍLOHA P IV: ZDROJOVÝ KÓD 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ tagliburi="/webMathematica-taglib" prefix="msp" %>
<html>
<head>
<meta HTTP-EQUIV="Content-type" Content="text/html; charset=UTF-8">
        <title>Řešení soustavy lineárních rovnic metodou Monte Carlo</title>
<meta name="generator" content="TSW WebCoder">
</head>
<msp:allocateKernel>
<body>
<table border=1 bordercolor="#5070a0">
<tr><td>
<font color="#5070a0">
<h1><center>Řešení soustavy lineárních rovnic metodou Monte Carlo</center></h1>
<center><h4> Zadejte počet iterací (čím větší číslo, tím větší přesnost). Dále zadejte soustavu rovnic ve tvaru Ax = b</h4></center>
</font><hr></center>
<form method="post" action="rovnice.jsp">
Počet bodů:<br>
<INPUT type="text" name="bodu" value=1000> <br>
Matice A: (Zadejte ve tvaru {{A11,A12,..},{A21,A22,..},..}, např. matici
<table border=0>
<tr><td>
A=
</td>
<td>
<table border=0>
        <tr><td>4</td><td>1</td> </tr>
        <tr><td>2</td><td>5</td> </tr>
</table>
</td></tr>
</table>
ve tvaru {{4,1},{2,5}}<br>


<INPUT type="text" name="maticeA" value="{{4,1},{2,5}}"><br>
Zadejte vektor b ve tvaru {b1,b2,..}<br>
<INPUT type="text" name="vektorB" value="{3,7}">
<input type="submit" value="Vypočítej">
</form>
<p>
<b>
<table border=1 bgcolor="#ECC50D">
<tr><td>
  <b>Zadaný počet iterací:</b>
 </td>
 <td>
<msp:evaluate>
opak=ToExpression[$$bodu,TraditionalForm];
A=ToExpression[$$maticeA,TraditionalForm];
B=ToExpression[$$vektorB,TraditionalForm];
MSPFormat[opak]
</msp:evaluate>
 </td></tr>
<msp:evaluate>
rozmery=Dimensions[A];
If[And[IntegerQ[opak],0<opak<=20000],
If[And[ArrayQ[A,_,NumberQ],rozmery[[1]]==rozmery[[2]],rozmery[[1]]<4,rozmery[[2]]<4],
If[And[VectorQ[B,NumberQ],Length[B]==rozmery[[1]]],
vysledek = {};
n = Length[B];
d = Tr[A, List];
c = d - 1;
zb = B/(c);
X = {};
Apom = A - DiagonalMatrix[c];
For[i = 1, i <= n,
   X = Append[X, Apom[[i]]/(-c[[i]])];
   i++;
```

```
    ];
vahy = IdentityMatrix[n + 1];
p = IdentityMatrix[n + 1];
podminka=0;
For[i = 1, i <= n,
    If[Total[Abs[X[[i]]]] >= 1, podminka = 1];
    p[[i]] = Append[X[[i]], 1 - Total[Abs[X[[i]]]]];
    i++;
    ];
p = Abs[p];
For[i = 1, i <= n,
    For[j = 1, j <= n,
     If[X[[i, j]] > 0, vahy[[i, j]] = 1;, vahy[[i, j]] = -1;];
     j++;];
    i++;];
vahy[[n + 1, n + 1]] = 0;
i = 0;
j = 0;
zb = Append[zb, 0];
If[podminka==0,
For[u = 1, u <= n,
    K = 0;
    maleK = 0;
    velkeX = 0;
    For[m = 1, m <= opak,
     y = Random[];
     i = u; Ypsilon = 1; velkeX = zb[[u]];
     While[i < n + 1,
      priznak = i;
      While[i == priznak,
       If[0 <= y < p[[priznak, 1]],
        j = 1; i = 1; y = Random[];,
        If[p[[priznak, 1]] <= y < p[[priznak, 1]] + p[[priznak, 2]], j = 2; i = 2; y = Random[];,
           If[p[[priznak, 1]] + p[[priznak, 2]] <= y < p[[priznak, 1]] + p[[priznak, 2]] + p[[priznak, 3]],
              i = 3; j = 3; y = Random[];,
              i = 4; j = 4; maleK = 4;
              ];
           ];  ];
       Ypsilon = Ypsilon*vahy[[priznak, j]];
       velkeX = velkeX + Ypsilon*zb[[j]];
       ];  ];
      K = K + velkeX;
     m++;
     ];
      u++;
    vysledek = Append[vysledek, 1/opak*K // N];
    ];
If[n==2,
presna=Solve[{A[[1,1]]x1 + A[[1,2]]x2 == B[[1]], A[[2,1]]x1 + A[[2,2]]x2 == B[[2]]}, {x1, x2}] // N; presna = {x1, x2} /. presna;,
presna=Solve[{A[[1,1]]x1 + A[[1,2]]x2+A[[1,3]]x3 == B[[1]], A[[2,1]]x1 + A[[2,2]]x2+A[[2,3]]x3 ==
B[[2]],A[[3,1]]x1+A[[3,2]]x2+A[[3,3]]x3==B[[3]]}, {x1, x2,x3}] // N;presna = {x1, x2,x3} /. presna;
];
presna=Flatten[presna];
procenta = Abs[vysledek - presna]/(Abs[presna]/100);,
vysledek="Nebyla splněna podmínka konvergence, výpočet nemohl proběhnout!!!"
];,
vysledek="Vektor b musí obsahovat jen reálná čísla a musí mít stejnou délku jako rozměry matice A";
];,
vysledek="Matice A musi být čtvercová (max 3x3) a musí obsahovat jen reálna čísla";
];,
vysledek="Zadejte počet bodů: 0<pocet_bodu<=20000";];
</msp:evaluate>
<tr><td>
  <b>Analyticky vypočítané hodnoty xi:</b>
 </td>
 <td>
<msp:evaluate>
MSPFormat[presna]
</msp:evaluate>
 </td></tr>
<tr><td>
  <b>Zjištěné hodnoty xi:</b>
 </td>
```

```
  <td>
<msp:evaluate>
MSPFormat[vysledek]
</msp:evaluate>
  </td></tr>
<tr><td>
   <b>Procentuelní odchylky výsledků od analytických hodnot:</b>
  </td>
  <td>
<msp:evaluate>
MSPFormat[procenta]
</msp:evaluate>
  </td></tr>
</table></b>
<br>
Více informací o algoritmu můžete získat <a href="teorie_rovnice.html" onclick="return !window.open(this.href)">zde!</a>
</td></tr></table>
</body>
</msp:allocateKernel>
</html>
```

# PŘÍLOHA P V: SOUBOR METODY_ZPRESNENI.NB

K práci je přiložen soubor metody_zpresneni.nb, který obsahuje naprogramované zpřesňující algoritmy metody Monte Carlo využité v experimentální části této práce (kapitola 4).