Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Bakalářská práce

Numerická integrace v matematickém sw

Plzeň, 2016

Nikola Kottová

Originální zadání

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

V Plzni dne .....

.....

Nikola Kottová

Poděkování

Na této stránce bych ráda poděkovala vedoucímu mé bakalářské práce Doc. Ing. Josefu Daňkovi, Ph.D. za obětavou práci a čas, který mi byl z jeho strany věnován.

Abstract

This document is focused on methods of numerical integration. The first part is theoretical, it describes the problems and methods of calculation based on Newton-Cotes and Gaussian quadrature formulas. In the next part there are described tested softwares, used methods and configuration options. There are also introduced tested functions. The next part focuses on comparation and evaluation obtained results. At first there are compared results of tests obtained in each software with different options. Thereafter follows overall evaluation of all used methods from all softwares. In the last part of this document there is outlined possibility of using precision arithmetic in numerical integration.

Keywords

Numerical integration, Newton-Cotes quadrature formula, Gaussian quadrature formula, Fortran, Matlab, Mathematica.

Anotace

Tato práce je zaměřena na metody numerické integrace. První část je teoretická, je zde popsaná problematika a metody výpočtu založené na Newtonových-Cotesových a Gaussových kvadraturních vzorcích. V následující části jsou popsané testované softwary, použité metody a možnosti nastavení. Také jsou zde představeny testované funkce. Následující část je zaměřena na porovnání a zhodnocení získaných výsledků. Nejprve jsou zde porovnány výsledky testů získaných v jednotlivých softwarech s různými nastaveními. Poté následuje celkové zhodnocení všech použitých metod ze všech softwarů. V poslední části této práce je nastíněna možnost využití přesnější aritmetiky při numerické integraci.

Klíčová slova

Numerická integrace, Newtonovy-Cotesovy kvadraturní vzorce, Gaussovy kvadraturní vzorce, Fortran, Matlab, Mathematica.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Numerická integrace je v matematice velmi často používaný aparát. Ne vždy je ale možné spočítat integrál funkce analyticky, nebo je analytické řešení příliš pracné. V takovém případě se daná funkce nahradí jinou, která jí je v nějakém vhodném smyslu podobná a přitom se snadno matematicky zpracovává nebo modeluje na počítači. Toto nahrazení se nazývá aproximace funkce. Existuje celá řada metod a vzorců pro aproximaci funkce, přičemž každý způsob je vhodný pro jiný druh výpočtu.

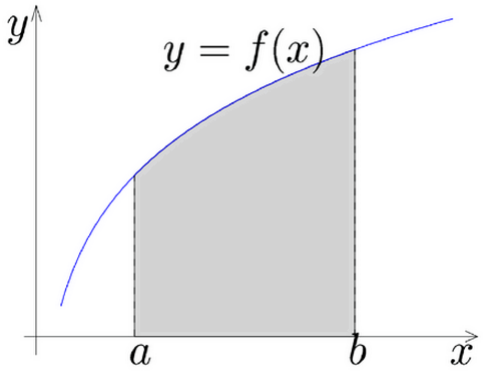
Cílem této práce je otestovat výpočet numerické integrace v matematických softwarech Matlab, Mathematica a v programovacím jazyce Fortran pomocí knihovny QUADPACK. První část této bakalářské práce je zaměřena na teoretické poznatky. Jsou zde uvedeny metody pro výpočet numerické integrace. Některé metody softwarů Matlab a Mathematica jsou založeny na Newtonových-Cotesových vzorcích. Mezi Newtonovy-Cotesovy vzorce patří obdélníkové pravidlo, lichoběžníkové pravidlo a Simpsonovo pravidlo. Jiným typem vzorců jsou Gaussovy kvadraturní vzorce. Jejich konstrukce je komplikovanější, ale dosahují vyššího algebraického řádu přesnosti. Na těchto vzorcích je založena většina metod testovaných softwarů.

Hlavní část této bakalářské práce je zaměřena na testování metod numerické integrace v softwarech Matlab, Mathematica a pomocí programovacího jazyku Fortran. Nejprve jsou dané softwary představeny a jsou uvedené různé možnosti nastavení. Poté jsou popsány jednotlivé metody použité pro numerickou integraci vybraných funkcí. Byly vybrány takové funkce, aby bylo možné otestovat integraci v komplexním oboru, výpočet nevlastních integrálů a integraci nespojitých funkcí s konečnými skoky. Následně jsou provedené testy integrace daných funkcí v uvedených matematických softwarech. Výpočty byly zaměřeny na přesnost výsledku, průměrný čas výpočtu a počet vyčíslení. Výsledky těchto testů jsou demonstrovány v následující kapitole. Získané výsledky jsou nejprve porovnané vzhledem k jednotlivým softwarům pro různá nastavení a parametry. Na základě získaných poznatků bylo pro každou metodu vyhodnoceno nejvhodnější nastavení. Poté následuje celkové porovnání všech výsledků získaných pomocí použitých metod s nejvhodněji nastavenými parametry. Poslední část této bakalářské práce je věnována finančnímu modelu, na kterém je demonstrován vliv použití přesnější aritmetiky při výpočtu numerické integrace.

Kapitola 1

Numerická integrace

Mějme na intervalu $\langle a, b\rangle$ zadanou integrovatelnou funkci $f = f(x)$. Pod pojmem určitý integrál $I(f) = \int_a^b f(x)\ \mathrm{d}x$, rozumíme obsah plochy, který je vymezen grafem funkce f, osou x a svislými přímkami $x = a$, $x = b$.



Obrázek 1.1: Určitý integrál [7]

Naším cílem je určit přibližnou hodnotu integrálu $I(f)$. Numerické metody výpočtu integrálu se používají především tehdy, když není možné spočítat integrál funkce analyticky, nebo je analytické řešení velmi pracné. V případě, že máme funkci zadanou tabulkou, není ani jiný přístup možný. Danou funkci f tedy nahrazujeme jinou funkcí φ, která je v nějakém vhodném smyslu funkci f podobná a přitom se snadno matematicky zpracovává nebo modeluje na počítači. Tuto funkci φ nazýváme aproximací funkce f.

Princip numerických metod pro výpočet integrálu tedy vychází z aproximace funkce. Danou funkci f nahradíme její vhodnou aproximací φ a jako aproximaci integrálu $I(f)$ prohlásíme hodnotu integrálu $I(\varphi)$

$$I(f) \approx I(\varphi) = \int_a^b \varphi(x)\ \mathrm{d}x. \tag{1.1}$$

Tato metoda je stabilní, protože je-li φ dobrou aproximací funkce f na intervalu $\langle a, b\rangle$, tzn. liší-li se od f nejvýše o malé ε, je integrál $I(\varphi)$ dobrou aproximací $I(f)$ a rozdíl integrálů si můžeme shora omezit výrazem

$$\left| \int_a^b f(x)\ \mathrm{d}x - \int_a^b \varphi(x)\ \mathrm{d}x \right| \leq \int_a^b \mid f(x) - \varphi(x) \mid\ \mathrm{d}x \leq (b-a)\varepsilon. \tag{1.2}$$

Princip většiny metod pro výpočet určitého integrálu $I(f) = \int_a^b f(x)\ \mathrm{d}x$ je založen na tom, že interval $\langle a, b\rangle$ rozdělíme na N dílčích podintervalů $\langle x_k, x_{k+1}\rangle$ tak, aby platilo

$$a = x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N = b. \tag{1.3}$$

Na daných podintervalech nahradíme funkci f polynomem a integrujeme tento polynom. Existuje mnoho vzorců pro nahrazení funkce f. U základních vzorců pro jednoduchost předpokládáme, že jsou podintervaly $\langle x_k, x_{k+1}\rangle$ stejně velké, tzv. ekvidistantní. Jiné vzorce předpokládají naopak neekvidistantní dělení podintervalů. U některých funkcí není nutné, aby každá část byla dělena na stejný počet uzlových bodů. Některé složitější části funkce vyžadují dělení na více podintervalů a to by u dobře aproximovatelných částí funkce bylo zbytečné. Tato metoda se nazývá adaptivní a řadí se ke zpřesňujícím metodám numerické integrace (viz dále) [7, 8].

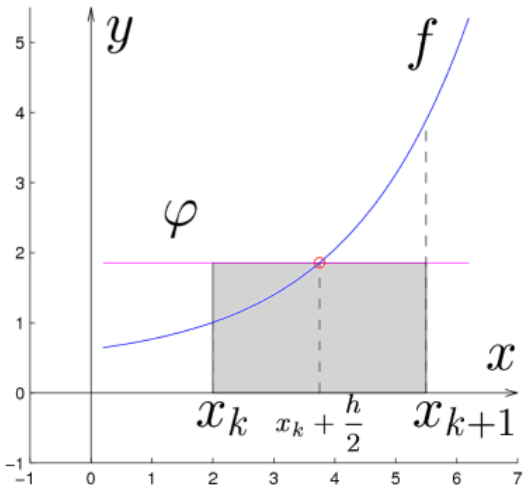
1.1 Newtonovy-Cotesovy kvadraturní vzorce

U Newtonových-Cotesových vzorců pro jednoduchost předpokládáme, že všechny podintervaly jsou stejně velké. Interval $\langle a, b\rangle$ tedy rozdělíme na N stejně velkých částí h, kde $h = \dfrac{b-a}{N} = konst$ a tím vytvoříme ekvidistantní uzlové body, které vyjádříme ve tvaru $x_k = x_0 + kh$, kde $k = 0, 1, \ldots, N-1$. Funkci f tedy aproximujeme funkcí φ na intervalu $\langle x_k, x_{k+1}\rangle$. Pro výpočet určitého integrálu můžeme sečíst základní kvadraturní vzorce a získáme tak vzorce složené, kde aplikujeme některou z metod nezávisle na každém podintervalu [3, 12].

1.1.1 Obdélníkové pravidlo

U obdélníkového pravidla funkci f aproximujeme na intervalu $\langle x_k, x_{k+1}\rangle$ konstantní funkcí φ v prostředním bodě intervalu, tedy $x_k + \dfrac{h}{2}$ a dostáváme

$$\int_{x_k}^{x_{k+1}} f(x)\ \mathrm{d}x \approx hf\left(x_k + \frac{h}{2}\right) + \underbrace{\frac{h^3}{24} f''(\xi)}_{\text{chyba}}. \tag{1.4}$$

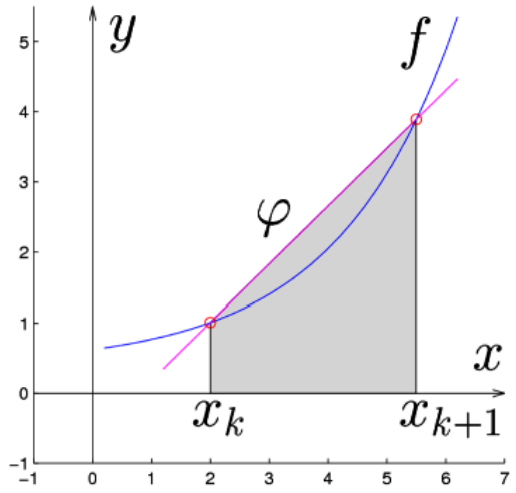


Obrázek 1.2: Obdélníkové pravidlo [7]

1.1.2 Lichoběžníkové pravidlo

Principem lichoběžníkového pravidla je nahrazení funkce f lineární funkcí φ. Tím získáme

$$\int_{x_k}^{x_{k+1}} f(x)\ \mathrm{d}x \approx \frac{h}{2}[f(x_k) + f(x_{k+1})] - \underbrace{\frac{h^3}{12} f''(\xi)}_{\text{chyba}}. \tag{1.5}$$

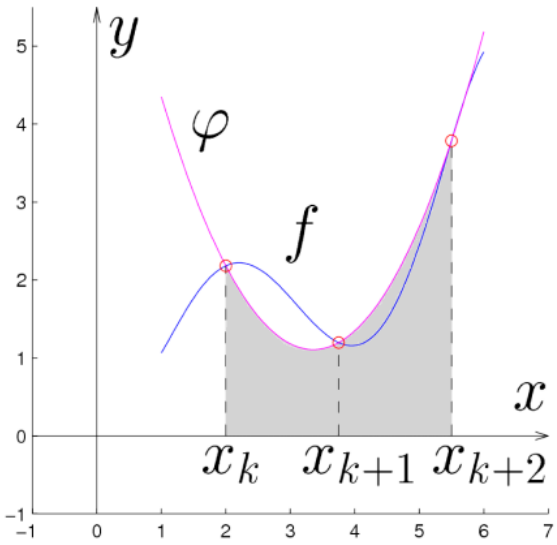


Obrázek 1.3: Lichoběžníkové pravidlo [7]

1.1.3 Simpsonovo pravidlo

U Simpsonova pravidla je funkce f nahrazena kvadratickou funkcí φ. Volíme tři uzlové body, dva na krajích intervalu a jeden uprostřed

$$\int_{x_k}^{x_{k+2}} f(x)\,\mathrm{d}x \approx \frac{h}{3}[f(x_k) + 4f(x_{k+1}) + f(x_{k+2})] - \underbrace{\frac{h^5}{90} f^{(4)}(\xi)}_{\text{chyba}}. \tag{1.6}$$



Obrázek 1.4: Simpsonovo pravidlo [7]

1.2 Gaussovy kvadraturní vzorce

Cílem kvadraturních vzorců je, abychom mohli přesně integrovat polynomy co možná nejvyššího řádu. Newtonovy-Cotesovy vzorce integrují přesně polynomy do stupně nejvýše m. Máme-li na intervalu $m+1$ bodů, potom pomocí Gaussových kvadraturních vzorců můžeme přesně integrovat polynomy stupně nejvýše $2m+1$. Cenou za vyšší přesnost budou ovšem neekvidistantní uzlové body. Základní Gaussův vzorce má tvar

$$K(f) = \sum_{i=0}^{m} w_i f(x_i), \tag{1.7}$$

kde w_i jsou tzv. váhy a x_i jsou uzly.

Maximální možné algebraické přesnosti je dosaženo správnou volbou vah a uzlů. Gaussovy vzorce dosahují tedy maximální možné přesnosti, ale oproti Newtonovým-Cotesovým vzorcům je obtížnější jejich odvození. Je nutné vyřešit nelineární soustavu rovnic, kde jsou váhy a uzly neznámými. Řešit tuto soustavu pro obecný interval $\langle a, b\rangle$ není vhodné, proto se základní Gaussovy vzorce odvozují na nějakém konkrétním intervalu, nejčastěji $\langle -1, 1\rangle$ a poté se na obecný interval transformují [9].

1.2.1 Odvození dvoubodového kvadraturního vzorce

Předvedeme si tedy odvození dvoubodového kvadraturního vzorce na intervalu $\langle -1, 1\rangle$, viz [1, 8]. V intervalu uvažujeme dva uzly, tzn. že $m = 1$ a vzorec má tedy tvar

$$K(f) = w_0 f(x_0) + w_1 f(x_1), \tag{1.8}$$

kde vystupují 4 neznámé w_0, w_1, $f(x_0)$, $f(x_1)$.

Vzorec musí přesně integrovat polynom až třetího stupně, tzn.

$$K(f) = w_0(ax_0^3 + bx_0^2 + cx_0 + d) + w_1(ax_1^3 + bx_1^2 + cx_1 + d). \tag{1.9}$$

Pro $a, b, c, d = 1$ získáme nelineární soustavu rovnic pro 4 neznámé

$$w_0 + w_1 = \int_{-1}^{1} 1 \,\mathrm{d}x = 2 \tag{1.10}$$

$$w_0 x_0 + w_1 x_1 = \int_{-1}^{1} x \,\mathrm{d}x = 0 \tag{1.11}$$

$$w_0 x_0^2 + w_1 x_1^2 = \int_{-1}^{1} x^2 \,\mathrm{d}x = \frac{2}{3} \tag{1.12}$$

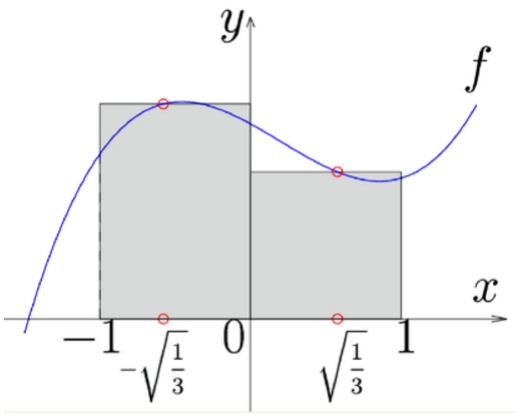
$$w_0 x_0^3 + w_1 x_1^3 = \int_{-1}^{1} x^3 \,\mathrm{d}x = 0. \tag{1.13}$$

Řešením této nelineární soustavy rovnic jsou hodnoty

$$w_0 = w_1 = 1,$$
$$x_0 = -\frac{1}{\sqrt{3}},$$
$$x_1 = \frac{1}{\sqrt{3}}.$$

Po dosazení získáme dvoubodový Gaussův vzorec ve tvaru

$$K(f) = \int_{-1}^{1} f(x) \,\mathrm{d}x = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) + \underbrace{\frac{1}{135} f^{(4)}(\xi)}_{\text{chyba}}. \tag{1.14}$$



Obrázek 1.5: Gaussovo dvoubodové pravidlo [7]

1.2.2 Kvadraturní vzorec pro obecný interval

Pro získání obecného kvadraturního vzorce na intervalu $\langle \alpha, \beta \rangle$

$$\int_{\alpha}^{\beta} f(x)\ \mathrm{d}x \approx \sum_{i=0}^{m} w_i f(x_i), \tag{1.15}$$

budeme aproximovat integrál přes interval $\langle a, b \rangle$

$$I(g) = \int_{a}^{b} g(t)\ \mathrm{d}t. \tag{1.16}$$

Nejprve musíme provést substituci, která bude transformovat x v intervalu $\langle \alpha, \beta \rangle$ na t v intervalu $\langle a, b \rangle$. K tomu použijeme lineární transformaci

$$t = \frac{(b-a)x + a\beta - b\alpha}{\beta - \alpha}. \tag{1.17}$$

Tím získáme integrál [1] ve tvaru

$$I(g) = \frac{b-a}{\beta-\alpha} \int_{\alpha}^{\beta} g\left(\frac{(b-a)x + a\beta - b\alpha}{\beta - \alpha}\right)\ \mathrm{d}x \approx \frac{b-a}{\beta-\alpha} \sum_{i=0}^{m} w_i g\left(\frac{(b-a)x_i + a\beta - b\alpha}{\beta - \alpha}\right). \tag{1.18}$$

1.2.3 Kvadraturní vzorce s využitím ortogonálních polynomů

Dalším způsobem, jak získat Gaussovy kvadraturní vzorce je využít ortogonální polynomy. Řekneme, že dva polynomy $p(x)$, $q(x)$ jsou na intervalu $\langle a, b \rangle$ ortogonální, jestliže platí

$$\int_{a}^{b} p(x)q(x)\ \mathrm{d}x = 0. \tag{1.19}$$

Nechť p je ortogonální ke všem polynomům stupně menšího než m na intervalu $\langle a, b\rangle$, potom lze dokázat, že platí

1. Polynom p má m různých reálných kořenů, které leží na otevřeném intervalu (a, b).
2. Při uzlech zvolených jako kořeny polynomu p je možné sestrojit kvadraturní vzorec, jehož váhy jsou řešením lineární soustavy rovnic. Tyto váhy jsou kladné a řád takového vzorce je $2m - 1$, získáme tedy jednoznačně určený m-bodový Gaussův kvadraturní vzorec.

Existuje celá řada takovýchto ortogonálních polynomů, pomocí kterých můžeme sestrojit Gaussův kvadraturní vzorec [10].

1.2.4 Gaussovy-Legendrovy kvadraturní vzorce

Uvažujme váhovou funkci $w(x) = 1$ na intervalu $I = \langle -1, 1\rangle$. Polynomy, které získáme ortogonalizací posloupnosti $1, x, x^2, ...$ vzhledem ke skalárnímu součinu

$$\langle u, v\rangle = \int_{-1}^{1} u(x)v(x)\ \mathrm{d}x, \tag{1.20}$$

jsou Legendrovy polynomy. Gaussovu kvadraturní formuli pro aproximaci integrálu

$$\int_{-1}^{1} f(x)\ \mathrm{d}x, \tag{1.21}$$

nazýváme Gaussova-Legendrova kvadraturní formule [4, 11].

1.2.5 Gaussovy-Čebyševovy kvadraturní vzorce

Uvažujme váhovou funkci $w(x) = \dfrac{1}{\sqrt{1 - x^2}}$ na intervalu $I = (-1, 1)$. Polynomy, které získáme ortogonalizací posloupnosti $1, x, x^2, ...$ vzhledem ke skalárnímu součinu

$$\langle u, v\rangle = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} u(x)v(x)\ \mathrm{d}x, \tag{1.22}$$

jsou Čebyševovy polynomy. Gaussovu kvadraturní formuli pro aproximaci integrálu

$$\int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x)\ \mathrm{d}x, \tag{1.23}$$

nazýváme Gaussova-Čebyševova kvadraturní formule [4, 11].

1.2.6 Gaussovy-Jacobiho kvadraturní vzorce

Uvažujme váhovou funkci $w(x) = (1-x)^\alpha(1+x)^\beta$, kde $\alpha, \beta > -1$ na intervalu $I = (-1, 1)$. Polynomy, které získáme ortogonalizací posloupnosti $1, x, x^2, ...$ vzhledem ke skalárnímu součinu

$$\langle u, v\rangle = \int_{-1}^{1} (1-x)^\alpha(1+x)^\beta u(x)v(x)\ \mathrm{d}x, \tag{1.24}$$

jsou Jacobiho polynomy. Gaussovu kvadraturní formuli pro aproximaci integrálu

$$\int_{-1}^{1} (1-x)^\alpha(1+x)^\beta f(x)\ \mathrm{d}x, \tag{1.25}$$

nazýváme Gaussova-Jacobiho kvadraturní formule [4, 11].

1.2.7 Gaussovy-Laguerrovy kvadraturní vzorce

Uvažujme váhovou funkci $w(x) = e^{-x}$ na intervalu $I = \langle 0, +\infty)$. Polynomy, které získáme ortogonalizací posloupnosti $1, x, x^2, ...$ vzhledem ke skalárnímu součinu

$$\langle u, v\rangle = \int_{0}^{\infty} e^{-x}u(x)v(x)\ \mathrm{d}x, \tag{1.26}$$

jsou Laguerrovy polynomy. Gaussovu kvadraturní formuli pro aproximaci integrálu

$$\int_{0}^{\infty} e^{-x}f(x)\ \mathrm{d}x, \tag{1.27}$$

nazýváme Gaussova-Laguerrova kvadraturní formule [4, 11].

1.2.8 Gaussovy-Hermitovy kvadraturní vzorce

Uvažujme váhovou funkci $w(x) = e^{-x^2}$ na intervalu $I = (-\infty, +\infty)$. Polynomy, které získáme ortogonalizací posloupnosti $1, x, x^2, ...$ vzhledem ke skalárnímu součinu

$$\langle u, v\rangle = \int_{-\infty}^{+\infty} e^{-x^2}u(x)v(x)\ \mathrm{d}x, \tag{1.28}$$

jsou Hermitovy polynomy. Gaussovu kvadraturní formuli pro aproximaci integrálu

$$\int_{-\infty}^{+\infty} e^{-x^2}f(x)\ \mathrm{d}x, \tag{1.29}$$

nazýváme Gaussova-Hermitova kvadraturní formule [4, 11].

1.2.9 Gaussovy-Kronrodovy kvadraturní vzorce

Princip této metody spočívá v tom, že jeden odhad integrálu spočítáme Gaussovým vzorcem a pro druhý odhad přidáme k použitým m uzlům dalších $m+1$ uzlů [13] a tím dostáváme výraz

$$GK = \sum_{i=0}^{m} a_i f(x_i) + \sum_{i=0}^{m+1} b_i f(x_i). \tag{1.30}$$

1.2.10 Gaussovy-Lobattovy kvadraturní vzorce

Princip tohoto kvadraturního vzorce je velmi podobný klasickému Gaussovu kvadraturnímu vzorci. Rozdíl je, že koncové body integračního intervalu bereme jako pevně dané uzly. Zbývá nám tedy $n-2$ uzlů a n vah jako volné parametry. Výsledná algebraická přesnost je $2n-3$ [14]. Gaussův-Lobattův kvadraturní vzorec má tedy tvar

$$GL = w_0 f(a) + w_m f(b) + \sum_{i=1}^{m-1} + w_i f(x_i). \tag{1.31}$$

1.3 Zpřesňující metody numerické integrace

1.3.1 Richardsonova metoda

Richardsonova extrapolace je obecná zpřesňující metoda, která se používá např. i u numerického derivování. Principem této metody je, že ze znalosti výrazu pro rozvoj chyby můžeme využít dvou přibližných výsledků a získat tak třetí, který bude přesnější. Slovo extrapolace se v názvu vyskytuje proto, že nová hodnota je lineární kombinací dvou hodnot, ale neleží mezi nimi (kdyby ano, tak bychom mluvili o interpolaci).

Předpokládejme, že výraz pro chybu má tvar

$$e(f) = h^k M, \text{ kde } h = \frac{b-a}{N}. \tag{1.32}$$

Přesná hodnota integrálu je tedy

$$I = K(h) + h^k M. \tag{1.33}$$

Integrál vypočítáme znovu stejným vzorcem, ale s krokem $\frac{h}{2}$. Dostaneme tedy výraz

$$I = K\left(\frac{h}{2}\right) + \left(\frac{h}{2}\right)^k M. \tag{1.34}$$

Výraz pro chybu s krokem $\frac{h}{2}$ označíme jako ε a vyjádříme si h^k

$$\varepsilon = \left(\frac{h}{2}\right)^k M, \tag{1.35}$$

$$h^k = \frac{\varepsilon 2^k}{M}. \tag{1.36}$$

Po dosazení h^k do výrazu (1.32), získáme

$$I = K(h) + \frac{\varepsilon 2^k M}{M}. \tag{1.37}$$

A ε vyjádříme z rovnic pro přesný integrál s kroky h a $\frac{h}{2}$ jako

$$\varepsilon \approx \frac{1}{2^k - 1}\left[K\left(\frac{h}{2}\right) - K(h)\right]. \tag{1.38}$$

Přesnější hodnota integrálu má tedy tvar

$$I = K\left(\frac{h}{2}\right) + \frac{1}{2^k - 1}\left[K\left(\frac{h}{2}\right) - K(h)\right], \tag{1.39}$$

kde k je řád eliminované chyby [2, 7].

1.3.2 Adaptivní metody

Dosud jsme se věnovali pouze ekvidistantnímu dělení intervalu, tzn., že interval jsme dělili na stejně velké podintervaly. U některých funkcí není nutné, aby každá část byla rozdělena na stejný počet uzlových bodů. Pokud je část intervalu $\langle a, b\rangle$ hladká a mění se pomalu, bude nám pro výpočet stačit hrubší dělení. Ale pokud je výpočet integrálu v dané části intervalu obtížný, je nutné použít jemnější dělení. Adaptivní metody přizpůsobují hustotu uzlů v závislosti na dané části intervalu. Rozdělení intervalů není určeno dopředu, určuje se na základě splnění testu chyby. Test je založen na základě odhadu pomocí metody poloviničního kroku.

Podinterval $\langle \alpha, \beta\rangle$ rozpůlíme a pro odhad chyby ε_f použijeme vzorec

$$\varepsilon_f(\alpha, \beta) \approx \frac{1}{2^k - 1}\left[I\left(\frac{h}{2}\right) - I(h)\right]. \tag{1.40}$$

Aby byl test chyby splněn, musí platit

$$\varepsilon_f(\alpha, \beta) \leq \varepsilon\frac{\beta - \alpha}{b - a}, \tag{1.41}$$

kde ε je požadovaná přesnost [2, 8] .

Kapitola 2

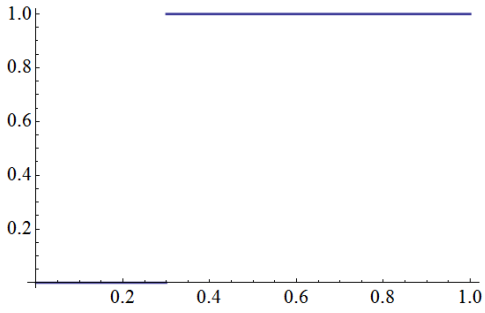
Matematický software

V této kapitole jsou představeny softwary pro výpočet numerické integrace, použité metody a možnosti nastavení. Výpočty jsou provedeny na testovacích funkcích v softwarech Matlab, Mathematica a v programovacím jazyce Fortran pomocí knihovny QUADPACK. U každého softwaru jsou popsány všechny metody použité k integraci, jejich možná nastavení a parametry. Výběr funkcí je inspirován článkem [15] a částečně upraven tak, aby funkce byly rozmanité a splňovaly požadovaná kritéria. Vybrali jsme funkce na integraci v komplexním oboru, výpočet nevlastních integrálů a integraci nespojitých funkcí s konečnými skoky.

2.1 Testovací funkce

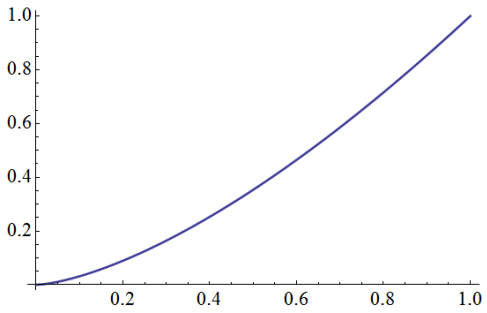
V matematických softwarech Matlab, Mathematica a programovacím jazyce Fortran jsou testovány následující funkce (všechny funkce jsou vykresleny na intervalu $\langle 0, 1 \rangle$).

1. $\int_0^1 f(x)\,\mathrm{d}x,$ kde $f = 1$ pokud $x > 0.3$ jinak $f = 0$



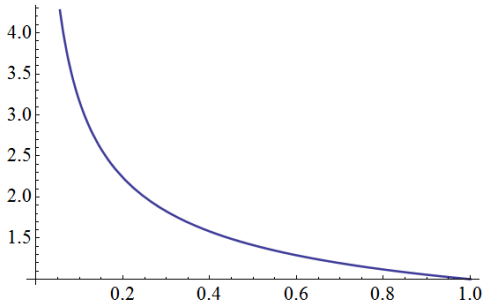
Obrázek 2.1: 1. funkce

2. $\int_0^1 \sqrt{x^3}\,\mathrm{d}x$



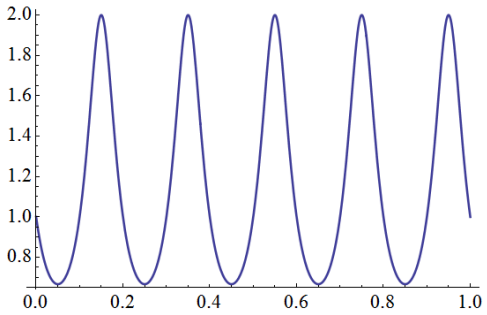
Obrázek 2.2: 2. funkce

3. $\int_0^1 \frac{1}{\sqrt{x}}\,\mathrm{d}x$



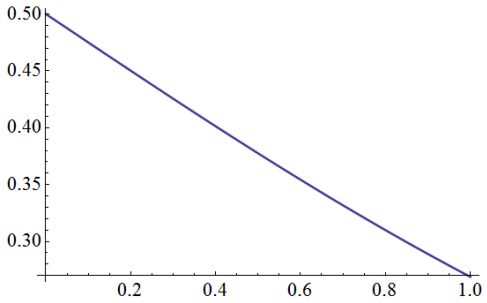
Obrázek 2.3: 3. funkce

4. $\int_0^1 \frac{2}{2+\sin(10\pi x)}\,\mathrm{d}x$



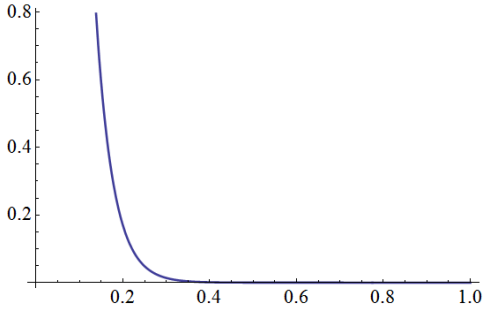
Obrázek 2.4: 4. funkce

5. $\int_0^1 \frac{1}{1+\mathrm{e}^x}\,\mathrm{d}x$



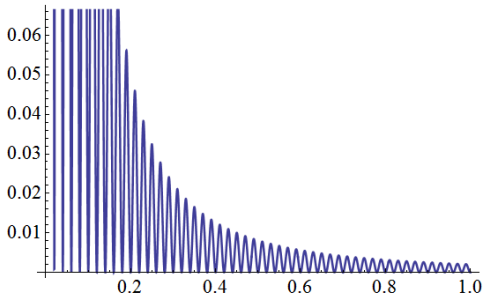
Obrázek 2.5: 5. funkce

6. $\int_0^{+\infty} 25\mathrm{e}^{-25x}\,\mathrm{d}x$



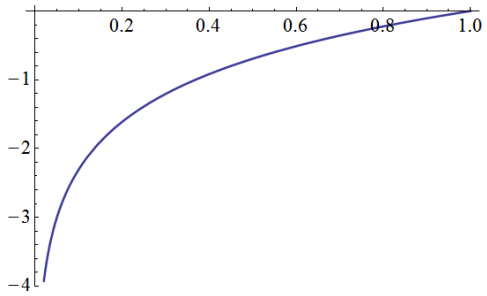
Obrázek 2.6: 6. funkce

7. $\int_{0.01}^1 50\left(\frac{\sin(50\pi x)}{50\pi x}\right)^2\,\mathrm{d}x$



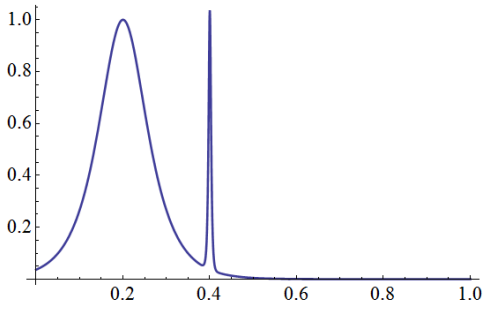
Obrázek 2.7: 7. funkce

8. $\int_0^1 f(x)\,\mathrm{d}x,$ kde $f = \log(x)$ pokud $x > 10^{-15}$ jinak $f = 0$



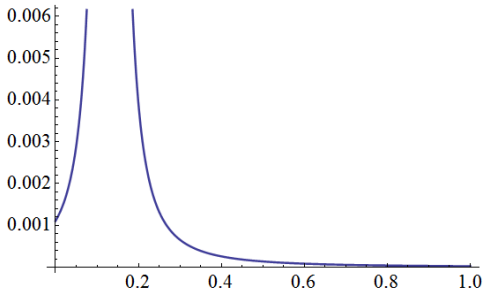
Obrázek 2.8: 8. funkce

9. $\int_0^1 \sum_{i=1}^{3} \frac{1}{\cosh(20^i(x - \frac{2i}{10}))}\,\mathrm{d}x$



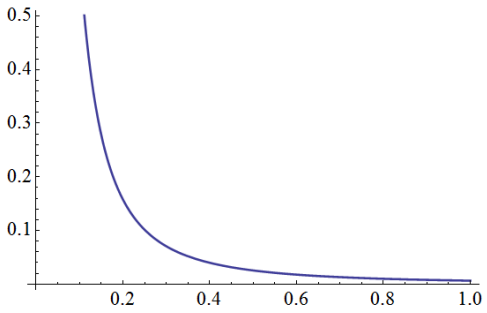
Obrázek 2.9: 9. funkce

10. $\int_0^1 \frac{1}{1 + (230x - 30)^2}\,\mathrm{d}x$



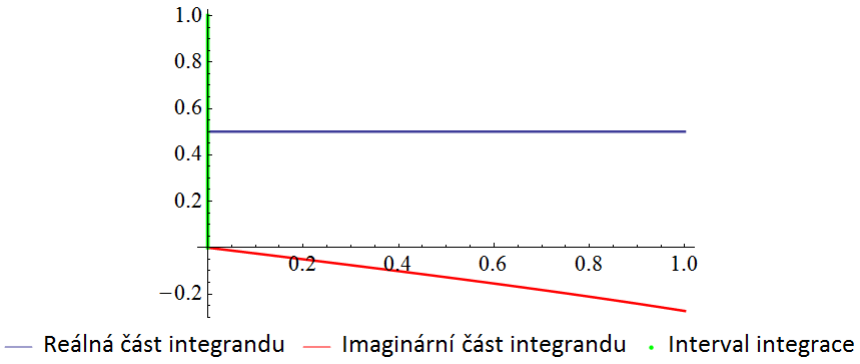
Obrázek 2.10: 10. funkce

11. $\int_0^{+\infty} \frac{50}{\pi(1+2500x^2)}\,\mathrm{d}x$



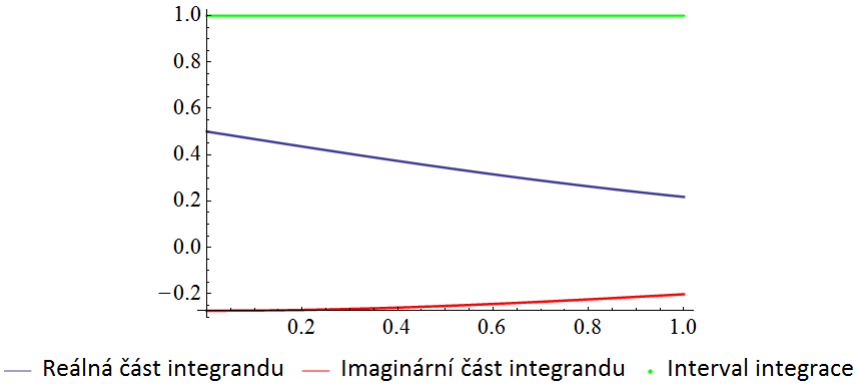
Obrázek 2.11: 11. funkce

12. $\int_0^{i} \frac{1}{1+\mathrm{e}^x}\,\mathrm{d}x$



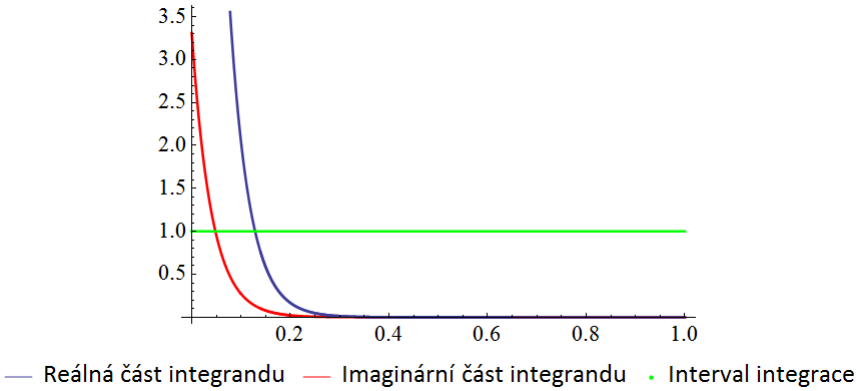
Obrázek 2.12: 12. funkce

13. $\int_{i}^{1+i} \frac{1}{1+\mathrm{e}^{x}}\,\mathrm{d}x$



Obrázek 2.13: 13. funkce

14. $\int_{i}^{+\infty+i} 25\mathrm{e}^{-25x}\,\mathrm{d}x$



Obrázek 2.14: 14. funkce

Tabulka 2.1: Přesné výsledky (s přesností formátu *double*) pro testované příklady

Funkce	Přesný výsledek
1	0.7
2	0.4
3	2
4	1.1547005383792515
5	0.3798854930417224
6	1
7	0.1121393035410214
8	-0.9999999999999644
9	0.1634949430186372
10	0.0134924856494677
11	0.5
12	$0.1305842404437227 + 0.5000000000000000\text{i}$
13	$0.3489928566738826 - 0.2472886217814119\text{i}$
14	$0.9912028118634830 + 0.1323517500977743\text{i}$

2.2 Fortran

Integrace funkcí pomocí programovacího jazyku Fortran 90 je umožněna díky překladači Intel Fortran Compiler, viz [21]. Metody pro výpočet numerické integrace jsou získány z knihovny QUADPACK, která je dostupná na [17]. Knihovna obsahuje různé metody pro výpočet určitého integrálu. Testované metody jsou založeny na Gaussově-Kronrodově pravidle. Použity jsou 15-ti, 21-ti, 31-ti, 41-ti, 51-ti a 61-ti bodové vzorce. Testovány jsou dva druhy knihoven s různými přesnostmi výpočtu - *double* a *real*. Metody s přesností *double* jsou od *realu* odlišeny písmenem *d*, např. $dqk15(double)$ a $qk15(real)$. Obě knihovny obsahují metody pro adaptivní i neadaptivní integraci. Pro výpočet integrandu na nekonečném intervalu je použita speciální metoda k tomu uzpůsobena - *dqagi*, resp. *qagi*. U ostatních metod není integrace na nekonečném intervalu možná. Daný test je ale možné uskutečnit s omezeným intervalem. Pro integraci v komplexním oboru nemá knihovna QUADPACK žádné metody. V jiných knihovnách se metody pro integraci v komplexním oboru jistě nacházejí, ale naším cílem bylo otestovat základní knihovnu QUADPACK. Bylo tedy nutné metody upravit, aby bylo možné otestovat dané funkce v komplexním oboru. Upravili jsme neadaptivní metody založené na Gaussových-Kronrodových vzorcích, např. metoda $cdqk15$.

Testujeme přesnost výsledku, průměrný čas výpočtu a počet vyčíslení. Rychlost výpočtu je nutné brát s rezervou. I když výpočet probíhal v cyklu (1000 opakování) a jedná se tedy o průměrnou hodnotu rychlosti výpočtu, tak jsou dané výsledky ovlivněny různými faktory, jako např. rychlostí výpočetního stroje. Metody umožňují různá nastavení parametrů. U adaptivních metod je možné nastavit relativní a absolutní přesnost. Při výpočtu je absolutní přesnost nastavena na hodnotu nula a relativní přesnost pro formát *double* na hodnoty $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$ a pro *real* na hodnoty 10^{-3} a 10^{-6}. Metody ve formátu *real* počítají přesně na sedm desetinných míst, vyšší hodnota relativní přesnosti by tedy neměla smysl.

Fortran 90 umožňuje nastavení různých optimalizací - $O0$, $O1$, $O2$ a $O3$. Parametr optimalizace $O0$ je výchozí nastavení a výpočet není optimalizací nijak ovlivněn. Optimalizace $O1$ umožňuje využít více paměti, ale dojde ke zvýšení časové náročnosti výpočtu. Nastavení optimalizace $O2$ zvyšuje rychlost výpočtu. Nastavení $O3$ je nejvyšší možný stupeň optimalizace. Toto nastavení umožňuje nejvyšší rychlost a přesnost výpočtu. Test je zaměřen na porovnání všech nastavení optimalizace v kombinaci s různými metodami a nastaveními přesnosti.

2.2.1 Přehled použitých metod

- $dqk15(qk15)$ - neadaptivní metoda využívající 15-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- $dqk21(qk21)$ - neadaptivní metoda využívající 21-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu

- *dqk*31(*qk*31) - neadaptivní metoda využívající 31-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- *dqk*41(*qk*41) - neadaptivní metoda využívající 41-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- *dqk*51(*qk*51) - neadaptivní metoda využívající 51-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- *dqk*61(*qk*61) - neadaptivní metoda využívající 61-ti bodové Gaussovo-Kronrodovo pravidlo, neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- *dqag*(*qag*) - adaptivní metoda využívající Gaussovo-Kronrodovo pravidlo (15-ti, 21-ti, 31-ti, 41-ti, 51-ti a 61-ti bodové), neumožňuje výpočet v komplexním oboru ani na nekonečném intervalu
- *dqagi*(*qags*) - adaptivní metoda využívající Gaussovo-Kronrodovo pravidlo, umožňuje výpočet na nekonečném intervalu, neumožňuje výpočet v komplexním oboru
- *cdqk*15 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 15-ti bodové Gaussovo-Kronrodovo pravidlo
- *cdqk*21 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 21-ti bodové Gaussovo-Kronrodovo pravidlo
- *cdqk*31 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 31-ti bodové Gaussovo-Kronrodovo pravidlo
- *cdqk*41 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 41-ti bodové Gaussovo-Kronrodovo pravidlo
- *cdqk*51 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 51-ti bodové Gaussovo-Kronrodovo pravidlo
- *cdqk*61 - upravená metoda pro výpočet integrálu v komplexním oboru, neadaptivní metoda využívající 61-ti bodové Gaussovo-Kronrodovo pravidlo

Podrobnější popis je k nalezení na [6, 16].

2.3 Matlab

Matematický software Matlab obsahuje řadu metod pro numerickou integraci. Metody jsou adaptivní i neadaptivní. Pro výpočet je možné nastavit požadovanou absolutní a relativní přesnost. Zadané funkce testujeme pro absolutní přesnost nastavenou na hodnotu nula a relativní přesnost pro hodnoty $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$. Některé metody mají nastavenou maximální možnou relativní přesnost, tzn., že požadované přesnosti 10^{-12} není dosaženo. Jedná se o metodu *quad*, která má maximální možnou relativní přesnost 10^{-11} a metodu *quadl*, která umožňuje dosáhnout maximální relativní přesnosti v řádu 10^{-9}.

Metody *quad* a *quadl* nejsou vhodné pro integraci na nekonečném intervalu. Opět je ale možné tento problém vyřešit omezením intervalu. Metody jsou vhodné i pro integraci v komplexním oboru. Funkce jsou testovány na přesnost výsledku, průměrný čas výpočtu a počet vyčíslení funkce. Rychlost výpočtu je nutné brát s rezervou. I když výpočet probíhal v cyklu (1000 opakování) a jedná se tedy o průměrnou hodnotu rychlosti výpočtu, tak jsou dané výsledky ovlivněné různými faktory, jako např. rychlostí výpočetního stroje.

2.3.1 Přehled použitých metod

- *trapz* - neadaptivní metoda využívající lichoběžníkové pravidlo, neumožňuje integraci na nekonečném intervalu, není možné nastavit absolutní ani relativní přesnost, důležitý parametr je jemnost dělení intervalu, kterou lze ovlivnit pomocí parametru krok h

- *quad* - adaptivní metoda využívající Simpsonovo pravidlo, maximální relativní přesnost je možné nastavit na 10^{-11}, neumožňuje výpočet na nekonečném intervalu, umožňuje výpočet v komplexním oboru

- *quadl* - adaptivní metoda využívající Gaussovo-Lobattovo pravidlo, maximální relativní přesnost je možné nastavit na 10^{-9}, neumožňuje výpočet na nekonečném intervalu, umožňuje výpočet v komplexním oboru

- *quadgk* - adaptivní metoda využívající 15-ti bodové Gaussovo-Kronrodovo pravidlo, umožňuje výpočet na nekonečném intervalu, umožňuje výpočet v komplexním oboru

- *integral* - adaptivní metoda využívající 15-ti bodové Gaussovo-Kronrodovo pravidlo, umožňuje výpočet na nekonečném intervalu, umožňuje výpočet v komplexním oboru

Podrobnější popis je k nalezení na [5, 18].

2.4 Mathematica

Dalším testovaným softwarem je Mathematica. Tento matematický software obsahuje několik metod pro numerickou integraci. Metody jsou testovány pro relativní přesnost nastavenou na hodnoty $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$. V Mathematice je možné nastavit pro každou metodu formu adaptivity - globální a lokální. U obou možností je interval rozdělen na dílčí podintervaly a na nich je daná funkce jednotlivě integrována. V případě globální adaptivity se výpočet zastaví, pokud je součet chyb ze všech podintervalů menší než požadovaná relativní přesnost. Principem je, že se dělí podinterval s největší chybou a tím se v součtu docílí dostatečně malé chyby výpočtu. U lokální adaptivity musí být test chyby splněn jednotlivě na dílčích podintervalech a dělení tedy probíhá na všech podintervalech, na kterých není chyba dostatečně malá. V testu jsou zahrnuty obě tyto varianty výpočtu.

Všechny funkce jsou testovány na přesnost výsledku, průměrný čas výpočtu a počet vyčíslení funkce. Rychlost výpočtu je nutné brát s rezervou. I když výpočet probíhal v cyklu (1000 opakování) a jedná se tedy o průměrnou hodnotu rychlosti výpočtu, tak jsou dané výsledky ovlivněny různými faktory, jako např. rychlostí výpočetního stroje. Všechny použité metody jsou vhodné pro výpočet integrálu funkce na nekonečném intervalu i v komplexním oboru. Název metody se odvíjí od výpočetního pravidla, na kterém je založena.

2.4.1 Přehled použitých metod

- *GaussKronrodRule* (GK) - adaptivní metoda využívající Gaussovo-Kronrodovo pravidlo, umožňuje výpočet na nekonečném intervalu i v komplexním oboru
- *NewtonCotesRule* (NC) - adaptivní metoda využívající Newtonovo-Cotesovo pravidlo, umožňuje výpočet na nekonečném intervalu i v komplexním oboru
- *LobattoKronrodRule* (LK) - adaptivní metoda využívající Lobattovo-Kronrodovo pravidlo, umožňuje výpočet na nekonečném intervalu i v komplexním oboru

Podrobnější popis je k nalezení na [20].

Kapitola 3

Test benchmarkových úloh

V této kapitole jsou zhodnocené vybrané výsledky testů integrace v matematických softwarech Matlab, Mathematica a v programovacím jazyce Fortran. Z důvodu velkého množství dat, jsou zde uvedené pouze vybrané výsledky. Veškeré získané výsledky jsou obsažené na přiloženém CD. Výpočty byly zaměřeny na přesnost výsledku, průměrný čas výpočtu a počet vyčíslení. Získané výsledky jsou nejprve porovnány vzhledem k jednotlivým softwarům pro různá nastavení a parametry. Na základě získaných poznatků bylo pro každou metodu vyhodnoceno nejvhodnější nastavení. Poté následuje celkové porovnání vybraných výsledků ze všech softwarů získaných pomocí použitých metod s nejvhodněji nastavenými parametry. Poslední část této kapitoly je věnována finančnímu modelu, na kterém je demonstrován vliv použití přesnější aritmetiky při výpočtu numerické integrace.

3.1 Fortran

V rámci programovacího jazyku Fortran porovnáváme nejprve výsledky neadaptivních metod vzhledem k nastavené optimalizaci ($O0$, $O1$, $O2$, $O3$) u vybrané funkce číslo 4, viz tabulka 3.1. Další zhodnocení je zaměřeno na celkový vliv optimalizace na všechny testované funkce, viz tabulka 3.2. Následuje srovnání adaptivních metod s již pevnou optimalizací $O3$. Výsledné hodnoty se porovnávají vzhledem k nastavení relativní přesnosti pro vybranou funkci číslo 8, viz tabulka 3.3. Následuje porovnání formátů *double* a *real* pro pevně nastavenou optimalizaci $O3$, relativní přesnost 10^{-6} a vybranou adaptivní metodu založenou na 61-ti bodovém Gaussově-Kronrodově vzorci, viz tabulka 3.4. Další srovnání je zaměřeno na adaptivní a neadaptivní metody pro vybranou 6. funkci testovanou na více intervalech, optimalizaci $O3$, relativní přesnost adaptivních metod 10^{-12} a vybrané metody založené na 15-ti, 31-ti a 61-ti bodových vzorcích, viz tabulka 3.5. Následují výsledky testů funkcí integrovaných na nekonečném intervalu pomocí metody *dqagi* a na omezeném intervalu $\langle 0, 1000\rangle$ pomocí adaptivních i neadaptivních metod využívajících Gaussovy-Kronrodovy vzorce, viz tabulky 3.6 a 3.7. Další srovnání je zaměřeno na test funkcí integrovaných v komplexním oboru (výpočet je proveden pomocí upravených metod), viz tabulky 3.8, 3.9 a 3.10.

3.1.1 Porovnání neadaptivních metod

V následující tabulce jsou výsledky testů integrace 4. funkce pomocí neadaptivních metod. Metody jsou založeny na Gaussově-Kronrodově pravidle s různým počtem bodů. Test ukazuje, že výsledky obecně získané neadaptivními metodami nejsou příliš přesné. Metody založené na vícebodových vzorcích mají ovšem v porovnání s metodami založenými na méně bodových pravidlech vyšší přesnost výpočtu. Metody využívající pravidla s použitím méně bodů mají ale rychlejší výpočet. Dané metody byly testovány i vzhledem k použité optimalizaci. Z výsledného srovnání je viditelné, že použití libovolné optimalizace ($O1$, $O2$, $O3$) má stejné zlepšení přesnosti výsledků i časové náročnosti výpočtu.

Tabulka 3.1: (Fortran) Test neadaptivních metod pro různé optimalizace na konečném intervalu, integrace 4. funkce, formát *double*

Metoda	Optimalizace	Výsledek	Chyba	Čas $[s]$
dqk15	O0	1.06214822007464	$9.255231830460997.10^{-2}$	$9.375.10^{-7}$
	O1	1.06214821947875	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
	O2	1.06214821947875	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
	O3	1.06214821947875	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
dqk21	O0	1.15364525058378	$1.055287795466819.10^{-3}$	$1.563.10^{-6}$
	O1	1.15364526209477	$1.055276284482876.10^{-3}$	$1.094.10^{-6}$
	O2	1.15364526209477	$1.055276284482876.10^{-3}$	$9.375.10^{-7}$
	O3	1.15364526209477	$1.055276284482876.10^{-3}$	$9.375.10^{-7}$
dqk31	O0	1.15964612049922	$4.945582119968872.10^{-3}$	$2.031.10^{-6}$
	O1	1.15964612116378	$4.945582784531943.10^{-3}$	$1.406.10^{-6}$
	O2	1.15964612116378	$4.945582784531943.10^{-3}$	$1.406.10^{-6}$
	O3	1.15964612116378	$4.945582784531943.10^{-3}$	$1.406.10^{-6}$
dqk41	O0	1.15496902599292	$2.684876136691550.10^{-4}$	$2.500.10^{-6}$
	O1	1.15496902798783	$2.684896085780597.10^{-4}$	$1.875.10^{-6}$
	O2	1.15496902798783	$2.684896085780597.10^{-4}$	$1.719.10^{-6}$
	O3	1.15496902798783	$2.684896085780597.10^{-4}$	$1.875.10^{-6}$
dqk51	O0	1.15466496377492	$3.557460433634141.10^{-5}$	$3.125.10^{-6}$
	O1	1.15466496618834	$3.557219090666308.10^{-5}$	$2.343.10^{-6}$
	O2	1.15466496618834	$3.557219090666308.10^{-5}$	$2.188.10^{-6}$
	O3	1.15466496618834	$3.557219090666308.10^{-5}$	$2.188.10^{-6}$
dqk61	O0	1.15472641381172	$2.587543247223323.10^{-5}$	$3.750.10^{-6}$
	O1	1.15472641615169	$2.587777243645561.10^{-5}$	$2.813.10^{-6}$
	O2	1.15472641615169	$2.587777243645561.10^{-5}$	$2.813.10^{-6}$
	O3	1.15472641615169	$2.587777243645561.10^{-5}$	$2.656.10^{-6}$

3.1.2 Vliv optimalizace

Vliv optimalizace nemá bohužel tak dobré výsledky, jak jsme očekávali. Z následující tabulky 3.2 je zřejmé, že ve většině případů není vliv optimalizace znatelný. Pouze u dvou testovaných funkcí je při použití libovolné optimalizace ($O1$, $O2$, $O3$) dosaženo vyšší přesnosti výsledků. Rychlost výpočtu se ve většině případů zlepšila, ale neplatí to vždy. Pro námi testované funkce se tedy vliv optimalizace příliš nepotvrdil.

Tabulka 3.2: (Fortran) Test vlivu optimalizace pro různé funkce na konečném intervalu, neadaptivní metoda *dqk*15, formát *double*

Funkce	Optimalizace	Výsledek	Chyba	Čas $[s]$
1. funkce	O0	0.654587005308831	$4.541299469116855.10^{-2}$	$7.813.10^{-7}$
	O1	0.654587005308831	$4.541299469116855.10^{-2}$	$4.688.10^{-7}$
	O2	0.654587005308831	$4.541299469116855.10^{-2}$	$4.688.10^{-7}$
	O3	0.654587005308831	$4.541299469116855.10^{-2}$	$4.688.10^{-7}$
2. funkce	O0	0.399999982866058	$1.713394215396846.10^{-8}$	$7.813.10^{-7}$
	O1	0.399999982866058	$1.713394215396846.10^{-8}$	$6.250.10^{-7}$
	O2	0.399999982866058	$1.713394215396846.10^{-8}$	$6.250.10^{-7}$
	O3	0.399999982866058	$1.713394215396846.10^{-8}$	$1.094.10^{-6}$
3. funkce	O0	1.954321589568490	$4.567841043151000.10^{-2}$	$9.375.10^{-7}$
	O1	1.954321589568490	$4.567841043151000.10^{-2}$	$6.250.10^{-7}$
	O2	1.954321589568490	$4.567841043151000.10^{-2}$	$4.688.10^{-7}$
	O3	1.954321589568490	$4.567841043151000.10^{-2}$	$7.813.10^{-7}$
4. funkce	O0	1.062148220074640	$9.255231830460997.10^{-2}$	$9.375.10^{-7}$
	O1	1.062148219478750	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
	O2	1.062148219478750	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
	O3	1.062148219478750	$9.255231890049864.10^{-2}$	$6.250.10^{-7}$
5. funkce	O0	0.379885493041722	$5.551115123125783.10^{-17}$	$9.375.10^{-7}$
	O1	0.379885493041722	$5.551115123125783.10^{-17}$	$6.250.10^{-7}$
	O2	0.379885493041722	$5.551115123125783.10^{-17}$	$6.250.10^{-7}$
	O3	0.379885493041722	$5.551115123125783.10^{-17}$	$6.250.10^{-7}$
7. funkce	O0	0.103470288041285	$8.669015499736213.10^{-3}$	$1.250.10^{-6}$
	O1	0.103470279598376	$8.669023942645418.10^{-3}$	$7.813.10^{-6}$
	O2	0.103470279598376	$8.669023942645418.10^{-3}$	$7.813.10^{-6}$
	O3	0.103470279598376	$8.669023942645418.10^{-3}$	$7.813.10^{-6}$

Funkce	Optimalizace	Výsledek	Chyba	Čas $[s]$
8. funkce	O0	-0.998307326901658	$1.692673098306563.10^{-3}$	$9.375.10^{-7}$
	O1	-0.998307326901658	$1.692673098306563.10^{-3}$	$6.250.10^{-7}$
	O2	-0.998307326901658	$1.692673098306563.10^{-3}$	$6.250.10^{-7}$
	O3	-0.998307326901658	$1.692673098306563.10^{-3}$	$6.250.10^{-7}$
9. funkce	O0	0.198355385320272	$3.486044230163440.10^{-2}$	$2.500.10^{-6}$
	O1	0.198355475102177	$3.486053208353967.10^{-2}$	$3.906.10^{-6}$
	O2	0.198355475102177	$3.486053208353967.10^{-2}$	$3.906.10^{-6}$
	O3	0.198355475102177	$3.486053208353967.10^{-2}$	$3.906.10^{-6}$
10. funkce	O0	$6.606412979949390.10^{-2}$	$5.257164415002620.10^{-2}$	$7.813.10^{-7}$
	O1	$6.606412979949390.10^{-2}$	$5.257164415002620.10^{-2}$	$4.688.10^{-7}$
	O2	$6.606412979949390.10^{-2}$	$5.257164415002620.10^{-2}$	$4.688.10^{-7}$
	O3	$6.606412979949390.10^{-2}$	$5.257164415002620.10^{-2}$	$4.688.10^{-7}$

3.1.3 Porovnání adaptivních metod

V následující tabulce 3.3 jsou porovnané výsledky získané adaptivními metodami s použitím různých nastavení relativní přesnosti. Optimalizace je v tomto případě již pevně nastavená na $O3$. Vliv relativní přesnosti je při výpočtu velmi znatelný. S vyšší relativní přesností roste i přesnost výsledných hodnot. Zlepšení je dosaženo u všech testovaných metod. S vyšší relativní přesností ovšem také roste počet vyčíslení a délka výpočtu. Vliv použité metody nemá v tomto případě na výsledné hodnoty žádný vliv.

Tabulka 3.3: (Fortran) Test adaptivních metod pro různé nastavení relativní přesnosti, integrace 8. funkce, optimalizace $O3$, formát *double*

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyčíslení
dqag15	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$1.563.10^{-5}$	615
	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$1.563.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$1.563.10^{-5}$	1215
dqag21	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$3.125.10^{-5}$	615
	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$4.688.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$6.250.10^{-5}$	1215
dqag31	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$3.125.10^{-5}$	615

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyčíslení
dqag31	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$4.688.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$6.250.10^{-5}$	1215
dqag41	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$4.688.10^{-5}$	615
	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$4.688.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$6.250.10^{-5}$	1215
dqag51	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$4.688.10^{-5}$	615
	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$4.688.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$6.250.10^{-5}$	1215
dqag61	10^{-3}	-0.999998346998927	$1.653001036938662.10^{-6}$	$1.563.10^{-5}$	315
	10^{-6}	-0.999999998385741	$1.614223088530764.10^{-9}$	$3.125.10^{-5}$	615
	10^{-9}	-0.999999999998424	$1.540434446667405.10^{-12}$	$4.688.10^{-5}$	915
	10^{-12}	-0.999999999999999	$3.430589146091734.10^{-14}$	$6.250.10^{-5}$	1215

3.1.4 Porovnání formátů double a real

Použitý formát má na výsledné hodnoty samozřejmě velký vliv. Výpočetní přesnost formátu *double* je na 15 platných cifer a formátu *real* na 7 platných cifer. V některých případech bylo dosaženo vyšší přesnosti, než je zmíněný počet platných číslic. Tyto výsledky je nutné brát s rezervou, jedná se pouze o výpis na vyšší počet cifer. Ve většině případů má formát *double* řádově přesnější výsledky výpočtu. Formát *real* má ale nižší počty vyčíslení a nižší časovou náročnost výpočtu.

Tabulka 3.4: (Fortran) Porovnání formátů *double* a *real* pro různé funkce (na konečném intervalu), relativní přesnost 10^{-6}, optimalizace $O3$, metody *dqag*61 a *qag*61

Fce	Metoda	Výsledek	Chyba	Čas $[s]$	Vyčíslení
1. fce	dqag61	0.699999967322676	$3.267732395784151.10^{-8}$	$2.188.10^{-5}$	615
	qag61	0.6999999	$5.9604645.10^{-8}$	$1.563.10^{-5}$	357
2. fce	dqag61	0.399999999905347	$9.465256356477880.10^{-11}$	$4.688.10^{-6}$	105
	qag61	0.4000000	$2.9802322.10^{-8}$	$7.812.10^{-7}$	21
3. fce	dqag61	1.99999991287535	$8.712465371374378.10^{-8}$	$4.688.10^{-5}$	1155
	qag61	2.000000	$4.7683716.10^{-7}$	$1.563.10^{-5}$	231
4. fce	dqag61	1.15470053407436	$1.641993851997370.10^{-8}$	$3.125.10^{-5}$	405
	qag61	1.154701	$4.7683716.10^{-7}$	$1.563.10^{-5}$	315

Fce	Metoda	Výsledek	Chyba	Čas $[s]$	Vyčíslení
5. fce	dqag61	0.379885493041722	$1.667292415241661.10^{-9}$	$7.813.10^{-7}$	15
	qag61	0.3798855	$< 10^{-7}$	$9.375.10^{-7}$	21
7. fce	dqag61	0.112139299511450	$7.451040562322930.10^{-9}$	$6.250.10^{-5}$	1035
	qag61	0.1121393	$7.4505806.10^{-9}$	$6.250.10^{-5}$	1281
8. fce	dqag61	-0.999999998385741	$1.614223088530764.10^{-9}$	$3.125.10^{-5}$	615
	qag61	-0.9999999	$1.1920929.10^{-7}$	$3.250.10^{-5}$	441
9. fce	dqag61	0.163102243938141	$3.926990804959951.10^{-4}$	$8.813.10^{-5}$	315
	qag61	0.1631022	$3.9270520.10^{-4}$	$2.188.10^{-4}$	735
10. fce	dqag61	$1.349248564946776.10^{-2}$	$6.418476861114186.10^{-17}$	$1.250.10^{-5}$	345
	qags61	$1.3492486.10^{-2}$	$1.8626451.10^{-9}$	$2.813.10^{-5}$	819

3.1.5 Porovnání adaptivních a neadaptivních metod

V následující tabulce 3.5 jsou porovnané výsledky získané pomocí adaptivních a neadaptivních metod. Test byl proveden s pevným nastavením optimalizace $O3$. Adaptivní metody mají prokazatelně přesnější výsledky. Test je proveden pro funkci číslo 6 pro různé délky intervalu. Přesnost výsledku je porovnávaná vždy s přesnou hodnotou (s přesností formátu *double*) integrálu funkce na daném intervalu. Vyšší přesnost výsledků adaptivních metod je znatelnější se zvětšujícím se intervalem. Na intervalu $\langle 0, 100\rangle$ dochází k úplnému selhání neadaptivních metod, které využívají 15-ti a 31-ti bodové vzorce. Na intervalu $\langle 0, 1000\rangle$ dochází k selhání i 61-ti bodového vzorce. Adaptivní metody mají naopak se zvětšujícím se intervalem přesnější výsledky. Neadaptivní metody mají nižší časovou náročnost výpočtu, ale vzhledem k nepřesným výsledkům můžeme toto kritérium zanedbat.

Tabulka 3.5: (Fortran) Porovnání adaptivních a neadaptivních metod (15-ti, 31-ti, 61-ti bodové vzorce), integrace 6. funkce na různých intervalech, relativní přesnost adaptivních metod 10^{-12}, optimalizace $O3$, formát *double*

Interval	Metoda	Výsledek	Chyba	Čas $[s]$
$\langle 0, 1\rangle$	dqag15	0.999999999986112	$3.996802888650563.10^{-15}$	$6.250.10^{-6}$
	dqk15	1.000000000468280	$4.821640864349774.10^{-10}$	$6.250.10^{-7}$
	dqag31	0.999999999986112	$3.996802888650563.10^{-15}$	$6.250.10^{-6}$
	dqk31	0.999999999986112	$3.996802888650563.10^{-15}$	$1.875.10^{-6}$
	dqag61	0.999999999986112	$3.996802888650563.10^{-15}$	$6.250.10^{-6}$
	dqk61	0.999999999986112	$3.996802888650563.10^{-15}$	$2.343.10^{-6}$

Interval	Metoda	Výsledek	Chyba	Čas $[s]$
$\langle 0, 10\rangle$	dqag15	1.000000000000000	$2.220446049250313.10^{-16}$	$1.406.10^{-5}$
	dqk15	0.998886298720673	$1.113701279326862.10^{-3}$	$1.094.10^{-6}$
	dqag31	1.000000000000000	$2.220446049250313.10^{-16}$	$1.250.10^{-5}$
	dqk31	1.000021669953030	$2.166995303176655.10^{-5}$	$1.250.10^{-6}$
	dqag61	1.000000000000000	$2.220446049250313.10^{-16}$	$1.250.10^{-5}$
	dqk61	0.999999999999997	$3.330669073875470.10^{-15}$	$2.343.10^{-6}$
$\langle 0, 100\rangle$	dqag15	1.000000000000000	$< 10^{-15}$	$2.188.10^{-5}$
	dqk15	$6.588831326799394.10^{-44}$	0.999341116867320	$1.719.10^{-6}$
	dqag31	1.000000000000000	$< 10^{-15}$	$2.188.10^{-5}$
	dqk31	0.553356417730364	0.446643582269636	$2.188.10^{-6}$
	dqag61	1.000000000000000	$< 10^{-15}$	$2.031.10^{-5}$
	dqk61	1.011769833758270	$1.176983375827167.10^{-2}$	$4.219.10^{-6}$
$\langle 0, 1000\rangle$	dqag15	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
	dqk15	$1.178555945780107.10^{-44}$	1.000000000000000	$9.375.10^{-7}$
	dqag31	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
	dqk31	$9.607394679090523.10^{-10}$	0.999999999039261	$2.031.10^{-6}$
	dqag61	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
	dqk61	$2.758310590461750.10^{-2}$	0.972416894095382	$3.750.10^{-6}$

3.1.6 Test integrace funkcí na nekonečném intervalu

Předešlé metody nejsou uzpůsobeny pro výpočet integrace na nekonečném intervalu. Proto je horní hranice intervalu omezena a funkce jsou testovány pro dané metody na konečném intervalu $\langle 0, 1000\rangle$. U 6. funkce je rozdíl téměř neznatelný, protože absolutní hodnota integrálu na intervalu $\langle 1000, +\infty)$ je řádově 10^{-10858}. Pro 11. funkci je absolutní hodnota integrálu na intervalu $\langle 1000, +\infty)$ řádově 10^{-6}.

Knihovna *QUADPACK* obsahuje ale i metody zaměřené speciálně na výpočet integrálu na nekonečném intervalu. Pro test byla použita metoda *dqagi* a její srovnání s ostatními metodami je zobrazeno v následujících tabulkách. V tabulce 3.6 jsou výsledky testu 6. funkce. Je zřejmé, že neadaptivní metody v tomto testu absolutně neuspěly. I když jsou výsledky s použitím vícebodových Gaussových-Kronrodových vzorců lepší, nejsou stále dostatečně přesné. Adaptivní metody si s tímto problémem poradily znatelně lépe. Výsledky výpočtu jsou srovnatelné s metodou *dqagi*. V tabulce 3.7 jsou výsledky výpočtu integrálu 11. funkce. Integrace byla pro dané metody (mimo metodu *dqagi*) opět provedena na omezeném intervalu $\langle 0, 1000\rangle$. Výsledky jsou velmi podobné testu 6. funkce. I když jsou neadaptivní metody stále velmi nepřesné, mají v tomto případě nižší chybu výpočtu, než v předchozím testu.

Tabulka 3.6: (Fortran) Test integrace 6. funkce na nekonečném (* a omezeném) intervalu, adaptivní i neadaptivní metody, relativní přesnost 10^{-6}, formát *double*, optimalizace $O3$

Metoda	Výsledek	Chyba	Čas $[s]$
dqk15*	$1.178555945780107.10^{-44}$	1.000000000000000	$9.375.10^{-7}$
dqk21*	$3.881683817605077.10^{-22}$	1.000000000000000	$1.563.10^{-6}$
dqk31*	$9.607394679090523.10^{-10}$	0.999999999039261	$2.031.10^{-6}$
dqk41*	$2.458111024032043.10^{-5}$	0.999975418889760	$3.281.10^{-6}$
dqk51*	$2.451292601308603.10^{-3}$	0.997548707398691	$3.306.10^{-6}$
dqk61*	$2.758310590461750.10^{-2}$	0.972416894095382	$3.750.10^{-6}$
dqag15*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
dqag21*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
dqag31*	1.000000000000000	$< 10^{-15}$	$1.563.10^{-5}$
dqag41*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
dqag51*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
dqag61*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$
dqagi	1.000000000000000	$4.440892098500626.10^{-16}$	$4.063.10^{-6}$

Tabulka 3.7: (Fortran) Test integrace 11. funkce na nekonečném (* a omezeném) intervalu, adaptivní i neadaptivní metody, relativní přesnost 10^{-6}, formát *double*, optimalizace $O3$

Metoda	Výsledek	Chyba	Čas $[s]$
dqk15*	$4.441015361815842.10^{-3}$	0.495558984638184	$6.250.10^{-7}$
dqk21*	$8.763940428267520.10^{-3}$	0.491236059571732	$9.375.10^{-7}$
dqk31*	$1.904817247088206.10^{-2}$	0.480951827529118	$1.563.10^{-6}$
dqk41*	$3.334904355527137.10^{-2}$	0.466650956444729	$1.875.10^{-6}$
dqk51*	$5.148237634172964.10^{-2}$	0.448517623658270	$2.344.10^{-6}$
dqk61*	$7.349329102591214.10^{-2}$	0.426506708974088	$2.656.10^{-6}$
dqag15*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqag21*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqag31*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqag41*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqag51*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqag61*	0.499993619888687	$6.380111312653813.10^{-6}$	$1.563.10^{-5}$
dqagi	0.499999986086234	$1.391376625026197.10^{-8}$	$8.125.10^{-6}$

3.1.7 Test integrace funkcí v komplexním oboru

Pro výpočet integrálu v komplexním oboru jsme použili upravené metody, vycházející z neadaptivních metod, založených na vícebodových Gaussových-Kronrodových vzorcích. Výsledky testů integrace 12. a 13. funkce jsou velmi podobné, viz tabulky 3.8 a 3.9. Vzhledem k tomu, že test je proveden pomocí neadaptivních metod, tak je přesnost výsledků velmi dobrá. Chyba od přesného řešení je dostatečně malá (v tabulkách jsou uvedené absolutní hodnoty chyb). S použitím vícebodových vzorců se přesnost výsledků nemění, ale roste časová náročnost výpočtu.

Zmíněné upravené metody nejsou vhodné pro výpočet integrálu na nekonečném intervalu. Proto byl pro integraci funkce číslo 14 použit omezený interval $\langle \mathrm{i}, 1000{+}\mathrm{i} \rangle^*$. Absolutní hodnota integrálu na zbývajícím intervalu $\langle 1000{+}\mathrm{i}, +\infty{+}\mathrm{i})$ je velmi malá, řádově 10^{-10858}. Výsledky jsou zaznamenány v tabulce 3.10. Je zřejmé, že metody při výpočtu dané funkce naprosto selhaly. Se zvyšujícím se počtem bodů v jednotlivých metodách, jsou výsledky sice lepší, ale ani tak nejsou dostatečně přesné.

*Pozn.: $z \in \langle \mathrm{i}, 1000{+}\mathrm{i} \rangle, z = z_r + i$, kde $z_r \in \langle 0, 1000 \rangle$

Tabulka 3.8: (Fortran) Test integrace 12. funkce v komplexním oboru, upravené neadaptivní metody, optimalizace $O3$, formát *double*

Metoda	Výsledek	Chyba	Čas $[s]$
cdqk15	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$2.031.10^{-6}$
cdqk21	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$2.656.10^{-6}$
cdqk31	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$3.750.10^{-6}$
cdqk41	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$5.000.10^{-6}$
cdqk51	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$6.094.10^{-6}$
cdqk61	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$7.188.10^{-6}$

Tabulka 3.9: (Fortran) Test integrace 13. funkce v komplexním oboru, upravené neadaptivní metody, optimalizace $O3$, formát *double*

Metoda	Výsledek	Chyba	Čas $[s]$
cdqk15	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$1.875.10^{-6}$
cdqk21	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$2.656.10^{-6}$
cdqk31	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$3.750.10^{-6}$
cdqk41	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$5.000.10^{-6}$
cdqk51	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$6.094.10^{-6}$
cdqk61	0.348992856673883 – 0.247288621781412i	$2.3171169982738604.10^{-9}$	$7.344.10^{-6}$

Tabulka 3.10: (Fortran) Test integrace 14. funkce v komplexním oboru na omezeném intervalu $\langle \mathrm{i}, 1000{+}\mathrm{i} \rangle$, upravené neadaptivní metody, optimalizace $O3$

Metoda	Výsledek	Chyba	Čas $[s]$
cdqk15	$1.168187967395657.10^{-44}$ – $1.559839420121332.10^{-45}$ i	0.991202811863483	$4.688.10^{-6}$
cdqk21	$3.847535914775095.10^{-22}$ – $5.137476465862367.10^{-23}$ i	0.991202811863483	$6.875.10^{-6}$
cdqk31	$9.522876620596702.10^{-10}$ – $1.271555499657663.10^{-10}$ i	0.991202810911195	$1.000.10^{-5}$
cdqk41	$2.436486558893165.10^{-5}$ – $3.253352959652700.10^{-6}$ i	0.991178446997894	$1.688.10^{-5}$
cdqk51	$2.429728119117216.10^{-3}$ – $3.244328657849163.10^{-4}$ i	0.988773083744365	$2.063.10^{-5}$
cdqk61	$2.734045213258485.10^{-2}$ – $3.650672339608343.10^{-3}$ i	0.963862359730898	$1.922.10^{-5}$

3.2 Matlab

V softwaru Matlab srovnáváme nejprve výsledky všech použitých adaptivních metod, viz tabulka 3.11. Následuje porovnání metod *integral* a *quadgk*, které jsou založeny na Gaussových-Kronrodových vzorcích, viz tabulka 3.12. Následuje zhodnocení výsledků získaných neadaptivní metodou *trapz*, viz tabulka 3.13, a podrobnější testování 3. funkce pomocí této metody, viz tabulka 3.14. V další části je porovnání formátů *double* a *single*, viz tabulka 3.15. Následuje vyhodnocení výpočtů na nekonečném intervalu, viz tabulka 3.16. U metod, které nejsou uzpůsobené pro výpočet na nekonečném intervalu je použit omezený interval. V poslední části jsou porovnány výsledky pro výpočet funkcí v komplexním oboru, viz tabulky 3.17 pro 12. funkci, 3.18 pro 13. funkci a 3.19 pro 14. funkci.

3.2.1 Porovnání adaptivních metod

Adaptivní metody jsou testovány i vzhledem k relativní přesnosti výpočtu. Je patrné, že má velký vliv na přesnost u všech testovaných metod. Nejlepší výsledky vykazují metody *integral* a *quadgk*. Přesnost výsledku mají tyto metody totožnou, ale metoda *quadgk* je o něco rychlejší a má podstatně nižší počet vyčíslení. Tyto metody jsme otestovali podrobněji, viz tabulka 3.12. Metoda *quadl* má také vyšší počet vyčíslení a je velmi pomalá.

Tabulka 3.11: (Matlab) Porovnání adaptivních metod, integrace 1. funkce, formát *double*

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyčíslení
quad	10^{-3}	0.719537000000000	0.01953700000000	$2.104.10^{-4}$	21
	10^{-6}	0.699987708007813	0.00001229199219	$6.400.10^{-4}$	61
	10^{-9}	0.699999995902538	0.00000000409746	$1.128.10^{-3}$	105
	10^{-12}	0.700000000001779	0.00000000000178	$1.552.10^{-3}$	142
quadl	10^{-3}	0.702009163168487	0.00200916316849	$4.435.10^{-1}$	78
	10^{-6}	0.700000002437545	0.00000000243754	$4.450.10^{-1}$	798
	10^{-9}	0.700000000309577	0.00000000030958	$4.476.10^{-1}$	318
	10^{-12}	0.700000000000276	0.00000000000028	$4.509.10^{-1}$	408
quadgk	10^{-3}	0.700305046989458	0.00030504698946	$9.689.10^{-4}$	4
	10^{-6}	0.700000180530904	0.00000018053090	$2.004.10^{-3}$	14
	10^{-9}	0.700000000020454	0.00000000002045	$3.105.10^{-3}$	24
	10^{-12}	0.699999999999938	0.00000000000006	$4.338.10^{-3}$	35
integral	10^{-3}	0.700305046989458	0.00030504698946	$2.666.10^{-3}$	240
	10^{-6}	0.700000180530904	0.00000018053090	$5.718.10^{-3}$	540
	10^{-9}	0.700000000020454	0.00000000002045	$8.639.10^{-3}$	840
	10^{-12}	0.699999999999938	0.00000000000006	$1.204.10^{-2}$	1170

3.2.2 Porovnání metod integral a quadgk

V následující tabulce je podrobnější porovnání metod *integral* a *quadgk*. Jejich přesnost výpočtu je naprosto totožná. Liší se pouze v počtu vyčíslení a časové náročnosti výpočtu. Metoda *integral* má vyšší čas výpočtu a znatelně vyšší počet vyčíslení. Tato metoda funguje na principu rozdělení intervalu integrace na 10 podintervalů a jejich následném dělení. Je to lokálně adaptivní metoda a to způsobuje vyšší počet vyčíslení. Metoda *quadgk* má tak nízký počet vyčíslení, protože volání funkce probíhá pouze jednou, ale argumentem je vektor hodnot. Tento způsob vyčíslení je možné využít i u metody *integral*, ale je nutné zadat při volání metody jako parametr příkaz $ArrayValued, True$. Metoda *quadgk* má tento způsob výpočtu napevno nastaven. Obě metody jsou si velmi podobné. Výpočet probíhá až na drobnosti na podobném principu.

Tabulka 3.12: (Matlab) Porovnání adaptivních metod *integral* a *quadgk* na konečném intervalu, integrace různých funkcí, relativní přesnost 10^{-12}, formát *double*

Funkce	Metoda	Výsledek	Chyba	Čas $[s]$	Vyčíslení
1. funkce	quadgk	0.699999999999938	0.00000000000006	$4.338.10^{-3}$	35
	integral	0.699999999999938	0.00000000000006	$1.204.10^{-2}$	1170
2. funkce	quadgk	0.400000000000000	$< 10^{-15}$	$5.412.10^{-4}$	1
	integral	0.400000000000000	$< 10^{-15}$	$1.675.10^{-3}$	150
3. funkce	quadgk	1.999999999999763	0.00000000000024	$5.210.10^{-4}$	1
	integral	1.999999999999763	0.00000000000024	$1.623.10^{-3}$	150
4. funkce	quadgk	1.154700538379252	$< 10^{-15}$	$9.494.10^{-4}$	4
	integral	1.154700538379252	$< 10^{-15}$	$4.241.10^{-3}$	1200
5. funkce	quadgk	0.379885493041722	$< 10^{-15}$	$5.271.10^{-4}$	1
	integral	0.379885493041722	$< 10^{-15}$	$8.680.10^{-4}$	150
7. funkce	quadgk	0.112139303741637	0.00000000020062	$1.231.10^{-3}$	5
	integral	0.112139303741637	0.00000000020062	$1.079.10^{-2}$	3330
8. funkce	quadgk	-1.000000000000010	$< 10^{-15}$	$1.864.10^{-3}$	13
	integral	-1.000000000000010	$< 10^{-15}$	$6.476.10^{-3}$	510
9. funkce	quadgk	0.163102243936939	0.00039269908170	$1.342.10^{-3}$	7
	integral	0.163102243936938	0.00039269908170	$3.701.10^{-3}$	870
10. funkce	quadgk	0.013492485649468	$< 10^{-15}$	$1.290.10^{-3}$	7
	integral	0.013492485649468	$< 10^{-15}$	$3.360.10^{-3}$	840

3.2.3 Test neadaptivní metody trapz

Metoda *trapz* je založena na Newtonově-Cotesově lichoběžníkovém pravidle. Přesnost výpočtu je závislá na zvolené jemnosti dělení intervalu, tedy na velikosti kroku h. Výsledky integrace všech testovaných funkcí jsou v tabulce 3.13. S vyšší jemností dělení intervalu se zvyšuje i přesnost výpočtu. Problémem u této metody je nevlastní integrál. To můžeme pozorovat u funkce číslo 3, kde je výsledná hodnota nekonečno. Je to způsobeno tím, že metoda *trapz* dosazuje okrajové body do funkce a tím vzniká problém dělení nulou. Řešením tohoto problému je posunutí dolní meze integrace o malé ε, tato možnost byla otestována v tabulce 3.14.

Tabulka 3.13: (Matlab) Neadaptivní metoda *trapz* s použitím různé velikosti kroku h, integrace různých funkcí na konečném intervalu, formát *double*

Funkce	Krok	Výsledek	Chyba	Čas $[s]$
1. funkce	0.01	0.695000000000000	0.00500000000000	$6.364.10^{-5}$
	0.001	0.699500000000000	0.00050000000000	$7.759.10^{-5}$
	0.0001	0.699950000000000	0.00005000000000	$1.575.10^{-4}$
2. funkce	0.01	0.400012245153189	0.00001224515319	$6.523.10^{-5}$
	0.001	0.400000124194088	0.00000012419409	$7.083.10^{-5}$
	0.0001	0.400000001247452	0.00000000124745	$1.501.10^{-4}$
3. funkce	0.01	inf	inf	$6.746.10^{-5}$
	0.001	inf	inf	$7.036.10^{-5}$
	0.0001	inf	inf	$1.493.10^{-4}$
4. funkce	0.01	1.154700538387657	0.00000000000841	$6.572.10^{-5}$
	0.001	1.154700538379252	$< 10^{-15}$	$7.115.10^{-5}$
	0.0001	1.154700538379252	$< 10^{-15}$	$1.503.10^{-4}$
5. funkce	0.01	0.379885937943524	0.00000044490180	$6.841.10^{-5}$
	0.001	0.379885497490728	0.00000000444901	$7.616.10^{-5}$
	0.0001	0.379885493086213	0.00000000004449	$1.419.10^{-4}$
7. funkce	0.01	0.147665638290240	0.03552633474922	$6.698.10^{-5}$
	0.001	0.112477199032314	0.00033789549129	$7.582.10^{-5}$
	0.0001	0.112142681130212	0.00000337758919	$1.495.10^{-4}$
8. funkce	0.01	-0.967776430432457	0.03222356956751	$6.698.10^{-5}$
	0.001	-0.995627100493974	0.00437289950599	$7.048.10^{-5}$
	0.0001	-0.999447588294747	0.00055241170522	$1.707.10^{-4}$
9. funkce	0.01	0.175988215950032	0.01249327293139	$6.933.10^{-5}$
	0.001	0.164103525247205	0.00060858222886	$7.262.10^{-5}$
	0.0001	0.163494949298871	0.00000000628023	$1.502.10^{-4}$

Funkce	Krok	Výsledek	Chyba	Čas $[s]$
10. funkce	0.01	0.015309127743214	0.00181664209375	$6.694.10^{-5}$
	0.001	0.013492484228055	0.00000000142141	$7.158.10^{-5}$
	0.0001	0.013492485635254	0.00000000001421	$1.504.10^{-4}$

3.2.4 Test metody trapz pro funkci číslo 3

U výpočtu integrálu pomocí neadaptivní metody *trapz* nastal problém s integrací funkce číslo 3, kdy je výsledná získaná hodnota nekonečno. Je to způsobeno tím, že se jedná o nevlastní integrál a u metody *trapz* dochází k přímému dosazování okrajových bodů intervalu do dané funkce. Tím vzniká problém dělení nulou. Jako vhodné řešení se ukázalo posunutí dolní meze integrace o malé ε. Nový interval integrace je tedy $\langle 0 + \varepsilon, 1\rangle$. Různé hodnoty ε jsou porovnány s různými kroky dělení intervalu. Výsledky jsou zaznamenané v tabulce 3.14. Největší přesnost výsledku integrace jsme získali pro kombinaci velikosti kroku o délce 0.001 a hodnoty ε o velikosti 0.0001.

Tabulka 3.14: (Matlab) Neadaptivní metoda *trapz* pro různé kroky h, integrace 3. funkce s posunutím dolní meze intervalu o různé hodnoty ε, formát *double*

Epsilon	Krok h	Výsledek	Chyba	Čas $[s]$
0	0.01	inf	inf	$6.746.10^{-5}$
	0.001	inf	inf	$7.036.10^{-5}$
	0.0001	inf	inf	$1.493.10^{-4}$
0.01	0.01	1.803960382478415	0.19603961752158	$6.594.10^{-5}$
	0.001	1.800041599055133	0.19995840094487	$7.104.10^{-5}$
	0.0001	1.800000416247396	0.19999958375260	$1.490.10^{-4}$
0.001	0.01	1.990462490073892	0.00953750992611	$6.185.10^{-5}$
	0.001	1.938008105641740	0.06199189435885	$6.955.10^{-5}$
	0.0001	1.936767614332395	0.06323238566730	$1.471.10^{-4}$
0.0001	0.01	2.342734503264946	0.34273450326495	$6.154.10^{-5}$
	0.001	1.998937548311578	0.00106245168842	$6.780.10^{-5}$
	0.0001	1.980396454495238	0.01960354550476	$1.758.10^{-4}$

3.2.5 Porovnání formátů double a single

V následující tabulce 3.15 jsou porovnány výsledky integrace vzhledem k použitému formátu - *double* nebo *single*. Přesnost výpočtu ve formátu *double* je na 16 platných cifer a u formátu *single* na 7 platných cifer. Porovnání počtů vyčíslení a časové náročnosti výpočtu dopadlo pro oba formáty velmi podobně. Největší rozdíl při porovnání výsledků integrace v různých formátech je v přesnosti výpočtu. Překvapivé mohou být hodnoty, kde je ve formátu *single* chyba $< 10^{-15}$, tento výsledek ale musíme brát s rezervou. V tomto případě se jedná pouze o výpis na 16 platných cifer, ale přesnost výsledku je pouze na 7. Další cifry nenesou ve formátu *single* potřebnou informaci. S použitím formátu *double* jsme tedy získali vyšší přesnost výsledků.

Tabulka 3.15: (Matlab) Porovnání formátů *double* a *single* pro různé relativní přesnosti, integrace 6. funkce na intervalu $\langle 0, 100 \rangle$

Metoda	Formát	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyčíslení
quad	double	10^{-3}	1.000174014033678	0.00017401403368	$4.674.10^{-4}$	45
	single		1.000174045562744	0.00017404556274	$4.927.10^{-4}$	45
quadl	double		1.000000006830328	0.00000000683033	$7.521.10^{-4}$	108
	single		0.999999940395355	0.00000005960464	$1.376.10^{-3}$	108
quadgk	double		1.000000000207906	0.00000000020791	$6.651.10^{-4}$	2
	single		1.000000000000000	$< 10^{-15}$	$7.520.10^{-4}$	2
integral	double		1.000000000207906	0.00000000020791	$9.693.10^{-4}$	2
	single		1.000000000000000	$< 10^{-15}$	$1.471.10^{-4}$	2
quad	double	10^{-6}	1.000000244096885	0.00000024409689	$9.251.10^{-4}$	85
	single		1.000000357627869	0.00000035762787	$9.442.10^{-4}$	85
quadl	double		1.000000000178012	0.00000000017801	$1.399.10^{-3}$	198
	single		1.000000000000000	$< 10^{-15}$	$2.608.10^{-3}$	198
quadgk	double		1.000000000000034	$< 10^{-15}$	$7.773.10^{-4}$	3
	single		0.999999940395355	0.00000005960464	$4.457.10^{-3}$	3
integral	double		1.000000000000034	0.00000000000003	$1.109.10^{-3}$	3
	single		0.999999940395355	0.00000005960464	$1.797.10^{-3}$	3

3.2.6 Test integrace funkcí na nekonečném intervalu

Pro test integrace funkcí číslo 6 a 11 na nekonečném intervalu byly použity všechny adaptivní metody. Metody *integral* a *quadgk* umožňují integraci i na nekonečném intervalu, ale metody *quad* a *quadl* nejsou pro integraci na nekonečném intervalu uzpůsobeny. Aby bylo možné integraci na nekonečném intervalu pomocí daných metod uskutečnit, bylo nutné omezit interval integrace. Proto jsou funkce testovány pro dané metody na konečném

intervalu $\langle 0, 1000\rangle$. U 6. funkce je rozdíl téměř neznatelný, protože absolutní hodnota integrálu na zbývajícím intervalu $\langle 1000, +\infty)$ je řádově 10^{-10858}. Pro 11. funkci je absolutní hodnota na daném intervalu řádově 10^{-6}.

Výsledky integrace na nekonečném a omezeném intervalu jsou v tabulce 3.16. Integrace přes omezený interval se ukázala jako vhodné řešení pro výpočet integrálu pomocí metod *quad* a *quadl*. Metody mají sice vyšší počet vyčíslení, ale nejen že jsme získali výsledek integrace, ale dosáhli jsme i uspokojivé přesnosti. Nicméně pro výpočet na nekonečném intervalu jsou jednoznačně lepší metody *integral* a *quadgk*.

Tabulka 3.16: (Matlab) Výpočet integrálu na nekonečném a omezeném intervalu $\langle 0, 1000\rangle$, adaptivní metody, integrace 6. a 11. funkce, relativní přesnost 10^{-12}

Fce	Metoda	Interval	Výsledek	Chyba	Čas $[s]$	Vyčíslení
6. fce	quad	$\langle 0, +\infty)$	NaN	NaN	$1.291.10^{-3}$	13
		$\langle 0, 1000\rangle$	1.000000000003364	0.00000000000336	$9.885.10^{-3}$	897
	quadl	$\langle 0, +\infty)$	inf	inf	$1.109.10^{-3}$	13
		$\langle 0, 1000\rangle$	1.000000000000001	$< 10^{-15}$	$5.276.10^{-3}$	738
	quadgk	$\langle 0, +\infty)$	1.000000000000000	$< 10^{-15}$	$5.521.10^{-4}$	1
	integral	$\langle 0, +\infty)$	1.000000000000000	$< 10^{-15}$	$5.521.10^{-4}$	1
11. fce	quad	$\langle 0, +\infty)$	NaN	NaN	$1.316.10^{-3}$	13
		$\langle 0, 1000\rangle$	0.499993633802405	0.00000636619760	$1.769.10^{-2}$	1557
	quadl	$\langle 0, +\infty)$	inf	inf	$1.388.10^{-3}$	13
		$\langle 0, 1000\rangle$	0.499993633802277	0.00000636619772	$1.195.10^{-2}$	1608
	quadgk	$\langle 0, +\infty)$	0.500000000000000	$< 10^{-15}$	$6.879.10^{-4}$	2
	integral	$\langle 0, +\infty)$	0.500000000000000	$< 10^{-15}$	$7.573.10^{-4}$	2

3.2.7 Test integrace funkcí v komplexním oboru

V tabulkách 3.17 a 3.18 jsou znázorněny výsledky integrace funkcí číslo 12 a 13. Všechny metody vykazují přesné výsledky. Výsledky testu integrace 14. funkce jsou v tabulce 3.19. Tato funkce je integrována v komplexním oboru na nekonečném intervalu. S touto variantou si použité metody nedokázaly poradit a bylo nutné integraci provést na intervalu $\langle \mathrm{i}, 1000{+}\mathrm{i}\rangle$*. Absolutní hodnota integrálu na zbývajícím intervalu $\langle 1000{+}\mathrm{i}, +\infty{+}\mathrm{i})$ je řádově 10^{-10858}. Pro výpočet integrálu na omezeném intervalu mají metody dobré výsledky (v tabulkách jsou uvedené absolutní hodnoty chyb).

*Pozn.: $z \in \langle \mathrm{i}, 1000{+}\mathrm{i}\rangle, z = z_r + i$, kde $z_r \in \langle 0, 1000\rangle$

Tabulka 3.17: (Matlab) Test integrace 12. funkce v komplexním oboru, adaptivní metody

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
quad	10^{-9}	0.130584240446267 + 0.500000000000000i	0.00000000000254	$4.438.10^{-4}$	33
quadl		0.130584240451501 + 0.500000000000000i	0.00000000000778	$1.297.10^{-4}$	18
quadgk		0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$5.737.10^{-4}$	1
integral		0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$6.534.10^{-4}$	1
quad	10^{-12}	0.130584240443724 + 0.500000000000000i	$< 10^{-15}$	$1.773.10^{-2}$	121
quadl		0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$4.258.10^{-4}$	48
quadgk		0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$5.664.10^{-4}$	1
integral		0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$6.514.10^{-4}$	1

Tabulka 3.18: (Matlab) Test integrace 13. funkce v komplexním oboru, adaptivní metody

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
quad	10^{-9}	0.348992856677378 – 0.247288621780367i	0.00000000000365	$4.549.10^{-4}$	33
quadl		0.348992856701793 – 0.247288621780367i	0.00000000004266	$1.269.10^{-4}$	18
quadgk		0.348992856673883 – 0.247288621780367i	$< 10^{-15}$	$5.849.10^{-4}$	1
integral		0.348992856673883 – 0.247288621780367i	$< 10^{-15}$	$6.518.10^{-4}$	1
quad	10^{-12}	0.130584240443724 – 0.247288621780367i	$< 10^{-15}$	$1.794.10^{-2}$	129
quadl		0.348992856673883 – 0.247288621780367i	$< 10^{-15}$	$4.162.10^{-4}$	48
quadgk		0.348992856673883 – 0.247288621780367i	$< 10^{-15}$	$5.712.10^{-4}$	1
integral		0.348992856673883 – 0.247288621780367i	$< 10^{-15}$	$6.903.10^{-4}$	1

Tabulka 3.19: (Matlab) Test integrace 14. funkce v komplexním oboru na omezeném intervalu $\langle \mathrm{i}, 1000{+}\mathrm{i} \rangle$, adaptivní metody, formát *double*

Metoda	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
quad	10^{-9}	0.991202811982718 + 0.132351750113695i	0.00000000012030	$3.576.10^{-3}$	257
quadl		0.991202811863490 + 0.132351750113695i	0.00000000000002	$2.941.10^{-3}$	318
quadgk		0.991202811863474 + 0.132351750113695i	$< 10^{-15}$	$1.793.10^{-3}$	10
integral		0.991202811863474 + 0.132351750113695i	$< 10^{-15}$	$1.895.10^{-3}$	10
quad	10^{-12}	0.991202811866808 + 0.132351750113695i	0.00000000000336	$1.306.10^{-2}$	897
quadl		0.991202811863475 + 0.132351750113695i	$< 10^{-15}$	$6.954.10^{-3}$	738
quadgk		0.991202811863474 + 0.132351750113695i	$< 10^{-15}$	$1.950.10^{-3}$	11
integral		0.991202811863474 + 0.132351750113695i	$< 10^{-15}$	$2.031.10^{-3}$	11

3.3 Mathematica

V softwaru Mathematica nejprve porovnáváme vliv globální a lokální adaptivity při použití metod založených na Gaussových-Kronrodových vzorcích, viz tabulka 3.20. Následuje srovnání všech použitých metod pro různé funkce, viz tabulka 3.21. Dále je testován vliv relativní přesnosti při výpočtu integrálu, viz tabulka 3.22. Další výsledky jsou pro funkce integrované na nekonečném intervalu vzhledem ke globální i lokální adaptivitě a různým relativním přesnostem, viz tabulka 3.23 pro integraci 6. funkce a tabulka 3.24 pro integraci 11. funkce. V poslední části jsou porovnány výsledky pro výpočet funkcí v komplexním oboru, viz tabulky 3.25 pro 12. funkci, 3.26 pro 13. funkci a 3.27 pro 14. funkci.

3.3.1 Vliv lokální a globální adaptivity

V následující tabulce 3.20 jsou porovnány výsledky výpočtu integrálu s použitím různých druhů adaptivity - globální (Glob) a lokální (Lok). Obě formy adaptivity se na námi testovaných funkcích projevují velmi podobně. Přesnost výsledku je také ovlivněna nastavenou relativní přesností. S vyšší relativní přesností získáváme pro oba druhy adaptivity přesnější výsledek.

Tabulka 3.20: (Mathematica) Vliv lokální a globální adaptivity, metoda založená na Gaussových-Kronrodových vzorcích, integrace různých funkcí s různými relativními přesnostmi

Fce	Adapt.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
4. fce	Glob	10^{-3}	1.154700124418672	$3.585003777080622.10^{-7}$	$1.047.10^{-3}$	156
	Lok		1.154700124418673	$3.585003775157660.10^{-7}$	$1.156.10^{-3}$	156
	Glob	10^{-6}	1.154700538380521	$1.099550064074122.10^{-12}$	$1.500.10^{-3}$	451
	Lok		1.154700538379081	$1.472989417769810.10^{-13}$	$1.688.10^{-3}$	473
	Glob	10^{-9}	1.154700538379254	$2.307555223660277.10^{-15}$	$2.156.10^{-3}$	869
	Lok		1.154700538379253	$1.153777611830138.10^{-15}$	$2.375.10^{-3}$	869
	Glob	10^{-12}	1.154700538379254	$2.307555223660277.10^{-15}$	$3.594.10^{-3}$	1749
	Lok		1.154700538379253	$1.153777611830138.10^{-15}$	$3.438.10^{-3}$	1485
8. fce	Glob	10^{-3}	−0.9999007867262959	$9.921327366846721.10^{-5}$	$9.531.10^{-4}$	121
	Lok		−0.9999503933631486	$4.960663681574839.10^{-5}$	$1.109.10^{-3}$	143
	Glob	10^{-6}	−0.9999999031120383	$9.688792601547993.10^{-8}$	$1.281.10^{-3}$	341
	Lok		−0.9999999515560198	$4.844394452252661.10^{-8}$	$1.484.10^{-3}$	363
	Glob	10^{-9}	−0.9999999999526922	$4.727218616551623.10^{-11}$	$1.766.10^{-3}$	649
	Lok		−0.9999999999526925	$4.727185309860885.10^{-11}$	$1.922.10^{-3}$	627
	Glob	10^{-12}	−0.9999999999999775	$1.310063169057732.10^{-14}$	$2.672.10^{-3}$	1243
	Lok		−0.9999999999999549	$9.436895709314168.10^{-15}$	$2.828.10^{-3}$	1155

Fce	Adapt.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
10. fce	Glob	10^{-3}	0.01349248550868603	$1.043407986009685.10^{-8}$	$1.094.10^{-3}$	209
	Lok		0.01349248550868603	$1.043407986009685.10^{-8}$	$1.122.10^{-3}$	209
	Glob	10^{-6}	0.01349248564949134	$1.751760792950442.10^{-12}$	$1.234.10^{-3}$	297
	Lok		0.01349248564949133	$1.751632223350959.10^{-12}$	$1.359.10^{-3}$	297
	Glob	10^{-9}	0.01349248564946779	$6.942758372060466.10^{-15}$	$1.687.10^{-3}$	583
	Lok		0.01349248564946779	$6.685619173095264.10^{-15}$	$1.859.10^{-3}$	583
	Glob	10^{-12}	0.01349248564946779	$6.942758372060466.10^{-15}$	$2.516.10^{-3}$	1111
	Lok		0.01349248564946779	$6.428479974130061.10^{-15}$	$2.656.10^{-3}$	1045

3.3.2 Porovnání použitých metod

V následující tabulce 3.21 jsou porovnány výsledky integrace 1. funkce získané pomocí různých metod. Metoda GK je založena na Gaussových-Kronrodových vzorcích, metoda NC na Newtonových-Cotesových vzorcích a metoda LK na Lobattových-Kronrodových vzorcích. Použité metody jsou testovány i vzhledem ke druhu adaptivity a relativní přesnosti. Přesnost výpočtu i časová náročnost jsou pro testované metody podobné. Nejvyšší přesnost výpočtu má ale metoda LK s použitím lokální adaptivity. Nejnižší počet vyčíslení má ve všech případech metoda NC.

Tabulka 3.21: (Mathematica) Porovnání použitých metod, globální i lokální adaptivita, různá nastavení relativní přesnosti, integrace 1. funkce

Met.	Adapt.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
GK	Globální	10^{-3}	0.6997714598005036	$3.264859992805188.10^{-4}$	$1.094.10^{-3}$	165
NC			0.6995225694444445	$6.820436507935591.10^{-4}$	$1.156.10^{-3}$	85
LK			0.7001833695636295	$2.619565194707439.10^{-4}$	$1.094.10^{-3}$	153
GK	Lokální	10^{-3}	0.7004326863286723	$6.181233266747529.10^{-4}$	$1.078.10^{-3}$	143
NC			0.7001193576388889	$1.705109126984691.10^{-4}$	$1.250.10^{-3}$	65
LK			0.6999977858002219	$3.163142540119068.10^{-6}$	$1.297.10^{-3}$	233
GK	Globální	10^{-6}	0.7000002231837892	$3.188339846106812.10^{-7}$	$1.344.10^{-3}$	385
NC			0.7000004662407769	$6.660582528374346.10^{-7}$	$1.344.10^{-3}$	185
LK			0.6999998209281603	$2.558169138427197.10^{-7}$	$1.297.10^{-3}$	333
GK	Lokální	10^{-6}	0.6999995774547579	$6.036360600258917.10^{-7}$	$1.138.10^{-3}$	363
NC			0.6999998834398058	$1.665145630904062.10^{-7}$	$1.422.10^{-3}$	125
LK			0.7000000153437383	$2.191962627280069.10^{-8}$	$1.594.10^{-3}$	401

Met.	Adapt.	Rel. p.	Výsledek	Chyba	Čas [s]	Vyč.
GK	Globální	10^{-9}	0.6999999997820477	$3.113603026199436.10^{-10}$	$1.656.10^{-3}$	605
NC			0.6999999995446867	$6.504474736601651.10^{-10}$	$1.547.10^{-3}$	285
LK			0.6999999998952485	$1.496448996464648.10^{-10}$	$1.547.10^{-3}$	495
GK	Lokální	10^{-9}	0.7000000004126425	$5.894894096668071.10^{-10}$	$1.781.10^{-3}$	583
NC			0.7000000001138283	$1.626119873675082.10^{-10}$	$1.594.10^{-3}$	185
LK			0.7000000000261767	$3.739532493187068.10^{-11}$	$1.938.10^{-3}$	569
GK	Globální	10^{-12}	0.7000000000002136	$3.051527284826859.10^{-13}$	$1.969.10^{-3}$	825
NC			0.7000000000004446	$6.352061733748217.10^{-13}$	$1.734.10^{-3}$	385
LK			0.7000000000001021	$1.459150260935920.10^{-13}$	$1.813.10^{-3}$	675
GK	Lokální	10^{-12}	0.6999999999995978	$5.744611135989025.10^{-13}$	$2.125.10^{-3}$	803
NC			0.6999999999998888	$1.587618925213974.10^{-13}$	$1.766.10^{-3}$	245
LK			0.7000000000000556	$7.946024790531478.10^{-14}$	$1.250.10^{-3}$	737

3.3.3 Vliv nastavení relativní přesnosti

Vliv relativní přesnosti se projevil již v předchozích testech. V následující tabulce 3.22 je podrobněji znázorněno jak nastavení relativní přesnosti ovlivní výpočet. Relativní přesnost má vliv na všechny zkoumané hodnoty získané integrací 2. a 3. funkce. Přesnost výsledku se s vyšší relativní přesností zlepšuje a tím klesá hodnota chyby výpočtu. Časová náročnost a počet vyčíslení se s rostoucí relativní přesností zvyšují. Vzhledem k porovnání všech testovaných kritérií jsou nejlepší výsledky získané pomocí metody GK, která je založena na Gaussových-Kronrodových vzorcích.

Tabulka 3.22: (Mathematica) Vliv nastavení relativní přesnosti, globální adaptivita, metody GK, NC, LK, integrace 2. a 3. funkce

Fce	Met.	Rel. p.	Výsledek	Chyba	Čas [s]	Vyč.
2. fce	GK	10^{-3}	0.3999999127434368	$2.181414079927002.10^{-7}$	$7.500.10^{-4}$	11
		10^{-6}	0.3999999995179715	$1.205071320509177.10^{-9}$	$8.750.10^{-4}$	77
		10^{-9}	0.3999999999995297	$1.175864960956119.10^{-12}$	$1.000.10^{-3}$	165
		10^{-12}	0.4000000000000000	$1.387778780781446.10^{-16}$	$1.250.10^{-3}$	319
2. fce	NC	10^{-3}	0.4000536050074883	$1.340125187206109.10^{-4}$	$9.844.10^{-4}$	15
		10^{-6}	0.4000000101404132	$2.535103293954677.10^{-8}$	$1.203.10^{-3}$	115
		10^{-9}	0.4000000000017749	$4.437283873670594.10^{-12}$	$2.484.10^{-3}$	695
		10^{-12}	0.4000000000000005	$1.110223024625157.10^{-15}$	$9.797.10^{-3}$	3885

Fce	Met.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
2. fce	LK	10^{-3}	0.400004532484393	$1.133121098256806.10^{-5}$	$7.500.10^{-4}$	9
		10^{-6}	0.400000004426256	$1.106564076480687.10^{-8}$	$8.750.10^{-4}$	81
		10^{-9}	0.400000000004322	$1.080691092170127.10^{-11}$	$1.031.10^{-3}$	171
		10^{-12}	0.400000000000004	$1.026956297778270.10^{-14}$	$1.453.10^{-3}$	405
3. fce	GK	10^{-3}	1.999023468256129	$4.882658719356225.10^{-4}$	$1.156.10^{-3}$	275
		10^{-6}	1.999999046355722	$4.768221392170702.10^{-7}$	$1.828.10^{-3}$	715
		10^{-9}	1.999999999670740	$1.646298652957512.10^{-10}$	$2.797.10^{-3}$	1353
		10^{-12}	1.999999999999889	$5.562217353372034.10^{-14}$	$4.813.10^{-3}$	2651
3. fce	NC	10^{-3}	1.998327300975448	$8.363495122759579.10^{-4}$	$2.141.10^{-3}$	157
		10^{-6}	1.999999861751883	$6.912405825509893.10^{-8}$	$5.047.10^{-3}$	1080
		10^{-9}	1.999999999976234	$1.188293907716798.10^{-11}$	$1.735.10^{-2}$	6305
		10^{-12}	1.999999999999996	$1.887379141862766.10^{-15}$	$9.714.10^{-2}$	35870
3. fce	LK	10^{-3}	1.997863374090280	$1.068312954860207.10^{-3}$	$1.890.10^{-3}$	246
		10^{-6}	1.999998524589982	$7.377050088575032.10^{-7}$	$3.156.10^{-3}$	603
		10^{-9}	1.999999999745298	$1.273510186194926.10^{-10}$	$5.422.10^{-3}$	1460
		10^{-12}	1.999999999999912	$4.418687638008123.10^{-14}$	$9.656.10^{-3}$	3453

3.3.4 Test integrace funkcí na nekonečném intervalu

Všechny použité metody jsou vhodné i k výpočtu integrálu funkce na nekonečném intervalu. Následující porovnání je zaměřené na integraci na nekonečném intervalu. Všechny metody jsou testovány s použitím globální i lokální adaptivity a s různými nastaveními relativní přesnosti. Nejprve zhodnotíme výsledky získané při integraci funkce číslo 6, viz tabulka 3.23. Vzhledem ke všem testovaným kritériím, tedy přesnosti výsledku, časové náročnosti výpočtu a počtu vyčíslení, jsme získali nejlepší výsledky pomocí metody LK. Metoda NC má ve většině případů řádově nižší přesnost výpočtu a také má výrazně vyšší počet vyčíslení, především s použitím globální adaptivity.

V tabulce 3.24 jsou porovnány výsledky integrace 11. funkce. V tomto případě byly nejpřesnější výsledky ve většině případů získané pomocí metody GK, která má i nejnižší počet vyčíslení. Metoda NC má až na poslední výsledek řádově horší přesnost výpočtu. V posledním případě se s použitím globální adaptivity prokazuje nejvyšší přesností, ovšem s příliš velkým počtem vyčíslení. U obou testovaných funkcí se opět projevuje vliv relativní přesnosti.

Tabulka 3.23: (Mathematica) Výpočet integrálu funkce na nekonečném intervalu, metody GK, NC, LK, globální i lokální adaptivita, různé relativní přesnosti, integrace 6. funkce

Met.	Adapt.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
GK	Globální	10^{-3}	1.000000000005117	$5.116795875892421.10^{-12}$	$8.281.10^{-4}$	33
NC			1.000023333498363	$2.333349836280973.10^{-5}$	$1.500.10^{-3}$	63
LK			0.999999999999049	$9.510170428939091.10^{-13}$	$1.219.10^{-3}$	43
GK	Lokální	10^{-3}	1.000000000005117	$5.117017920497346.10^{-12}$	$9.688.10^{-4}$	33
NC			1.000023320693448	$2.332069344790355.10^{-5}$	$1.672.10^{-3}$	40
LK			1.000000000000000	$4.440892098500626.10^{-16}$	$1.469.10^{-3}$	64
GK	Globální	10^{-6}	1.000000000000010	$9.769962616701378.10^{-15}$	$8.750.10^{-4}$	55
NC			1.000000008075326	$8.075325963474711.10^{-9}$	$2.188.10^{-3}$	293
LK			1.000000000000594	$5.939693181744587.10^{-13}$	$1.313.10^{-3}$	79
GK	Lokální	10^{-6}	1.000000000005117	$5.117017920497346.10^{-12}$	$9.688.10^{-4}$	33
NC			1.000000000000000	$2.930263143241518.10^{-9}$	$1.922.10^{-3}$	112
LK			0.700000015343738	$4.440892098500626.10^{-16}$	$1.469.10^{-3}$	64
GK	Globální	10^{-9}	1.000000000000007	$7.327471962526033.10^{-15}$	$9.688.10^{-4}$	99
NC			1.000000000002214	$2.214006755707487.10^{-12}$	$6.547.10^{-3}$	1682
LK			1.000000000000594	$5.935252289646087.10^{-13}$	$1.484.10^{-3}$	151
GK	Lokální	10^{-9}	1.000000000005117	$5.117017920497346.10^{-12}$	$9.844.10^{-4}$	33
NC			1.000000000000000	$2.261590914542921.10^{-11}$	$2.891.10^{-3}$	412
LK			0.700000000026176	$2.220446049250313.10^{-16}$	$1.859.10^{-3}$	232
GK	Globální	10^{-12}	1.000000000000001	$1.332267629550188.10^{-15}$	$2.972.10^{-2}$	209
NC			1.000000000000001	$8.881784197001252.10^{-16}$	$1.734.10^{-3}$	9152
LK			1.000000000000000	$4.440892098500626.10^{-16}$	$2.344.10^{-3}$	420
GK	Lokální	10^{-12}	1.000000000000010	$9.769962616701378.10^{-15}$	$1.016.10^{-3}$	55
NC			1.000000000000002	$1.776356839400250.10^{-15}$	$1.766.10^{-3}$	1648
LK			1.000000000000000	$1.110223024625157.10^{-16}$	$2.109.10^{-3}$	344

Tabulka 3.24: (Mathematica) Výpočet integrálu funkce na nekonečném intervalu, metody GK, NC, LK, globální i lokální adaptivita, různé relativní přesnosti, integrace 11. funkce

Met.	Adapt.	Rel. p.	Výsledek	Chyba	Čas $[s]$	Vyč.
GK	Globální	10^{-3}	0.5000000213823564	$4.276471288200412.10^{-8}$	$1.000.10^{-3}$	99
NC			0.4998865010485600	$2.269979028799440.10^{-4}$	$1.906.10^{-3}$	102
LK			0.4999512310601093	$9.753787978139528.10^{-5}$	$1.391.10^{-3}$	97
GK	Lokální	10^{-3}	0.4999925425908296	$1.491481834081654.10^{-5}$	$1.094.10^{-3}$	77
NC			0.4998937722017088	$2.124555965824504.10^{-4}$	$1.750.10^{-3}$	40
LK			0.4999946550149126	$1.068997017483753.10^{-5}$	$1.688.10^{-3}$	120
GK	Globální	10^{-6}	0.5000000000000009	$1.776356839400250.10^{-15}$	$1.094.10^{-3}$	143
NC			0.4999999699017412	$6.019651754840538.10^{-8}$	$4.188.10^{-3}$	610
LK			0.4999999047456834	$1.905086332021355.10^{-7}$	$2.703.10^{-3}$	322
GK	Lokální	10^{-6}	0.5000000000127399	$2.547984045975227.10^{-11}$	$1.188.10^{-3}$	121
NC			0.4999998814975697	$2.370048605149222.10^{-7}$	$2.422.10^{-3}$	244
LK			0.4999999839509330	$3.209813403959316.10^{-8}$	$2.047.10^{-3}$	288
GK	Globální	10^{-9}	0.5000000000000009	$1.776356839400250.10^{-15}$	$1.406.10^{-3}$	275
NC			0.4999999999963128	$7.374434396467677.10^{-12}$	$1.279.10^{-2}$	3177
LK			0.4999999999767447	$4.651068419292415.10^{-11}$	$4.313.10^{-3}$	688
GK	Lokální	10^{-9}	0.5000000000000009	$1.776356839400250.10^{-15}$	$1.359.10^{-3}$	209
NC			0.4999999998826817	$2.346366434480274.10^{-10}$	$4.500.10^{-3}$	868
LK			0.4999999999518068	$9.638645437348714.10^{-11}$	$2.844.10^{-3}$	624
GK	Globální	10^{-12}	0.5000000000000007	$1.332267629550188.10^{-15}$	$1.969.10^{-3}$	539
NC			0.4999999999999997	$5.551115123125783.10^{-16}$	$5.720.10^{-2}$	16495
LK			0.4999999999999777	$4.451994328746878.10^{-14}$	$6.547.10^{-3}$	1362
GK	Lokální	10^{-12}	0.5000000000000007	$1.332267629550188.10^{-15}$	$1.734.10^{-3}$	407
NC			0.4999999999998831	$2.338129689860580.10^{-13}$	$1.219.10^{-2}$	3184
LK			0.4999999999999920	$1.598721155460225.10^{-14}$	$4.500.10^{-3}$	1352

3.3.5 Test integrace funkcí v komplexním oboru

Všechny použité metody jsou vhodné i pro výpočet integrálu funkce v komplexním oboru. Výsledky všech metod jsou velmi přesné, metoda NC má ale vyšší počet vyčíslení a časovou náročnost výpočtu, viz tabulky 3.25 pro výsledky integrace 12. funkce, 3.26 pro výsledky integrace 13. funkce a 3.27 pro výsledky integrace 14. funkce.

Tabulka 3.25: (Mathematica) Výpočet integrálu 12. funkce v komplexním oboru, metody GK, NC, LK, globální i lokální adaptivita, relativní přesnost 10^{-12}

Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
GK, glob	0.1305842404437229 + 0.5000000000000006i	$1.121489901770251.10^{-15}$	$8.594.10^{-4}$	33
NC, glob	0.1305842404437226 + 0.5000000000000000i	$1.611288759788450.10^{-16}$	$5.000.10^{-3}$	1065
LK, glob	0.1305842404437227 + 0.4999999999999998i	$3.267028964510625.10^{-16}$	$9.688.10^{-4}$	63
GK, lok	0.1305842404437229 + 0.5000000000000006i	$1.121489901770251.10^{-15}$	$9.688.10^{-4}$	33
NC, lok	0.1305842404437227 + 0.5000000000000000i	$5.370962532628168.10^{-17}$	$2.281.10^{-3}$	257
LK, lok	0.1305842404437227 + 0.4999999999999998i	$3.267028964510625.10^{-16}$	$1.094.10^{-3}$	65

Tabulka 3.26: (Mathematica) Výpočet integrálu 13. funkce v komplexním oboru, metody GK, NC, LK, globální i lokální adaptivita, relativní přesnost 10^{-12}

Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
GK, glob	0.3489928566738830 – 0.2472886217814123i	$1.116432948315769.10^{-15}$	$8.906.10^{-4}$	33
NC, glob	0.3489928566738824 – 0.2472886217814119i	$6.121837435030051.10^{-16}$	$9.844.10^{-4}$	63
LK, glob	0.3489928566738826 – 0.2472886217814119i	$1.835404460974467.10^{-16}$	$9.688.10^{-4}$	63
GK, lok	0.3489928566738830 – 0.2472886217814123i	$1.116432948315769.10^{-15}$	$1.047.10^{-3}$	55
NC, lok	0.3489928566738826 – 0.2472886217814120i	$6.489134702875428.10^{-17}$	$5.656.10^{-3}$	1001
LK, lok	0.3489928566738826 – 0.2472886217814119i	$1.835404460974467.10^{-16}$	$1.078.10^{-3}$	65

Tabulka 3.27: (Mathematica) Výpočet integrálu 14. funkce v komplexním oboru, metody GK, NC, LK, globální i lokální adaptivita, relativní přesnost 10^{-12}

Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
GK, glob	0.9912028118634753 + 0.1323517500977733i	$1.680072297996111.10^{-15}$	$1.563.10^{-3}$	209
NC, glob	0.9912028118634745 + 0.1323517500977732i	$9.091804728055929.10^{-16}$	$4.475.10^{-2}$	9152
LK, glob	0.9912028118634734 + 0.1323517500977731i	$2.237726045655905.10^{-16}$	$3.031.10^{-3}$	420
GK, lok	0.9912028118634753 + 0.1323517500977733i	$1.680072297996111.10^{-15}$	$1.641.10^{-3}$	231
NC, lok	0.9912028118634753 + 0.1323517500977732i	$1.676629758422416.10^{-15}$	$9.438.10^{-3}$	1648
LK, lok	0.9912028118634736 + 0.1323517500977730i	$2.775557561562891.10^{-17}$	$2.563.10^{-3}$	344

3.4 Celkové porovnání

V této části jsou porovnané výsledky integrace funkcí vzhledem ke všem použitým matematickým softwarům. Porovnávaná kritéria jsou opět přesnost výsledku, průměrný čas výpočtu a počet vyčíslení funkce. Všechny použité metody jsou vždy v tabulce seřazeny podle druhu vzorce, ze kterého vycházejí. V první části každé tabulky jsou metody založené na Gaussových-Kronrodových vzorcích, jedná se tedy o všechny testované metody z knihovny *QUADPACK*, metodu *quadgk* a *integral* softwaru Matlab a *GK* softwaru Mathematica. V druhé části každé tabulky jsou metody založené na Newtonových-Cotesových vzorcích. Jedná se o metody *quad* a *trapz* z Matlabu a *NC* ze softwaru Mathematica. V poslední části každé tabulky jsou metody založené na Lobattových vzorcích, tedy *quadl* z Matlabu a *LK* ze softwaru Mathematica. Každá metoda je přiřazena k danému softwaru podle příslušné zkratky - Fortran (F), Matlab (Mb) a Mathematica (Ma).

Výsledky všech použitých metod jsou vždy pro maximální testovanou relativní přesnost, formát *double* a nejvyšší stupeň optimalizace ($O3$, Fortran). Porovnávané metody jsou adaptivní i neadaptivní. Pro software Mathematica jsou srovnávány obě varianty adaptivity - globální i lokální. V první části jsou porovnané výsledky všech metod na vybrané 9. funkci, viz tabulka 3.28, a 2. funkce, viz tabulka 3.29. Dále porovnáváme výsledky integrace na nekonečném intervalu funkce číslo 6, viz tabulka 3.30, a funkce číslo 11, viz tabulka 3.31. Metody *quad*, *quadl* a všechny metody softwaru Fortran (mimo *dqagi*) neumožňují výpočet integrálu na nekonečném intervalu. Proto je hranice integrace omezena a funkce jsou testovány pro dané metody na konečném intervalu $\langle 0, 1000\rangle$. U funkce číslo 6 je rozdíl téměř neznatelný, protože absolutní hodnota integrálu na intervalu $\langle 1000, +\infty)$ je řádově 10^{-10858}. A absolutní hodnota integrálu pro funkci číslo 11 na intervalu $\langle 1000, +\infty)$ je řádově 10^{-6}. Všechny metody softwaru Mathematica umožňují výpočet integrálu funkce na nekonečném intervalu, nebylo ho tedy nutné nijak omezovat.

Poslední porovnání je věnováno integraci funkcí v komplexním oboru. Tabulka 3.32 obsahuje výsledky integrace 12. funkce, tabulka 3.33 výsledky pro 13. funkci a tabulky 3.34 výsledky integrace funkce číslo 14. Funkce číslo 14 je integrována v komplexním oboru na nekonečném intervalu. S touto variantou si použité metody softwarů Fortran a Matlab nedokázaly poradit a bylo nutné interval omezit na $\langle \mathrm{i}, 1000{+}\mathrm{i}\rangle$. Absolutní hodnota integrálu na zbývajícím intervalu $\langle 1000{+}\mathrm{i}, +\infty{+}\mathrm{i})$ je řádově 10^{-10858}. Všechny metody v softwaru Mathematica umožňují výpočet integrálu funkce na nekonečném intervalu i v komplexním oboru, nebylo ho tedy nutné nijak omezovat.

3.4.1 Porovnání použitých metod

V následující tabulce 3.28 jsou porovnány výsledky pro všechny testované metody. Jedná se o integraci na konečném intervalu pro vybranou funkci číslo 9. Z celkového porovnání výsledků je viditelné, že nejlépe v testu obstály adaptivní metody softwaru Fortran, které mají nejen nejpřesnější výsledek, ale dokonce i nejkratší dobu výpočtu. Většina metod soft-

waru Mathematica má také velmi dobré výsledky, ale metoda založená na Newtonových-Cotesových vzorcích má příliš velký počet vyčíslení. U softwaru Matlab jsou překvapením výsledky u metod *integral* a *quadgk*, které by měly být srovnatelné s ostatními adaptivními metodami.

Tabulka 3.28: Celkové porovnání použitých metod, softwary Fortran (optimalizace $O3$), Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená), integrace 9. funkce

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	dqk15	0.198355475102177	$3.486053208353967.10^{-2}$	$3.906.10^{-6}$	–
	dqk21	0.155296837086209	$8.198105932428146.10^{-3}$	$5.156.10^{-6}$	–
	dqk31	0.204290212468742	$4.079526945010484.10^{-2}$	$7.656.10^{-6}$	–
	dqk41	0.155541924583399	$7.953018435237741.10^{-3}$	$1.000.10^{-5}$	–
	dqk51	0.157586893212364	$5.908049806273491.10^{-3}$	$1.250.10^{-5}$	–
	dqk61	0.172777384008952	$9.282440990315000.10^{-3}$	$1.484.10^{-5}$	–
	dqag15	0.163494943018637	$2.775557561562891.10^{-17}$	$2.500.10^{-4}$	1155
	dqag21	0.163494943018637	$2.775557561562891.10^{-17}$	$2.500.10^{-4}$	1155
	dqag31	0.163494943018637	$2.775557561562891.10^{-17}$	$2.656.10^{-4}$	1155
	dqag41	0.163494943018637	$2.775557561562891.10^{-17}$	$2.500.10^{-4}$	1155
	dqag51	0.163494943018637	$2.775557561562891.10^{-17}$	$2.500.10^{-4}$	1155
	dqag61	0.163494943018637	$2.775557561562891.10^{-17}$	$2.656.10^{-4}$	1155
Mb	quadgk	0.163102243936939	$3.926990817000000.10^{-4}$	$1.342.10^{-3}$	7
	integral	0.163102243936938	$3.926990817000000.10^{-4}$	$3.701.10^{-3}$	870
Ma	GK, glob	0.163494943018637	$1.527877106952380.10^{-15}$	$4.000.10^{-3}$	1815
	GK, lok	0.163494943018637	$1.188348860962960.10^{-15}$	$3.984.10^{-3}$	1639
Mb	quad	0.163494943019167	$5.300000000000000.10^{-13}$	$1.869.10^{-2}$	1577
	trapz, 0.01	0.175988215950321	$1.249327293139000.10^{-2}$	$6.933.10^{-5}$	–
	trapz, 0.001	0.164103525247502	$6.085822288600000.10^{-4}$	$7.262.10^{-5}$	–
	trapz, 0.001	0.163494949298871	$6.280230000000000.10^{-9}$	$1.502.10^{-4}$	–
Ma	NC, glob	0.163494943018637	$6.201483412996730.10^{-13}$	0.116	39915
	NC, lok	0.163494943018637	$4.074338951873022.10^{-15}$	$2.156.10^{-2}$	6137
Mb	quadl	0.163494943018638	$< 10^{-15}$	$6.521.10^{-2}$	2268
Ma	LK, glob	0.163494943018637	$1.697641229947093.10^{-16}$	$5.828.10^{-3}$	2655
	LK, lok	0.163494943018637	$1.697641229947093.10^{-16}$	$6.156.10^{-3}$	2305

V tabulce 3.29 jsou porovnány výsledky integrace metody číslo 2. Adaptivní metody opět obstály v tomto testu lépe než neadaptivní. Nejpřesnější výsledek můžeme pozorovat u metody založené na Gaussových-Kronrodových vzorcích softwaru Mathematica s použitím globální adaptivity, ale metody založené na Newtonových-Cotesových vzorcích mají opět nejvyšší počet vyčíslení. Nejnižší počty vyčíslení jsou u metod softwaru Matlab. Nejrychlejší výpočet a velmi přesné výsledky mají opět adaptivní metody softwaru Fortran.

Tabulka 3.29: Celkové porovnání použitých metod, softwary Fortran (optimalizace $O3$), Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená), integrace 2. funkce

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	dqk15	0.399999982866058	$1.71339421539684.10^{-8}$	$1.094.10^{-6}$	–
	dqk21	0.399999997163017	$2.836982759824025.10^{-9}$	$9.375.10^{-7}$	–
	dqk31	0.399999999578007	$4.219932736582166.10^{-10}$	$1.250.10^{-6}$	–
	dqk41	0.399999999898660	$1.013403805316671.10^{-10}$	$1.719.10^{-6}$	–
	dqk51	0.399999999965505	$3.449540653122085.10^{-11}$	$2.344.10^{-6}$	–
	dqk61	0.399999999986063	$1.393724025078313.10^{-11}$	$2.500.10^{-6}$	–
	dqag15	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
	dqag21	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
	dqag31	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
	dqag41	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
	dqag51	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
	dqag61	0.400000000000000	$< 10^{-15}$	$1.719.10^{-5}$	345
Mb	quadgk	0.400000000000000	$< 10^{-15}$	$5.412.10^{-4}$	1
	integral	0.400000000000000	$< 10^{-15}$	$1.675.10^{-3}$	150
Ma	GK, glob	0.400000000000000	$1.387778780781446.10^{-16}$	$1.250.10^{-3}$	319
	GK, lok	0.399999999999917	$2.069178162145136.10^{-13}$	$1.141.10^{-3}$	187
Mb	quad	0.400000000002120	$2.120000000000000.10^{-12}$	$3.295.10^{-3}$	301
	trapz, 0.01	0.400012245153189	$1.224515319000000.10^{-5}$	$6.523.10^{-5}$	–
	trapz, 0.001	0.400000124194088	$1.241940900000000.10^{-7}$	$7.083.10^{-5}$	–
	trapz, 0.001	0.400000001247452	$1.247450000000000.10^{-10}$	$1.501.10^{-4}$	–
Ma	NC, glob	0.400000000000000	$1.110223024625157.10^{-15}$	$9.797.10^{-3}$	3885
	NC, lok	0.400000000000009	$2.359223927328458.10^{-14}$	$5.234.10^{-3}$	1325
Mb	quadl	0.400000000000297	$3.000000000000000.10^{-13}$	$4.540.10^{-1}$	258
Ma	LK, glob	0.400000000000004	$1.026956297778270.10^{-14}$	$1.453.10^{-3}$	405
	LK, lok	0.400000000000001	$3.608224830031759.10^{-15}$	$1.813.10^{-3}$	457

3.4.2 Porovnání použitých metod na nekonečném intervalu

V tabulce 3.30 jsou porovnány výsledky integrace 6. funkce na nekonečném intervalu (na omezeném intervalu $\langle 0, 1000 \rangle$ v případě metod *quad* a *quadl* z Matlabu a většiny metod z Fotranu (mimo *dqagi*), metody jsou v tabulce označené znakem *, např. *dqk*15*). Je zjevné, že neadaptivní metody softwaru Fortran nejsou pro integraci na velkém intervalu vhodné, v tomto testu naprosto selhávají. Nejlepší výsledky ve všech ohledech přinesla integrace metodou *dqagi*, za ní následují metody softwaru Mathematica. Ale metody založené na Newtonových-Cotesových vzorcích mají opět velký počet vyčíslení. Nejnižší počet vyčíslení a přesné výsledky poskytují metody *quadgk* a *integral* z Matlabu.

Tabulka 3.30: Celkové porovnání integrace 6. funkce na nekonečném (* a omezeném) intervalu, softwary Fortran (optimalizace $O3$), Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená)

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	dqk15*	$1.178555945780107.10^{-44}$	1.000000000000000	$9.375.10^{-7}$	–
	dqk21*	$3.881683817605077.10^{-22}$	1.000000000000000	$1.563.10^{-6}$	–
	dqk31*	$9.607394679090523.10^{-10}$	0.999999999039261	$2.031.10^{-6}$	–
	dqk41*	$2.458111024032043.10^{-5}$	0.999975418889760	$3.281.10^{-6}$	–
	dqk51*	$2.451292601308603.10^{-3}$	0.997548707398691	$3.306.10^{-6}$	–
	dqk61*	$2.758310590461750.10^{-2}$	0.972416894095382	$3.750.10^{-6}$	–
	dqag15*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$	465
	dqag21*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$	465
	dqag31*	1.000000000000000	$< 10^{-15}$	$1.563.10^{-5}$	465
	dqag41*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$	465
	dqag51*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$	465
	dqag61*	1.000000000000000	$< 10^{-15}$	$3.125.10^{-5}$	465
	dqagi	1.000000000000000	$2.220446049250313.10^{-16}$	$5.131.10^{-6}$	75
Mb	quadgk	1.000000000000000	$< 10^{-15}$	$5.521.10^{-4}$	1
	integral	1.000000000000000	$< 10^{-15}$	$5.521.10^{-4}$	1
Ma	GK, glob	1.000000000000001	$1.332267629550188.10^{-15}$	$2.972.10^{-2}$	209
	GK, lok	1.000000000000010	$9.769962616701378.10^{-15}$	$1.016.10^{-3}$	55
Mb	quad	NaN	NaN	$1.291.10^{-3}$	13
	quad*	1.000000000003364	$3.360000000000000.10^{-12}$	$9.885.10^{-3}$	897
Ma	NC, glob	1.000000000000001	$8.881784197001252.10^{-16}$	$1.734.10^{-3}$	9152
	NC, lok	1.000000000000002	$1.776356839400250.10^{-15}$	$1.766.10^{-3}$	1648

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
Mb	quadl	inf	inf	$1.109.10^{-3}$	13
	quadl*	1.000000000000001	$< 10^{-15}$	$5.276.10^{-3}$	738
Ma	LK, glob	1.000000000000000	$4.440892098500626.10^{-16}$	$2.344.10^{-3}$	420
	LK, lok	1.000000000000000	$1.110223024625157.10^{-16}$	$2.109.10^{-3}$	344

V tabulce 3.31 jsou porovnány výsledky integrace 11. funkce na nekonečném intervalu (na omezeném intervalu $\langle 0, 1000\rangle$ v případě metod *quad* a *quadl* z Matlabu a většiny metod z Fotranu (mimo *dqagi*), metody jsou v tabulce označené znakem *, např. *dqk*15*). Neadaptivní metody softwaru Fortran v tomto testu opět selhaly. Nejpřesnější výsledek přinesla metoda *NC* softwaru Mathematica za použití globální adaptivity, ovšem za cenu obrovského počtu vyčíslení a velké časové náročnosti výpočtu. Vzhledem ke všem testovaným kritériím jsme získali nejlepší výsledky pomocí metod *quadgk* a *integral* z Matlabu.

Tabulka 3.31: Celkové porovnání integrace 11. funkce na nekonečném (* a omezeném) intervalu, softwary Fortran (optimalizace $O3$), Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená)

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	dqk15*	$4.441015361815842.10^{-3}$	0.495558984638184	$6.250.10^{-7}$	–
	dqk21*	$8.763940428267520.10^{-3}$	0.491236059571732	$9.375.10^{-7}$	–
	dqk31*	$1.904817247088206.10^{-2}$	0.480951827529118	$1.563.10^{-6}$	–
	dqk41*	$3.334904355527137.10^{-2}$	0.466650956444729	$1.875.10^{-6}$	–
	dqk51*	$5.148237634172964.10^{-2}$	0.448517623658270	$2.344.10^{-6}$	–
	dqk61*	$7.349329102591214.10^{-2}$	0.426506708974088	$2.656.10^{-6}$	–
	dqag15*	0.499993619888687	$6.380111312875858.10^{-6}$	$3.125.10^{-5}$	585
	dqag21*	0.499993619888687	$6.380111312875858.10^{-6}$	$1.563.10^{-5}$	585
	dqag31*	0.499993619888687	$6.380111312875858.10^{-6}$	$3.125.10^{-5}$	585
	dqag41*	0.499993619888687	$6.380111312875858.10^{-6}$	$1.563.10^{-5}$	585
	dqag51*	0.499993619888687	$6.380111312875858.10^{-6}$	$1.563.10^{-5}$	585
	dqag61*	0.499993619888687	$6.380111312875858.10^{-6}$	$3.125.10^{-5}$	585
	dqagi	0.499999986086233	$1.391376713844039.10^{-8}$	$1.109.10^{-5}$	195
Mb	quadgk	0.500000000000000	$< 10^{-15}$	$6.879.10^{-4}$	2
	integral	0.500000000000000	$< 10^{-15}$	$7.573.10^{-4}$	2
Ma	GK, glob	0.500000000000000	$1.332267629550188.10^{-15}$	$1.969.10^{-3}$	539
	GK, lok	0.500000000000000	$1.332267629550188.10^{-15}$	$1.734.10^{-3}$	407

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
Mb	quad	NaN	NaN	$1.291.10^{-3}$	13
	quad*	0.499993633802405	$6.366197600000000.10^{-5}$	$1.769.10^{-2}$	1557
Ma	NC, glob	0.499999999999999	$5.551115123125783.10^{-16}$	$5.720.10^{-2}$	16495
	NC, lok	0.499999999999883	$2.338129689860580.10^{-13}$	$1.219.10^{-2}$	3184
Mb	quadl	inf	inf	$1.109.10^{-3}$	13
	quadl*	0.499993633802277	$6.366197720000000.10^{-5}$	$1.195.10^{-2}$	1608
Ma	LK, glob	0.499999999999977	$4.451994328746878.10^{-14}$	$6.547.10^{-3}$	1362
	LK, lok	0.499999999999992	$1.598721155460225.10^{-14}$	$4.500.10^{-3}$	1352

3.4.3 Porovnání výsledků integrace v komplexním oboru

Výsledky integrace funkcí v komplexním oboru jsou porovnány v následujících tabulkách. Pomocí programovacího jazyku Fortran jsme otestovali dané funkce pomocí upravených neadaptivních metod (např. metoda *cdqk*15), které umožňují výpočet integrace v komplexním oboru. Základní knihovna QUADPACK takové metody neobsahuje. Výsledky integrace 12. funkce jsou znázorněny v tabulce 3.32. Nejpřesnější výsledky jsme získali pomocí metody *NC* ze software Mathematica s použitím lokální adaptivity. Ovšem délka výpočtu patří k nejpomalejším a počet vyčíslení také není ideální.

V tabulce 3.33 jsou zaznamenány výsledky integrace 13. funkce. V tomto případě jsou výsledky velmi podobné jako u předchozí funkce. Při celkovém porovnání přesnosti, doby výpočtu a počtu vyčíslení mají nejlepší výsledky metody *quadgk* a *integral* z Matlabu.

Tabulka 3.34 obsahuje srovnání výsledků integrace 14. funkce. Tato funkce je integrovaná v komplexním oboru na nekonečném intervalu. S touto variantou si použité metody softwarů Fortran a Matlab nedokázaly poradit a bylo nutné interval omezit na $\langle \mathrm{i}, 1000{+}\mathrm{i} \rangle$, metody jsou v tabulce označeny znakem *, např. *cdqk*15*). Absolutní hodnota integrálu na zbývajícím intervalu $\langle 1000{+}\mathrm{i}, +\infty{+}\mathrm{i} \rangle$ je řádově 10^{-10858}. Metody softwaru Mathematica umožňují výpočet integrálu funkce v komplexním oboru na nekonečném intervalu, nebylo ho tedy nutné nijak omezovat. Upravené metody softwaru Fortran pro výpočet integrálu v komplexním oboru v tomto testu naprosto selhaly, je zjevné, že nejsou vhodné pro integraci na velkém intervalu. Nejlepší výsledky vzhledem k přesnosti, rychlosti výpočtu a počtu vyčíslení jsme získali pomocí metody *LK* softwaru Mathematica s použitím lokální adaptivity.

Tabulka 3.32: Celkové porovnání integrace 12. funkce v komplexním oboru, softwary Fortran, Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená)

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	cdqk15	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$2.031.10^{-6}$	–
	cdqk21	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$2.656.10^{-6}$	–
	cdqk31	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$3.750.10^{-6}$	–
	cdqk41	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$5.000.10^{-6}$	–
	cdqk51	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$6.094.10^{-6}$	–
	cdqk61	0.130584240443723 + 0.500000000000000i	$4.840060019439818.10^{-10}$	$7.188.10^{-6}$	–
Mb	quadgk	0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$5.664.10^{-4}$	1
	integral	0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$6.514.10^{-4}$	1
Ma	GK, glob	0.130584240443722 + 0.500000000000000i	$1.121489901770251.10^{-15}$	$8.594.10^{-4}$	33
	GK, lok	0.130584240443722 + 0.500000000000000i	$1.121489901770251.10^{-15}$	$9.688.10^{-4}$	33
Mb	quad	0.130584240443724 + 0.500000000000000i	$< 10^{-15}$	$1.773.10^{-2}$	121
Ma	NC, glob	0.130584240443722 + 0.500000000000000i	$1.611288759788450.10^{-16}$	$5.000.10^{-3}$	1065
	NC, lok	0.130584240443722 + 0.500000000000000i	$5.370962532628168.10^{-17}$	$2.281.10^{-3}$	257
Mb	quadl	0.130584240443723 + 0.500000000000000i	$< 10^{-15}$	$4.258.10^{-4}$	48
Ma	LK, glob	0.130584240443722 + 0.499999999999999i	$3.267028964510625.10^{-16}$	$9.688.10^{-4}$	63
	LK, lok	0.130584240443722 + 0.499999999999999i	$3.267028964510625.10^{-16}$	$1.094.10^{-3}$	65

Tabulka 3.33: Celkové porovnání integrace v komplexním oboru, softwary Fortran, Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená), integrace 13. funkce

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	cdqk15	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$1.875.10^{-6}$	–
	cdqk21	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$2.656.10^{-6}$	–
	cdqk31	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$3.750.10^{-6}$	–
	cdqk41	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$5.000.10^{-6}$	–
	cdqk51	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$6.094.10^{-6}$	–
	cdqk61	0.348992856673883 – 0.2472886217781412i	$2.3171169982738604.10^{-9}$	$7.344.10^{-6}$	–
Mb	quadgk	0.348992856673883 – 0.2472886217781412i	$< 10^{-15}$	$5.712.10^{-4}$	1
	integral	0.348992856673883 – 0.2472886217781412i	$< 10^{-15}$	$6.903.10^{-4}$	1
Ma	GK, glob	0.348992856673883 – 0.2472886217781412i	$1.116432948315769.10^{-15}$	$8.906.10^{-4}$	33
	GK, lok	0.348992856673883 – 0.2472886217781412i	$1.116432948315769.10^{-15}$	$1.047.10^{-3}$	55
Mb	quad	0.348992856673884 – 0.2472886217781412i	$< 10^{-15}$	$1.794.10^{-2}$	129
Ma	NC, glob	0.348992856673882 – 0.2472886217781412i	$6.121837435030051.10^{-16}$	$9.844.10^{-4}$	63
	NC, lok	0.348992856673882 – 0.2472886217781412i	$6.489134702875428.10^{-17}$	$5.656.10^{-3}$	1001
Mb	quadl	0.348992856673883 – 0.2472886217781412i	$< 10^{-15}$	$4.162.10^{-4}$	48
Ma	LK, glob	0.348992856673882 – 0.2472886217781412i	$1.835404460974467.10^{-16}$	$9.688.10^{-4}$	63
	LK, lok	0.348992856673882 – 0.2472886217781412i	$1.835404460974467.10^{-16}$	$1.078.10^{-3}$	65

Tabulka 3.34: Celkové porovnání integrace v komplexním oboru na nekonečném (* a omezeném) intervalu, softwary Fortran, Matlab, Mathematica, maximální relativní přesnost (adaptivní metody 10^{-12}, neadaptivní metody defaultně nastavená), integrace 14. funkce

SW	Metoda	Výsledek	Chyba	Čas $[s]$	Vyč.
F	cdqk15*	$1.168187967395657.10^{-44}$ – $1.559839420121332.10^{-45}$ i	0.991202811863483	$4.688.10^{-6}$	–
	cdqk21*	$3.847535914775095.10^{-22}$ – $5.137476465862367.10^{-23}$ i	0.991202811863483	$6.875.10^{-6}$	–
	cdqk31*	$9.522876620596702.10^{-10}$ – $1.271555499657663.10^{-10}$ i	0.991202810911195	$1.000.10^{-5}$	–
	cdqk41*	$2.436486558893165.10^{-5}$ – $3.253352959652700.10^{-6}$ i	0.991178446997894	$1.688.10^{-5}$	–
	cdqk51*	$2.429728119117216.10^{-3}$ – $3.244328657849163.10^{-4}$ i	0.988773083744365	$2.063.10^{-5}$	–
	cdqk61*	$2.734045213258485.10^{-2}$– $3.650672339608343.10^{-3}$ i	0.963862359730898	$1.922.10^{-5}$	–
Mb	quadgk*	0.991202811863474 + 0.132351750097773i	$< 10^{-15}$	$1.950.10^{-3}$	11
	integral*	0.991202811863474 + 0.132351750097773i	$< 10^{-15}$	$2.031.10^{-3}$	11
Ma	GK, glob	0.9912028118634753 + 0.132351750097773i	$1.680072297996111.10^{-15}$	$1.563.10^{-3}$	209
	GK, lok	0.9912028118634753 + 0.132351750097773i	$1.680072297996111.10^{-15}$	$1.641.10^{-3}$	231
Mb	quad*	0.991202811866808 + 0.132351750098218i	0.000000000003360	$1.306.10^{-2}$	897
Ma	NC, glob	0.991202811863474 + 0.132351750097773i	$9.091804728055929.10^{-16}$	$4.475.10^{-2}$	9152
	NC, lok	0.991202811863475 + 0.132351750097773i	$1.676629758422416.10^{-15}$	$9.438.10^{-3}$	1648
Mb	quadl*	0.991202811863475 + 0.132351750097773i	$< 10^{-15}$	$6.954.10^{-3}$	738
Ma	LK, glob	0.991202811863473 + 0.132351750097773i	$2.237726045655905.10^{-16}$	$3.031.10^{-3}$	420
	LK, lok	0.991202811863473 + 0.132351750097773i	$2.775557561562891.10^{-17}$	$2.563.10^{-3}$	344

3.4.4 Implementační složitost

V této části je krátké porovnání implementační složitosti v jednotlivých softwarech. U programovacího jazyku Fortran je nutné, aby byla každá proměnná přesně deklarována. Pokud některá z proměnných není deklarována, software nenahlásí chybu, ale za danou proměnnou zvolí hodnotu uloženou v paměti, která se pro nás jeví jako náhodná. Je také nutný přesný zápis čísel ve formátu *double* ve tvaru např. 2.0 místo 2, významný rozdíl se projeví především při jejich umocňování. V softwaru Fotran jsme museli upravit metody pro výpočet integrace funkce v komplexním oboru. U neadaptivních metod bylo nutné změnit deklaraci proměnných na formát $complex * 16$. Pro zaznamenávání výsledků jsme použili jejich ukládání do textových souborů.

Software Matlab není tolik citlivý na zápis číslic a není zde nutné deklarovat proměnné. Komplexní číslo i je zde nutné zapisovat ve tvaru $1i$, aby nedošlo k záměně v případě, že je tento parametr použit pro označení jiné proměnné např. typu *integer* např. ve $for-cyklu$. Pro zaznamenávání výsledků jsme použili jejich ukládání do textových souborů.

V softwaru Mathematica také není nutné přesně deklarovat proměnné. Při implementaci je nutné zachovávat správný zápis, který se týká především velkých písmen, např. komplexní číslo i je nutné zapsat ve tvaru I. Pro výpis výsledků byla použita přehledná tabulka, kterou jsme vytvořili pomocí vzoru na [20]. Pro přehlednost výsledků jsme textové soubory, získané z Fortranu a Matlabu, převedli pomocí softwaru Mathematica také do zmíněných tabulek, které jsou obsaženy na přiloženém CD.

3.5 Integrace v Batesově-Lewisově modelu

V této části je ukázané chování numerické integrace v případě, že funkce není přesně vyčíslena. Tento problém demonstrujeme na finančním modelu, který já známý jako Batesův stochastický nestabilní skokový model. Jedná se o model cen akcií, který vytvořil David Bates, viz [22]. Nejprve si tento model představíme

$$V(S,\upsilon,\tau) = S - Ke^{-r\tau}\frac{1}{2\pi}\int_{-\infty+ik_i}^{\infty+ik_i} e^{-ikX}\frac{H(k,\upsilon,\tau)}{k^2-ik}e^{\lambda(\varphi(-k)-1)\tau}\mathrm{d}k, \tag{3.1}$$

kde $k_i = \frac{1}{2}$, i je imaginární jednotka a

$$X = ln\frac{S}{K} + (r-\lambda\beta)\tau,$$

$$\beta = exp\left\{\mu_J + \frac{1}{2}\sigma_J^2\right\} - 1,$$

$$\varphi(u) = exp\left\{i\mu_J u - \frac{1}{2}\sigma_J^2 u^2\right\},$$

$$H(k,\upsilon,\tau) = exp\left\{\frac{2\kappa\theta}{\sigma^2}\left[qg - ln\left(\frac{1-he^{-\xi q}}{1-h}\right)\right] + \upsilon g\left(\frac{1-e^{-\xi q}}{1-he^{-\xi q}}\right)\right\},$$

$$g = \frac{b-\xi}{2}, \quad h = \frac{b-\xi}{b+\xi}, \quad q = \frac{\sigma^2\tau}{2},$$

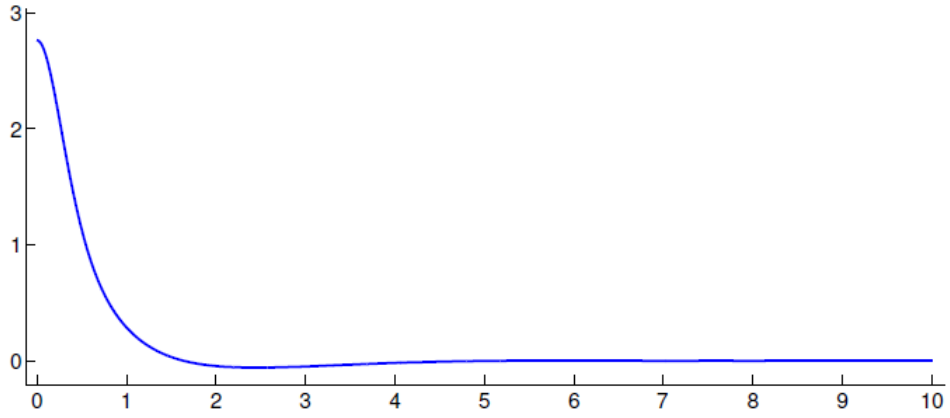
$$\xi = \sqrt{b^2 + \frac{4(k^2-ik)}{\sigma^2}},$$

$$b = \frac{2}{\sigma^2}\left(ik\rho\sigma + \kappa\right).$$

Graf integrované funkce je znázorněn na obrázku 3.1. Vzhledem ke komplikovanosti daného modelu byl test proveden pouze v softwaru Matlab. Je ale velmi pravděpodobné, že bychom se s demonstrovanými problémy setkali i v jiných matematických softwarech. Model jsme testovali s použitím metody *integral*. Batesův model je závislý na velkém množství parametrů, jejichž volbou je výpočet silně ovlivněn. Jedním z nejvíce problematických parametrů je σ. Představme si dvě sady testovaných parametrů ve formátu *double*, které se liší pouze ve volbě parametru σ [23].

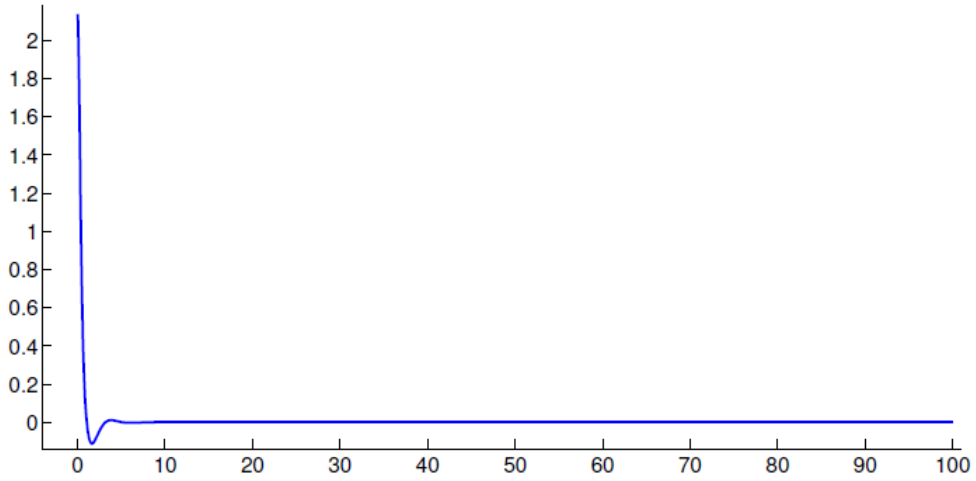
První testovaná sada parametrů $x_0 : S = 3721.8$, $\upsilon = 0.97114$, $\tau = 0.120548$, $K = 6250$, $r = 0.009$, $\lambda = 11.70271$, $\mu_J = -6.65876$, $\sigma_J = 1.0069$, $\theta = 0.95333$, $\kappa = 17.6751$, $\rho = -0.86219$, $\sigma = 0.000028$.

Druhá testovaná sada parametrů x_1: $S = 3721.8$, $\upsilon = 0.97114$, $\tau = 0.120548$, $K = 6250$, $r = 0.009$, $\lambda = 11.70271$, $\mu_J = -6.65876$, $\sigma_J = 1.0069$, $\theta = 0.95333$, $\kappa = 17.6751$, $\rho = -0.86219$, $\sigma = 0.000022$.



Obrázek 3.1: Funkce vykreslená na intervalu $\langle 0, 10\rangle$ [23]

Zásadní rozdíl při testování dvou zmíněných sad parametrů je v počtu vyčíslení funkce. V případě sady x_0 byl počet vyčíslení 150 a v případě sady x_1 nesrovnatelných 127680. Na tomto příkladě je znatelné, jak moc je model citlivý na zvolené parametry. Při pohledu na obrázek 3.1 se zdá, že je Batesův model hladký. Při globálnějším pohledu je ale vidět, že funkce není tak hladká, jak se zdálo, viz obrázek 3.2. Tento jev je způsoben nepřesnou aritmetikou.



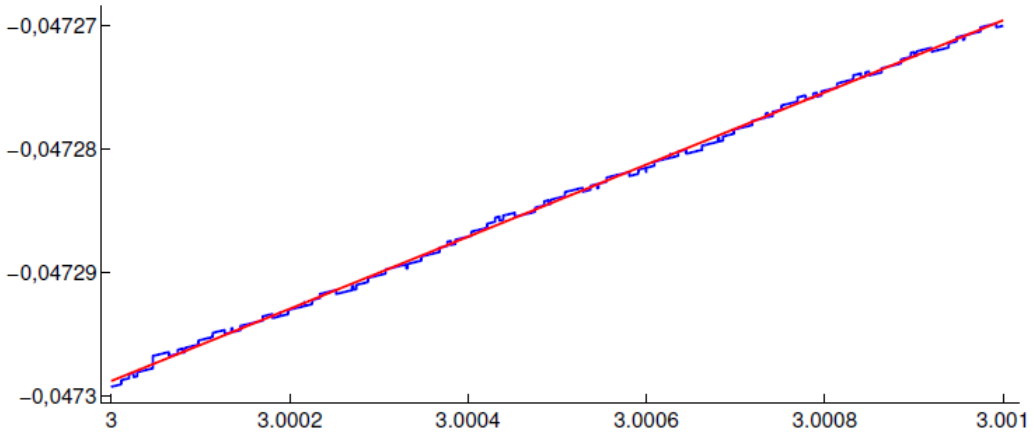
Obrázek 3.2: Globálnější pohled na funkci [23]

Test pro sady parametrů x_0 a x_1 byl proveden ve standardním formátu *double*. Ověřili jsme, že řešením tohoto problému je použití přesnější aritmetiky *vpa* (variable-precision arithmetic). V tabulce 3.35 můžeme pozorovat výsledky integrace s použitím formátu *double* a aritmetiky *vpa*. Je zjevné, že použití přesnější aritmetiky přináší značné zlepšení.

Tabulka 3.35: (Bates-Lewis) Porovnání výsledků integrace Batesova modelu pro různé parametry σ, výpočet s použitím formátu *double* a přesnější aritmetiky *vpa*

σ	Formát	Výsledek	Čas $[s]$	Vyčíslení
0.000028	double	0.776588380039885	0.73	150
	vpa	0.776585856534337	4.63	180
0.000022	double	0.776583174460231	3.82	127680
	vpa	0.776585856549349	4.49	180

Použitím přesnější aritmetiky jsme docílili výrazného zlepšení. Na obrázku 3.3 můžeme pozorovat vliv aritmetiky na daný model. Při výpočtu ve formátu *double* dochází k velkým nepřesnostem a kvůli tomu není funkce hladká. Použitím přesnější aritmetiky *vpa* dojde k odstranění těchto nepřesností.



Obrázek 3.3: Vliv použití přesnější aritmetiky [23]

Použití přesnější aritmetiky *vpa* ovšem zvyšuje časovou náročnost výpočtu. Proto jsme se rozhodli otestovat parametry modelu, pro které je nutné danou aritmetiku použít. Problémové úseky integrace nastávají především při výpočtu parametrů b a ξ. V obou případech dochází k dělení umocněným parametrem σ, tím získáme velkou hodnotu, kterou následně přičítáme k řádově o dost nižší hodnotě. Při výpočtu těchto dvou parametrů ve formátu *double* dochází k velkým aritmetickým chybám. Použití přesnější aritmetiky na dané dva parametry přineslo přesnější výsledek, ale časová náročnost se nesnížila. Zjistili jsme, že pokud se aritmetika *vpa* použije na jediný parametr modelu, tak je použita i pro všechny následující výpočty. Daný parametr, vypočítaný přesněji v aritmetice *vpa*, je nutné převést opět do formátu *double*, aby nebyla aritmetika *vpa* použita při dalším výpočtu a tím nedocházelo ke zpomalení v úsecích, kde přesnější výpočet není nutný.

Integrace v Batesově modelu byla provedena pomocí metody *integral* v softwaru Matlab. Metoda *integral* je založena na 15-ti bodovém Gaussově-Kronrodově pravidle a před integrací dochází k rozdělení počátečního intervalu na 10 dílčích podintervalů. Testovali jsme vliv použití dalších Gaussových-Kronrodových vzorců s použitím jiného počtu bodů a různá dělení počátečního intervalu. Pro tyto testy bylo nutné zasáhnout do kódu softwaru Matlab a vytvořit metody založené na výpočtu 7-mi, 17-ti a 25-ti bodových vzorců. Metoda *integral* obsahuje pro výpočet pomocné metody s názvy $integralParseArgs$ a $integralCalc$. Pro otestování daných změn, bylo nutné upravit zdrojový kód metody $integralParseArgs$. Upravení defaultně nastaveného parametru `s.Rule = Gauss7Kronrod15` umožňuje použití vzorce s jiným počtem bodů, např. na `s.Rule = Gauss8Kronrod17` . Úprava parametru `s.InitialIntervalCount=10` umožňuje nastavení různého počtu dělení počátečního intervalu.

V tabulce 3.36 jsou porovnány výsledky výpočtu Batesova modelu pro sadu parametrů x_0 s různým nastavením dělení počátečního intervalu. Výpočet je proveden s použitím přesnější aritmetiky *vpa* nejprve na všechny a poté pouze na část parametrů modelu. Použití aritmetiky pouze na kritické parametry přineslo ve všech případech především snížení časové náročnosti výpočtu. Počet dělení počátečního intervalu měl také vliv na výpočetní čas, nejlepší hodnoty jsme získali pro dělení na 3 podintervaly.

Tabulka 3.36: (Bates-Lewis) Porovnání výpočtu s aritmetikou *vpa* na všechny a pouze na část parametrů modelu, různý počet dělení počátečního intervalu, sada parametrů x_0

Děl.	**Celkové použití vpa**			**Částečné použití vpa**		
int.	Výsledek	Čas $[s]$	Vyčíslení	Výsledek	Čas $[s]$	Vyčíslení
1	0.776585856534361	**5.018**	135	0.776585856534361	**3.984**	135
2	0.776585856534361	**5.098**	120	0.776585856534361	**3.632**	120
3	0.776585856534319	**4.150**	105	0.776585856534320	**3.260**	105
4	0.776585856534361	**4.585**	120	0.776585856534361	**3.651**	135
5	0.776585856534333	**5.239**	135	0.776585856534333	**4.121**	120
6	0.776585856534316	**4.529**	120	0.776585856534316	**3.718**	120
7	0.776585856534351	**5.012**	135	0.776585856534351	**4.041**	135
8	0.776585856534339	**5.523**	150	0.776585856534339	**4.418**	150
9	0.776585856534337	**5.899**	165	0.776585856534337	**4.788**	165
10	0.776585856534337	**6.418**	180	0.776585856534337	**5.192**	180
12	0.776585856534300	**6.171**	180	0.776585856534300	**5.070**	180
14	0.776585856534340	**7.636**	210	0.776585856534341	**5.652**	210
16	0.776585856534338	**8.697**	240	0.776585856534337	**6.313**	240
18	0.776585856534337	**10.096**	270	0.776585856534337	**7.092**	270
20	0.776585856534337	**10.276**	300	0.776585856534337	**7.830**	300

Další test byl zaměřen na změnu výpočetního vzorce metody *integral*. Testovali jsme použití 7-mi, 15-ti, 17-ti a 25-ti bodových Gaussových-Kronrodových vzorců. V tabulce 3.37 jsou porovnány výsledky výpočtů v Batesově modelu pro sadu parametrů x_0 s různým nastavením dělení počátečního intervalu. Výpočet byl proveden s použitím přesnější aritmetiky *vpa* pouze na kritické parametry modelu. Při použití 7-mi bodového vzorce docházelo ve všech testovaných parametrech ke zhoršení výsledku. Vzhledem k počtu vyčíslení a časové náročnosti bylo pro nižší počet dělení intervalu dosaženo nejlepších výsledků s použitím defaultního 15-ti bodového vzorce. Chyba přesného výsledku byla v daném případě řádově 10^{-14}, kde docházelo k odchylce od přesného řešení, tzn. od hodnoty 0.776585856534337. Při ponechání defaultně nastaveného počtu dělení na 10 podintervalů, docházelo při použití 17-ti bodového vzorce ke zlepšení všech testovaných hodnot.

Tabulka 3.37: (Bates-Lewis) Porovnání výpočtu s použitím aritmetiky *vpa* pouze na část parametrů modelu, různé vzorce a počty dělení intervalu, sada parametrů x_0

Int.	Vzorec	Výsledek	Čas $[s]$	Vyčíslení
1	3 - 7	0.776585856534353	7.381	287
	7 - 15	0.776585856534361	4.056	135
	8 - 17	0.776585856534337	4.375	153
	12 - 25	0.776585856534337	4.881	175
3	3 - 7	0.776585856533987	6.357	245
	7 - 15	0.776585856534320	3.265	105
	8 - 17	0.776585856534338	3.637	119
	12 - 25	0.776585856534336	3.745	125
6	3 - 7	0.776585856533987	5.955	224
	7 - 15	0.776585856534316	3.589	120
	8 - 17	0.776585856534338	3.944	136
	12 - 25	0.776585856534336	4.284	150
10	3 - 7	0.776585856534337	6.751	252
	7 - 15	0.776585856534361	7.256	180
	8 - 17	0.776585856534324	4.887	170
	12 - 25	0.776585856534337	6.612	250

Ověřili jsme, že použití přesnější aritmetiky je při integraci v Batesově modelu velmi důležité. Použití *vpa* ovšem není nutné pro všechny parametry modelu, dochází tím ke zpomalování výpočtu. Testování různého počtu dělení intervalu a použití různých integračních vzorců ukázalo, že některé kombinace daných nastavení mohou přispět ke zlepšení výpočtu. Test byl ovšem proveden pouze na jedné sadě parametrů a výsledná zjištění nemusí platit pro všechny hodnoty.

Závěr

Cílem této bakalářské práce bylo otestovat a porovnat metody výpočtu numerické integrace v různých matematických softwarech. Otestovali jsme výpočet v softwarech Matlab a Mathematica a v programovacím jazyce Fortran na vybraných benchmarkových úlohách. Výsledky jsme nejprve demonstrovali a porovnali zvlášť v jednotlivých softwarech.

Pro testování funkcí pomocí Fortranu byla použita základní knihovna QUADPACK, která obsahuje adaptivní i neadaptivní metody založené na Gaussových-Kronrodových vzorcích ve formátech *double* a *real*. Ve Fortranu je možné nastavit různé optimalizace překladu ($O1$, $O2$, $O3$), kdy by každé nastavení mělo přinést různá zlepšení. Naše testy vliv volby optimalizace příliš nepotvrdily. Při zvolení libovolné optimalizace došlo v některých případech ke snížení časové náročnosti a mírně i ke zvýšení přesnosti výpočtu. Na počet vyčíslení neměla optimalizace žádný vliv. Dalším testovaným parametrem byl vliv relativní přesnosti. S rostoucí požadovanou relativní přesností se zvyšovala přesnost výsledku, ale i počet vyčíslení funkce. Pro test integrace na nekonečném intervalu byla použita metoda k tomu uzpůsobená, která podávala nejlepší výsledky, a ostatní metody, pro které musel být interval integrace omezen. Při zvětšujícím se intervalu začaly selhávat neadaptivní metody. Oproti tomu adaptivní metody měly velmi dobré výsledky během celého testu. Základní knihovna QUADPACK neobsahuje žádné metody pro integraci v komplexním oboru, proto jsme si upravili neadaptivní metody, aby bylo možné tento test provést. Až na integraci poslední funkce definované v komplexním oboru na nekonečném intervalu jsme získali celkově dobré výsledky. U poslední funkce muselo dojít k omezení intervalu a v tomto případě neadaptivní metody selhaly.

V matematickém softwaru Matlab jsme otestovali různé adaptivní i neadaptivní metody pro numerickou integraci s použitím různých nastavení relativní přesnosti. S rostoucí relativní přesností se opět zvyšovala přesnost výsledku i počet vyčíslení. V některých případech ale rostl řádově i čas výpočtu. Adaptivní metody *integral* a *quadgk* jsou založené na stejném principu výpočtu. Jediné, v čem se výsledky výpočtu lišily, byl počet vyčíslení. To bylo ovšem způsobeno tím, že u metody *quadgk* je defaultně nastaveno vektorové vyčíslení funkce. Neadaptivní metoda *trapz* umožňuje nastavení jemnosti dělení intervalu. Se zvyšující se jemností dělení docházelo ke zpřesňování výsledku, ale i ke zvyšování časové náročnosti výpočtu. Problém při výpočtu nevlastního integrálu jsme vyřešili posunutím počátku integrace o malé ε. Metody *quad* a *quadl* nejsou uzpůsobené pro integraci na

nekonečném intervalu, proto musel být opět omezen. Metoda *quadl* ani poté nepodávala příliš dobré výsledky, u ostatních metod byl výpočet znatelně lepší. Stejný problém nastal při integraci poslední funkce v komplexním oboru na nekonečném intervalu.

V softwaru Mathematica jsme testovali vliv nastavení adaptivity. Ve většině případů jsme získali velmi podobné výsledky. V některých případech měla ale lokální adaptivita za cenu vyššího počtu vyčíslení lepší přesnost výsledků. Vliv relativní přesnosti byl viditelně prokazatelný. S rostoucí relativní přesností se zvyšovala přesnost výpočtu i počet vyčíslení. V některých případech docházelo i k řádově vyšší časové náročnosti výpočtu. U žádné z metod nenastal problém s integrací funkcí na nekonečném intervalu nebo v komplexním oboru. Přesnost výpočtu byla u všech metod velmi dobrá. Metoda založená na Gaussově-Kronrodově pravidle měla ve většině případů nejnižší počet vyčíslení a dobu výpočtu.

Celkové srovnání metod všech použitých softwarů jsme nejprve provedli pro integraci na konečném intervalu. Při porovnání výsledků integrace funkce číslo 9 nás velmi překvapil nepřesný výsledek metod *integral* a *quadgk* softwaru Matlab. U ostatních adaptivních metod byla přesnost výsledků velmi vysoká a proto rozhodoval čas výpočtu a počet vyčíslení. Vzhledem ke všem testovaným kritériím měly nejlepší výsledky adaptivní metody softwaru Fortran. Podobně dopadlo porovnání výsledků integrace 2. funkce. V tomto případě měly i metody *integral* a *quadgk* velmi vysokou přesnost výpočtu. Při srovnání výsledků integrace na nekonečném intervalu (pro některé metody na omezeném intervalu) měla u testu 6. funkce nejlepší výsledky metoda *dqagi* softwaru Fortran. Hlavní rozdíl od ostatních metod byl především v rychlosti výpočtu. U testu 11. funkce měla tato metoda také nejrychlejší výpočet, ale přesnost již nebyla dostačující. Vzhledem ke všem testovaným kritériím dopadly nejlépe metody *integral* a *quadgk* z Matlabu. Podobné výsledky jsme získali pro integraci v komplexním oboru. Ovšem u integrace v komplexním oboru na nekonečném intervalu musel být interval omezen i pro metody *integral* a *quadgk*.

Adaptivní metody měly při porovnání výsledků jasně lepší výsledky než neadaptivní. Přesnost adaptivních metod byla u všech testů velmi vysoká, proto hlavním rozhodovacím kritériem byl počet vyčíslení a doba výpočtu. Nejkratší dobu výpočtu měly metody softwaru Fortran. S komplikovanějšími úlohami ale klesala přesnost výpočtu. Metody softwaru Matlab měly až na jednu výjimku velmi stabilní výsledky. Metody softwaru Mathematica měly ve většině případech nejpřesnější výsledky, ale vzhledem k tomu, že testy probíhaly ve formátu *double*, tak pro nás již není přesnost na vyšší počet cifer prioritou. Vyšší přesnost výsledku se bohužel projevila v počtu vyčíslení a době výpočtu.

V poslední části této práce jsme ukázali vliv přesnější aritmetiky při výpočtu numerické integrace. Při použití standardního formátu *double* docházelo k velkým nepřesnostem. Přesnější aritmetika ovšem zvyšovala délku výpočtu. Pomocí testů jsme ověřili, že aritmetiku *vpa* stačí použít pouze na kritické parametry modelu. Tím se výpočet stabilizuje a nedochází k nepřesnostem. Výpočet byl proveden pouze v softwaru Matlab. Při změně defaultního nastavení v metodě *integral* jsme získali lepší výsledky výpočtu.

Literatura

[1] *HEATH, M. T.:* **Scientific computing - An introductory survey.**
Boston : McGraw-Hill 2002. ISBN 0-07-112229.

[2] *QUARTERONI, A., SACCO, R., SALERI, F.:* **Numerical mathematics.**
2nd ed. Berlin 2007. ISBN 978-3-540-34658-6.

[3] *QUARTERONI, A., SALERI, F., GERVASIO, P.:* **Scientific computing with MATLAB and Octave.**
3rd ed. Berlin 2010. ISBN 978-3-642-12429-7.

[4] *DAVIS, P. J., RABINOWITZ, P. :* **Methods of Numerical Integration.**
2nd ed. Academic Press 1984.

[5] *SHAMPINE, L. F.:* **Vectorized Adaptive Quadrature in MATLAB.**
Journal of Computational and Applied Mathematics 211. roč. 2008. pp. 131-140.

[6] *PIESSENS, R., DE DONCKER-KAPENGA, E., ÜBERHUBER, CH. W., KAHANER, D.:* **QUADPACK: A subroutine package for automatic integration.**
Springer-Verlag. 1983. ISBN 987-3-540-12553-2.

[7] *BRANDNER, M.:* **Numerické metody.**
Přednášky. Západočeská univerzita. Plzeň 2015.

[8] *PŘIKRYL, P., BRANDNER, M.:* **Numerické metody II.**
Skripta. Západočeská univerzita. Plzeň 2001.

[9] *VITÁSEK, E.:* **Numerické metody.**
SNTL. Praha 1987. ISBN 04-009-87.

[10] *WEISSTEIN, E. W.:* **Gaussian Quadrature.**
A Wolfram Web Resource.
Dostupné z http://mathworld.wolfram.com/GaussianQuadrature

[11] *RALSTON, A.:* **Základy numerické matematiky.**
Academia. Praha 1973. ISBN 104-21-825.

[12] *VICHER, M.:* **Numerická matematika.**
Skripta. Univerzita J. E. Purkyně. Ústí nad Labem 2003.

[13] *LAURIE, D.:* **Calculation of Gauss-Kronrod quadrature rules.**
Mathematics of Computation of the American Mathematical Society. 1997.

[14] *WEISSTEIN, E. W.:* **Lobatto Quadrature.**
A Wolfram Web Resource.
Dostupné z http://mathworld.wolfram.com/LobattoQuadrature

[15] *ESPELID, T. O.:* **A test of QUADPACK and Four Doubly Adaptive Quadrature Routines.**
Bergen 2004. roč. 281. ISSN 0333-3590.

[16] *NETLIB :* **QUADPACK.**
Netlib is a collection of mathematical software, papers, and databases.
Dostupné z http://www.netlib.org/quadpack/

[17] *BURKARDT, J. :* **QUADPACK.**
Department of Scientific Computing.
Dostupné z https://people.sc.fsu.edu/~jburkardt/f_src/quadpack

[18] *GANDER, W., GAUTSCHI W.:* **Adaptive Quadrature – Revisited.**
BIT. roč. 40. 2000.

[19] *MATHWORKS:* **Matlab.**
Accelerating the pace of engineering and science.
Dostupné z http://www.mathworks.com/products/matlab/

[20] *WOLFRAM :* **Mathematica.**
Computation meet knowledge.
Dostupné z https://reference.wolfram.com/language/tutorial/NIntegrateIntegrationStrategies

[21] *INTEL DEVELOPER ZONE :* **Intel Software Development Products.**
Dostupné z
https://software.intel.com/en-us/qualify-for-free-software/student

[22] *BATES, D. S.:* **Jumps and Stochastic volatility: Exchange Rate Processes Implicity in Deutsche Mark Opitions.**
The Review of Financial Studies. vol. 9. num. 1. 1996. p. 69–107.

[23] *DANĚK, J., POSPÍŠIL, J.:* **Numerical integration of inaccurately evaluated functions.**
In 23rd Annual Conference Proceedings of Technical Computing Prague 2015. Praha 2015. ISBN 978-80-7080-936-5.